Minimizing Current Effects on Autonomous Surface Craft Operations in Singapore Harbor

by

Lynn M. Sarcione

B.S. Mechanical Engineering
Georgia Institute of Technology, 2007

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

June 2010

Author .....
Department of Mechanical Engineering
May 14, 2010

Certified by .....
Franz S. Hover
Doherty Assistant Professor in Ocean Utilization
Thesis Supervisor

Accepted by .....
David E. Hardt
Graduate Officer, Department of Mechanical Engineering

Minimizing Current Effects on Autonomous Surface Craft Operations in Singapore Harbor

by

Lynn M. Sarcione

Submitted to the Department of Mechanical Engineering
on May 14, 2010, in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE IN MECHANICAL ENGINEERING

Abstract

The nation of Singapore is seeking new ways to adapt to its rapid growth. With the help of researchers and staff at the Singapore-MIT Alliance for Research and Technology (SMART), numerous data collection and analysis efforts are underway. One such effort involves the harbor where operations and oceanographic data collection missions are being completed using various sensors and autonomous systems. Utilizing Autonomous Surface Crafts (ASCs) to collect data is one such method and provides a level of robustness unachievable by humans. To improve these operations we seek to address the issue of efficiency in an environment containing surface waves, strong winds, and multiple current shears. In this thesis we present two new heading control algorithms for ASCs under the influence of currents. The first, *Cross-Track Error Minimization*, ensures that the vehicle follows a straight-line trajectory between two waypoints under the effects of surface waves and currents. The second controller, *Time-Optimal*, uses Zermelo's problem and function minimization to find the time-optimal trajectory between two waypoints using a known current field. We further define near-time-optimal paths covering a set of waypoints by defining an asymmetric Traveling Salesman Problem (TSP) where the graph nodes are the waypoints and the edges are the corresponding travel times between the waypoints. Tour construction and local search heuristics are then utilized to build near time-optimal paths. These new paths show a potential time savings of 89% leading to overall mission efficiency.

Thesis Supervisor: Franz S. Hover
Title: Doherty Assistant Professor in Ocean Utilization

Acknowledgments

First, I would like to thank my advisor, Franz Hover, for presenting me with such an amazing opportunity here at MIT. The knowledge I've gained and the experiences I've been given are invaluable. I cannot thank you enough.

I would also like to thank some of the amazing friends I've made for the great memories they've supplied. Brendan, Charlie, Josh, and Kyle, I don't know how I would have survived without all of you and your shenanigans. I am also grateful to all of the folks over in Singapore who have supported my research on numerous occasions: Alfred, Andrew, Anthony, Haining, Hannah, Boon Hooi, Rubaina, Tawfiq and Tirtha.

To my wonderful friends and family, thank you for all the love and encouragement. A big thanks to all of the incredible friends I've made while here in Cambridge. April, Leah, and Jenn, you are wonderful dance partners! Thank you for all the memories made and to be made. And also to my Douglas, you make me smile every day and for that I am so very grateful.

Lastly, I would like to thank my parents. I can most certainly say that I am luckiest daughter in the world. Not only do I have you as inspirational role models, but I have been blessed with your unconditional support with whichever path I choose to take. Thank you so very much.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

As the center of Southeast Asia, Singapore is an evolving nation. It hosts one of the busiest ports in the world [24] and is currently seeking new ways to grow and develop. With the help of the Singapore-MIT Alliance for Research and Technology's (SMART) Center for Environmental Sensing and Monitoring (CENSAM) [18], Singapore aims to utilize its resources in order to enhance its growth. Acquiring data from the marine environment will greatly support urban planning, forecasting, and environmental risk assessment efforts [19].

Autonomous surface craft (ASC) missions are currently underway to obtain information pertaining to seafloor topography, water composition, and surface activity. In the future, CENSAM hopes to utilize a fleet of autonomous systems with the advanced capabilities required for monitoring port operations, evaluating harbor security, and participating in oceanographic surveys. To ensure that these operations are successful and energy-efficient, researchers must take a number of steps. One such step is done by minimizing current effects. Current effects are cumbersome; they can force vehicles away from their desired trajectories and tend to increase mission time. This thesis presents multiple ways to minimize current effects leading to improved mission performance in Singapore harbor. Chapter 2 defines a cross-track error minimization control law which eliminates surface effects and allows the vehicle to follow a straight line. Chapters 3-4 present a procedure for formulating local and global time-optimal paths within known current fields.

Figure 1-1: Map of Singapore
The nation of Singapore is situated in the Singapore Straits between Europe and Asia. Its location has enabled it to be one of the busiest ports in the world making is surrounding marina an important resource.

1.1 Motivation: Monitoring the Singapore Marine Environment

Singapore's future requires that its marine environment be monitored and evaluated continuously. Port operations must remain consistent and potential security threats must be assessed. Environmental changes must be recorded and analyzed for potential risks. Together, these efforts will maintain a safe and valuable harbor.

1.1.1 Port Operations and Harbor Security

Singapore's location is ideal for large port operations, being situated along a major trade route between Europe and Asia. According to the Maritime and Port Authority of Singapore, 472,300 tonnes of cargo went through its port in 2009, making it the busiest in the world for total shipping tonnage [40]. Scheduling, technology, and security issues create a complex system which must be optimized. Coordina-

tion of container transfers and minimization of ship turnaround times are required for effective port operation. Currently, Singapore port personnel use the Computer Integrated Terminal Operations System (CITOS) to proceed with operations planning [24], but improvements supplied by real-time data transfer and utilization of autonomous systems will assist with future growth.

Harbor security is another issue emerging in Singapore. The maritime industry in Singapore contributes about seven percent to its Gross Domestic Product [40] and security threats could severely hamper its contribution [1]. Pirating and weapon threats lead to productivity slowdowns alongside fear. Autonomous systems provide the flexibility and safety necessary to conduct thorough searches of ship exteriors minimizing the time required by port personnel.

Figure 1-2: Photograph of Singapore Harbor
Singapore's port is the busiest in the world for total shipping tonnage [40]. Its operation must be consistent and free of security threats.

1.1.2 Oceanographic Surveys

As Singapore develops, effort is being placed on data collection in the marine environment. Oceanographic surveys provide information on water composition (salinity, turbidity, etc.), seafloor topography, and surface activity. As the country searches

for ways to accommodate its growth through land reclamation and urban sustainability efforts, environmental changes must be monitored and evaluated. Complex autonomous systems will simplify the process by providing real-time data collection and transfer.

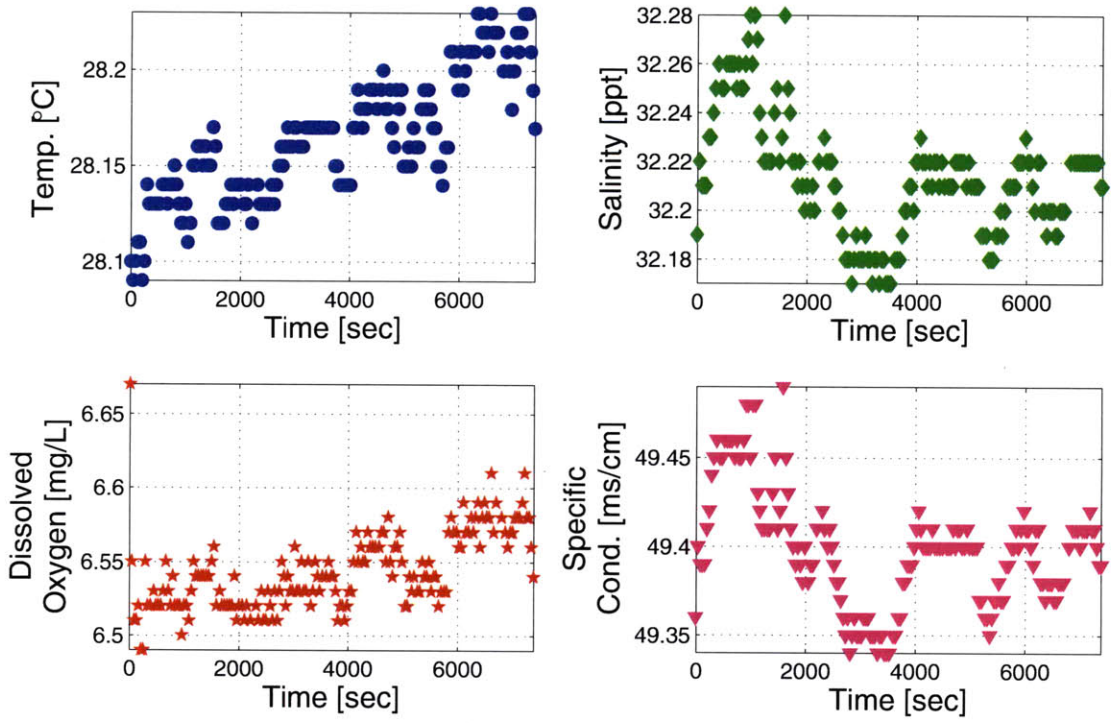


Figure 1-3: Water Composition Data
Water composition data collected on January 15, 2010 in Selat Pauh using an autonomous underwater vehicle (AUV). Oceanographic data acquisition is very important for Singapore's development. It assists in identifying environmental changes and exploring seafloor topography.

1.1.3 Sea Trial Operations

Currently, sea trials are utilized to test theories developed at CENSAM in a real ocean environment. To date, the research staff and students have participated in four, one month-long trials since its founding in 2008. Two main harbor regions around Singapore have been used: the Selat Pauh area, in the south, and the Pulau Ubin area, located in the channel which divides Singapore from Malaysia.

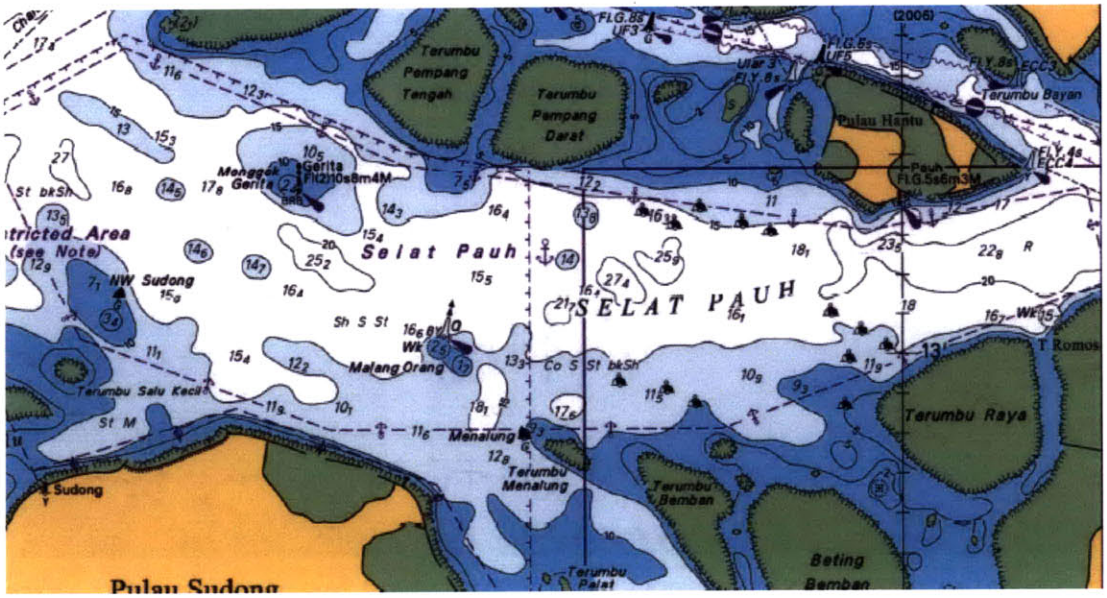
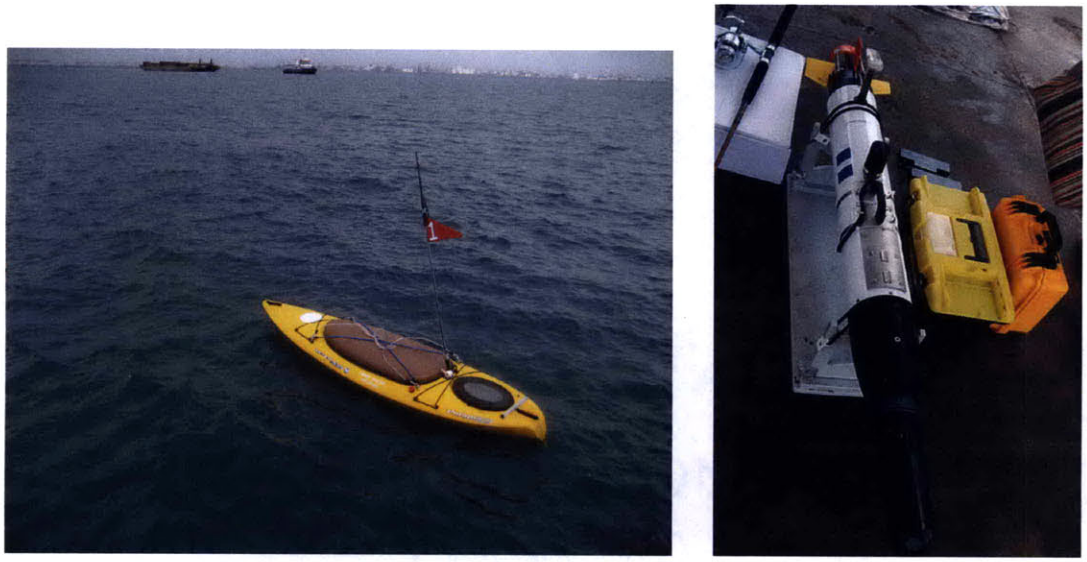


Figure 1-4: Selat Pauh Area of Singapore
Two main areas were utilized for sea trials: Selat Pauh and Pulau Ubin. Selat Pauh, pictured above, is located south of the Singapore mainland. Pulau Ubin is located in the north between Singapore and Malaysia.

The equipment for these trials is quite extensive and allows the team to capture a wealth of data. The current work utilizes a number of sensor platforms, both autonomous and human-operated. Our autonomous systems consist of autonomous surface crafts (ASCs) and autonomous underwater vehicles (AUVs). Navigation is performed using a Global Positioning System (GPS), a Doppler Velocity Log (DVL), and an Inertial Measurement Unit (IMU) with a compass. Communication is achieved using a wireless modem and remote control. The human-operated sensor platforms include numerous ships and a quadrotor. In the past CENSAM has utilized personal fishing vessels, speed boats, and barges.

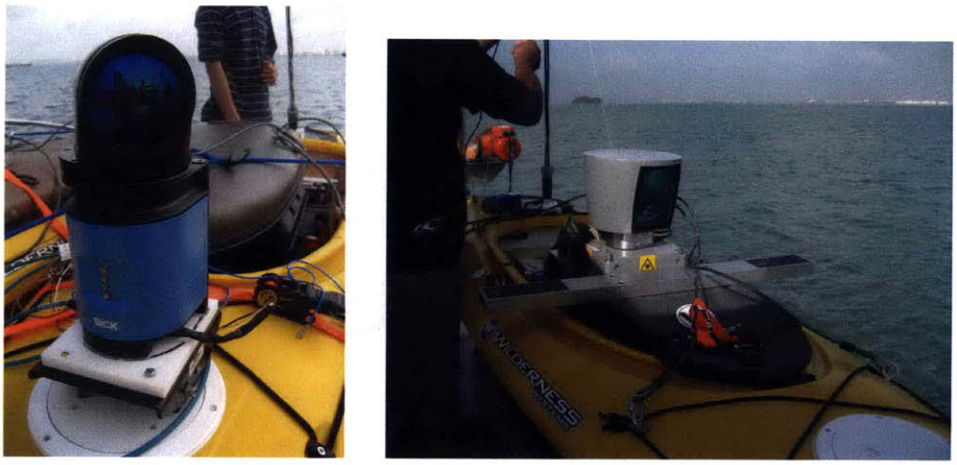
Each platform can be equipped with various sensors for data acquisition. A blazed array 900kHz imaging sonar and a micro-bathymetry blazed array are utilized to obtain information pertaining to the seafloor and subsea structures. Two different lasers are used to obtain surrounding surface data: a 2-D SICK scanning laser and a 3-D Velodyne laser range finder.



(a) Autonomous Surface Craft (ASC)

(b) Autonomous Underwater Vehicle (AUV)

Figure 1-5: Photographs of Autonomous Vehicles Used in Singapore Harbor
Autonomous ocean vehicles are utilized during CENSAM's sea trial missions. These systems can be equipped with various sensors including scanning lasers and blazed-array sonars.



(a) SICK 2-D Scanning laser

(b) Velodyne 3-D Laser Range Finder

Figure 1-6: Photographs of Range Sensors
Lasers are utilized to obtain the surrounding surface data. Applications for the collected data include mapping and obstacle avoidance.

Autonomous Surface Craft (ASC) Operation

The theory presented in this paper pertains to path optimization in the horizontal plane for ASCs, thus further discussion will be provided about their operation.

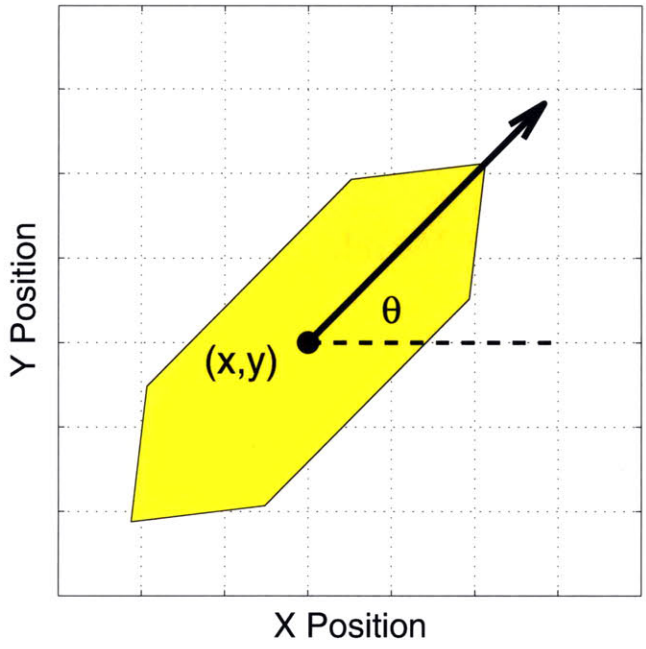


Figure 1-7: ASC Orientation.
The ASC's equations of motion are defined using the current location in x and y and the heading of the vehicle, θ.

The ASCs utilized by CENSAM and built by Robot Marine Systems [57] in their simplest form consist of a kayak hull containing an onboard computer and a rotating thruster. The onboard computer permits autonomous operation through wireless communication. The ASCs are also equipped with a Global Positioning System (GPS) and an Inertial Measurement Unit (IMU) for positioning feedback. Commands sent to the rotating thruster are provided as a thrust percentage and a heading angle.

The software used to control the kayaks is MOOS-IVP (Mission Oriented Operating Suite Interval Programming) [56]. It is comprised of a number of C++ modules which allows the user to operate the kayak using pre-defined or user-written control sequences or behaviors. Pre-defined control behaviors provided by the MOOS-IVP authors utilizing the GPS and IMU feedback information include the following: *Waypoint-to-Waypoint*, *Station-Keeping*, *ASC Follow Behavior*, etc. *Station-Keeping* ensures that the ASC remains at a set GPS location. *ASC Follow Behavior* utilizes

wireless communication to locate another ASC and maintain its position at a set angle relative to the other ASC. *Waypoint-to-Waypoint* behavior pertains to the work of this paper and will be discussed in Section 2.1.

1.2 Relevant Prior Work in Current Minimization and Path Planning

1.2.1 Cross-Track Error Minimization

One of the most useful ASC control laws for oceanographic surveys in Singapore harbor is the *Cross-Track Error Minimization* controller. Traditional marine systems control, such as *Waypoint-to-Waypoint*, drives the vehicle from one point to the next without any constraints on its trajectory. In situations where strong currents are present the trajectory may not be ideal for the given mission due to time or location constraints. Oceanographic surveys can require that the vehicle follow a defined straight-line path in situations which call for complete sensor coverage such as mapping.

Cross-track error minimization falls within the realm of track-keeping or trajectory tracking. For underactuated vehicles traveling in a horizontal plane, track-keeping forces the vehicle to follow a defined reference path by controlling its heading assuming a constant forward velocity. For most instances of underactuated systems, the tracking problem is challenging due to the fact that the systems are not fully feedback linearizable and exhibit nonholonomic constraints.

Trajectory tracking control consists of controlling the heading of the vehicle, θ, so as to minimize the cross-track error in its simplest form [20], [21]. The cross-track error, e, between a desired trajectory and a vehicle is defined as the following,

$$e = -(x - x_d)\sin\theta + (y - y_d)\cos\theta \tag{1.1}$$

where x and y are the coordinates of the vehicle and x_d and y_d define the desired

trajectory. Proportional-Integral-Derivative (PID) control can be used to regulate the error to zero [43]. Other techniques utilize optimal control through linear quadratic Gaussian (LQG) and H_∞ methods [21].

Aguiar and Hespanha [2] present a nonlinear tracking controller for two or three-dimensional spaces derived using integrator backstepping. Their iterative Lyapunov-based technique achieves global stability and exponential convergence of the position tracking error, $\|p - p_d\|$, to an arbitrarily small neighborhood of the origin. This is a strong contribution to the field because it does not limit the trajectory type and can instead be a bounded curve parameterized by time.

Do, *et al.* have contributed many works to the subject of tracking [13], [14], [15], [16]. Their latest work focuses on including the ship's unknown nonlinear damping terms by way of a known compact set allowing for both low- and high-speed applications. The use of the ships dynamic structure, a smooth approximation of the damping terms, and backstepping present a globally robust adaptive controller.

Repoulias and Papadopoulos [48], [49] provide theory which delves into trajectory design. In the derivation the reference trajectory is designed to be feasible based on the vehicle dynamics. It encompasses the desired inertial position and the corresponding velocities. Error dynamics and reference orientations are used in conjunction with design methods such as partial state-feedback linearization, backstepping, and nonlinear damping to force the tracking error the neighborhood of zero.

Although many trajectory tracking methods utilize backstepping and optimal controllers such as LQG [29], this work focuses on using Proportional-Integral (PI) control to minimize the cross-track error. Our efforts center around ASC capabilities and applications in Singapore harbor. The ASC software limits the inputs to desired waypoints rather than desired trajectories. With this limitation we focus on designing a controller capable of tracking straight-line trajectories between waypoints. Curved paths, though out of the scope of this work, are achievable through path discretization. The *Cross-Track Error Minimization* controller developed for these applications in Singapore will be further discussed in Chapter 2.

1.2.2 Time-Optimal Path Planning

Although the *Waypoint-to-Waypoint* and *Cross-Track Error Minimization* controllers are useful in many instances, optimizing mission time is always desirable. Using the current energy supply, the ASCs operating in Singapore at full thrust level are only able to operate for an average of four hours. To maximize the usage capability we derive a *Time-Optimal* control. This control law finds the trajectory which minimizes the time required to move between waypoints under known current fields.

One of the most common solutions involving optimal paths in currents, known as Zermelo's problem, comes from Arthur Bryson and Yu-Chi Ho [9]. First, it is assumed that the current is known and time-invariant. Secondly, the ship's velocity relative to the water is assumed to be constant. From these assumptions a control law can be derived from the system's Hamiltonian using time minimization. As presented, Zermelo's problem remains a boundary value problem with the correct trajectory dependent on choice of initial heading. Since [9] is a key contribution to the work presented in this thesis further discussion will be presented in Section 3.1.

Jardin extends the idea of Zermelo's problem to situations where a parameter change occurs for aircrafts such as perturbations in wind shear [30]. By supplementing the model with additional states, the solver accounts for these parameter perturbations and applies Neighboring Optimal Control (NOC) feedback gains to determine the state augmented trajectory. The NOC method is able to handle both time-varying perturbations and well as bias perturbations and its trajectories are shown to be nearly identical to optimal.

Research by Soulignac, Taillibert, and Rueher utilizes symbolic wavefront expansion to determine the departure time along with the planar time-optimal path in a given time window [52], [53]. Unlike many time-optimal trajectory solvers, this work has the capability of accounting for both spatially and time-varying currents. Current forecasts are discretized by space and time into cells. Each cell is given a cost which is a function of the departure time. Paths are built using the hill-climbing algorithm where each iteration chooses the neighboring cell with minimized cost. The optimal

path is found through minimization over all departure times.

Garau, *et al.* present path planning using the A* search method and delve into the benefits of planning for certain current fields [23]. A* is a best-first search method which uses a heuristic to determine the next step to take [42]. Though many heuristics may be chosen, Garau, et. al. present a simple choice where the heuristic is defined as the time required to reach the goal destination in a virtual current field with constant velocity. The magnitude of the velocity is equal to maximum velocity found in the actual current field. The direction of the velocity is towards the goal destination. This heuristic is successful for a range of current intensities and does not depend on eddy size. It is also noted that path planning is beneficial only in cases where the velocity of the vehicle is comparable to the current speeds.

Several researchers propose using a genetic algorithm (GA) for minimum energy cost path planning [55], [28], [3], [4]. Using a GA for path-planning permits both advanced planning and planning in real-time because of its ability to handle spatial and time variability in the environment. Alvarez *et al.* [4] proposed approach uses GAs in conjunction with dynamic programming on reduced-dimension subspaces. As an initialization, the authors present a population of N individuals which correspond a possible solution paths, generated as random walks. Next, the costs required to complete the possible solutions paths are computed. These costs are found by adding up the energy required to propel the vehicle through the given current field, $\sum W_i$. The energy cost at each segment of the tour is given by the following:

$$W_i = \frac{\rho \int\int\int_{X_{i-1}X_i} \| \mathbf{v}_i(x,y,z) \|^3 \, dxdydz}{c} \tag{1.2}$$

where ρ is a constant that is depends on the vehicle dimensions and water properties, c is the nominal speed of the vehicle, and $\mathbf{v}_i(x,y,z)$ is the velocity of the vehicle along a segment X_{i-1} to X_i. Unfeasible nodes are penalized with extra energy costs. After energy summation, $N/2$ nodes with the lowest energy cost are chosen. From here crossover operations are performed to create path offspring and a small percentage of the paths are mutated ensuring proper convergence. The process is repeated until a

stop criteria has been met. To ensure the discovery of an optimal solution the GA is run several times using different initial conditions. A random immigrants mechanism is also utilized to diversify the population.

Pêtrès *et al.* [45], [44] utilize Fast Marching (FM), a breadth-first search approach, to address underwater environmental issues with path planning such as currents, vehicle kinematics, and real-time constraints. Using trial configurations, FM chooses the one with the lowest distance function estimate, the heuristic, while simultaneously updating the neighbors of the trial configuration. The authors combine FM with A* search to achieve both accuracy and efficiency to create FM*.

A paper written by Kim and Ura [34] utilizes the theory developed by Bryson and Ho [9] to find the minimum-time planar trajectory between two locations in undersea environments among current disturbances. Using the control law developed by [9], Kim and Ura focus their efforts on solving a two-point boundary value problem by finding the initial vehicle heading necessary to proceed to another location in the minimum amount of time. By integrating the vehicle's equations of motion forward till a reference time, t_{ref}, the point of smallest deviation from the destination along the trajectory is obtained. Minimization of this process over all initial headings obtains the optimal solution. The work of Kim and Ura will be further discussed in Section 3.2.

1.2.3 Networks and the Traveling Salesman Problem (TSP)

In order to develop a global time-optimal path among a set of waypoints, we propose utilizing local time-optimal solvers to build a graph, or network, for tour minimization. Graphs are comprised of two finite sets: nodes, $N = \{n_1, n_2, n_3, ..., n_k\}$, and edges, $E = \{e_1, e_2, e_3, ..., e_m\}$ [10]. Edges define relationships between nodes meaning if e_1 connects n_1 to n_4 then $e_1 = n_1 n_4$. Graphs may also be directed. In directed graphs edges become arcs defining the directional relationship between nodes [10]. If arc a_1 connects n_1 to n_2 in the specified direction then $a_1 = (n_1, n_2)$. Both edges and arcs contain costs associated with their traversal.

Tours are essentially paths taken which visit all the desired nodes in a network.

In our case, the nodes are waypoints and we aim to minimize the cost of that tour. Formulating the problem as a combinatorial optimization traveling salesman problem (TSP) provides us with the ability to minimize the cost using a number of techniques. The traditional goal of the TSP is to find the route a salesperson must take to visit a set of cities and return to the starting location so that the total distance traveled is minimized [25]. In this traditional sense the graph contains the cities as nodes and the distances between cities as edges. The distances are said to be same whether you travel from city A to city B or from city B to city A thus creating a symmetric cost matrix.

TSPs are NP-hard and computation time for exact solutions grows exponentially as you acquire more and more nodes [41]. At the cost of accuracy we can acquire speed using a number of heuristic techniques. To proceed using any tour improvement methods we must first utilize classical tour construction techniques. One such technique is the Nearest-Neighbor method of order $O(n^2)$ [41]. Nearest-Neighbor chooses a random node then proceeds to add the next closest (with respect to cost) unvisited node and adds it to the tour. This process continues until all nodes are a part of the tour. We chose to implement Nearest-Neighbor in our work due to the fact that nearly all construction heuristics lead to similar results when used to initiate a local search [25]. However, a number of other construction heuristics exist. The Greedy Algorithm constructs a tour by ordering the node-to-node edges based on their weights. Nearest Insertion starts with an edge and inserts the next non-tour node with the minimum cost into the current tour until all nodes have been added [31]. Farthest Insertion operates in a similar manner to that of Nearest Insertion except those nodes with a maximum weight are added. In 1976, Christofides presented a new construction method which utilizes minimum spanning trees and minimum length matching [11].

Tour improvement heuristics are utilized to acquire near-optimal solutions to the TSP using tours provided through tour construction. These heuristics generate tours at a local optimum. Edge swapping is the basis of many of these with the simplest being 2-Opt. 2-Opt, first presented by Croes [12], iteratively swaps edges and reeval-

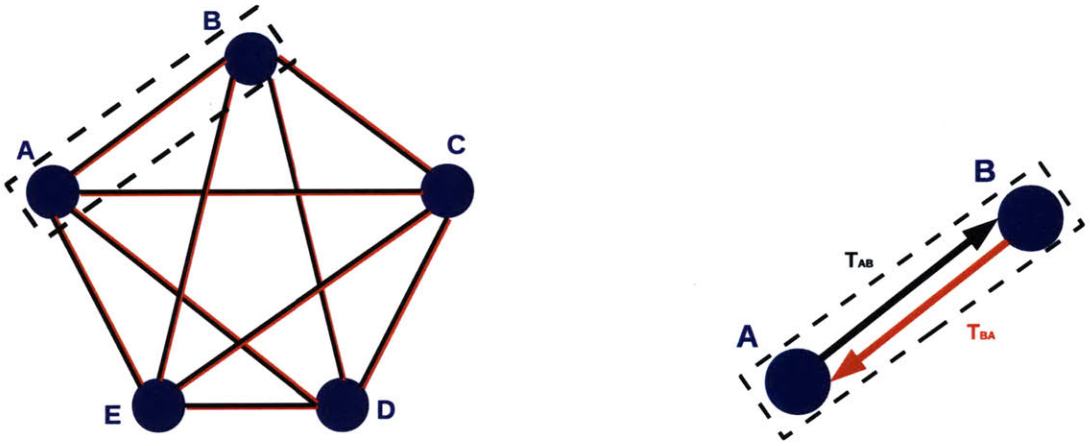
uates the change in tour cost to determine if a better tour has been found. Variants on this extend the idea to 3-Opt and k-Opt [38]. These edge exchanging techniques will be further discussed in Chapter 4.

The Lin-Kernighan heuristic [39], another tour improvement algorithm, is known for its impressive capabilities [31]. This algorithm utilizes k-Opt but does not fix the value of k. Instead it attempts to find the best choice of k by representing each k move as a sequence of 2-Opt exchanges. During a single iteration, a series of 2-Opt moves are run until a stopping requirement is met [26]. The best sequence is then utilized as the k-Opt move. Limited backtracking is used to save computation effort resulting in a local optimal solution. Implementation research has been conducted in this area as well [27].

Of course many exact solvers exist for the TSP but computational requirements are unrealistic for applications in Singapore harbor. The best solver was developed and presented by Applegate *et al.* under the name of Concorde [5], [6]. The program utilizes branch-and-cut-and-price methods which initially relaxes constraints. Throughout the computation these constraints reemerge dynamically. In 2009, Applegate *et al.* was able to compute the optimal solution for a TSP containing 85900 nodes using Concorde, the largest of its kind [7].

The related research presented in the previous paragraphs focused on symmetric TSPs which are represented using undirected graphs. In very rare situations do the networks for this work become symmetric. Our goal is to obtain near-time-optimal paths for ASCs in ocean currents. Because of this we use time, instead of the traditional distance, to construct a directed graph. The graph is directed due to the fact that the time to travel from waypoint A to waypoint B is not necessarily equal to the time to travel from B to A. The travel time is dependent on the current field and the ASC thrust level. This directed graph then becomes an asymmetric TSP. Figure 1-8 describes the problem at hand where $T_{AB} \neq T_{BA}$ due to these current effects.

One approach to dealing with asymmetric TSPs is to use matrix transformation [32]. This method utilizes ghost nodes as place holders within the cost matrix forming a new, symmetric cost matrix. The transformation effectively doubles the size of the



(a) Asymmetric TSP Graph for ASC Path Planning

(b) One Edge of an Asymmetric TSP Graph

Figure 1-8: Asymmetric TSP Graph
Asymmetric TSPs emerge during our research due to the fact that ocean currents greatly influence the time required by an ASC to move from waypoint A to waypoint B and vice versa. These currents lead to a directed network and a corresponding asymmetric TSP.

problem and 2-Opt exchanges usually fail to improve tours making it unrealistic for our purposes [47].

Tour improvement methods for asymmetric TSPs also exist. Traditional 3-Opt utilizes edge exchanges but chooses the one orientation which does not reverse the tour [25]. A variant of the Lin-Kernighan algorithm has also been devised. This heuristic, known as the Kanellakis-Papadimitriou algorithm, utilizes a variable-depth local search [33] to acquire new tours during a single iteration. The k exchange is limited to odd $k \geq 3$ to ensure that the tour is not reversed. A gain for each new tour is evaluated to confirm that it is better than its predecessor.

One method suitable for both symmetric and asymmetric TSPs was developed by Stützle and Hoos [54]. They present the *MAX-MIN* Ant System which builds a tour using pheromone trails τ_{ij} associated with each arc of a TSP. Starting with an initial node, consecutive nodes are added probabilistically according to a probability distribution proportional to Equation (1.3),

$$p_{ij} \sim \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta} \quad \text{if } j \text{ not yet visited, else 0,} \tag{1.3}$$

where $\eta_{ij} = 1/d_{ij}$ and d_{ij} is the cost associated with the the arc connecting node i with node j. Each ant constructs a tour using α and β as the probability distribution bias parameters and trail intensities are computed and updated based on the the result of the best ant. The trail strengths are limited between maximum and minimum values and in situations of stagnation intensities undergo a proportional update. The Ant System is combined with local search techniques to obtain near-optimal solutions.

Genetic local search algorithms are also used to solve symmetric and asymmetric TSPs [22]. Local search techniques find local optimal solutions. The GAs then search the local optima to find the global optimum efficiently. When applied to asymmetric algorithms local search is completed using a *fast-3-Opt* algorithm [8] whereas Lin-Kernighan is applied for symmetric cases.

Since the contribution of our work focuses on presenting the novel idea of solving for a near-time-optimal solution by modeling the problem as a TSP we do not focus our efforts on improving current TSP solvers. As this is the case the research presented utilizes one construction method and three modified local search methods. These methods will be further discussed and analyzed in Chapter 4.

1.3 Problem Statement: Minimization of Ocean Current Effects

The objective of this work is to improve current research operations in Singapore harbor. One way to achieve this is to minimize ocean current effects. By constructing a *Cross-Track Error Minimization* control law, the ASC is able to follow a straight-line trajectory. This control law counteracts the external forces consisting of ocean surface waves and currents. Another improvement introduced in Singapore is ASC path-planning for known currents. Using local time-optimal paths found using *Time-Optimal* control, we are able to build a network defining the paths between mission

waypoints. An optimal global tour of the network is then found using TSP techniques.

Figure 1-9 presents the structure of this thesis. Chapter 2 will discuss the methodology for developing and tuning the *Cross-Track Error Minimization* controller as well as present and discuss its improvements over the traditional *Waypoint-to-Waypoint* controller. Chapter 3 will present the theory behind Bryson and Ho's local time-optimal path work [9] and show how this boundary-value problem can be solved for two waypoints. Chapter 4 will continue the work of Bryson and Ho by using local time-optimal paths to construct and define TSP networks. TSP tour construction and local search techniques will be presented and analyzed. Throughout the thesis, experimental and simulation data will be presented and discussed.

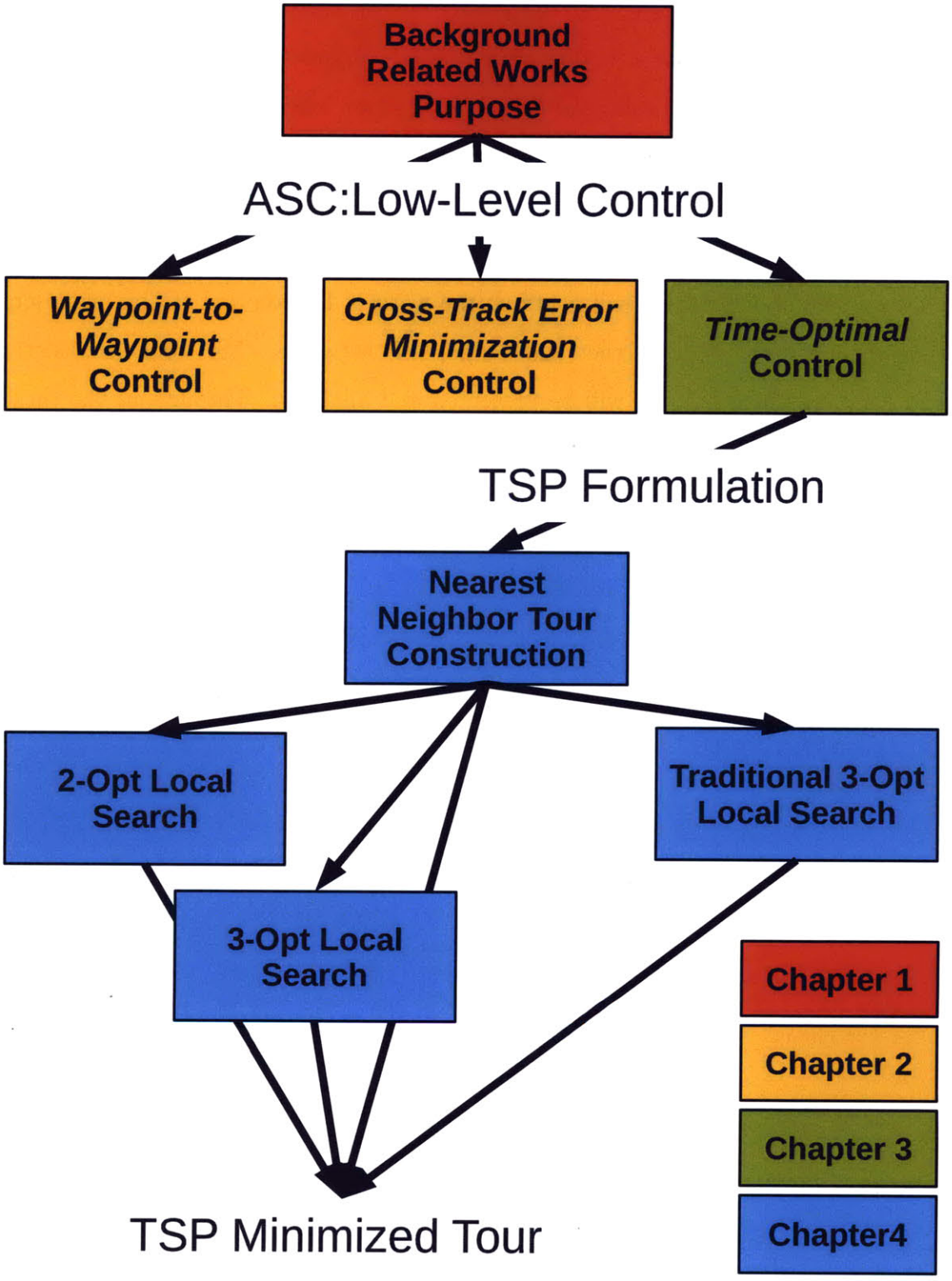


Figure 1-9: Thesis Structure

Chapter 2

Low-Level Autonomous Surface Craft Control

As mentioned previously, ASC operations in Singapore harbor are used for the purpose of data collection and harbor exploration. Current control laws are capable of conducting basic waypoint surveys and following behavior. By adding the *Cross-Track Error Minimization* control law, ASCs are able to follow straight-line trajectories with minimal error which greatly benefits missions involving the mapping and oceanography communities.

2.1 *Waypoint-to-Waypoint* Control

Currently, a few basic control laws constructed using MOOS-IVP [56] are present on the ASCs. One such low-level control law is *Waypoint-to-Waypoint* control. Earlier sea trial operations used this control law exclusively because research was focused on visiting specified waypoints for data collection.

2.1.1 *Waypoint-to-Waypoint* Heading Control Algorithm

The *Waypoint-to-Waypoint* controller is simply constructed. The basic algorithm is presented in Algorithm 2.1. This control law requires that two locations must be

known: the current position of the ASC and the destination waypoint. Line 1 in Algorithm 2.1 demonstrates the simplicity of the controller. Essentially the ASC is asked to proceed in the direction of destination waypoint, *destwpx* and *destwpy*. The underlying ASC controller prevents the ASC from overshooting and saturates the heading inputs.

Algorithm 2.1 *Waypoint-to-Waypoint* Heading Control

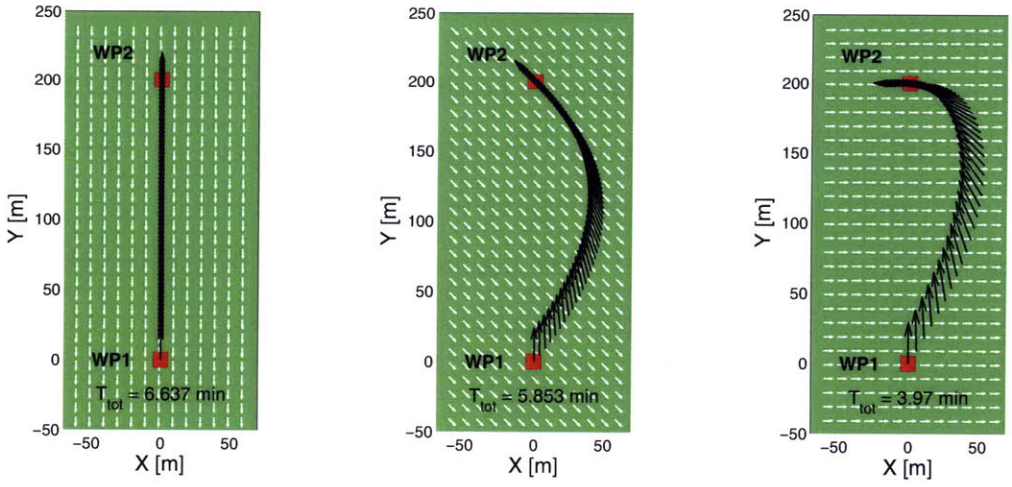
headcmd = wp2wp $(x, y, destwpx, destwpy)$ where x and y are the current coordinates of the ASC and *destwpx* and *destwpy* are the coordinates of the destination waypoint

1: $headcmd = \text{atan2}(destwpy - y, destwpx - x)$
2: **return** *headcmd*

2.1.2 *Waypoint-to-Waypoint* Heading Control Simulation

The main use of *Waypoint-to-Waypoint* control is to visit waypoints in a manner that does not take trajectories or currents into consideration. Figure 2-1 demonstrates this fact by depicting five cases of directionally-varying currents. In each case, the magnitude of the current is the same at $V_{curr} = \frac{2}{3}V$ where V is the constant velocity of the ASC over water equatable to thrust. Here we can see that the ASCs have straight-line trajectories using *Waypoint-to-Waypoint* control when the current is aligned with the waypoints themselves (Figures 2-1(a) and 2-1(e)). Otherwise a strong deviation from the line is observed (Figures 2-1(b), 2-1(c), and 2-1(d)). The time necessary to proceed from one waypoint to the other is also dependent on the current direction as shown in the figures.

Although *Waypoint-to-Waypoint* heading control is useful for a number of data collection missions, it does not encompass all. For certain missions such as structure surveys, ASCs must be able to follow a specified trajectory with minimal error. This is not possible through *Waypoint-to-Waypoint* control unless the currents are aligned with the direction of motion, a highly unlikely scenario. Because of this, a new control law must be developed to enhance the capabilities of the ASC. *Cross-Track Error Minimization* control presented in Section 2.2 presents this new controller and



(a) $\theta_{curr} = 270°$. (b) $\theta_{curr} = 315°$. (c) $\theta_{curr} = 0°$.

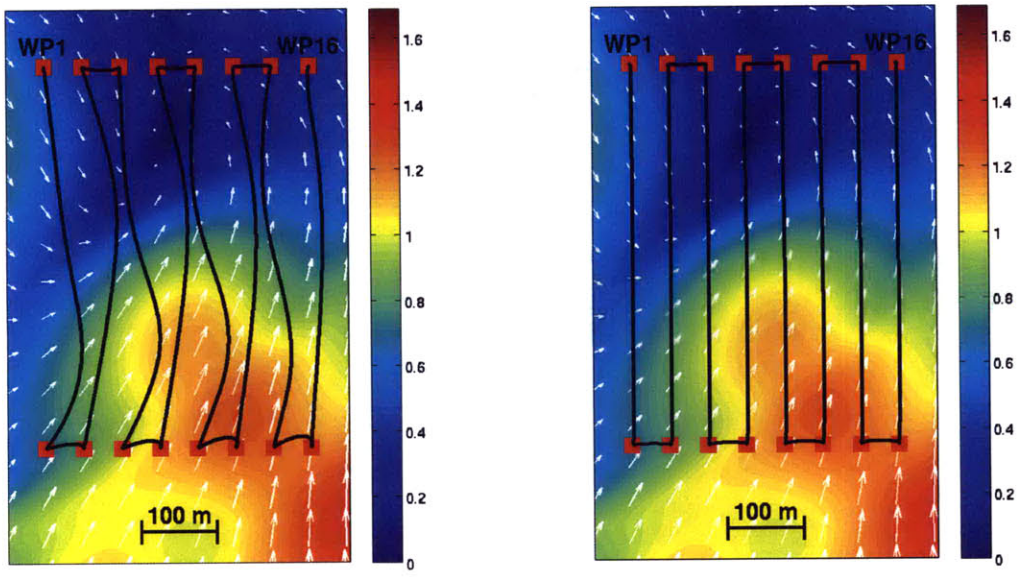
(d) $\theta_{curr} = 45°$. (e) $\theta_{curr} = 90°$.

Figure 2-1: *Waypoint-to-Waypoint* control for Directionally-Varying Currents
The direction of the current field can vary the response of the *Waypoint-to-Waypoint* control. Here we have five cases in which the magnitude of the current remains constant ($V_{curr} = \frac{2}{3}V$) but the direction changes. As is shown, an opposing current increases the time required to move from one waypoint to the other. Direct routes are only achievable when the current direction is aligned with the waypoints themselves.

demonstrates its straight-line trajectory capabilities.

2.2 *Cross-Track Error Minimization* Control

The goal of *Cross-Track Error Minimization* heading control is to ensure that the ASC follows a specified straight-line trajectory. Straight-line trajectories are necessary for certain oceanographic surveys and subsea exploration. Straight-line trajectories can also be used to discretized parabolic paths making the *Cross-Track Error Minimization* controller robust and adaptable to many research missions.



(a) *Waypoint-to-Waypoint* (b) *Cross-Track Error Minimization*

Figure 2-2: Low-Level Control for Oceanographic Surveys
Choice of low-level control is dependent on the mission. If the goal is to follow several straight-line trajectories throughout a specified area *Cross-Track Error Minimization* control is ideal.

2.2.1 *Cross-Track Error Minimization* Control Algorithm

The method utilized to implement the *Cross-Track Error Minimization* control is presented in Algorithm 2.2. The controller is PI (proportional-integral) where the proportional and integral terms are related to the distance from the line the ASC is attempting to follow between waypoints. Their gains, K_p and K_i respectively, are

tuned to the specific vehicle as shown in Line 1.

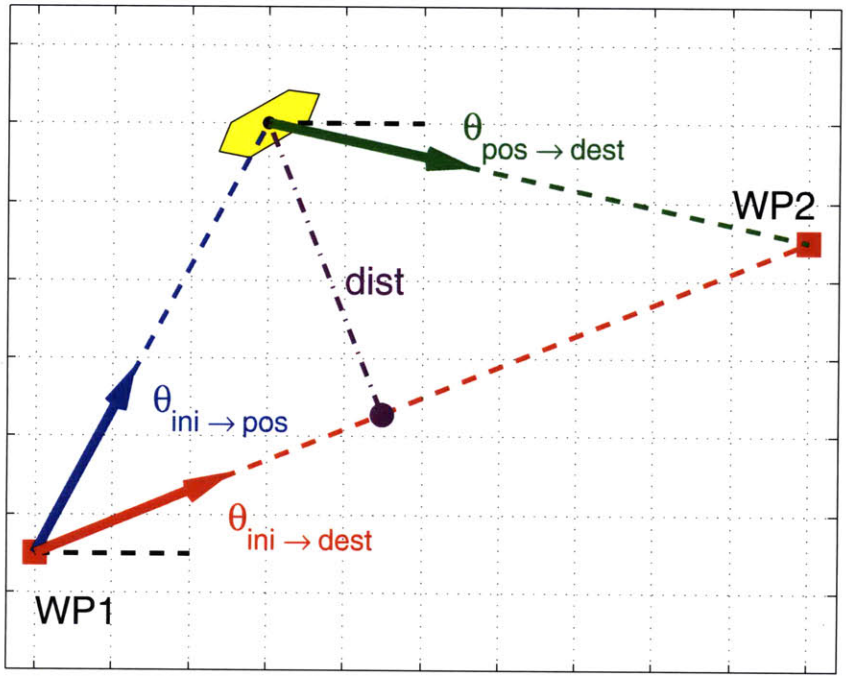


Figure 2-3: *Cross-Track Error Minimization* Heading Control Angles
Cross-Track Error Minimization control (Algorithm 2.2) utilizes three main angles: $\theta_{ini \Rightarrow dest}$ $(wayang)$, $\theta_{pos \Rightarrow dest}$ $(ang2way)$, and $\theta_{ini \Rightarrow pos}$ $(ascang)$. PI control is used to relate the *dist* variable to the desired heading.

To begin, the algorithm finds the relative angles between all important positions in the plane: $\theta_{ini \rightarrow dest}$, the angle from the initial waypoint to the destination waypoint, $\theta_{pos \rightarrow dest}$, the angle from the current position of the ASC to the destination waypoint, and $\theta_{ini \rightarrow pos}$, the angle from the initial waypoint to the current ASC position. Within the algorithm, these variables are set to be *wayang*, *ang2way*, and *ascang* respectively and are presented in Lines 4-6. Figure 2-3 further clarifies these important angles. Using *wayang* we must then find the equation for the straight-line which connects the initial and destination waypoints, our reference path, and the corresponding closest point along that line to our current position, *closestx* and *closesty*. This is completed in Lines 7-15. Using these points, we can compute the ASC's distance from the line, *dist* (Line 16), the basis for our PI control. The proportional control contribution (Lines 17-24) is based on the relative angle between *ascang* and *wayang* to ensure

Algorithm 2.2 *Cross-Track Error Minimization* Heading Control

$[headcmd, intsumout]$ = cxtrack $(x, y, iniwpx, inixpy, destwpx, destwpy, intsum)$ where x and y are the current coordinates of the ASC, *iniwpx* and *iniwpy* are the coordinates of the initial waypoint, *destwpx* and *destwpy* are the coordinates of the destination waypoint, and *intsum* is the sum of the integral error

1: K_p AND K_i ARE THE PROPORTIONAL AND INTEGRAL GAINS RESPECTIVELY AND ARE TUNED TO THE VEHICLE IN USE THROUGH EXPERIMENTAL TESTING.
2: $wayxdif = destwpx - iniwpx$
3: $wayydif = destwpy - iniwpy$
4: $wayang = \text{atan2}(wayydif, wayxdif)$ {INITIAL TO DESTINATION WAYPOINT ANGLE}
5: $ang2way = \text{atan2}(destwpy - y, destwpx - x)$ {POSITION TO DESTINATION WAYPOINT ANGLE}
6: $ascang = \text{atan2}(y - iniwpy, x - iniwpx)$ {INITIAL WAYPOINT TO POSITION ANGLE}
7: **if** $\text{abs}(wayydif) < 0$ **then**
8: $\quad closestx = iniwpx$
9: $\quad closesty = y$
10: **else**
11: $\quad m = wayydif/wayxdif$
12: $\quad b = iniwpy - m * iniwpx$
13: $\quad closestx = (my + x - mb)/(m^2 + 1)$
14: $\quad closesty = (m^2y + mx + b)/(m^2 + 1)$
15: **end if**
16: $dist = \text{sqrt}[(closestx - x)^2 + (closesty - y)^2]$ {DISTANCE FROM THE LINE}
17: $sinrelang = \sin(ascang - wayang)$ {PROPORTIONAL CONTROL CONTRIBUTION}
18: $possin = 1$
19: **if** $sinrelang < 0$ **then**
20: $\quad possin = 0$
21: **end if**
22: **if** $possin == 1$ **then**
23: $\quad dist = -dist$
24: **end if**
25: $intsum = intsum + dist$ {INTEGRAL CONTROL CONTRIBUTION}
26: $sumlimit = .25/sumin$
27: **if** $intsum > sumlimit$ **then**
28: $\quad intsum = sumlimit$
29: **end if**
30: **if** $intsum < -sumlimit$ **then**
31: $\quad intsum = -sumlimit$
32: **end if**
33: $headcmd = ang2way + K_p dist + K_i intsum$
34: $intsumout = intsum$
35: **return** $headcmd, intsumout$

that the heading of the ASC turns in the appropriate direction. The integral control contribution (Lines 25-32) integrates the distance and limits its contribution using *sumlimit*. The new heading command, $headcmd = ang2way + K_p dist + K_i intsum$ (Line 33), and the integrated sum, *intsumout*, are returned.

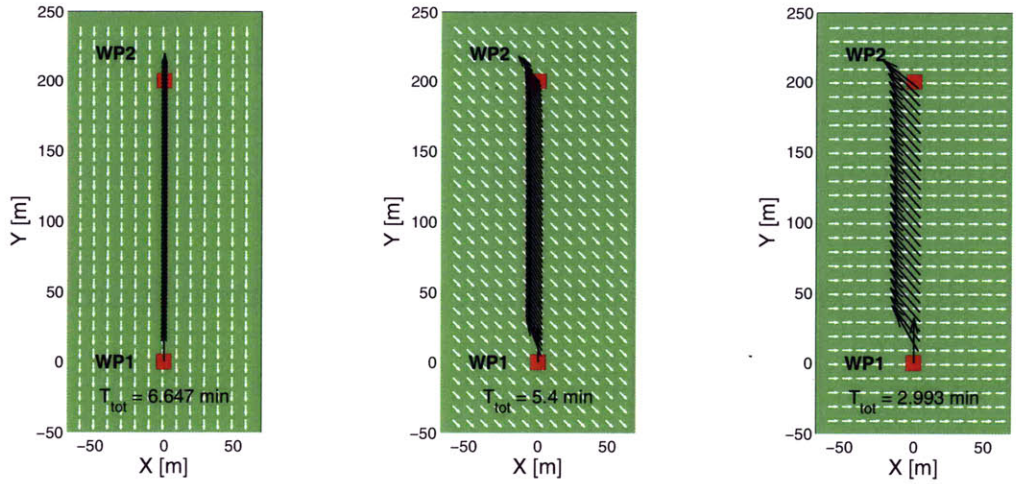
2.2.2 *Cross-Track Error Minimization* Control Simulation

As a comparison to *Waypoint-to-Waypoint* control (Figure 2-1), we chose to demonstrate how the *Cross-Track Error Minimization* controller works in similar cases as shown in Figure 2-4. As the current changes direction, the *Cross-Track Error Minimization* controller adjusts the heading of the ASC so as to be able to follow the straight-line path connecting the two waypoints. A straight-line path no longer depends on the direction of the current, but instead on a properly tuned controller.

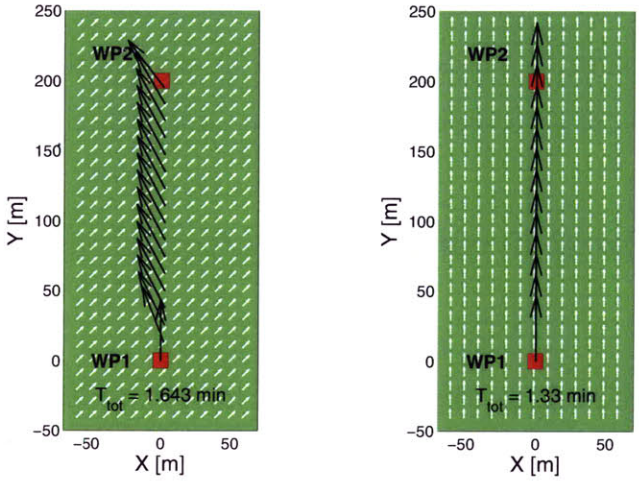
2.2.3 *Cross-Track Error Minimization* Control Experimentation

In order to successfully utilize the *Cross-Track Error Minimization* controller for research operations in Singapore harbor it must first be tuned to the vehicle in use. In January 2010 we had the opportunity complete the tuning in Singapore harbor on one of the four ASCs. Over the course of one week both the K_p and K_i values were adjusted and tested to reveal the ideal gains. Experimentation consisted of running the ASC at 100% thrust resulting in an average vehicle speed over water of 1.7 m/sec. Presented in Figure 2-5 are three sets of gains which were tested. The left-hand column displays the trajectories of the ASC whereas the right-hand column shows the distance, or *dist*, value over the course of the test. The goal is to minimize the distance over time. Based on the runs completed, the best gains for this specific ASC resulted in $K_p = 0.04$ rad/m and $K_i = 0.001$ rad/(m-sec).

The tuning of the vehicle in use is of great importance. With every additional sensor or weight the dynamics of the vehicle change. Future extensions of this work may look to create an adaptable tuner for each vehicle. This will help to minimize



(a) $\theta_{curr} = 270°$. (b) $\theta_{curr} = 315°$. (c) $\theta_{curr} = 0°$.

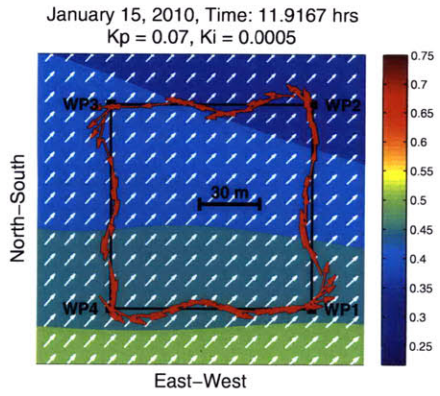


(d) $\theta_{curr} = 45°$. (e) $\theta_{curr} = 90°$.

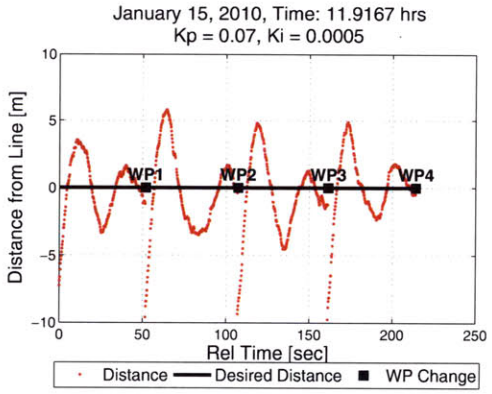
Figure 2-4: *Cross-Track Error Minimization* Control for Directionally-Varying Currents

Here we have five cases in which the magnitude of the current remains constant ($V_{curr} = \frac{2}{3}V$), but the direction changes similarly to Figure 2-1. *Cross-Track Error Minimization* control forces the ASC to follow a straight-line trajectory between the two waypoints.

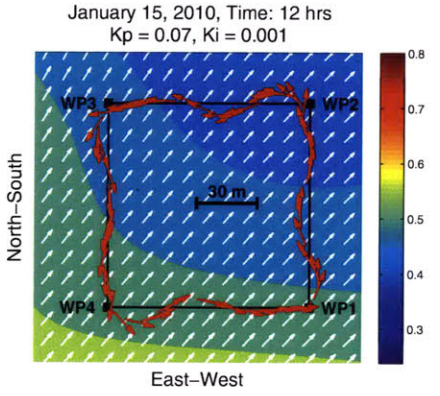
the preparation time required to complete the mission.



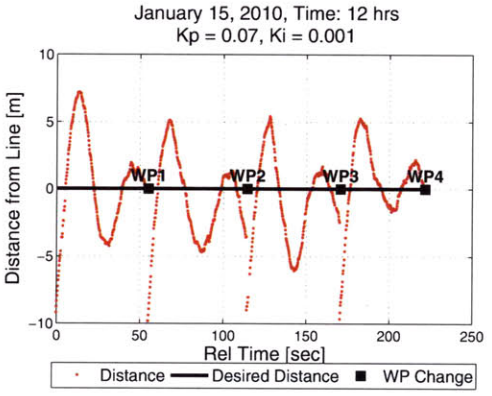
(a) Run 1 Navigation



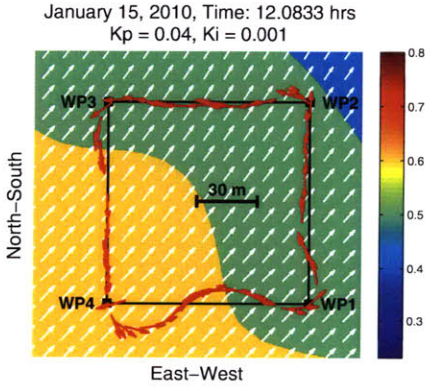
(b) Run 1 Distance



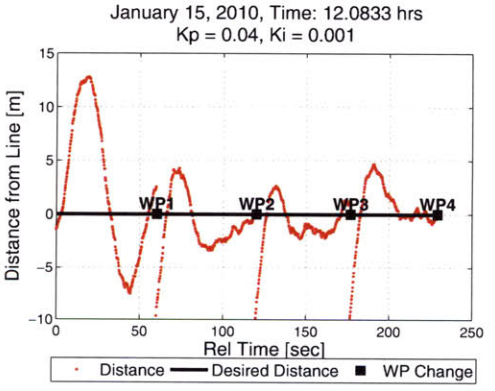
(c) Run 2 Navigation



(d) Run 2 Distance



(e) Run 3 Navigation



(f) Run 3 Distance

Figure 2-5: *Cross-Track Error Minimization* Controller Tuning in Singapore Harbor The gains, K_p and K_i, are very important to the success of the controller and must be tuned to the vehicle in use. Shown here are three of many tuning runs completed in Singapore Harbor on January 15, 2010. Based on experimental results, $K_p = 0.04$ rad/m and $K_i = 0.001$ were found to be the best gains for the ASC used. rad/(m-sec).

Chapter 3

Time-Optimal Local Path and Control

Chapter 2 presented two heading control algorithms. The first, *Waypoint-to-Waypoint* control, centered on surveying specific waypoints and did not consider the trajectory between waypoints. The second, *Cross-Track Error Minimization* control, emphasized the trajectories between the waypoints by forcing the ASC to follow a straight-line between the two. Neither of these previously mentioned control laws account for the time required to proceed between the two waypoints.

Efficiency is of great importance to research operations in Singapore harbor. ASCs run on a limited battery supply resulting in 4 hours of operation for 100% thrust and because of this, surveying an entire harbor with such little efficiency must be addressed. A simple way of improving mission performance is to implement a controller which minimizes the time required to move between waypoints, or *Time-Optimal* control. Using the work of Bryson and Ho [9] we are able to search for and define the time-minimal path using known current fields, effectively reducing the energy required by the ASC during operation.

3.1 Zermelo's Problem

The work of Bryson and Ho [9] introduces us to Zermelo's problem, the idea that a minimum-time path can be found for an ocean vehicle using the Hamiltonian and time minimization. The theory requires two assumptions. First, the magnitude and the direction of the current must be time-invariant and known functions of position, $u(x, y)$ and $v(x, y)$. The variables u and v indicate the velocity components of the current in the $x-$ and $y-$ directions. Secondly, the magnitude of the ship's velocity relative to the water, V, is constant. Using these two assumptions, two of the three equations of motion can be defined as the following:

$$\dot{x} = V\cos\theta + u(x, y) \tag{3.1}$$

$$\dot{y} = V\sin\theta + v(x, y) \tag{3.2}$$

where θ is the heading angle and x and y define the position of the ocean vehicle. Through computation and minimization of the Hamiltonian, $H = \lambda_x(V\cos\theta + u) + \lambda_y(V\sin\theta + v) + 1$, over time the final equation of motion is found [9]:

$$\dot{\theta} = \sin^2\theta\frac{\partial v}{\partial x} + \sin\theta\cos\theta\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) - \cos^2\theta\frac{\partial u}{\partial y} \tag{3.3}$$

Equations (3.1)-(3.3) can be solved simultaneously to obtain the minimum-time path where Equation (3.3) is the specific control law. However, in order to obtain the correct path between two waypoints we must determine the correct initial heading, θ_0, for the vehicle making Zermelo's solution a boundary-value problem.

3.2 Solving Zermelo's Boundary-Value Problem

As mentioned previously, Zermelo's time-optimal control law presented in Equation (3.3) is a boundary-value problem. In order to use it effectively we must determine the correct initial heading, θ_0. Figure 3-2 displays the importance of utilizing the correct initial heading. If it is not chosen properly, the vehicle will not proceed to the

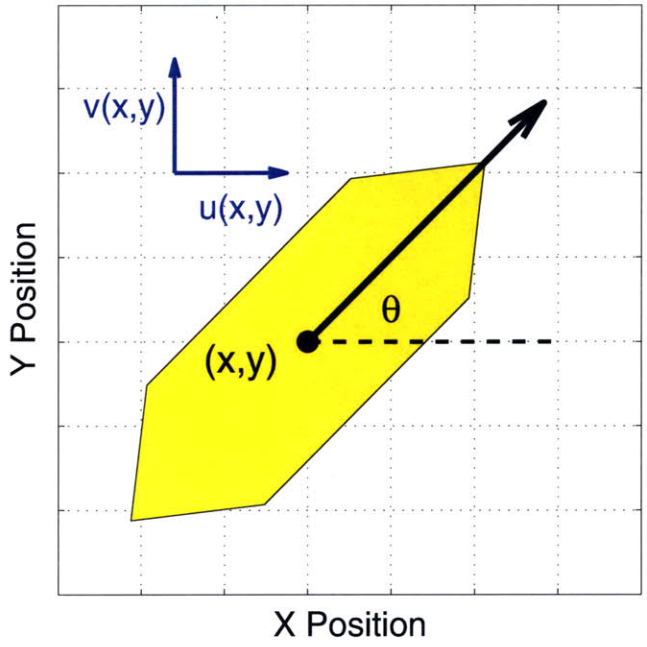


Figure 3-1: ASC Orientation with Current
The solution to Zermelo's problem requires knowledge of the ASC's position, x and y, and heading θ as well as the current's velocity components in the $x-$ and $y-$ directions

destination waypoint.

To solve Zermelo's boundary-value problem we use function minimization. We know that the vehicle must pass through the destination waypoint and that it must be time-optimal. Thus, we must find the correct initial heading which gets the vehicle to the destination waypoint with the smallest deviation in the least amount of time.

Similarly to Kim and Ura [34], we start by using reference navigation to determine an upper bound on the time. The reference navigation can be either *Waypoint-to-Waypoint*, *Cross-Track Error Minimization*, or any other non-time-optimal controller which ensures arrival at the destination waypoint. Using this reference we can compute t_{ref}, the time required to travel from the initial waypoint to the final waypoint using the reference navigation. Because the reference navigation is non-time-optimal we know the following:

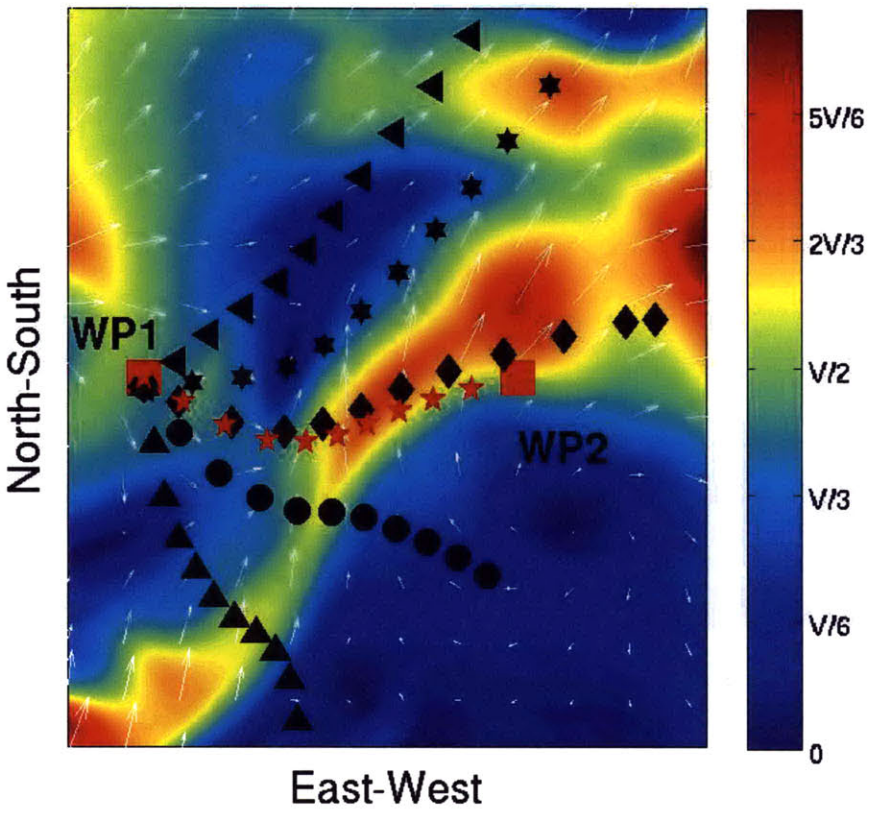


Figure 3-2: Zermelo's Solution for Multiple Initial Headings
Solving Zermelo's boundary-value problem determines the path the *Time-Optimal* control will result in. In this example six initial headings were provided to the controller at *WP1* for a given current field. The magnitude of the current is a function of the vehicle's velocity over the water V. The black trajectories depict the vehicle's controller response with an incorrect θ_0. The red trajectory depicts the correct solution to the boundary-value problem, driving the vehicle to the destination waypoint, *WP2*, successfully.

$$0 < t^* \leq t_{ref}, \tag{3.4}$$

where t^* is the time to travel between the two waypoints using *Time-Optimal* control and the correct θ_0. This upper bound, t_{ref}, allows us to utilize the minimum principle to ensure that any θ_0 which arrives at the destination in a total time less than t_{ref} could be a near-optimal trial trajectory [34].

Using the reference time, we must find the initial headings which drive the vehicle to the destination waypoint. The approach we take is the following: For each initial heading, simulate the vehicle's *Time-Optimal* control trajectory $t = 0$ to t_{ref}. Then, compute the minimum achievable distance to the destination waypoint over the entire trajectory. If the minimum achievable distance is close to zero, this indicates that the vehicle successfully arrives at the destination waypoint during the time interval. The process is summarized within function *findmindist*, shown in Algorithm 3.1. Figure 3-3 depicts the output of function *findmindist* for the initial headings presented in Figure 3-2. Using function minimization to minimize *findmindist* outputs over all initial headings, we can determine the correct *Time-Optimal* control initial heading, θ_0^*, and the corresponding trajectory time, t^*.

Algorithm 3.1 *findmindist* Minimized Function

mindist = findmindist $(t_{ref}, \theta_0, iniwpx, iniwpy, destwpx, destwpy)$ where t_{ref} is the time required to travel between the waypoints using a non-time-optimal controller, θ_0 is the initial heading of the ASC, *iniwpx* and *iniwpy* are the coordinates of the initial waypoint, and *destwpx* and *destwpy* are the coordinates of the destination waypoint

1: Set the initial position to *iniwpx* and *iniwpy*.
2: **while** $t \leq t_{tot}$ **do**
3: Integrate the time-optimal equations of motion, Equations (3.1)-(3.3), forward.
4: Compute the distance to *destwpx* and *destwpy*.
5: **end while**
6: **return** The minimum distance to *destwpx* and *destwpy*, *mindist*.

3.3 *Time-Optimal* Control Analysis

In order to successfully implement *Time-Optimal* control in a Singapore harbor mission we must analyze its impact and computation costs.

3.3.1 Low-Level Controller Comparison

As mentioned previously, we developed this controller to enhance what *Waypoint-to-Waypoint* and *Cross-Track Error Minimization* control do not: efficiency. For

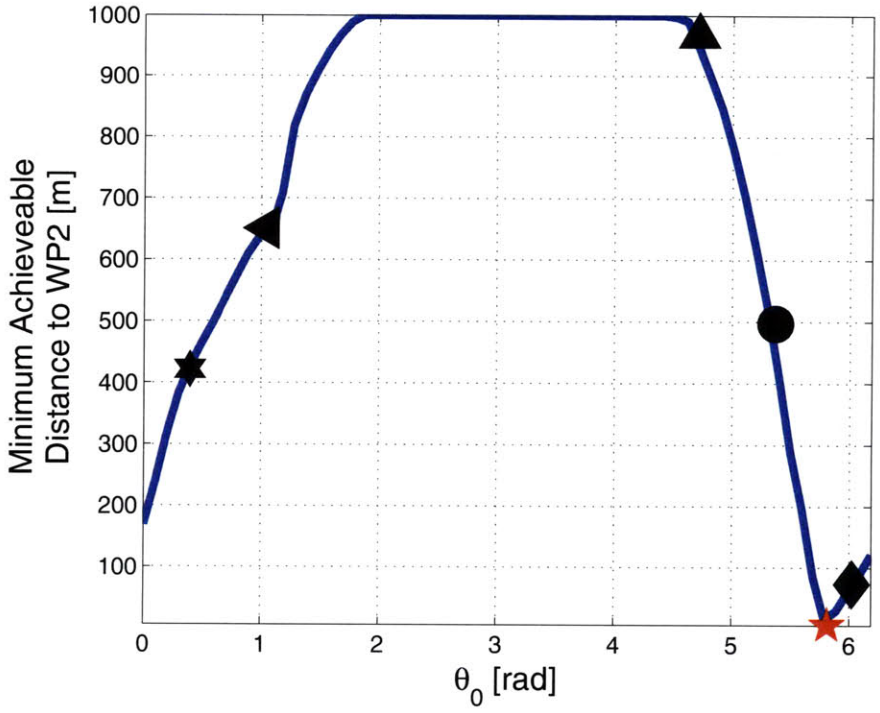


Figure 3-3: *findmindist* Outputs for the Example Posed in Figure 3-2
The function *findmindist* is utilized to determine which initial headings pass through the destination waypoint by computing the minimum achievable distance to that waypoint during $t = 0$ to t_{ref}. The output of *findmindist* is then minimized over all possible initial headings to determine the correct *Time-Optimal* control initial heading, θ_0^*, and the corresponding trajectory time, t^*. Using the example posed in Figure 3-2, we show the output of *findmindist*. The red star indicates the correct *Time-Optimal* control initial heading.

situations which do not put a constraint on the path taken from one waypoint to the next, implementing *Time-Optimal* control will allow us to save time and energy.

In order to compare time savings for each of the controllers, we must define a metric:

$$\text{Relative Time Savings} = \frac{t_{ref} - t^*}{t_{ref}} \times 100\%, \tag{3.5}$$

where t_{ref} is the total trajectory time of the non-time-optimal controller and t^* is the trajectory time of the *Time-Optimal* controller.

Time-Optimal Control Simulation

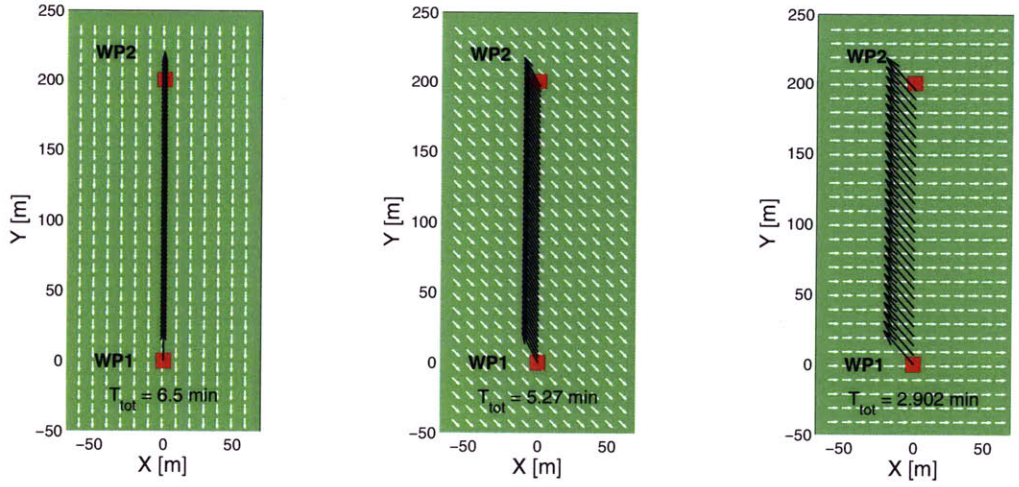
In a similar simulation to those performed in Chapter 2 for the other low-level controllers (*Waypoint-to-Waypoint* - Section 2.1.2, *Cross-Track Error Minimization* - Section 2.2.2), Figure 3-4 presents results of *Time-Optimal* control for an ASC. Similarly to *Cross-Track Error Minimization* control (Figure 2-4), *Time-Optimal* control follows a straight-line trajectory for each of the five current directions. The main difference between the two controllers is the initial heading, θ_0^*, where the *Time-Optimal* trajectory traversal is faster. The Relative Time Savings using *Time-Optimal* control is shown in Table 3.1. *Time-Optimal* control shows a significant improvement over that of *Waypoint-to-Waypoint* control during instances of cross-currents. Although less significant, the savings over *Cross-Track Error Minimization* will prove to be important for longer missions and more spatially-varying current fields.

Table 3.1: Relative Time Savings for Low-Level Controller Simulations
Based on Equation (3.5) we can compute the relative time savings for *Time-Optimal* control compared to those of *Waypoint-to-Waypoint* and *Cross-Track Error Minimization* control for the simulation results presented in Sections 2.1.2, 2.2.2, and 3.3.1.

Relative Time Savings [%]		
θ_{curr}	*Waypoint-to-Waypoint*	*Cross-Track Error Minimization*
270°	2.06	2.21
315°	9.96	2.41
0°	26.9	3.04
45°	23.5	2.80
90°	2.26	2.26

1 km Simulation Comparison

To re-emphasize the impact of *Time-Optimal* control, we chose to simulate the three low-level controllers over a 1 km dist using current forecasts provided by the Tropical Marine Science Institute in Singapore [58] for June 12, 2010. We assume a constant ASC velocity, $V = 1.5$ m/sec. We also assume that the current does not change over time. The resulting trajectories and travel times are shown in Figure 3-5. Using the



(a) $\theta_{curr} = 270°$. (b) $\theta_{curr} = 315°$. (c) $\theta_{curr} = 0°$.

(d) $\theta_{curr} = 45°$. (e) $\theta_{curr} = 90°$.

Figure 3-4: *Time-Optimal* Heading Control for Directionally-Varying Currents Here we have five cases in which the magnitude of the current remains constant ($V_{curr} = \frac{2}{3}V$) similarly to those presented for *Waypoint-to-Waypoint* control and *Cross-Track Error Minimization* control in Sections 2.1.2 and 2.2.2 respectively. Although the trajectory appears similar to those of the *Cross-Track Error Minimization* control the initial headings allow it save mission time.

time savings metric provided in Equation (3.5) we find that *Time-Optimal* control saved 18.37% time over *Waypoint-to-Waypoint* control. Additionally, we find that *Time-Optimal* control saved 7.73% time over *Cross-Track Error Minimization* control. In addition to these savings, it is important to note that the time-minimal trajectory does not follow that of the *Cross-Track Error Minimization* as it did in Section 3.3.1. Instead it follows a route which manipulates both the magnitude and direction of the

current to permit time savings.

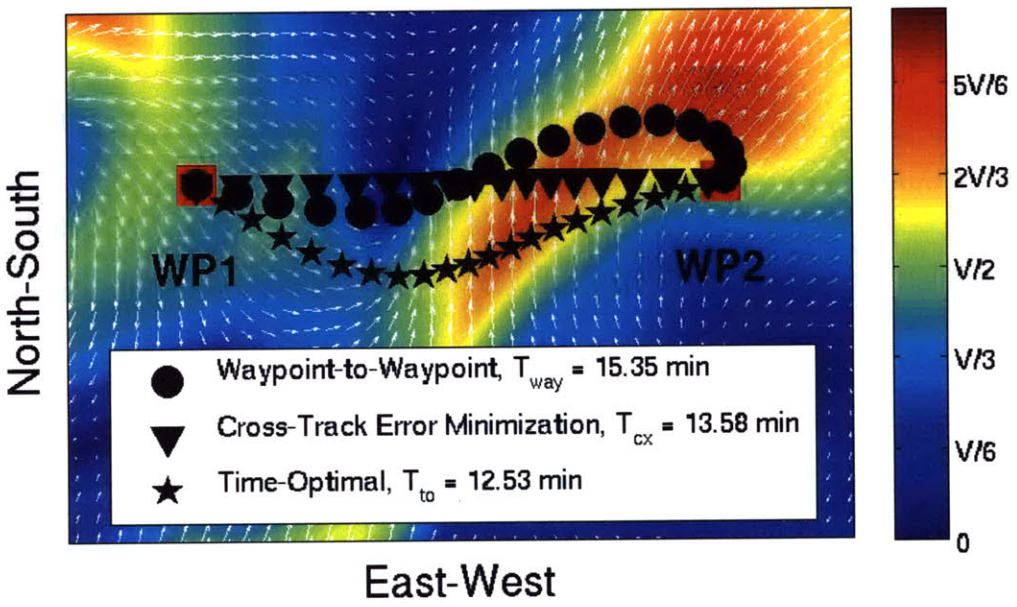


Figure 3-5: 1 km Low-Level Controller Simulation
Here we present a simulated run using the three low-level controllers in the same current field. The colorbar represents the magnitude of the current and V is the velocity of the vehicle relative to the current. As is expected, the time required by the *Time-Optimal* control to complete the trajectory is less than those of *Waypoint-to-Waypoint* and *Cross-Track Error Minimization* control.

3.3.2 *Time-Optimal* Control Computation

There are a few items which must be noted when discussing the *Time-Optimal* controller and its computation.

Case 1: $mindist \gg 0$

Arrival at the destination waypoint using *Time-Optimal* control is greatly dependent on the operating current field and the vehicle's velocity relative to

the current. Because of this, there are cases where the function minimization of *findmindist* returns a *mindist* which does not indicate convergence, i.e. $mindist \gg 0$. In situations such as these, steps must be taken to determine whether the current field is inoperable due to the current magnitude or if it is due to the vehicle settings such as thrust level.

Case 2: Function Minimization - The Importance of the Initial Guess

For our function minimization we utilize Matlab's *fminsearch* which requires an initial heading guess. Because this function is a local minimizer, the possibility of incorrect returns such as presented in Case 1 exists. In hopes of bypassing local minima, which occur regularly in spatially-varying fields, we perform function minimization using eight regularly-spaced initial guesses throughout the 0 to 2π range. Using these returns, we choose the minimum of all eight *mindist* returns and proceed with our regular analysis.

Case 3: Function Minimization Returns Multiple θ_0^*

Due to the fact that current fields tend to be highly dynamic, two or more relatively global minima may be found for *mindist* through function minimization. In situations such as these it is important to explore the t^* associated with the minima. The correct initial heading should be the one corresponding to the minimum t^*.

Case 4: Computation Time

Computation time for *Time-Optimal* control planning is highly dependent on the magnitude and direction of the given current field, the speed of the vehicle relative to the current field, and the distance between waypoints. The contributions are due to the fact that during *findmindist* we must simulate the vehicle's motion which is dependent on these factors. Additionally, computation time is also dependent on the current data form. In all operating harbor environments, current forecasts are discretized in space requiring interpolation during the vehicle's traversal. The interpolation will add to the total computation time.

Case 5: Incorrect Current Field

The *Time-Optimal* control's motion is dependent on the current field. Incorrect data, unless statistically similar in direction and magnitude to the experimental environment, will result in vehicle missteps in the forms of incorrect θ_0^* computation and trajectory.

Chapter 4

Time-Optimal Global Path Planning and Control

The previous chapter presented *Time-Optimal* heading control which utilizes work from Bryson and Ho [9] to find the time-minimal path between two waypoints. This controller implemented in Singapore will decrease the energy required to complete missions where the trajectory is not constrained. Extending this theory to multiple waypoints is also quite useful. Several missions require visiting a number of waypoints, in an unordered fashion, to collect oceanographic or surface data. We propose that by ordering the waypoints in a manner which effectively decreases the time required to complete the entire tour, both energy and time will be saved.

Ordering waypoints is no easy feat when given a large number of waypoints and a spatially-varying current. To tackle the problem at hand, we chose to utilize network optimization methods. Network optimization is a large field ranging from power system research to scheduling [35]. The extensive field of research presents us with a number of system analyzers and solvers to obtain a time-optimal global path between all waypoints.

4.1 The Traveling Salesman Problem (TSP)

4.1.1 Basic Graph Concepts

Network graphs are a fundamental combinatorial structure [35]. Undirected graphs are comprised of two finite sets $G_U = (N, E)$: nodes, $N = \{n_1, n_2, n_3, ..., n_k\}$, and edges, $E = \{e_1, e_2, e_3, \ldots, e_m\}$ [10]. Edges define relationships between nodes meaning if e_1 connects n_1 to n_2 then $e_1 = n_1n_2$. Graphs may also be directed. In directed graphs, $G_D = (N, A)$, there exists a set of nodes, N, and a set of arcs, $A = \{a_1, a_2, a_3, \ldots, a_p\}$. Arcs are ordered and define the direction of the relationship between nodes [10]. When indicating that n_3 is at the head of an arc a_2 and the tail is at n_1, we write $a_2 = (n_1, n_3)$. Figure 4-1(a) shows an example of an undirected graph where

$$N = \{n_1, n_2, n_3, n_4\}$$

and

$$\begin{aligned} E &= \{e_1, e_2, e_3, e_4, e_5, e_6\} \\ &= \{n_1n_2, n_1n_3, n_1n_4, n_2n_3, n_2n_4, n_3n_4\}. \end{aligned}$$

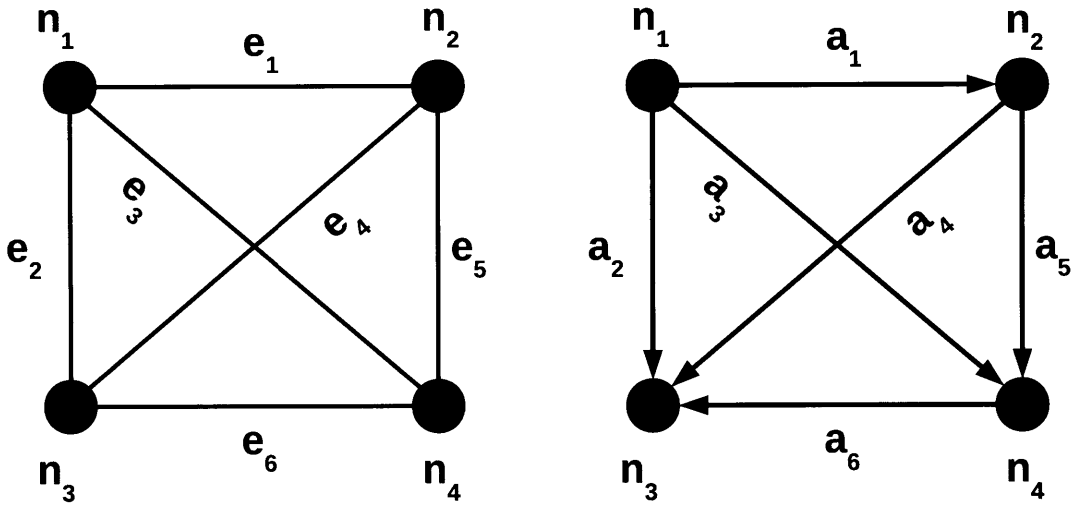
Figure 4-1(b) presents an example of a directed graph where

$$N = \{n_1, n_2, n_3, n_4\}$$

and

$$\begin{aligned} A &= \{a_1, a_2, a_3, a_4, a_5, a_6\} \\ &= \{(n_1, n_2), (n_1, n_3), (n_1, n_4), (n_2, n_3), (n_2, n_4), (n_4, n_3)\}. \end{aligned}$$

Using a network or graph we can then define a cost matrix, C. The cost matrix stores the edge weights and nodal relationships in matrix form [10]. Given a graph containing m nodes we will obtain an $m \times m$ matrix C. Each matrix entry c_{ij} holds the weight associated with its traversal defined by



(a) An undirected graph, G_U.

(b) A directed graph, G_D.

Figure 4-1: Undirected vs. Directed Graphs
Undirected graphs have edges which do not indicate direction. The arcs of directed graphs indicate the directional relationship between nodes.

$$c_{ij} = c(n_i, n_j), \tag{4.1}$$

where $c(.)$ is the cost function. The cost matrix associated with the undirected graph featured in Figure 4-1(a) is the following:

$$C_U = \begin{bmatrix} 0 & c_{12} & c_{13} & c_{14} \\ c_{12} & 0 & c_{23} & c_{24} \\ c_{13} & c_{23} & 0 & c_{34} \\ c_{14} & c_{24} & c_{34} & 0 \end{bmatrix}.$$

In the case of undirected graphs, the cost matrix is symmetric meaning $c_{ij} = c_{ji}$ as indicated in C_U. Whereas the cost matrix associated with the directed graph featured in Figure 4-1(b) is the following:

$$C_D = \begin{bmatrix} 0 & c_{12} & c_{13} & c_{14} \\ \infty & 0 & c_{23} & c_{24} \\ \infty & \infty & 0 & \infty \\ \infty & \infty & c_{43} & 0 \end{bmatrix}.$$

With directed graphs, the cost matrix may not be symmetric meaning $c_{ij} \neq c_{ji}$. It is also important to note that when relationships do not exist between nodes, $c_{ij} = \infty$ indicating that traversal is impossible.

4.1.2 The Traveling Salesman Problem (TSP)

In the traditional sense the TSP sets out to find the shortest route, in terms of distance, for a traveling salesperson both starting and ending at a home city which ensures that each of his or her prescribed cities have been visited. Using a graph, the cities are nodes and the distances between cities are the edges. Traditionally the distances are Euclidean leading to an undirected graph and a symmetric cost matrix.

4.1.3 TSP Formulation for Global Time-Optimal Path Planning in Currents

The aim of global time-optimal path planning in currents is to find the shortest ASC path between all mission waypoints, both starting and ending at the same waypoint. Since we are operating in spatially-varying currents and mission efficiency is important, our aim is not to minimize Euclidean distance. Instead, we aim to minimize the time required to complete the tour.

Given an unordered set of m waypoints and no trajectory constraints we can define a network. The network is composed of the m waypoints as nodes, $N = \{n_1, n_2, \ldots, n_m\}$, and m^2-m arcs, $A = \{(n_1, n_2), (n_2, n_1), \ldots, (n_{m-1}, n_m), (n_m, n_{m-1})\}$. Due to the fact that the arcs of the graph are defined in time, each nodal pair, n_i and n_j, has two directed arcs in reversed directions, i.e. $a_1 = (n_i, n_j)$ and $a_2 = (n_j, n_i)$. The weight corresponding to each arc is now a function of time,

$$t^*_{ij} = \begin{cases} t^*(n_i, n_j), & \text{if the path is possible} \\ \infty, & \text{if the path is not possible} \end{cases} \tag{4.2}$$

where $t^*(n_i, n_j)$ is the time to travel between waypoint i, n_i, and waypoint j, n_j using *Time-Optimal* control and the correct θ^*_0. The cost matrix, C, is instead defined to be a time-optimal cost matrix, C_{T^*}.

$$C_{T^*} = \begin{bmatrix} 0 & t^*_{12} & t^*_{13} & \dots & t^*_{1m} \\ t^*_{21} & 0 & t^*_{23} & \dots & t^*_{2m} \\ t^*_{31} & t^*_{32} & 0 & \dots & t^*_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t^*_{m1} & t^*_{m2} & t^*_{m3} & \dots & 0 \end{bmatrix}.$$

As shown, the time-optimal network graph is directed and the associated cost matrix, C_{T^*}, is asymmetric. Only when the current is effectively zero will C_{T^*} become symmetric.

Since the global time-optimal network has been defined, the next step is to utilize TSP heuristic methods to solve for a time-optimal global path. This will be presented in Section 4.2

A TSP Formulation Example

Given the four waypoints and the time-invariant current forecast shown in Figure 4-2, we must build the network and corresponding cost matrices.

If we were attempting to minimize the total tour based on Euclidean distance we would form the undirected graph featured in Figure 4-3. The corresponding symmetric distance cost matrix would be the following:

$$C_{dist} = \begin{bmatrix} 0 & 283 & 304 & 112 \\ 283 & 0 & 269 & 180 \\ 304 & 269 & 0 & 224 \\ 112 & 180 & 224 & 0 \end{bmatrix},$$

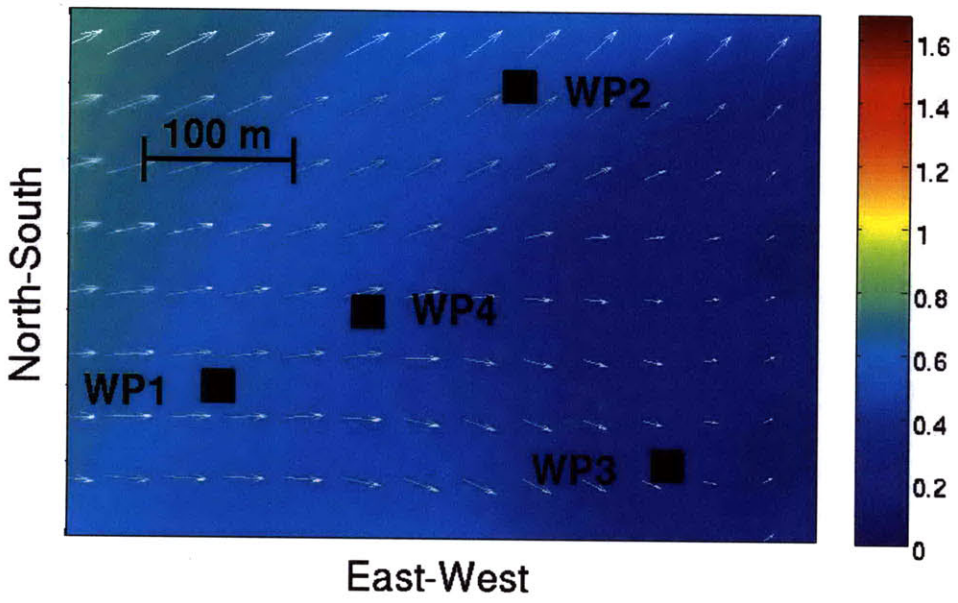


Figure 4-2: TSP Formulation Example - Waypoints and Current Forecast
Four waypoints must be visited during a mission and their locations are indicated here. The colorbar signifies the magnitude of the current as each point in space in units of m/sec.

where the c_{ij} values are in units of meters.

However, since our goal is to minimize the total tour based on time we first determine the t^*_{ij} for each waypoint pair using *Time-Optimal* control. We assume that our vehicle's velocity relative to water is constant, $V = 1.5$ m/sec. The resulting directed graph is shown in Figure 4-4 and the corresponding asymmetric time-optimal cost matrix is the following:

$$C_{T^*} = \begin{bmatrix} 0 & 2.3 & 1.1 & 5.1 \\ 7.5 & 0 & 8.4 & 5.2 \\ 3.8 & 1.8 & 0 & 2.2 \\ 2.3 & 1.3 & 4.8 & 0 \end{bmatrix},$$

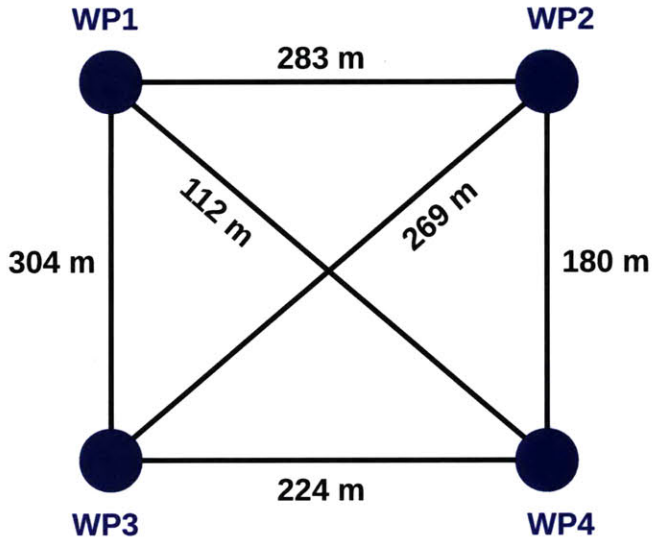


Figure 4-3: TSP Formulation Example - Euclidean Distance Graph
When minimizing a tour over Euclidean distances we first obtain an undirected graph and a corresponding symmetric cost matrix, C_{dist}

where the c_{ij} values are in units of minutes.

4.2 TSP Heuristic Methods

For real-time operation, we do not need to implement exact solvers. Instead, we rely on heuristic methods to provide us with a near-optimal solution showing significant improvement over otherwise implemented missions. Construction heuristics are utilized to produce a good starting tour. Local search, in contrast, is utilized to iteratively improve an already existing tour through sub-tour exchanges.

4.2.1 Tour Construction

As previously mentioned in Section 1.2.3, several tour construction heuristics exist. It has been shown that the initial tour does not greatly impact the final solution produced by local search [36], [25]. Due to this fact, we chose to implement only one

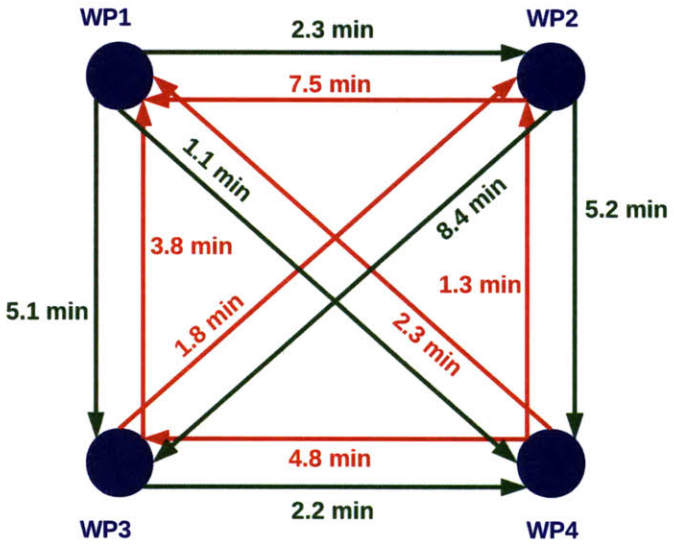


Figure 4-4: TSP Formulation Example - Time-Optimal Graph
When minimizing the total time required to visit all the waypoints we first obtain a directed graph and a corresponding asymmetric cost matrix, C_{T^*}. This result is due to the spatially-varying magnitude and direction of the current field.

type of tour construction: Nearest-Neighbor.

Our procedure, shown in Algorithm 4.1, is based off that of Nilsson [41]. The Nearest-Neighbor method bases its construction off of the costs associated with each arc. After randomly choosing the first arc, the algorithm proceeds to choose the next 'closest' node based on the weight of the arc between them.

Algorithm 4.1 Nearest Neighbor Tour Construction

1: Select a random graph arc and add the corresponding nodes to the current tour.
2: **while** there exists nodes not in the current tour **do**
3: Choose the nearest node and add it to the tour.
4: **end while**
5: **return** The nearest neighbor tour.

Nearest-Neighbor tour construction is of order $O(m^2)$ where m is the number of nodes in the graph.

4.2.2 Local Search

Local search is utilized to improve the tour iteratively by recomputing the cost of the tour after each step to determine if there was an improvement. By removing k edges from a tour and reconnecting them in a new fashion, this altered tour is created. Flood and Croes introduced 2-Opt and 3-Opt, which chooses 2 and 3 edges to remove respectively [17], [12]. Lin and Kernighan introduced the more general form as k-Opt where k edges are removed and rearranged [39].

The traditional k-Opt process removes k edges from a tour, exchanges them in a manner which does not reverse the tour, and then recomputes the difference in costs between the initial tour edges and the rearranged tour edges. An example of this for 2-Opt is the following: Given an initial tour encompassing all nodes,

$$1, 2, \ldots, n_0, n_1, n_2, \ldots, n_k, n_{k+1}, \ldots, m$$

we would like to exchange edge n_0n_1 with edge n_kn_{k+1} to see if there is an improvement. The resulting tour containing an inversion would look like the following:

$$1, 2, \ldots, n_0, n_k, n_{k-1}, \ldots, n_2, n_1, n_{k+1}, \ldots, m$$

[12]. The corresponding cost evaluation for 2-Opt would be the following,

$$\Delta M_{trad}(n_0n_1, n_kn_{k+1}) = c_{0,1} + c_{k,k+1} - c_{0,k} - c_{1,k+1} \tag{4.3}$$

where $c_{i,j}$ is the cost of the edge associated with moving from node n_i to node n_j [12]. If $\Delta M_{trad} > 0$, then this inversion is accepted as an improved tour.

Traditional 3-Opt

Traditional 3-Opt utilizes three arc exchanges to create a new tour in a traditional fashion. What is meant here by 'traditional' is that when exchanging 3 arcs, we must ensure that the tour is not reversed, meaning $c_{ij} \neq c_{ji}$, limiting our possible exchange to only one orientation. Figure 4-5 **H** shows an example of this orientation. Because

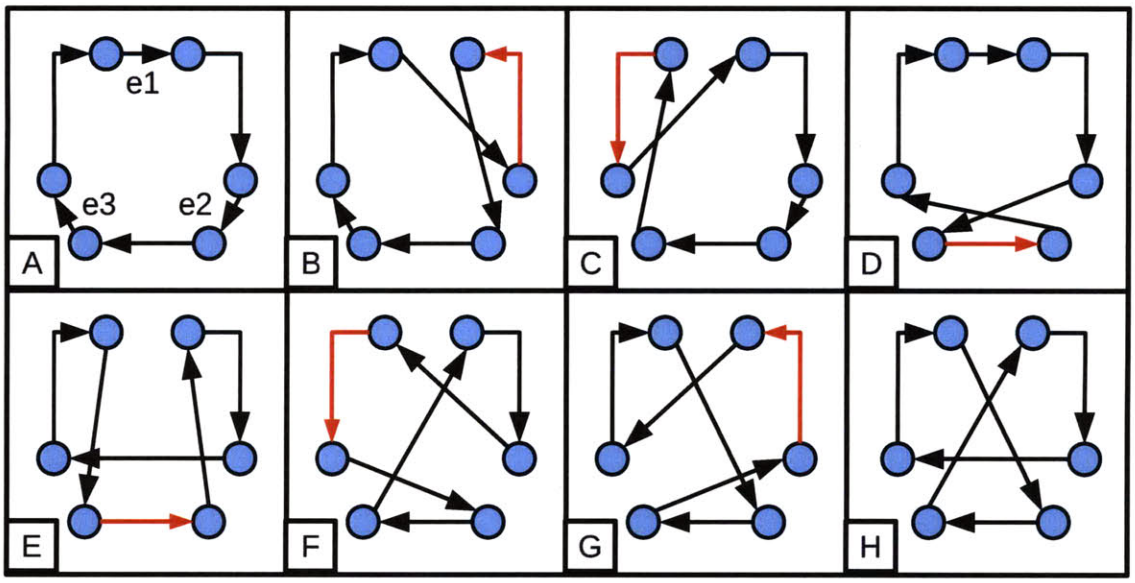


Figure 4-5: Local Search Operations
Local search such as Extended 2-Opt, Extended 3-Opt, and Traditional 3-Opt are utilized to improve an already existing tour. In the above schematic, an existing tour in **A** is to be improved. Red arrows indicate a tour reversal thus requiring that the entire tour cost be recomputed. **B**-**H** demonstrate the seven possible interchanges performed by Extended 3-Opt. Subsets of these are done in cases of Extended 2-Opt shown in **B**-**D**. The only interchange performed by Traditional 3-Opt is shown in **H** where no tour reversal occurs.

we are not reversing the tour in any way, we can continue with the traditional cost evaluation presented in Equation (4.3) with an extension for three edges. The method utilized for Traditional 3-Opt is shown in Algorithm 4.2. Three edges are iteratively removed and exchanged until no more improvements can be found. Due to the fact the three edges must be exchanged to acquire results, $m \geq 6$ where m is the number of nodes in the graph.

Reversing a tour requires more computation time but produces improved final results because more exchange possibilities are evaluated. The computation time comes from the fact that instead of utilizing the difference in edge costs, we must look at the difference in total tour costs due to the fact that our cost matrix is

Algorithm 4.2 Traditional 3-Opt Local Search

1: Compute the cost of the current tour.
2: **while** 3-Opt moves which do not reverse the current tour can improve the cost of the current tour **do**
3: Choose 3 arcs of the current tour and compute their total cost.
4: Remove the 3 arcs of the current tour and reconnect them so that the tour is not reversed.
5: Compute the total cost of the new edges.
6: **if** the total cost of the new edges is less than the total cost of the current edges **then**
7: Set the current tour equal to the new tour.
8: **end if**
9: **end while**
10: **return** The Traditional 3-Opt tour.

asymmetric and $c_{ij} \neq c_{ji}$. For a given tour,

$$1, 2, \ldots, n_0, n_1, n_2, \ldots, n_k, n_{k+1}, \ldots, m$$

this total tour cost, T_C, is equal to

$$T_C = c_{m,1} + \sum_{i=1}^{m-1} c_{i,i+1}. \tag{4.4}$$

The resulting cost evaluation between the initial tour i and the exchanged tour j is

$$\Delta M(\text{tour}_i, \text{tour}_j) = T_{C_i} - T_{C_j}. \tag{4.5}$$

where T_{C_i} is the total tour cost of tour i described by Equation (4.4). Again, if $\Delta M > 0$ then the new exchanged tour is an improvement over the initial tour. We will see this non-traditional cost evaluation applied using Extended 2-Opt and Extended 3-Opt in the following sections.

Extended 2-Opt

Extended 2-Opt is an extension of the previously mentioned 2-Opt where two edges are removed and exchanged to create a new tour. Due to the fact that the cost

matrices described in Section 4.1.3 are asymmetric, meaning $c_{ij} \neq c_{ji}$, we cannot utilize traditional opt-exchange cost evaluation presented in Equation (4.3). Instead we must evaluate the change in cost of the entire tour as described by Equation (4.5). The resulting method, Extended 2-Opt, is described in Algorithm 4.3. The two edge exchange process is continued iteratively until no new improved tour can be found. Figure 4-5 **B-D** show an examples of possible Extended 2-Opt exchanges for the given tour in **A**.

Algorithm 4.3 Extended 2-Opt Local Search

```
1: Compute the cost of the current tour.
2: while 2-Opt moves can improve the cost of the current tour do
3:    Remove 2 arcs of the current tour and reconnect them in the most improved
      manner.
4:    Compute the cost of the new tour.
5:    if the cost of the new tour is less than the cost of the current tour then
6:       Set the current tour equal to the new tour.
7:    end if
8: end while
9: return The Extended 2-Opt tour.
```

Extended 3-Opt

Extended 2-Opt exchanges present good results as will be shown in Section 4.5.1, but limits the number of tour configurations explored. To attempt improvement in optimality, we broaden Extended 2-Opt to Extended 3-Opt utilizing the non-traditional cost evaluation presented by Equation (4.5) due to the fact that Extended 3-Opt can reverse the tour. As shown in Figure 4-5 **B-H**, Extended 3-Opt has seven tour exchanges for improvement, three of which are Extended 2-Opt. At each iteration, Extended 3-Opt evaluates all seven of these tour costs and chooses the best to test against the non-exchanged tour. Although computation time is much larger we see an improvement in the cost of the total tour. Algorithm 4.4 describes the implementation method for Extended 3-Opt. Also, due to the fact that we must exchange three arcs to get results, $m \geq 6$ where m is the number of nodes in the graph.

Algorithm 4.4 Extended 3-Opt Local Search

1: Compute the cost of the current tour.
2: **while** 3-opt moves can improve the cost of the current tour **do**
3: Remove 3 arcs of the current tour and reconnect them all possible configurations.
4: Compute the cost of each reconfigured tour and choose the most improved tour to be the new tour.
5: **if** the cost of the new tour is less than the cost of the current tour **then**
6: Set the current tour equal to the new tour.
7: **end if**
8: **end while**
9: **return** The Extended 3-Opt tour.

4.3 Time-Optimal Global Path Simulation

To emphasize the impact using *Time-Optimal* control in the field we now present a simulation encompassing this control law and the TSP heuristic methods mentioned previously. Using a given current field forecast and six waypoint locations in a 475 m $\times$ 450 m area we begin by obtaining our cost matrices. The first, obtained using the Eulerian distances between the waypoints is

$$C_{dist} = \begin{bmatrix} 0 & 282.8 & 158.1 & 111.8 & 161.5 & 292.6 \\ 282.8 & 0 & 158.1 & 180.2 & 264.7 & 125.0 \\ 158.1 & 158.1 & 0 & 111.8 & 232.5 & 230.4 \\ 111.8 & 180.2 & 111.8 & 0 & 120.8 & 182.0 \\ 161.5 & 264.7 & 232.5 & 120.8 & 0 & 203.0 \\ 292.6 & 125.0 & 230.4 & 182.0 & 203.0 & 0 \end{bmatrix}$$

where the values are in meters. The second matrix, the time-optimal cost matrix, is created under the assumption $V = 1.5$ m/sec where V is the velocity of the vehicle relative to water. To build the time-optimal cost matrix, C_{T^*}, we use *Time-Optimal* control for the technique described in Section 4.1.3. Using *Waypoint-to-Waypoint*

control for our reference navigation time, t_{ref}, we obtain

$$C_{T^*} = \begin{bmatrix} 0 & 159 & 109 & 59 & 77 & 146 \\ \infty & 0 & \infty & 189 & 188 & 59 \\ 118 & 66 & 0 & 57 & 117 & 95 \\ 103 & 112 & 118 & 0 & 57 & 84 \\ 164 & 233 & 249 & 131 & 0 & 136 \\ \infty & 159 & \infty & \infty & 175 & 0 \end{bmatrix}.$$

where the values are in seconds. The ∞ emerges in the time-optimal cost matrix when travel between those two waypoints is impossible due to relatively strong currents and insufficient thrust levels. To make our jobs easier, we form a utility matrix, Θ_0^*, which stores the correct *Time-Optimal* control initial heading θ_0^* associated with each waypoint-to-waypoint path. This matrix will not be presented here to maintain simplicity.

4.3.1 Tour Construction

In order to obtain our near-time-optimal global tours we must start through tour construction. Using arc $(WP5, WP6)$ as our randomly chosen starting arc for our Nearest-Neighbor tour, we continue by applying Algorithm 4.1 to build a tour. The process uses the cost matrix, C_{T^*}, to find the next 'closest' waypoint in time by comparing t^* values.

Initiation

$$\text{NN tour} = \{WP5, WP6\}$$

Iteration 1

$$\begin{aligned} \text{NN tour} &= \{WP5, WP6, WP(min\{t_{61}^*, t_{62}^*, t_{63}^*, t_{64}^*\})\} \\ &= \{WP5, WP6, WP2\} \end{aligned}$$

Iteration 2

$$\begin{aligned}\text{NN tour} &= \{WP5, WP6, WP2, WP(min\{t^*_{21}, t^*_{23}, t^*_{24}\})\} \\ &= \{WP5, WP6, WP2, WP4\}\end{aligned}$$

Final Iteration

$$\text{NN tour} = \{WP5, WP6, WP2, WP4, WP1, WP3\}$$

Proceeding to simulate this path using *Time-Optimal* control and the associated correct initial headings from Θ^*_0, we can compute the total travel time for the final Nearest-Neighbor tour, $T_{NN} = 13.6$ min. The tour is depicted in Figure 4-6.

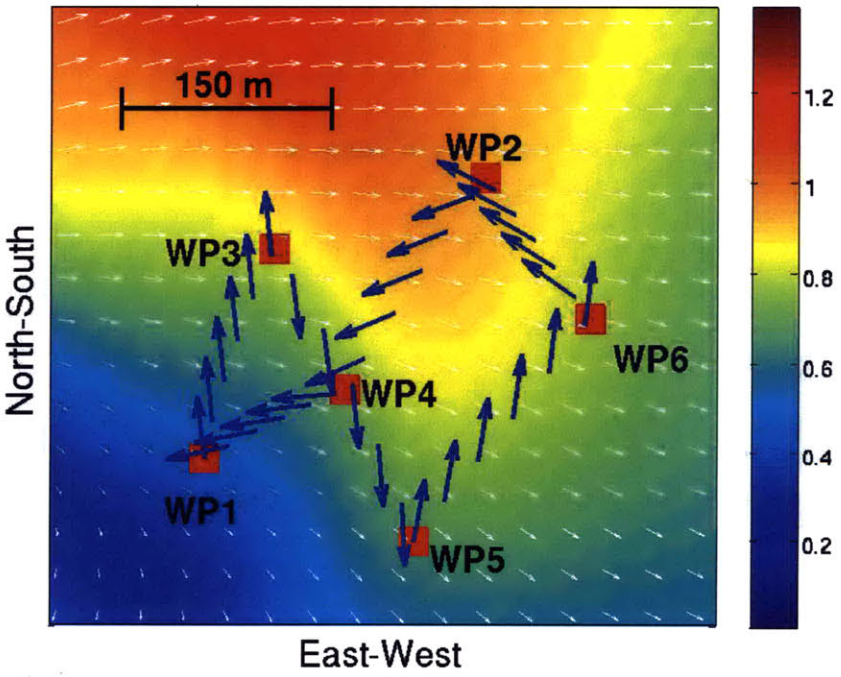


Figure 4-6: TSP Heuristics Simulation - NN Tour
Randomly choosing arc $(WP5, WP6)$ for the Nearest-Neighbor initialization leads to the tour depicted above. The total time to complete the tour was 13.6 min using the given current field and a constant vehicle velocity of $V = 1.5$ m/sec.

4.3.2 Local Search

Using the Nearest-Neighbor tour obtained in Section 4.3.1 we can proceed to improve the tour using local search. Because we are using so few waypoints we will only present Extended 2-Opt in detail using Algorithm 4.3.

Initiation

Set the current tour equal to the Nearest-Neighbor tour solution.

$$\text{tour}_{curr} = \{WP5, WP6, WP2, WP4, WP1, WP3\}$$

Compute the cost of the current tour.

$$T_{tour_{curr}} = 13.6 \text{ min}$$

Iteration 1

Attempt to exchange arc $(WP5, WP6)$ with arc $(WP2, WP4)$.

$$\text{tour}_{new} = \{WP5, WP2, WP6, WP4, WP1, WP3\}$$

Compute the new tour cost.

$$T_{tour_{curr}} = \infty$$

Check the cost evaluation between the current tour and the new tour.

$$\Delta M = -\infty < 0 \Rightarrow \text{ Do not accept this new tour.}$$

$$\text{tour}_{curr} = \{WP5, WP6, WP2, WP4, WP1, WP3\}$$

Iteration 2

Attempt to exchange arc $(WP5, WP6)$ with arc $(WP4, WP1)$.

$$\text{tour}_{new} = \{WP5, WP4, WP2, WP6, WP1, WP3\}$$

Compute the new tour cost.

$$T_{tour_{curr}} = \infty$$

Check the cost evaluation between the current tour and the new tour.

$$\Delta M = -\infty < 0 \Rightarrow \text{ Do not accept this new tour.}$$

$$\text{tour}_{curr} = \{WP5, WP6, WP2, WP4, WP1, WP3\}$$

Iteration 10

Attempt to exchange arc $(WP4, WP1)$ with arc $(WP3, WP5)$.

$$\text{tour}_{new} = \{WP4, WP3, WP1, WP5, WP6, WP2\}$$

Compute the new tour cost.

$$T_{tour_{curr}} = 13.3$$

Check the cost evaluation between the current tour and the new tour.

$$\Delta M = 13.6 - 13.3 > 0 \Rightarrow \text{ Accept this new tour.}$$

$$\text{tour}_{curr} = \{WP4, WP3, WP1, WP5, WP6, WP2\}$$

Final Iteration

$$\text{Extended 2-Opt tour} = \{WP1, WP4, WP3, WP2, WP6, WP5\}$$

Simulating this path using *Time-Optimal* control and the associated correct initial headings from Θ_0^*, we can compute the total travel time for the final Extended 2-Opt tour, $T_{Ext.2-Opt} = 10.7$ min. The Extended 2-Opt tour is shown in Figure 4-7.

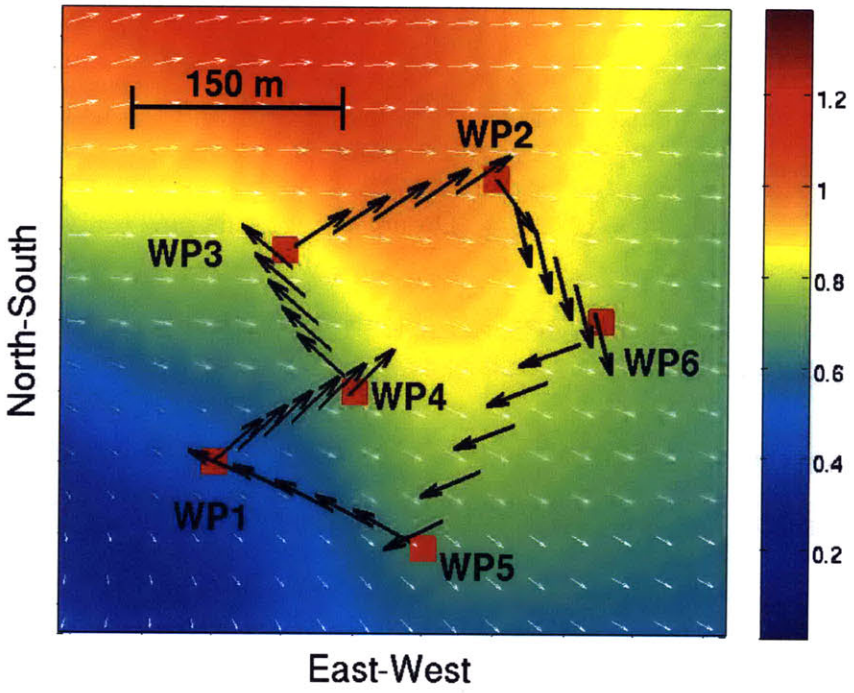


Figure 4-7: TSP Heuristics Simulation - Extended 2-Opt Tour
Extended 2-Opt Local search using the Nearest-Neighbor initial tour resulted in the tour depicted above. The total time needed to complete the tour was 10.7 min for the given current field and the constant vehicle velocity, $V = 1.5$ m/sec.

4.3.3 Generating a Reference Tour

To show the benefit of utilizing TSP heuristics to find a near-time-optimal tour we must provide a baseline tour. The reference tour we choose to utilize is composed using Nearest-Neighbor in conjunction with the cost matrix C_{dist}. Using this process to create a reference tour generalizes what researchers would do if current forecasts were not available. Using $(WP5, WP2)$ as our randomly selected initiation arc we find the Nearest-Neighbor reference tour by choosing the next closest waypoint in terms of Euclidean distance. The resulting reference tour is

$$\text{NN}_{dist} \text{ tour} = \{WP5, WP2, WP4, WP1, WP3, WP6\}$$

and the total travel time for this tour is $T_{NN_{dist}} = 15.1$ min. The resulting tour is shown in Figure 4-8.

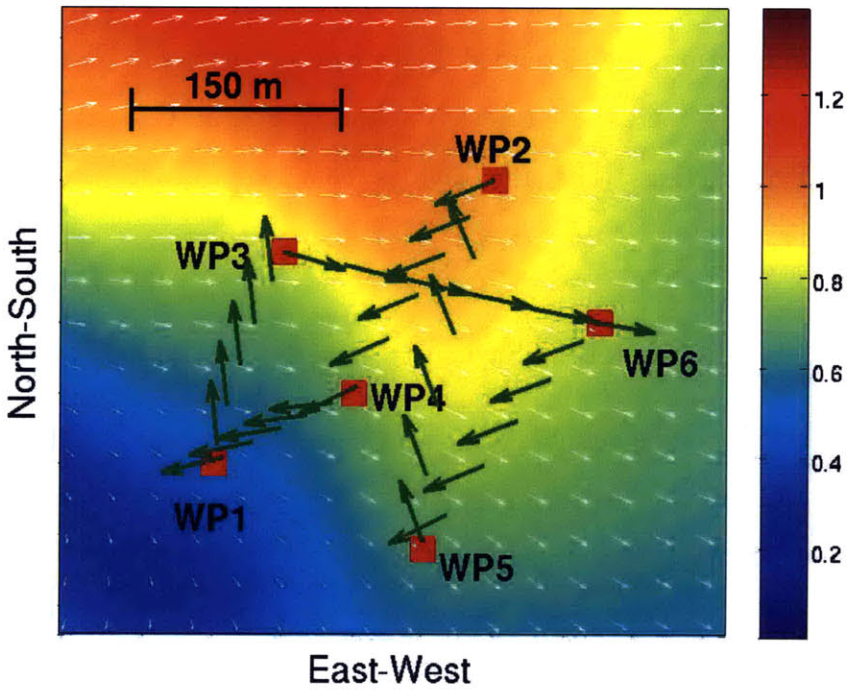


Figure 4-8: TSP Heuristics Simulation - NN_{dist} Tour
Randomly choosing arc $(WP5, WP2)$ for the Euclidean distance Nearest-Neighbor initialization leads to the tour depicted above. The total time required to complete the tour was 15.1 min using the given current field and vehicle velocity, $V = 1.5$ m/sec.

4.3.4 Simulation Analysis

We are aware that solving for the global time-optimal path problem must utilize heuristics. When we are presented with a set of waypoints we have a very small probability of choosing the optimal order using Nearest-Neighbor based on Euclidean distance. Given m waypoints, there exists $m!$ permutations of their order. Thus we have a probability,

$$P = \frac{1}{m!}$$

of choosing the correct optimal tour. In this simulation instance, that probability is $\frac{1}{720}$. It is evident that as m increases, the probability decreases. With such small probabilities it becomes clear why utilizing heuristics assists in the discovery of an optimal tour.

Within our simulation, the Extended 2-Opt tour outperformed both the Nearest-Neighbor based on C_{T^*} and the Nearest-Neighbor based on C_{dist} as is shown using Tour Relative Time Savings. The time savings from tour-to-tour is computed in the following manner:

$$\text{Tour Relative Time Savings} = \frac{T_{NN_{dist}} - T_{tsp}}{T_{NN_{dist}}} \times 100\%. \tag{4.6}$$

where T_{tsp} is the tour time required by the given TSP method. These results are shown in Table 4.1. In this instance, the final Extended 2-Opt tour was able to save 29.1% time over that of the reference tour, NN_{dist} tour. Of course NN_{dist} was generated using a randomly chosen edge. Due to this waypoint configuration and current forecast, it is possible that the reference tour would have generated the optimal tour, however as previously mentioned, the odds are unlikely. Instead, using heuristics allows us to reach that near-time-optimal tour. As an example, if our Nearest-Neighbor construction tour had turned out to be the following,

$$\text{NN}_{dist} \text{ tour} = \{WP6, WP2, WP5, WP4, WP1, WP3\}$$

our Extended 2-Opt local search would have still concluded with the same near-time-optimal result.

In Section 4.5 we will further discuss the implication of TSP heuristics and the computation time involved in the solutions gathered.

4.4 Heuristic Methods Experimentation

Implementation of theory in real oceanographic environments speaks well to the strengths of research. In January 2010, we set out to prove that the ordering of

Table 4.1: Tour Relative Time Savings for TSP Heuristics Example
Using Equation (4.6) we can evaluate the tour relative time savings for our NN_{dist}, NN, and Extended 2-Opt tours.

Tour Relative Time Savings [%]			
	NN_{dist}	NN	Extended 2-Opt
NN_{dist}	0	10	29
NN		0	21
Extended 2-Opt			0

mission waypoints has a significant impact on the total time and energy required by the mission by creating path plans.

Test Setup

From January 11-15, 2010 a group of CENSAM researchers and support staff [18] set out to the Selat Pauh area of Singapore to perform multiple oceanographic tests and data sampling. Using a test area of 900 m $\times$ 500 m, six waypoints were chosen randomly. A single ASC was used for all of the tests. It was equipped with GPS, an IMU, and wireless communication.

Prior to arrival the two cost matrices were computed: C_{dist}, a symmetric Euclidean distance cost matrix obtained using the waypoint GPS locations, and C_T, the asymmetric time cost matrix. C_T, though not optimal, was computed using *Waypoint-to-Waypoint* control predicted paths from a current forecast provided by TMSI for the proposed mission time. The velocity of the ASC, V, was assumed to be constant at 2 m/sec. Also, it was assumed that the current predictions were accurate and that the current field would not drastically change during mission operation.

In order to evaluate the benefit of utilizing TSP heuristics a reference tour was generated. The reference tour utilized Nearest-Neighbor and the cost matrix C_{dist} to obtain a tour encompassing all six waypoints. The proposed reference tour demonstrates what researchers would choose to do if current forecasts were not readily available.

Using C_T a Nearest-Neighbor tour was generated. A final tour was constructed

using Extended 2-Opt.

January 13, 2010

C_{dist} and C_T were computed using current forecasts from TMSI [58] in the Selat Pauh area for 3 p.m. Using methods previously described, NN_{dist} path was found:

$$\text{NN}_{dist} \text{ tour} = \{WP4, WP6, WP5, WP3, WP2, WP1\}.$$

Using a thrust percentage of 100% and a departure time of 2:50 p.m., the mission was run using *Waypoint-to-Waypoint* control. The total time required to complete the tour was $T_{NN_{dist}} = 9.14$ min. The mean velocity of the vehicle over water, V, during this period was 1.61 m/sec.

The tour generated by C_T using Nearest-Neighbor tour construction and then Extended 2-Opt local search was:

$$\text{Extended 2-Opt tour} = \{WP4, WP6, WP2, WP1, WP3, WP5\}.$$

Using a thrust percentage of 100 % and a departure time of 3:02 p.m., the mission was run using *Waypoint-to-Waypoint* control. The total time required to complete the Extended 2-Opt tour was $T_{Ext.2-Opt} = 7.15$ min. The mean velocity of the vehicle over water, V, during this period was 1.77 m/sec.

The time savings between tours, computed using Equation (4.6), was found to be 21.8% as shown below.

$$\text{Tour Relative Time Savings} = \frac{9.14 - 7.15}{9.14} \times 100\% \approx 22\%$$

The trajectories of these two tours are shown in Figure 4-9.

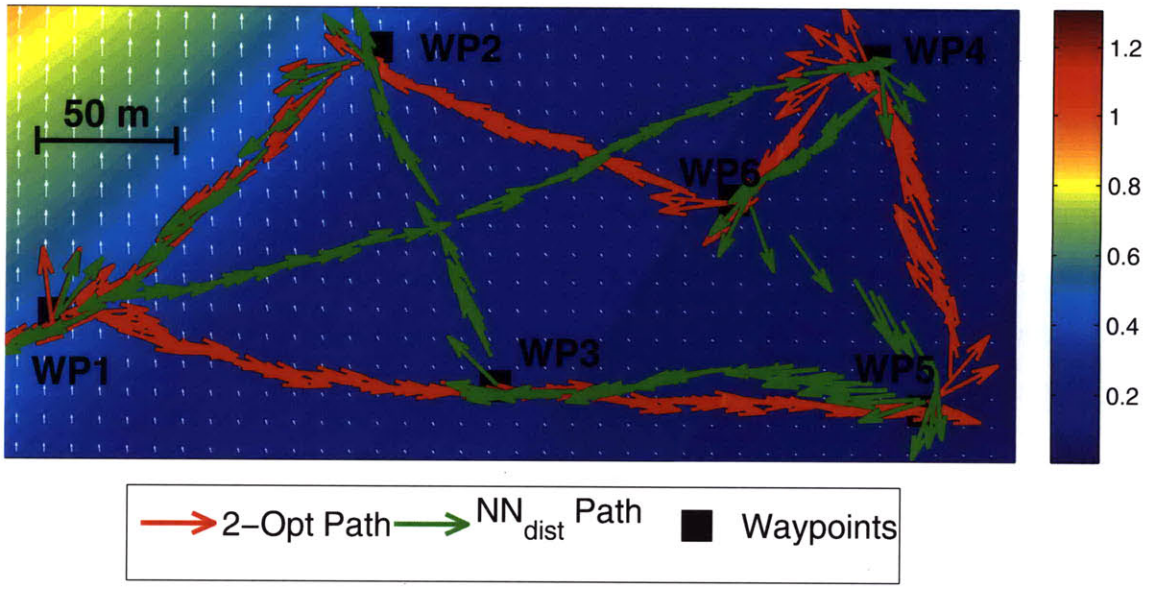


Figure 4-9: January 13, 2010 TSP Heuristic Experiment
Two tours were executed and analyzed. The first was found using Nearest-Neighbor tour construction based on the Euclidean distances between waypoints. The second utilized Nearest-Neighbor and then Extended 2-Opt to maximize time savings. In this experiment a relative time savings of about 22% was found.

January 15, 2010

C_{dist} and C_T were computed using current forecasts from TMSI [58] in the Selat Pauh area for 1:55 p.m. Using methods described in Section 4.4, NN_{dist} path was found:

$$\text{NN}_{dist} \text{ tour} = \{WP2, WP4, WP1, WP6, WP5, WP3\}.$$

Using a thrust percentage of 100% and a departure time of 1:53 p.m., the mission was run using *Waypoint-to-Waypoint* control. The total time required to complete the tour was $T_{NN_{dist}} = 8.12$ min. The mean velocity of the vehicle over water, V, during this period was 1.73 m/sec.

The tour generated by C_T using Nearest-Neighbor tour construction and then

Extended 2-Opt local search was:

$$\text{Extended 2-Opt tour} = \{WP4, WP6, WP2, WP1, WP3, WP5\}.$$

Using a thrust percentage of 100 % and a departure time of 2:08 p.m., the mission was run using *Waypoint-to-Waypoint* control. The total time required to complete the Extended 2-Opt tour was $T_{Ext.2-Opt} = 7.39$ min. The mean velocity of the vehicle over water, V, during this period was 1.70 m/sec.

The time savings between tours, computed using Equation (4.6), was found to be 8.99% as shown below.

$$\text{Tour Relative Time Savings} = \frac{8.12 - 7.39}{8.12} \times 100\% \approx 9\%$$

The trajectories of these two tours are shown in Figure 4-10.

Test Analysis

While we cannot confirm the accuracy of the current forecasts provided to us by TMSI [58], we can draw a solid conclusion that the order the waypoints are visited makes a difference in the amount of time it takes to complete a tour. In both experiments, the Extended 2-Opt tour have time savings over that of the Nearest-Neighbor from Euclidean distance. The experiment which took place on January 13, 2010 showed more savings due to the relatively small current magnitude and spacial variability. Also, the vehicle's velocity was 0.16 m/sec faster during its Extended 2-Opt tour than its Nearest-Neighbor tour which may in part be due to the waypoint ordering. The January 15, 2010 experiment showed less savings but more current variability and similar vehicle velocity.

Although we used six waypoints for these experiments we suspect that the inclusion of more waypoints will lead to further improved tours due to the increased number of tour configurations. We also would like to note that we did not use *Time-Optimal* control for these experiments. Implementation of this control law in addition

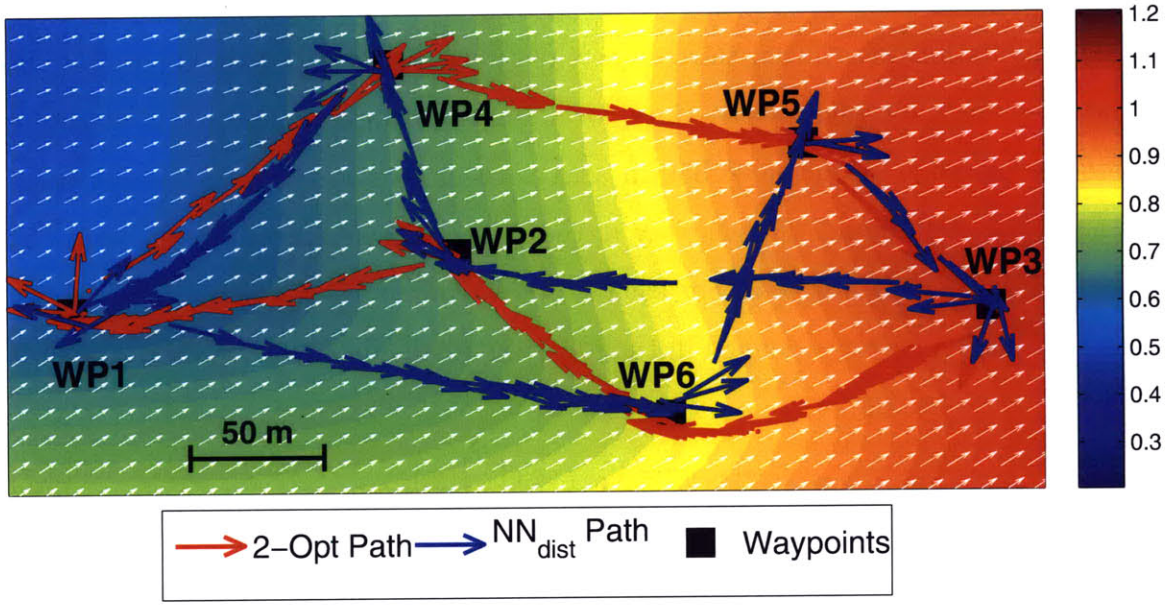


Figure 4-10: January 15, 2010 TSP Heuristic Experiment
Two tours were executed and analyzed. The first was found using Nearest-Neighbor tour construction based on the Euclidean distances between waypoints. The second utilized Nearest-Neighbor and then Extended 2-Opt to maximize time savings. In this experiment a relative time savings of about 9% was found.

to utilizing TSP heuristic techniques will lead to near-time-optimal global paths.

4.5 Time-Optimal Global Path Planning and Control Analysis

4.5.1 Time Savings Contribution

The impact of utilizing a TSP formulation to find a minimal time tour is really shown through numerous waypoint sets in a number of current fields. Within this section we hope to convince the reader that significant savings is achievable through this

methodology.

Time Savings Simulation Setup

In this simulation we utilized thirteen sets of twenty waypoints to take note of the trend in savings for each of the three local search methods. The waypoints were randomly placed in a 350 m $\times$ 350 m area with an average waypoint-to-waypoint distance, μ_{dist}, of 165 m. The velocity of the vehicle relative to water was set to be a constant, $V_{ASC} = 1.5$ m/sec. For each set of waypoints we simulated a number of spatially-varying currents defined in the following manner:

$$u(x, y) = V_{curr} \cos\left(\frac{2\pi}{\lambda} y\right) \tag{4.7}$$

$$v(x, y) = 0 \tag{4.8}$$

where $u(x, y)$ and $v(x, y)$ are the current velocities in the x- and y- directions, V_{curr} is the amplitude of the current velocity, and λ is the wavelength of the current field. Figure 4-11 displays these values pictorially. The V_{curr} test values ranged from 0 to V_{ASC} m/sec and the λ test values ranged from $5\mu_{dist}$ to $60\mu_{dist}$ m. For each combination of waypoint set, V_{curr} and λ, we used *Time-Optimal* control to build our time-optimal cost matrix, C_{T^*}, using *Waypoint-to-Waypoint* control as our reference control. Using these time-optimal control matrices we proceeded to construct twenty initial tours using Nearest-Neighbor tour construction for a randomly chosen arc. For each of these initial tours we used all three local search techniques mentioned previously to minimize the tour.

In order to determine time savings we once again used a reference tour for each of the thirteen sets of waypoints. Using the Euclidean distance between waypoints we found a symmetric cost matrix C_{dist}. To make our results fair we then proceeded to compute every possible Nearest-Neighbor solution based on Euclidean distance and its respective total trajectory time for the given set of waypoints and current field. Thus for each waypoint set we obtain two reference times for later analysis,

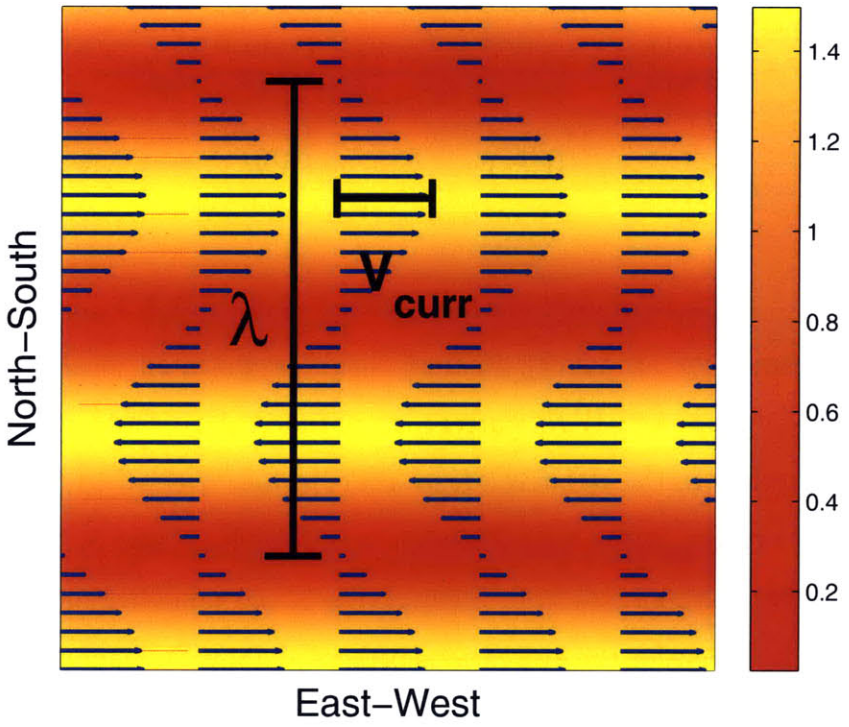


Figure 4-11: Time Savings Simulation Current Variables
This figure describes the varying current profile used during the time savings simulation and shown in Equations (4.7) and (4.8). V_{curr} refers to the amplitude of the current in the x-direction. λ is the wavelength of that current.

$$t_{ref,mean}(WP, V_{curr}, \lambda) = \text{mean}(T_{NN_{dist}}(WP, V_{curr}, \lambda)) \tag{4.9}$$

$$t_{ref,max}(WP, V_{curr}, \lambda) = \max(T_{NN_{dist}}(WP, V_{curr}, \lambda)) \tag{4.10}$$

where WP is the given set of waypoints and $T_{NN_{dist}}$ is the total tour time for all of the possible Nearest-Neighbor tours based on Euclidean distance. $t_{ref,mean}$ computes the mean time and $t_{ref,max}$ computes the maximum possible time.

In order to compute the time savings capabilities of the local search techniques we used the following computations:

$$t_{ls,mean}(V_{curr},\lambda) = \text{mean}\left[\frac{t_{ref,mean}(WP,V_{curr},\lambda) - T_{ls}(WP,V_{curr},\lambda,\text{NN tour})}{t_{ref,mean}(WP,V_{curr},\lambda)}\right]\times 100\% \tag{4.11}$$

$$t_{ls,max}(V_{curr},\lambda) = \max\left[\frac{t_{ref,max}(WP,V_{curr},\lambda) - T_{ls}(WP,V_{curr},\lambda,\text{NN tour})}{t_{ref,max}(WP,V_{curr},\lambda)}\right]\times 100\% \tag{4.12}$$

where ls refers to the local search method utilized and $T_{ls}(WP, V_{curr}, \lambda, \text{NN tour})$ is the total tour times for all twenty Nearest-Neighbor initiated local search results for the given waypoint set, WP, current amplitudes, V_{curr}, and wavelengths, λ. $t_{ls,mean}(V_{curr},\lambda)$ refers to the Relative Mean Tour Savings over all 13 (waypoint sets) $\times$ 20 (Nearest-Neighbor initial tours) instances of that V_{curr} and λ. $t_{ls,max}(V_{curr},\lambda)$ refers to the Relative Maximum Tour Savings over those same instances.

It must be noted that for the evaluations presented above, we ignored all instances where the tours resulted in impossible trajectories. Also, the figures presented have all been non-dimensionalized using λ/μ_{dist} and V_{curr}/V_{ASC} as the x- and y- axes respectively.

Simulation Results & Analysis

The following figures depicts the results of the analysis utilizing a log scale for clarity. Relative Mean Tour Savings and Relative Maximum Tour Savings for Traditional 3-Opt are shown in Figures 4-12 and 4-13. For Extended 2-Opt, these results are displayed in Figures 4-14 and 4-15. The Extended 3-Opt results are presented in Figures 4-16 and 4-17. The savings values were computed using Equations (4.11) and (4.12).

Based on these simulations, it is fair to state that at least 14% savings is possible using local search methods with Extended 3-Opt achieving the largest improvement. Using minimum Relative Mean Tour Savings as a lower bound, Extended 3-Opt presents the best opportunity for improvement. In these simulations the minimum time savings was 22% at $\frac{\lambda}{\mu_{dist}} = 35$ and $\frac{V_{curr}}{V_{ASC}} = 0.13$ using the Extended 3-Opt local search method. Extended 2-Opt and Traditional 3-Opt have minmas of 20% and

Table 4.2: Local Search Simulation Savings Statistics

Simulation Statistics			
		Mean[%](V_{curr}/V_{ASC},λ/μ_{dist})	Max[%](V_{curr}/V_{ASC}, λ/μ_{dist})
Traditional 3-Opt	Min	14 (0.4, 25)	40 (0.53, 35)
	Max	77 (1, 25)	88 (1, 15)
Extended 2-Opt	Min	20 (0, 35)	42 (0.53, 55)
	Max	70 (1, 25)	86 (1, 15)
Extended 3-Opt	Min	22 (0.13, 35)	42 (0.53, 55)
	Max	78 (1, 25)	89 (1, 15)

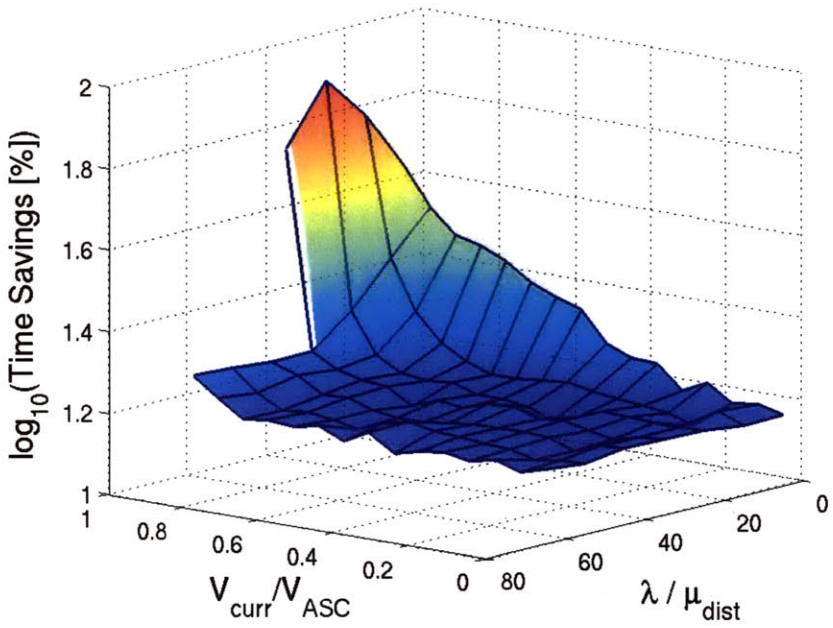


Figure 4-12: Traditional 3-Opt Relative Mean Tour Savings
The minimum savings of 14% was found at $\frac{\lambda}{\mu_{dist}} = 25$ and $\frac{V_{curr}}{V_{ASC}} = 0.4$. The maximum savings of 77% was noted at $\frac{\lambda}{\mu_{dist}} = 25$ and $\frac{V_{curr}}{V_{ASC}} = 1$.

14%.

Extended 2-Opt local search also achieved impressive savings, yet did not achieve the maximums like Extended 3-Opt and Traditional 3-Opt. This fact can be attributed to the limited number of configurations presented by a two arc exchange relative to the other three arc exchanges. Traditional 3-Opt presents the greatest fluctuation between savings with a minimum mean savings of 14% and a maximum mean savings of 77% over the various current fields. Because Traditional 3-Opt eval-

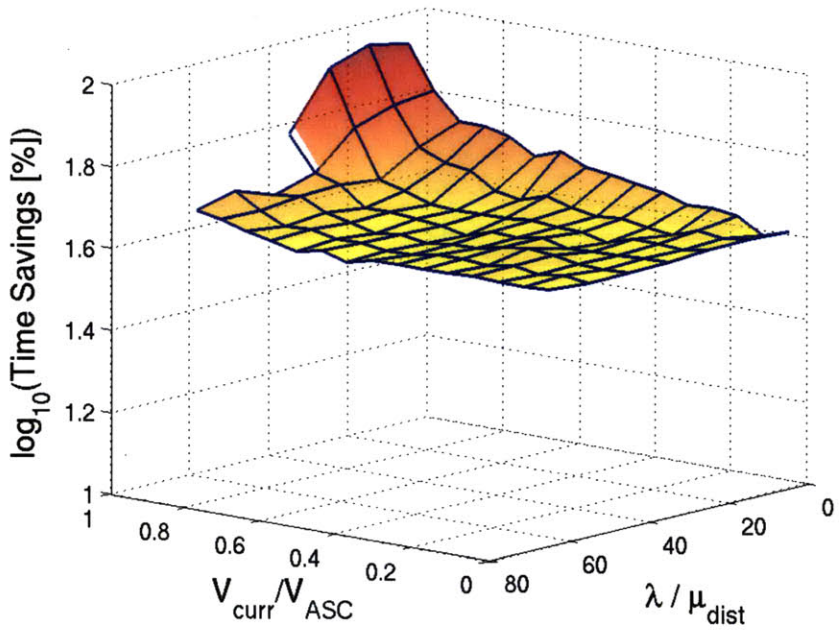


Figure 4-13: Traditional 3-Opt Relative Max Tour Savings
The minimum savings of 40% was found at $\frac{\lambda}{\mu_{dist}} = 35$ and $\frac{V_{curr}}{V_{ASC}} = 0.53$. The maximum savings of 88% was noted at $\frac{\lambda}{\mu_{dist}} = 15$ and $\frac{V_{curr}}{V_{ASC}} = 1$.

uates those tours which do not reverse the initial tour, the probability of savings is limited.

In each local search instance it is evident that with a decrease in current wavelength, λ, and an increase in the current amplitude, V_{curr}, potential savings increases. With such changes, the current becomes more and more influential on the path of the ASC. In some instances, the current was inoperable as is evidenced when V_{curr} = 1 m/sec and λ = 65, 55, and 45 m. Thus, TSP network heuristics can be use to pinpoint both successful and unsuccessful missions.

In addition to savings, it it also important to understand the computation time involved for each of the search methods. Section 4.5.2 will discuss this matter.

4.5.2 TSP Computation

Although maximum time savings is desirable, the computation time involved may be excessive. Because of this we must analyze the three local search heuristic methods

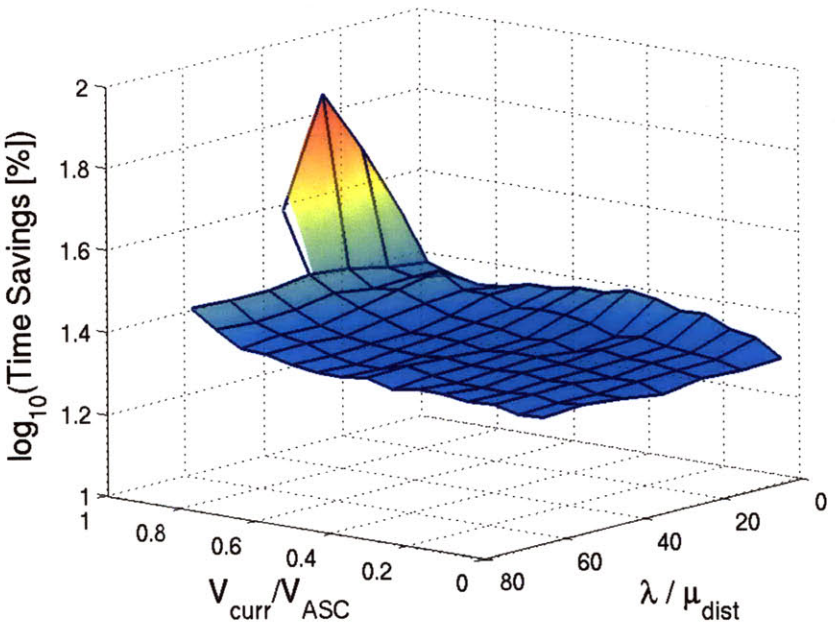


Figure 4-14: Extended 2-Opt Relative Mean Tour Savings
The minimum savings of 20% was found at $\frac{\lambda}{\mu_{dist}} = 35$ and $\frac{V_{curr}}{V_{ASC}} = 0$. The maximum savings of 70% was noted at $\frac{\lambda}{\mu_{dist}} = 25$ and $\frac{V_{curr}}{V_{ASC}} = 1$.

in detail.

TSP Local Search Computation vs. Time

Using the simulation presented in Section 4.5.1, we chose to take one current profile and analyze the computation time involved in all instances. The profile we chose encompassed the following non-dimensionalized values: $\frac{\lambda}{\mu_{dist}} = 25$ and $\frac{V_{curr}}{V_{ASC}} = 0.47$. All instances are presented in Figure 4-18.

It is clear from Figure 4-18 that Extended 3-Opt requires the most amount of computation time alongside the maximum amount of savings. Its variability with computation time and savings is minimal compared to those of Traditional 3-Opt and Extended 2-Opt. However, its computation time is about ten times greater that that of both Traditional 3-Opt and Extended 2-Opt. As more waypoints are added to the set, this computation time could become significant.

Figure 4-18 also tells us that Extended 2-Opt requires more time than that of

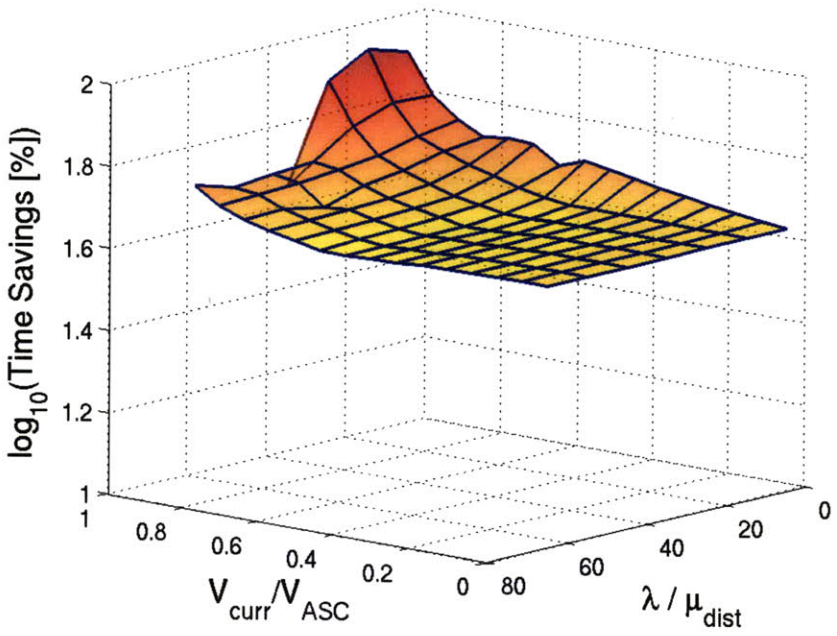


Figure 4-15: Extended 2-Opt Relative Max Tour Savings
The minimum savings of 42% was found at $\frac{\lambda}{\mu_{dist}} = 55$ and $\frac{V_{curr}}{V_{ASC}} = 0.53$. The maximum savings of 86% was noted at $\frac{\lambda}{\mu_{dist}} = 15$ and $\frac{V_{curr}}{V_{ASC}} = 1$.

Traditional 3-Opt. With Traditional 3-Opt we observe significant savings variability ranging from 0% to 30% savings. This trait can be observed in the overall simulation plot, Figure 4-12. Extended 2-Opt, in contrast, displays variability in both the savings and computation time directions.

To further assist our analysis we chose to construct a plot demonstrating the convergence rates for Traditional 3-Opt and Extended 2-Opt local search methods. Using four starting tours and the previously mentioned current field, we have plotted the current best tour time against the iteration number. This is shown in Figure 4-19. It is clear that Extended 2-Opt has a faster convergence rate to its local minima than that of Traditional 3-Opt. Evidenced by Figure 4-18, the time required to compute one Traditional 3-Opt iteration is less than that of Extended 2-Opt. However, Extended 2-Opt arrives at its local minima using less iterations and the savings observed is frequently greater than that of Traditional 3-Opt. Figure 4-18 coupled with Figure 4-19 inform us that Traditional 3-Opt is not an ideal methodology due

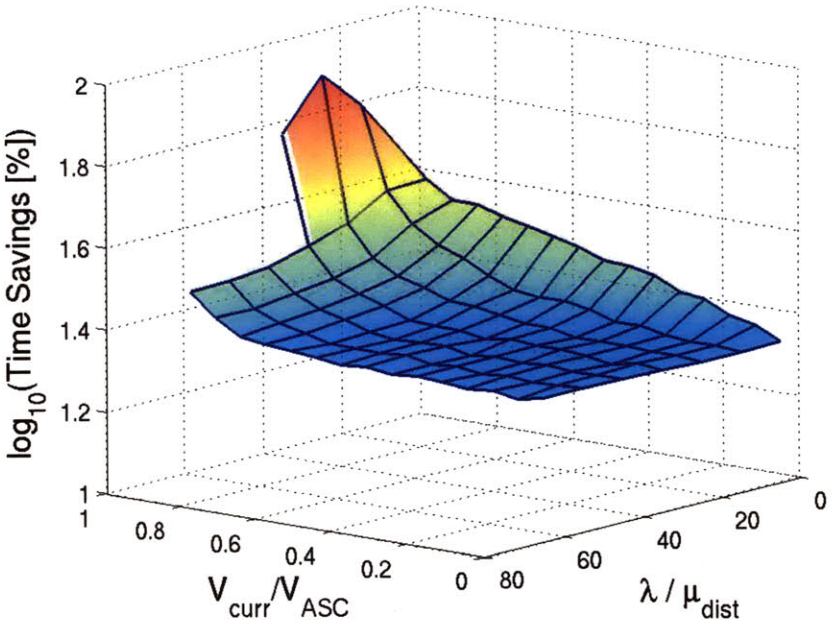


Figure 4-16: Extended 3-Opt Relative Mean Tour Savings
The minimum savings of 22% was found at $\frac{\lambda}{\mu_{dist}} = 35$ and $\frac{V_{curr}}{V_{ASC}} = 0.13$. The maximum savings of 78% was noted at $\frac{\lambda}{\mu_{dist}} = 25$ and $\frac{V_{curr}}{V_{ASC}} = 1$.

to its savings variability and unimpressive performance compared to both Extended 2-Opt and Extended 3-Opt. Instead, it is a trade-off between computation time and savings guarantees. Extended 2-Opt exhibits savings variability alongside minimal computation time. Extended 3-Opt, in contrast, guarantees greater savings with ten times as much computation time in this instance.

TSP Local Search Worst Case Iteration Analysis

It is important for us to note the worst case iteration scenario for each of the local search techniques. Starting with Extended 2-Opt and a tour consisting of m nodes, a single iteration chooses the first arc. The algorithm then iterates through all other subsequent arcs in the current tour to find an improvement. This leads to a worst case iteration of $(m-3)$ exchanges with the first arc and tour cost computations. Comparatively, during a single iteration in Traditional 3-Opt, a worst case iteration results in $(m-4)(m-5)$ exchanges. As an example, Table 4.3 presents

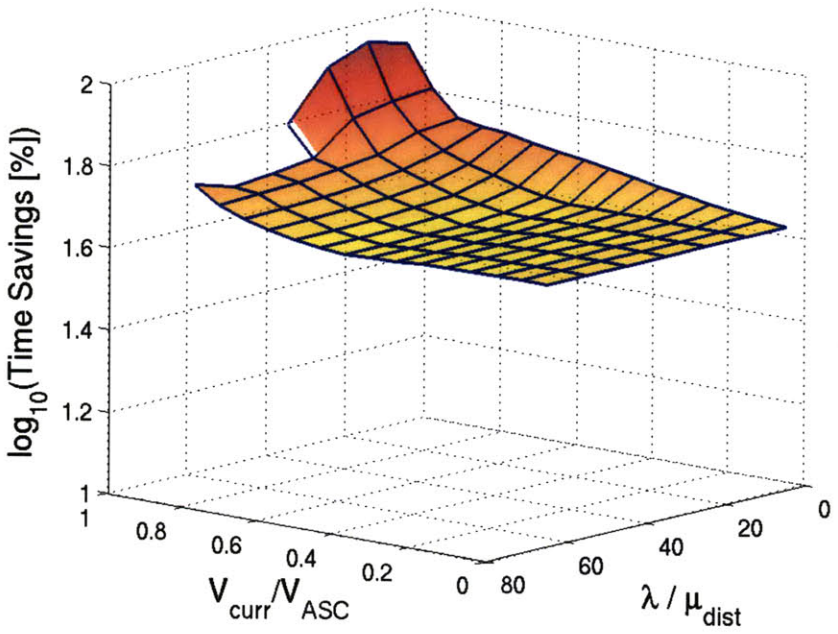


Figure 4-17: Extended 3-Opt Relative Max Tour Savings
The minimum savings of 42% was found at $\frac{\lambda}{\mu_{dist}} = 55$ and $\frac{V_{curr}}{V_{ASC}} = 0.53$. The maximum savings of 89% was noted at $\frac{\lambda}{\mu_{dist}} = 15$ and $\frac{V_{curr}}{V_{ASC}} = 1$.

all the possible arc combinations given the first arc is (1,2) and the initial tour $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ for Traditional 3-Opt. In this example, a maximum of $(m-4)(m-5) = (10-4)(10-5) = 30$ exchanges and cost computations occur in a single iteration. Extending Traditional 3-Opt to Extended 3-Opt, we find that a maximum of $7(m-4)(m-5)$ exchanges are computed due to the fact that it checks those tours that are reversed.

Using the derivations presented above, for each new tour analyzed we find the maximum number of iterations in increasing order to be the following: Extended 2-Opt - $m(m-3)$, Traditional 3-Opt - $m(m-4)(m-5)$, and Extended 3-Opt - $7m(m-4)(m-5)$. These results are summarized in Table 4.4.

4.5.3 Time-Optimal Global Path Analysis Summary

By defining the time-optimal global path problem as a TSP we are able to obtain near-optimal tours for a set of waypoints. In each instance we build a cost matrix

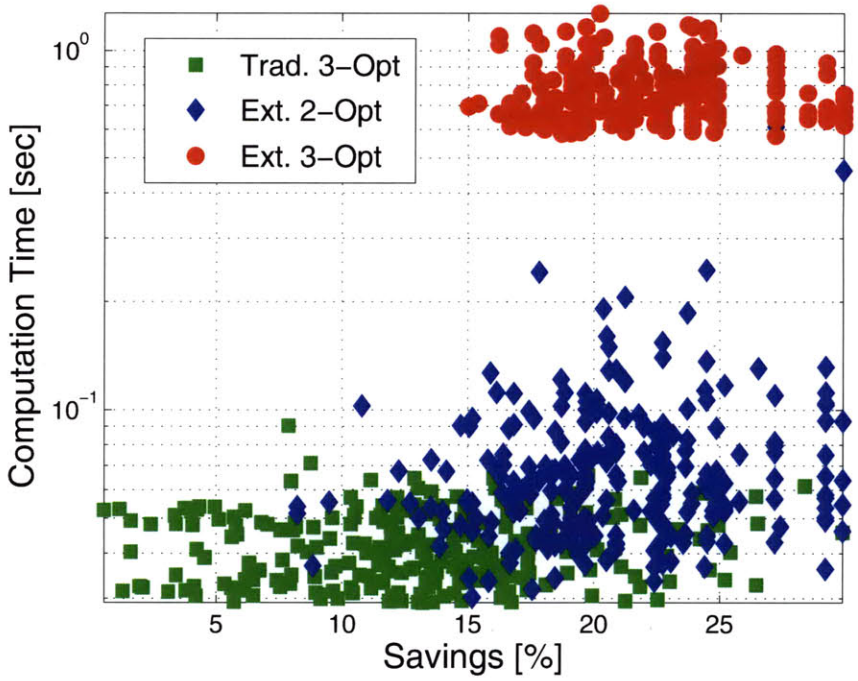


Figure 4-18: Computation Time vs. Savings
Using $\frac{\lambda}{\mu_{dist}} = 25$ and $\frac{V_{curr}}{V_{ASC}} = 0.47$ we have plotted all instances relating computation time to savings. The range of savings for Traditional 3-Opt is much larger than that of Extended 2-Opt or Extended 3-Opt. Also, it is evident that Extended 3-Opt requires more time than Extended 2-Opt but has a greater opportunity for increased savings.

then use the Nearest-Neighbor algorithm to construct an initial tour. TSP heuristic methods are used to obtain improved tours. The heuristic methods range in their abilities as shown in previous sections. Traditional 3-Opt does not achieve the level of savings as that of Extended 2-Opt and Extended 3-Opt as expected due to the limited number of configurations. The savings standard deviation is also quite large in comparison. This leads to the variable behavior shown in Figure 4-12. However, increased savings leads to increased computation time as evidenced by the Extended 2-Opt and Extended 3-Opt results. Though each achieve similar savings, Extended 2-Opt requires less time with noted variability. The Extended 3-Opt method presents the most improved tours and the least amount of variability thus making it ideal, however computation time is increased.

Choosing between the various heuristic methods then becomes a trade-off. Due

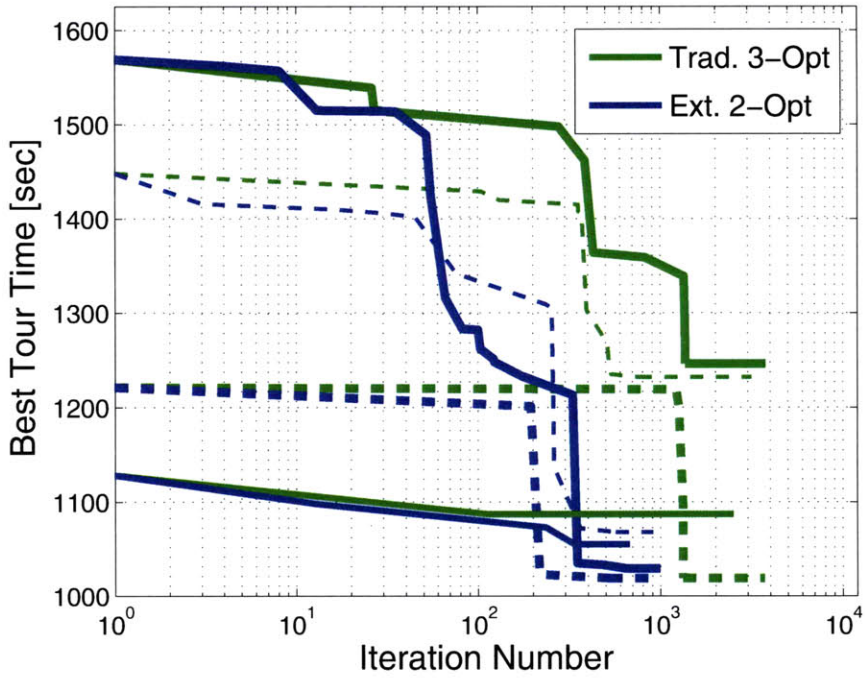


Figure 4-19: Best Tour vs. Iteration Number
Using a set current profile, we explore the convergence rates between Traditional 3-Opt and Extended 2-Opt. Upon observation, it is evident that Extended 2-Opt reaches its local minima in less iterations than that of Traditional 3-Opt. Also, the level of savings achieved by Traditional 3-Opt is frequently less than that of Extended 2-Opt making Extended 2-Opt a better local search heuristic choice.

to its variability, Traditional 3-Opt is not considered. It is suggested that when maximum savings is desired with a sacrifice of computation time, Extended 3-Opt should be the chosen heuristic. If computation time is of great importance with the potential for savings loss, Extended 2-Opt is suggested. In most real-time applications, such as mission planning in Singapore harbor, minimized computation time is desirable making Extended 2-Opt the chosen heuristic.

Table 4.3: Traditional 3-Opt Possible Arc Exchanges
When given an initial tour $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, there exist 30 possible exchanges to be evaluated for every given initial arc and are presented below.

1st Arc	2nd Arc	Possible 3rd Arc				
(1,2)	(3,4)	(5,6)	(6,7)	(7,8)	(8,9)	(9,10)
(1,2)	(4,5)	(6,7)	(7,8)	(8,9)	(9,10)	
(1,2)	(5,6)	(7,8)	(8,9)	(9,10)	(3,4)	
(1,2)	(6,7)	(8,9)	(9,10)	(3,4)	(4,5)	
(1,2)	(7,8)	(9,10)	(3,4)	(4,5)	(5,6)	
(1,2)	(8,9)	(3,4)	(4,5)	(5,6)	(6,7)	
(1,2)	(9,10)	(3,4)	(4,5)	(5,6)	(6,7)	(7,8)

Table 4.4: TSP Local Search - Maximum Iteration Number
Each local search procedure utilizes tour updating to obtain the resulting near-optimal solution. Upon every tour update there is a worst-case scenario where every arc combination must be analyzed. These values are presented below with the fastest being Extended 2-Opt and the slowest Extended 3-Opt.

Maximum Number of Iterations Possible		
Extended 2-Opt	Traditional 3-Opt	Extended 3-Opt
$m(m-3)$	$m(m-4)(m-5)$	$7m(m-4)(m-5)$

Chapter 5

Conclusions and Future Work

We present several methods for minimizing current effects on ASC operations in Singapore harbor for the purpose of benefiting oceanographic data collection and exploration. As an improvement over the presently used *Waypoint-to-Waypoint* control, *Cross-Track Error Minimization* control forces an ASC to follow a straight-line path under the influence of currents. This low-level controller is powerful for missions with constrained trajectories. An additional low-level controller, *Time-Optimal*, was developed to maximize operation efficiency by obtaining the time-optimal path between waypoints. Using *Time-Optimal* control we were able to produce near-time-optimal paths through TSP formulation and heuristics leading to overall mission efficiency due to time and energy savings.

5.1 Summary

Chapter 2 presented a new *Cross-Track Error Minimization* controller. Using this new controller we are able force the ASC to follow a straight-line trajectory connecting two waypoints. It is an improvement over *Waypoint-to-Waypoint* control where the trajectory is not constrained. The controller uses PI control where the distance from the desired trajectory is used to update the ASC heading. The capabilities provided by *Cross-Track Error Minimization* control add another dimension to research operations in Singapore harbor. Utilizing this new controller researchers are able to

conduct experiments requiring specified trajectories such as subsea surveys.

A third low-level controller for ASCs was described and analyzed in Chapter 3. Here we designed a controller for the purposes of minimizing the energy and time required for the ASC to move from one waypoint to the next. Using Zermelo's problem described by Bryson and Ho [9] we obtain an boundary-value problem for the path under the assumptions that the current is time-invariant and that the vehicle moves at a constant velocity relative to water. Through function minimization we solve the boundary-value problem for a given current field to obtain the optimal initial heading, θ_0^*, and the corresponding minimized travel time, t^*. The resulting *Time-Optimal* control law enables us to minimize the time it takes to move from one point to the next without trajectory constraints.

A methodology for forming near-time-optimal global paths is presented in Chapter 4. Here we use results obtained in Chapter 3 to formulate the problem as a TSP. The TSP is then solved using Nearest-Neighbor tour construction and local search heuristics. The first heuristic, Traditional 3-Opt, rearranges the time-optimal graph arcs so that tours are not reversed. Due to its savings variability Traditional 3-Opt is not the best heuristic choice when compared to Extended 2-Opt and Extended 3-Opt. Extended 3-Opt iteratively rearranges three arcs to create new tours. This heuristic produces the best savings results but at the cost of computation time. Extended 2-Opt uses two arc exchanges to create new tours. Computation time is the cheapest but it does not produce the maximum savings. Since ASCs operate in real-time harbor environments, computation time must be minimized. Thus, Extended 2-Opt is the most practical choice for these applications.

5.2 Future Work

The final goal of this work is to create control mechanisms which are operational in Singapore harbor for the use of scientists and research staff. In order to achieve this goal a few ideas must be explored and improvements must be made.

One main assumption pertaining to this work deals with the assumption that

the currents are time-invariant. Only during small periods of time can we assume that the current is time-invariant. As the missions grow to involve more waypoints and the mean distance between the waypoints increases, the time-invariant current assumption becomes invalid. We hope to extend this work to those cases of time-varying currents and utilize similar methods such as Extended 2-Opt and Extended 3-Opt to explore the additional time dimension.

A second assumption assumes that the currents are known leading to incorrect trajectories when current forecasts are inaccurate. To utilize the work presented in this thesis we must ensure that the forecasts are sufficiently accurate. In situations where it is not accurate we must find a method to continue the tour in a time-minimal manner. This process could involve switching the controller to a non-time-optimal one or using heuristic techniques to explore the space in real-time.

Another improvement to this work involves computation time. First, construction of the time-optimal cost matrix, C_{T^*} is a computationally hefty task. Each cost in the matrix must be filled using times obtained through *Time-Optimal* control. *Time-Optimal* control relies on function minimization to compute the optimal initial heading from one waypoint to the next. During this process it simulates the vehicle's movement till t_{ref} and computes the distance to the next waypoint over time. Effectively reducing t_{ref} would lead to faster computation times for the optimal time from waypoint i to waypoint j, t^*_{ij}. As mentioned within Chapter 3, t_{ref} can be obtained from any non-time-optimal control law. In our research we utilized the *Waypoint-to-Waypoint* control to determine t_{ref} due to the fact that it is not vehicle dependent. However, if the vehicle gains are correctly tuned and known, *Cross-Track Error Minimization* can be used in place of *Waypoint-to-Waypoint* control. This process would effectively lead to a smaller t_{ref} and faster computations times for each t^*_{ij} instance.

Other options for computation time minimization must be explored. In several instances presented, the *Time-Optimal* control path between two waypoints was comparable to that of *Cross-Track Error Minimization* control. This observation can go a long way when dealing with computation time factors. Specifically if we are able to derive a correlation between a current field, vehicle velocity over water, *Time-Optimal*

control paths, and *Cross-Track Error Minimization* control paths which better understands the convergence of the two paths we will save computation time on a number of fronts. As an example, building the time-optimal cost matrix, C_{T^*}, would be reduced under convergence because function minimization would no longer be necessary.

With regards to TSP formulation and local search, computation time could also be saved. Instead of computing t^*_{ij} values initially, we can compute then as they are needed. For instance, instead of utilizing Nearest-Neighbor to construct a tour using C^*_T we can instead construct a tour using the distance cost matrix, C_{dist}. From here we can compute arc weights as they are needed during local search exchanges.

Although several assumptions and computation time minimization require further investigation, we understand that this work contributes to the development of the field and the continuous study in Singapore harbor. Through two new control laws, *Cross-Track Error Minimization* and *Time-Optimal*, and time-optimal ASC path planning we are able to provide flexibility to research missions and able to maximize efficiency, a contribution of great importance as operations continue to expand.

Bibliography

[1] Security overview. *Singapore Defence & Security Report*, pages 36–40, 2010.

[2] A. Aguiar and J. Hespanha. Position tracking of underactuated vehicles. In *Proceedings of the American Control Conference.*

[3] A. Alvarez and A. Caiti. A genetic algorithm for autonomous underwater vehicle route planning in ocean environments with complex space-time variability. In *Proceedings IFAC Conference for Control Applications in Marine Systems*, 2001.

[4] A. Alvarez, A. Caiti, and R. Onken. Evolutionary path planning for autonomous underwater vehicles in variable ocean. *IEEE Journal of Oceanographic Engineering*, 29:418–429, April 2004.

[5] D.L. Applegate, R.E. Bixby, V. Chátal, and W.J. Cook. Implementing the dantzig-fulkerson-johnson algorithm for large scale traveling salesman problems. *Mathematical Programming*, 97:91–153, 2003.

[6] D.L. Applegate, R.E. Bixby, V. Chátal, and W.J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, Princeton, 2006.

[7] D.L. Applegate, R.E. Bixby, V. Chátal, and W.J. Cook. Certification of an optimal tsp tour through 85900 cities. *Operations Research Letters*, 37:11–15, 2009.

[8] J.L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4:387–411, 1992.

[9] A. Bryson and Y. Ho. *Applied Optimal Control: Optimization, Estimation, and Control.* Hemisphere Publishing Corporation, 1975.

[10] F. Buckley and M. Lewinter. *A Friendly Introduction to Graph Theory.* Pearson Education, Inc., 2003.

[11] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Report no. 388, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[12] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, pages 791–812, 1958.

[13] K.D. Do, Z.P. Jiang, and J. Pan. Underactuated ship global tracking under relaxed conditions. In *IEEE Transactions on Automatic Control*, 2002.

[14] K.D. Do, Z.P. Jiang, and J. Pan. Robust adaptive path following of underactuated ships. *Automatica*, 40:929–944, 2004.

[15] K.D. Do and J. Pan. Global tracking control of underactuated ships with nonzero off-diagonal terms in their system matrices. *Automatica*, 41:87–95, 2005.

[16] K.D. Do and J. Pan. Global robust adaptive path following of underactuated ships. *Automatica*, 42:1713–1722, 2006.

[17] M.M. Flood. The traveling-salesman problem. *Operations Research*, 1956.

[18] Center for Environmental Sensing and Monitoring (CENSAM). http://censam.mit.edu, 2008.

[19] SMART: Singapore-MIT Alliance for Research & Technology. http://web.mit.edu/smart, 2010.

[20] T. Fossen. *Guidance and Control of Ocean Vehicles.* John Wiley & Sons Ltd., 1994.

[21] T. Fossen. *Marine Control Systems: Guidance, Navigation, and Control of Ships, Rigs and Underwater Vehicles.* Marine Cybernetics AS, 2002.

[22] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *IEEE International Conference on Evolutionary Computation*, 1996.

[23] B. Garau, A. Alvarez, and G. Oliver. Path planning of autonomous underwater vehicles in current fields with complex spacial variability: an a* approach. In *IEEE International Conference on Robotics and Automation*, 2005.

[24] J. Gordon, P. Lee, and H. Lucas Jr. A resource-based view of competitive advantage at the port of singapore. *Journal of Strategic Information Systems*, 14:69–86, 2005.

[25] G. Gutin and A. Punnen, editors. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[26] M. Hahsler and K Hornik. Tsp - infrastructure for the traveling salesperson problem. *Journal for Statistical Software*, 23, December 2007.

[27] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[28] C. Hocaolglu and A.C. Sanderson. Planning multi-paths using speciation in genetic algorithms. In *IEEE International Conference for Evolutionary Computation*, pages 378–383, 1997.

[29] T. Holzhüter. A high precision track controller for ships. *Preprints of the 11th IFAC World Congress*, pages 118–123.

[30] M. Jardin. Neighboring optimal aircraft guidance in winds. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000.

[31] D. Johnson. *Automata, Languages and Programming*, chapter Local Optimization and the Traveling Salesman Problem. Springer Berline / Heidelberg, 1990.

[32] R. Jonker and T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2:161–163, 1983.

[33] P. Kanellakis and C. Papadimitriou. Local search for the asymmetric traveling salesman problem. *Operations Research*, 28:1086–1099, 1980.

[34] K. Kim and T. Ura. Optimal guidance for autonomous underwater vehicle navigation within undersea areas of current disturbances. *Advanced Robotics*, 23:601–628, 2009.

[35] B. Korte and J. Vygen. *Cominatorial Optimization: Theory and Algorithms.* Springer-Verlag Berlin Heidelberg, 2006.

[36] G. Laporte. A concise guide to the traveling salesman problem. *Journal of the Operational Research Society*, 61:35–40, 2010.

[37] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Cominatorial Optimization.* John Wiley & Sons Ltd., 1985.

[38] S. Lin. Computer solutions of the traveling-salesman problem. *Bell System Technology Journal*, 44:2245–2269.

[39] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[40] The Maritime and Port Authority of Singapore (MPA). http://www.mpa.gov.sg, 2009.

[41] C. Nilsson. Heuristics for the traveling salesman problem. Technical report, Linköping University, 2003.

[42] N.J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Company, Palo Alto, CA, 1980.

[43] K. Ogata. *System Dynamics.* Pearson Education, Inc., 4 edition, 2004.

[44] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, J. Evans, and D. Lane. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 2007.

[45] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, and D. Lane. Underwater path planning using fast marching algorithms. In *Oceans 2005 Europe International Conference*, pages 814–819, 2005.

[46] A. Punnen, F. Margot, and S. Kabadi. Tsp heuristics: Domination analysis and complexity. *Algorithmica*, 35:111–127, 2003.

[47] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications.* Springer-Verlag, Berlin, 1994.

[48] F. Repoulias and E. Papadopoulos. Trajectory planning and tracking control of underactuated auvs. In *IEEE International Conference on Robotics and Automation*, 2005.

[49] F. Repoulias and E. Papadopoulos. Planar trajectory planning and tracking control design for underactuated auvs. *Ocean Engineering*, 34:1650–1667, 2007.

[50] C. Ribeiro and P. Hansen, editors. *Essays and Surveys in Metaheuristics.* Kluwer Academic Publishers, 2002.

[51] D. Rosenkrantz, R. Stearns, and P. Lewis II. *Fundamental Problems in Computing*, chapter An Analysis of Several Heuristics for the Traveling Salesman Problem. Springer Netherlands, 2009.

[52] M. Soulignac, P. Taillibert, and M. Rueher. Adaptings the wavefront expansion in presence of strong currents. In *Proceedings of International Conference on Robotics and Automation*, pages 1352–1358, 2008.

[53] M. Soulignac, P. Taillibert, and M. Rueher. Time-minimal path planning in dynamic current fields. In *IEEE International Conference on Robotics and Automation*, 2009.

[54] T. Stützle and H. Hoos. *MAX-MIN* ant system and local search for the traveling salesman problem. In *IEEE International Conference on Evolutionary Computation*, 1997.

[55] K. Sugihara and J. Yuh. Ga-based motion planning for underwater robotic vehicle. In *10th International Symposium of Unmanned Untethered Submersible Technology*, pages 406–415, 1997.

[56] Mission Oriented Operating Suite. http://oceanai.mit.edu/moosivp/home.html.

[57] Robot Marine Systems. http://www.maribotics.com/index.html.

[58] Tropical Marine Science Institute (TMSI). http://www.tmsi.nus.edu.sg, 2007.