ASSIGNING CLOSELY SPACED TARGETS TO MULTIPLE AUTONOMOUS UNDERWATER VEHICLES

Beverley Chow[1] bchow@uwaterloo.ca
Christopher Michael Clark[2] cmclark@calpoly.edu
Jan Paul Huissoon[1] jph@uwaterloo.ca
1) University of Waterloo - Canada
2) California Polytechnic State University - USA

Abstract

This paper addresses the problem of allocating closely spaced targets to multiple autonomous underwater vehicles 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 proposed algorithm solves the task allocation problem with market-based auctions that minimize the total travel time to complete the mission. Simulations show that the proposed algorithm is able to create feasible paths with a lower cost when compared to solutions whose cost functions are calculated based solely on Euclidean distances. Field tests conducted on an Iver2 AUV validate the performance of the proposed algorithm in real world environments. Results show that the proposed algorithm generates paths that are feasible for an AUV to track closely, even in the presence of ocean currents.

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.

This paper presents an approximate algorithm for the task allocation 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 and the problem has been simplified to limit the robots to operate on the Euclidean plane. 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 to be minimized includes the total time to visit all of the task points.

The features that differentiate this research to similar problems previously studied are the kinematic constraints on the vehicle and the presence of a constant ocean current. This paper addresses the inability of an AUV to turn at any arbitrary yaw rate which becomes a problem when target points are close together. The Dubins model [1] 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.

In this paper, 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 model from [2] is used so that the computational complexity is reduced.

The remainder of this paper is organized as follows: Section 2 gives an overview of the task allocation problem and describes various other techniques that have been used to solve related problems. Section 3 begins with the formal problem definition. In Section 4, an overview of

the proposed algorithm is introduced with the details presented in Section 5. Section 6 discusses the results from simulations 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 as described in Section 7. The paper concludes with a summary of results in Section 8.

2 BACKGROUND

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 been studied extensively in combinatorial optimization. Even in the restricted case of one salesperson, the MTSP is strongly *NP*-hard [3].

Several approaches have been applied to the general problem of allocating tasks between multiple robots in a team; refer to [4] for a survey of these. Heuristic methods are typically used since optimizing the performance is often computationally intractable. Parker's ALLIANCE [5] is one of the earliest demonstrations of behaviour-based architectures for task allocation. Another frequently used method is based on market mechanisms, such as auctions, which have been demonstrated in [6] to be fast and robust on real robots. Specific work for AUVs, often called mission planning, include the work by Sariel et al. [7] and vehicles with bounded curvature are considered by Jeyaraman et al. [8]. Similar to the mission planning problem is the routing problem as investigated by Davis et al. [9] for the planning for underwater gliders in the presence of significant currents. However, the vehicle dynamics are not accounted for in their routing strategy which this paper aims to address.

3 PROBLEM STATEMENT

This paper 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

$$C_{\text{total}} = \max_i C(S_i) \tag{1}$$

subject to

$$D = \bigsqcup_i S_i \qquad \text{(disjoint union)} \tag{2}$$

$$\left.\begin{aligned} \frac{dx_{i,t}}{dt} &= u_0 \cos(\psi_{i,t}) \\ \frac{dy_{i,t}}{dt} &= u_0 \sin(\psi_{i,t}) \\ \frac{d\psi_{i,t}}{dt} &= r, \qquad r \in [-\omega, +\omega] \end{aligned}\right\} \tag{3}$$

where u_0 denotes the nominal vehicle speed, $\psi_{i,t}$ the yaw of the vehicle, and ω represents the bound on the yaw rate. In (1), $C(S_i)$ is the time required for V_i to complete its tour S_i. Note that (2) dictates all tasks to be visited and restricts each task to be assigned to only one vehicle and (3) considers the non-holonomic constraints of the vehicle.

4 OVERVIEW OF PLANNER

This paper 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 nonlinear, 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 paper is *k-means* [10], 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. For all $i = 1, 2, ..., n$

clusters, the three task points in each cluster i that are farthest from the centroid are assigned to the 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 does 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. [11]. The unassigned tasks are first ordered according to their distance from the 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:

$$B_i(d_j, S_i) = \min_{0 \le k \le l} C(s_1, ..., s_k, d_j, s_{k+1}, ..., s_l) \quad (4)$$

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 costs, as described in detail in Section 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.

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 section. 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 [1] 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 [12]. Dubins' result shows that, given any two points, the shortest path that considers the constraints expressed in (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. 1 for two waypoints with initial and terminal orientations, denoted α_k and α_{k+1} respectively. Note that α is measured counter-clockwise with respect to the positive x-axis.

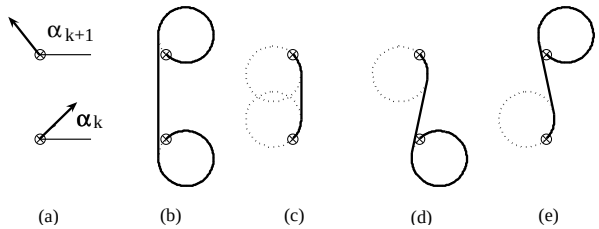

Figure 1: (a) Two waypoints (x_k, y_k) and (x_{k+1}, y_{k+1}) with $\alpha_k = \frac{\pi}{4}$ and $\alpha_{k+1} = \frac{3\pi}{4}$. (b)-(e) Four ways of connecting two waypoints using Dubins curves.

Finding the shortest path between two points requires repetitively solving the shortest time algorithm for various entry and exit AUV orientations (i.e. α_k, α_{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, α_k and α_{k+1} are constrained to $\Lambda = \{\lambda\pi/4 \mid \lambda = 0, ..., 7\}$. With 8 possibilities for α_k and α_{k+1} and 4 ways of connecting them, a total of $8 \times 8 \times 4 = 256$ paths is possible for every pair of waypoints, with one of them being the optimal path. Fig. 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 α_k and α_{k+1} yield different costs.

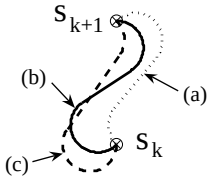

Figure 2: Multiple trajectories for different initial and final orientations. (a) $\alpha_k = \frac{3\pi}{4}$, $\alpha_{k+1} = \frac{-3\pi}{4}$ (b) $\alpha_k = \frac{-3\pi}{4}$, $\alpha_{k+1} = \pi$ (c) $\alpha_k = \frac{-\pi}{2}$, $\alpha_{k+1} = \frac{3\pi}{4}$.

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

$$C(S) = \min_{(\alpha_1,\ldots,\alpha_l)\in\Lambda^l} \sum_{k=1}^{l} \Delta t_{(s_k,\alpha_k)\rightarrow(s_{k+1},\alpha_{k+1})} \tag{5}$$

5.2 AUV Dynamic Model

Calculating the time required to follow a Dubins path requires the knowledge of the vehicle dynamics. This paper uses the REMUS AUV model created by Prestero [2] 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. 3.

Figure 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 [2] 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:

$$\mathbb{X}_{i,t} = [u_i \;\; v_i \;\; w_i \;\; p_i \;\; q_i \;\; r_i \;\; x_i \;\; y_i \;\; z_i \;\; \phi_i \;\; \theta_i \;\; \psi_i]^T . \tag{6}$$

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:

$$\mathbb{X}_{i,t+1} = f(\mathbb{X}_{i,t},\, \mathbb{U}_{i,t},\, u_c,\, \psi_c) \tag{7}$$

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 ψ_c are the magnitude and direction of the ocean current respectively. For the input vector, δ_s is the stern fin angle, δ_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 (7) uses the Euler method of numerical integration to yield the new vehicle state at each time step as:

$$\mathbb{X}'_{i,t+1} = \mathbb{X}_{i,t} + \left(\dot{\mathbb{X}}_{i,t} \cdot \Delta t\right) \tag{8}$$

where the state vector derivative $\dot{\mathbb{X}}_{i,t}$ is updated using the model $\dot{\mathbb{X}}_{i,t} = f(\mathbb{X}_{i,t},\, \mathbb{U}_{i,t})$ from [2]. With the presence of a fixed current, the position of the vehicle relative to the inertial-fixed frame is updated as follows:

$$\begin{aligned} x_i &= x'_i + (u_c \cos(\psi_c) \cdot \Delta t) \\ y_i &= y'_i + (u_c \sin(\psi_c) \cdot \Delta t) \end{aligned} \tag{9}$$

where x'_i and y'_i denote the position of the i^{th} vehicle after the integration step given in (8). By combining (8) and (9), the function f in (7) 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 α_k and α_{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. 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 ψ and the direction of the current ψ_c.

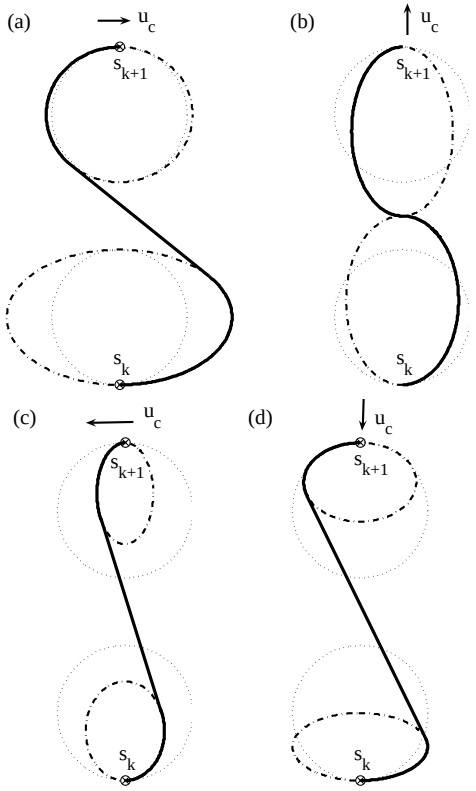

Figure 4: Dubins Curves between two waypoints with ocean currents u_c = 0.25 m/s. (a) ψ_c=0, (b) $\psi_c = \frac{\pi}{2}$, (c) $\psi_c = \pi$, and (d) $\psi_c = \frac{-\pi}{2}$.

To determine the shape of the ellipse, equation (7) 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 $\mathbb{U}_0 = [0\ 1.75\ 5.16\ 0]^T$, where 1.15 m/s is the nominal speed of the AUV, 1.75 rad is the rudder fin angle, and 5.16 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 1.75 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).

Using this data, a lower order model $\tilde{f}$ was created to

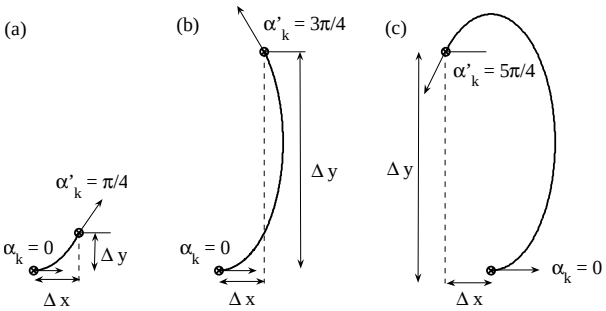

Figure 5: Vehicle position for various values of $\Delta\psi$ as the vehicle moves along the maximum curvature ellipse. (a) $\Delta\psi = \frac{\pi}{4}$. (b) $\Delta\psi = \frac{3\pi}{4}$. (c) $\Delta\psi = \frac{5\pi}{4}$.

determine the position of the vehicle given the change in the vehicle's heading, and the magnitude and velocity of the current.

$$(\Delta x, \Delta y) = \tilde{f}(\Delta\psi, u_c, \psi_c) \tag{10}$$

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. 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. 6b). The position of P_a and P_b are determined by using $\tilde{f}$ from (10). The slope of the line segment from P_a to P_b is calculated and becomes the new slope for the next iteration (Fig. 6c). The process continues until convergence (Fig. 6d).

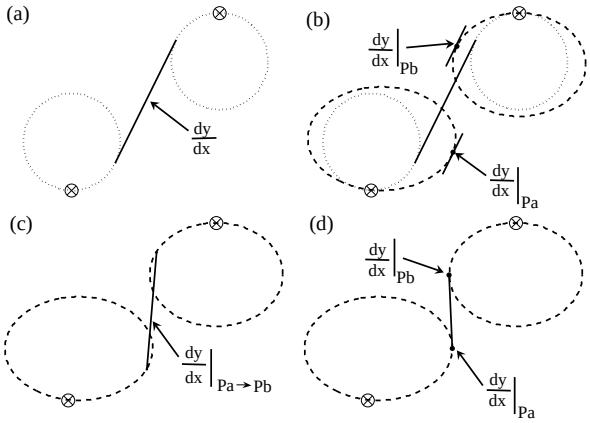

Figure 6: Illustration of the iterative process used to find a tangent to two curves.

Note that Fig. 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 ψ must be calculated to compensate for the effect of current currents.

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), a lower order model $\hat{f}$ is created based on the full model f from (7) as:

$$\Delta t_{(s_k,\alpha_k)\to(s_{k+1},\alpha_{k+1})} = \hat{f}\left[s_k, s_{k+1}, \alpha_k, \alpha_{k+1}, u_0, u_c, \psi_c\right] \tag{11}$$

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 $\kappa = 1, ..., 16$).

5.4.2 Straight line segments

For straight line segments, consider a vehicle moving at speed u and heading ψ through the water with current velocity u_c and direction ψ_c. The vehicle's velocity along the desired path has magnitude $u_{desired}$ and direction $\psi_{desired}$. These velocities are illustrated in Fig. 7. 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 ψ and speed $u_{desired}$ along the desired vehicle motion direction are:

$$\begin{aligned} \psi &= -\arcsin(u_{c_N}/u) + \psi_{desired} \\ u_{desired} &= u_{c_{\psi_{desired}}} + u\sqrt{1-(u_{c_N}/u)^2}. \end{aligned} \tag{12}$$

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 presence of currents.

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

$$\Delta t_{straight} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}/u_{desired} \tag{13}$$

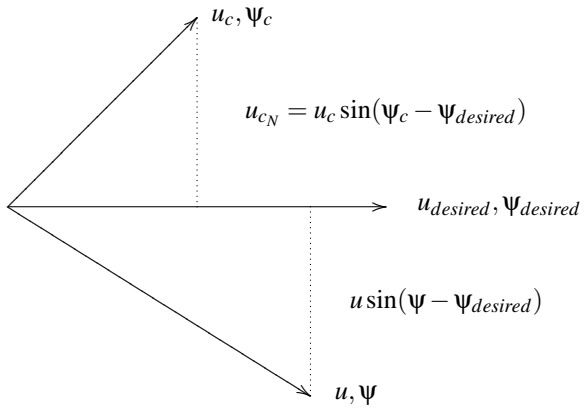

Figure 7: Illustration of the relative velocities.

Combining these results in

$$\begin{aligned} \Delta t_{(s_k,\alpha_k)\to(s_{k+1},\alpha_{k+1})} =& \Delta t_{arc(s_k\to P_a)} + \Delta t_{straight(P_a\to P_b)} + \\ & \Delta t_{arc(P_b\to s_{k+1})}. \end{aligned} \tag{14}$$

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 α_k and α_{k+1} also have to be recalculated with all values of Λ. 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 α_k and α_{k+1} are kept from the previous round of bidding, (11) 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.

6 SIMULATION RESULTS

The algorithm described in Section 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. [13] was used.

To illustrate the behaviour of the proposed algorithm, consider one particular trial with $n = 3$ and $m = 20$. Using the *k-means* clustering method described in Section 4, the 20 task points were partitioned into 3 clusters as shown in Fig. 8. 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 different clusters using the same dataset depending on the starting points chosen and Fig. 8 is one of many solutions.

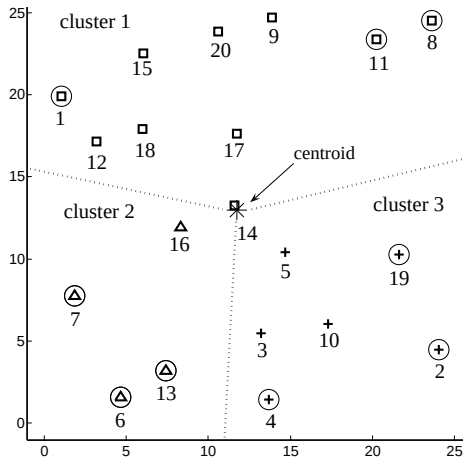

Figure 8: Results from clustering using *k*-means. Tasks with a circle around it are assigned to the vehicle responsible for that cluster.

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: $S_1 = \{d_8, d_{11}, d_1\}$; $S_2 = \{d_6, d_7, d_{13}\}$; $S_3 = \{d_2, d_4, d_{19}\}$.

As a baseline for comparison, the "alternating algorithm" described by Savla et al. [13] is used for the creation of Dubins TSP tours (i.e. sequences from applying Dubins' model). It works as follows: given a set of n points, the optimal Euclidean MTSP tours (i.e. that do not consider path curvature) are computed using auctions (Fig. 9). Then, it is necessary to obtain a feasible path through these ordered points using the method in [13] which includes the curvature constraints of the vehicle.

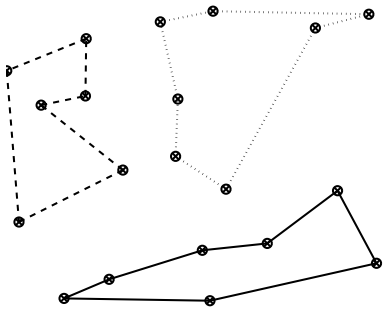

Figure 9: Euclidean MTSP solution for allocating 20 tasks to 3 robots

6.1 Discussion

The results from running the simulation on 50 different datasets are summarized in Table 1 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 (7). On average, the proposed algorithm reduced $T_{\max}$ by 43% over the "alternating algorithm" in the absence of currents and 45% with the presence of currents.

Table 1: 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
$n = 1$	36.1	36.1	42.8	42.8
$n = 2$	37.8	34.3	43.4	38.9
$n = 3$	47.2	37.6	48.6	46.7
$n = 4$	52.2	41.1	45.8	37.0
$n = 5$	41.7	31.2	44.1	40.5

Consider one particular trial illustrated in Fig. 10 and Fig. 11 whose results are presented in Table 2. For the case with no ocean currents, the "alternating algorithm" creates paths with numerous loops when two successive points are close together and the vehicle orientation does not allow for the second point to be reached without long maneuvers (Fig. 10(a)). This is avoided in the proposed algorithm by generating sequences that are feasible but limit the number of additional loops (Fig. 10(b)). Similar results are obtained with the presence of ocean currents as shown in Fig. 11(a) and Fig. 11(b).

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 considers the possibility that two successive points that

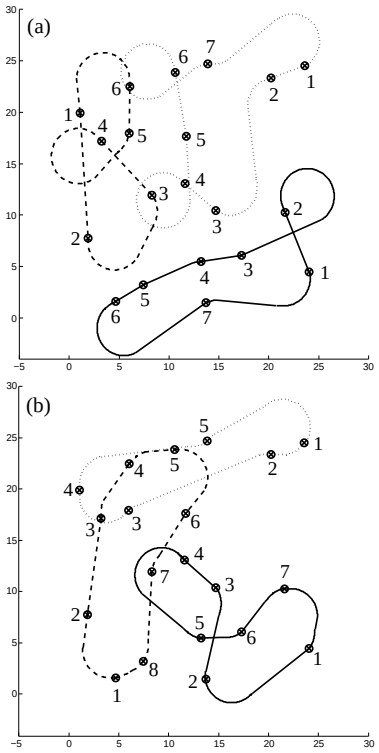

Figure 10: Sequences generated by the "alternating algorithm" (a) and the proposed algorithm (b) using the dataset in Fig. 8 with $n = 3$, $m = 20$, and $u_c = 0$.

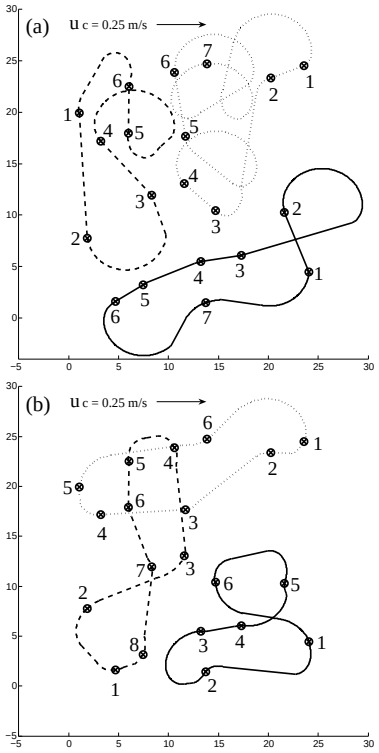

Figure 11: Sequences generated by the "alternating algorithm" (a) and the proposed algorithm (b) using the dataset in Fig. 8 with $n = 3$, $m = 20$, $u_c = 0.25m/s$, and $\psi_c = 0$.

Table 2: Summary of path costs for the dataset in Fig. 8 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	59.8
T_{avg} (s)	75.1	54.5	88.1	57.1

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 attempts to avoid paths that force the vehicles to drive against the ocean current. Instead, paths that allow the ocean current to aid the vehicle in the direction of travel are favoured.

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

6.2 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. 12. 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 $n = 1$.

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.

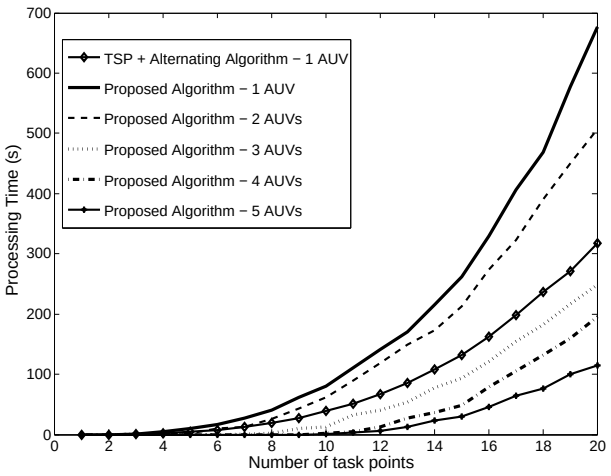

Figure 12: Processing Time.

7 EXPERIMENTAL RESULTS

Experiments were conducted at the Avila Pier in California using the Iver2 AUV as shown in Fig. 13.

Figure 13: 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. 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.

Missions 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:

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

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. Note that for Experiment 1, $T_{\max} = T_{\text{avg}}$ since there is only one vehicle, and therefore, it is denoted as $T_{\exp}$.

7.1 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 Underwater Vehicle Console (UVC) developed by the Ocean Server Technology Inc. The UVC provides an interface to the Iver2 AUV's sensors, motors, and control processes through a Remote Desktop Connection. However, 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 an 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. When using the UVC to control the Iver2, the AUV was not able to track waypoints precisely, which can be seen in the experimental results.

7.2 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.

The first method solves the traveling salesman problem without considering the curvature constraints of the vehicle or the effects of ocean current (Fig 14). 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 15). 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 16). Results from field tests are summarized in Table 3.

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 algo-

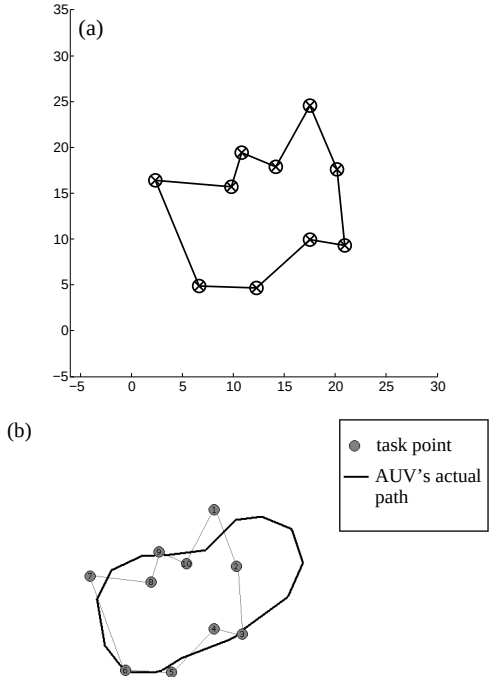

Figure 14: (a) Paths generated for the TSP from Matlab. (b) Field test results - $T_{\mathbf{exp}}$ = 85 s and $D_{\mathbf{avg}}$ = 1.71 m.

Table 3: Field test results from three randomly generated datasets of 10 task points for one vehicle.

	T_{exp} (s)	D_{avg} (m)
TSP	104	1.44
Alternating Algorithm	315	0.90
Proposed Algorithm	207	0.62

rithm". 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 only getting within 1.4 meters of the desired task point on average. At worst, the AUV only travelled to within 4.05 meters of one task point. 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 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.

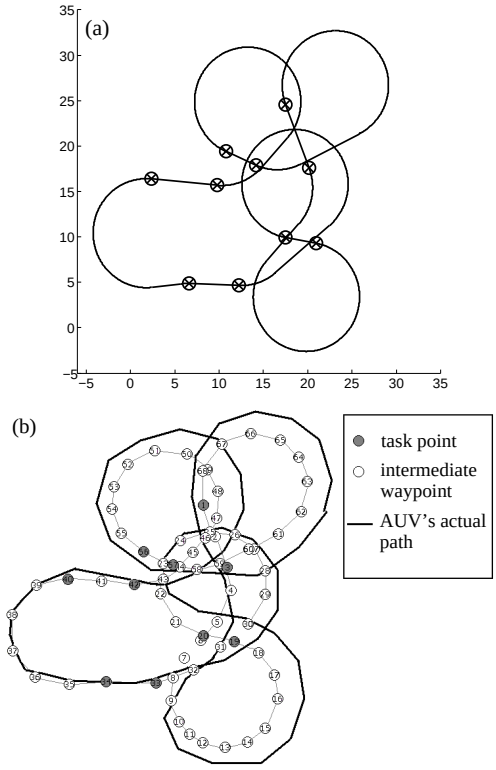

Figure 15: (a) Paths generated using the 'alternating algorithm' from Matlab. (b) Field test results - $T_{\mathbf{exp}}$ = 290 s and $D_{\mathbf{avg}}$ = 0.88 m.

7.3 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 a time.

The first method solves the multiple traveling salesman problem without considering the curvature constraints of the vehicle (Fig 17). 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 18). The third method uses the proposed algorithm which considers the curvature constraints of the vehicle in generating the sequence for the vehicle (Fig 19). Results from field tests are summarized in Table 4.

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

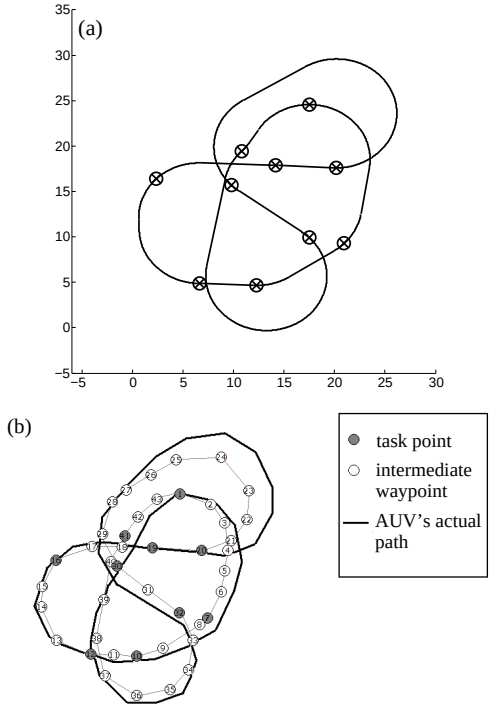

Figure 16: (a) Paths generated using the proposed algorithm from Matlab. (b) Field test results - $T_{\mathbf{exp}}$ = 209 s and $D_{\mathbf{avg}}$ = 0.42 m.

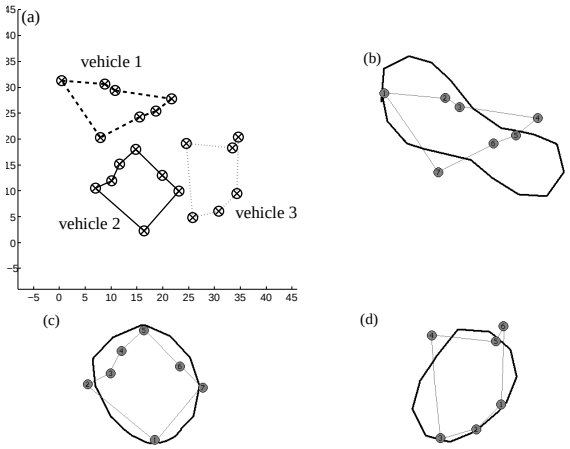

Figure 17: Paths generated for the TSP from Matlab. (b) Field test results for vehicle 1. (c) Field test results for vehicle 2. (d) Field test results for vehicle 3.

erage distance to task 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

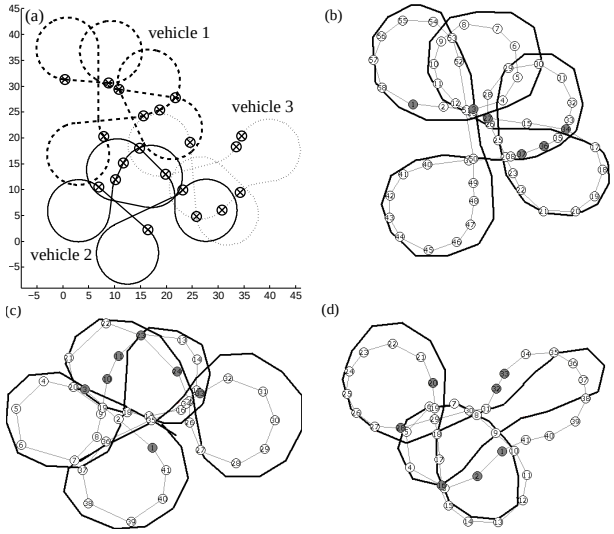

Figure 18: (a) Paths generated using the "alternating algorithm" from Matlab. (b) Field test results for vehicle 1. (c) Field test results for vehicle 2. (d) Field test results for vehicle 3.

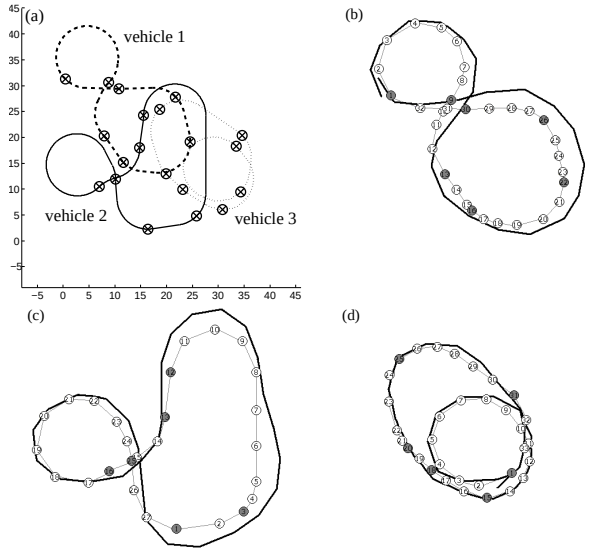

Figure 19: (a) Paths generated using the proposed algorithm from Matlab. (b) Field test results for vehicle 1. (c) Field test results for vehicle 2. (d) Field test results for vehicle 3.

Table 4: Field test results from three randomly generated datasets of 20 task points for 3 vehicles.

	$T_{\max}$ (s)	T_{avg} (s)	D_{avg} (m)
TSP	89.3	78.0	1.65
Alternating Algorithm	279	252	1.35
Proposed Algorithm	145	134	0.84

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 as 0.152 m/s at 224° (SW) and these values were used in the proposed algorithm to create paths for the Iver2 AUV. Unfortunately, the ocean currents had changed to 0.125 m/s at 216° (SW) 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. Ideally, 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.

8 CONCLUSION

This paper 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. 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.

References

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

[2] 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.

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

[4] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robotics Research*, vol. 23, no. 9, pp. 939-954, 2004.

[5] L. E. Parker, "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation," *IEEE Trans. Robotics and Automation*, vol. 14, no. 2, pp. 220-240, 1998.

[6] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. IEEE Conf. Robotics and Automation*, vol.3, Washington, DC, pp. 3016-3023, 2002.

[7] 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.

[8] S. Jeyaraman et al., "Formalised Hybrid Control Scheme for a UAV Group using Dubins Set and Model Checking," in *Proc. IEEE Conf. Decision and Control*, vol.4, Paradise Island, Bahamas, pp.4299-4304, 2004.

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

[10] J. A. Hartigan and M. A. Wong, "A K-Means Clustering Algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100-108, 1979.

[11] M. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, vol. 1, pp. 698-705, 2004.

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

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