# Assigning Closely Spaced Targets to Multiple Autonomous Underwater Vehicles

by

Beverley Chow

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

This research addresses the problem of allocating closely spaced targets to multiple autonomous underwater vehicles (AUV) in the presence of constant ocean currents. The main difficulty of this problem is that the non-holonomic vehicles are constrained to move along forward paths with bounded curvatures. The Dubins model is a simple but effective way to handle the kinematic characteristics of AUVs. It gives complete characterization of the optimal paths between two configurations for a vehicle with limited turning radius moving in a plane at constant speed.

In the proposed algorithm, Dubins paths are modified to include ocean currents, resulting in paths defined by curves whose radius of curvature is not constant. To determine the time required to follow such paths, an approximate dynamic model of the AUV is queried due to the computational complexity of the full model. The lower order model is built from data obtained from sampling the full model. The full model is used in evaluating the final tour times of the sequences generated by the proposed algorithm to validate the results.

The proposed algorithm solves the task allocation problem with market-based auctions that minimize the total travel time to complete the mission. The novelty of the research is the path cost calculation that combines a Dubins model, an AUV dynamic model, and a model of the ocean current. Simulations were conducted in Matlab to illustrate the performance of the proposed algorithm using various number of task points and AUVs. The task points were generated randomly and uniformly close together to highlight the necessity for considering the curvature constraints.

For a sufficiently dense set of points, it becomes clear that the ordering of the Euclidean tours are not optimal in the case of the Dubins multiple travelling salesmen problem. This is due to the fact that there is little relationship between the Euclidean and Dubins metrics, especially when the Euclidean distances are small with respect to the turning radius. An algorithm for the Euclidean problem will tend to schedule very close points in a successive order, which can imply long maneuvers for the AUV. This is clearly demonstrated by the numerous loops that become problematic with dense sets of points. The algorithm proposed in this thesis does not rely on the Euclidean solution and therefore, even in the presence of ocean currents, can create paths that are feasible for curvature bound vehicles.

Field tests were also conducted on an Iver2 AUV at the Avila Pier in California to validate the performance of the proposed algorithm in real world environments. Missions created based on the sequences generated by the proposed algorithm were conducted to observe the ability of an AUV to follow paths of bounded curvature in the presence of ocean currents. Results show that the proposed algorithm generated paths that were feasible for an AUV to track closely, even in the presence of ocean current.

**Acknowledgements**

First of all, I would like to thank my supervisors, Professor Chris M. Clark and Professor Jan P. Huissoon, for their guidance, inspiration, and understanding during the course of this project. I am grateful for the opportunity to work on such an interesting problem. I would also like to thank Professor Steven Waslander and Professor Hyock Ju Kwon for agreeing to read this thesis.

A special thanks to California Polytechnic State University for their research facilities and the use of the Iver2 AUV. I would like to thank Tom Moylan, Jason Felton, and Ian Robbins at the Center for Coastal Marine Sciences for their assistance with field testing.

Finally, I would like to thank my peers, who through collaboration and thoughtful discussions, have provided me with valuable advice and feedback along the way.

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# Chapter 1

# Introduction

Autonomous Underwater Vehicles (AUVs) have been used successfully in the past to solve geological, biological, chemical, and physical oceanographic problems. This has resulted in a variety of scientific and commercial AUVs to be designed, built, and deployed. With the increasing feasibility and decreasing expense of AUVs, interest in using them for ocean sampling, mapping, surveillance, and communication is growing and multi-AUV operations are beginning to be realized in the water. As with any multi-robot system, a challenge is to determine which robot should perform which task in order to cooperatively achieve the global goal in an optimal manner.

## 1.1 Autonomous Underwater Vehicles

An AUV is a robot which travels through the water by a propulsion system controlled by an on-board computer. Some of the first AUVs were developed by the Applied Physics Laboratory at the University of Washington as early as 1957. They are maneuverable in three dimensions which allows them to follow precise pre-programmed trajectories under most environmental conditions. Sensors on-board the AUV sample the ocean as the AUV moves through it, providing the ability to make both spatial and time series measurements. With the developments in artificial intelligence, control theory, and computer hardware, AUVs can operate completely autonomously which gives them the capability of accessing previously inaccessible ocean locations.

Traditional methods for oceanographic research include using ship based measurements and moorings for sampling the ocean. Over the past two decades, alternative technologies such as subsurface floats, remotely operated vehicles (ROVs) and AUVs have emerged to complement the existing sensing techniques. AUVs have the potential for large cost-savings compared to current ocean sampling technologies, especially for sustained real-time measurements [1]. With the development of solar-powered AUVs, vehicles are capable of operating unattended for months in

open ocean areas, periodically relaying data by satellite to shore, before returning to be picked up.

Some of the applications of AUVs include:

- *Commercial*: The oil and gas industry uses AUVs to make detailed maps of the sea-floor before they start building sub-sea infrastructure. Other applications include inspection of the underwater engineering structures and pipelines.
- *Military*: A typical military mission for an AUV is to map an area to determine if there are any mines, or to monitor a protected area (such as a harbour) for new unidentified objects. AUVs are also employed in anti-submarine warfare to aid in the detection of manned submarines.
- *Research*: Scientists use AUVs for ocean exploration, monitoring of water medium, and marine geological surveys. A variety of sensors can be added to AUVs to measure the concentration of various elements or compounds, the absorption or reflection of light, and the presence of microscopic life.

When used for oceanographic research, AUVs carry sensors to navigate autonomously and map features of the ocean. Typical sensors include compasses, thermometers, conductivity and depth sensors, sidescan and other sonars, magnetometers, and light scattering sensors. There are mainly three modes of navigation for AUVs. Long baseline navigation is based on triangulation and requires the AUV to operate within a net of acoustic beacons. This is mainly used for larger area missions. When a surface reference such as a support ship is available, ultra-short baseline positioning can be used to calculate where the sub-sea vehicle is relative to the known position of the surface transponder by means of acoustic range and bearing measurements. When it is operating completely autonomously, the AUV can surface and take its own GPS measurement. Between GPS readings, dead reckoning is used where the vehicle relies on the compass heading, the bottom-tracking Doppler signals, and the inertial navigation system on-board to measure the rate of travel.

Recently, a new class of AUVs were introduced which mimic designs found in nature. These biomimetic (or bionic) vehicles are able to achieve higher degrees of efficiency in propulsion and maneuverability by copying successful designs in nature. The *AquaJelly* by Festo [2] is an artificial autonomous jellyfish with an electric drive unit and an intelligent adaptive mechanism for swarm behaviour. The *Bionik Manta* by Evo Logics [3] is a buoyancy-driven glider primarily used for environment monitoring, hydrographic surveying, seabed mapping, and search and recovery missions. The concept of using a buoyancy engine instead of a propeller was first introduced by Webb [4]. A buoyancy engine works by alternating between floating and sinking, and uses wings to glide horizontally, both on the way up and on the way down. The advantage of such systems is that it could be deployed for weeks, months, and eventually years at a time [5].

## 1.2 Multiple Autonomous Underwater Vehicles

Coordinating groups of AUVs can provide significant benefits to a number of applications including ocean sampling, mapping, surveillance and communication. Multiple AUVs can acquire oceanographic data and information in spatial and temporal resolution far exceeding the capabilities of a single AUV.

The advantages to using multi-AUVs are:

- A single AUV cannot perform some tasks alone so a team is required for successful execution.
- A team can accomplish a given task more quickly than a single AUV can by dividing the task into sub-tasks and executing them concurrently.
- A team can make effective use of specialists designed for a single purpose, rather than requiring that a single AUV be capable of performing all tasks.
- A team of AUVs can localize themselves more efficiently if they exchange information about their position whenever they sense each other.
- A team of AUVs generally provides a more robust solution by introducing redundancy. Single point of failure can be avoided when there are multiple vehicles with the same capabilities.
- A team of AUVs can handle simultaneous measurements of spatially disparate phenomena.

Most current multi-AUV systems are capable of relatively simple missions involving a few, usually homogeneous, AUVs. The requirements of a multi-AUV system will differ depending on the application. When each vehicle is equipped with sensors for observing its environment, the group can sample the physical and/or biological variables in the water. Sampling could be performed in a relatively large area to observe large-scale processes (e.g. upwelling and relaxation), often referred to as *broad-area coverage*, or used to observe local phenomena such as fronts, plumes, eddies, and algae blooms, often referred to as *feature tracking*. For broad-area coverage, the requirements of the system will be vehicle endurance while vehicle speed may be of interest for feature tracking. Also, vehicle-to-vehicle communication may be impractical for broad-area coverage, but may be feasible for feature tracking. However, for all applications, coordination is central to the effective behaviour of the robot team.

Ideally, robots will coordinate and share resources amongst themselves to accomplish their mission efficiently and reliably. Coordination can lead to faster task completion, increased robustness, higher quality solutions, and the completion of tasks impossible for single robots [6]. One of the challenges with coordinating a team of robots is the problem of task allocation which is the focus of this thesis.

Using a team of AUVs to make measurements requires an algorithm for routing the AUVs to maximize the information they provide.

This thesis addresses a specific type of problem where $n$ vehicles are required to visit $m$ task points. The motion of the AUV satisfies a non-holonomic constraint (i.e. the yaw rate of the vehicle is bounded) which makes the costs of going from one point to another non-Euclidean and asymmetric. Each task point is to be visited by one and only one vehicle. Given a set of task points and the yaw rate constraints on the vehicles, the problem is to assign each vehicle a sequence of task points to visit and to find a feasible path for each vehicle to follow so that the vehicle passes through the assigned task points. Each task point is a subgoal that is necessary for achieving the overall goal of the system that can be achieved independently of other subgoals. Task independence is assumed, where individual task points can be considered and assigned independently of each other without ordering constraints. The objective function is to minimize the total time to visit all of the task points.

This thesis presents an approximate algorithm for the task allocation problem where vehicles are constrained to move forward along paths with bounded curvatures. The problem has been simplified to limit the robots to operate on the Euclidean plane. Intentional cooperation is used where robots cooperate explicitly and with purpose through task negotiations.

The motivation behind this thesis stems from the need to develop task allocation algorithms for AUVs to operate in real-world environments. An example of one particular application is the use of multiple AUVs to investigate a possible garbage dump site in Avila Bay, California. A recent coarse sonar image was used to identify several closely spaced points of interest. Multiple higher-resolution images taken from closely spaced locations are required to identify the foreign objects.

Most existing algorithms do not consider the effect of ocean currents which is inevitable in the ocean. In the presence of currents, the trajectories of AUVs may be significantly different from the desired path. Keeping the vehicle on the desired path is critical because measurements must be taken in the right place. AUVs may be aided by the currents at certain times and hindered at other times. Therefore, a task allocation algorithm that can use the ocean current to aid the vehicle in the direction of travel so that the vehicle can stay on the desired path is needed.

## 1.3 Contributions

The features that differentiate this research to similar problems previously studied are:

- *Kinematic constraints on the vehicle*: The non-holonomic constraints of the AUV causes difficulties when task points are close together. The Dubins model [7] is a simple but efficient way to handle the kinematic characteristics of AUVs. It gives complete characterization of the optimal paths between

two configurations for a vehicle with limited turning radius moving in a plane at constant speed.

- *Presence of a constant ocean current*: Dubins paths are modified to include ocean currents, resulting in paths defined by curves whose radius of curvature is not constant. To determine the time required to follow such paths, an approximate dynamic model of the AUV is queried. Specifically, a lower order model of the REMUS AUV from [8] is used so that the computational complexity is reduced.

## 1.4 Thesis Outline

The rest of the report is organized as follows. Chapter 2 gives an overview of the task allocation problem and describes various other techniques that have been used to solve related problems. Chapter 3 begins with the formal problem definition. Chapter 4 describes the two AUV platforms that were used in validating the proposed algorithm, the REMUS AUV used for simulations and the Iver2 AUV used in real-world field tests. In Chapter 5, an overview of the proposed algorithm is introduced with the details presented in Chapter 6. Chapter 7 discusses the results from a simulation done in Matlab to verify that the desired results are achieved. Following a satisfactory simulation, the algorithm was tested in the field at the Avila Pier in California. The report concludes with a summary of results and future work in Chapter 8.

# Chapter 2

# Background

Several approaches have been applied to the general problem of allocating tasks between multiple robots in a team. Consider the problem of assigning target points to a team of robots that are co-operatively exploring an unknown environment. The goal of the task allocation problem is to have robots visit all targets while minimizing the total travel time or distance of the robots. When targets are known before the mission, it is possible to build a schedule of targets for each robot. Unfortunately, this problem is not straight forward because the cost for a robot to visit target $C$ depends on whether that robot first visits target $A$ or target $B$. This problem is an instance of the multiple traveling salesperson problem (MTSP), which has received considerable attention in combinatorial optimization. Even in the restricted case of one salesperson, the MTSP is strongly $NP$-hard [9]. This chapter begins with a review of research on the MTSP followed by methods used to solve related problems.

## 2.1 The Multiple Traveling Salesmen Problem

A generalization of the well-known traveling salesman problem (TSP) is the MTSP, which consists of determining a set of routes for $n$ salesmen who all start from and return back to a home city (depot). Although the TSP has received a great deal of attention, the research on the MTSP is limited. The purpose of this section is to review the existing literature on the MTSP, with an emphasis on practical applications.

The MTSP can in general be defined as follows: Given a set of nodes, let there be $n$ salesmen located at a single depot node. The remaining nodes (cities) that are to be visited are called intermediate nodes. Then, the MTSP consists of finding tours for all $n$ salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized. The cost metric can be defined in terms of distance, time, etc. Possible variations of the problem are as follows:

- *Single vs. multiple depots*: In the single depot case, all salesmen start from and end their tours at a single point. On the other hand, if there exist multiple depots with a number of salesman located at each, the salesmen can either return to their original depot after completing their tour or return to any depot with the restriction that the initial number of salesmen at each depot remains the same after all the travel. The former is referred as the *fixed destination* case whereas the latter is named as the *nonfixed destination* case.

- *Number of salesmen*: The number of salesman in the problem may be a bounded variable or fixed a priori.

- *Fixed charges*: When the number of salesmen in the problem is not fixed, then each salesman usually has an associated fixed cost incurring whenever this salesman is used in the solution. In this case, the minimization of the number of salesman to be activated in the solution may also be of concern.

- *Time windows*: In this variation, certain nodes need to be visited in specific time periods, named as *time windows*. This is an important extension of the MTSP and referred to as the multiple traveling salesman problem with time windows. The MTSP with time windows has immediate applications in school bus, ship and airline scheduling problems.

- *Other special restrictions*: These restrictions may consist of bounds on the number of nodes each salesman visits, the maximum or minimum distance a salesman travels or other special constraints.

The task allocation problem described in this thesis includes a number of variations from the standard MTSP:

- The constraints of the vehicle must be considered when calculating the path costs. While the standard MTSP assumes a fully connected graph, where the salesman can travel directly between any two cities, this thesis considers the vehicle's kinematics and dynamics. As a result, the path connecting two points are Dubins paths.

- The task allocation algorithm must take into account the effects of ocean current. This is particularly significant when task points are closely spaced since some points may not be reachable by a vehicle with curvature constraints without long maneuvers.

Given the difficulty of the MTSP, researchers have not attempted to solve it directly or exactly. The majority of research has been experimental in nature, where researchers construct a multi-robot task allocation architecture, and then validate it in one or more application domains [10]. This proof-of-concept method has led to many proposals, each of which has been experimentally validated in simulation and/or with physical robots. The results of these research will be described in the next section.

## 2.2 Literature Review

Compared to the TSP, the MTSP is more adequate to model real life situations since it is capable of handling more than one salesman. These situations arise mostly in various routing and scheduling problems (e.g. print press schedule, crew scheduling, bus routing problem, interview scheduling, hot rolling scheduling). This thesis is an instance of the mission planning problem. Mission planning generally arises in the context of autonomous mobile robots, where a variety of applications include construction, military reconnaissance, warehouse automation, post-office automation and planetary exploration. The mission plan consists of determining the optimal path for each robot to accomplish the goals of the mission in the smallest possible time. The mission planner uses a variation of the MTSP where there are $n$ robots, $m$ goals which must be visited by some robot, and a base city to which all robots must eventually return. The application of the MTSP in mission planning was reported by Brummit and Stentz [11] for robots operating in unstructured environments using a dynamic planning technique. Planning of autonomous robots was modelled as a variant of the MTSP by Yu et al. [12] in the field of cooperative robotics and solved using Genetic Algorithm. Similarly, the routing problems arising in the planning of unmanned aerial vehicle applications, as investigated by Ryan et al. [13], was modelled as an MTSP with time windows and solved using a Reactive Tabu Search.

The problem of allocating tasks between multiple robots in a team has been studied extensively; refer to [10] for a survey of these. Heuristic methods are typically used since optimizing the performance is often computationally intractable. One of the earliest studies on social interactions among robots was conducted by Matarić [14] whose work included following, dispersion, aggregation, homing, and flocking. A unique feature of Matarić's work was the use of animal models to develop very simple algorithms for these behaviours. The Autonomous Robot Architecture developed in the Mobile Robot Laboratory at Georgia Institute of Technology is a hybrid reactive-deliberative architecture which is the basis for MissionLab [15]. MissionLab allows users to specify missions and automatically configures the software needed for behaviour-based robot control. Parker's ALLIANCE architecture [16] is a fault-tolerant, adaptive, distributed, behaviour-based software system for control of robot teams. It has no centralized control so all robots are fully autonomous and has the ability to perform useful actions even in the presence of failures of other robots. Also, the robots on the team can detect, with some finite probability, the effects of their own actions and those of other members of the team. This is accomplished with the presence of sensors and feedback control on the robot, as well as an explicit communication mechanism between robots.

Another frequently used method is based on market mechanisms, such as auctions, which have been demonstrated to be fast and robust on real robots. In market-based multi-robot systems, robots are designed as self-interested agents and the team of robots are modeled as an economy. The goal of the team is to complete the tasks successfully while minimizing its individual cost and maximizing

its individual profit. Determining the revenues and costs is the key to successful implementation of a market-based approach. A function is needed to map possible task outcomes onto revenue values. Another function is needed to map possible schemes for performing the task onto cost values. As a team, the goal is to execute the mission such that the profit, which is the revenue minus the cost, is maximized. The cost and revenue functions could potentially be complex functions and should be designed to reflect the nature of the application domain. Market-based approaches distribute planning required for task allocation through the auction process where each robot locally compute its cost, and encapsulates the costs in its bids. Auction-based methods balance the trade-off between purely centralized coordination methods which require a central controller and purely decentralized coordination methods without any communication between agents, both in terms of communication efficiency, computation efficiency, and the quality of the solution [6].

The concept of using an economic model to allocate tasks between agents was first proposed by Smith [17] in a system called *Contract Net*. M+ [18] is a distributed task allocation and achievement scheme for multi-robot cooperation, addressing many real time issues including plan merging paradigms. Gerkey and Matarić [19] presented MURDOCH which is a framework achieving a complete distributed, resource-centric, publisher/subscriber type allocation for instantaneous assignment. Zlot et al. [20] used TSP heuristics to build target schedules and derived costs that were used in Dias and Stentz's [21] market-based task allocation architecture. Dias and Stentz [22] later proposed TraderBots, a combinatorial auction based task allocation scheme. Lemarie et al. [23] proposed a task allocation scheme for multi-UAV cooperation with balanced workloads of robots .

Specific work for AUVs include the work by Sariel et al. [24] whose framework is a market-based mechanism capable of re-planning task distributions, re-decomposing tasks, rescheduling commitments, and re-planning coordination during execution. The Orca Project by Turner [25] focused on intelligent mission control for AUVs and inter-agent communication. The CoDA Project by Turner and Turner [26] focused on cooperation and used a type of constraint-based reasoning called *constrained heuristic search* for their task assignment mechanism. Their method combines heuristic search with constraint satisfaction problem techniques, deciding what to do at any point using heuristics based on the topology and other properties of the constraint network.

Jeyaraman et al. [27] proposed a hybrid control scheme for a decentralised, autonomous group of vehicles with bounded curvatures. They constrained their vehicles to move in a fixed rectangular area without allowing vehicles to cross paths. Also, they only considered the case where the Euclidean distance between task points is at least four times the turning radius. The work by Rathinam [28] also put a constraint on the minimum distance between the task points to be at least two times the minimum turning radius of the vehicle. Note that this thesis does not assume the minimum distance between any two task points and there are no restrictions on the paths of different vehicles crossing.

One of the earliest examples of a full-scale, cooperative multiple-AUV demonstration in the water was described by Schultz et al. [29]. They conducted missions where the vehicle's route was determined not by a preset list of waypoints or external commands, but based on in-flight observed data. Another example was investigated by Davis et al. [30] for the planning for underwater gliders in the presence of significant currents. Their work addressed the problem of AUVs operating in environments with significant currents, similar to this thesis. However, vehicle dynamics are not accounted for in their routing strategy which this thesis aims to address.

# Chapter 3

# Problem Statement

This thesis considers the allocation of $m$ targets to $n$ vehicles. Given a set of vehicles $\{V_1, V_2, ..., V_n\}$ and targets $D = \{d_1, d_2, ..., d_m\}$, the problem is to assign a sequence of targets $S_i$ to each vehicle to visit and a path through the sequence $S_i$. The objective is to:

Minimize

$$C_{\text{total}} = \max_i C(S_i) \tag{3.1}$$

subject to

$$D = \bigsqcup_i S_i \qquad \text{(disjoint union)} \tag{3.2}$$

$$\left.\begin{aligned} \frac{dx_{i,t}}{dt} &= u_0 \cos(\psi_{i,t}) \\ \frac{dy_{i,t}}{dt} &= u_0 \sin(\psi_{i,t}) \\ \frac{d\psi_{i,t}}{dt} &= r, \qquad r \in [-\omega, +\omega] \end{aligned}\right\} \tag{3.3}$$

where $u_0$ denotes the nominal vehicle speed, $\psi_{i,t}$ the yaw of the vehicle, and $\omega$ represents the bound on the yaw rate. In (3.1), $C(S_i)$ is the time required for $V_i$ to complete its tour $S_i$. Note that (3.2) dictates all tasks to be visited and restricts each task to be assigned to only one vehicle and (3.3) considers the non-holonomic constraints of the vehicle.

# Chapter 4

# Overview of Planner

This thesis proposes an approximate algorithm for the task allocation problem which is not possible to solve in polynomial time. The problem combines the exponential complexity of integer assignment decisions with non-linear, non-convex differential equation constraints, making it a Mixed Integer Non-linear Program with exponential growth in computational time.

The proposed planner constructs feasible paths based on Dubins' model which characterizes the optimal path between two configurations of a vehicle with limited turning radius. The vehicles have been constrained to move in a plane at constant speed. To determine the time required to track these paths, a lower order dynamic model is queried to reduce the computational time. The following sections describe the three main stages of the proposed algorithm.

## 4.1 Clustering

Let $(x_j, y_j)$ denote the position of target $d_j$. Given the positions of $j = 1, ..., m$ task points, the algorithm starts by creating $n$ clusters of task points, where $n$ is equal to the number of AUVs. The method used in this thesis is *k-means* [31], which partitions the $m$ points into $n$ clusters by minimizing the total intra-cluster variance, or the squared error function. For this implementation, *k-means* is minimized with respect to the squared Euclidean distance with the initial cluster centroid positions selected uniformly at random from the range of $x$. The *k-means* algorithm is repeated 3 times, each with a new set of initial centroids. If a cluster loses all of its member observations during the iterative process, a new cluster consisting of the one observation furthest from its centroid is created.

After partitioning all task points into clusters, the centroid of all task points is calculated as:

$$x_{centroid} = \frac{\sum_{j=1}^{m} x_j}{m}; \quad y_{centroid} = \frac{\sum_{j=1}^{m} y_j}{m}. \tag{4.1}$$

For all $i = 1, 2, ..., n$ clusters, the three task points in each cluster $i$ that are farthest from the centroid are assigned to $i^{th}$ vehicle. The number of initial task assignments was chosen to be three because for three tasks forming a loop, the ordering of the tasks do not matter and always produces the same loop.

## 4.2 Auctioning

Once each vehicle has three task points assigned to it, the remaining $m - 3n$ tasks are auctioned via a sequence of first-price one-round auctions similar to work by Lagoudakis et al. [32]. The unassigned tasks are first ordered according to their distance from ($x_{centroid}$, $y_{centroid}$). The greater the distance from the centroid, the higher the priority the task will have in the order.

Following this order, each task is auctioned off. Each vehicle $i$ can bid on the task $j$, where the bid $B_i$ is equal to the cost of traveling a path that consists of all previously won tasks and the current task being auctioned. Each vehicle considers the insertion of the new task at every point in the current sequence $S_i = (s_1, s_2, s_3, ..., s_l)$ where $l$ is the number of previously won tasks by vehicle $i$. Each vehicle submits a bid as the lowest cost (i.e. time) to complete the new tour as:

$$B_i\left(d_j, S_i\right) = \min_{0 \leq k \leq l} C\left(s_1, ..., s_k, d_j, s_{k+1}, ..., s_l\right) \tag{4.2}$$

The $i^{th}$ vehicle with the lowest $B_i$ wins target $d_j$ and updates its sequence of targets with $S_i' = (s_1, ..., s_k, d_j, s_{k+1}, ..., s_l)$. The calculation of $C(S_i')$ is the key to this algorithm's ability to reduce cost and details are found in Chapter 5.

After a task auction is completed, the auctioning process continues with the next round of bidding until all tasks are allocated. The advantage of using this algorithm is that it allows for a decentralized implementation. The calculation of the bids can be performed locally by each vehicle in parallel. Additionally, each vehicle can maintain its own sequence of tasks and the costs associated with it.

## 4.3 Post Processing

After all tasks have been auctioned off, each robot has a sequence of tasks to visit. Due to the inherent shortcomings of market-based auction mechanisms, the cost of going backwards through the same sequence of tasks may produce a lower cost value. A post processing step is added at the end of the algorithm to check for the possibility that reversing the order of the task points will produce a lower cost.

# Chapter 5

# Path Cost Calculation

To determine the path cost for bidding as described above, the time $C(S)$ for the AUV to traverse the sequence needs to be calculated. First, the shortest path between two points is solved using a Dubins model to consider the kinematic constraints of the vehicle. Then, the time required to follow the path is calculated using a dynamic model of the vehicle. However, due to the complexity of a full dynamic model, it is not possible to query it in a reasonable amount of time. Therefore, a lower order model based on the full model is described in this chapter. Note that both the kinematic and dynamic models are modified to consider the effects of ocean current.

## 5.1 Dubins Path

In order to calculate the time required to travel between two points, the Dubins shortest path problem must first be solved. Dubins' original work [7] derived conditions that characterize the optimal path between two points when both initial and terminal orientations were specified and his work has been widely studied in path planning [33]. Sussmann and Tang [34] rederived Dubins' results as an application of *Pontryagin's Maximum Principle*. Dubins' result shows that, given any two points, the shortest path that considers the constraints expressed in Eq. 3.3 consists of exactly three path segments consisting of a combination of a straight line segment and maximum curvature arcs.

Graphically, the algorithm starts by drawing two maximum curvature circles that are tangential to the initial state vector and two maximum curvature circles that are tangential to the terminal state vector. Dubins' result indicates that the optimal trajectory selects an arc on one of the two initial circles, and connects tangentially to an arc on one of the two terminal circles. If the separation between the initial and end points is sufficient, this can only be accomplished by a line segment. There are at most four such line segments, and computation of the travel distances is straightforward, as shown in Fig. 5.1 for two waypoints with initial and

terminal orientations, denoted $\alpha_k$ and $\alpha_{k+1}$ respectively. Note that $\alpha$ is measured counter-clockwise with respect to the positive $x$-axis.

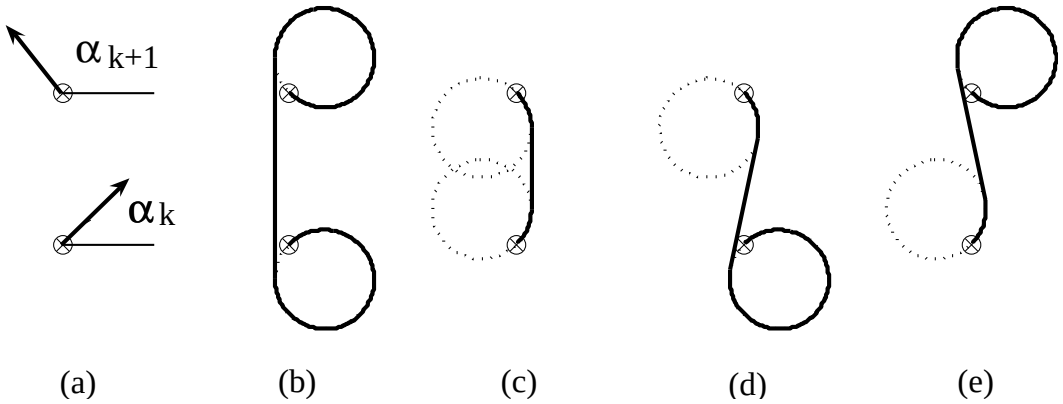


Figure 5.1: (a) Two waypoints $(x_k, y_k)$ and $(x_{k+1}, y_{k+1})$ with $\alpha_k = \frac{\pi}{4}$ and $\alpha_{k+1} = \frac{3\pi}{4}$. (b)-(e) Four ways of connecting two waypoints using Dubins curves.

Finding the shortest path between two points requires repetitively solving the shortest time algorithm for various entry and exit AUV orientations (i.e. $\alpha_k$, $\alpha_{k+1}$). The added challenge here is that there may be a family of paths that connects $s_k$ to $s_{k+1}$ with only one being the shortest. The multiplicity of paths connecting the two points complicates the search for initial and final headings so an exhaustive coarse-resolution search is implemented. In this algorithm, $\alpha_k$ and $\alpha_{k+1}$ is constrained to $\Lambda = \{\lambda\pi/4 \mid \lambda = 0, ..., 7\}$. With 8 possibilities for $\alpha_k$ and $\alpha_{k+1}$ and 4 ways of connecting them, a total of $8 \times 8 \times 4 = 256$ paths are possible for every pair of waypoints, with one of them being the optimal path. Fig. 5.2 shows three paths connecting $s_k$, to $s_{k+1}$. Note that there are additional paths connecting the same points which are not shown and that different values for $\alpha_k$ and $\alpha_{k+1}$ yield different costs.

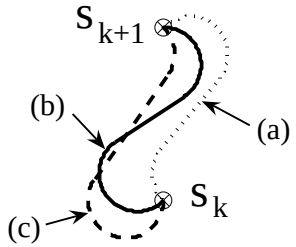


Figure 5.2: Multiple trajectories for different initial and final orientations. (a) $\alpha_k = \frac{3\pi}{4}$, $\alpha_{k+1} = \frac{-3\pi}{4}$ (b) $\alpha_k = \frac{-3\pi}{4}$, $\alpha_{k+1} = \pi$ (c) $\alpha_k = \frac{-\pi}{2}$, $\alpha_{k+1} = \frac{3\pi}{4}$.

The cost of traversing the sequence $S$ can then be calculated as:

$$C(S) = \min_{(\alpha_1,...,\alpha_l)\in\Lambda^l} \sum_{k=1}^{l} \Delta t_{(s_k,\alpha_k)\to(s_{k+1},\alpha_{k+1})} \tag{5.1}$$

## 5.2 AUV Dynamic Model

Calculating the time required to follow a Dubins path requires the knowledge of the vehicle dynamics. This thesis uses the REMUS AUV model created by Prestero [8] to which readers are referred to for the full derivation. The REMUS AUV has a torpedo shape with an ellipsoidal nose, a cylindrical constant radius mid-section, and a cubic spline tail section as illustrated in Fig. 5.3.


Figure 5.3: Picture of the REMUS AUV.

The vehicle has 6 degrees of freedom (DOF), namely *surge*, *sway*, *heave*, *pitch*, *roll*, and *yaw*. The AUV is assumed to be neutrally buoyant, completely rigid, and interacting with an ideal fluid. The vehicle is propelled by a thruster at its tail and steered by two independent pairs of fins for pitch and yaw control. With 6-DOF and only three independent actuators, the system is considered to be an underactuated system.

The 6-DOF non-linear model described in Appendix A can be used to simulate how different control and hydrodynamic forces affect the body-fixed velocities and the overall change in position and orientation of the vehicle. The simulation requires the ability to represent the vehicle motion with respect to both body-fixed and inertial coordinates. Therefore, the twelve states of a vehicle $V_i$ consisting of body-fixed velocities and inertial coordinates at time $t$ are given by:

$$\mathbb{X}_{i,\,t} = [u_i \;\; v_i \;\; w_i \;\; p_i \;\; q_i \;\; r_i \;\; x_i \;\; y_i \;\; z_i \;\; \phi_i \;\; \theta_i \;\; \psi_i]^T . \tag{5.2}$$

Given the complex and highly non-linear nature of the problem, numerical integration is used to solve for the vehicle position and orientation in time. At each time step, the vehicle state is updated by the general equation:

$$\mathbb{X}_{i,\,t+1} = f\left(\mathbb{X}_{i,\,t},\; \mathbb{U}_{i,\,t},\; u_c,\; \psi_c\right) \tag{5.3}$$

where $\mathbb{X}_{i,\,t}$ is the vehicle state vector, $\mathbb{U}_{i,\,t} = [\delta_s \;\; \delta_r \;\; X_{prop} \;\; K_{prop}]^T$ is the input vector, $u_c$ and $\psi_c$ are the magnitude and direction of the ocean current respectively.

For the input vector, $\delta_s$ is the stern fin angle, $\delta_r$ is the rudder fin angle, $X_{prop}$ is the surge force, and $K_{prop}$ is the yaw torque provided by the propeller.

The function $f$ in (5.3) uses the Euler method of numerical integration to yield the new vehicle state at each time step as:

$$\mathbb{X}'_{i,\,t+1} = \mathbb{X}_{i,\,t} + \left(\dot{\mathbb{X}}_{i,\,t} \cdot \Delta t\right) \tag{5.4}$$

where the state vector derivative $\dot{\mathbb{X}}_{i,\,t}$ is updated using the model $\dot{\mathbb{X}}_{i,\,t} = f\left(\mathbb{X}_{i,\,t},\ \mathbb{U}_{i,\,t}\right)$ from [8]. With the presence of a fixed current, the position of the vehicle relative to the inertial-fixed frame is updated as follows:

$$\begin{aligned} x_i &= x'_i + \left(u_c \cos\left(\psi_c\right) \cdot \Delta t\right) \\ y_i &= y'_i + \left(u_c \sin\left(\psi_c\right) \cdot \Delta t\right) \end{aligned} \tag{5.5}$$

where $x'_i$ and $y'_i$ denote the position of the $i^{th}$ vehicle after the integration step given in (5.4). By combining (5.4) and (5.5), the function $f$ in (5.3) is realized and can be used to update the general state of each vehicle. This full 6-DOF non-linear model is used in evaluating the final tour times of the sequences generated by the proposed algorithm.

## 5.3 Modification of Dubins Path

To calculate the shortest path between two points in the presence of ocean current, the dynamic model is used to determine the feasible states of the vehicle. In the presence of ocean currents, the shortest path between two points given $\alpha_k$ and $\alpha_{k+1}$ consists of arcs that are no longer circular but elliptic. These ellipses will have different curvatures depending on the magnitude and direction of the current (Fig. 5.4). The shape of the ellipse depends on the vehicle's orientation at the start of the turn and is calculated using the difference between the vehicle's heading $\psi$ and the direction of the current $\psi_c$.

To determine the shape of the ellipse, Eq. (5.3) was used to determine the state of the vehicle at each time step with $\mathbb{X}_0 = [1.15\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$, and $\mathbb{U}_0 = [0\ \ 1.7453\ \ 5.1553\ \ 0]^T$, where 1.15 m/s is the nominal speed of the AUV, 1.7453 rad is the rudder fin angle, and 5.1553 N is the propeller surge force. Note that finding the maximum curvature for the ellipse requires running the simulation using a maximum rudder fin angle of 1.7453 radians. Data was obtained for discrete cases of $u_c = \{0.1, 0.2, 0.3, 0.4, 0.5\}$, and $\psi_c = \{\kappa\pi/8 \text{ for } \kappa = 1, ..., 16\}$ and the vehicle's position was recorded at $\Delta\psi = \{\kappa\pi/8 \text{ for } \kappa = 1, ..., 16\}$, where $\Delta\psi$ is the fraction of a complete circumnavigation of the ellipse the vehicle travels (Fig. 5.5). A sample set of data for $u_c = 0.5$ and $\psi_c = \pi$ is shown in Table 5.1.

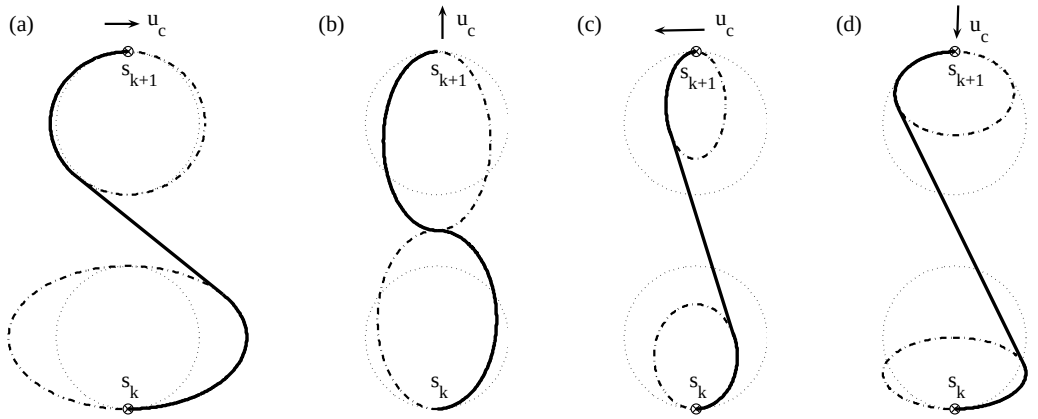


Figure 5.4: Dubins Curves between two waypoints with ocean currents $u_c = 0.25$ m/s. (a) $\psi_c$=0, (b) $\psi_c = \frac{\pi}{2}$, (c) $\psi_c = \pi$, and (d) $\psi_c = \frac{-\pi}{2}$.

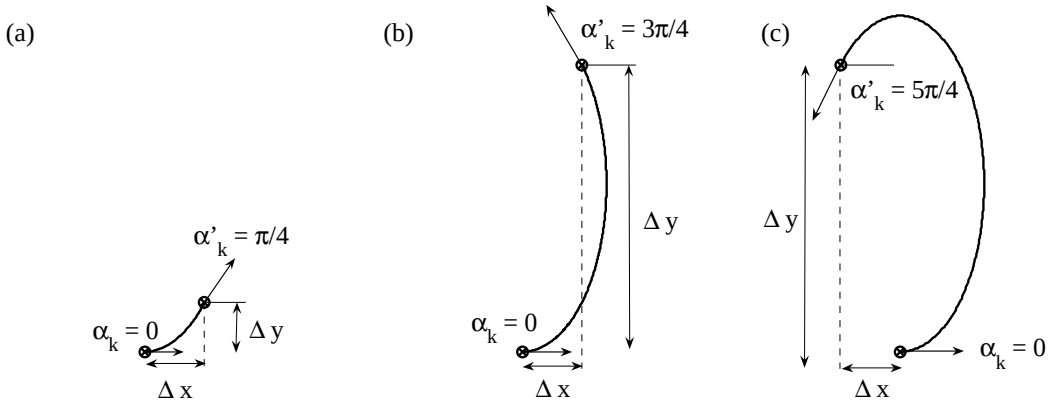


Figure 5.5: Vehicle position for various values of $\Delta\psi$ as the vehicle moves along the maximum curvature ellipse. (a) $\Delta\psi = \pi/4$. (b) $\Delta\psi = 3\pi/4$. (c) $\Delta\psi = 5\pi/4$.

Using this data, a lower order model $\tilde{f}$ was created to determine the position of the vehicle given the change in the vehicle's heading, and the magnitude and velocity of the current.

$$(\Delta x, \Delta y) = \tilde{f}(\Delta\psi, u_c, \psi_c) \tag{5.6}$$

The next step is to find the fraction of a complete circumnavigation of the ellipse to travel before and after the straight line segment. Finding a line segment tangent to two curves is solved by using an iterative process. As a starting point, the slope of the tangent line to two circular arcs of minimum radius (when $u_c = 0$) is calculated (Fig. 5.6a). Using that value, $P_a$ and $P_b$ are found on the respective ellipses whose slope is equal to the slope of the tangent (Fig. 5.6b). The position of $P_a$ and $P_b$ are determined by using $\tilde{f}$ from (5.6). The slope of the line segment from $P_a$ to $P_b$ is calculated and becomes the new slope for the next iteration (Fig. 5.6c). The process continues until convergence (Fig. 5.6d).

Table 5.1: Simulation results to map $\Delta\psi$ to $\Delta x$ and $\Delta y$ ($u_c = 0.5$ and $\psi_c = \pi$).

| $\Delta\psi$ | $\Delta x$ (m) | $\Delta y$(m) |
|---|---|---|
| $\pi/8$ | 0.70448 | 0.076994 |
| $\pi/4$ | 1.1176 | 0.51249 |
| $3\pi/8$ | 1.2667 | 1.2823 |
| $\pi/2$ | 1.0244 | 2.3036 |
| $5\pi/8$ | 0.39305 | 3.3099 |
| $3\pi/4$ | -0.69614 | 4.2793 |
| $7\pi/8$ | -2.0557 | 4.9614 |
| $\pi$ | -3.6749 | 5.3166 |
| $9\pi/8$ | -5.35 | 5.2631 |
| $5\pi/4$ | -6.9225 | 4.8129 |
| $11\pi/8$ | -8.2502 | 4.0362 |
| $3\pi/2$ | -9.2591 | 3.0093 |
| $13\pi/8$ | -9.8188 | 1.9654 |
| $7\pi/4$ | -9.9858 | 1.0254 |
| $15\pi/8$ | -9.8295 | 0.33069 |
| $2\pi$ | -9.4509 | -0.0197 |

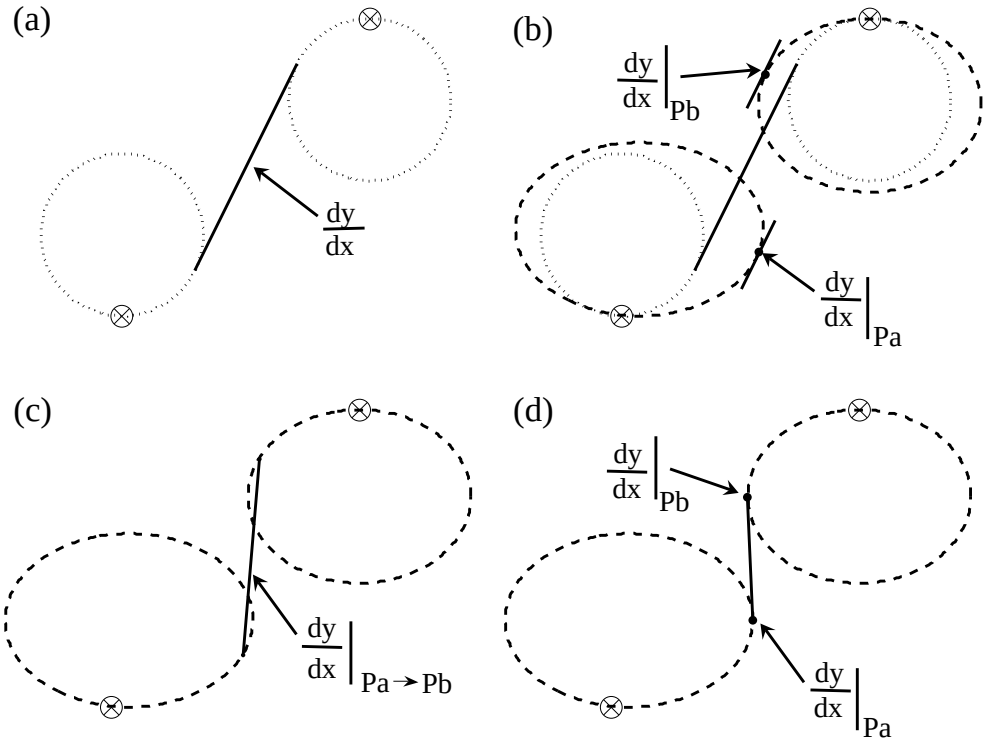


Figure 5.6: Illustration of the iterative process used to find a tangent to two curves.

Note that Fig. 5.6 depicts the desired path for the AUV to follow in the presence of ocean current. In order for the vehicle to stay on the desired path, the vehicle

heading $\psi$ must be calculated to compensate for the effect of current currents. Fig. 5.7 shows the vehicle heading as the vehicle moves from one wayoint to another.

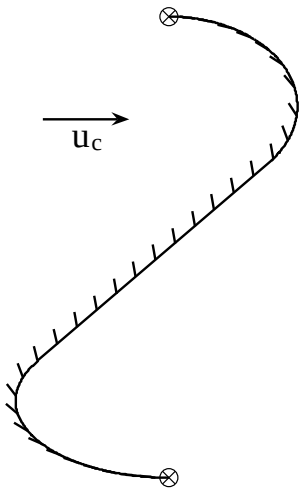


Figure 5.7: Path between two waypoints for a vehicle moving through water with current speed $u_c$ and $\psi_c = 0$ rad. The short tics equally spaced along the path shows the vehicle heading but not speed.

A summary of the procedure necessary to calculate the path cost between two points is presented in Algorithm 1. Note that this function uses the function *getPosition*. The *getPosition* function takes as inputs the fraction of a complete circumnavigation of the ellipse to travel, the position and heading of the vehicle at the starting point, and the magnitude and orientation of the ocean current and outputs the final position of the vehicle using the equation in (5.6).

**Algorithm 1** Iterative Algorithm for finding a tangent to two curves.

$(x_1, y_1)$ := coordinate of starting point
$(x_2, y_2)$ := coordinate of final point
$\theta_1$ := vehicle heading at starting point
$\theta_2$ := vehicle heading at final point
$u_c$ := current velocity
$\psi_c$ := current direction
$\psi_{desired}$ := initialize to zero
$(x_{1circle}, y_{1circle})$ := center of maximum curvature circle (starting point)
$(x_{2circle}, y_{2circle})$ := center of maximum curvature circle (final point)
$r$ := radius of maximum curvature circle
$\rho$ := sqrt($(x_1$-$x_2)^2$ + $(y_1$-$y_2)^2$)
$\rho_{circle}$ := sqrt($(x_{1circle}$-$x_{2circle})^2$ + $(y_{1circle}$-$y_{2circle})^2$)
$\theta_{circle}$ := atan2($y_{2circle}$-$y_{1circle}$, $x_{2circle}$-$x_{1circle}$)
$P_a$ := ($x_{1circle}$ + $r$*cos($\theta_{circle}$ + $\pi/2$), $y_{1circle}$ + $r$*sin($\theta_{circle}$ + $\pi/2$))
$P_b$ := ($x_{2circle}$ + $r$*cos($\theta_{circle}$ + $\pi/2$), $y_{2circle}$ + $r$*sin($\theta_{circle}$ + $\pi/2$))
$\theta_{P_a,P_b}$ := atan2($P_{by}$ - $P_{ay}$,$P_{bx}$ - $P_{ax}$)
**while** ($abs(\psi_{desired} - \theta_{P_a,P_b}) > epsilon$) **do**
    $\psi_{desired}$ := atan2($P_{by}$ - $P_{ay}$,$P_{bx}$ - $P_{ax}$)
    $u_{c_{\psi_{desired}}}$ := $u_c$ cos($\psi_c$ - $\psi_{desired}$)
    $u_{c_N}$ := $u_c$ sin($\psi_c$ - $\psi_{desired}$)
    $\psi$ := $-$arcsin($u_{c_N}/u$) + $\psi_{desired}$
    $\Delta\theta_{P_a}$ := $\theta_1$ - $\psi$
    $\Delta\theta_{P_b}$ := $\psi$ - $\theta_2$
    $P_a$ := getPosition($\Delta\theta_{P_a}$, $x_1$, $y_1$, $theta_1$, $u_c$, $\psi_c$)
    $P_b$ := getPosition($\Delta\theta_{P_b}$, $x_2$, $y_2$, $theta_2$, $u_c$, $\psi_c$)
    $\theta_{P_a,P_b}$ := atan2($P_{by}$ - $P_{ay}$,$P_{bx}$ - $P_{ax}$)
**end while**
**return** $P_a$, $P_b$.

## 5.4 Shortest Time Between Two Waypoints

To calculate the time $\Delta t_{(s_k,\alpha_k)\to(s_{k+1},\alpha_{k+1})}$ in (5.1), a lower order model $\hat{f}$ is created based on the full model $f$ from (5.3) as:

$$\Delta t_{(s_k,\alpha_k)\to(s_{k+1},\alpha_{k+1})} = \hat{f}\left[s_k, s_{k+1}, \alpha_k, \alpha_{k+1}, u_0, u_c, \psi_c\right] \tag{5.7}$$

### 5.4.1 Arcs

For arcs, the lower order model is a piecewise linear function built from sampling the full model. Using the full model, the vehicle orientation can be determined at a certain time $t$. In order to find the time required to obtain a specific heading, linearly interpolation is used on the data obtained from the full model at various

fractions of a complete circumnavigation of the ellipse ($\kappa\pi/8$ for $\kappa = 1, ..., 16$). The results are shown in Table 5.2.

Table 5.2: Simulation results to map $\Delta\psi$ to $\Delta t_{arc}$

| $\Delta\psi$ | $\Delta t_{arc}$ (s) |
|---|---|
| $\pi/8$ | 1.2 |
| $\pi/4$ | 2.2 |
| $3\pi/8$ | 3.3 |
| $\pi/2$ | 4.5 |
| $5\pi/8$ | 5.65 |
| $3\pi/4$ | 6.9 |
| $7\pi/8$ | 8.1 |
| $\pi$ | 9.35 |
| $9\pi/8$ | 10.6 |
| $5\pi/4$ | 11.85 |
| $11\pi/8$ | 13.1 |
| $3\pi/2$ | 14.4 |
| $13\pi/8$ | 15.65 |
| $7\pi/4$ | 16.9 |
| $15\pi/8$ | 18.15 |
| $2\pi$ | 19.3 |

### 5.4.2 Straight line segments

For straight line segments, consider a vehicle moving at speed $u$ and heading $\psi$ through the water with current velocity $u_c$ and direction $\psi_c$. The vehicle's velocity along the desired path has magnitude $u_{desired}$ and direction $\psi_{desired}$. These velocities are illustrated in Fig. 5.8. Let $u_{c_{\psi_{desired}}} = u_c \cos(\psi_c - \psi_{desired})$ be the current component assisting motion along the desired direction and $u_{c_N} = u_c \sin(\psi_c - \psi_{desired})$ be the current component $\pi/2$ radians to the left of the desired direction. Staying on the desired path requires the perpendicular component of the vehicle velocity $u \sin(\psi - \psi_{desired})$ to cancel the perpendicular component of the current $u_{c_N}$. The heading $\psi$ and speed $u_{desired}$ along the desired vehicle motion direction are

$$\psi = -\arcsin(u_{c_N}/u) + \psi_{desired}, \quad u_{desired} = u_{c_{\psi_{desired}}} + u\sqrt{1 - (u_{c_N}/u)^2}. \tag{5.8}$$

As long as $|u_{c_N}| < u$, the vehicle can stay on the desired path, but the velocity decreases as $|u_{c_N}| \to u$. Keeping the vehicle on the desired path is critical because making measurements in the right places requires the vehicle to stay on track in the face of currents.

The time required to travel from $P_a$ to $P_b$ can then be calculated as follows:

$$\Delta t_{straight} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}/u_{desired} \tag{5.9}$$

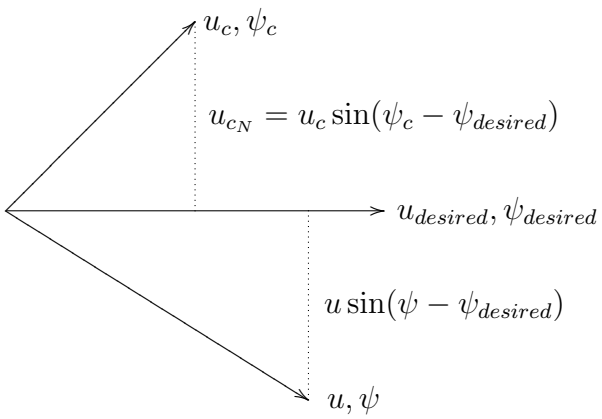


Figure 5.8: Illustration of the relative velocities. The vehicles has nominal speed $u$ and heading $\psi$, the ocean current has speed $u_c$ and direction $\psi_c$, and the vehicle's velocity along the desired path has magnitude $u_{desired}$ and orientation $\psi_{desired}$. Also shown are the perpendicular components of the vehicle's velocity $u \sin(\psi - \psi_{desired})$ and current velocity $u_{c_N} = u_c \sin(\psi_c - \psi_{desired})$ that cancel to give (5.8).

Combining these results in

$$\Delta t_{(s_k,\alpha_k)\to(s_{k+1},\alpha_{k+1})} = \Delta t_{arc(s_k \to P_a)} + \Delta t_{straight(P_a \to P_b)} + \Delta t_{arc(P_b \to s_{k+1})}. \quad (5.10)$$

## 5.5 Path Time Calculation

When a vehicle bids for task $d_j$ and tries to add the new task at every point in the current sequence $S$ as described in Section 4.2, the vehicle must try every value of $\Lambda = \{\lambda\pi/4 \mid \lambda = 0, ..., 7\}$ for the orientation at task $d_j$. With the insertion of task $d_j$ in between $s_k$ and $s_{k+1}$, the optimal orientations $\alpha_k$ and $\alpha_{k+1}$ also have to be recalculated with all values of $\Lambda$. The orientations at all other task points in the sequence $S$ is kept from the previous round of bidding since the addition of task $d_j$ has minimal effect on the rest of the tour. Because the sequence $S$ and the values for the optimal orientation at each task point with the exception of $\alpha_k$ and $\alpha_{k+1}$ are kept from the previous round of bidding, (5.7) only needs to be calculated between tasks ($s_{k-1}$, $s_k$), ($s_k$, $d_j$), ($d_j$, $s_{k+1}$), and ($s_{k+1}$, $s_{k+2}$). This simplifies the algorithm and significantly decreases the processing time.

To illustrate this, consider the sequence $S = \{s_1, s_2, s_3, s_4\}$ and optimal orientations $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ for a list of tasks a vehicle currently owns (Fig. 5.9). The vehicle starts by inserting task $d_j$ in between tasks $s_1$ and $s_2$, trying all values of $\Lambda$ for $\alpha_1$, $\alpha_2$, and $\alpha_{d_j}$. The path cost is calculated for all combinations of $\alpha_1$, $\alpha_2$, and $\alpha_{d_j}$ and the configuration that yielded the lowest cost is shown in Fig. 5.10a. The vehicle then tries to add task $d_j$ in between $s_2$ and $s_3$ (Fig. 5.10b), between $s_3$ and $s_4$ (Fig. 5.10c), and finally between $s_4$ and $s_1$ (Fig. 5.10d). The results are shown in Table 5.3.

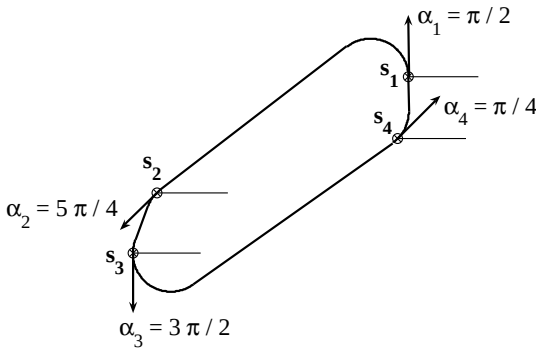


Figure 5.9: Sequence $\{s_1, s_2, s_3, s_4\}$ and optimal orientations $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ for tasks the vehicle currently owns.

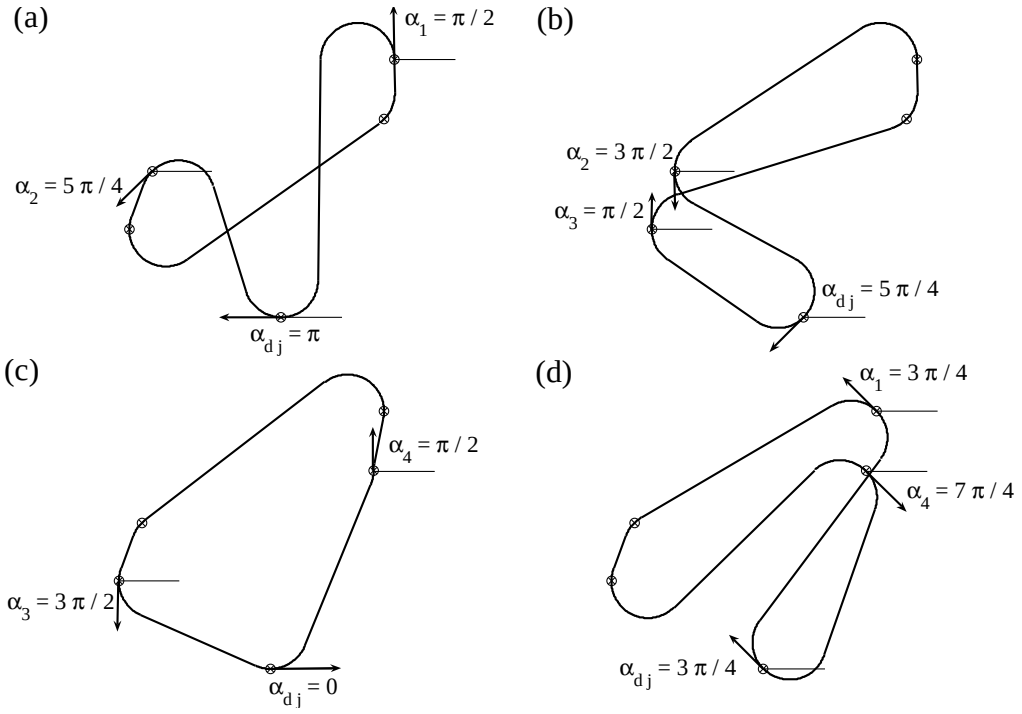


Figure 5.10: Inserting task $d_j$ in between every pair of tasks in sequence $S$.

Table 5.3: Path costs for the insertion of $d_j$ in between every pair of task points.

| Insertion point | $\alpha_k$ | $\alpha_{k+1}$ | $\alpha_{d_j}$ | Cost (sec) |
|---|---|---|---|---|
| Between $s_1$ and $s_2$ | $\pi/2$ | $5\pi/4$ | $\pi$ | 111.3 |
| Between $s_2$ and $s_3$ | $3\pi/2$ | $\pi/2$ | $5\pi/4$ | 110.2 |
| Between $s_3$ and $s_4$ | $3\pi/2$ | $\pi/2$ | 0 | 82.9 |
| Between $s_4$ and $s_1$ | $7\pi/4$ | $3\pi/4$ | $3\pi/4$ | 126.3 |

Since the insertion of task $d_j$ between tasks $s_3$ and $s_4$ produced the lowest cost, the vehicle set $S' = \{s_1, s_2, s_3, d_j, s_4\}$ and submits a bid value of $B(d_j) = 82.9$.

# Chapter 6

# Simulation Results

The algorithm described in Chapter 4 was implemented in Matlab and was developed on an Intel 1.66 GHz Core 2 Duo processor T5500 with 2GB RAM and running Windows XP SP3. To demonstrate the performance of the proposed algorithm, computer simulations were carried out with a model of the REMUS AUV. Simulations were conducted on 50 datasets, each set containing between 6 and 20 task points. The task points were generated randomly and uniformly inside a square with side lengths of 25 meters. The task points were generated close together to highlight the necessity for considering the curvature constraints. As a baseline for comparison, the "alternating algorithm" described by Savla et al. [35] was used. Simulations were conducted with the parameters listed in Table 6.1.

Table 6.1: Simulation Parameters

| Symbol | Value | Description |
|---|---|---|
| $n$ | 1 to 5 | Number of AUVs |
| $m$ | 1 to 20 | Number of task points |
| $u$ | 1.15 | Nominal Translational velocity along x-axis (meters/sec) |
| $u_c$ | 0 to 0.5 | Current velocity (meters/sec) |
| $\psi_c$ | $-\pi$ to $+\pi$ | Current direction from positive x-axis (rad) |
| $\delta_r$ | 1.7453 | Rudder fin angle (rad) |
| $X_{prop}$ | 5.1553 | Propeller speed (N) |

To illustrate the behaviour of the proposed algorithm, consider one particular trial with $n = 3$ and $m = 20$ using the dataset shown in Fig. 6.1.

Using the *k-means* clustering method described in Chapter 4, the 20 task points were partitioned into 3 clusters as shown in Fig. 6.2. It should be noted that Matlab uses a two-phase iterative algorithm for *k-means* clustering that only converges to a local minimum. The problem of finding the global minimum can only be solved in general by an exhaustive choice of starting points. Therefore, Matlab produces

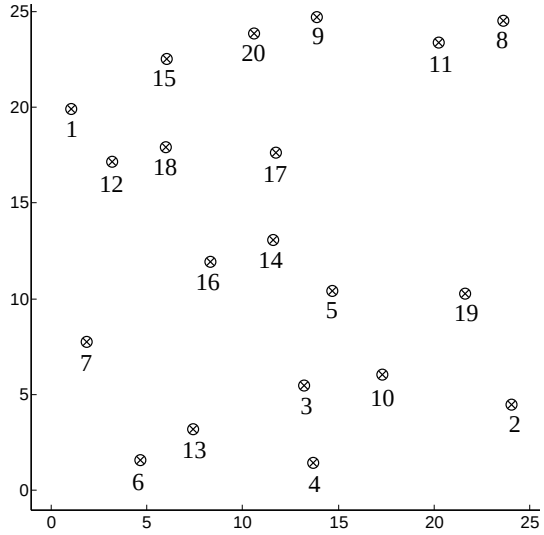


Figure 6.1: Dataset of 20 task points randomly generated.

different clusters using the same dataset depending on the starting points chosen and Fig. 6.2 is one of many solutions.

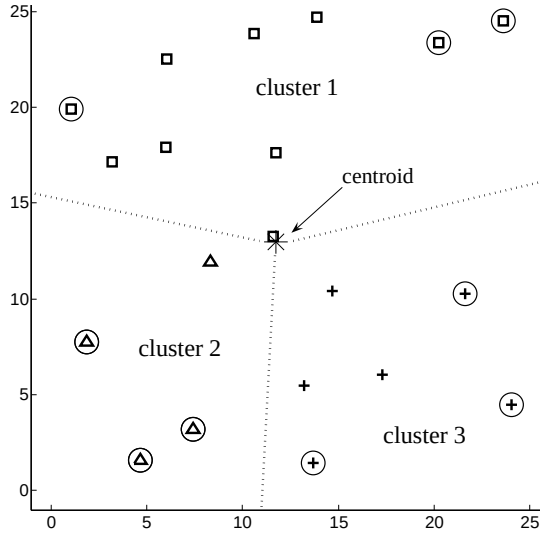


Figure 6.2: Results from clustering using $k$-means. Tasks with a circle around it are assigned to the vehicle responsible for that cluster.

The centroid for this dataset was calculated to be at (11.7429, 13.3626) and the distance from each task point to the centroid is shown in Table 6.2.

The three task points in each cluster that were farthest from the centroid were assigned to the vehicle responsible for the cluster. This resulted in the following sequence for each vehicle: $S_1 = \{d_8, d_{11}, d_1\}$; $S_2 = \{d_6, d_7, d_{13}\}$; $S_3 = \{d_2, d_4, d_{19}\}$.

Table 6.2: Task points are ordered in decreasing distance from centroid.

| Task Number | Cluster Number | Distance to Centroid |
|---|---|---|
| 8 | 1 | 16.281 |
| 2 | 3 | 15.184 |
| 6 | 2 | 13.732 |
| 11 | 1 | 13.126 |
| 1 | 1 | 12.521 |
| 4 | 3 | 12.067 |
| 9 | 1 | 11.549 |
| 7 | 2 | 11.357 |
| 13 | 2 | 11.057 |
| 15 | 1 | 10.773 |
| 20 | 1 | 10.554 |
| 19 | 3 | 10.358 |
| 12 | 1 | 9.335 |
| 10 | 3 | 9.1847 |
| 3 | 3 | 8.0338 |
| 18 | 1 | 7.3387 |
| 17 | 1 | 4.2854 |
| 5 | 3 | 4.1767 |
| 16 | 2 | 3.7267 |
| 14 | 1 | 0.32833 |

## 6.1 MTSP Solution

The MTSP solution uses the clustering and auctioning method as described in Chapter 4. The difference between the MTSP solution and that of the proposed algorithm is in the calculation of the bids. The MTSP solution does not consider the curvature constraints of the vehicle and therefore, assumes the vehicle can turn on the spot. This results in straight line segments between task points with the vehicle changing orientations at each task point. The MTSP solution also does not consider the effect of ocean currents when creating the sequence for each vehicle.

The MTSP solution begins with the sequences generated from clustering. For this solution, the following sequences were used as a starting point: $S_1 = \{d_8, d_{11}, d_9\}$; $S_2 = \{d_1, d_7, d_{15}\}$; $S_3 = \{d_2, d_6, d_4\}$. The unassigned task points were ordered in decreasing distance from the centroid as $\{d_{13}, d_{20}, d_{19}, d_{12}, d_{10}, d_3, d_{18}, d_{17}, d_5, d_{16}, d_{14}\}$ and were auctioned off in that sequence.

The first step was to create a path for each vehicle as shown in Fig. 6.3(a). The shortest times between every pair of task points in the sequence were calculated and stored in memory. The auction began with task $d_{13}$ up for bidding. Each vehicle calculated the minimum cost of adding task $d_{13}$ to its sequence $S$ and the values submitted were $B_1 = 53.296$, $B_2 = 39.717$, and $B_3 = 36.166$. Since vehicle 3 submitted the lowest bid, it won the task and updated its sequence as

$S_3 = \{d_2, d_{13}, d_6, d_4\}$ (Fig. 6.3(b)). The process continued until all unallocated tasks were auctioned off as illustrated in Fig. 6.3.

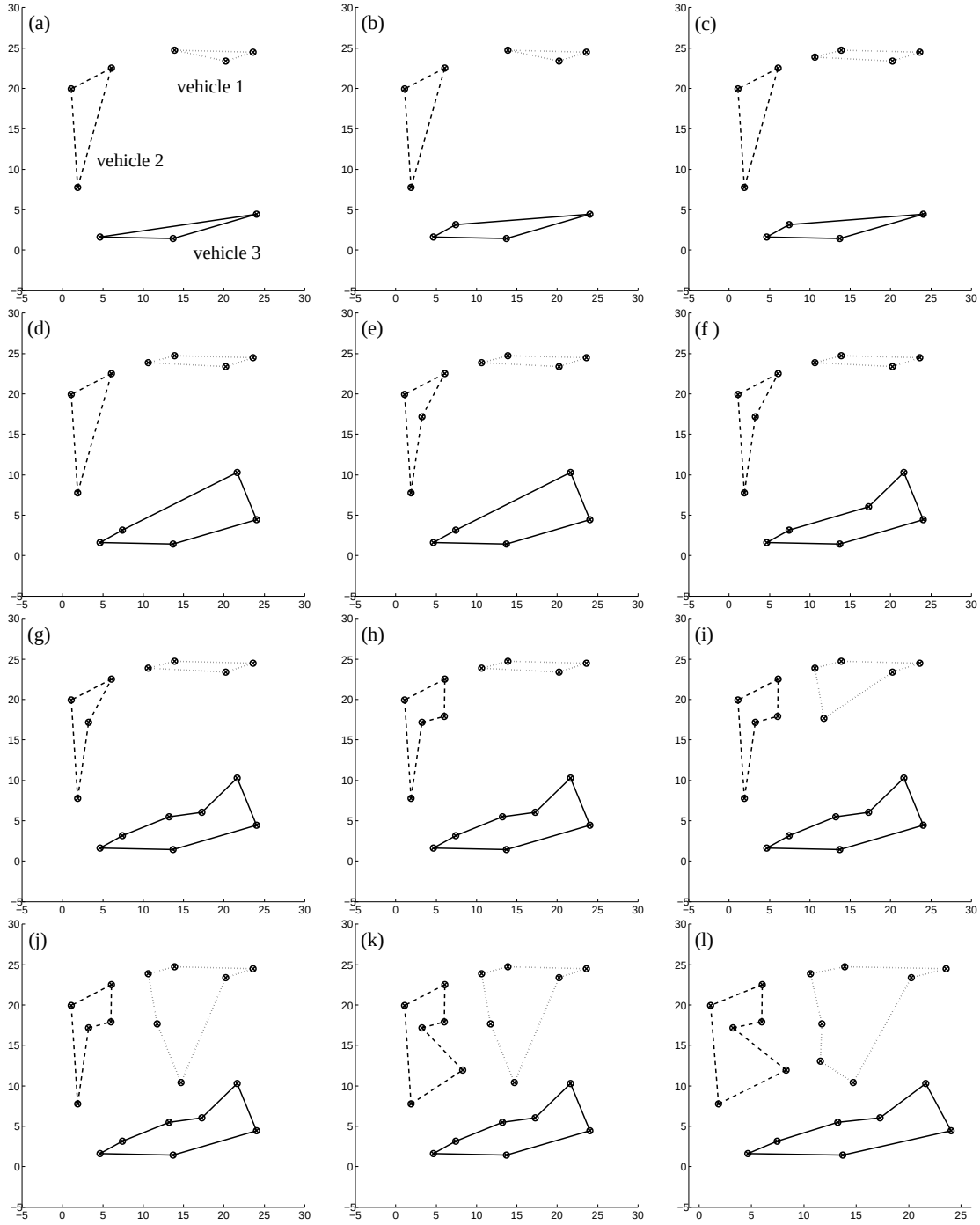


Figure 6.3: Illustration of task allocation using auctions for solving the MTSP. (a) All vehicles began with 3 task points determined from clustering. (b)-(l) Vehicles bid for task points as they were auctioned off and added the task point to their tour if it submitted the lowest cost.

The final sequences for each vehicle were $S_1 = \{d_8, d_{11}, d_5, d_{14}, d_{17}, d_{20}, d_9\}$; $S_2 = \{d_1, d_7, d_{16}, d_{12}, d_{18}, d_{15}\}$; $S_3 = \{d_2, d_{19}, d_{10}, d_3, d_{13}, d_6, d_4\}$ and the path costs were 41.1820, 36.1909, and 41.4692 respectively.

The next step was to find feasible paths for each vehicle to follow taking into consideration the curvature constraints of the vehicle as well as the effects of ocean current. The "alternating algorithm" from [35] works as follows: Given a sequence $S$ from solving the Euclidean MTSP (i.e. tours that do not consider path curvature), label the edges on the tour in order with consecutive integers. A Dubins TSP tour can be constructed by retaining all odd-numbered (except the $n^{th}$) edges, and replacing all even-numbered edges with minimum length Dubins paths preserving the point ordering. In other words, the algorithm consists of the following steps:

**Algorithm 2** Alternating Algorithm

set $(s_1,...,s_n)$ := optimal Euclidean TSP ordering $S$.
set $\alpha_1$ := orientation of segment from $s_1$ to $s_2$
**for** $i = 2$ ,..., $n-1$ **do**
  **if** $i$ is even **then**
    set $\alpha_i := \alpha_{i-1}$
  **else**
    set $\alpha_i$ := orientation of segment from $s_i$ to $s_{i+1}$
  **end if**
**end for**
**if** $n$ is even **then**
  set $\alpha_n := \alpha_{n-1}$
**else**
  set $\alpha_n$ := orientation of segment from $s_n$ to $s_1$
**end if**
**return** sequence of configurations $\{(s_i, \alpha_i)\}_{i\in\{1,...,n\}}$.

The output of the "alternating algorithm" applied to the sequences generated above from solving the MTSP is illustrated in Fig. 6.4. The path cost for each AUV to traverse the sequence was calculated using the method described in Section 5.5 except the orientations at each task point did not need to be tested for all values of $\Lambda$ since it was determined using the "alternating algorithm".

## 6.2 Proposed Planner Solution

The proposed planner solution was generated by using the path cost calculations described in Section 5.5. For this solution, the following sequences were used as a starting point: $S_1 = \{d_8, d_{11}, d_1\}$; $S_2 = \{d_6, d_7, d_{13}\}$; $S_3 = \{d_2, d_4, d_{19}\}$. The unassigned task points were ordered in decreasing distance from the centroid as $\{d_9, d_{15}, d_{20}, d_{12}, d_{10}, d_3, d_{18}, d_{17}, d_5, d_{16}, d_{14}\}$ and were auctioned off in that order.

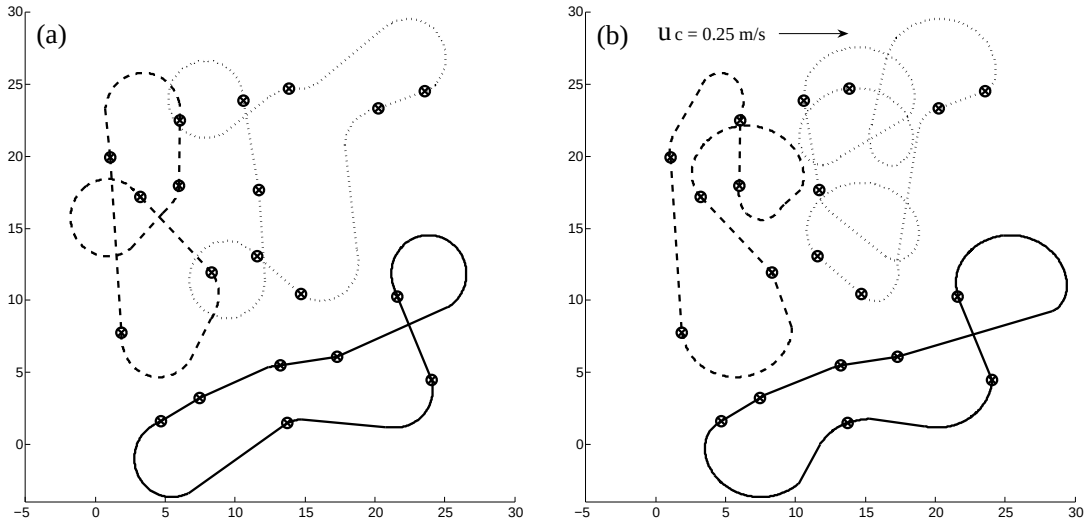


Figure 6.4: Dubins path using sequence generated from solving the MTSP. (a) $u_c = 0$, (b) $u_c = 0.25$ m/s, $\psi_c = 0$ rad.

First consider the case when there was no ocean current. The path cost calculations considered the curvature constraints of the vehicle and generated Dubins paths between pairs of task points. The first step was to create a path for each vehicle as shown in Fig. 6.5(a). The auction began with task $d_9$ up for bidding. Each vehicle calculated the minimum cost of adding task $d_9$ to its sequence $S$ and the values submitted were $B_1 = 51.3776$, $B_2 = 56.6235$, and $B_3 = 55.1427$. Since vehicle 1 submitted the lowest bid, it won the task and updated its sequence as $S_1 = \{d_8, d_{11}, d_1, d_9\}$ (Fig. 6.5(b)). The process continued until all unallocated tasks were auctioned off as illustrated in Fig. 6.5. The final sequences for each vehicle were $S_1 = \{d_8, d_{11}, d_{18}, d_1, d_9\}$; $S_2 = \{d_6, d_7, d_{12}, d_{15}, d_{20}, d_{17}, d_{16}, d_{13}\}$; $S_3 = \{d_2, d_4, d_5, d_{14}, d_3, d_{10}, d_{19}\}$ and the path costs were 52.8016, 52.4127, and 58.3678 respectively.

Next consider the case when there was ocean current with values $u_c = 0.25$ m/s and $\psi_c = 0$ rad. The path cost calculations considered the curvature constraints of the vehicle as well as the effect of ocean currents. The first step was to create a path for each vehicle as shown in Fig. 6.6(a). The auction began with task $d_9$ up for bidding. Each vehicle calculated the minimum cost of adding task $d_9$ to its sequence $S$ and the values submitted were $B_1 = 52.8952$, $B_2 = 54.0549$, and $B_3 = 58.8124$. Since vehicle 1 submitted the lowest bid, it won the task and updated its sequence as $S_1 = \{d_8, d_{11}, d_1, d_9\}$ (Fig. 6.6(b)). The process continued until all unallocated tasks were auctioned off as illustrated in Fig. 6.6. The final sequences for each vehicle were $S_1 = \{d_8, d_{11}, d_{17}, d_{12}, d_1, d_9\}$; $S_2 = \{d_6, d_7, d_{14}, d_{20}, d_{15}, d_{18}, d_{16}, d_{13}\}$; $S_3 = \{d_2, d_4, d_3, d_{10}, d_{19}, d_5\}$ and the path costs were 54.1014, 57.8366, and 57.5093 respectively.

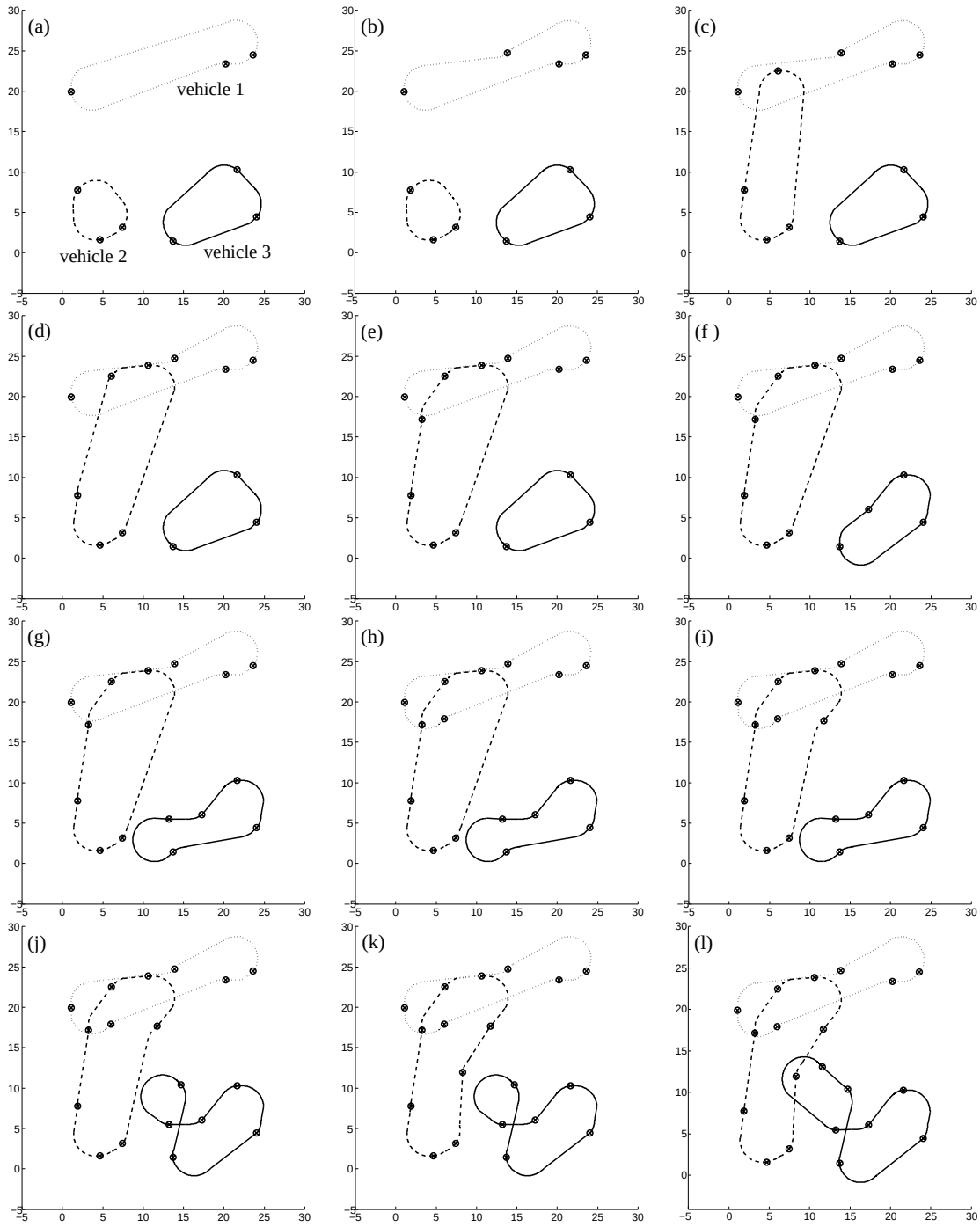


Figure 6.5: Illustration of task allocation using the proposed algorithm without ocean current.

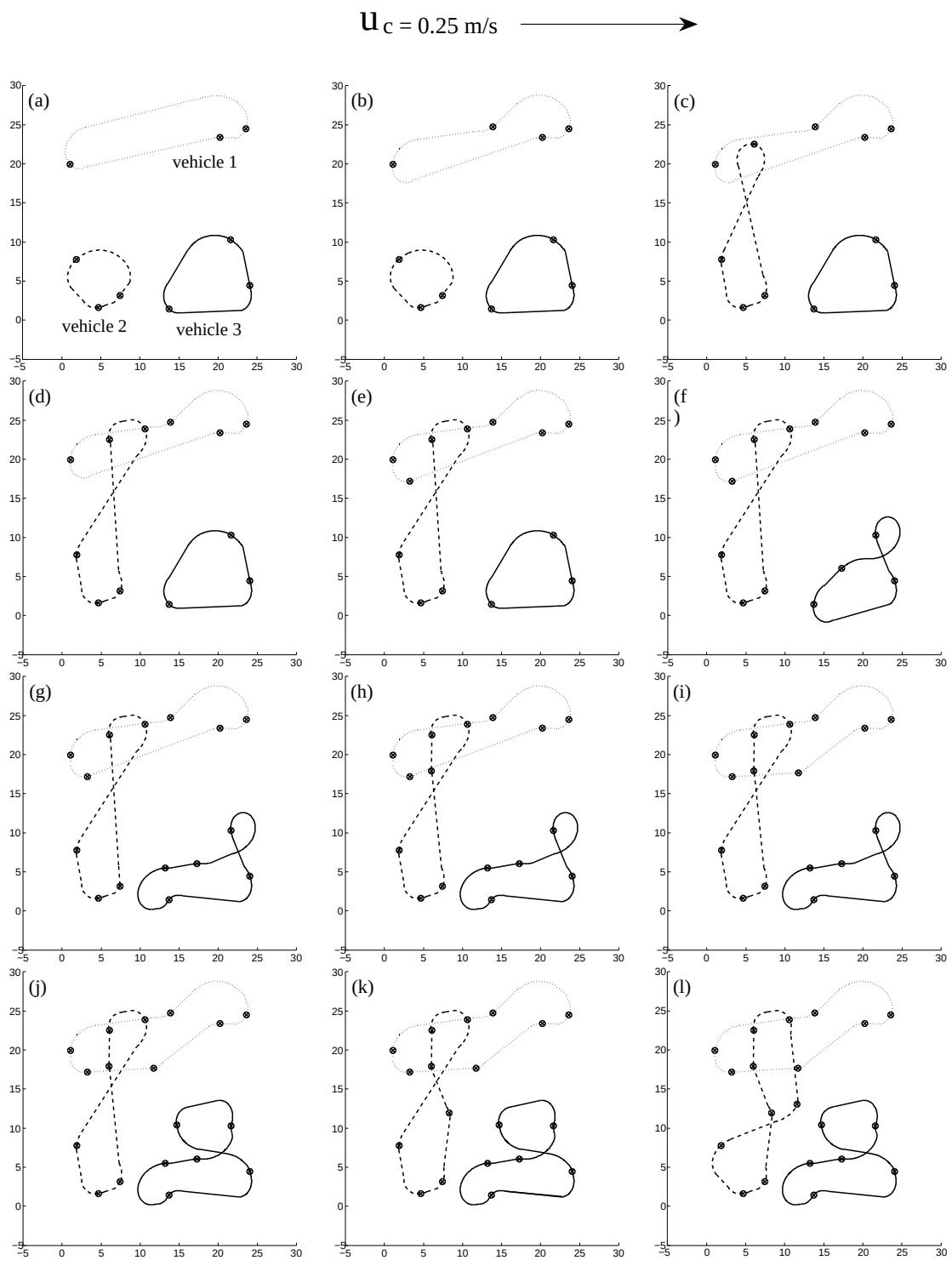


Figure 6.6: Illustration of task allocation using the proposed algorithm with $u_c = 0.25$ m/s and $\psi_c = 0$ rad.

## 6.3 Discussion

The results from running the simulation on 50 different datasets are summarized in Table 6.3 using the following criteria:

$$T_{\max} = \max C_{sim}(S_i) \quad \text{and} \quad T_{\text{avg}} = \frac{\sum_{i=1}^{n} C_{sim}\,(S_i)}{n}.$$

$C_{sim}$ is the cost calculated by running the planned tours $S_i$ through the *full* dynamic model in Equation (5.3). On average, the proposed algorithm reduced $T_{\max}$ by 43% over the "alternating algorithm" in the absense of currents and 45% with the presence of currents.

Table 6.3: Summary of simulation results using $n = \{1, 2, 3, 4, 5\}$ and $m = \{6, 7, 8,..., 20\}$.

| | No current | | With current | |
|---|---|---|---|---|
| | $T_{max}$ % Improvement | $T_{avg}$ % Improvement | $T_{max}$ % Improvement | $T_{avg}$ % Improvement |
| $n = 1$ | 36.06 | 36.06 | 42.83 | 42.83 |
| $n = 2$ | 37.83 | 34.26 | 43.39 | 38.90 |
| $n = 3$ | 47.17 | 37.63 | 48.64 | 46.68 |
| $n = 4$ | 52.22 | 41.05 | 45.83 | 37.03 |
| $n = 5$ | 41.66 | 31.21 | 44.13 | 40.46 |

Consider the solutions generated by the "alternating algorithm" and the proposed algorithm for the case where $n = 3$ and $m = 20$ using the dataset in Fig. 6.1. The results are summarized in Table 6.4 and Fig. 6.7 for the different cases. For the case with no ocean currents, the "alternating algorithm" created paths with numerous loops when two successive points were close together and the vehicle orientation did not allow for the second point to be reached without long maneuvers (Fig. 6.7(a)). This was avoided in the proposed algorithm by generating sequences that were feasible but limited the number of additional loops (Fig. 6.7(c)). Similar results were obtained with the presence of ocean currents as shown in Fig. 6.7(b) and Fig. 6.7(d).

Note that the proposed algorithm produced different sequences for the case with no ocean currents and the case with ocean currents. This is because the bidding scheme considered the possibility that two successive points that were reachable in the absence of ocean currents may no longer be reachable without extra loops due to the increase in turning radius from the ocean currents. Also, the paths generated by the proposed algorithm attempted to avoid paths that forced the vehicles to drive against the ocean current. Instead, paths that allowed the ocean current to aid the vehicle in the direction of travel were favoured.

For a sufficiently dense sets of points, it becomes clear that the ordering of the Euclidean tours are not optimal in the case of the Dubins MTSP. This is due to

Table 6.4: Summary of path costs for the dataset in Fig. 6.1 with $n = 3$ and $m = 20$.

| | No current | | With current | |
|---|---|---|---|---|
| | Alternating Algorithm | Proposed Algorithm | Alternating Algorithm | Proposed Algorithm |
| $T_{max}$ (s) | 89.9 | 58.4 | 101.2 | 59.8 |
| $T_{avg}$ (s) | 75.1 | 54.5 | 88.1 | 57.1 |

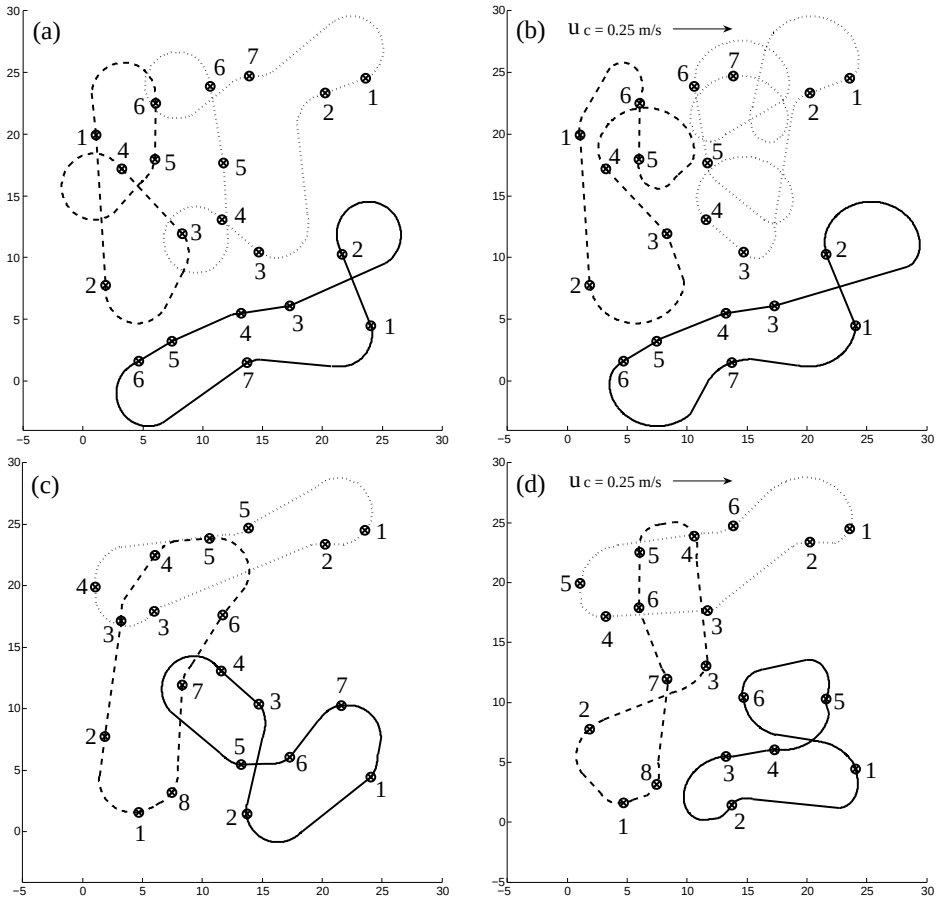


Figure 6.7: Sequences generated by the "alternating algorithm" and the proposed algorithm using the dataset in Fig. 6.1 with $n = 3$ and $m = 20$. (a) Alternating algorithm, $u_c = 0$, (b) Alternating algorithm, $u_c$= 0.25 m/s, $\psi_c$= 0, (c) Proposed algorithm, $u_c$= 0, (d) Proposed algorithm, $u_c$= 0.25 m/s, $\psi_c$= 0.

the fact that there is little relationship between the Euclidean and Dubins metrics, especially when the Euclidean distances are small with respect to the turning radius. An algorithm for the Euclidean problem will tend to schedule very close points in a successive order, which can imply long maneuvers for the AUV. This is clearly

demonstrated by the numerous loops that become problematic with dense sets of points. The algorithm proposed in this thesis does not rely on the Euclidean solution and therefore, even in the presence of ocean currents, can create paths that are feasible for curvature bound vehicles.

## 6.4 System Performance

The processing time for running the proposed algorithm using $n = \{1,2,3,4,5\}$ and $m = \{1,2,3,...,20\}$ are plotted in Fig. 6.8. As a baseline for comparison, the processing time for running the TSP solution followed by the "alternating algorithm" is also shown on the graph for $n = 1$.

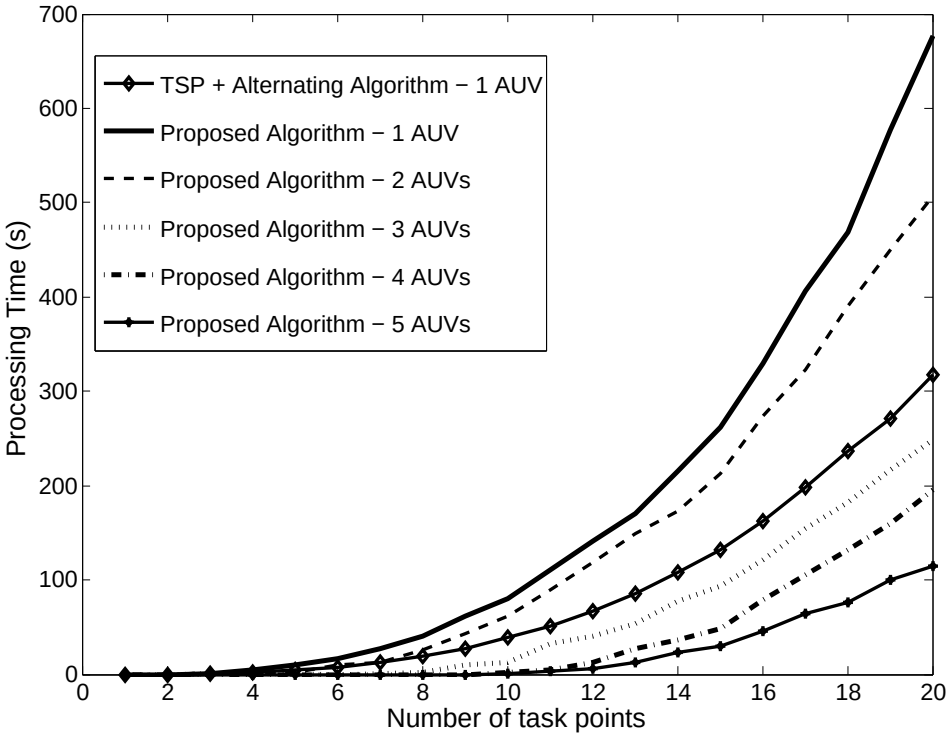


Figure 6.8: Processing Time.

The complexity of the algorithm is a result of each vehicle $i$ trying to insert the new task at every point in the current sequence $S_i = \{s_1, s_2, ..., s_l\}$, where $l$ is the number of previously won tasks by $V_i$, when calculating the bid cost. As $l$ increases by 1, the number of calculations required by the vehicle $i$ increases by $8 \times 8 \times 8 = 512$ configurations for testing the different orientations at $s_k$, $d_j$, and $s_{k+1}$, and for every configuration, there are 4 possible Dubins paths.

# Chapter 7

# Experiment Description

Experiments were conducted at the Avila Pier in California using the Iver2 Autonomous Underwater Vehicle as shown in Fig. 7.1.


Figure 7.1: Picture of the Iver2 AUV.

The Iver2 is a small, low cost AUV developed by Ocean Server Technology Inc. It is 4 feet long, 6 inches in diameter, and weighs less than 50 pounds. It has independent control of all 4 control surfaces, a wireless network interface, a simple user interface, and a robust mechanical design [36]. The Iver2 AUV is similar to the REMUS AUV in many aspects, and therefore, the governing equations of motion described above for the REMUS AUV also apply to the Iver2 AUV.

## 7.1 Mission Planning Software

Planning a mission for the Iver2 consists of a user-defined sequence of GPS referenced waypoints that the vehicle follows. To plan a mission, the VectorMap software that accompanies the Iver2 was used in conjunction with a standard geo-referenced NOAA Raster Navigational Chart as shown in Fig. 7.2.

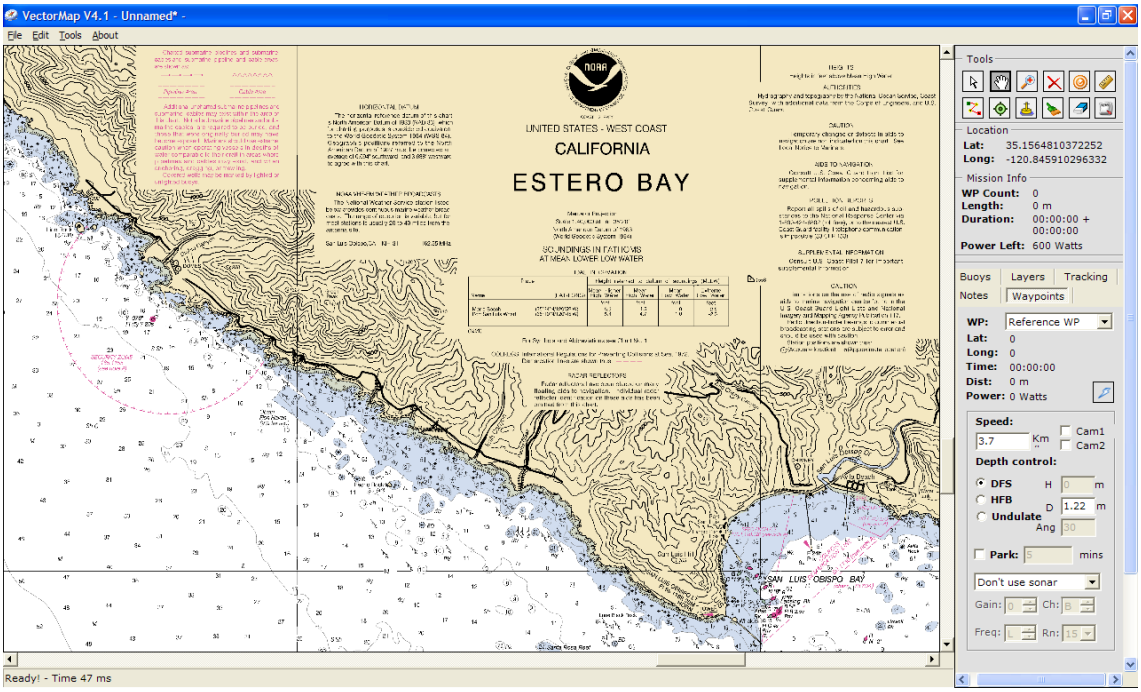


Figure 7.2: VectorMap software with the NOAA Raster Navigational Chart of the Estero Bay area where the missions took place.

The VectorMap GUI allows for quick access to all necessary tools for planning a mission. After loading a chart in VectorMap, waypoints can be added by navigating the cursor to the desired location on the map and left clicking the mouse. After adding the first waypoint, the distance and angle from the first to the second waypoint is displayed on the screen to allow the user to accurately determine the heading and distance to the new waypoint.

Besides using the GUI, the ASCII mission files can be created using a text editor when the waypoint properties such as latitude, longitude, speed, and depth are known. Fig. 7.3 is a sample mission file called "MTSP-1" as indicated under the *MISSION NAME* section. This mission uses the NOAA raster map "18703-1" as indicated under the *FILES* section which specifies the directory where the map is located.

```
MISSION FILE VERSION V1.0
2X ;Latitude ;Longitude ;Distance ;Heading ;Command String
FILES
C:\maps\18703-1.KAP
MISSION NAME
MTSP-1
START
1;35.1701128117688;-120.7413124037010;0;0;D0 P0 VC0 S2;0
2;35.1700499970337;-120.7412832254800;7.4651819;159.207701;D0 P0 VC0 S2;0
3;35.1699753033922;-120.7412747032370;8.3348159;174.671637;D0 P0 VC0 S2;0
4;35.1699810583344;-120.7413128331160;3.521533;280.461139;D0 P0 VC0 S2;0
5;35.1699335175075;-120.7413705619520;7.4423499;224.787693;D0 P0 VC0 S2;0
6;35.1699354922249;-120.7414322546210;5.6072939;272.242419;D0 P0 VC0 S2;0
7;35.1700391153866;-120.7414794903080;12.286304;339.563486;D0 P0 VC0 S2;0
8;35.1700328420136;-120.7413973882980;7.4891049;95.3401622;D0 P0 VC0 S2;0
9;35.1700663870082;-120.7413865384040;3.8550670;14.8098304;D0 P0 VC0 S2;0
10;35.1700529222076;-120.7413495426220;3.6779907;114.00045;D0 P0 VC0 S2;0
11;35.1701128117688;-120.7413124037010;7.4600834;26.881048;D0 P0 VC0 S2;0
END
```

Figure 7.3: Sample mission file.

Under the *START* section, all the waypoints and their properties are listed. The format for each line is: Waypoint Number; Latitude; Longitude; Distance; Heading; D# P# VC# S# SS; 0, where,

- D = Depth from surface
- H = Height from bottom
- P = Amount of time the vehicle will park
- VC = Video Camera; 0=none, 1=1 camera, 2=2 camera, 3=both cameras
- S = Side Scan; if side scan is enabled, the format is SS Gain Range Channel Frequency 1

## 7.2 Connecting to the Iver2

Establishing communication between a notebook and the Iver2 is accomplished using a portable battery powered WiFi router/access point. The user must first connect the operating computer to the portable WiFi access point by setting up the WiFi access point as a wireless network. All vehicles are configured to connect to the access point as soon as it is powered on. Through the WiFi connection, a

user can connect up to 127 different Iver2 AUVs using Windows Remote Desktop Connection from a single notebook computer. The connection also allows a user to easily transfer data such mission files or logs between a notebook and the AUV.

## 7.3 Running a mission

The Underwater Vehicle Console (UVC) resides in the Iver2 and is responsible for running missions (Fig. 7.4). The first step is to remotely connect to the vehicle to open the UVC to load a VectorMap mission file. The mission file should be stored in the shared folder called *Missions* on the Iver2. Once the mission is loaded, the UVC can be used to adjust the properties of a mission and also alter the behaviour of the vehicle. Under the setup menu, the *waypoint properties* allows the user to specify the distance at which the Iver2 can continue to the next waypoint when the current destination waypoint is within those values. For this thesis, the "waypoint success radius" was set to 4 meters which is the minimum value the UVC will allow (Fig. 7.5).

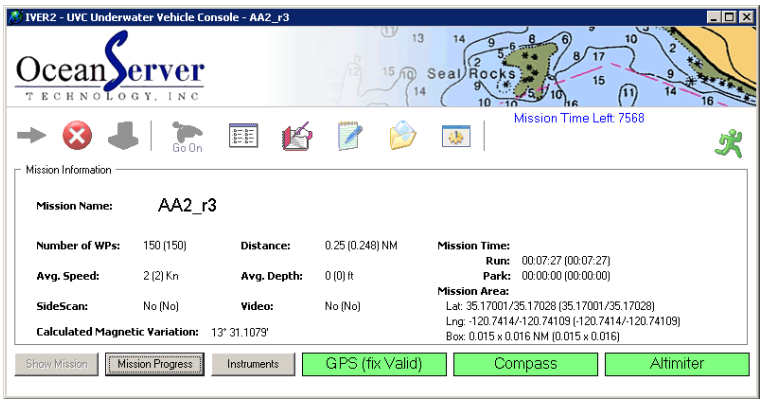


Figure 7.4: UVC software.

At this stage, the vehicle's devices should be checked using the Instruments panel before launching the vehicle in the water. The Iver2's devices such as the GPS, batteries, compass, altimter, and YSI sensor can be checked anytime while the vehicle is within WiFi range. The GPS reading provides the vehicle's current latitude and longitude location and also includes the following:

- **True Heading** - heading compensated for magnetic variation
- **Magnetic Variation** - the amount of error between actual North and the vehicle's North
- **Current Speed** - vehicle's speed according to the GPS in Knots
- **Number of Satellites** - provides the amount of satellites associated with the vehicle's GPS signal

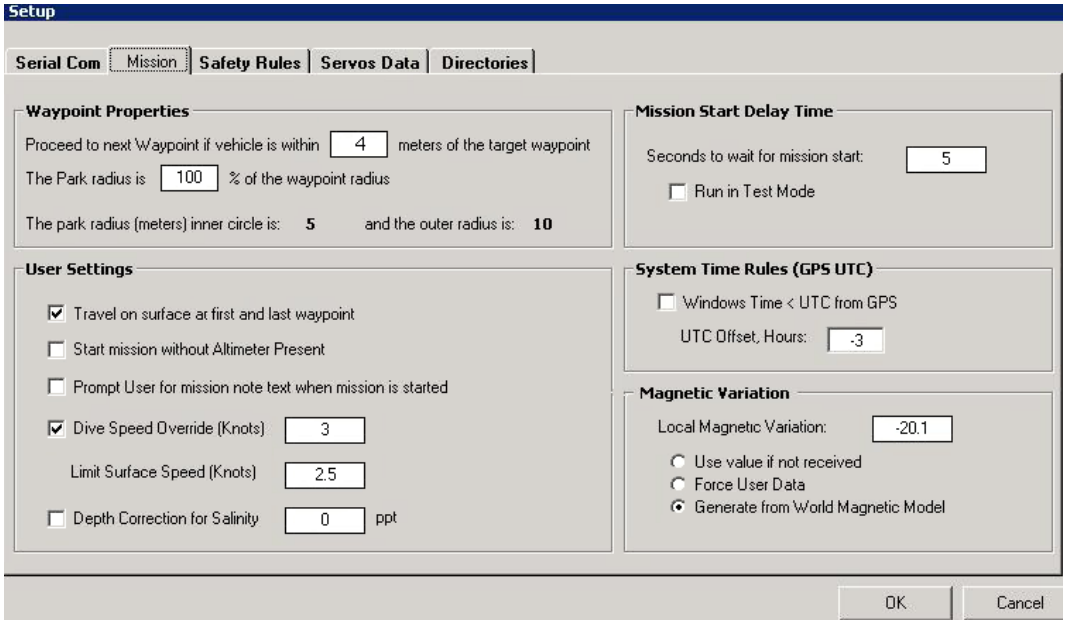


Figure 7.5: UVC software setup screen.

- **Data Age** - the time in seconds that has passed since the new data has arrived from the GPS

Next, the vehicle can be placed in the water in a safe place and driven to a starting location. The UVC has a manual mode that allows the user to manually control the Iver2 AUV when the vehicle is within WiFi range. The propeller can be forced to move the vehicle in the forward direction or in the reverse direction. The yaw is controlled by moving the top and bottom fins together.

After driving the vehicle to the start position, the GPS, compass, and depth sensors should be checked to be receiving valid data. Once the mission starts, the Iver2's progress can be checked only if the vehicle is within WiFi range and it is running a surface mission. For this reason, mission files were created so that task points were located within a square with side lengths of 35 meters (within WiFi range) and the vehicle operated at the surface throughout the mission.

## 7.4 Post-mission

While the vehicle is running a mission, it is continuously logging data that is time and GPS referenced. After a mission ends, data can be transferred from the vehicle to the notebook for analysis. The log data can be overlayed on the mission in VectorMap. After the log file is loaded, a red line is overlayed on the mission map indicating the vehicle's actual position during the mission (Fig. 7.6). The data of the log file is placed in an excel sheet at the bottom of VectorMap. Selecting any

point on the vehicles actual position will activate a 'X' and the record corresponding to that point will be highlighted. To further analyze the log data, the file can be opened with Microsoft Excel.

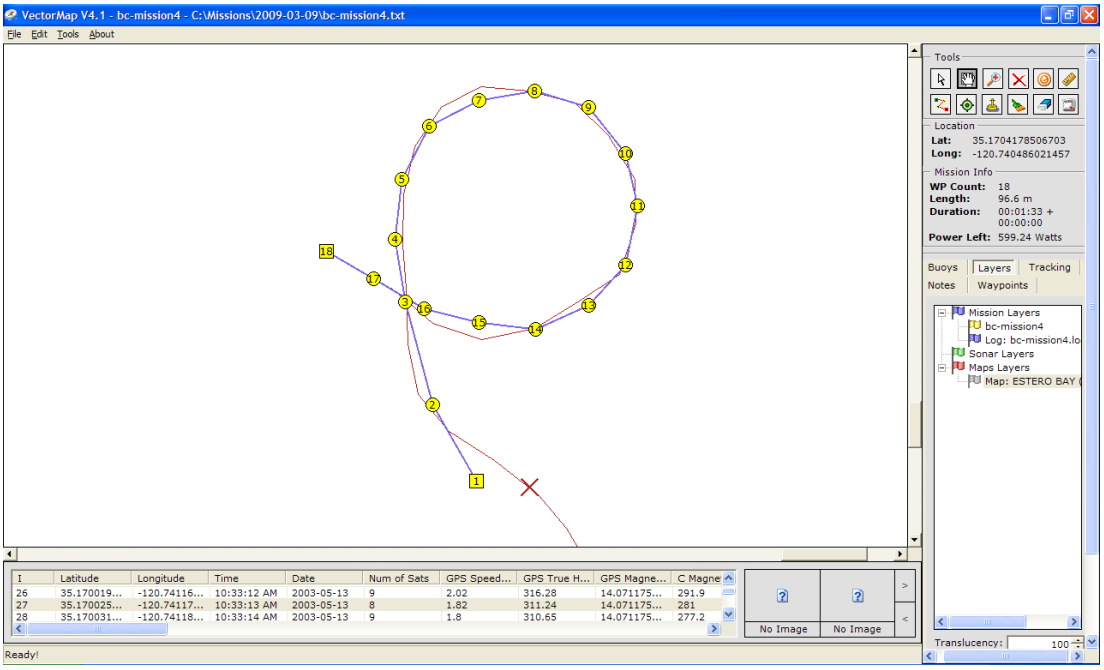


Figure 7.6: Data of the log file overlayed on the mission map.

# Chapter 8

# Experimental Results

Mission files were created based on the sequences generated by the different algorithms using Matlab and were tested on the Iver2 AUV. The results from running the experiments were analyzed based on following criteria:

$$T_{\max} = \max C_{\exp}(S_i); \quad T_{\text{avg}} = \frac{\sum_{i=1}^{n} C_{\exp}(S_i)}{n}; \quad D_{\text{avg}} = \frac{\sum_{j=1}^{m} D_{\min}(d_j)}{m}.$$

where $C_{exp}$ is the time taken by the Iver2 AUV to traverse the sequence $S_i$ during a mission and $D_{min}$ is the minimum distance between a task point and the line indicating the actual position of the AUV during the mission as shown in Fig. 8.1. Note that for Experiment 1, $T_{\max} = T_{\text{avg}}$ since there is only one vehicle, and therefore, it is denoted as $T_{\exp}$.

## 8.1 Control Architecture

Before describing the results from field tests, the limitations on the control architecture of the Iver2 AUV must be addressed. The Iver2 AUV control architecture is based on the UVC developed by the Ocean Server Technology Inc. The UVC provides an interface to the Iver2 AUV's sensors, motors, and control processes through the Remote Desktop Connection. As described in Section 7.3, the UVC declares victory on the approaching waypoint and will move to the next waypoint when it has reached the "waypoint success radius" which was set to 4 meters (minimum allowed value on the UVC).

To execute a planned path, the UVC uses the sequence of waypoints listed in the ASCII mission file. The goal is to drive the AUV to the waypoint latitude and longitude coordinate but unfortunately, the UVC will only track the waypoint until it is within 4 meters, at which point it will start tracking the next waypoint. Due to the Iver2's method of control, the AUV was not actually able to track waypoints precisely, which can be seen in the experimental results.

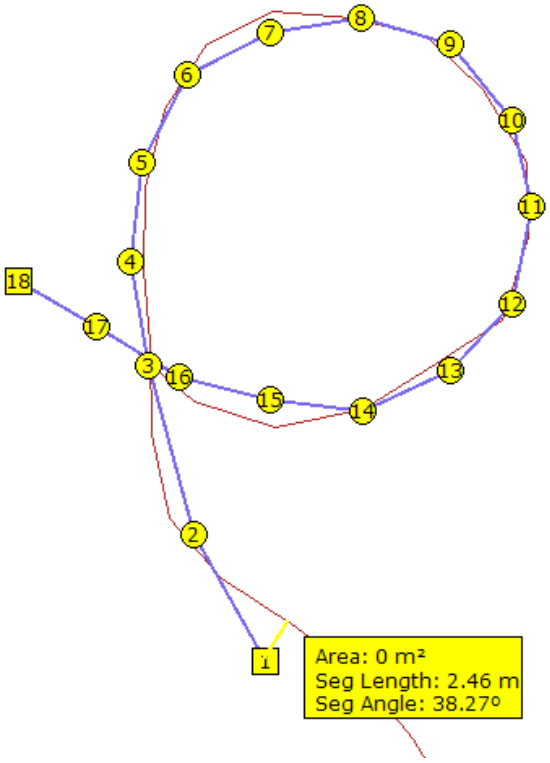


Figure 8.1: Minimum distance between a task point and the line indicating the actual position of the AUV during a mission.

## 8.2 Experiment 1 - Traveling Salesman Problem

The first set of experiments were conducted on 3 datasets, each containing 10 task points generated randomly and uniformly inside a square with side lengths of 25 meters. Consider one particular trial using the dataset shown in Fig. 8.2. The task positions were converted to latitude and longitude as shown in Table 8.1 to be used in the mission file.

The first method solves the traveling salesman problem without considering the curvature constraints of the vehicle or the effects of ocean current (Fig 8.3). The second method takes the sequence of points generated by the first method and tries to find a feasible path taking into consideration the curvature constraints of the vehicle using the "alternating algorithm" (Fig 8.4). The third method uses the proposed algorithm which considers the curvature constraints of the vehicle as well as the effect of ocean currents in generating the sequence for the vehicle (Fig 8.5). Results from field tests are summarized in Table 8.2.

On average, the proposed algorithm reduced the mission time by 34% and reduced the average minimum distance to task points by 31% over the "alternating algorithm". Although the TSP solution was 50% faster than the proposed solution, the paths generated were not feasible for the Iver2 AUV which had a turning radius of 6 meters. This resulted in the Iver2 AUV travelling at times very far from the

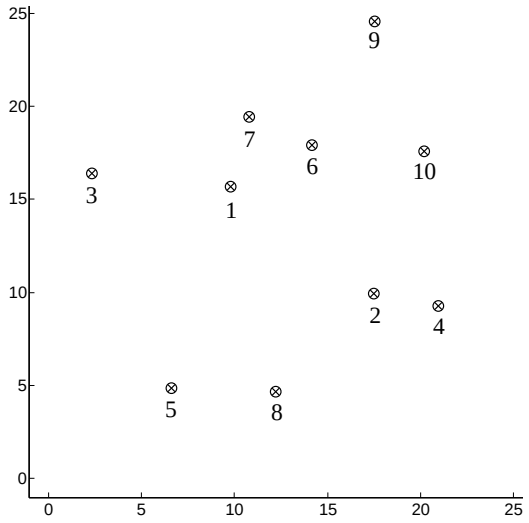


Figure 8.2: Dataset of 20 task points randomly generated.

Table 8.1: Converting to latitude and longitude.

| x (m) | y (m) | Latitude | Longitude |
|---|---|---|---|
| 9.7976 | 15.683 | 35.1700328420136 | -120.7413973882980 |
| 17.477 | 9.9296 | 35.1699810583344 | -120.7413128331160 |
| 2.341 | 16.38 | 35.1700391153866 | -120.7414794903080 |
| 20.94 | 9.2902 | 35.1699753033922 | -120.7412747032370 |
| 6.631 | 4.867 | 35.1699354922249 | -120.7414322546210 |
| 14.143 | 17.914 | 35.1700529222076 | -120.7413495426220 |
| 10.783 | 19.41 | 35.1700663870082 | -120.7413865384040 |
| 12.234 | 4.6476 | 35.1699335175075 | -120.7413705619520 |
| 17.516 | 24.568 | 35.1701128117688 | -120.7413124037010 |
| 20.166 | 17.589 | 35.1700499970337 | -120.7412832254800 |

Table 8.2: Field test results from three randomly generated datasets of 10 task points for one vehicle.

| | $T_{\text{exp}}$ (s) | $D_{\text{avg}}$ (m) |
|---|---|---|
| TSP | 104.3 | 1.44 |
| Alternating Algorithm | 315.3 | 0.90 |
| Proposed Algorithm | 207 | 0.62 |

desired task point, with a maximum value of 4.05 meters. The paths generated by the "alternating algorithm" improved this factor but at the expense of increasing the mission time. The numerous loops created by the "alternating algorithm" created complex missions for the Iver2 AUV and resulted in longer missions. The

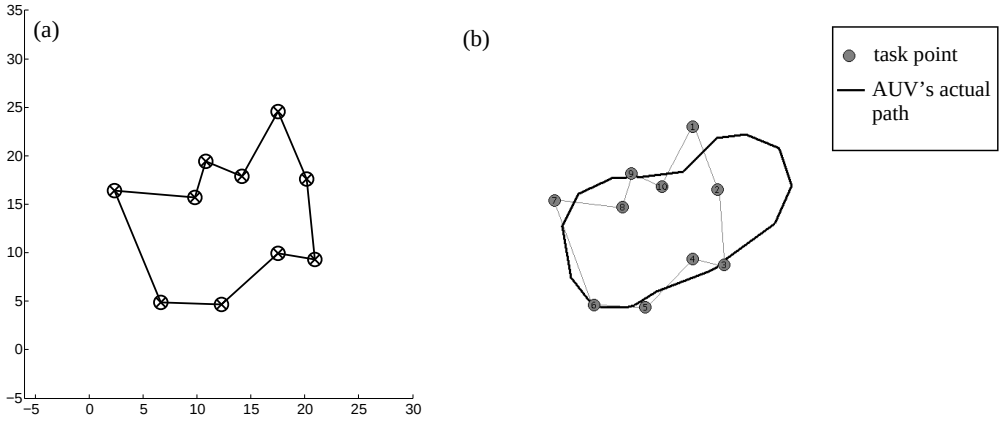


Figure 8.3: (a) Paths generated for the TSP from Matlab. (b) Field test results - $T_{\text{exp}} = 85$ s and $D_{\text{avg}} = 1.71$ m.

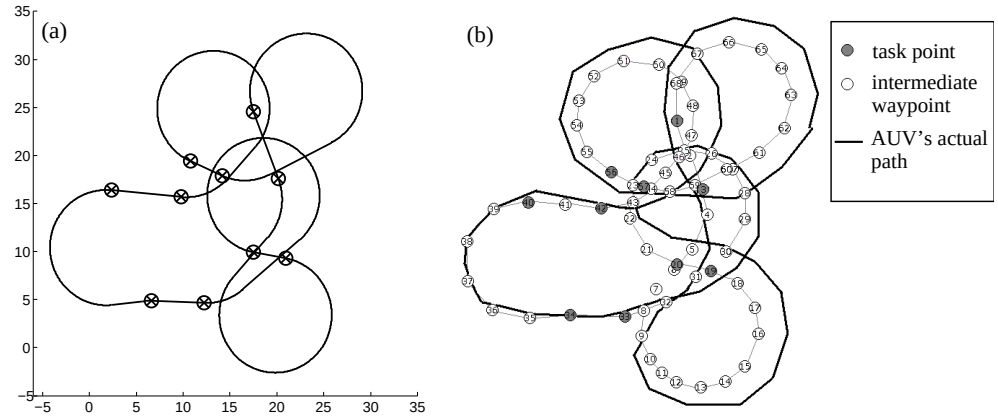


Figure 8.4: (a) Paths generated using the 'alternating algorithm' from Matlab. (b) Field test results - $T_{\text{exp}} = 290$ s and $D_{\text{avg}} = 0.88$ m.

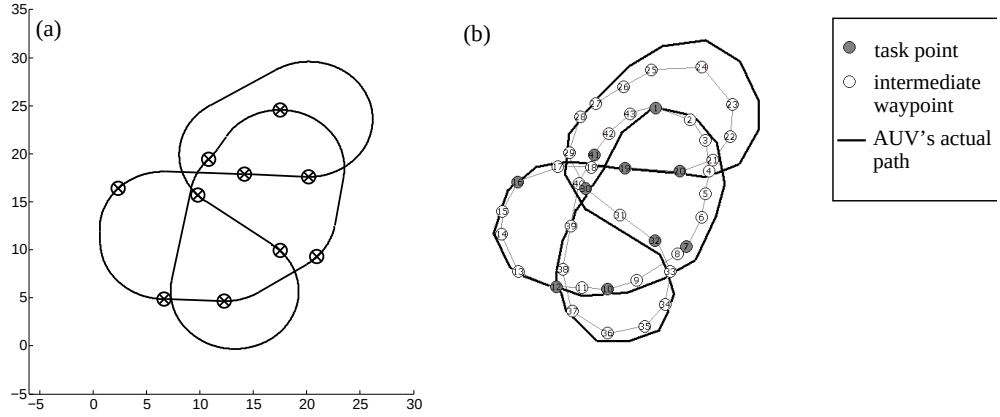


Figure 8.5: (a) Paths generated using the proposed algorithm from Matlab. (b) Field test results - $T_{\text{exp}} = 209$ s and $D_{\text{avg}} = 0.42$ m.

mission file had 69 lines of code, reflecting the number of intermediate waypoints used to guide the Iver2 AUV to follow the desired path. The proposed algorithm had 44 lines of code and this reduced complexity resulted in shorter missions with better performance.

## 8.3 Experiment 2 - Multiple Traveling Salesman Problem

The second set of experiments were conducted on 3 datasets, each containing 20 task points generated randomly and uniformly inside a square with side lengths of 35 meters. These task points were allocated to 3 vehicles, similar to the multiple traveling salesmen problem. Note that the experiments were conducted using only one Iver2 AUV. The three sequences generated from the different algorithms were run one at time.

Consider one particular trial using the dataset shown in Fig. 8.6. The first method solves the multiple traveling salesman problem without considering the curvature constraints of the vehicle (Fig 8.7). The second method takes the sequences generated by the first method and tries to find feasible paths for each vehicle using the "alternating algorithm" (Fig 8.8). The third method uses the proposed algorithm which considers the curvature constraints of the vehicle in generating the sequence for the vehicle (Fig 8.9). Results from field tests are summarized in Table 8.3.

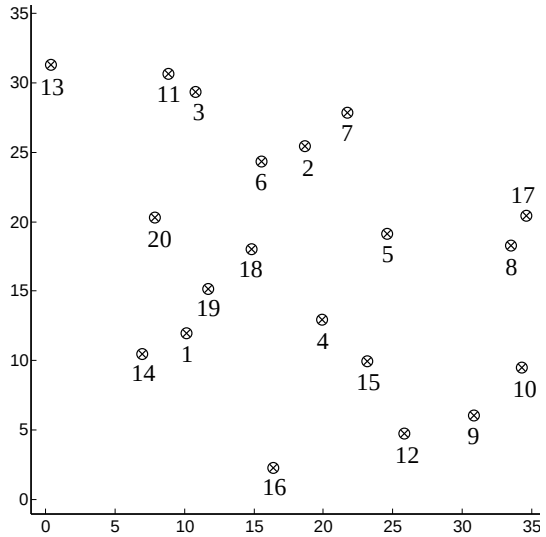


Figure 8.6: Dataset of 20 task points randomly generated.

On average, the proposed algorithm reduced $T_{\max}$ by 47% and reduced $D_{\text{avg}}$ by 34% over the "alternating algorithm". Again, the TSP solution was able to produce results 39% faster than the proposed algorithm, but the average distance to task

Table 8.3: Field test results from three randomly generated datasets of 20 task points for 3 vehicles.

| | $T_{\max}$ (s) | $T_{\text{avg}}$ (s) | $D_{\text{avg}}$ (m) |
|---|---|---|---|
| TSP | 89.3 | 78 | 1.65 |
| Alternating Algorithm | 279 | 252 | 1.35 |
| Proposed Algorithm | 145.3 | 133.5 | 0.84 |

point is 49% larger. In one instance, the Iver2 AUV only got within 7.11 meteres from a task point when using the TSP sequence of points. The largest distance the AUV got to a task point was 3.53 meters using the "alternating algorithm" and 2.71 meters using the proposed algorithm.

The proposed algorithm also performed better with respect to overall mission time when compared to the "alternating algorithm" because paths were in general simpler with less loops. The "alternating algorithm" is based on the sequence of points generated by the solving the Euclidean TSP which tends to schedule closely spaced points in a successive order. Similar to simulation results from Matlab, the Iver2 AUV was not able to drive from one point to another point nearby without long maneuvers when the orientation of the vehicle was not "ideal". This resulted in additional loops which are harder to execute on the Iver2 AUV than straight paths, leading to longer mission times.

Note that all experiments were conducted on the same day in an attempt to test the different algorithms against each other in similar conditions. However, ocean currents are constantly changing in magnitude and direction. Before the mission, real-time information regarding the ocean current along the central California coast was retrieved from the California Polytechnic State University's Marine Science Research and Education Pier. At 15:00 UTC, the ocean current was measured to be going 0.1515 m/s at 3.9078 radians from the North and these values were used in the proposed algorithm to create paths for the Iver2 AUV. Unfortunately, the ocean currents had changed to 0.1248 m/s at 3.7671 radians from the North by the time the first experiment was conducted and continued changing throughout the course of the experiments. Because all experiments were conducted on one AUV, experiments were conducted sequentially and the ocean conditions were not identical. In an ideal experiment, all test cases would be run at the same time but since this was not possible, the three methods were alternated such that experiments using the same dataset were conducted as close as possible in time.

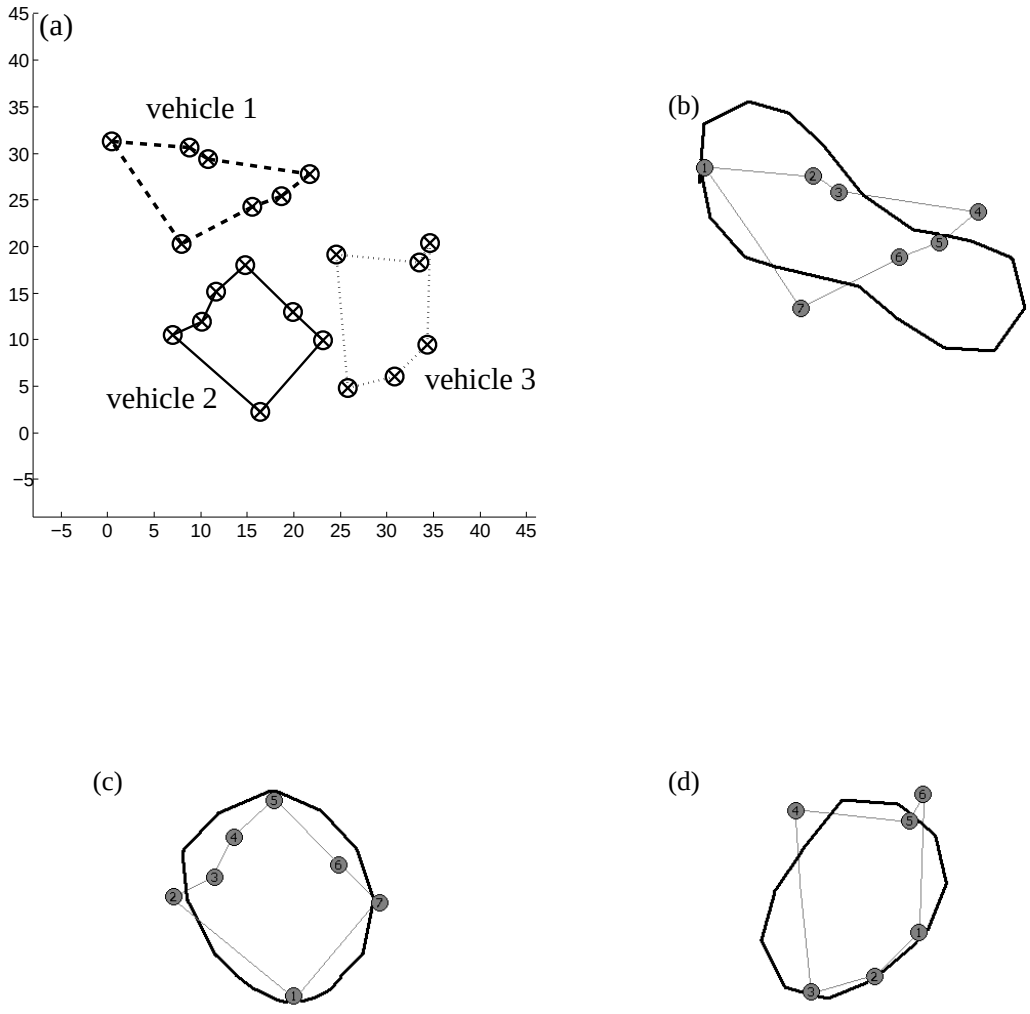


Figure 8.7: Paths generated for the TSP from Matlab. (b) Field test results for vehicle 1. (c) Field test results for vehicle 2. (d) Field test results for vehicle 3.

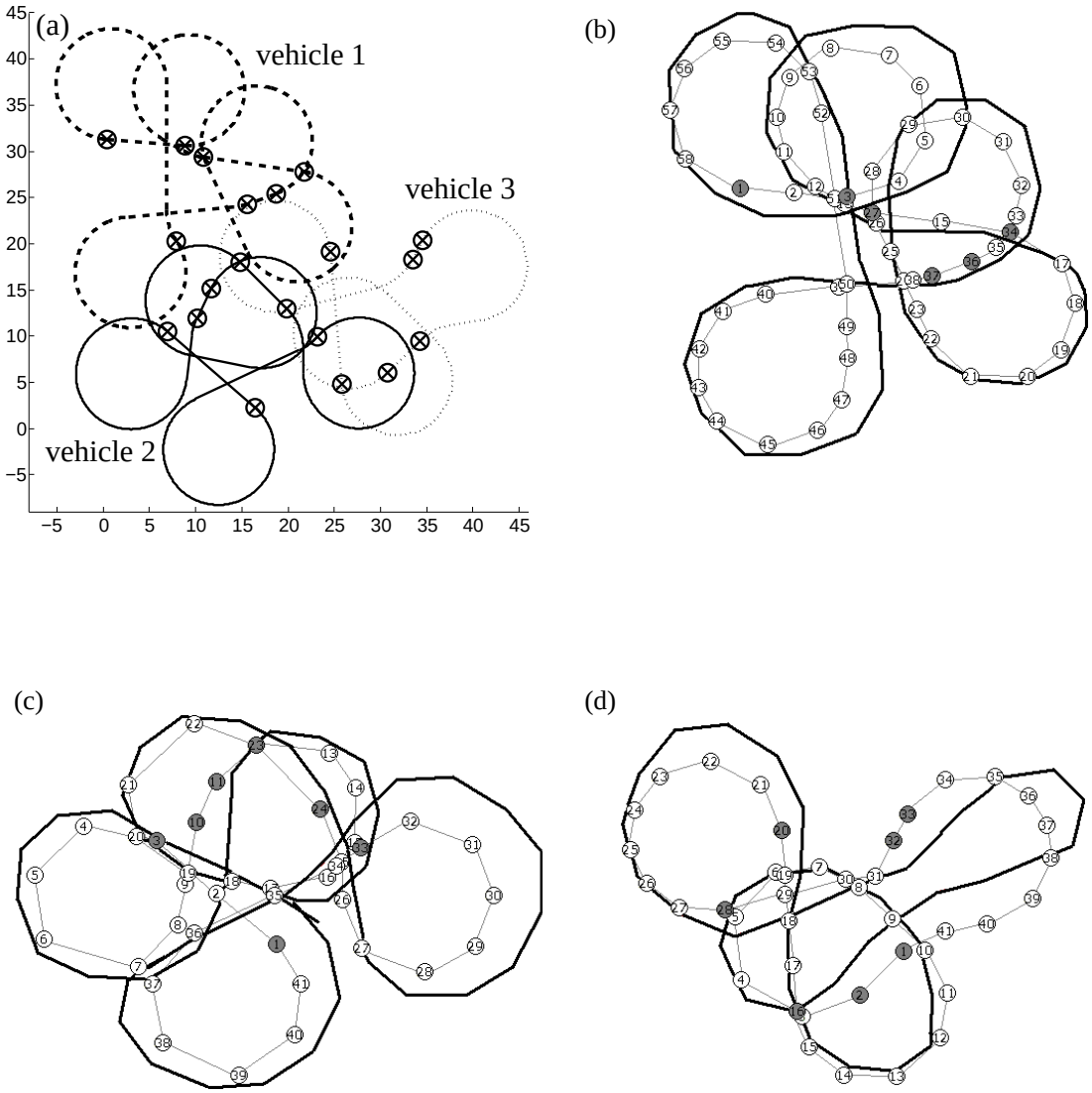


Figure 8.8: (a) Paths generated using the "alternating algorithm" from Matlab. (b) Field test results for vehicle 1. (c) Field test results for vehicle 2. (d) Field test results for vehicle 3.

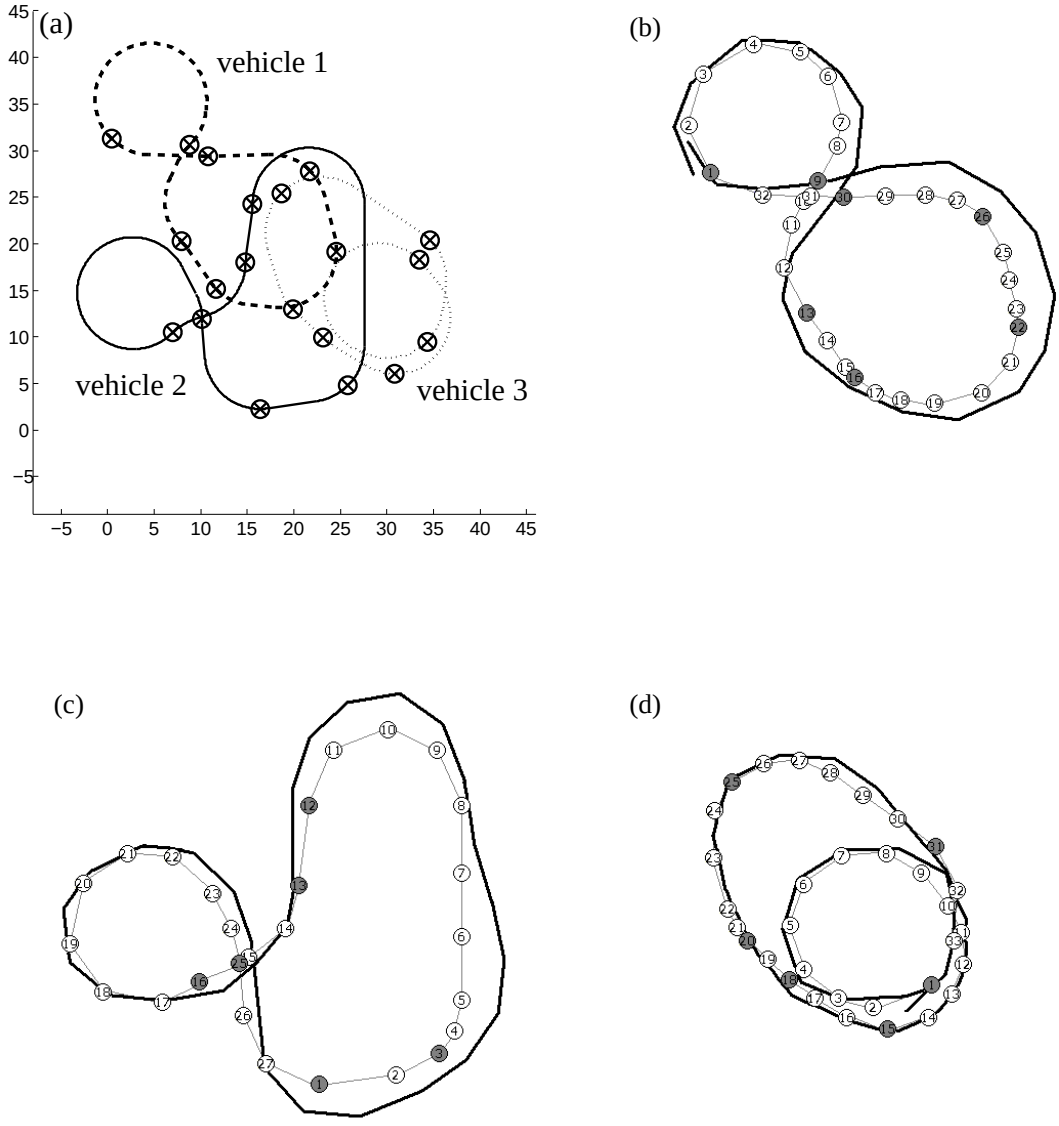


Figure 8.9: (a) Paths generated using the proposed algorithm from Matlab. (b) Field test results for vehicle 1. (c) Field test results for vehicle 2. (d) Field test results for vehicle 3.

# Chapter 9

# Conclusion

AUVs are seen as ideal platforms for oceanographic research such as ocean sampling, mapping, and monitoring. The main advantage of using AUVs is that they are a cost-effective way for collecting data on a large spatial or temporal scale. Task allocation problems naturally arise in these applications where the objective is to optimally assign task points to a given set of vehicles. The feature that differentiates this task allocation problem from similar problems previously studied in the literature is that there are constraints on the motion of the vehicle and the presence of ocean currents. Here, the objective is to minimize the time required to navigate between task points through a current field.

This thesis addresses the task allocation of closely spaced targets for vehicles that follow paths of bounded curvature in the presence of constant ocean currents. The proposed algorithm is based on using a bidding scheme to allocate tasks to multiple AUVs while using the Dubins set to calculate the path costs for vehicles with non-holonomic constraints. Bid costs are calculated using a lower order model created from the 6-DOF non-linear model to reduce the complexity of the algorithm.

The proposed algorithm was developed in Matlab and tested in simulations. Simulations using the full non-linear model of the REMUS AUV indicate that the proposed algorithm yield better performance for dense sets of points when compared to the "alternating algorithm". It is shown that solutions based on computing Euclidean tours that do not have curvature constraints have extra loops when task points are close together relative to the turning radius of the vehicle.

To validate the proposed algorithm in a real world application, the Iver2 AUV was used for testing at the Avila Pier in California. The Iver2 AUV was equipped with the VectorMap software for mission planning and UVC for mission execution. The VectorMap software generated mission files specifying the latitude and longitude of a sequence of waypoints. It also allowed for waypoint properties such as speed and depth to be specified. A WiFi access point was used to connect the Iver2AUV with an operating computer, communicating with the Iver2 AUV via a wireless Ethernet connection. Localization and status information was logged on the Iver2's main processor on-board during a mission.

Analysis of the log files indicated that the proposed algorithm outperformed the "alternating algorithm" with respect to the overall mission time as well as the average distance to task point. The proposed algorithm produced paths through a set of task points that were feasible for the Iver2 AUV to track closely, even in the presence of ocean currents.

## 9.1 Future Work

The developments presented in this thesis may be extended by future research in several areas. First, this thesis assumes the AUVs travel through water that has a known uniform velocity. Instantaneous currents at the location of vehicles are available for certain bodies of water through websites such as the NASA Jet Propulsion Lab and the Center for Coastal Marine Sciences at the California Polytechnic State University. This information could be integrated into future developments for dynamic task allocation. By utilizing real-time information about the ocean currents, the task allocation algorithm can update the paths for each vehicle during a mission as new information is received. This will also allow the system to adapt to dynamic or unknown scenarios which could result in new task points being generated.

Future developments of the task allocation algorithm could also focus on precedence constraints. Adaptive sampling strategies can be used to change the priority of task points depending on prior measurements or analysis. When using multiple AUVs for environmental monitoring of large bodies of water, predictive models and maps are created by repeated measurements. However, because the sampling volume could be quite large, only a limited number of measurements are usually available. A task allocation algorithm that can handle ordering constraints on task points could prioritize the task points according to their impact on these predictive models and maps.

Another extension of is to implement the task allocation algorithm on a distributed architecture. This development would eliminate the requirement of a centralized controller and allow distribution of the computational effort of calculating path costs across all of the robots in the system.

Finally, the proposed algorithm for path planning can be extended to perform trajectory planning. A path defines the sequence of task points for the AUV to follow without regard for timing. A trajectory is a path parameterized by time. With a trajectory planner, the algorithm should be capable of performing collision checks as well as handling time-indexed waypoints.

# APPENDICES

# Appendix A

# REMUS AUV Dynamic Model

## A.1 Vehicle Kinematics

The equations of motion will be defined in terms of linear and angular motion components about the body-fixed coordinate system. The origin of the vehicle body-fixed coordinate system is chosen to coincide with the center of buoyancy. The motion of the body-fixed frame of reference is described relative to an inertial-fixed reference frame. The general motion of the vehicle in six degrees of freedom can be described by the following vectors:

$$\begin{aligned} \eta_1 &= [x \;\; y \;\; z]^T; & \eta_2 &= [\phi \;\; \theta \;\; \psi]^T \\ \nu_1 &= [u \;\; v \;\; w]^T; & \nu_2 &= [p \;\; q \;\; r]^T \\ \tau_1 &= [X \;\; Y \;\; Z]^T; & \tau_2 &= [K \;\; M \;\; N]^T \end{aligned} \tag{A.1}$$

where $\eta$ describes the position and orientation of the vehicle with respect to the inertial-fixed reference frame, $\nu$ the translational and rotational velocities of the vehicle with respect to the body-fixed frame, and $\tau$ the total forces and moments acting on the vehicle with respect to the body-fixed reference frame. See Fig. A.1 for a diagram of the vehicle coordinate system. Note that $\phi$, $\theta$, and $\psi$ are Euler angles used to define the relative orientation of the AUV.

The following direction cosine matrix relates translational velocities between body-fixed and inertial-fixed coordinates:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{J}_1(\eta_2) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A.2}$$

where

$$\mathbf{J}_1(\eta_2) = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

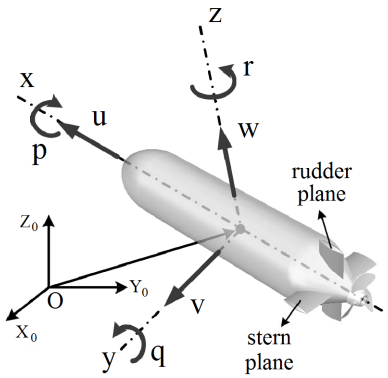


Figure A.1: Inertial and body-fixed coordinate systems and motion representations.

Note that $\mathbf{J}_1(\eta_2)$ is orthogonal:

$$(\mathbf{J}_1(\eta_2))^{-1} = (\mathbf{J}_1(\eta_2))^T \tag{A.3}$$

The second direction cosine matrix relates rotational velocities between body-fixed and inertial-fixed coordinates:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{J}_2(\eta_2) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{A.4}$$

where

$$\mathbf{J}_2(\eta_2) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}$$

The vehicle position and orientation at a given time can be found by integrating (A.2) and (A.4) respectively. The next step is to investigate the vehicle rigid body dynamics in order to find $\nu_1$ and $\nu_2$, which are necessary to perform the calculations in (A.2) and (A.4).

## A.2 Vehicle Rigid-Body Dynamics

The locations of the vehicle centers of gravity and buoyancy are defined in terms of the body-fixed coordinate system as follows:

$$\mathbf{r}_G = \begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix}, \qquad \mathbf{r}_B = \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{A.5}$$

Given that the origin of the body-fixed coordinate system is located at the center of buoyancy, the following represent the equations of motion for a rigid body in six degrees of freedom, defined in terms of the body-fixed coordinates:

$$\begin{aligned}
m_v\left[\dot{u} - vr + wq - x_g\left(q^2 + r^2\right) + y_g\left(pq - \dot{r}\right) + z_g\left(pr + \dot{q}\right)\right] &= \sum X_{ext} \\
m_v\left[\dot{v} - wp + ur - y_g\left(r^2 + p^2\right) + z_g\left(qr - \dot{p}\right) + x_g\left(qp + \dot{r}\right)\right] &= \sum Y_{ext} \\
m_v\left[\dot{w} - uq + vp - z_g\left(p^2 + q^2\right) + x_g\left(rp - \dot{q}\right) + y_g\left(rq + \dot{p}\right)\right] &= \sum Z_{ext} \\
I_{xx}\dot{p} + \left(I_{zz} - I_{yy}\right)qr - \left(\dot{r} + pq\right)I_{xz} + \left(r^2 - q^2\right)I_{yz} + \left(pr - \dot{q}\right)I_{xy} & \\
+m_v\left[y_g\left(\dot{w} - uq + vp\right) - z_g\left(\dot{v} - wp + ur\right)\right] &= \sum K_{ext} \\
I_{yy}\dot{q} + \left(I_{xx} - I_{zz}\right)rp - \left(\dot{p} + qr\right)I_{xy} + \left(p^2 - r^2\right)I_{xz} + \left(qp - \dot{r}\right)I_{yz} & \\
+m_v\left[z_g\left(\dot{u} - vr + wq\right) - x_g\left(\dot{w} - uq + vp\right)\right] &= \sum M_{ext} \\
I_{zz}\dot{r} + \left(I_{yy} - I_{xx}\right)pq - \left(\dot{q} + rp\right)I_{yz} + \left(q^2 - p^2\right)I_{xy} + \left(rq - \dot{p}\right)I_{xz} & \\
+m_v\left[x_g\left(\dot{v} - wp + ur\right) - y_g\left(\dot{u} - vr + wq\right)\right] &= \sum N_{ext}
\end{aligned} \tag{A.6}$$

where $m_v$ is the vehicle mass. The first three equations represent translational motion, the second three represent the rotational motion.

Note that these equations neglect the zero-valued center of buoyancy terms. These equations can be further simplified with the assumption that the vehicle products of inertia are small, and that $y_g$ is small compared to the other terms [8]:

$$\begin{aligned}
m_v\left[\dot{u} - vr + wq - x_g\left(q^2 + r^2\right) + z_g\left(pr + \dot{q}\right)\right] &= \sum X_{ext} \\
m_v\left[\dot{v} - wp + ur + z_g\left(qr - \dot{p}\right) + x_g\left(qp + \dot{r}\right)\right] &= \sum Y_{ext} \\
m_v\left[\dot{w} - uq + vp - z_g\left(p^2 + q^2\right) + x_g\left(rp - \dot{q}\right)\right] &= \sum Z_{ext} \\
I_{xx}\dot{p} + \left(I_{zz} - I_{yy}\right)qr + m_v\left[-z_g\left(\dot{v} - wp + ur\right)\right] &= \sum K_{ext} \\
I_{yy}\dot{q} + \left(I_{xx} - I_{zz}\right)rp + m_v\left[z_g\left(\dot{u} - vr + wq\right) - x_g\left(\dot{w} - uq + vp\right)\right] &= \sum M_{ext} \\
I_{zz}\dot{r} + \left(I_{yy} - I_{xx}\right)pq + m_v\left[x_g\left(\dot{v} - wp + ur\right)\right] &= \sum N_{ext}
\end{aligned} \tag{A.7}$$

## A.3 Vehicle Mechanics

In the vehicle equations of motion, external forces and moments are described in terms of vehicle coefficients. The external forces and moments acting on an underwater vehicle can be classified as added mass, hydrodynamic damping, restoring forces, currents, thruster/propeller forces, and control surface/fin forces. Combin-

ing these forces and moments on the vehicle yields the following equations:

$$\begin{aligned}
\sum X_{ext} &= X_{HS} + X_{u|u|}u|u| + X_{\dot{u}}\dot{u} + X_{wq}wq + X_{qq}qq + X_{vr}\dot{v}r + X_{rr}rr \\
&\quad + X_{prop} \\
\sum Y_{ext} &= Y_{HS} + Y_{v|v|}v|v| + Y_{r|r|}r|r| + Y_{\dot{v}}\dot{v} + Y_{\dot{r}}\dot{r} + Y_{ur}ur + Y_{wp}wp \\
&\quad + Y_{pq}pq + Y_{uv}uv + Y_{uu\delta_r}u^2\delta_r \\
\sum Z_{ext} &= Z_{HS} + Z_{w|w|}w|w| + Z_{q|q|}q|q| + Z_{\dot{w}}\dot{w} + Z_{\dot{q}}\dot{q} + Z_{uq}uq + Z_{vp}vp \\
&\quad + Z_{rq}rq + Z_{uw}uw + Z_{uu\delta_s}u^2\delta_s \\
\sum K_{ext} &= K_{HS} + K_{p|p|} + K_{\dot{p}}\dot{p} + K_{prop} \\
\sum M_{ext} &= M_{HS} + M_{w|w|}w|w| + M_{q|q|}q|q| + M_{\dot{w}}\dot{w} + M_{\dot{q}}\dot{q} + M_{uq}uq \\
&\quad + M_{vp}vp + M_{rp}rp + M_{uw}uw + M_{uu\delta_s}u^2\delta_s \\
\sum N_{ext} &= N_{HS} + N_{v|v|}v|v| + N_{r|r|}r|r| + N_{\dot{v}}\dot{v} + N_{\dot{r}}\dot{r} + N_{ur}ur + N_{wp}wp \\
&\quad + N_{pq}pq + N_{uv}uv + N_{uu\delta_r}u^2\delta_r
\end{aligned} \tag{A.8}$$

The actual values of these coefficients are derived theoretically and experimentally in [8] and are listed in Appendix B and Appendix C.

## A.4 6-DOF Non-linear Model

The completion of the 6-DOF model requires the algebraic manipulation and combination of several equations of motion. When (A.8) is substituted into (A.7), and all acceleration terms are grouped together, the resulting system can then be represented in matrix form as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = [\mathbf{J}_3]^{-1} \begin{bmatrix} \Sigma X \\ \Sigma Y \\ \Sigma Z \\ \Sigma K \\ \Sigma M \\ \Sigma N \end{bmatrix} \tag{A.9}$$

where

$$\mathbf{J}_3 = \begin{bmatrix} m_v - X_{\dot{u}} & 0 & 0 & 0 & m_v z_G & 0 \\ 0 & m_v - Y_{\dot{v}} & 0 & -m_v z_G & 0 & m_v x_G - Y_{\dot{r}} \\ 0 & 0 & m_v - Z_{\dot{w}} & 0 & -m_v x_G - Z_{\dot{q}} & 0 \\ 0 & -m_v z_G & 0 & I_X - K_{\dot{p}} & 0 & 0 \\ m_v z_G & 0 & -m_v z_G - M_{\dot{w}} & 0 & I_Y - M_{\dot{q}} & 0 \\ 0 & m_v x_G - N_{\dot{v}} & 0 & 0 & 0 & I_Z - N_{\dot{r}} \end{bmatrix}$$

Integrating (A.9) provides the solution for the states given by $\nu$ in (A.1).

# Appendix B

# Tables of REMUS Parameters

Table B.1: STD REMUS Hull Parameters

| Parameter | Value | Units | Description |
|---|---|---|---|
| $\rho$ | $+1.03e+003$ | kg/m$^3$ | Seawater Density |
| $A_f$ | $+2.85e-002$ | m$^2$ | Hull Frontal Area |
| $A_p$ | $+2.26e-001$ | m$^2$ | Hull Projected Area (xz plane) |
| $S_w$ | $+7.09e-001$ | m$^2$ | Hull Wetted Surface Area |
| $\nabla$ | $+3.15e-002$ | m$^3$ | Estimated Hull Volume |
| $W$ | $+2.99e+002$ | N | Measured Vehicle Weight |
| $B$ | $+3.08e+002$ | N | Measured Vehicle Buoyancy |
| $B_{\text{est}}$ | $+3.17e+002$ | N | Estimated Hull Buoyancy |
| $x_{\text{cb(est)}}$ | $+5.54e-003$ | m | Est. Long. Center of Buoyancy |
| $c_{\text{d}}$ | $+3.00e-001$ | N/A | REMUS Axial Drag Coeff. |
| $c_{\text{dc}}$ | $+1.10e+000$ | N/A | Cylinder Crossflow Drag Coeff. |
| $c_{\text{yd}\beta}$ | $+1.20e+000$ | N/A | Hoerner Body Lift Coeff. |
| $x_{\text{cp}}$ | $-3.21e-001$ | N/A | Center of Pressure |
| $\alpha$ | $+3.59e-002$ | N/A | Ellipsoid Added Mass Coeff. |

Table B.2: Hull Coordinates for Limits of Integration

| Parameter | Value | Units | Description |
|---|---|---|---|
| $x_{\text{t}}$ | $-7.21e-001$ | m | Aft End of Tail Section |
| $x_{\text{t2}}$ | $-2.18e-001$ | m | Forward End of Tail Section |
| $x_{\text{f}}$ | $-6.85e-001$ | m | Aft End of Fin Section |
| $x_{\text{f2}}$ | $-6.11e-001$ | m | Forward End of Fin Section |
| $x_{\text{b}}$ | $+4.37e-001$ | m | Aft End of Bow Section |
| $x_{\text{b2}}$ | $+6.10e-001$ | m | Forward End of Bow Section |

Table B.3: Center of Buoyancy wrt Origin at Vehicle Nose

| Parameter | Value | Units |
|---|---|---|
| $x_{\text{cb}}$ | $+0.00e+000$ | m |
| $y_{\text{cb}}$ | $+0.00e+000$ | m |
| $z_{\text{cb}}$ | $+0.00e+000$ | m |

Table B.4: Center of Gravity wrt Origin at CB

| Parameter | Value | Units |
|---|---|---|
| $x_{\text{cg}}$ | $+0.00e+000$ | m |
| $y_{\text{cg}}$ | $+0.00e+000$ | m |
| $z_{\text{cg}}$ | $+0.00e+000$ | m |

Table B.5: Moments of Inertia wrt Origin at CB

| Parameter | Value | Units |
|---|---|---|
| $I_{xx}$ | $+1.77e-001$ | kg$\cdot$ m$^2$ |
| $I_{yy}$ | $+3.45e+000$ | kg$\cdot$ m$^2$ |
| $I_{zz}$ | $+3.45e+000$ | kg$\cdot$ m$^2$ |

Table B.6: REMUS Fin Parameters

| Parameter | Value | Units | Description |
|---|---|---|---|
| $S_{\text{fin}}$ | $+6.65e-003$ | m$^2$ | Planform Area |
| $b_{\text{fin}}$ | $+8.57e-002$ | m | Span |
| $x_{\text{finpost}}$ | $-8.19e-001$ | m | Moment Arm wrt Vehicle Origin at CB |
| $\delta_{max}$ | $+1.75e+000$ | rad | Maximum Fin Angle |
| $a_{\text{fin}}$ | $+5.14e+000$ | m | Max Fin Height Above Centerline |
| $c_{\text{mean}}$ | $+7.47e-002$ | m | Mean Chord Length |
| $t$ | $+6.54e-001$ | N/A | Fin Taper Ratio (Whicker-Felner) |
| $c_{\text{df}}$ | $+5.58e-001$ | N/A | Fin Crossflow Drag Coefficient |
| $AR_e$ | $+2.21e+000$ | N/A | Effective Aspect Ratio |
| $\overline{a}$ | $+9.00e-001$ | N/A | Lift Slope Parameter |
| $c_{L\alpha}$ | $+3.12e+000$ | N/A | Fin Lift Slope |

# Appendix C

# Tables of Combined Non-Linear Coefficients

Note that all coefficients are calculated for the STD REMUS hull profile as described in [8].

Table C.1: Axial Drag Coefficient

| Parameter | Value | Units |
|---|---|---|
| $X_{uu}$ | $-3.90e+000$ | kg/m |

Table C.2: Crossflow Drag Coefficients

| Parameter | Value | Units |
|---|---|---|
| $Y_{vv}$ | $-1.31e+003$ | kg/m |
| $Y_{rrd}$ | $+6.32e-001$ | kg$\cdot$ m/rad$^2$ |
| $Z_{ww}$ | $-1.31e+002$ | kg/m |
| $Z_{qqd}$ | $-6.32e-001$ | kg$\cdot$ m/rad$^2$ |
| $M_{wwd}$ | $+3.18e+000$ | kg/m |
| $M_{qq}$ | $-1.88e+002$ | kg$\cdot$ m$^2$/rad$^2$ |
| $N_{vvd}$ | $-3.18e+000$ | kg/m |
| $N_{rr}$ | $-9.40e+001$ | kg$\cdot$ m$^2$/rad$^2$ |

Table C.3: Rolling Resistance Coefficient

| Parameter | Value | Units |
|---|---|---|
| $K_{pp}$ | $-1.30e-001$ | kg$\cdot$ m$^2$/rad$^2$ |

Table C.4: Body Lift and Moment Coefficients

| Parameter | Value | Units |
|---|---|---|
| $Y_{vv}$ | $-2.86e+001$ | kg/m |
| $Z_{ww}$ | $-2.86e+001$ | kg/m |
| $M_{uwb}$ | $-4.47e+000$ | kg |
| $N_{uvb}$ | $+4.47e+000$ | kg |

Table C.5: Added Mass Coefficients

| Parameter | Value | Units |
|---|---|---|
| $X_{\dot{u}}$ | $-9.30e-001$ | kg |
| $X_{\dot{v}}$ | $+0.00e+000$ | kg |
| $X_{\dot{w}}$ | $+0.00e+000$ | kg |
| $X_{\dot{p}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $X_{\dot{q}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $X_{\dot{r}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Y_{\dot{u}}$ | $+0.00e+000$ | kg |
| $Y_{\dot{v}}$ | $-3.55e+001$ | kg |
| $Y_{\dot{w}}$ | $+0.00e+000$ | kg |
| $Y_{\dot{p}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Y_{\dot{q}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Y_{\dot{r}}$ | $+1.93e+000$ | kg$\cdot$ m/rad |
| $Z_{\dot{u}}$ | $+0.00e+000$ | kg |
| $Z_{\dot{v}}$ | $+0.00e+000$ | kg |
| $Z_{\dot{w}}$ | $-3.55e+001$ | kg |
| $Z_{\dot{p}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Z_{\dot{q}}$ | $-1.93e+000$ | kg$\cdot$ m/rad |
| $Z_{\dot{r}}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{\dot{u}}$ | $+0.00e+000$ | kg |
| $K_{\dot{v}}$ | $+0.00e+000$ | kg |
| $K_{\dot{w}}$ | $+0.00e+000$ | kg |
| $K_{\dot{p}}$ | $-7.04e-002$ | kg$\cdot$ m$^2$/rad |
| $K_{\dot{q}}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad |
| $K_{\dot{r}}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad |
| $M_{\dot{u}}$ | $+0.00e+000$ | kg |
| $M_{\dot{v}}$ | $+0.00e+000$ | kg |
| $M_{\dot{w}}$ | $-1.93e+000$ | kg |
| $M_{\dot{p}}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad |
| $M_{\dot{q}}$ | $-4.88e+000$ | kg$\cdot$ m$^2$/rad |
| $M_{\dot{r}}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad |
| $N_{\dot{u}}$ | $+0.00e+000$ | kg |
| $N_{\dot{v}}$ | $+1.93e+000$ | kg |
| $N_{\dot{w}}$ | $+0.00e+000$ | kg |
| $N_{\dot{p}}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad |
| $N_{\dot{q}}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad |
| $N_{\dot{r}}$ | $-4.88e+000$ | kg$\cdot$ m$^2$/rad |

Table C.6: Added Mass Force Cross-term Coefficients

| Parameter | Value | Units |
|---|---|---|
| $X_{uq}$ | $+0.00e+000$ | kg/rad |
| $X_{wq}$ | $-3.55e+001$ | kg/rad |
| $X_{qq}$ | $-1.93e+000$ | kg$\cdot$ m/rad |
| $X_{vr}$ | $+3.55e+001$ | kg/rad |
| $X_{rp}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $X_{rr}$ | $-1.93e+000$ | kg$\cdot$ m/rad |
| $X_{ur}$ | $+0.00e+000$ | kg/rad |
| $X_{wr}$ | $+0.00e+000$ | kg/rad |
| $X_{vq}$ | $+0.00e+000$ | kg/rad |
| $X_{pq}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $X_{qr}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Y_{vr}$ | $+0.00e+000$ | kg/rad |
| $Y_{vp}$ | $+0.00e+000$ | kg/rad |
| $Y_{rra}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Y_{rp}$ | $+0.00e+000$ | kg/rad |
| $Y_{pp}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Y_{up}$ | $+0.00e+000$ | kg/rad |
| $Y_{wr}$ | $+0.00e+000$ | kg/rad |
| $Y_{ura}$ | $-9.30e-001$ | kg/rad |
| $Y_{wp}$ | $+3.55e+001$ | kg/rad |
| $Y_{pq}$ | $+1.93e+000$ | kg$\cdot$ m/rad |
| $Y_{qr}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Z_{wq}$ | $+0.00e+000$ | kg/rad |
| $Z_{uqa}$ | $+9.30e-001$ | kg/rad |
| $Z_{qqa}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Z_{vp}$ | $-3.55e+001$ | kg/rad |
| $Z_{rp}$ | $+1.93e+000$ | kg/rad |
| $Z_{pp}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Z_{up}$ | $+0.00e+000$ | kg/rad |
| $Z_{wp}$ | $+0.00e+000$ | kg/rad |
| $Z_{vq}$ | $+0.00e+000$ | kg/rad |
| $Z_{pq}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $Z_{qr}$ | $+0.00e+000$ | kg$\cdot$ m/rad |

Table C.7: Added Mass K-Moment Cross-term Coefficients

| Parameter | Value | Units |
|---|---|---|
| $K_{wu}$ | $+0.00e+000$ | kg |
| $K_{uq}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{ww}$ | $+0.00e+000$ | kg |
| $K_{wq}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{qq}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad$^2$ |
| $K_{vv}$ | $+0.00e+000$ | kg |
| $K_{vr}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{vp}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{rr}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad$^2$ |
| $K_{rp}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad$^2$ |
| $K_{uv}$ | $+0.00e+000$ | kg |
| $K_{vw}$ | $+0.00e+000$ | kg |
| $K_{wr}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{wp}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{ur}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{vq}$ | $+0.00e+000$ | kg$\cdot$ m/rad |
| $K_{pq}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad$^2$ |
| $K_{qr}$ | $+0.00e+000$ | kg$\cdot$ m$^2$/rad$^2$ |

Table C.8: Added Mass M-, N-Moment Cross-term Coefficients

| Parameter | Value | Units |
|---|---|---|
| $M_{wq}$ | $+0.00e+000$ | kg· m/rad |
| $M_{uqa}$ | $+1.93e+000$ | kg· m/rad |
| $M_{uu}$ | $+0.00e+000$ | kg |
| $M_{wwa}$ | $+0.00e+000$ | kg |
| $M_{uwa}$ | $+3.46e+001$ | kg |
| $M_{vr}$ | $+0.00e+000$ | kg· m/rad |
| $M_{vp}$ | $-1.93e+000$ | kg· m/rad |
| $M_{pp}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $M_{rr}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $M_{rp}$ | $+4.86e+000$ | kg· $m^2$/$rad^2$ |
| $M_{uv}$ | $+0.00e+000$ | kg |
| $M_{vw}$ | $+0.00e+000$ | kg |
| $M_{up}$ | $+0.00e+000$ | kg· m/rad |
| $M_{wr}$ | $+0.00e+000$ | kg· m/rad |
| $M_{wp}$ | $+0.00e+000$ | kg· m/rad |
| $M_{ur}$ | $+0.00e+000$ | kg· m/rad |
| $M_{pq}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $M_{qr}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $N_{uu}$ | $+0.00e+000$ | kg |
| $N_{wu}$ | $+0.00e+000$ | kg |
| $N_{uq}$ | $+0.00e+000$ | kg· m/rad |
| $N_{wq}$ | $+0.00e+000$ | kg· m/rad |
| $N_{qq}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $N_{vva}$ | $+0.00e+000$ | kg |
| $N_{vr}$ | $+0.00e+000$ | kg· m/rad |
| $N_{vp}$ | $+0.00e+000$ | kg· m/rad |
| $N_{rp}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $N_{pp}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |
| $N_{uva}$ | $-3.46e+001$ | kg |
| $N_{vw}$ | $+0.00e+000$ | kg |
| $N_{up}$ | $+0.00e+000$ | kg· m/rad |
| $N_{ura}$ | $+1.93e+000$ | kg· m/rad |
| $N_{wp}$ | $-1.93e+000$ | kg· m/rad |
| $N_{vq}$ | $+0.00e+000$ | kg· m/rad |
| $N_{pq}$ | $-4.86e+000$ | kg· $m^2$/$rad^2$ |
| $N_{qr}$ | $+0.00e+000$ | kg· $m^2$/$rad^2$ |

Table C.9: Propeller Terms

| Parameter | Value | Units |
|---|---|---|
| $X_{prop}$ | $5.16e+000$ | N |
| $K_{prop}$ | 0 | N· m |

Table C.10: Control Fin Coefficients

| Parameter | Value | Units |
|---|---|---|
| $Y_{uu\delta_r}$ | $+9.64e+000$ | kg/(m· rad) |
| $Z_{uu\delta_s}$ | $-9.64e+000$ | kg/(m· rad) |
| $M_{uu\delta_s}$ | $-6.15e+000$ | kg/rad |
| $N_{uu\delta_r}$ | $-6.15e+000$ | kg/rad |
| $Y_{uvf}$ | $-9.64e+000$ | kg/m |
| $Z_{uwf}$ | $-9.64e+000$ | kg/m |
| $Y_{urf}$ | $+6.15e+000$ | kg/rad |
| $Z_{uqf}$ | $-6.15e+000$ | kg/rad |
| $M_{uwf}$ | $-6.15e+000$ | kg |
| $N_{uvf}$ | $+6.15e+000$ | kg |
| $M_{uqf}$ | $-3.93e+000$ | kg·m/rad |
| $N_{urf}$ | $-3.93e+000$ | kg·m/rad |

# References


[1] R. Bachmayer, N. Leonard, J. Graver, E. Fiorelli, P. Bhatta, and D. Paley, "Underwater gliders: recent developments and future applications," *International Symposium on Underwater Technology (UT'04)*, pp. 195–200, April 2004. 1

[2] *AquaJelly, An artificial jellyfish with electric drive unit*, Festo AG & Co. KG, 2008. [Online]. Available: http://www.festo.com/rep/en-us_us/assets/Corporate_img/Festo_AquaJelly_en.pdf 2

[3] *Subsea Glider*, Evo Logics, 2008. [Online]. Available: http://www.evologics.de/documents/BionikManta_web.pdf 2

[4] D. C. Webb and P. J. Simonetti, "The SLOCUM AUV: An environmentally propelled underwater glider," in *Proc. of the 11th International Symposium on Unmanned Untethered Submersible Technology*, August 23-25 1999, pp. 75–85. 2

[5] P. Simonetti, "Slocum glider: Design and 1991 field trials," Webb Research Corp., Falmouth, MA, Tech. Rep. N00014-90C-0098, September 1992, under subcontract from Woods Hold Oceanographic Institution. 2

[6] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006. 3, 9

[7] L. Dubins, "On curves of minimum length with a constraint on average curvature and with prescribed initial and terminal position and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, Jul. 1957. 4, 14

[8] T. Prestero, "Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle," Master's thesis, Massachusetts Institute of Technology, Cambridge, 1994. 5, 16, 17, 56, 57, 62

[9] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 3rd ed. Germany: Springer, 2006. 6

[10] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Intl. Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, September 2004. 7, 8

[11] B. L. Brummit and A. Stentz, "GRAMMPS: a generalized mission planner for multiple mobile robots in unstructured environments," in *Proc. IEEE International Conference on Robots and Automation (ICRA'98)*, May 16-20 1998, pp. 1564–1571. 8

[12] Z. Yu, L. Jinhai, G. Guochang, Z. Rubo, and Y. Haiyan, "An implementation of evolutionary computation for path planning of cooperative mobile robots," in *Proc. of the fourth world congress on intelligent control and automation*, 2002, pp. 1798–1802. 8

[13] J. L. Ryan, T. Bailey, J. Moore, and W. Carlton, "Reactive Tabu search in unmanned aerial reconnaissance simulations," in *Proc. of the 1998 winter simulation conference*, 1998, pp. 873–879. 8

[14] M. J. Matarić, "Minimizing complexity in controlling a mobile robot population," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'92)*, vol. 1, May 1992, pp. 830–835. 8

[15] *MissionLab, User Manual for MissionLab version 7.0*, College of Computing, Georgia Institute of Technology, 2006. [Online]. Available: http://www.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab/mlab_manual-7.0.pdf 8

[16] L. Parker, "Heterogeneous multi-robot cooperation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Feb. 1994. 8

[17] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, Dec 1980. 9

[18] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'99)*, vol. 2, 1999, pp. 1234–1239. 9

[19] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, October 2002. 9

[20] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'02)*, Washington, DC, 2002, pp. 3016–3023. 9

[21] M. Dias and A. Stentz, "A market approach to multirobot coordination," The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, Tech. Rep. CMU-RI-TR-01-26, August 2001. 9

[22] ——, "Opportunistic optimization for market-based multirobot control," in *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, vol. 3, 2002, pp. 2714–2720. 9

[23] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *Proc. IEEE International Conference on Robots and Automation (ICRA'04)*, vol. 4, April 26-May 1 2004, pp. 3622–3627. 9

[24] S. Sariel, T. Balch, and J. Stack, "Distributed multi-AUV coordination in naval mine countermeasure missions," Georgia Institute of Technology, Atlanta, Georgia, 30332, Tech. Rep. GIT-GVU-06-04, 2006. 9

[25] R. Turner, "Intelligent control of autonomous underwater vehicles: the Orca project," in *IEEE International Conference on Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century.*, vol. 2, Oct 1995, pp. 1717–1722. 9

[26] R. M. Turner and E. H. Turner, "Self-organization and reorganization of multi-AUV systems: CoDA project overview," in *Proc. IEEE Workshop on Multiple* AUV *Operations*, Sebasco Estates, Maine, Jun. 2004. 9

[27] S. Jeyaraman, A. Tsourdos, R. Zbikowski, B. White, L. Bruyere, C. A. Rabbath, and E. Gagnon, "Formalised hybird control scheme for a uav group using dubins set and model checking," in *Proc. IEEE Conference on Decision and Control (CDC'04)*, Atlantis, Paradise Island, Bahamas, Dec. 14–17, 2004, pp. 4299–4304. 9

[28] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 98–104, Jan 2007. 9

[29] B. Schulz, B. Hobson, M. Kemp, J. Meyer, R. Moody, H. Pinnix, and M. St Clair, "Field results of multi-UUV missions using ranger micro-UUVs," in *Proc. OCEANS 2003*, vol. 2, Sept. 2003, pp. 956–961. 10

[30] R. E. Davis, N. E. Leonard, and D. M. Fratantoni, "Routing strategies for underwater gliders," *Deep-Sea Research II*, 2008. 10

[31] J. A. Hartigan and M. A. Wong, "A $K$-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979. 12

[32] M. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation,"

in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS'04)*, vol. 1, Sept. 28-Oct. 2 2004, pp. 698–705. 13

[33] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–274, Mar. 2001. 14

[34] H. J. Sussmann and G. Tang, "Shortest paths for the reeds-shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control," Department of Mathematics, Rutgers University, New Jersey, Tech. Rep. SYNCON 91-10, 1991. 14

[35] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins vehicle," in *American Control Conference*, Portland, OR, Jun. 2005, pp. 786–791. 25, 29

[36] (2008) Ocean Server Technology Inc - Iver2: Mobile Sensor Platform. [Online]. Available: http://www.iver-auv.com/products_iver2.html 36

[37] W. Malik, S. Rathinam, S. Darbha, and D. Jeffcoat, "Combinatorial motion planning for multiple vehicle systems," in *Proc. IEEE Conference on Decision and Control (CDC'06)*, San Diego, CA, USA, Dec. 13–15, 2006, pp. 5299–5304.

[38] M. P. Georgeff, "Planning," in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate, Eds. San Mateo, CA: Kaufmann, 1990.

[39] J. Latombe, *Robot Motion Planning*. New York: Kluwer Academic Publishers, 1991.

[40] D. Parsons and J. Canny, "A motion planner for multiple mobile robots," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'00)*, vol. 1, May 1990, pp. 8–13.

[41] C. L. Pape, "A combination of centralized and distributed methods for multi-agent planning and scheduling," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'90)*, vol. 1, May 1990, pp. 488–493.

[42] N. Rugg-Gunn and S. Cameron, "A formal semantics for multiple vehicle task and motion planning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'94)*, vol. 3, May 1994, pp. 2464–2469.

[43] M. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[44] B. Schulz, B. Hobson, M. Kemp, J. Meyer, R. Moody, H. Pinnix, and M. St Clair, "Field results of multi-uuv missions using ranger micro-uuvs," in *Proc. OCEANS 2003.*, vol. 2, September 2003, pp. 956–961.

[45] E. A. Leveille, "Analysis, redesign and verification for the Iver2 autonomous underwater vehicle motion controller," Master's thesis, University of Massachusetts Dartmouth, Dartmouth, MA, 2007.

[46] B. Golden, L. Levy, and R. Dahl, "Two generalizations of the traveling salesman problem," *Omega*, vol. 9, no. 4, pp. 439–441, 1981.

[47] B. L. Brummit and A. Stentz, "Dynamic mission planning for multiple mobile robots," in *Proc. IEEE International Conference on Robots and Automation (ICRA'96)*, April 22-28 1996, pp. 2396–2401.