ABSTRACT

Title of dissertation: PLANNING FOR AUTONOMOUS OPERATION OF UNMANNED SURFACE VEHICLES

Brual Shah, Doctor of Philosophy, 2016

Dissertation directed by: Professor Satyandra K. Gupta
Department of Mechanical Engineering

The growing variety and complexity of marine research and application oriented tasks requires unmanned surface vehicles (USVs) to operate fully autonomously over long time horizons even in environments with significant civilian traffic. The autonomous operations of the USV over long time horizons requires a path planner to compute paths over long distances in complex marine environments consisting of hundreds of islands of complex shapes. The available free space in marine environment changes over time as a result of tides, environmental restrictions, and weather. Secondly, the maximum velocity and energy consumption of the USV is significantly influenced by the fluid medium flows such as strong currents. Finally, the USV have to operate in an unfamiliar, unstructured marine environment with obstacles of variable dimensions, shapes, and motion dynamics such as other unmanned surface vehicles, civilian boats, shorelines, or docks poses numerous planning challenges. The proposed Ph.D. dissertation explores the above mentioned problems by developing computationally efficient path and trajectory planning algorithms that

enables the long term autonomous operation of the USVs. We have developed a lattice-based 5D trajectory planner for the USVs operating in the environment with the congested civilian traffic. The planner estimates collision risk and reasons about the availability of contingency maneuvers to counteract unpredictable behaviors of civilian vessels. Secondly, we present a computationally efficient and optimal algorithm for long distance path planning in complex marine environments using A* search on visibility graphs defined over quad trees. Finally, we present an A* based path planning algorithm with newly developed admissible heuristics for computing energy efficient paths in environment with significant fluid flows. The effectiveness of the planning algorithms is demonstrated in the simulation environments by using systems identified dynamics model of the wave amplitude modular vessel (WAM-V) USV14.

Planning for Autonomous Operation of Unmanned Surface Vehicles

by

Brual Shah

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Professor Satyandra K. Gupta, Chair/Advisor
Professor Hugh Bruck
Associate Professor Nikhil Chopra
Assistant Professor Mark D. Fuge
Professor Dana S. Nau (Dean's representative)

Dedication

To my parents, my brother, and all my teachers.

Acknowledgments

The completion of this thesis is a result of hard work, dedication and constant support of many people. I would like to take this opportunity to thank them all for their roles in my journey.

First and foremost, I would like to express my sincerest gratitude to Prof. Satyandra K. Gupta, my mentor and advisor. His ideas inspired me and his passion motivated me to pursue them. I would always be grateful to him for providing me with the right prospects to fructify them and further steer me towards the goal with his knowledge and wisdom. His attitude towards work and life, his thoughtfulness and his finesse in handling challenges has influenced me and encouraged me to be a better person.

I would like to thank Prof. Hugh Bruck, Prof. Nikhil Chopra, Prof. Mark D. Fuge, and Prof. Dana S. Nau for accepting my request and thus being part of my dissertation committee. I am thankful to them for their valuable time given the busy academic schedules. I am also thankful for their expertise, comments and suggestions which helped me identify unforeseen obstacles and areas of improvement. I am grateful to Office of Naval Research and National Science Foundation for supporting and funding by research work.

I am obliged to University of Maryland, Department of Mechanical Engineering and Institute of Systems Research for use of equipment and facility. I would like to thank the administrative staff for their help with administrative work.

I am also obliged to Florida Atlantic University for use of equipment, and

facility at Sea Tech. I would like to thank Prof. Karl von Ellenrieder, Ivan R. Bertaska, Armando J. Sinisterra, Wilhelm Klinger and Prof. Manhar Dhanak for their expertise and skill set which made it possible for me in conduct my experiments with more ease.

I would like acknowledge my success to Dr. Petr Švec. I would like to thank him for giving me a chance to work for him and eventually introducing me to the field of motion planning. He provided me with stimulating work that triggered me to learn more and expand my skill set. He cheered me to learn by questioning and thus inculcated a new approach towards learning in me. He has grown from a supervisor to a friend who continues to support me with his presence in my life.

I would like to thank my colleagues from Simulation-based System Design lab and friends Sagar, Atul, Carlos, Josh, Michael, Iain, Pradeep, Di, Kajal, Apurva, and Aditya for their support, encouragement and thought-provoking discussion on varied topics. I am indebted for their company in the lab, their help with my work and their friendship. I will always cherish my time spent with them.

Lastly but not the least, I would like to thank my family and the almighty. The values taught by my parents make me the person I am today. Their backing and emotional support has compelled me to continue my journey when faced with impasse. Their constant reassurance and care has driven me to achieve success and make them proud.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In recent years, Unmanned Surface Vehicles (USVs) have been increasingly used in many marine applications including ocean sampling, maritime search and rescue, hydrologic surveys, harbor surveillance, and defense [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. USVs are also used to assist Autonomous Underwater Vehicles (AUVs) for studying various types of marine species, coral reefs, and searching for natural resources [26].

There are a number of civilian applications where deploying a single USV or a team of small USVs can significantly reduce costs, improve safety, and increase operational efficiencies. Some representative applications include:

- **Remote/Persistent Ocean Sensing:** The use of a team of USVs for remote sensing can improve the measurement processes by permitting scientists to adaptively change sensing strategies as data is collected or to reposition sensors, as needed. There are certain types of measurements that can only be made at the air water interface, such as enthalpy/entropy flux, that are important for obtaining a complete picture of the energy transfer in the ocean. The use of small USV teams will also permit scientists to perform coordinated measurements utilizing multiple platforms and sensors.

- **Marine Search and Rescue:** USVs can be deployed to perform finer-scale searches with data obtained from aerial vehicles. Search and rescue operations typically use drift model simulations, supplemented with periodically updated meteorological conditions to direct search efforts by estimating the possible locations of people or objects at sea. Wide-area satellite-based meteorological measurements could be supplemented by point measurements acquired from the USVs, refining the simulation estimates and improving the search process. USV teams could also be used to assist unmanned underwater vehicles as surface-underwater communications gateways and to improve underwater localization.

- **Maritime Operations in Congested Port Environments:** When large ships arrive at a port, the control of the vessel is transferred from the ship captain to an experienced pilot, an officer specifically trained in the conditions specific to the port (such as currents, bathymetry, underwater obstacles, and local traffic). In coordination with tugboats and dock workers, the pilot guides the ship into port and docks it. Ship-piloting operations appear to be an area that can substantially benefit from human-guided USV teams by providing improved situational awareness to the pilot.

  According to the survey done by 16 UK volunteer ports, covering 594,126 docking operations in 2012, reported 292 casualties (injury/loss of life; damage to ship or property; environmental pollution resulting from a collision or incident) and 348 incidents (near misses or accidents, not included in the num-

ber of casualties) [27]. Most casualties and incidents occurred while underway in harbor waters and 84% of the incidents happened in either good or fair weather conditions. The top four incident types (74%) were: (1) contact with a fixed object, (2) other on-board incidents, (3) machinery or hull failure, and (4) collision with another vessel. The top four incident factors (70%) were: (1) equipment failure, (2) breach of regulation bylaw or direction, (3) inappropriate vessel navigation, and (4) incorrect procedures. Given that most incidents occur in fair weather, where communication, sensing, and perception are easier, and that most of the factors leading to marine incidents in harbors involve navigation or a breach of procedure/regulation, it is expected that a team of USVs can make substantial improvements in port safety.

- **Industrial Offshore Supply and Support**: The offshore extraction of both fossil fuels and renewable energies requires the maintenance and sustainment of large, distributed offshore systems, often in harsh environments, such as the North Sea, where operations can be costly, monotonous, and dangerous. Human-guided USV teams could reduce the risk to life and extend the spatial coverage of prognostic, health- and condition-monitoring systems.

Each USV operating in the complex marine environment encounter certain unique challenges that are not faced by robots operating indoors:

- The USVs will need to comply with the International Regulations for the Prevention of Collisions at Sea (COLREGs) [28] to avoid accidents and to ensure nearby manned vessels are not threatened by unexpected movements

(a) (b)

Figure 1.1: *(a)Topography of the complex marine environment and (b)A typical bay with multiple moving boats.*

of the USVs.

- Local current, wave, and wind conditions can impact a USV's performance by reducing the dynamic range of its sensors and actuators; the USVs must be able to compensate for this.

- Outdoor on-water operations involve some degree of risk and urgency. In most situations, completely eliminating risk is not feasible. Therefore, USVs must assess risks and make risk-informed decisions.

- Operations are carried out in highly dynamic environments and roles of individual USVs may need to change rapidly in response to changing situations.

1.2 Goal and Scope

This work presents the trajectory and path planning algorithms for USVs operating in dynamic and complex marine environments.

The main research issues this work addresses are as follows:

i. *Resolution-Adaptive Risk-Aware Trajectory Planning for Surface Vehicles Operating in Congested Civilian Traffic:*

The USVs will be operating in an unfamiliar, unstructured marine environment (see Figure 1.1 (b)) with obstacles of variable dimensions, shapes, and motion dynamics, such as other unmanned surface vehicles, civilian boats, shorelines, or docks posing numerous planning challenges. Efficient and safe navigation in a highly cluttered, dynamic environment requires prediction of the future movement of dynamic obstacles that can interact with each other in complex ways. The planner needs to perform reasoning about the risk associated with each expected avoidance maneuver. Additionally, it also needs to reason about the availability of contingency maneuvers to counteract the unpredicted behaviors of the vessels. These features cannot be incorporated into existing purely reactive planners [1, 29, 30]. By contrast, traditional, lattice-based, deliberative planners [31, 32, 33] can find a global optimal trajectory by employing multi-step look-ahead search. However, they are computationally slow when dealing fast moving obstacles. We have developed an algorithm that integrates a lattice-based, risk and contingency-aware planner (RCAP) which searches for trajectories in a 5D state space and reasons about the collision

risk and availability of contingency maneuvers (see Section 3.3). An adaptive variant (A-RCAP) dynamically scales the control action primitives based on the estimated spatio-temporal complexity of the local work-space around each state being expanded during the search for a trajectory.

ii. *Speeding up A* Search on Visibility Graphs Defined Over Quadtrees to Enable Long Distance Path Planning for Unmanned Surface Vehicles:*

Unmanned surface vehicles are increasingly used on missions with a large operating environment and long operating times. Often the marine environments are comprised of complex convoluted polygons that cannot be represented by the standard simple geometrical shapes. We have developed an algorithm that computes optimal paths using A* search on visibility graphs defined over quadtrees. We have also developed an admissible heuristic that accounts for large islands while estimating cost-to-go and provides a better lower bound than Euclidean distance-based heuristic.

iii. *Path Planning for Unmanned Vehicles Operating in Time-Varying Flow Fields:*

The USVs operating in the marine environment encounter significant fluid medium ows such as strong currents. These fluid flows influences the maximum operating velocity and energy consumption of the vehicle. Weather forecast reports provide an estimate of the medium flow and can be utilized to generate low-cost paths that exploit medium flow to aid the motion of the vehicle and conserve energy. We have developed an A* based path planning approach that can handle complex dynamic obstacles and arbitrary cost functions. The

efficiency of the A* search is improved by the new admissible heuristics for estimating cost-to-go by taking into account flow considerations.

1.3 Overview

An unmanned surface vehicle on a long voyage (see Figure 1.1(a)) and 5.1) will need to employ the three following planner types:

i. A global path planner to compute a sequence of waypoints from the start to the goal location which form a collision-free path [34, 35, 36, 37]. Often, only static obstacles are handled by this planner.

ii. A trajectory planner to compute risk-aware, dynamically feasible trajectories [38, 32] via the waypoints generated by the global path planner.

iii. A reactive planner for locally avoiding highly dynamic obstacles [39, 30, 40].

The planners described in Chapter 4 and 5 fall into the global path planning category and the planner in Chapter 3 fall into risk-aware trajectory planning category. Figure 1.2 shows a typical planning architecture used for unmanned surface vehicles. The path planner (shown in green) computes the geometrically feasible collision free path given the geographical map (typically of the order of several hundred kilometers) and the weather model to compute energy efficient path planner. The paths computed by the path planner are provided as an input to the trajectory planner which then computes risk-aware, COLREGs-compliant, and dynamically feasible trajectory for shorter portions on map (typically of the order of several

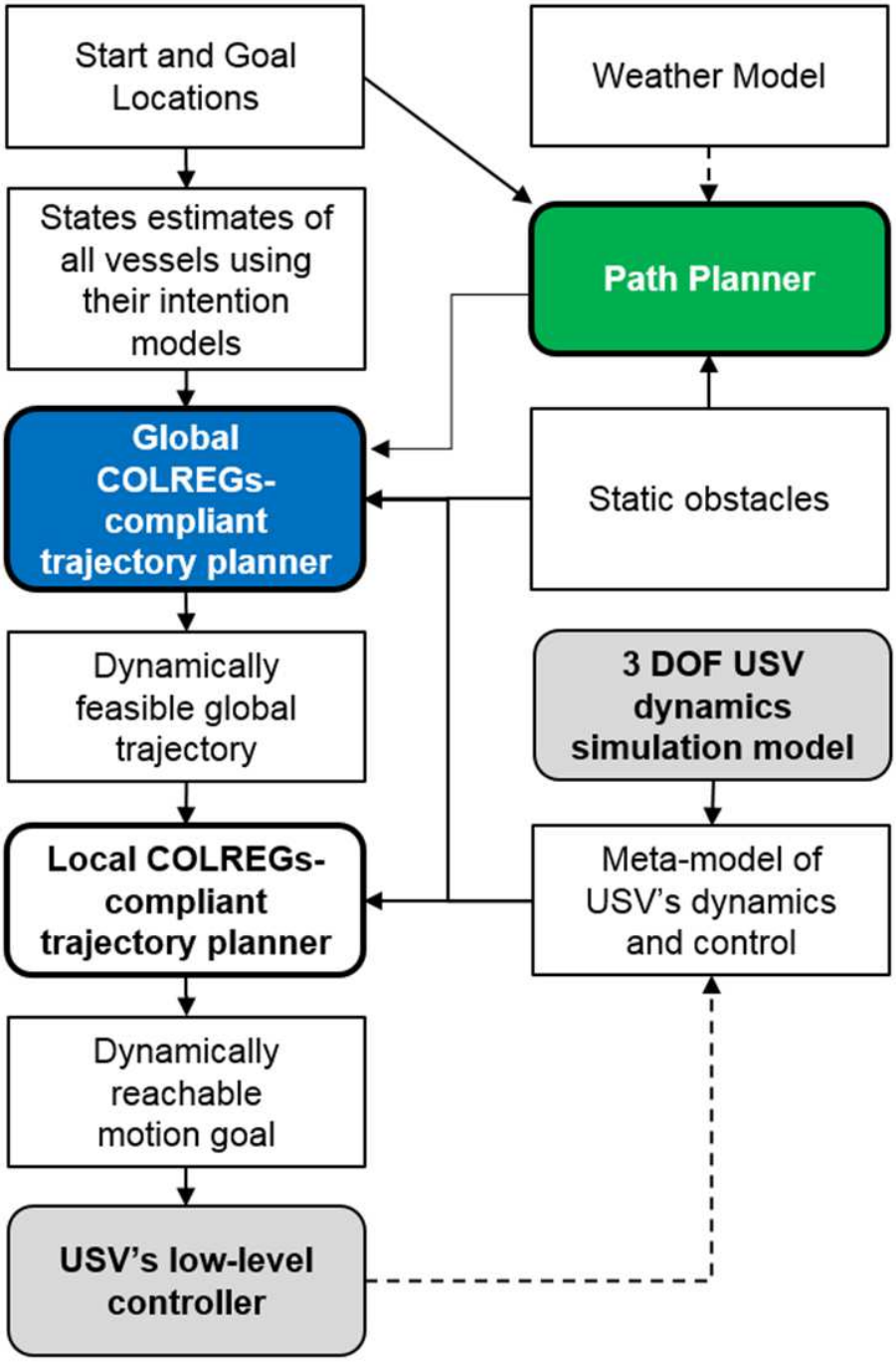


Figure 1.2: *Planning architecture.*

hundred meters). During the computation of risk-aware trajectory the planner accounts for the motion of all the civilian traffic and considers the USVs dynamical model and low-level controller. Finally, the trajectory computed by the trajectory planner is followed using a COLREGs-compliant reactive planner described in one of our previous works [41]. The reactive planner is deployed to ensure collision free operation in the case of emergencies.

In the scenario where the geometrically feasible path computed by the global path planners are not valid and the USV cannot compute a dynamically feasible trajectory, then it sends an exception to the global path planner forcing it to compute a new trajectory. For example, the trajectory planner was unable to find a valid trajectory due to the discrepancy of obstacles regions in the real-world and the geographical map provided to the global path planner. In these situations, the trajectory planner is expected to throw an exception to the global path planner along with the information of the newly discovered obstacles. Thus, the global planner will compute a new path that can be followed by the USV.

Chapter 2

Literature Review

2.1 Dynamic Obstacle Avoidance

In this section, we provide a review of existing research paradigms related to motion planning for highly dynamic environments. Our primary focus is on approaches that consider other robots and human driven vehicles as actively avoiding entities, as opposed to considering them only as passively moving obstacles.

In the broader context of robotics, most of the obstacle avoidance paradigms can be categorized into 1) local reactive approaches that consider static as well as dynamic obstacles only in the vicinity of the robot, 2) deliberative approaches that require the complete state of the environment in order to compute a globally optimal trajectory, and 3) hybrid approaches that use a deliberative planner to compute a nominal trajectory and a reactive planner to follow the nominal trajectory while yielding to dynamic obstacles according to a set of predefined navigation rules (e.g., COLREGs).

The review is divided into three subsections, namely, motion planning algorithms for dynamic environments (see Section 2.1.1), algorithms incorporating COLREGs into planning (see Section 2.1.2), and motion planning algorithms which dynamically scale motion primitives based on the spatio-temporal complexity of the scene to improve the efficiency of the search (see Section 2.1.3).

2.1.1 Motion planning algorithms for dynamic environments

Planning algorithms developed for robotic systems operating in dynamic environments are highly dependent on their kinodynamical constrains [42], the number of look-ahead planning steps, behavior models [43] of dynamic obstacles, etc. In this subsection, wefirst summarize the more widely used reactive or local planning approaches and discuss their limitations. Second, we will present major sampling-based deliberative planning approaches. Finally, we will summarize graph search-based approaches.

Most local obstacle avoidance paradigms [44, 45, 46, 30, 47, 29, 48] compute greedy avoidance actions to avert a collision with obstacles. These planners rely on a very high replanning rate to deal with fast and dynamic moving obstacles. In some cases, the algorithms take the robot's kinematics and dynamics into account when deciding which avoidance action to execute. The other approaches [49, 50, 51, 52, 53, 40] estimate the future positions of obstacles by projecting their observed velocities in time. In an environment with a high traffic density, every obstacle usually yields to one or more other obstacles and thus does not maintain a constant velocity for a finite duration of time. Hence, it is very difficult to infer the future positions of the obstacles by only considering their current velocity vectors. This is the reason why these approaches do not perform efficiently in environments with a large number of obstacles.

Other techniques for dynamic obstacle avoidance [54, 55, 56, 57, 58] assume to have a model of the robot's environment up to a given time horizon. The approaches

typically assume dynamic obstacles to behave strictly as per the given model. However, in reality, the actions performed by the robot generate a reciprocal response by the dynamic obstacles which is very difficult to accurately predict. One approach to improve this prediction is through gathering statistics of the past obstacle's behavior for different configurations of the environment [59, 43].

Several descendants [60, 61, 62] of the classical Velocity Obstacles (VO) technique [39] address the above mentioned problem of reciprocal motion planning. But, these planners rely on the assumption of precisely knowing the avoidance strategies of dynamic obstacles. Also, the approach presented in [63] presents the probabilistic version of VO which computes a collision cone by incorporating the uncertainty in the behavior of the obstacles. These extensions of VO assume the robot to be holonomic and capable of adapting to abrupt changes in the velocity vectors of the obstacles. This assumption is unrealistic given the kinodynamic constraints of physical systems. These methods are extended by [52, 64, 65, 66] for robots with kinodynamic constraints. The approach presented in [52] combines the probabilistic VO [63] with an occupancy grid which incorporates the dynamics and sensing uncertainty of the robot. The approach presented in [64], extends the optimal reciprocal collision avoidance (ORCA) [62] for multiple robots with car-like dynamics. Alonso-Mora in [65] also extends ORCA by adding non-holonomic constraints in the velocity space to achieve dynamically feasible velocity vectors for non-holonomic robots. Recently, Bareiss in [66] extended the concept of LQR-obstacles [67] (extension of VO) for robots with non-homogeneous and non-linear dynamics.

Rufli et. al. [68] proposed a reciprocal local obstacle avoidance method as

an extension to RVO [60]. The technique computes so-called continuous controlled obstacles CO^n-CO which represent feasible feedback controlled trajectories that may lead to a collision. The set outside CO^n-CO represents a set of safe trajectories that ensure continuity of a control sequence rather than piecewise linearity as proposed in VO-based techniques.

There are many hybrid approaches [69, 70, 1, 71, 72, 73, 74, 75] which compute a nominal trajectory by only considering static obstacles and use a local planner to deal with dynamic obstacles when tracking the nominal trajectory. These approaches work reliably when the density of dynamic obstacles in the environment is less and the obstacles do not share any complex reciprocal relationship among themselves and the robotic system.

In the domain of sampling-based motion planning algorithms, the approach presented in [51] uses a variant of the rapidly exploring random tree (RRT) algorithm to compute efficient paths in high dimensional state spaces for robots with kinodynamical constraints. The work introduces a concept of partial motion planning (PMP) which returns partial plans at the end of a given planning interval. The algorithm relies on the inevitable collision state concept to ensure safety of the partial plans. However, in general, the computed paths are not smooth because of the random sampling used during the search. This promblem is solved by recently developed variant of the algorithm named RRT* [76]. In addition, the approach presented in [51] does not consider the uncertainty in the motion of dynamic obstacles.

Kushleyev et al. [77] developed a deliberative motion planning approach as

a variant of the A* algorithm that searches in a time-extended state space to be able to consider future predicted states of dynamic obstacles. The algorithm models dynamic obstacles using the differentially-constrained Ackerman steering model [42]. It estimates their future positions and accounts for the corresponding uncertainty using a sequence of Gaussian distributions with increasing variances in time. The developed approach introduces a concept of a time-bounded lattice, which ignores the presence of a dynamic obstacle after a certain time horizon when the probability of the obstacle being at its mean drops below a given threshold. In addition, the search beyond the time horizon does not consider the time dimension; it only uses position and orientation state variables. This reduces the computational resources and makes the planning efficient. However, this approach also considers obstacles as passive agents (i.e., agents that do not reciprocally react to the robot's maneuvers) and assumes that they maintain their trajectories for the entire planning cycle. This assumption stands void in a cluttered, real-world environment where each dynamic obstacle reacts to actions the robot takes.

Philips et al. introduced an algorithm named SIPP (Safe Interval Path Planning) for path planning in dynamic environments [78]. Unlike the algorithm presented in [77], the authors do not explicitly include the time dimension into the search, which increases the computation time and decreases the re-planning frequency. They define a notion of a safe interval, during which a robot is collision-free for a specified time period if it maintains its current position. Hence, given deterministic paths of dynamic obstacles, a search space can be constructed in which each state consists of the position and the safe interval period. A path can be com-

puted in this new search space by finding safe intervals starting from the robot's current state to the goal state. This planning approach is efficient when the robotic system has the knowledge of the deterministic movement of all dynamic obstacles. However, it is prone to fail in highly uncertain and cluttered environments. Finally, the approach presented in [79] is a combination of [78] and [77] and also assumes deterministic motion of dynamic obstacles.

In a real-world scenario, it is challenging to accurately predict the motion of obstacles. This is because the motion of an obstacle is sometimes dependent on the path the robot chooses to pursue. The developed RCAP algorithm models dynamic obstacles as intelligent decision making agents. In particular, the planner predicts the motion of a vessel in response to the avoidance maneuvers performed by the USV at each node during the search for a trajectory. The RCAP algorithm searches for the trajectory in 5D state space with the time as one of the dimensions. It also reasons about the availability of contingency maneuvers along the trajectory which are executed in response to unpredictable behavior exhibited by obstacle vessels.

2.1.2 COLREGs compliant motion planning algorithms

The International Maritime Organization (IMO) has developed a standardized set of rules called the International Regulations for the Prevention of Collisions at Sea (COLREGs) [28] which serves as a guide for selecting avoidance maneuvers. In order to ensure increased operational safety, it is mandatory for all vessels to comply with these rules throughout their missions.

The development of long-term autonomy for USVs requires incorporating COLREGs directly into their behavior-based architectures. The recent survey by Campbell et al. [80] highlights the fact that a majority of guidance systems does not implement these rules satisfactorily to ensure safe operation of unmanned systems in complex scenes.

Purely rule-based planning systems for USVs which implement COLREGs guidelines [28] include the Springer USV planner [81]. The planner integrates a simple waypoint guidance using line-of-sight (LOS) with a manual biasing schema to follow the rules of the road.

Another approach presented in [82] solves collision avoidance in accordance with COLREGs by choosing an appropriate short-term trajectory or a maneuver from a list of candidate trajectories using a prioritized list of predefined criteria. The criteria include the shortest execution time, safety distance, rules 14, 15, 16, and 17 from the COLREGs guidelines, and minimize the deviation of the vehicle from the user specified path. In [83], the deliberative part of the navigation planner contains a rule-based module to compute trajectories consistent with COLREGs during all planning stages. The developed rule-based system determines the compliance of the planned trajectory with COLREGs by finding points of closest approach along the trajectory and applying a set of hand-coded rules that make decisions based on the course and velocity of other friendly vehicles. A 2D-grid based path planning algorithm (DPSS) was adapted in [84] to generate COLREGs-compliant paths. To handle head-on collision scenarios according to the rule 14, a virtual obstacle is placed on the left side of the vehicle. Similarly for the MEREDITH UAV, a set of

simple rules to handle a head-on situation was developed in [85].

Dynamic obstacle avoidance (OA) combined with the VO method and a technique for computing projected obstacle areas, which moving obstacles could occupy along their future trajectories, is presented in [70]. In this approach, the shape of the projected obstacle area is skewed to allow the OA to comply with the basic rules of the road. Similarly, the VO method was also adapted for following COLREGs in [86] and demonstrated using both simulation and on-water experimental tests. The technique generates a safety buffer to handle the uncertain movement of other boats and prevents oscillation in the execution of rules through hysteresis. The developed technique was integrated into the NASA JPL CARACaS system [87].

The Interval Programming (IP) within a behavior-based architecture for optimal blending of action outputs of behaviors implementing COLREGs has been developed in [3, 88, 89]. Each behavior produces a single objective function over the actuator's space of the vehicle that is combined with objective functions of other behaviors to produce a single action for execution. The developed approach was tested on the MIT kayak platform.

In terms of variations of the potential field method, the work in [90] uses the Virtual Force Field method modified to operate in either collision avoidance or track-keeping modes. The collision avoidance mode has the ability to handle both static and dynamic obstacles and specifically follow the COLREGs guidelines. The Virtual Force Field method computes the overall force generated by the combination of the force for pulling the vehicle toward the next waypoint and the force for directing the vehicle away from an obstacle. The method is parameterized so that the user can

define a typical behavior of the vehicle using fuzzy logic expert rules. These rules influence the behavior of the vehicle in dangerous and safe regions of the space and also serve in solving more complex obstacle avoidance situations. Similarly in [91], the work adapts the artificial potential field method for planning in accordance with COLREGs. A fuzzy logic based decision making process was used to implement COLREGs for autonomous guidance and navigation in [92], [93] and [94].

Evolution-based Cooperative Planning System (ECo PS) was adapted to support COLREGs in [95]. The planner considers head-on and crossing collision scenarios. The work in [96] reported preliminary investigation of COLREGs-compliant collision avoidance maneuvers bas- ed on a biased waypoint guidance and genetic algorithm. The genetic algorithm is used for computation of safe, evasive trajectories compatible with COLREGs. The stu- dy in [97] presents another EA-based path planning algorithm for dealing with multiple obstacles in close-range. The algorithm determines the collision risk with each obstacle and suitability of candidate paths to comply with COLREGS.

Several recent efforts have been made to integrate COLREGs into a path planning algorithm for USVs [86, 1]. However, these approaches can only operate safely in relatively simple scenarios with less civilian traffic. The approaches also assume that each vessel has the same amount of information about the current COLREGs situation and thus it perceives and reacts in the same way. This assumption may not hold in real-world scenarios where every sailor has his own interpretation of COLREGs depending upon his perception, the size, speed, and heading estimate of other vessels in the boat's surrounding. Additional factors include a limited field of

view, environmental conditions (e.g., ocean currents or wind), and the sailor's own experience to infer when COLREGs is breached by any of the civilian vessels, etc. Hence, it is non-trivial to encode standard COLREGs rules into the general path planning framework used in complex and busy scenarios that involve many vessels.

The approaches described above work satisfactorily in low congested scenario with a few civilian vessels. The current state-of-the-art planning approaches for COLREGs compliant obstacle avoidance [86, 1] are local in nature. This is because local planners have lower computational requirements and thus can rely on frequent replanning to handle the dynamicity of the environment. However, reactive planners work satisfactorily only in scenarios with simple and relatively low traffic.

2.1.3 Adaptive-search based motion planning algorithms

The search-based deliberative planners have been widely used in the robotics community for automated guidance of unmanned vehicles [98, 99, 100, 101]. The challenge with these planners is that the computation time significantly increases with the increase in the dimension of the state space, complexity of the motion primitive set, complexity of the environment, number of dynamic obstacles, etc. Hence, researchers have developed several techniques to reduce the computation time to achieve planning within allowable time limit. In our review, we will be primarily focusing on techniques involving grid and lattice-based planning paradigms [33, 102, 103] that were developed to enhance the capability as well as performance of the classical discrete search algorithms such as A* [104], D* Lite [34], Anytime

Dynamic A* [35], etc.

The computation efficiency of a path planning algorithm is highly dependent on the resolution of the grid that is used to discretize the robot's continuous state space. This selection is dependent on several factors such as the minimum distance between obstacles, size of the obstacles, the speed of the obstacles, the complexity of the control action set of a robot, etc. However, the greater the resolution is, the higher is the computation time, and better is the quality of a computed path and vice-versa.

In recent years, several extensions to the standard grid-based algorithms have been proposed to achieve balance between their computational efficiency and trajectory optimality. For example, the Accelerated A* technique developed by Šišlák et al.[105, 106] varies the magnitude of the transition between the current state and its neighbor based on the available free space. This magnitude is determined through finding an empty square region with the maximum size around the current state. The detection of the maximum size of the square is quad- ratic in the number of cells. This approach closely resembles our approach. However, our planner expands control primitives (i.e., determines the magnitude of state transitions) in an asymmetric way. In particular, the presence of an obstacle in one direction does not affect the magnitude of the state transition in the other. Also, in our application, the position of an obstacle is dynamic and stochastic in nature. Hence, the algorithm needs to compute a collision probability to determine the magnitude of a specific state transition. Our algorithm gradually varies the magnitude of the state transition at every successor state with respect to the collision probability. Hence,

the magnitude of a state transition is determined in constant time.

In addition, Yap et. al. developed an algorithm called Block A* [107, 108] in which the whole workspace is divided into several disjoint rectangular blocks of a predefined constant size. Each block is further discretized into grid cells. For each block, the algorithm precomputes a database of all possible configurations with free and obstacle cells, and the list of optimal distances for each pair of boundary cells for each obstacle configuration. Unlike standard A*, the Block A* expands each block at a time, instead of expanding each cell as it is in the standard A*. It looks up the best optimal path through a particular block.

Barraquand and Latombe in [109] proposed a motion planning algorithm to find a resolution complete path for a non-holonomic mobile robot. The algorithm expands a search tree using control of non-holonomic mobile robot and is dependent on the time discretization and depth of the search tree. If the solution exists, the algorithm guarantees to provide one by finely discretizing the time space and increasing the depth of the search tree. Thus, making the search exhaustive. Lindemann in [110] developed a multi-resolution algorithm which incrementally reduces the time discretization and increases the depth of a search tree until the feasible solution is found. Thus, one can quickly find feasible paths without exhaustively searching the space. Unlike our algorithm, this approach does not vary the resolution based on the distribution of obstacles and is unable to deal with the uncertainty in position of the obstacles.

The computation time of a path planning algorithm is also dependent on the dimension of the state space. Researchers in the robotics community have devel-

oped several problem-specific algorithms which vary the set of motion primitives used during the search to save computational resources. For example, Cohen et al. demonstrates a variant of the A* algorithm with adaptive motion primitives using a 7 DOF manipulator in [111]. In this work, the authors define a set of motion primitives used in a state space with reduced dimension (i.e., defined by the number of joints that are modified by the motion primitives). They also define a set of primitives used in a state space with full dimensionality. The algorithm switches from the reduced primitive set to a complete set when the currently expanded set is below a certain threshold distance from the goal state. Another recent example includes the work by Gochev et al. [112, 113, 114], where the algorithm starts with a lower dimensional state space and iteratively keeps adding higher dimensional states to the state space until it finds an entire trajectory made up of a sequence of higher dimensional states. In other words, the algorithm iteratively decides where to replace lower dimensional regions by higher dimensional ones in order to make the trajectory feasible for the robot to successfully execute. In our previous work [101], we developed a planning approach which utilizes a set with a higher number of motion primitives in regions close to the initial state of an unmanned vehicle and vice versa in distant regions to improve the computational efficiency.

The developed RCAP algorithm searches in a 5D state space which requires high computational resources. We have enhanced RCAP to dynamically scale control action primitives depending upon the estimated spatio-temporal complexity of the workspace. This significantly improves the computational performance of the planner. It exploits the fact that dense sampling is only required in highly complex

regions of the workspace where the paths of most civilian vessels intersect. However, it sparsely samples the regions of the workspace which are relatively less complex with lower civilian traffic.

2.2 Path Planning

2.2.1 Path planning in geometric spaces

Path planning is a well-studied problem in robotics and AI communities. Many different approaches have been developed to solve the path planning problem [42, 115]. The body of work that is most closely related to the theme of this dissertation is the path planning problem for a given complete map. We will review methods that deal with known stationary obstacles with no uncertainty in the environment or the outcome of the vehicle actions. Readers are referred to [116, 117, 118] for an overview the planning methods in partially known maps. Methods for planning under uncertainty are discussed in [119]. The planning methods developed for dealing with dynamic obstacles is given in [62, 120, 121, 86].

Path planning problems over the long distances can be divided into two categories. The first category includes problems where configuration spaces associated with the collision-free regions of the space can be easily computed explicitly. Problems in 2D workspaces (i.e., 3D configuration spaces) belong to this category. Path planning for unmanned surface vehicles belongs to this category. The second category belongs to the class of problems where explicitly computing configuration space is computationally challenging. Sampling based methods such as Rapidly Exploring

Random Trees (RRT) [36] and Probabilistic Road Maps (PRM) [122] have been successfully used to deal with such problems. In this dissertation, we will focus on methods that use explicitly computed configuration spaces.

Finding optimal paths requires abstracting the given configuration space into a discrete graph over which search can be performed to compute the optimal path. If the application requires solving the planning problem multiple times over the same configuration space, then it is useful to construct roadmaps [123], and Voronoi graphs [124, 125] associated the configuration space. Even though this takes significant computational effort upfront, the roadmap and/or Voronoi graphs can be reused over the multiple planning instances. If the planning problem is not being solved multiple times, then it is computationally preferable to construct the relevant portions of the search graph on-the-fly. In this dissertation, we are interested in methods that do not precompute the search graph.

There are three main methods for representing the path planning problems as graph search. The first class of methods represents the configuration space as uniform grids [126, 104, 34, 35] or multi-resolution grids [107]. At any point in the grid, the vehicle can move only to the adjacent grid points using a fixed number of actions. This method limits the branching in the search trees, but leads to a large number of nodes in the search tree. Paths produced by these methods are not smooth and often not optimal. The second class of the methods is based on the idea that the optimal path will move the vehicle in straight line paths between obstacles and it will only pass through visible vertices of the obstacles. The underlying representation used during the search is called visibility graph [127, 128, 129]. These methods

significantly reduce the number of nodes in the search graph. However, the branching factor can be high. This leads to computationally slow performance when the spatial region over which the planning is being done is large. These methods produce optimal paths. Recent development in any-angle search represents a third class of methods. These methods combine features from the above two class of methods and limit the branching at the search node and yet do not constrain the vehicle to move along the grid edges. These methods are fast and produce significantly better paths than the grid based methods. However, paths produced by these methods may not be optimal. Notable methods belonging to this class are Incremental Phi* [130], Theta* [131] and its variants [132, 133, 134], and Field D* [135].

Several researchers have used quadtrees as the underlying representation for path planning [136]. Recent work in the area that uses quadtree to represent the operating environment of the robot included [137], [138].

2.2.2 Path planning in time-varying flow fields

Several path planning approaches to realize energy-efficient, autonomous operations of robotic systems in non-linear, time-varying fields were developed in the past. In particular, a purely local path optimization technique was developed by Kruger et al. [139] to allow autonomous guidance of an autonomous underwater vehicle (AUV) in a fast flowing tidal river. The technique employs a gradient based approach to locally modify an initial straight path between two given locations according to a predefined cost function. Similarly, Witt et al. [140] developed a technique

for searching paths in a time-extended state space that balances the path execution time and energy requirements. The approach searches over a predefined set of global static paths represented as splines and is combined with a local random, simulated annealing-inspired search. This choice of search techniques, however, does not allow a systematic exploration of complex search spaces.

Thompson et al. [141] developed a wavefront based path planning algorithm for an UAV to follow paths that ensure fastest arrival of the vehicle to given locations through uncertain, time-varying current fields. The algorithm, however, does not explicitly balance the energy expenditure of the vehicle with the path execution time, prune the search space, or account for the uncertain dynamics of the field (contrary to the claims in the paper). Similarly, the approach developed by Lolla et al. [142] is also based on the forward evolution of a wavefront from the initial to the goal vehicle's states, and determining a series of states along the evolving wavefronts that optimize a given objective (in this case, travel time). In contrast to this approach, the technique presented in this paper prioritizes states during the expansion, which increases its computational performance. In addition, the use of the free-flow action allows a variable resolution search. Soulignac [143] developed a sliding wavefront expansion algorithm that computes physically controllable, globally optimal and feasible paths for a vehicle operating in an environment with strong currents (i.e., currents that may overcome the physical capabilities of the vehicle). Although theoretically sound, the algorithm is based on the classical wavefront expansion and as such does not consider the evolution of currents in time.

Garau et al. [144] evaluated several heuristic functions as candidate compo-

nents of the A* algorithm. The developed approach, however, is suitable only for static fields. Similarly, Isern-Gonzalez et al. [145] developed a method based on the A* and Nearest Diagram (ND) algorithms. The method finds an initial path that is further locally optimized. The A* algorithm was also combined with the fast marching algorithm into the FM* algorithm in [137] that has the capability of computing smooth paths in continuous environments. However, the work does not explicitly address the energy-efficiency as well as time-varying fields.

Al-Sabban et al. [146] developed an energy-efficient path planning algorithm for an unmanned aerial vehicle (UAV) operating in an uncertain wind field. The problem was defined as a Markov Decision Process (MDP) to consider local, stochastic nature of the field vectors. The algorithm, however, does not explicitly account for a time-varying field. The work was further adapted in [147] for path planning of AUVs.

Sampling-based methods were also used for energy-efficient, probabilistic-complete path planning. For example, a path planner based on the Rapidly Exploring Random Trees (RRT) was introduced in [148] for computing paths to realize a long-term, autonomous operation of underwater gliders.

Long-term path planning in time-varying fields with obstacles is computationally as well as space expensive due to the large size and complexity of the search space. Most recently, the approach developed by Fathpour et al. [149, 150] computes paths or navigation functions for autonomous guidance and reachability analysis of a hot-air balloon in time-invariant, time-varying, as well as stochastic wind fields. The approach allows space and computationally (as a by-product of

the space-efficient planning) efficient planning through decomposing the planning problem into subproblems, and solving each of them sequentially.

Chapter 3

Resolution-Adaptive Risk-Aware Trajectory Planning for Surface Vehicles Operating in Congested Civilian Traffic

In this chapter[1], we present a lattice-based 5D trajectory planner for unmanned surface vehicles (USVs) operating autonomously over long time horizons in environments with significant civilian traffic.

3.1 Introduction

The growing variety and complexity of research and application oriented tasks requires unmanned systems to operate fully autonomously over long time horizons even in environments with significant traffic comprised of manned, commercial, and recreational vehicles (these vessels are referred to as "Civilian Vessels" (CVs) in this chapter). The complexity of the environments creates significant challenges for autonomous avoidance of CVs by the USVs. For example, the vessels as well as the unmanned systems have different dynamic characteristics depending on their types and dimensions which in turn affects their braking distance and minimum turning radius.

Let us consider an example of a harbor scenario shown in Figure 3.1. The USV enters the harbor from the south channel with the objective of reaching the

[1] The work in this chapter is derived from the published work in [38] and accepted work in [151]

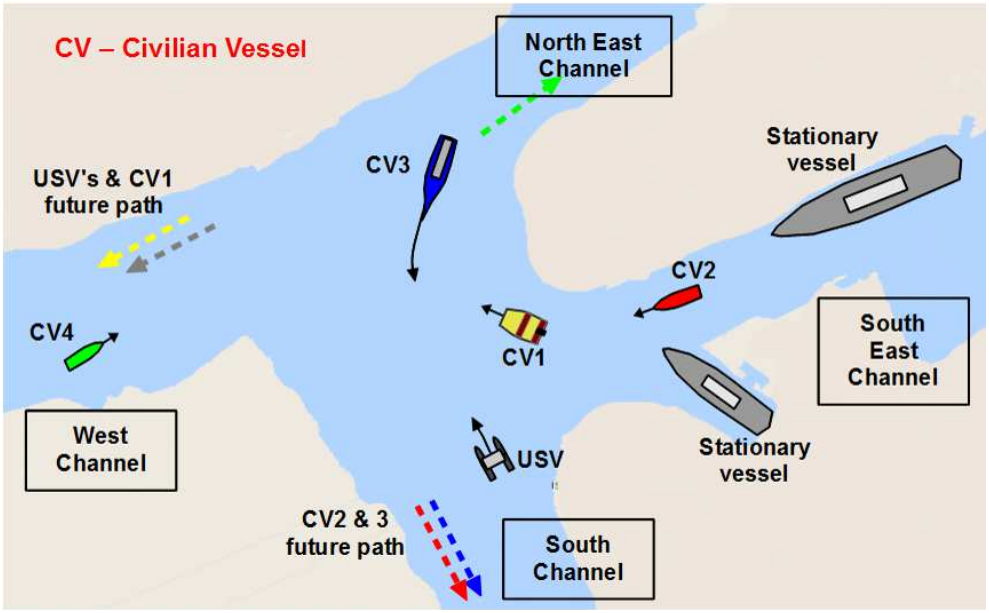


Figure 3.1: *A harbor scenario with an USV and several civilian vessels approaching their destinations.*

west channel. Under ideal conditions, each vessel is assumed to follow COLREGs while avoiding its fellow vessels.

As per the scenario described in Figure 3.1, CV1 is crossing from the right side and the COLREGs "Rule 15" applies (see Figure 3.2). In this situation, the USV can either yield to CV1 (the vessel has the right of way) by slowing down to a steady state or passing it to the right. Or alternatively, the USV can breach COLREGs by not yielding to CV1 and CV3 while moving to the west channel.

In both cases, CV1 is under a "crossing-from-right situation" with respect to CV3, and thus it is supposed to yield to CV3 from right. It may be in CV1's best interest to slow down and avoid passing through the narrow space between CV3 and the land mass. This is mostly because of the risk of collision with the land mass.

If the USV chooses the first option, i.e., to slow down and wait for CV1 to

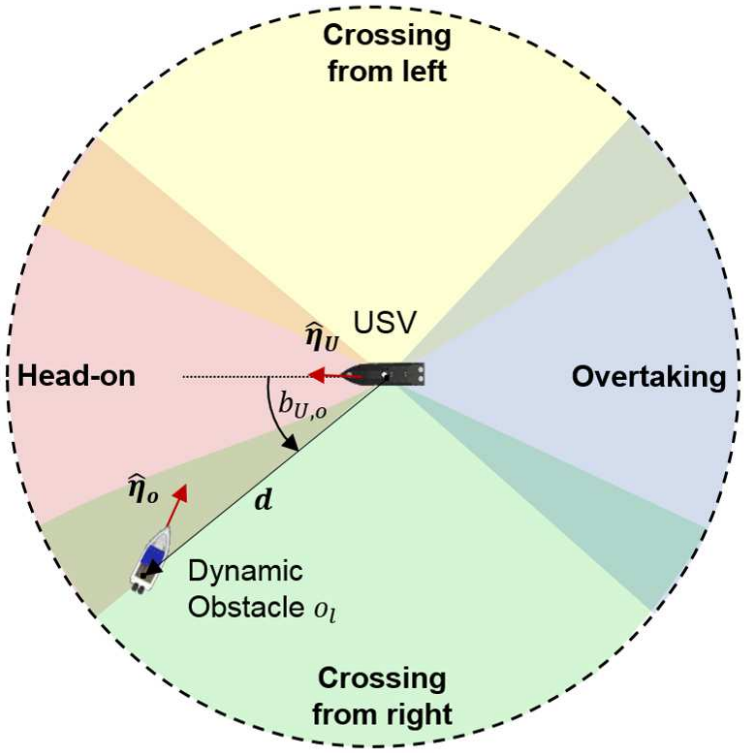


Figure 3.2: *COLREGs head-on, crossing from right, and overtaking behaviors.*

clear its way, then it will block the entire south east channel for some time and thus obstruct the way of CV2. In such a scenario, the USV as well as CV2 will have to wait and try to maintain their states and avoid collisions with each other as well as the surrounding land areas. This may not only be highly inefficient, but also risky. In a marine environment, it may be a challenge to maintain vessel position because of forces due to winds, ocean waves and wake generated due to the motion of other vessels.

On the contrary, if the USV breaches COLREGs with respect to CV1 and CV3, the intersection of the south and south east channels becomes free for CV2 to pass through. However, in this case, the USV will need to evaluate the collision risk with the vessels (e.g., CV4) coming out of the west and north east channels.

Although the USV has the right of way with respect to the vessels coming from the west channel, the vessels might not be able to see the USV because of the limited visibility due to the land area. In this example, we assume the USV knows about the presence of CV4 in the west channel and is able to execute the appropriate avoidance maneuvers.

This example shows that trajectory planning in an environment with significant civilian traffic requires enhanced reasoning about the motion of each vessel in the entire scenario in response to a planned trajectory of the USV.

Safe navigation in a highly dense and dynamic environment is a challenging problem [152] and requires good prediction capabilities of the future movement of dynamic obstacles. In addition, the trajectory planner needs to perform enhanced reasoning about the risk associated with each avoidance maneuver by predicting response behaviors of civilian vessels. It also needs to reason about the availability of contingency maneuvers to counteract the unpredicted behaviors of the vessels. These advanced features cannot be incorporated into local planners [1, 29, 30] with a limited number of look-ahead steps.

On the other hand, traditional, lattice-based, deliberative planners [31, 32, 33] require a significant amount of computation to find a global trajectory that optimizes a given performance measure. This is mostly because they employ multi-step look-ahead search in a higher dimensional state space to globally reason about the collision risk. Even though the operating frequency of these planners is generally lower than that of the local planners, it is still necessary that they keep their computational efficiency at a satisfactory level.

The classical lattice-based deliberative approaches use control action primitives with constant execution times while searching the state space for dynamically feasible trajectories. The planners that use control primitives with short time scales are successful in finding trajectories in highly complex and cluttered scenarios, but they are computationally expensive, which leads to low re-planning rates. On the other hand, the approaches that use primitives with longer time scales fail to produce effective trajectories in dense scenarios, even though they are computationally efficient and thus have a satisfactory replanning rate. So, the right balance between the computational demand and optimality of a trajectory can be achieved by finding the correct combination of long and short time-scaled control action primitives.

We have developed an adaptive, risk and contingency-aware trajectory planning algorithm (A-RCAP, see Section 3.4) for dynamically scaling control action primitives based on estimated spatio-temporal complexity of the local work-space around each state being expanded during the search for a trajectory. This spatio-temporal complexity depends on the distribution and concentration of civilian vessels and their future predicted trajectories. It also depends on the history of spatio-temporal complexity of the states along a trajectory leading to the local workspace. The planner estimates the complexity of each region around the USV by actively probing the state space during the search. It dynamically scales the control action primitives while preserving their dynamical feasibility. Dividing the workspace into several regions allows the planner to independently scale the primitives.

The developed algorithm integrates a lattice-based, risk and contingency-aware planner (RCAP) that search- es for trajectories in a 5D state space and reasons about

the collision risk and availability of contingency maneuvers (see Section 3.3). Incorporating the knowledge of avoidance behaviors of civilian vessels into trajectory planning allows us to find safe trajectories in complex and congested scenes. Integrating contingency maneuvers into a trajectory significantly reduces the collision rate of the USV with the vessels that may breach COLREGs or behave unpredictably in general.

Our results in Section 3.5.4 demonstrate that the developed planners significantly reduce the number of collisions in comparison to a baseline variation of the Velocity Obstacles (VO) planner [86]. In order to increase the replanning frequency, the adaptive planner incorporates the capability of dynamically scaling control action primitives during the search for a trajectory. This leads to significant reduction in the number of states expanded and thus improved performance. In particular, the results presented in Section 3.5.4 shows a 500% reduction in the number of states expanded while not sacrificing the quality of computed trajectories. This computational enhancement leads to greater replanning frequency which in turn leads to shorter trajectories with smaller execution times as compared to the baseline VO-based planner [86].

3.2 Problem Formulation

3.2.1 Definitions

Let $\mathcal{X} = \mathcal{X}_\eta \times \mathcal{X}_\nu \times \mathcal{T}$ be a continuous state space of the USV that consists of states $\mathbf{x} = [\eta^T, \nu^T, t]^T$. Here, $\eta = [x, y, \psi]^T \in \mathcal{X}_\eta \subset \mathbb{R}^2 \times \mathbb{S}^1$ is the USV's pose

and $\nu = [u, v, r]^T \in \mathcal{X}_\nu \subset \mathbb{R}^3$ is its velocity composed from the surge u, sway v, and angular r speeds about the z axis in the North-East-Down (NED) coordinate system [153], and $t \in \mathcal{T}$ is time. A lower dimensional, discrete 5D version of this space is defined as $\mathcal{S}$. Each state $\mathbf{s} = [x, y, \psi, u, t]^T \in \mathcal{S}$ consists of the position, orientation, surge speed, and time state variables.

Let $\mathcal{U}_c(\mathbf{x_U}) \subset \mathbb{R} \times \mathbb{S}^1$ be a continuous, state-dependent, control action space of the USV in which each control action $\mathbf{u_c} = [u_d, \psi_d]^T$ consists of the desired surge speed u_d and heading ψ_d variables. A discrete version of this set is defined as $\mathcal{U}_{c,d}(\mathbf{s_U})$. Each discrete control action in $\mathcal{U}_{c,d}(\mathbf{s_U})$ is defined as $\mathbf{u_{c,d}}$.

Let $\mathcal{U}_e(\mathbf{x_U}) \subset \mathcal{U}_c(\mathbf{x_U})$ be a set of contingency maneuvers. A discrete version of this subset is defined as $\mathcal{U}_{e,d}(\mathbf{s_U})$ which allows to keep the computation requirements within given bounds. Each discrete control action in $\mathcal{U}_{e,d}(\mathbf{s_U})$ is defined as $\mathbf{u_{e,d}}$.

Let $\dot{\mathbf{x}}_\mathbf{U} = f_U(\mathbf{x_U}, \mathbf{u_h})$ be a 3 degree of freedom (DOF) dynamic model [1] of the USV. The simulation model of the vehicle includes actuators that produce thrust and moment by taking $\mathbf{u_h}$ as the control input. This control input is determined by the controller $h_U(\mathbf{x_U}, \mathbf{u_c}, P_U)$, where P_U is the set of its parameters.

Let $B = \{b_l\}_{l=1}^L$ be a set of all the civilian vessels, where b_l represents a single civilian vessel and L is the total number of civilian vessels. Let $\mathcal{X}_{b_l}$ denote the continuous state space of a civilian vessel b_l. The estimated state of b_l is given as $\mathbf{x_{b_l}} \in \mathcal{X}_{b_l}$. The geometric region occupied by all the civilian vessels is defined as $\mathcal{O}_B = \bigcup_{l=1}^L b(\mathbf{x_{b_l}}) \subset \mathbb{R}^2$. Similarly, the geometric region occupied by static obstacles is defined as $\mathcal{O}_S = \bigcup_{k=1}^K o_{s,k} \subset \mathbb{R}^2$.

Finally, let $m(\mathbf{x_U}, B, \{\mathbf{x_{b_l}}\}_{l=1}^L, \{O_l\}_{l=1}^L, \mathcal{O}_S)$ be an intention model that is used

to estimate future trajectories of the civilian vessels (see Section 3.3.4). Here, $\{O_l\}_{l=1}^{L}$ are history states of the civilian vessels.

3.2.2 State action space representation

The continuous state space $\mathcal{X}$ is discretized into a lower-dimensional 5D state space $\mathcal{S}$. Each state $\mathbf{s} = [x, y, \psi, u, t]^T \in \mathcal{S}$ consists of the position, orientation, surge speed, and time state variables. The continuous control action space $\mathcal{U}_c(\mathbf{x})$ is discretized into a discrete control action set $\mathcal{U}_{c,d}(\mathbf{s})$ consisting of control actions that allow dynamically feasible transitions between adjacent states in $\mathcal{S}$. A discrete control action $\mathbf{u_{c,d}} = [u_d, \psi_d]^T \in \mathcal{U}_{c,d}(\mathbf{s})$ is defined as a combination of the desired surge speed u_d and heading ψ_d of the vehicle.

For planning purposes, the discrete control actions are mapped to motion primitives $\{[x, y, \psi, t]^T\}_{i=1}^{L_k}$ of desired poses and arrival times of the USV. We have used 3 degrees of freedom system identified model of the USV [1] to generate the motion primitives. The continuity of a trajectory is retained by selecting the final state of each motion primitive in the center of its respective state space cube.

We have designed a discrete contingency control action set $\mathcal{U}_{e,d}(\mathbf{s}) \subset \mathcal{U}_c(\mathbf{x_U})$. This control action set consists of extreme input values of the surge speed and heading state variables, i.e., $u_{d,min}$, $u_{d,max}$, $-\psi_{d,max}$, $\psi_{d,0}$, and $\psi_{d,max}$. This contingency control action primitive set is used to determine the overall safety of the USV's trajectory. In particular, the planner incorporates the collision probability of executing the contingency maneuvers into the evaluation of the overall collision probability

of the trajectory (see Section 3.3.1). The contigency maneuvers are a vital part of the nominal trajectory and can be executed in response to any of the civilian vessel breaching COLREGs.

The designed state-action space enables us to perform efficient discrete search. It captures the vehicle's differential constraints and thus ensures computation of smooth and dynamically feasible trajectories. The user can define the resolution of the state-action space depending upon the complexity of the scene and the dynamic characteristics of the USV.

3.2.3 Problem statement

We are interested in developing a deliberative planning algorithm for efficient computation of dynamically feasible, risk and contingency-aware trajectories for the USV. The trajectories should minimize the execution time and risk of collision with obstacles. Further, the algorithm should reason about the availability of contingency maneuvers along a planned trajectory in order to avoid an imminent collision due to unpredictable, COLREGs breaching behaviors of civilian vessels. In addition, the civilian vessels are assumed to be intelligent decision making agents whose behaviors are influenced by the maneuvers of the USV and other vessels. This requires the algorithm to perform enhanced reasoning regarding the reciprocal avoidance strategies exhibited by the civilian vessels in response to the planned trajectory of the USV.

More specifically, given, (1) a continuous state space $\mathcal{X}$ of the environment,

(2) the initial state $\mathbf{x_{U,I}}$ of the USV, (3) the final state $\mathbf{x_{U,G}}$ of the USV, (4) a 3 degree of freedom dynamic model f_U of the USV, (5) static obstacle regions $\mathcal{O}_S$, (6) estimated states $\{\mathbf{x_{b_l}}\}_{l=1}^{L}$ of civilian vessels B, (7) the geometric regions $\mathcal{O}_B$ occupied by the vessels, and (8) the intention model m along with a classifier c for predicting future trajectories of the civilian vessels.

The task is to compute a collision-free, dynamically feasible trajectory $\tau : [0, T] \to \mathcal{X}$ such that $\tau(0) = \mathbf{x_{U,I}}$, $\tau(T) = \mathbf{x_{U,G}}$ and its cost is minimized. Each state $\mathbf{x_U}(t)$ along τ thus belongs to the free state space $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{obs} = \{\mathbf{x_U}(t) | U(\eta_{\mathbf{U}}(t)) \cap \mathcal{O}(t) = \emptyset\}$ for $t \in [0, T]$, where $\mathcal{O}(t) = \mathcal{O}_S \cup \mathcal{O}_B(t)$ and $U(\eta_{\mathbf{U}}(t)) \subseteq \mathbb{R}^2$ is a region occupied by the USV at $\eta_{\mathbf{U}}(t)$.

We assume civilian vessels to follow COLREGs unless the classifier $c(b_l, O_l)$ reports otherwise. We also assume that the USV uses a Kalman filter [154] to estimate its own state and that the states of civilian vessels are either provided through radio communication or estimated using Kalman filtered sensor information acquired by the USV.

3.3 Risk and Contingency-Aware Trajectory Planning

The deliberative risk and contingency-aware trajectory planner (RCAP) searches in a 5D state space to compute a low risk, dynamically feasible trajectory $\tau : [0, T] \to \mathcal{X}$ from the initial state $\mathbf{x_{U,I}}$ to the goal state $\mathbf{x_{U,G}}$. The entire trajectory τ is computed by concatenating action primitives from $\mathcal{U}_{c,d}$. The primitives are designed using system identified dynamics model of the USV [1], and thus guaranteeing the

dynamical feasibility of τ. The planner is based on the lattice-based A* heuristic search [33].

3.3.1 Modeling risk consideration in cost function

The cost function used during the search for a trajectory τ is given by $f(\mathbf{s}') = g(\mathbf{s}')+\epsilon h(\mathbf{s}')$, where $g(\mathbf{s}')$ is *cost-to-come*, $h(\mathbf{s}')$ is *cost-to-go*, and ϵ is the parameter for balancing the computational speed of the search and the optimality of the trajectory. The expected *cost-to-come* from the initial state $\mathbf{s_I}$ and the current state under evaluation $\mathbf{s}'$ is given as follows.

$$\begin{aligned} g(\mathbf{s}') &= g(\mathbf{s}) + p_s(c_{g,s'}/c_{g,max}) \\ c_{g,s'} &= (1 - p^n_{c,\mathbf{s}'})c_{s'} + p^n_{c,\mathbf{s}'}((1 - p^e_{c,\mathbf{s}'})c_e + p^e_{c,\mathbf{s}'}c_{e,c})) \end{aligned} \tag{3.1}$$

In the above equation, $\mathbf{s}$ is the predecessor state and $\mathbf{s}'$ is the current state under evaluation. The probability of success to reach the state $\mathbf{s}$ from the initial state $\mathbf{s_I}$ without any collision over K trajectory segments is given by $p_\mathbf{s} = \prod_{k=1}^{K} 1 - p_{c,\mathbf{s_k}}$, where $p_{c,\mathbf{s_k}}$ is the collision probability of transition between two consecutive states $\mathbf{s}_k$ and $\mathbf{s}_{k+1}$ (see Section 3.3.2 for calculation of collision probabilities). The transition cost between $\mathbf{s}$ and $\mathbf{s}'$ by using control action $\mathbf{u_{c,d}}$ is given by $c_{s'} = \omega_n(\omega_c t_{s,s'}/t_{max} + (1 - \omega_c)d_{s,s'}/d_{max}) + c_{\neg COLREGs}$, where ω_n and ω_c are user-specified weights, $t_{s,s'}$ is the execution time, $d_{s,s'}$ is the length of the control action, and $c_{\neg COLREGs}$ is the penalty for the state $\mathbf{s}'$ being in a COLREGs-breach region (see Section 3.3.3). Finally, the cost of executing emergency contingent action $\mathbf{u_{c,e}}$ is given by c_e and the cost of collision during the execution of contingency action is $c_{e,c}$. The user defined

cost weights follow the order $c_{s'} < c_e < c_{e,c}$. Hence, the maximum value which $c_{g,s'}$ can take is equal to $c_{g,max} = c_{e,c}$, when $p^n_{c,\mathbf{s}'} = p^e_{c,\mathbf{s}'} = 1$.

The *cost-to-go* from the current state $\mathbf{s}'$ to the final goal state $\mathbf{s_G}$ is the weighted sum of the heuristic time estimate and distance required by the USV to reach $\mathbf{s_G}$, and is given by $h(\mathbf{s}') = \omega_c(t_{s',s_G}/t_{max}) + (1-\omega_c)(d_{s',s_G}/d_{max})$.

3.3.2 Calculation of collision probabilities

Evaluation of each state requires calculation of several types of collision probabilities during the search for a trajectory τ.

(i.) $p^n_{c,\mathbf{s}'}$ is the probability of collision when executing a control action $\mathbf{u_{c,d}}$ from $\mathbf{s}$ to reach its neighboring state $\mathbf{s}'$. It is calculated by taking a weighted sum of $p_{c,\mathbf{s}',U}$ (see Definition ii.) and $p_{c,\mathbf{s}',B}$ (see Definition iii.) given by $p^n_{c,\mathbf{s}'} = e^{-\gamma t_{s_I,s}}((1-\omega_{c,U,B})p_{c,\mathbf{s}',U} + \omega_{c,U,B}\, p_{c,\mathbf{s}',B})$. In the above equation, $\gamma \geq 0$ is a discount factor and $t_{s_I,s}$ is the total traversal time of the USV to arrive at the current state $\mathbf{s}$ being expanded from the initial state $\mathbf{s_I}$.

(ii.) $p_{c,\mathbf{s}',U}$ is the probability of collision of the USV with civilian vessels B. Let $p_{c,\mathbf{s}',U,b_l} =$ $\iint_{[x,y]^T \in \mathcal{P}_c} p_{b_l,t}(x,y;\mu_{b_l,t},\sum_{b_l,t})\,\mathrm{d}x\,\mathrm{d}y$ be the probability of collision of the USV with the civilian vessel b_l. Here, $\mathcal{P}_c = \{[x_{b_l,t}, y_{b_l,t}]^T | b_l(x_{b_l,t}, y_{b_l,t}) \cap U(\eta_{\mathbf{U}}{}^T(t)) \neq \emptyset\}$ is the set of all locations at which the civilian boat b_l may collide with the USV at time t. The geometric regions occupied by the civilian vessel b_l and the USV is given by $U(\eta_{\mathbf{U}}{}^T(t))$ and $b_l(x_{b_l,t}, y_{b_l,t})$, respectively. Finally, $\mu_{b_l,t}$

and $\sum_{b_l,t}$ are the mean and the covariance matrix of the uncertainty in the position of civilian vessel b_l at time t (see Section 3.3.4). Then, the collision probability with respect to all the vessels is $p_{c,\mathbf{s}',U} = 1 - \prod_{l=1}^{L}(1 - p_{c,\mathbf{s}',U,b_l})$.

(iii.) $p_{c,\mathbf{s}',B}$ is the probability of collision among the civilian vessels themselves. In this chapter, we assume the positions of all civilian vessels to be normally distributed. Hence, computing $p_{c,\mathbf{s}',B}$ is a computationally demanding task. In order to maintain the efficiency of the search, we precompute the collision probabilities $p_{c,\mathbf{s}',b_i,b_j}$ of the civilian vessels b_i and b_j by sampling their bivariate Gaussian probability densities. More specifically, if the distance between the two sampled positions of the civilian vessels is less than the user-specified distance threshold $d_{c,min}$, then we consider it as a collision. The user-specified threshold $d_{c,min}$ is estimated based on the size and profile of the civilian vessels. The collision probability $p_{c,\mathbf{s}',b_i,b_j}$ is then calculated by taking the ratio of the number of samples that resulted in collision to the total number of samples. The calculated look-up table outputs the probability $p_{c,\mathbf{s}',b_i,b_j}$ given the input discrete values of variances $\sigma^2_{x,i}$, $\sigma^2_{y,i}$ (see Equation 3.2) of the Gaussian probability density function of b_i, discrete values of variances $\sigma^2_{x,j}$, $\sigma^2_{y,j}$ of the Gaussian probability density function of b_j, the relative mean position x_j, y_j of b_j with respect to b_i, and the relative mean heading ψ_{b_j}. The probability of collision is then calculated as $p_{c,\mathbf{s}',B} = \max_{b_i \in B, b_j \in B, i \neq j} p_{c,\mathbf{s}',i,j}$.

(iv.) $p^e_{c,\mathbf{s}'}$ is the probability of collision of a contingency control action primitive $\mathbf{u_{c,e}}$ when transitioning between the state $\mathbf{s}$ to the contingency state $\mathbf{s}'_e$. It

is calculated as $p^e_{c,\mathbf{s}'} = \min_{\mathbf{u_{e,d,i}} \in \mathcal{U}_{e,d}} p^e_{c,\mathbf{s}',i}$, where $\mathcal{U}_{e,d}$ is the discrete set of contingency control action primitive (see Section 3.2.2), and $p^e_{c,\mathbf{s}',i}$ is calculated the same way as $p_{c,\mathbf{s}',U}$ (see Definition ii.).

3.3.3 Evaluation of USV's state for COLREGs compliance

Each candidate USV's state $\mathbf{s}$ is evaluated for its COLREGs compliance with respect to all civilian vessels during the search for a trajectory. Primarily, we determine whether the USV at the state $\mathbf{s}$ is on a collision course with any of the civilian vessels. This is evaluated through the conditions $d_{CPA} < d_{CPA,min}$ and $t_{CPA} < t_{CPA,max}$. Here, the closest distance between a civilian vessel and the USV when it follows its planned trajectory for a given time horizon is regarded as the closest point of approach (CPA). The computed distance from the current USV's state $\mathbf{s}$ to CPA is termed as the distance to CPA, d_{CPA}, and the time to reach CPA at planned surge speed is termed as the time to CPA, t_{CPA} [1] (See Figure 3.3). The user-specified distance $d_{CPA,min}$ and time $t_{CPA,max}$ thresholds are determined based on the minimum turning radius, maximum surge speed, and acceleration of the USV.

If the above stated primary conditions hold true, the USV determines the appropriate COLREGs rule such as, for example, the "head-on" (rule 14), "crossing from right" (rule 15), and "overtaking" (rule 13) [28] as described in [1]. If the USV's state $\mathbf{s}$ is in any of the "give-away" situations, the planner evaluates the state $\mathbf{s}$ for its COLREGs compliance with respect to the appropriate COLREGs rule.

More specifically, let $\eta_{\mathbf{U}}(t)$ and $\eta_{\mathbf{U}}(t_0)$ be the poses of the USV at $\mathbf{s}$ and its

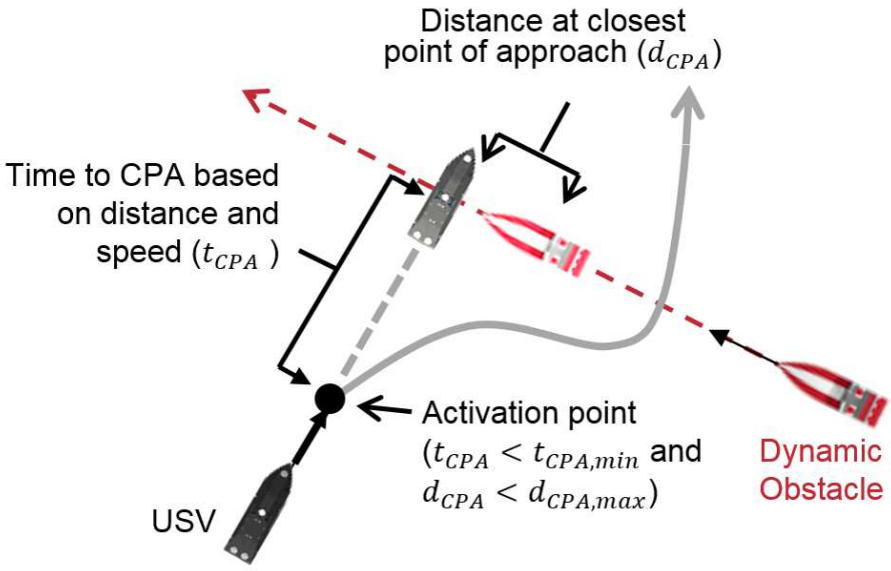


Figure 3.3: *Calculation of CPA time and CPA distance.*

parent state $\mathbf{s_0}$, respectively, and t be the actual transition time between the states. Let $\eta_\mathbf{b}(t)$ be the pose of the civilian vessel at time t. Let $\hat{\mathbf{n}}_{U,b} = [n_{U,b,x}, n_{U,b,y}]^T$ be the unit vector in the direction between $\eta_\mathbf{U}(t_0)$ and $\eta_\mathbf{b}(t)$. Let $\hat{\mathbf{n}}_{U,t_0,t}$ be the unit vector in the direction between $\eta_\mathbf{U}(t_0)$ and $\eta_\mathbf{U}(t)$. Then, the state $\mathbf{s}$ is COLREGs-compliant with respect to a civilian vessel b_l in NED if $([-n_{U,b,y}, n_{U,b,x}]^T \cdot \hat{\mathbf{n}}_{U,t_0,t}) > 0$ ($\mathbf{s}$ is in the right half-plane between $\eta_\mathbf{U}(t_0)$ and $\eta_\mathbf{b}(t)$) [1].

3.3.4 Intention motion model of civilian vessels

In this chapter, we consider civilian vessels to be intelligent decision making agents. Hence, the planning algorithm needs to account for reciprocal avoidance maneuvers performed by the vessels. It also needs to consider the variations in the avoidance maneuvers due to the different driving styles and intentions of the vessels. For example, some sailors might be more conservative and COLREGs compliant, while

some might demonstrate risky, COLREGs-breaching behaviors. We have developed an intention motion model that estimates the future trajectories of the civilian vessels given their current and past states, dimensions, estimated goals, and the knowledge of static obstacle regions. The model consists of the following three components:

(i.) Classifier $c(b_l, O_l)$: Each civilian vessel b_l is classified as COLREGs compliant or COLREGs breaching given the observation history of its past states $O_l = \{\mathbf{x_{b_l}}(t), \mathbf{x_{b_l}}(t-1), ..., \mathbf{x_{b_l}}(t-\Delta t)\}$.

(ii.) Motion prediction model $m(\mathbf{x_U}, B, \{\mathbf{x_{b_l}}\}_{l=1}^{L}, \{O_l\}_{l=1}^{L}, \mathcal{O}_S)$: The inputs to the model are the current state $\mathbf{x_U}$ of the USV, the set of civilian vessels B together with their characteristics (i.e., their dimensions), the current and past states $\{\mathbf{x_{b_l}}\}_{l=1}^{L}$, $\{O_l\}_{l=1}^{L}$ of the civilian vessels, respectively, and the geometric region $\mathcal{O}_S$ occupied by static obstacles.

The output of the model are estimated future trajectories $\tau_{b_l} : [0, T_{b_l}] \to \mathcal{X}_{x,y}$ for each vessel $b_l \in B$. Here, $\mathcal{X}_{x,y} \subset \mathcal{X}$ represents the position subset of the entire state space $\mathcal{X}$, and T_{b_l} is the time horizon.

We assume that the prediction model has a priori knowledge of local goals of all the civilian vessels (e.g., by estimating their mission goals or through radio communication). In addition, the motion prediction model utilizes a classifier $c(b_l, O_l)$ to determine whether a vessel follows COLREGs based on the history of its states. The prediction model for each of the civilian vessel incorporates the reactive obstacle avoidance component that is used to avoid the USV as

well as all the other civilian vessels present in the environment.

(iii.) Distribution of position uncertainty of a civilian vessel along its predicted trajectory: We represent the uncertainty of a civilian vessel b_l deviating from its estimated trajectory $\tau_{b_l} : [0, T_{b_l}] \rightarrow \mathcal{X}_{x,y}$ using a bivariate normal distribution for different time slices t along the trajectory. The mean of the USV's positions along the trajectory is defined as $\mu_{b_l,t} \in \mathbb{R}^2$. The corresponding covariance matrix that captures the deviations of the vessel from this trajectory at time t is defined as $\sum_{b_l,t} \in \mathbb{R}^{2\times2}$. Initially, the variances of x and y coordinates increase with time as the civilian vessel deviates from its planned trajectory. Later, the variances remain the same or decrease as the civilian vessel approaches its intended goal location.

We performed Monte Carlo simulations to estimate $\sum_{b_l,t}$ for a scene with a given number of civilian vessels. We recorded deviation of the civilian vessel b_l from a trajectory τ_{b_l} along the x and y coordinates at discrete time intervals t. The trajectory was estimated using the motion prediction model. We then calculated the covariance matrix $\sum_{b_l,t}$ with respect to t. We carried out a polynomial regression to fit a two degree polynomial $\alpha_\Sigma(t) = \alpha_1 t^2 + \alpha_2 t + \alpha_3$ to the deviation for both of the coordinates. The actual covariance matrix for continuous time t is given by:

$$\sum_{b_l,t} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}^{-1} \begin{bmatrix} \alpha_{\Sigma,x}(t) & 0 \\ 0 & \alpha_{\Sigma,y}(t) \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \tag{3.2}$$

Here, c stands for $\cos(\psi_{b_l})$, s stands for $\sin(\psi_{b_l})$, and $\alpha_{\Sigma,x}(t)$ and $\alpha_{\Sigma,y}(t)$ are

variances for the x and y coordinates, respectively.

3.3.5 Search

The RCAP algorithm is based on the lattice-based A* heuristic search [33]. During the search, each state $\mathbf{s}$ is evaluated using the cost function described in Section 3.3.1. The cost function assigns a cost to each state based on the collision risk (see Section 3.3.2), COLREGs compliance (see Section 3.3.3), and availability of contingency maneuvers with respect to the vessels.

Unlike traditional approaches (see Section 2.1.1), the developed trajectory planner considers the reciprocal behaviors of civilian vessels in each USV's future expanded state $\mathbf{s}$. More formally, the planner employs the motion prediction model $m(\mathbf{s}, B, \{\mathbf{x_{b_l}}\}_{l=1}^{L}, \{O_l\}_{l=1}^{L}, \mathcal{O}_S)$ (see Section 3.3.4) to determine the desired action the other civilian vessels B will pursue with respect to the current discrete state of the USV and its control action $\mathbf{u_{c,d}} \in \mathcal{U}_{c,d}$. The future state of each civilian vessel $b_l \in B$ is determined by forward simulating it for the execution time of $\mathbf{u_{c,d}}$. The uncertainty in the state of the civilian vessels is determined by performing a Monte Carlo simulation as described in Section 3.3.4.

3.4 Adaptive Risk and Contingency-Aware Planning

The search for a risk and contingency-aware, dynamically feasible trajectory in a complex 5D state space is computationally expensive, which significantly reduces the replanning frequency. In this section, we introduce an adaptive risk and contingency-

aware planner (A-RCAP) that dynamically scales control action primitives to speed up the search and thus significantly reduce the computational requirements. The algorithm takes advantage of the fact that dense sampling is needed only in regions with a high number of civilian vessels, where it is necessary to utilize high-fidelity maneuvers to minimize the execution time of a trajectory and probability of collision. On the other hand, the planner saves considerable amount of computational effort by sparsely sampling the regions with low or no congestion. Thus, dynamic adaptation of the execution time of action primitives during the search substantially increases the efficiency of the planner, while sacrificing the quality of a computed trajectory.

3.4.1 Estimation of spatio-temporal workspace complexity

The developed approach is dependent upon the estimation of spatio-temporal complexity of sub-regions of the surrounding workspace of each expanded state $\mathbf{s}$ during the search. The spatio-temporal complexity is determined using the collision probability $p_{c,\mathbf{s}',U}$ (see Section 3.3.2) of each control action primitive $\mathbf{u_{c,d}} \in \mathcal{U}_{c,d}(\mathbf{s})$ emanating from $\mathbf{s}$. It also depends on the spatio-temporal complexity of past states along a trajectory leading to $\mathbf{s}$.

We divide the workspace around the current state $\mathbf{s}$ into the left, front, and right regions (see Figure 3.4). Let $\mathcal{U}_{c,d,l}(\mathbf{s})$, $\mathcal{U}_{c,d,f}(\mathbf{s})$, and $\mathcal{U}_{c,d,r}(\mathbf{s})$ be the subsets of the set of discrete control action primitives $\mathcal{U}_{c,d}(\mathbf{s})$ used by the USV. The spatio-temporal complexity for the left region is given by $\lambda_l = \max_{\mathbf{u_{c,d,i}} \in \mathcal{U}_{c,d,l}} p_{c,\mathbf{s}',U}$. Similarly, we compute the spatio-temporal complexity values λ_f and λ_r for the front and the right

workspace regions, respectively. The spatio-temporal complexity values are always between 0 and 1, where 0 signifies minimum complexity and 1 signifies maximum complexity. Naturally, the measure of the spatio-temporal complexity provides an estimation of the risk of collision when executing a control action primitive from the USV's state $\mathbf{s}$ in each region.

3.4.2 Adaptive sampling

The adaptive sampling of the state space is achieved by time scaling of individual control action primitives in their corresponding subsets $\mathcal{U}_{c,d,l}(\mathbf{s})$, $\mathcal{U}_{c,d,f}(\mathbf{s})$, and $\mathcal{U}_{c,d,r}(\mathbf{s})$. Let m_l, m_f, and $m_r \in \mathcal{M}_u$ be the multipliers used for scaling the control action primitives in the left, front, and right region, respectively. The multipliers are computed by Algorithm 2 using the estimated collision probabilities λ_l, λ_f, and λ_r. The multiplier values of states that are successors of $\mathbf{s}$ are computed using the multiplier values of that state.

The partitioning of the workspace into the three regions around $\mathbf{s}$ allows the algorithm to independently scale the primitives in their corresponding subsets. In other words, the spatio-temporal complexity of one region will not affect the time scaling of primitives in other regions. For instance, as the algorithm advances the search tree through the state space, the USV might encounter a heavy traffic on its left and almost zero traffic on its right in one of the states of the search tree. In such a situation, the algorithm increases the multiplier value m_r of the right region and reduces the multiplier value m_l of the left region. Thus, the control primitives

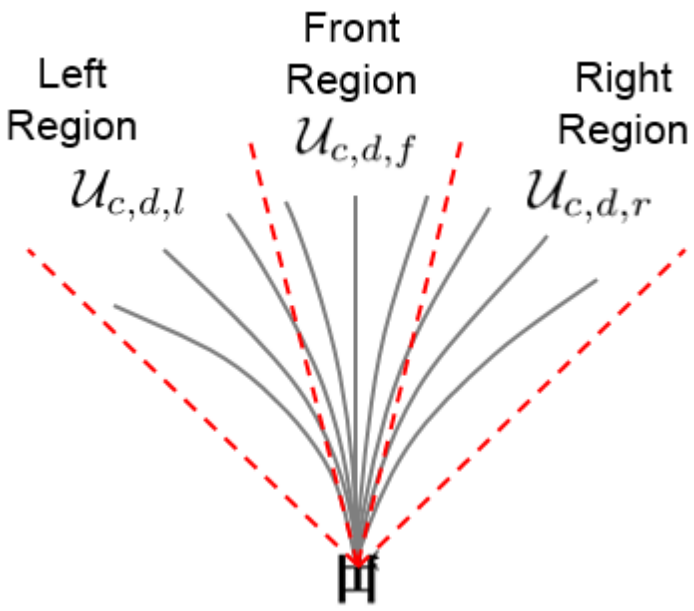


Figure 3.4: *A set of control action primitives divided into three sub-regions.*

in the right region will be expanded and those in the left region will be contracted.

In addition, there may be a situation during the search in which all of the possible action primitives expanded using the scaling multipliers m_l, m_f, and $m_r \in \mathcal{M}_{\mathbf{u}}$ lead to a collision from a given state $\mathbf{s}$ (i.e., λ_l, λ_f, and λ_l are close to 1). This may occur mostly when transitioning from states in regions with a low spatio-temporal complexity to states in regions with a high spatio-temporal complexity. In this case, the algorithm reconsiders the same state $\mathbf{s}$ (that is already in the closed set) by adding it to the priority queue (i.e., to the open set) and reduces all the scaling multipliers m_l, m_f, and m_r by half. This state is then reevaluated using the reduced control action primitives in each region. If the value of all the spatio-temporal complexity parameters (λ_l, λ_f, and λ_r) is again close to 1, then the algorithm backtracks to the default, minimum value of the multiplier values, i.e.,

$m_l = m_f = m_r = 1$, and the state is reinserted into the priority queue. In Algorithm 1 and 2, the boolean variable that triggers the reinsertion of the state into the queue is labeled as r_r, r_f and $r_l \in \mathcal{R}$ for right, front and left region respectively.

The discrete state transition function used to determine the neighboring states during the search is given by $f_{U,d}(\mathbf{s}, \mathbf{u_{c,d}}, m_x) = [x + l_u(m_x - 1)cos(\psi_d), y + l_u(m_x - 1)sin(\psi_d), t + (l_u m_x)/u_d]^T$, where m_x can be substituted by m_l, m_f or m_r. The input of the state transition function are the current state $\mathbf{s}$, the default control action primitive $\mathbf{u_{c,d}}$, and one of the scaling multipliers m_l, m_f, or m_r depending on the subset $\mathcal{U}_{\mathbf{c,d,l}}$, $\mathcal{U}_{\mathbf{c,d,f}}$, or $\mathcal{U}_{\mathbf{c,d,r}}$ the control action $\mathbf{u_{c,d}}$ belongs to. After the execution of $\mathbf{u_{c,d}}$ for the time t, the terminal position and orientation of the USV is given by $[x, y]^T$ and ψ_d, respectively, and the length of the entire executed trajectory from the initial state of the USV is given by l_u. At all times, the scaling multiplier $m_u \geq 1$. This guarantees the expanded control action primitives $\mathcal{U}'_{c,d}(\mathbf{s})$ to be dynamically feasible and executable by the USV's low level controller (e.g., PID, backstepping, etc.).

Figure 3.5 shows a risk and contingency-aware trajectory computed using adaptive control action primitives. The scenario consists of the USV and 3 civilian vessels moving towards their motion goals. The intention model described in 3.3.4 is used to predict the reciprocal trajectories of all the vessels with respect to the USV's trajectory. The uncertainty in a vessel's position is represented as a sequence of Gaussian distributions (shown as a multi-color cloud around the planned trajectory) and increases with time. The trajectory consists of control action primitives of varying magnitude and thus execution times. In this figure, the magnitude of

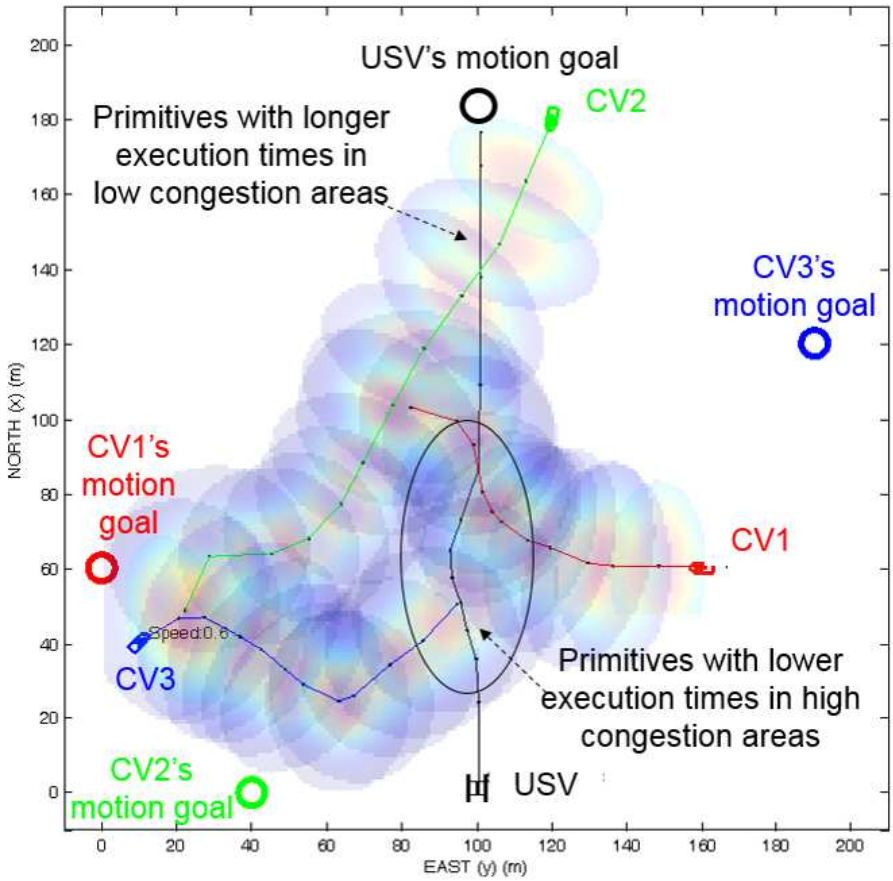


Figure 3.5: *An example of a computed risk and contingency-aware, dynamically feasible trajectory using adaptive control action primitives in a scenario with 3 civilian vessels*

control action primitives near the initial location of the USV is high and gradually decreases as the search progresses towards areas with high congestion (from 40 m to 80 m north along the x axis). The magnitude of the primitives starts to increase again at the distance of 80 m along the x axis until the search reaches the motion goal of the USV.

Algorithm 1 COMPUTETRAJECTORY($\mathbf{s}_I, S_G, \mathcal{U}_{c,d}$)

Input: USV's initial state $\mathbf{s_I}$, desired goal region S_G, and a default, dynamically feasible control action primitive set $\mathcal{U}_{c,d}$.

Output: A trajectory τ.

Let $S_O \leftarrow \{\mathbf{s_I}\}$ be a priority queue of states sorted in ascending order according to the cost function f (see (3.3.1)).
Let $S_C \leftarrow \emptyset$ be the set of all expanded/closed states.
Let $\mathcal{M} = \{m_r, m_f, m_l\} \leftarrow 1$ be the set containing scaling multipliers for the left, front, and right workspace regions (see Figure 3.4).
Let $\mathcal{R} = \{r_r, r_f, r_l\} \leftarrow false$ be the sets containing a re-evaluation indicator for the left, front, and right workspace regions.
Let f_{new} be a boolean function which is used to distinguish newly opened states from states undergoing re-evaluation.
while S_O not empty **do**
    $S_C \leftarrow \mathbf{s} \leftarrow S_O$.FIRST()
    if $\mathbf{s} \in S_G$ **then**
        return A trajectory τ generated by recursively tracing the predecessors of $\mathbf{s}$ up to $\mathbf{s}_I$.
    end if
    $f_{new}(\mathbf{s}) = \neg r_l \wedge \neg r_f \wedge \neg r_r$
    for all $\mathcal{U}_{c,d,x} \in \{\mathcal{U}_{c,d,l}, \mathcal{U}_{c,d,f}, \mathcal{U}_{c,d,r}\} \subset \mathcal{U}_{c,d}$ **do**
        Let $r_x \in \mathcal{R}$ be a re-evaluation indicator corresponding to region x.
        if $r_x \vee f_{new}(\mathbf{s})$ **then**
            for all $\mathbf{u_{c,d}} \in \mathcal{U}_{c,d,x}$ **do**
                Let $m_x \in \mathcal{M}$ be a scaling multiplier corresponding to region x.
                $\mathbf{s}' \leftarrow f_{U,d}(\mathbf{s}, \mathbf{u_{c,d}}, m_x)$ (see Section 3.4.2)
                if $\mathbf{s}' \notin S_C$ **then**
                    Estimate current states $\{\mathbf{x_{b_l}}(t')\}_{l=1}^{L}$ of all civilian vessels at time t' by forward simulating their intention models $\{m_l\}_{l=1}^{|B|}$.
                    Compute $p^n_{c,s'}$ and $p^e_{c,s'}$ (see Section 3.3.1) for the USV moving between $\mathbf{s}$ and $\mathbf{s}'$.
                    if $(\mathbf{s}' \notin S_O) \vee (\mathbf{s}' \in S_O \wedge (f_{new}(\mathbf{s}') < f_{old}(\mathbf{s}')))$ **then**
                        Set $\mathbf{s}'$ as the best successor state of $\mathbf{s}$.
                        Insert/update $\mathbf{s}'$ into/in S_O.
                    end if
                end if
            end for
            $\{m_x, r_x\} \leftarrow$ COMPUTESCALINGFACTOR($\mathcal{U}_{c,d,x}, m_x, r_x$)
        end if
    end for
    if $r_l \vee r_f \vee r_r$ **then**
        Remove $\mathbf{s}$ from S_C and insert it into S_O.
    end if
end while
return $\tau = \emptyset$ (no suitable trajectory has been found).

Algorithm 2 ComputeScalingFactor($\mathcal{U}_{c,d,x}, m_x, r_x$)

Input: A set $\mathcal{U}_{c,d,x}$ is a set of control action primitive in region x, the control action primitive scaling multiplier m_x, and the re-evaluation boolean indicator for region x. The region x can be either left, front, or right regions (see Section 3.4.1).

Output: A new control action primitive scaling factor m'_x and state re-evaluation boolean indicator r'_x for the current region x.

1: Let $\lambda_1, \lambda_2, \lambda_3$, and $\lambda_4 \in [0, 1]$ be the user-defined, ascending spatio-temporal complexity levels, i.e., $0 < \lambda_1 < \lambda_2 < \lambda_3 < \lambda_4 < 1$.

2: Let $\lambda_{stc} = \max_{\mathbf{u_c},\mathbf{d},\mathbf{i} \in \mathcal{U}_{c,d,x}} p_{c,\mathbf{s}',U}$ be the measure of the spatio-temporal complexity for the region x.

3: Let $\delta_e > 1$ be the exponential increase of the multiplier m_x.

4: Let δ_m be the linear increment/decrement of the multiplier m_x.

5:

$$m'_x \leftarrow \begin{cases} \delta_e m_x & \text{if} \quad \lambda_{stc} < \lambda_1 \\ m_x + \delta m & \text{if} \quad \lambda_1 < \lambda_{stc} < \lambda_2 \\ m_x & \text{if} \quad \lambda_2 < \lambda_{stc} < \lambda_3 \\ m_x - \delta m & \text{if} \quad \lambda_3 < \lambda_{stc} < \lambda_4 \\ m_x/2 & \text{if} \quad \lambda_4 < \lambda_{stc} \text{ and } r_x \text{ is false} \\ 1 & \text{otherwise} \end{cases} \tag{3.3}$$

6: $m'_x \leftarrow min(m'_x, m_{x,max})$, where $m_{x,max}$ is the user-specified scaling factor threshold.

7: $m'_x \leftarrow max(m'_x, 1)$.

8: $r'_x \leftarrow \lambda_{stc} > \lambda_4$.

9: **return** $\{m'_x, r'_x\}$

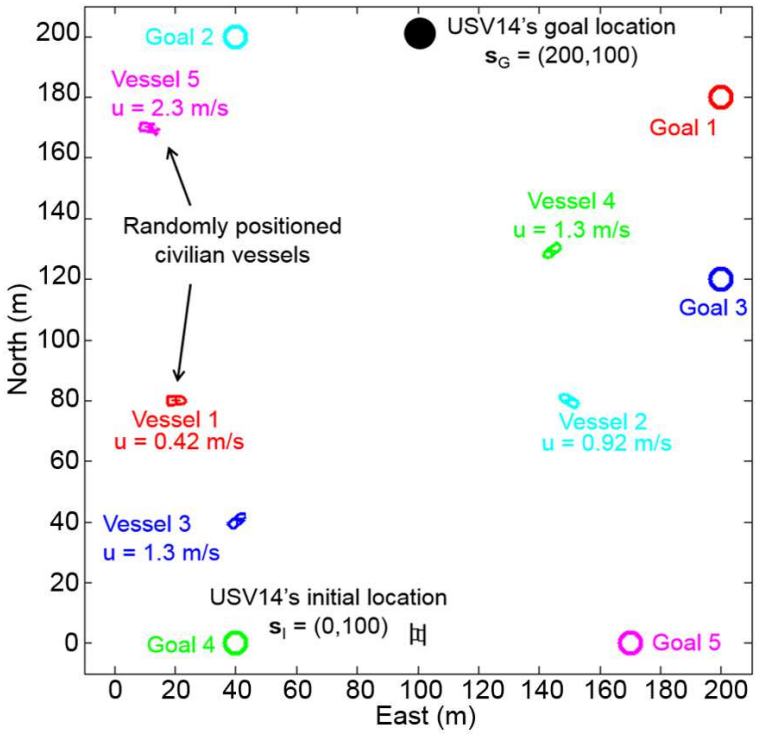


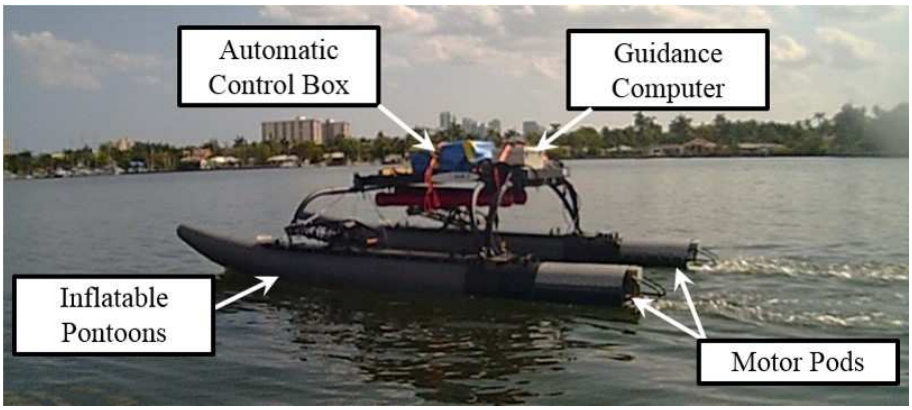
Figure 3.6: *Simulation Scenario.*

3.5 Computational Experiments

3.5.1 Simulation setup

The developed planning algorithms were evaluated using a simulation setup (see Figure 4.22) consisting of an experimental area of 200m X 200m, the wave amplitude modular USV (labeled as the USV14), and civilian vessels (see Figure 3.7). The motion of the USV14 was simulated using a system-identified 3 DOF dynamic model [1]. Each civilian vessel was modeled using a simple car kinematic model with the Ackermann steering geometry [42].

The set of default as well as contingency control action primitives of the USV14 were designed using its dynamics model. The default control primitive set $\mathcal{U}_{c,d}$ (see



(a)

(b)

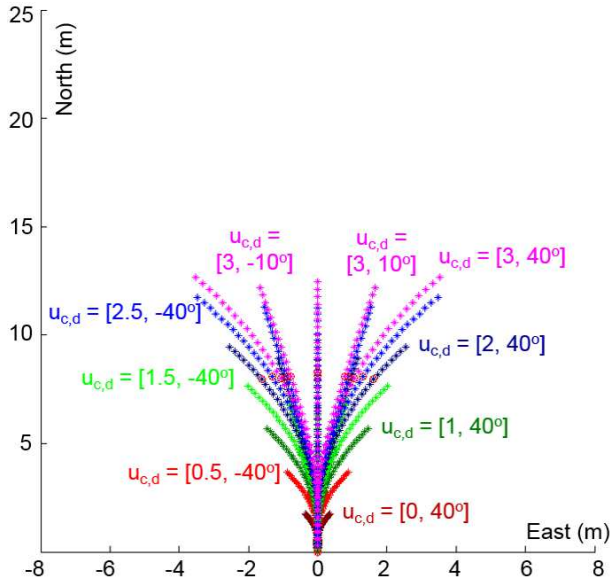
Figure 3.7: *(a) The autonomous USV14; (b) The human-controlled johnboat [1].*

Figure 3.8(a)) consisted of $\psi_d = 0^o, \pm 10^o, \pm 40^o$ and u_d ranging from 0 to 3 m/s in steps of 0.5 m/s. This resulted in the total of 30 default control action primitives consisting of 5 levels of ψ_d and 6 levels of u_d.

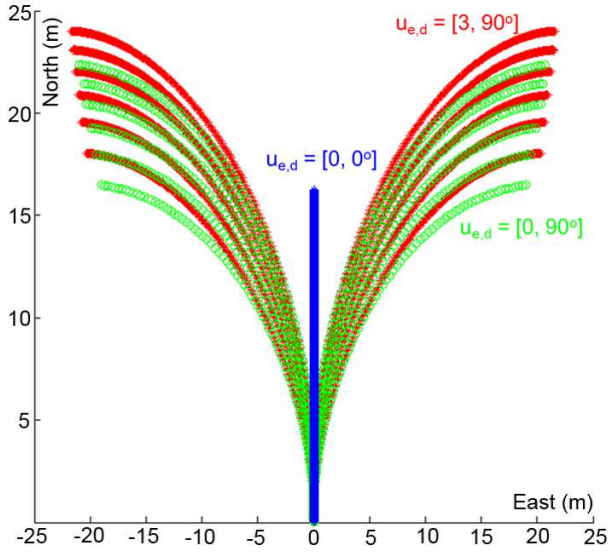
The control action primitives were precomputed by placing the USV14 at the initial state and applying control inputs through a PD-controller until the vehicle achieved the desired heading ψ_d and surge speed u_d. Each resulting trajectory was recorded as a finite sequence of $[x, y, t]^T$ positions, speeds, and arrival times of the USV14. We have discretized the heading of the USV14 into 5 levels and continuous surge speed into 6 levels, which results in 30 control actions in the control primitive action set. During the search, the transitions between states were only allowed to adjacent levels of speed.

The contingency actions (see Figure 3.8(b)) of the vehicle include $\mathbf{u_{e,d,1}} = [0, 0]^T$ (i.e., maintain the same heading with the minimum desired speed), $\mathbf{u_{e,d,2}} = [3, -90]^T$ (i.e., steer left with the maximum desired speed), $\mathbf{u_{e,d,3}} = [0, -90]^T$ (i.e., steer left with the minimum desired speed), $\mathbf{u_{e,d,4}} = [3, 90]^T$ (i.e., steer right with the maximum desired speed), and $\mathbf{u_{e,d,5}} = [0, 90]^T$ (i.e., steer left with the minimum desired speed). These maneuvers were designed by considering the most common reaction exhibited by humans on a collision course. The USV14 always selects a contingency maneuver with the maximum d_{CPA} (see Section 3.3.3) with respect to all civilian vessels to maximize safety.

For all experiments, the values of constants required by the cost function for trajectory planning (see Section 3.3.1) are $d_{max} = 200$ (i.e., twice the length of a straight path between $\mathbf{s_{U,I}}$ and $\mathbf{s_{U,G}}$), $t_{max} = d_{max}/1.5$ (i.e., the average speed of



(a)



(b)

Figure 3.8: *(a) A dynamically feasible control action set $\mathcal{U}_{c,d}$, and (b) a dynamically feasible contingency control action set $\mathcal{U}_{e,d}$ for the USV14 with different initial surge speeds.*

1.5 m/s), $\omega_n = 1000$, $\omega_c = 0.5$, $c_{\neg COLREGs} = 1000$, $c_e = 500$, $c_{e,c} = 10000$, $\gamma = 0.1$, $\omega_{c,U,B} = 0.3$, and $\epsilon = 4$. The threshold values for CPA parameters used to evaluate USV's states for COLREGs compliance are $d_{CPA,min} = 50$ m and $t_{CPA,max} = 30$ s.

The obstacle avoidance behavior of civilian vessels is realized through the Velocity Obstacle (VO) [39] based local obstacle avoidance planner. The planner allows us to forward simulate the motion of a civilian vessel with increased realism. The parameters of the planner were tuned during physical tests [1], which involved COLREGs compliant obstacle avoidance by civilian vessels [155] (see Figure 3.10).

During the search for a global trajectory, we place the USV14 at each expanded state and forward simulate the behavior of each vessel using the VO planner. The planner computes the best velocity vector for each civilian vessel given its pose, speed, and motion goal while ignoring its dynamics. This allows us to capture a reciprocal trajectory computed by the civilian vessels in response to the USV14's trajectory.

The time for which the civilian vessels are forward simulated is equal to the execution time of the USV14's control action primitive. The computed velocity vector of each civilian vessel is integrated throughout the forward simulation time of control action to determine its future state. We assume that all civilian vessels maintain their velocity vectors for the entire forward simulation time of each control action. This increases the uncertainty in the actual states of the vessels while executing their planned trajectories. This uncertainty increases with the simulation time as civilian vessels tend to wander away more from their trajectories.

We have modeled the uncertainty in the positions of a civilian vessel using a

sequence of 2D Gaussian distributions with the means corresponding to the way-points of the planned trajectory. The variances of the distributions are estimated by performing a Monte Carlo simulation (see Section 3.3.4) for each scenario. We have calculated the variances for discrete time slices of 5 to 100 s and interpolated them by fitting a 2 degree polynomial with the forward simulation time as a variable.

3.5.2 Modeling scenario congestion

In order to evaluate the performance of the planner, we have designed a metric for measuring congestion of a scenario with respect to a given position of the USV (see Figure 3.9). A scenario can be classified as congested, for example, if there are only a few large civilian vessels with a large turning radius and low speed. Alternatively, a scenario is considered to be congested if there are many small highly maneuverable vessels moving at high speeds within the same region. In addition, congestion of a scene also varies depending on how far the vessels are from the current position of the USV. For example, the vessels that are far away are considered to contribute less to the congestion of the scenario as compared to the vessels that are close to the USV.

The developed metric Λ_{cgn} considers the size, velocity, and the distance d_l of each civilian vessel b_l from the USV within a circular area A_r with the radius r around the USV. The characteristics of a vessel b_l such as its size and velocity are used to compute a region of inevitable collision RIC_l [156] with respect to the USV and the vessel. The USV is inside RIC_l if there is no control action $\mathbf{u_{c,d}} \in \mathcal{U}_{c,d}(\mathbf{x}_U)$

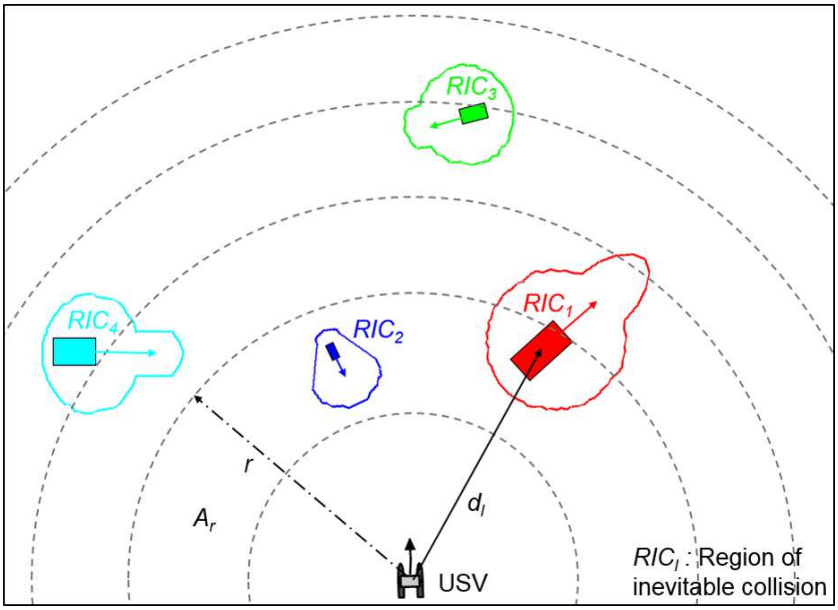


Figure 3.9: *Computation of the congestion metric for an example scenario.*

that would prevent it from colliding with b_l. We assume the USV to move at its maximum speed when computing RIC_l. We define $RIC_r = \cup_{l=1,d_l<r}^{L} RIC_l$ as the union of all the inevitable collision regions of civilian vessels up to the given distance threshold r. The congestion of the whole scenario is calculated as $\Lambda_{cgn} = \int_{r=0}^{\infty} RIC_r/A_r \, \mathrm{d}r$.

For scenarios with civilian vessels in the vicinity to the USV, the degree of congestion Λ_{cgn} gradually increases with the increase of the radius r as more vessels fall into A_r. Beyond a certain radius, Λ_{cgn} gradually decreases as the total area A_r becomes more significant compared to RIC_r.

The regions of inevitable collision RIC_l were precomputed for the vessels with the maximum surge speed between 1 and 5 m/s in increments of 0.1 m/s and the length between 4 and 30 m in increments of 1 m. During the computation of Λ_{cgn}, we increased the radius r of the area A_r starting from 5 to 200 m (max. size of the

workspace) in increments of 5 m. Note that scenarios with different types of civilian vessels and their distribution in the area can lead to the same congestion value. For example, the congestion value of 7 corresponds to a scenario with 4 civilian vessels with the length of 10 m and the maximum surge speed of 4 m/s. This congestion value also corresponds to a scenario with 8 civilian vessels with the length of 5 m and the maximum surge speed of 1 m/s.

3.5.3 Design of evaluation scenarios

We have designed 10 groups of evaluation scenarios with a similar congestion value ranging from 0 to 50. For each congestion group, we randomly generated 1000 evaluation scenarios. In each evaluation scenario, the USV14 was positioned at the same initial state $\mathbf{s_{U,I}} = [0, 100, 0, 0, 0]^T$ and commanded to reach its stationary goal state $\mathbf{s_{U,G}} = [200\ , 100]^T$ (we neglected the heading and surge speed state variables in the goal state). The simulation runs were performed on Intel(R) Core(TM) i7-2600 CPU @ 3.4 GHz machine with 8GB RAM.

The number of vessels was generated uniformly at random ranging from 3 to 8. The poses of the civilian vessels were randomly generated with no vessels positioned inside a circle of 20 m radius around the USV14 (i.e., corresponding to the minimum turning radius of the USV14) to avoid an imminent collision at the beginning of an evaluation run. The length of each vessel was randomly generated to be between 4 and 30 m, and its maximum surge speed was determined to be between 1 and 5 m/s.

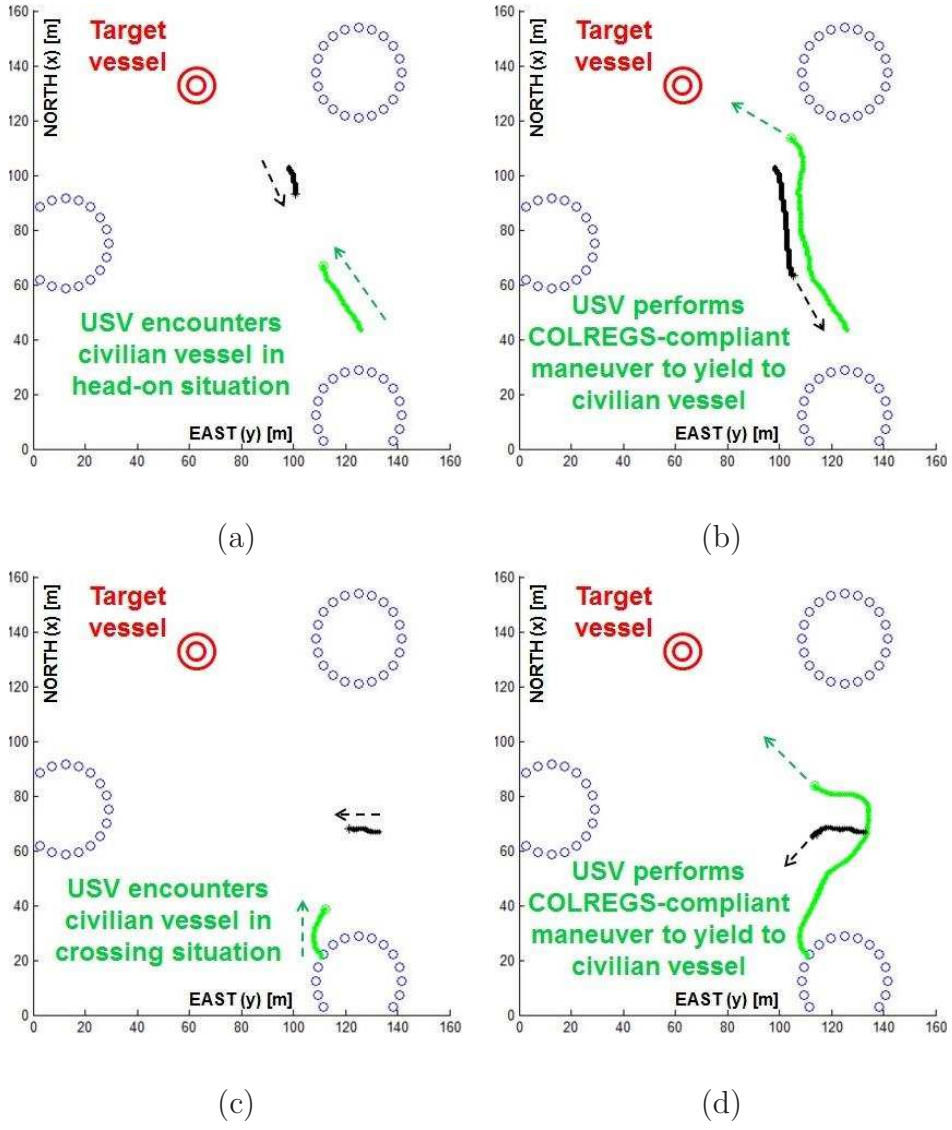
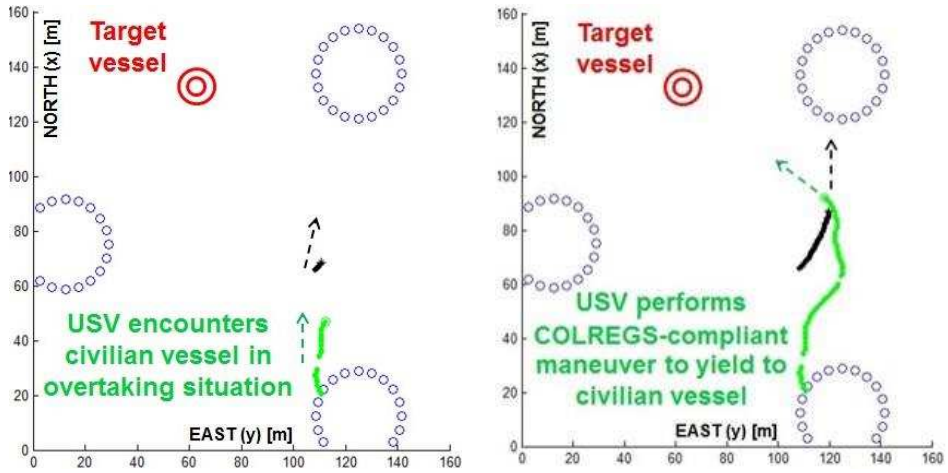


Figure 3.10: *The experimental results of the USV14 autonomously dealing with "head-on" (see a) and b)), "crossing from right" (see c) and d)), and "overtaking" (see e) and f)) situations*



(e) (f)

Figure 3.10: *The experimental results of the USV14 autonomously dealing with "head-on" (see a) and b)), "crossing from right" (see c) and d)), and "overtaking" (see e) and f)) situations*

3.5.4 Results

We have carried out simulation experiments to evaluate the collision risk, length, and execution time of trajectories computed by the RCAP and A-RCAP algorithms. The planners were evaluated using multiple scenarios with different congestion levels.

The RCAP algorithm is computationally expensive, which leads to a low re-planning frequency. To be able to avoid highly dynamic obstacles, we use a VO-based local planner [1] to track a global trajectory computed by the RCAP algorithm. In the simulation results presented below, the global risk and contingency-aware trajectory is computed once by the RCAP at the beginning of a simulation run. The trajectory is then tracked by the VO-based local planner [1] until the USV reaches

its goal.

As can be seen in the plot in Figure 3.11, the use of the RCAP algorithm combined with the VO-base local planner results in significantly fewer number of collisions as compared to the state-of-the-art VO-based local planner [86] used in marine missions. The collision rate represents the percentage reduction in the number of collisions per 1000 boat lengths traveled distance by the USV14. The percentage reduction in the number of collisions by the A-RCAP is approximately the same to its non-adaptive predecessor. This signifies that the gain in the computational efficiency does not degrade the quality of risk-aware trajectories.

There are several reasons for the reduction in the collision rate. First, the planners consider differential constraints of the USV when computing a trajectory. In case of RCAP, this allows the VO-based local planner to successfully track the trajectory. Second, the planners minimize the overall collision risk by breaching COLREGs with respect to selected vessels and choose a lower risk workspace over a high risk, dense workspace. A local VO-based planner with a limited number of look-ahead steps cannot reason about the collision risk over a larger portion of the workspace. Thus, breaching COLREGs locally may prove disastrous in later stages. Finally, by incorporating contingency maneuvers into a global trajectory ensures that the USV has at least one contingency maneuver available to avoid an imminent collision due to unpredictable behaviors of civilian vessels.

The results presented in Figure 3.12 and Figure 3.13 describe the percentage of additional distance and time traveled by the USV using the VO-based planner, the RCAP, and the A-RCAP over the distance and time computed for a scene with

zero congestion. The distance traveled in a scene with zero congestion is equivalent to the Euclidean distance from the initial to the goal state. The travel time in this zero congestion scenario is equivalent to the time required by the USV14 to travel from the initial to the goal state at the maximum surge speed of 3 m/s. The additional travel time (see Figure 3.13) is greater than the additional traveled distance (see Figure 3.12) for all the planners. This is due to the fact that the USV yields to several civilian vessels by slowing down and not performing large avoidance maneuvers, which decreases the distance traveled.

It can be seen from the plots in Figs. 3.12 and 3.13 that the additional traveled distance and time significantly increase for the RCAP with the increase in congestion as compared to the VO-based and A-RCAP planners. Due to the lack of replanning capability of global, risk-aware trajectories by the RCAP, the local VO-based planner used in conjunction with the RCAP has to perform large avoidance maneuvers to handle the dynamics of the environment while simultaneously tracking a global trajectory. In other words, the USV14 has to return to the global trajectory after performing a local avoidance maneuver, which results in larger travel distance and time. On the other hand, the replanning rate of the local VO-based planner is high. This allows to deal with the highly dynamic nature of the environment and to find shorter and faster paths as compared to RCAP, but at the cost of higher collision rate (see Figure3.11).

The results presented in Table 3.1 show approximately 500% improvement in the computational performance of the A-RCAP as compared to its non-adaptive predecessor RCAP. The performance is given in terms of the number of states expanded

during the search for a trajectory. The computational speed-up allowed the USV to replan its trajectory with the average frequency of 0.1 Hz per 7.5 boat lengths of traveled distance. This frequent replanning of global risk and contingency-aware trajectories results in minimizing the USV's total travel distance and time to reach the goal. Hence, A-RCAP shows approximately 20% reduction in the travel distance and 25% reduction in the travel time as compared to RCAP. The replanning frequency of the A-RCAP is smaller as compared to the replanning frequency of the VO-based planner. However, the local VO-based planner does not minimize the global objective of the total traversal time and distance. Also, the VO-based planner does not have multi-step look-ahead capability to predict the motion of other civilian vessels, which results in longer travel time and distance.

3.6 Tuning Planner Performance

3.6.1 Planning parameter tuning

(i.) User preference parameters: These parameters alter trajectories generated by the planner according to the preference of the user. The parameters that fall under this category are:

- ω_c: The parameter $\omega_c \in [0, 1]$ is used to balance the total travel time and distance of the computed trajectory. The value of $\omega_c = 0$ will provide trajectories with short travel distance and long travel time (i.e. the USV will travel less distance at slower speed) and vice versa with $\omega_c = 1$.

Table 3.1: *Average number of states expanded by the RCAP and A-RCAP algorithms.*

Congestion Value (Λ_{cgn})	RCAP		A-RCAP	
	Mean	Std dev.	Mean	Std dev.
0-5	940.54	55.41	188.92	12.65
5-10	1006.78	62.58	195.7	13.86
10-15	1055.48	68.14	203.21	16.9
15-20	1155.82	75.37	215.87	16.8
20-25	1213.86	84.24	220.55	17.01
25-30	1266.29	91.43	221.24	17.56
30-35	1320.78	90.89	229.61	18.08
35-40	1388.35	98.54	237.23	18.73
40-45	1480.22	110.19	249.1	18.6
45-50	1575.18	118.72	257.96	18.9

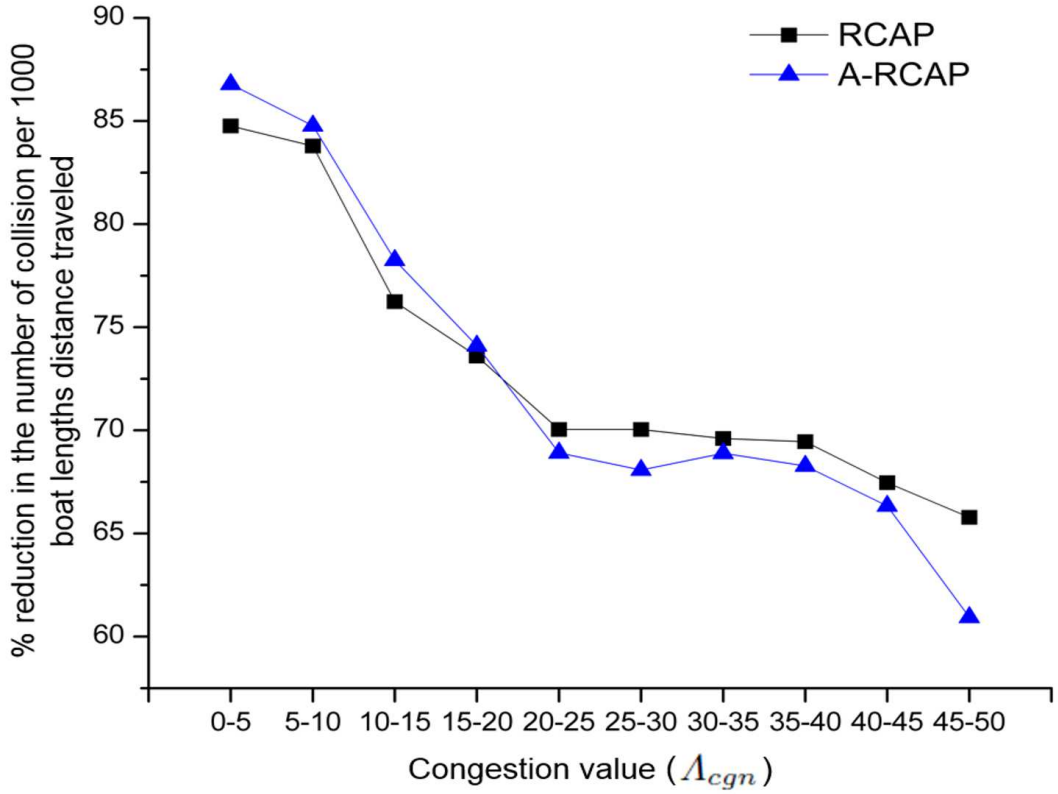


Figure 3.11: *The percentage reduction in the number of collisions recorded per 1000 boat lengths traveled by the USV using the RCAP and A-RCAP algorithms over the VO-based planner*

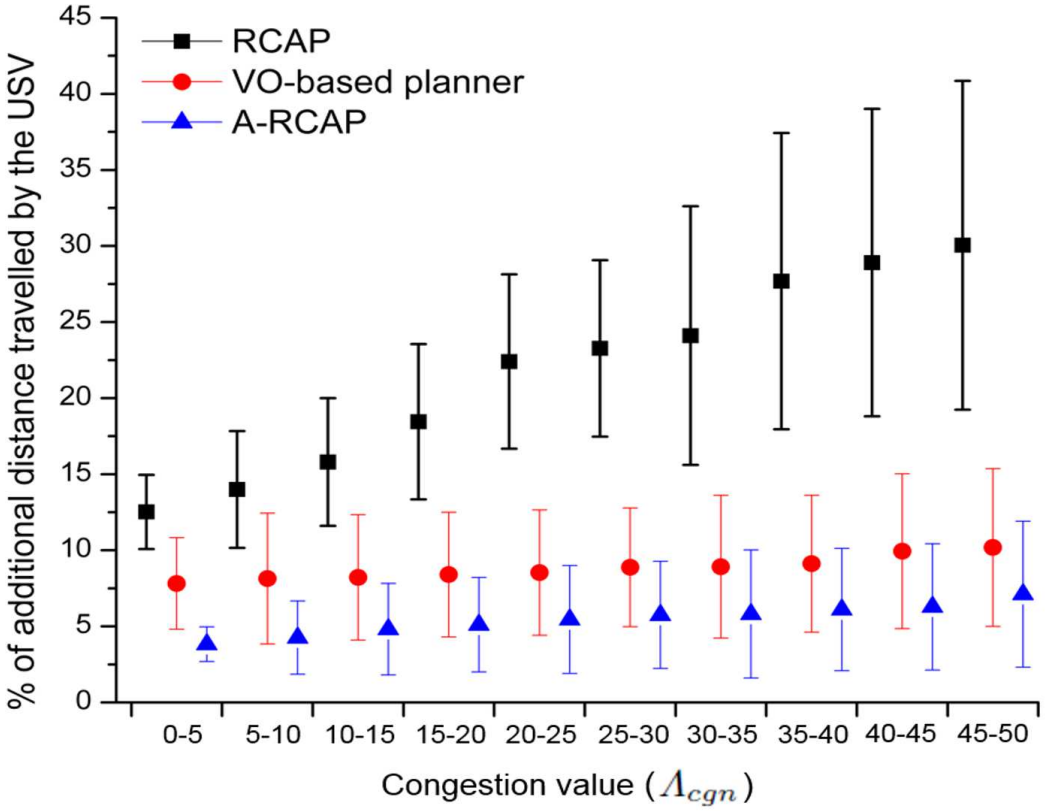


Figure 3.12: *Percentage of the additional distance traveled by the USV over the zero congestion travel distance for the RCAP, VO-based, and A-RCAP planners.*

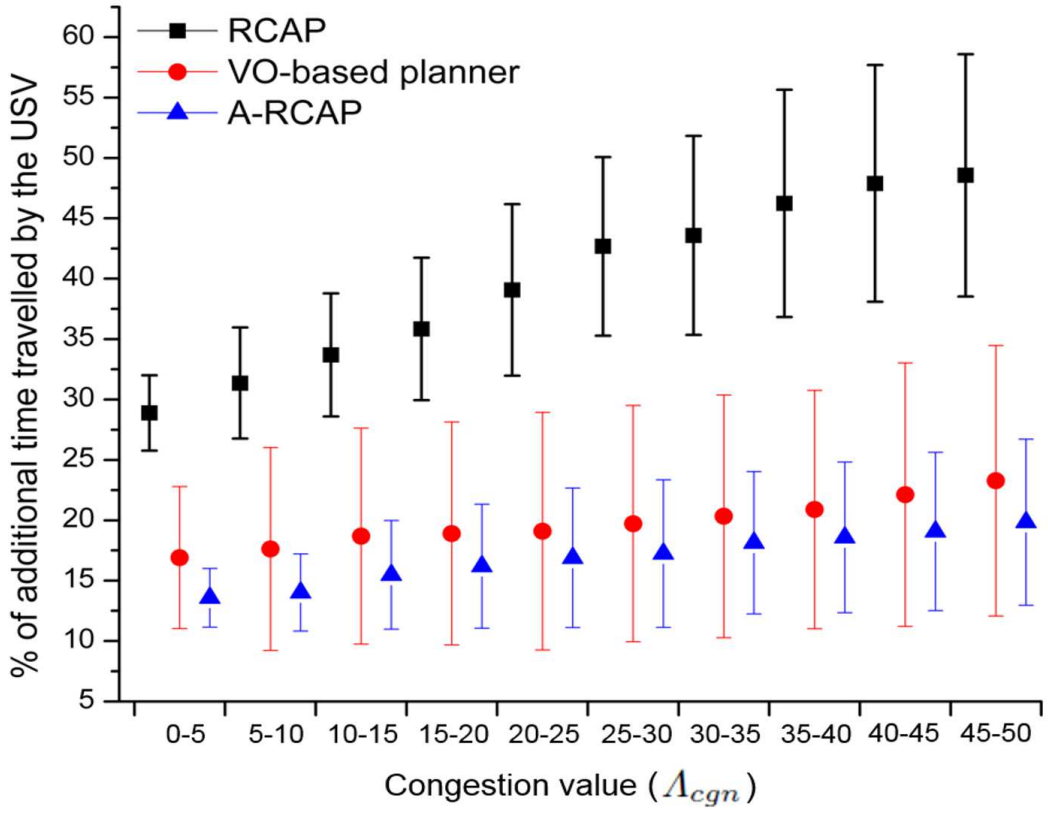


Figure 3.13: *Percentage of the additional time traveled by the USV over the zero congestion travel time for the RCAP, VO-based, and A-RCAP planners*

Thus, ω_c helps the user to capture the USV's mission objective in terms of total travel time and distance.

– ω_n: The parameter $\omega_n \in [0, 1]$ balances the USV's mission objective (i.e. total travel time and distance) against its COLREGs-compliance. The planner will provide the safest COLREGs-compliant trajectory with no constraint on the travel time and distance at lower values of ω_n. At $\omega_n = 0$, the planner may provide a solution that will command the USV to remain at its current position until the civilian traffic vehicles are cleared from the environment. However, at $\omega_n = 1$ the planner will compute trajectories that are optimal in travel time and distance (depending on ω_c) and disregard the COLREGs rules. Thus, the user need to tune this parameter depending upon the assigned mission to find a right balance between the COLREGs-compliance and the mission objectives.

– $\omega_{c,U,B}$: The parameter $\omega_{c,U,B} \in [0, 1]$ balances the collisions between the USV and the civilian vessels against the collisions among the civilian vessels themselves. At the values below 0.5, the planner will prioritize the collisions of the USV over the other civilian vessels and vice versa for the values above 0.5.

– t_{max} and d_{max}: These parameters are dependent on the mission objectives of the USV, where t_{max} specifies the maximum allotted time and d_{max} specifies the maximum allocated distance that the USV can travel in the currently assigned mission.

Initially, the user can set the default parameter values and evaluate the trajectory produced by the planner on three fronts: travel time, travel distance, and risk of breaching COLREGs. Then, the user can tweak these parameters (ω_c, ω_n, $\omega_{c,U,B}$) to improve the quality of the trajectory on each front until the computed trajectory meets the desired quality.

(ii.) COLREGs parameters: These parameters are used to perform COLREGs-based obstacle avoidance and are usually tuned by the experts with the knowledge of the physical capabilities of the vessel. Parameters that fall in this category are divided into two subgroups:

- $d_{CPA,min}$ and $t_{CPA,max}$: These parameters are distance and time to closest point of approach and are used for collision detection. The USV will perform large and conservative avoidance maneuvers with high values of $d_{CPA,min}$ and $t_{CPA,max}$, and with lower values of $d_{CPA,min}$ and $t_{CPA,max}$ the USV will perform short and risky avoidance maneuvers. Thus, it requires an expert to tune these parameters in order to achieve optimum avoidance maneuvers.
- Cost parameters: The execution cost of the emergency control action (c_e), the collision cost of emergency control action ($c_{e,c}$), and the penalty for breaching COLREGs ($c_{\neg COLREGs}$). These parameters are used to model risk of the computed trajectory.

The COLREGs parameters have to be tuned in two phases. Firstly, the default values of the planner are selected by interviewing the experts about the

standard practices and avoidance strategies given the specification of the USV and the type of the environment. The user can use these default values in the planner and generate sample trajectories. Secondly, the generated sample trajectories are critiqued by the expert in a Learning from Critique (LfC) framework. Depending upon the information provided by the expert the user can tweak the parameter set and generate a new trajectory. This procedure is iteratively performed until the planner provides trajectories that meet the satisfactory limits of the expert.

(iii.) Planning speed parameters: These parameters are us- ed to improve the computation performance of the planners and are primarily used by A-RCAP. Most of unmanned systems (including USVs) have finite on-board computational resources. The planner take large/short computational time to produce optimal trajectories which may not be viable in a dynamic and rapidly changing environment having high congestion. On the other hand, the trajectories computed by the planner in short amount of computational time will be highly sub-optimal. Thus, given the on-board computational resources and the congestion of the environment the user has to optimize and find the right set of parameters that will enable the planner to compute trajectories in the required computational time. The parameters falling under this category are the spatio-temporal complexity levels (λ_1,λ_2, λ_3 , and λ_4) and the exponential (δ_e) and linear (δ_l) scaling factors that are used for adaptively scaling the control primitives in A-RCAP.

Ideally, these parameters should be optimized by performing combinatorial search over the entire parameter set [λ_1, λ_2, λ_3, λ_4, δ_e, δ_l]. We decided to perform partitioned optimization to reduce computational overhead. For partitioned optimization, the parameter set has been divided into spatio-temporal complexity parameters [λ_1, λ_2, λ_3, λ_4], and scaling factors [δ_e, δ_l].

The values of spatio-temporal complexity parameters λ_1, λ_2, λ_3, and λ_4 significantly influence the collision rate, and thus the overall performance of the A-RCAP algorithm. The parameters are optimized by performing combinatorial search over 15 different combinations of [λ_1, λ_2, λ_3, λ_4] to achieve the least collision rate. The parameters in each combination follow a strict precedence order $0 < \lambda_1 < \lambda_2 < \lambda_3 < \lambda_4 < 1$. Each combination was evaluated using 1000 randomly generated test cases for three congestion groups with congestion values $\Lambda_{cgn} = 10 - 15$, $25 - 30$, and $40 - 45$. For optimization of the spatio-temporal complexity parameters, we randomly sampled the values of the exponential (δ_e) and linear (δ_l) scaling factors. Finally, the parameters set $\Lambda^*_{stc} = \{0.25, 0.4, 0.55, 0.7\}$ resulted in the minimum collision rate for all the three congestion groups.

Secondly, the values for scaling factors of control action primitives (δ_e and δ_l) influences the computational performance, the total travel time, and the total travel distance by the USV. The parameters are optimized by performing combinatorial search over the range of discrete values of δ_e and δ_l. The table presented in Table 3.2 is the number of states expanded with respect to δ_e

Table 3.2: *Mean number of states expanded by varying the exponential and linear increment parameters for scenario with different congestion value.*

Exponential Increment δ_e	Linear Increment δ_l	Mean Number of States Expanded		
		Congestion 10 - 15	Congestion 25 - 30	Congestion 40 - 45
1.4	0.1	318.36	235.84	365.15
	0.4	224.83	206.76	328.63
	0.7	218.3	198.23	284.2
	1	212.83	196.89	243.11
1.6	0.1	241.02	229.95	269.8
	0.4	214.87	196.73	260.42
	0.7	213.96	198.28	233.88
	1	204.51	192.52	241.34
1.8	0.1	216.62	221.6	261.12
	0.4	215.98	226.44	230.48
	0.7	188.32	203.78	243.72
	1	176.88	188.78	218.78
2	0.1	152.27	196.45	262.46
	0.4	142.34	184.24	181.84
	0.7	146.86	173.63	176.15
	1	144.73	165.69	175.41

and δ_l at constant optimum value of the spatio-temporal complexity levels (Λ^*_{stc}). We can observe from the table that at higher values of δ_e and δ_l, the number of states expanded reduces. At higher values of δ_e and δ_l, the planner rapidly increases the time scaling of the control action primitives resulting in an improvement in the computational performance. However, this improvement in the computation performance of the planner comes at the cost of increase in total travel distance by the USV. This behavior of the planner is demonstrated in Table 3.3, where the percentage additional distance traveled by the USV is evaluated with respect to discrete values of δ_e and δ_l.

Ideally, the best values of δ_e and δ_l are the ones which provide the shortest travel distance and the fewest expanded states. Unlike, spatio-temporal complexity parameters it is difficult to select one optimum value of δ_e and δ_l that will work for all the scenarios with different level of congestion value. For the scenarios with high congestion value (i.e. 40-45), the environment has large number of civilian vessels and less free space. Thus, it is beneficial to select lowest values of δ_e and δ_l and gradually scale the control action primitives. From the Table 3.3 we can see that in the scenarios with high congestion value (i.e. $\Lambda_{cgn} = 40 - 45$), it is beneficial to select the lowest values of δ_e and δ_l (i.e. $\delta_e = 1.4$ and $\delta_l = 0.1$) because it provides the shortest travel distance. Similarly, for the scenarios with low congestion values (i.e. $\Lambda_{cgn} = 10 - 15$) we want to rapidly scale the control primitives. Thus we can select the higher values of δ_e and δ_l (i.e. $\delta_e = 2$ and $\delta_l = 0.1$) which provide only 0.415% of

Table 3.3: *Mean of percentage additional distance traveled by varying the exponential and linear increment parameters for scenario with different congestion value.*

Exponential Increment δ_e	Linear Increment δ_l	Mean % of Additional Travel Distance		
		Congestion 10 - 15	Congestion 25 - 30	Congestion 40 - 45
1.4	0.1	0.59	1.235	2.07
	0.4	1.38	1.955	2.177
	0.7	0.56	0.375	5.17
	1	0.49	1.98	5.025
1.6	0.1	0.435	2.13	2.87
	0.4	1.25	1.715	4.36
	0.7	3.395	1.22	5.42
	1	1.975	2.385	6.325
1.8	0.1	0.75	1.365	2.99
	0.4	1.755	1.75	4.94
	0.7	3.68	2.6	5.76
	1	2.14	3.93	6.445
2	0.1	0.415	2.745	3.145
	0.4	2.38	2.61	5.575
	0.7	4.745	3.74	6.405
	1	5.655	4.81	7.46

additional travel distance .

3.6.2 Selection of re-planning frequency

Unmanned surface vehicles (USVs) rely on several finite time variables for successfully avoiding static as well as dynamic obstacles in the environment. These variables include:

(i.) $t_{perception}$: Time taken by the perception system to identify and estimate the state of the obstacle in the given environment.

(ii.) $t_{planning}$: Computational time required by the planner to compute avoidance strategies after getting state information from the perception system.

(iii.) $t_{reaction}$: Finite reaction time required by the USV to start executing commands issued by the high level planner (deliberative or reactive).

(iv.) $t_{execution}$: Time required by the USV to execute an emergency control action to avoid collision.

Now, the planning frequency required for collision free operation of USV in an environment with given congestion value is governed by the time equation $t_{CPA} > (t_{perception} + t_{reaction} + t_{execution} + t_{planning})$. The planning time can be further divided into the computation time required by the deliberative planner A-RCAP ($t_{delibrative}$) and the computational time required by the VO-based reactive planner ($t_{reactive}$) i.e. $t_{planning} = t_{delibrative} + t_{reactive}$. The computation

time for the deliberative planner is higher than that of the reactive planner (i.e. $t_{delibrative} > t_{reactive}$). Thus, the planning cycle for the reactive planner is significantly smaller than that of the deliberative planner. The deliberative planner provides the global path by considering the current state of the world. However, until the next planning cycle of deliberative planner if the USV comes across any unpredicted situation the reactive planner is used perform the avoidance maneuver.

In our experiments, the computation time for the reactive planner is approximately $t_{reactive} = 300$ ms and the average planning time for the deliberative planer (A-RCAP) is approximately $t_{deliberative} = 10$ s. The approximate values of the reaction time $t_{reaction} = 1$ s and the execution time $t_{execution} = 10$ s are computed using the system identified dynamics model of the USV [1]. The execution time $t_{execution}$ is equal to the maximum of all the execution time taken by the USV to execute contingency control action primitives. Finally, the value of time to CPA used by the A-RCAP planner to perform the avoidance maneuver is set to be $t_{CPA,max} = 30$ s. This value of $t_{CPA,max}$ is determined from our previous physical experiment described in [1]. We assume the time taken by the perception system to be approximately $t_{perception} = 1$ s. Thus, the deliberative planner (A-RCAP) can successfully yield to all the civilian vessels having time to CPA above $t_{CPA} > 30$ s.

Now, the value of $t_{CPA,max}$ will depend upon the relative velocity of the USV and the civilian vessels in the current environment. If the USV has to

operate in environment with high-speed civilian vessels (i.e. lower values of $t_{CPA,max}$), then the computational performance can be further improved by better hardware and parallelization.

3.7 Summary

Dynamic obstacle avoidance is one of the most extensively studied research areas in the field of mobile robotics. However, the existing planners developed for obstacle avoidance in highly dynamic environments have one or more of the following limitations:

- Do not consider the civilian vessels (or dynamic obstacles) as an active agents and assume the obstacles will continue with the same heading and velocity.
- Do not consider the uncertainty in the predicted state of the civilian vessels.
- Do not consider the kinodynamical constraints of the USV (or robot).
- Avoids the collision locally and do not reduce the collision risk globally.

In this chapter, we introduced a lattice-based, 5D trajectory planner for an unmanned surface vehicle (USV) operating in an environment with civilian traffic. The planner:

- Estimates the collision risk of a trajectory and reasons about the availability of contingency maneuvers to counteract the unpredictable behaviors of civilian vessels.

- Incorporates the avoidance behaviors of civilian vessels into the search to minimize collision risk.
- Introduces uncertainty in the predicted state and the avoidance strategy of the civilian vessels.
- Considers the USV's dynamics and the tuned low-level controller (used to track the computed trajectory) while computing the trajectory.
- Globally minimizes the risk of collision not only between the USV and the civilian vessels but also between the civilian vessels themselves.
- Dynamically scales control action primitives based on the congestion of state space regions to maximize search performance.

We have carried out simulations to evaluate the collision rate, length, and execution time of trajectories, and computational efficiency in terms of the number of states expanded during the search for a trajectory by the developed planner. We have introduced a novel congestion metric to compare the complexity of different scenarios when evaluating the planner. The trajectories computed by the planner result in a smaller number of collisions as compared to a variation of the VO-based local planner [86] used in marine missions. This is mostly due to the consideration of the USV's dynamics, contingency maneuvers, and fast replanning through a dynamic adaptation of control action primitives. In addition, the adaptive planner computes shorter trajectories with smaller execution times compared to its non-adaptive variant as well as the VO-based planner.

The planner developed in this chapter is designed for under actuated, two-dimensional non-holonomic robots. We assume that the position of the civilian vessels and the USV does not change significantly in the time interval required by the planner to generate two consecutive paths, thus the spatial uncertainty in the position of the vessels is low. At the planned operating speed of the USV, the re-planning frequency of the planner (A-RCAP) is sufficient to maintain the positional uncertainty of the vessels within satisfactory bounds. This enables the planner to generate low-risk paths without increasing the travel path length and travel time. However, when the developed planner with same re-planning frequency is deployed on USVs traveling at high speeds (especially with vehicles traveling above 15 knots) this results in high uncertainties in the position of the civilian vessels. This increased positional uncertainty will result in correspondingly higher chances of collision between the vessels. In order to minimize the risk of collision, the planner will generate long paths with large travel times. In the future, it will be advantageous to develop an anytime-variant of the planner that can compute paths quickly to deal with highly dynamic environment and compute low-risk paths that have low travel time and distance.

Chapter 4

Speeding up A* Search on Visibility Graphs Defined Over Quadtrees to Enable Long Distance Path Planning for Unmanned Surface Vehicles

In this chapter[2], we introduce an algorithm for long distance path planning in complex marine environments.

4.1 Introduction

Over the last ten years, substantial progress has been made in the development of low-cost unmanned surface vehicles (USVs) [2, 4]. There are a number of civilian applications where deploying USVs can significantly reduce costs, improve safety, and increase operational efficiencies. Representative applications include remote/persistent ocean sensing, marine search and rescue, and industrial offshore supply and support.

In this chapter, we are interested in path planning over long distances in complex marine environments. Figure 4.1 shows an example of an environment that consists of hundreds of islands of complex shapes. The available free space in such marine environments changes over time as a result of tides, environmental restrictions, and weather. Low tides may make it infeasible to go through regions with shallow waters. Environmental restrictions may prevent the unmanned surface

[2] The work in this chapter is derived from the published work in [37]

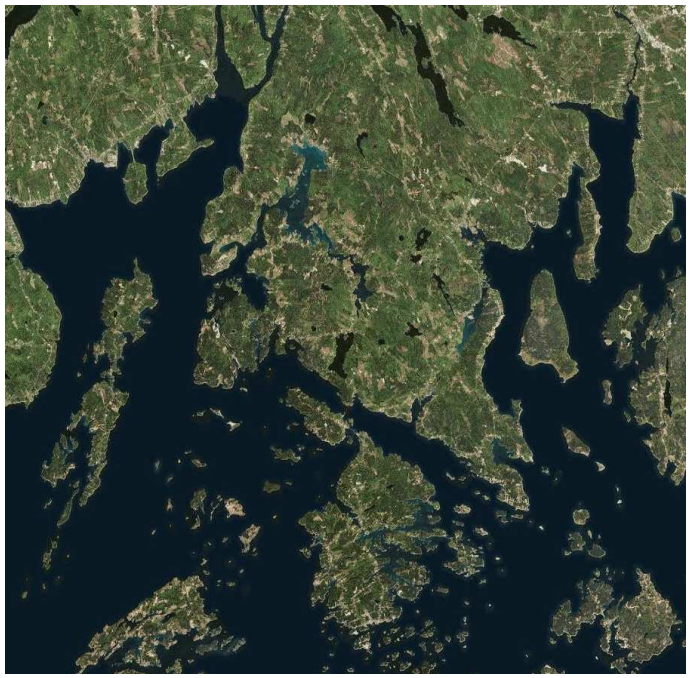


Figure 4.1: *Topography of a complex marine environment.*

vehicle from passing through certain protected marine regions for certain periods of times. Weather induced waves may prohibit traveling over certain areas due to high collision risks. As a result of these considerations, the free space region in marine environments needs to be dynamically generated and updated.

Consider a representative marine region of 100 sq. km. This region may need thousands of complex polygons to represent the land areas (or obstacles) in the marine environment. Roadmap-based methods work well with polygons-based representations [123] and have been shown to be quite useful in ground applications. However, as mentioned earlier, the free space may change in marine environments. Hence, we cannot justify the computational time and efforts needed to build roadmaps. Instead, we will need to import the obstacle field data from NOAA nautical charts by applying the appropriate height filters based on the tide conditions

at the time of the mission. Additional obstacles will need to be identified based on weather and environmental restrictions applicable at the time of the mission. These requirements restrict us to only consider those methods that can compute plans without the need for computing roadmaps.

Grid-based methods can be used to represent complex obstacle fields [126, 104]. If a grid is defined over a 100 km by 100 km region using a 10 m resolution in each dimension, the resulting grid will contain 100 million nodes. The large regions typically contain large land masses that may span several kilometers. It appears that quadtrees [136] are better spatial data structures to represent complex marine environments compared to grids (see Figure 4.2). A 100km by 100km region can be represented with a hundred thousand or fewer nodes in a quadtree. This leads to a reduction of more than a thousand nodes in terms of spatial complexity. Environmental and weather based restricted areas can be easily incorporated in quadtrees as obstacles. Hence, in this chapter we will use quadtrees as the spatial data structure to represent the free space.

In this chapter, we present an algorithm for long distance path planning in complex marine environments using A* search on visibility graphs defined over quad trees. Section 5.3 defines the visibility graphs. Visibility graphs have been shown to be useful in computing optimal paths in the presence of complex obstacle fields. However, the computational performance of visibility graphs degrades as the region to be searched becomes large and the number of nodes in the visibility graph increases rapidly. This chapter introduces a number of techniques to speed up the A* search process and makes it feasible to compute paths using visibility graphs over

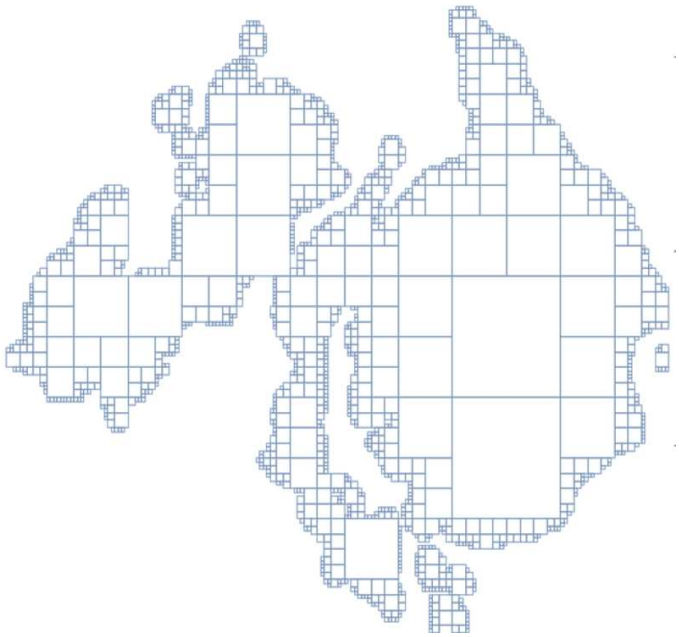


Figure 4.2: *Quadtree representation of a complex polygon.*

large regions. Previous work has shown that the optimal path goes through tangent edges in the visibility graph [157]. We exploit the data structures and the modified rotational plane sweep algorithm (RPS) [158] to compute tangent edges efficiently (see Section 4.3).

We have presented an improved heuristic to handle large obstacles in the region (see Section 4.4). Finally, we have described methods for focusing the search by looking for child nodes in certain spatial regions (see Section 4.5). The path computed using the proposed approach can be used to generate trajectories and support a wide variety of missions [151, 159, 160].

4.2 Approach

In this paper, we are interested in computing an optimal path τ on a workspace with large maps represented using a quadtree $\mathcal{M}_Q$ (see Figure 4.24), given the start $\mathbf{n_I}$ and goal $\mathbf{n_G}$ node. Each leaf node in a quadtree is represented as $\mathbf{l} = [\eta^T, d, t] \in \mathcal{M}_Q$, where $\eta = [x, y]^T$ is the center of the node in 2D space, $d \in [0, d_{max}]$ is the current depth of the node ranging from 0 (i.e. the rootnode) up to the maximum depth of the quadtree d_{max}, and t denotes the type of the node, i.e. a free node ($t = 0$), a solid or obstacle node ($t = 1$), and a node with additional branches ($t = 2$). In this paper, the word 'node' (or 'nodes') is used for the nodes of the visibility graph [128]. The quadtree nodes will be referred to as 'leaf nodes' (see Figure 4.3).

Any shortest path between the start $\mathbf{n_I}$ and the goal $\mathbf{n_G}$ node in the workspace with a set of polygonal obstacles $\mathcal{O}$ is a poly-line path whose inner vertices are vertices of $\mathcal{O}$ [158]. Two vertices $\mathbf{v}$ and $\mathbf{v}'$ are mutually visible if the line segment connecting $\mathbf{v}$ and $\mathbf{v}'$ does not intersect with the interior of the polygonal obstacle $o_i \in \mathcal{O}$, where o_i is the i^{th} obstacle in the set $\mathcal{O}$. Now, vertices $\mathbf{v}$ and $\mathbf{v}'$ will be nodes $\mathbf{n}$ and $\mathbf{n}'$ in the visibility graph $\mathcal{V}$ with an edge between them. The Visibility graph $\mathcal{V}$ is a graph whose nodes are vertices of polygonal obstacles along with the initial $\mathbf{n_I}$ and the goal $\mathbf{n_G}$ nodes. The edges of the graph represent the pair of mutually visible vertices. The shortest path between the start $\mathbf{n_I}$ and goal $\mathbf{n_G}$ nodes is the shortest path in the visibility graph $\mathcal{V}$.

In our problem, the polygonal obstacles are the solid leaf nodes ($t = 1$) of the quadtree $\mathcal{M}_Q$ and the vertices of these solid leaf nodes are the nodes of the visibility

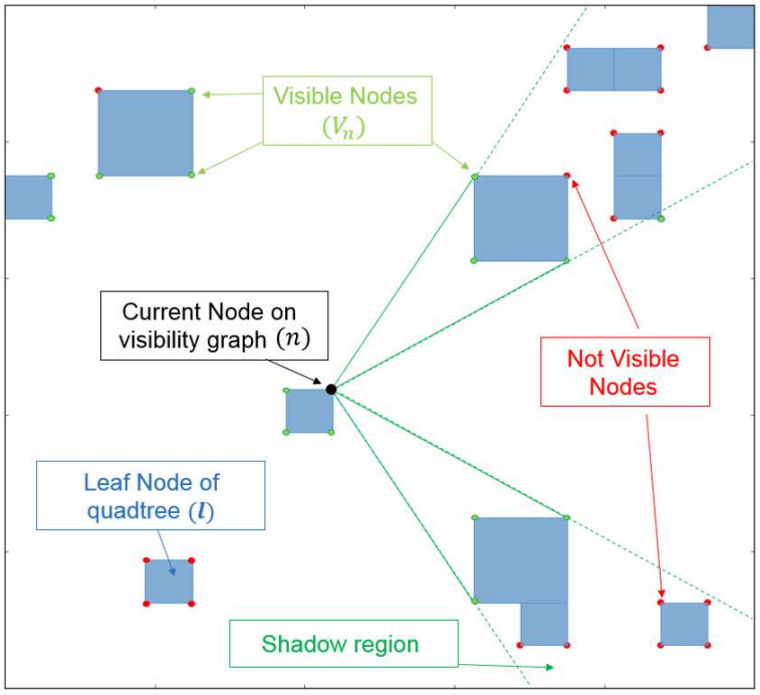


Figure 4.3: *Computation of visible nodes in quadtree.*

graph $\mathcal{V}$. The set of visible nodes at node $\mathbf{n}$ is denoted as $\mathbf{v^n}$. The example in Figure 4.3 shows the candidate visible nodes (marked by green) and non-visible nodes (marked by red) of the current node $\mathbf{n}$.

The set of visible nodes $\mathbf{v^n}$ is calculated by iterating over all the N nodes in the visibility graph $\mathcal{V}$ and performing $N-1$ collision checks to determine the visibility of the nodes. This computation proves to be computationally expensive. The complexity of the visibility graph (i.e. the number of visibility checks per node $\mathbf{n}$) can be reduced by incorporating the concept of tangent graphs (i.e. reduced visibility graph) [157].

To reduce the complexity of the graph we need to eliminate the nodes that will never be part of the optimal path τ_{opt}. First, we accumulate the connected solid leaf nodes of the quadtree tree which are termed island $\mathbf{h}_i$. Each map represented

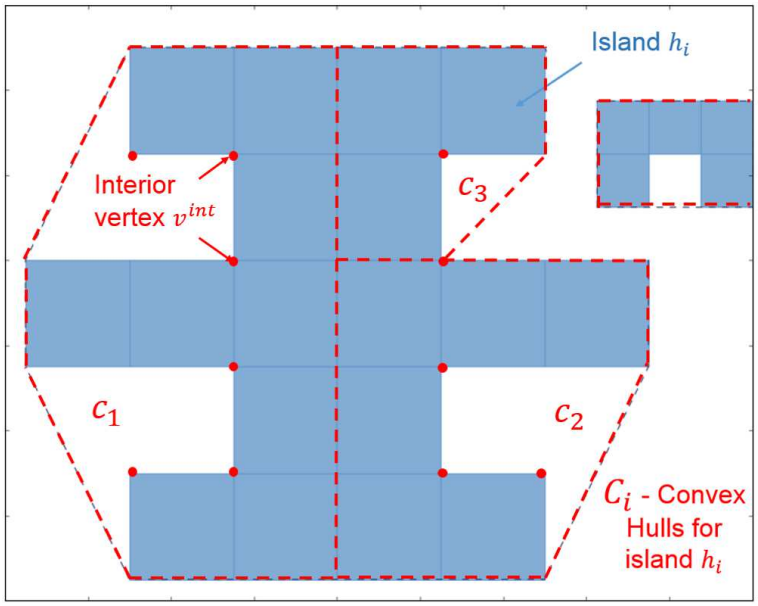


Figure 4.4: *Eliminating visible interior vertices from the visibility graph (.*

by quadtree $\mathcal{M}_Q$ may have several such islands denoted by the set H. Second, we compute the convex hull regions $\mathbf{c}_j \in C_i$ for all the island regions, where $C_i \in \mathcal{C}$ is the set of all the convex hulls for the i^{th} island in H. The convex hull $\mathbf{c}_j$ is computed such that the hull does not intersect with obstacle region $\mathcal{O}$. Thus, each island $\mathbf{h}_i$ can be represented by multiple convex hulls.

Let us consider a simple case shown in Figure 4.4, where (1) $\mathbf{h}_i \in H$ is an island (i.e. the solid quadtree leaf node). Let, C_i be the set of convex hulls of island $\mathbf{h}_i$, (2) $\tau_{feasible}$ is a feasible path that includes an interior visible node $\mathbf{n}^{int}$ (i.e. interior vertex) of the island $\mathbf{h}_i$ that is not on the boundary of $\mathbf{c}_j \in C_i$, and (3) the start $\mathbf{n_I}$ and the goal $\mathbf{n_G}$ node are outside of both the island $\mathbf{h}_i$ and all the convex hulls in $\mathbf{c}_j \in C_i$ and no other obstacle in $\mathcal{O}$ intersects with $\mathbf{c}_j \in C_i$. In this case, $\tau_{feasible}$ will cross the convex hull twice on its course to reach interior visible node $\mathbf{n}^{int}$ and then to come out of $\mathbf{c}_j$. This path $\tau_{feasible}$ can always be improved by just moving

along the convex hull rather than going inside and coming out. There is no other obstacle intersecting with $\mathbf{c}_j$ to prevent this. Therefore, the interior vertex $\mathbf{n}^{int}$ will never be in the optimal path τ_{opt}.

Now, let us consider a more general case as depicted in Figure 4.5 where (1) $C_i = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$, (2) $\mathbf{c}_j \in C_i$ does not intersect with any other obstacle in $\mathcal{O}$, and (3) $\mathbf{n}^{int}$ is an interior node that belongs to island $\mathbf{h}_i$ but not to $\mathbf{c}_j$. The optimal path τ_{opt} will not pass through $\mathbf{n}^{int}$ if $\mathbf{n_I}$ and $\mathbf{n_G}$ are outside of the convex hulls in C_i. Hence $\mathbf{n}^{int}$ can be removed from the list of potential candidate nodes for visibility graph $\mathcal{V}$. Similarly, in Figure 4.5, the nodes of the visibility graph marked with red color are interior nodes $\mathbf{n}^{int}$ that lie in the interior of the convex hulls $\mathcal{C}$. Now, as per the theorem stated above, these nodes marked with red color can be eliminated because they will not be included in the optimal path τ_{opt}. Also, the nodes that are shared by the convex hulls within the same island can be eliminated. Each island will have a set of candidate nodes after the elimination of the interior nodes $\mathbf{n}^{int}$ that are used for computing tangent edges.

The interior nodes $\mathbf{n}^{int}$ lying in the same convex hull as the initial node $\mathbf{n_I}$ or the goal node $\mathbf{n_G}$ are not eliminated, thus not altering the optimality of the path. It has been shown that the optimal path only passes through edges that are tangent to the polygonal obstacles in the visibility graph [157]. Let us denote the graph comprised of only tangent edges (i.e. tangent graph or reduced visibility graph) by $\mathcal{V}_t$. The shortest distance path between $\mathbf{n_I}$ and $\mathbf{n_G}$ is comprised of a combination of shortest straight line paths between two polygons and line segments along the exterior boundary of the polygonal obstacles. Now, the number of visible edges of

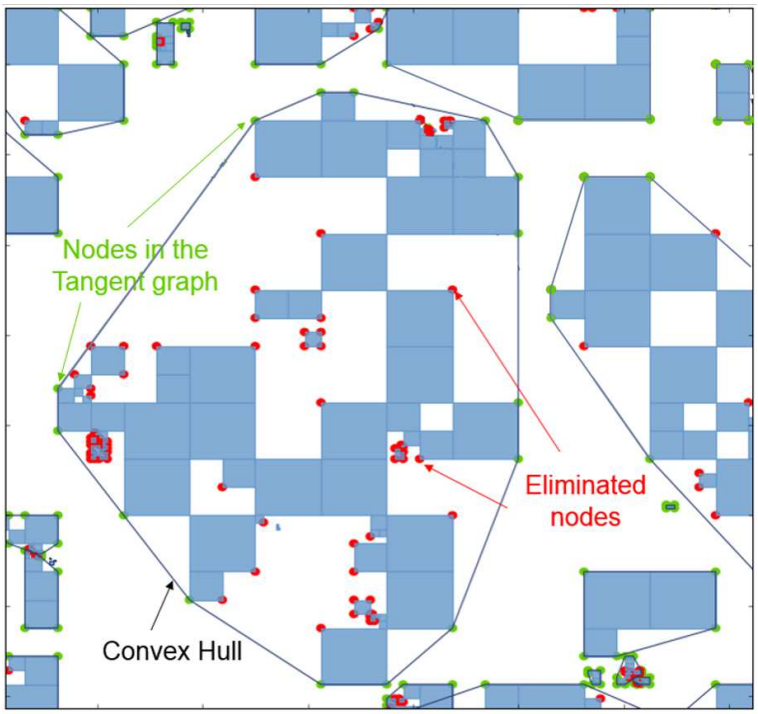


Figure 4.5: *Eliminating nodes in the interior of the convex hull (we assume the start and the goal node are not inside any convex hull)*

the current node $\mathbf{n} \in \mathcal{V}_t$ can be restricted to the tangent edges from the current node $\mathbf{n}$. This reduces the size of the visibility graph without altering the quality and optimality of the path.

4.3 Computation of Edges on Tangent Graph

Traditionally, the visibility and tangent graph-based path planning approaches pre-compute the entire structure of the graph with edges connecting the visible nodes. The computation of all the edges of the tangent graph $\mathcal{V}_t$ is computationally expensive. In our approach, the edges of the graph are computed on the fly during the search for the optimal path τ. The search is performed by expanding nodes in the least-cost A* [104] fashion according to the cost function $f(\mathbf{n}) = g(\mathbf{n}) + h(\mathbf{n})$, where $g(\mathbf{n})$ is the cost-to-come to node $\mathbf{n}$ from the initial node $\mathbf{n_I}$, and $h(\mathbf{n})$ is the heuristic estimate of the cost-to-go from the current node $\mathbf{n}$ to the goal node $\mathbf{n_G}$.

During the expansion of each node $\mathbf{n} \in \mathcal{V}_t$, we need to determine tangents for each island $\mathbf{h}_i \in H$ and these tangents have to be checked for collisions in order to determine the visible nodes $\mathbf{v_t^n}$. This process of determining the visible nodes $\mathbf{v_t^n}$ is computationally intensive and hence reduces the performance of the search algorithm.

The computational performance of the search can be improved by reducing the number of collision checks required during the determination of visible nodes. The reduction in collision checks is achieved by our implementation of a modified variant of the rotational plane sweep algorithm (RPS) [158], in which we compute

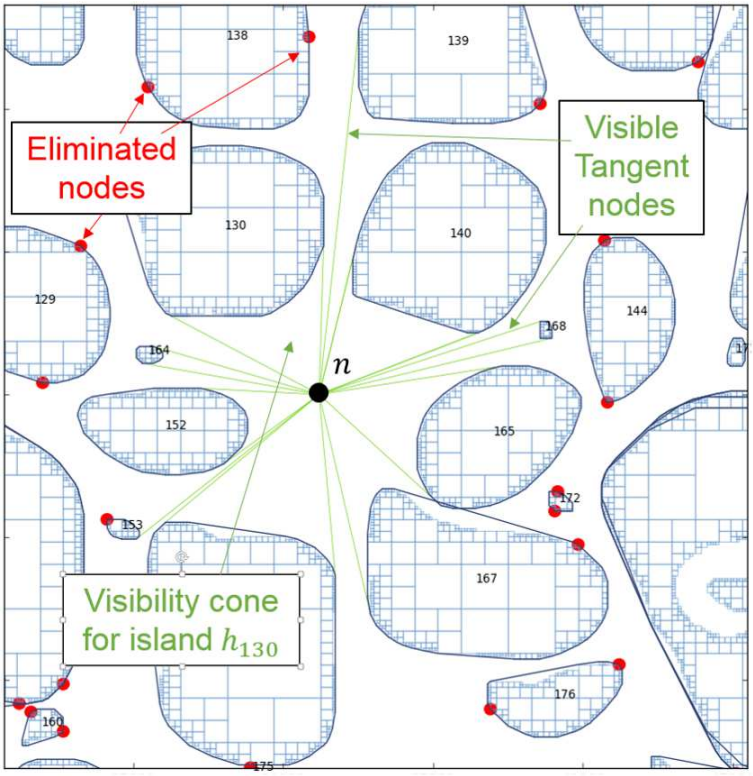


Figure 4.6: *Elimination of non-visible nodes using the computed tangents for the islands in H*

the angle and distance to each node in $\mathcal{V}_t$ from the current node $\mathbf{n_c}$. During the computation of these angles and distances, we determine maximum and minimum angle for each island $\mathbf{h}_i$ which serve as tangents from the current node. The list of all the tangents are sorted based on their angles with respect to the current node $\mathbf{n_c}$.

Now, these tangents to each island $\mathbf{h}_i \in H$ form cone like structures with the apex at the current node $\mathbf{n}$. We refer to these cones as visibility cones for the island $\mathbf{h}_i$. During the visibility check, we can directly eliminate all the candidate nodes with an angle lying in between the two edges of the cone and distances greater than the tangent nodes for the nearest visible convex hull $\mathbf{h}_i$. This drastically reduces the number of collision checks required during the determination of visible tangent nodes and improves the efficiency of the algorithm. For example, in Figure 4.6 the tangent nodes of island 138 and one of the tangent nodes of island 129 can be directly eliminated by a visibility cone of island 130. Finally, the collision checks for determining visible tangent edges that cannot be eliminated by the modified RPS algorithm are performed by a modified Bresenham's collision test algorithm described in [161].

4.4 A New Heuristic

The performance of the A* based path planning algorithm depends on the estimation of the cost-to-go (or h-cost) from the current state $\mathbf{n_I}$ to the goal state $\mathbf{n_G}$. If the path planner significantly underestimates the cost-to-go, then it has to expand more

nodes until it finds an optimal path to the goal. On the other hand, if the path planner overestimates the cost-to-go, then the h-cost is inadmissible and the paths are no longer optimal.

The most widely used heuristic by the path planning algorithms [1, 159, 38, 34, 35] is the Euclidean distance from the current node $\mathbf{n_I}$ to the goal node $\mathbf{n_G}$. This heuristic expands nodes nearest to the goal in a greedy fashion. This assumption works perfectly on maps without any large obstacle regions. In scenarios with large obstacles, the shortest optimal path has to circumvent at least one obstacle before heading towards the goal, unless the goal is in line-of-sight to the initial location. The heuristic based on Euclidean distance will expand all the nodes lying on the perimeter of the obstacle in a greedy fashion until a straight line path is available to the goal. However, in most of the complex marine environments (see Figure 4.1) we have large islands and the use of the heuristic based on Euclidean distance often degrades the performance of the path planner.

We have developed a heuristic that exploits the fact that the optimal path reaching towards the goal has to pass through one of the corners of the obstacle obstructing the straight line path to the goal node. Let us consider a scenario shown in Figure 4.7. In this example, the straight line path from the current node $\mathbf{n_c}$ to the goal node $\mathbf{n_G}$ intersects two obstacles o_1 and o_2. The Euclidean distance heuristic from $\mathbf{n}$ to $\mathbf{n_G}$ is given by $h_E(\mathbf{n}) = d(\mathbf{n}, \mathbf{n_G})$, where $d(\mathbf{n}, \mathbf{n_G})$ is the straight line distance. The vertices $\mathbf{v}_{R1}$ and $\mathbf{v}_{R2}$ are the rightmost vertices of obstacles o_1 and o_2 and their corresponding orthogonal distances are denoted by d_{R1} and d_{R2}. Similarly, the leftmost vertices are $\mathbf{v}_{L1}$ and $\mathbf{v}_{L2}$ and their corresponding orthogonal

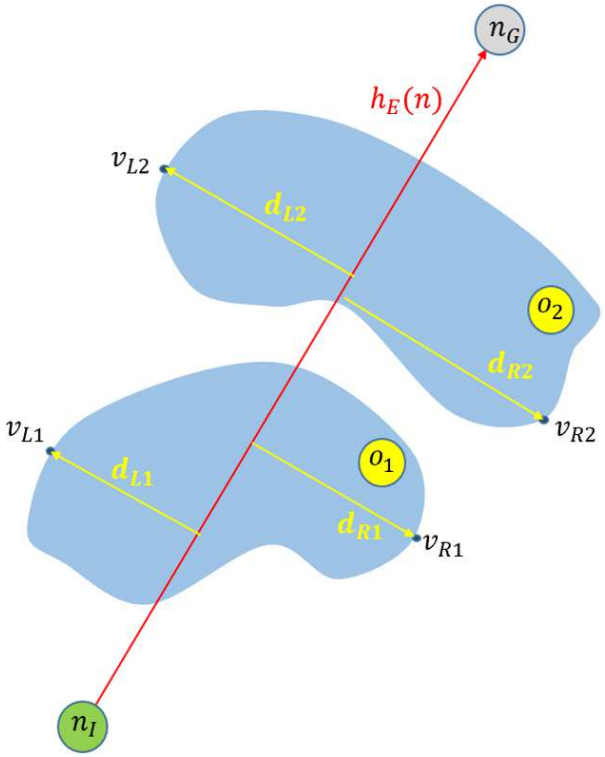


Figure 4.7: *Designed heuristic*

distances are denoted by d_{L1} and d_{L2}. The shortest route to the goal will have to pass through at least one of the extreme vertices of obstacle o_1 or o_2 represented in the configuration space, i.e. the path will be triangular with the middle vertex $\mathbf{v}_{Xn} \in \{\mathbf{v}_{R1}, \mathbf{v}_{R2}, \mathbf{v}_{L1}, \mathbf{v}_{L2}\}$, where $X \in \{R, L\}$ and $n \in \{1, 2\}$. Let the extreme vertex corresponding to distance d_{Xn} be denoted by $\mathbf{v}_{Xn}$.

The triangular path length to travel from the right side of the obstacle is given by $h_T^R(\mathbf{n}) = d(\mathbf{n}, \mathbf{v}(max(d_{R1}, d_{R2}))) + d(\mathbf{v}(max(d_{R1}, d_{R2})), \mathbf{n_G})$. Similarly, the path length to travel from the left side is denoted by $h_T^L(\mathbf{n})$. Finally, the admissible triangular heuristic cost is computed as $h_T(\mathbf{n}) = min(h_T^L(\mathbf{n}), h_T^R(\mathbf{n}))$.

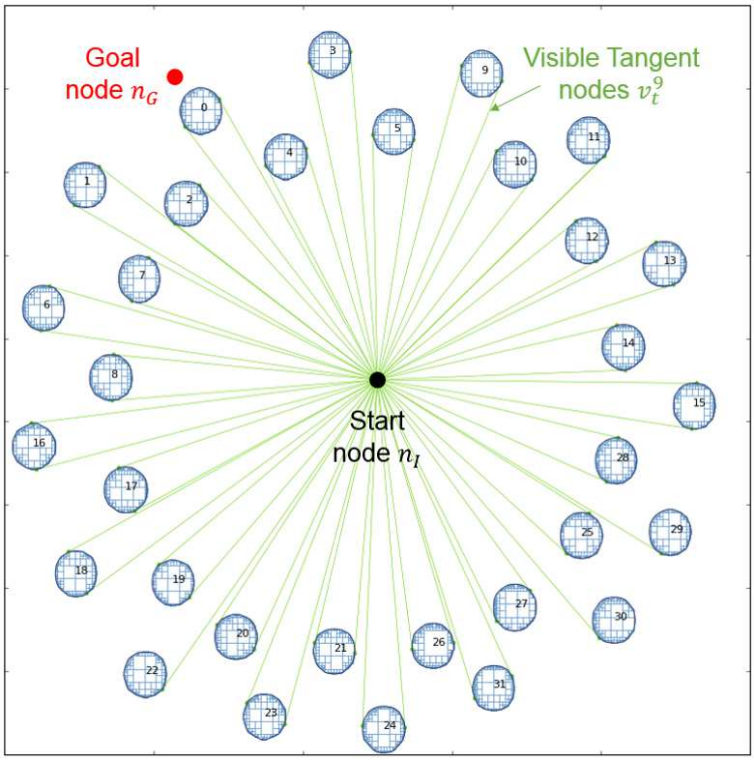


Figure 4.8: *Pathological scenario where a node in tangent graph has a large branching factor.*

4.5 Focusing A* Search

In order to guarantee optimality, the A* algorithm needs to explore all possible child nodes for a node being expanded. In large spatial regions, there can be a large number of nodes that need to be examined to determine if the straight line between them and the node being expanded belong to the tangent graph. In certain pathological cases, the number of edges on the tangent graph can be very large (see Figure 4.8). This can lead to poor computational performance during the search.

In order to improve the computational performance of the algorithm, we can focus the search and examine only certain kinds of edges on the tangent graph. For

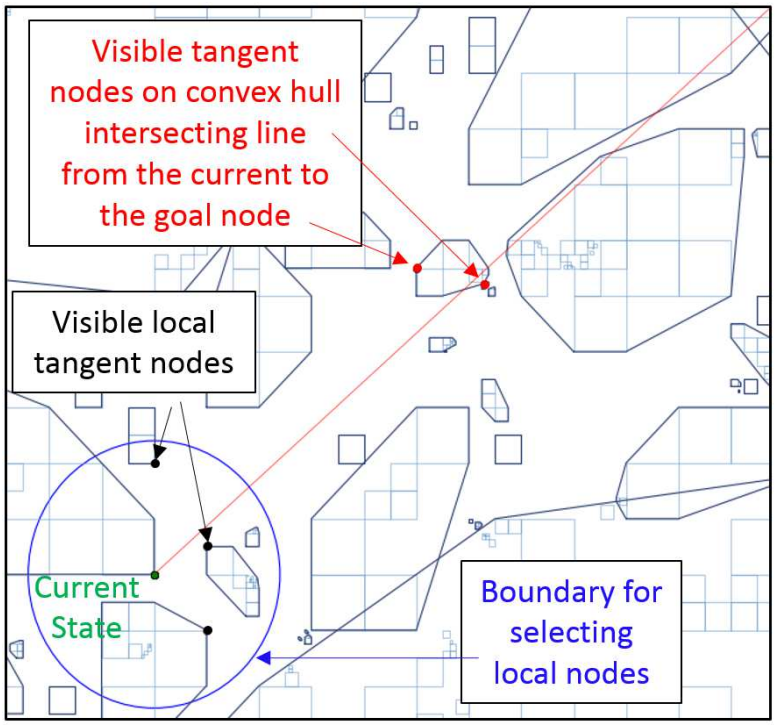


Figure 4.9: *Procedure to add nodes to the focused visibility graph.*

example, we can search for the edges in a spatial region that lies within a certain radius of the current node. However, if a fixed radius is used, then we may not be able to find any edge to explore if all other obstacles lie outside of the given radius. Therefore, in addition to the radius, we also consider adding the edges that lie on obstacles that intersect with the straight line path from the node being expanded and the goal (see Figure 4.9). This approach focuses the search and ensures that we do not encounter pathological cases. Also, during the search the child nodes of the current node $\mathbf{n_c}$ are checked for direct line-of-sight connection with the parent node. This enables a line-of-sight connection between the current node and the nodes that lie outside the constrained region.

The path generated by the focused search is not necessarily optimal. Hence

we are interested in characterizing the path with respect to the optimal path. Let us assume that L is the length of the path generated using the focused search. Now, let us construct a circle of radius L at the start node $\mathbf{n_I}$ and identify the set of visible nodes $\mathbf{v^n}_{t,L} \in \mathcal{V}_t$ that lie inside the circle of radius L. Any node on the reduced visibility graph (i.e. tangent graph) that is outside of the radius L will have a path length of more than L from $\mathbf{n_I}$, hence it cannot be on the optimal path.

Now we will compute the sum of cost-to-come $g(\mathbf{n})$ and cost-to-go $h(\mathbf{n})$ for all the visible node $\mathbf{n} \in \mathbf{v^n}_{t,L}$. Let L' be the minimum among all the visible nodes in $\mathbf{v^n}_{t,L}$. The optimal path length cannot exceed L' because the optimal path has to go through nodes in $\mathbf{v^n}_{t,L}$. If $L \geq L'$, then L is the optimal solution. If this condition is not satisfied, then L' can be used to compute a bound on how far off L is from the optimal solution. The optimal solution cannot improve the path length given by L by more than $100(L - L')/L$ percent. If this bound is relatively small, then the search can be terminated and L can be returned as the solution.

If L' is much smaller than L, then a second round of the search can be conducted with an adaptive radius of focus. At the start node we can use L as the radius of focus. As the search progresses, this radius can be reduced to L minus the cost-to-come for the node being expanded. Let $\mathbf{n_c}$ be the current node being expanded and $g(\mathbf{n_c})$ be its cost-to-come, then the remaining cost of the optimal path cannot exceed $L - g(\mathbf{n_c})$, therefore there is no need to look for successor nodes that are more than $L - g(\mathbf{n_c})$ distance away from $\mathbf{n_c}$. This search always produces the optimal answer.

4.6 Assessing Effectiveness of Focused Search

In the previous section, we have introduced the concept of focusing the A* search to reduce the branching factor of the search tree and improving the computational efficiency of the algorithm. The approach used to focus the A* search selectively considers the nodes that have higher chances of being on the optimal path and prunes the rest. The nodes that are selected can be broadly categorized in two regions. First, the nodes that lie inside a circular region around the current node $\mathbf{n}_c$ and is termed the "local search region" (see Figure 4.10). Let L be the radius of the local search region, then all the visible nodes inside the local search region is given by $\mathbf{v^n}_{t,L}$. Second, the nodes that lie on the nearest island that intersects the direct line connecting the current node $\mathbf{n}_c$ and the goal node $\mathbf{n_G}$ (see Figure 4.10). This region is named the "extended search region". The visible nodes selected from the extended search region guides the search in the direction of the goal and are denoted as by $\mathbf{v^n}_{t,E}$.

We are interested in studying the influence of the radius of local search region on the probability of finding the optimal path. The optimal path τ^{opt} comprises of linear line segments that connect nodes in direct line-of-sight from the initial node $\mathbf{n_I}$ to the goal node $\mathbf{n_G}$. The focused search approach is able to compute the optimal path provided it captures the next node on the optimal path from the current node. Let the next node on the optimal path be denoted by $\mathbf{n}^{opt}_{next}$.

The effect of the size of local search region L on the probability of capturing the next node on the optimal path is governed by the distribution of the obstacles

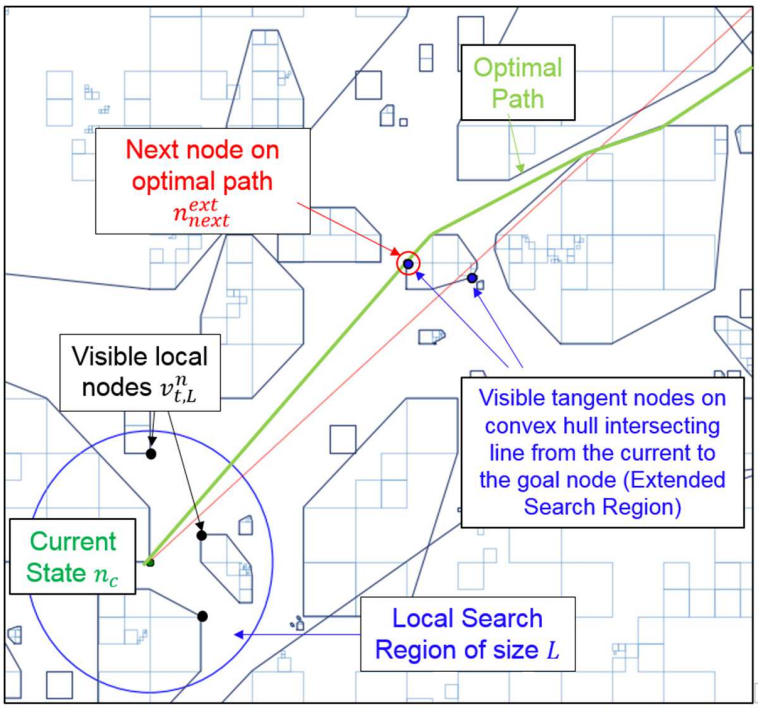


Figure 4.10: *Selection of nodes that lie inside the local and extended search regions.*

or nodes in the given scene. For example, if the region is sparse, then the absolute size of the local search region required to capture the next node on the optimal path should be large and vice versa. Thus, we require a common metric that is independent of the distribution of the scenario and can be used to study the effect of L on probability of capturing $\mathbf{n}_{next}^{opt}$. We have selected the mean length of tangents L_{scn}^{mean} between the islands in the scene as a metric to study the effect of size of local search region on probability of capturing the next node on the optimal path.

We are interested in characterizing the probability of capturing the next node on the optimal path when using the focused search. The next node that lies on the optimal path falls into any one of the following four possibilities:

1. The next node on the optimal path lies in the local search region i.e., $\mathbf{n}_{next}^{opt} \in$

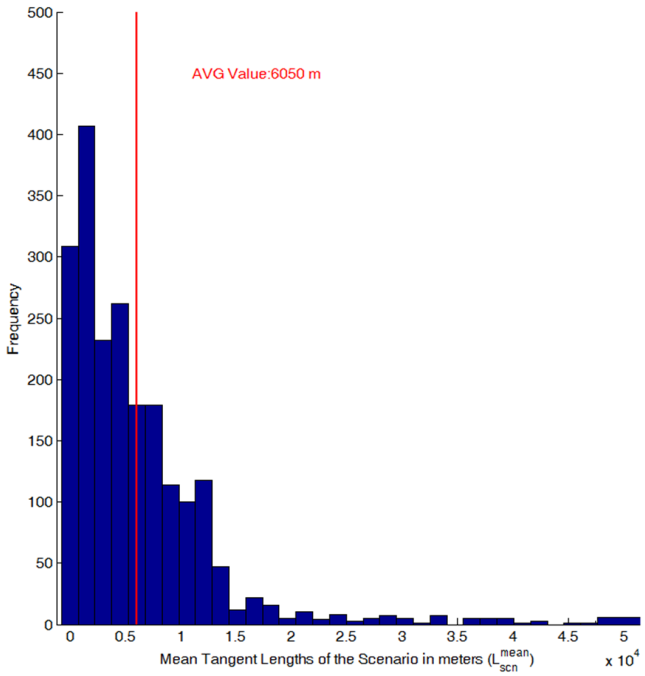
$\mathbf{v^n}_{t,L}$, which is denoted as $\mathbf{n}_{local}^{opt}$

2. The next node on the optimal path lie in the extended search region i.e., $\mathbf{n}_{next}^{opt} \in \mathbf{v^n}_{t,E}$, which is denoted as $\mathbf{n}_{ext}^{opt}$

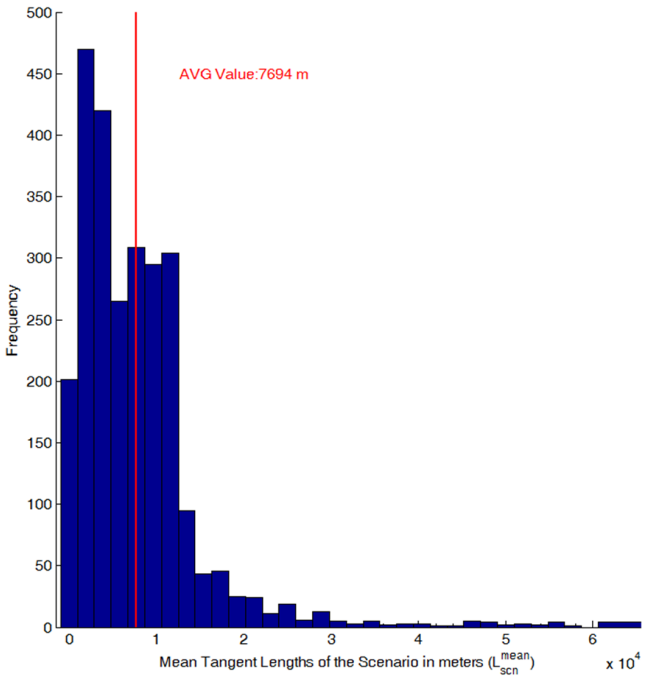
3. The next node on the optimal path does not lie in the local or extended search region but is able to connect to the current node $\mathbf{n}_c$ via back connection. Back connection is a method in which the child nodes are checked for a direct line-of-sight connection with the parent of current node i.e. grandparent. This method helps us to determine the direct connection between the nodes that lie marginally outside the local search region and the current node. The next node on the optimal path that are discovered by back connection are denoted as $\mathbf{n}_{back}^{opt}$.

4. The last possibility is that the next node on the optimal path cannot be found as a result of focusing the search.

The distance distribution plot shown in Figure 4.11(a-d) shows the distances between the islands in few of the scenarios. The plots shows that length of the tangents gradually decays and most of the tangents in the scenario are less than the mean (shown by the red line). Figure 4.12 shows the percentage of next node on the optimal path that lie inside the local search region by varying the size of the local search region from 10 to 250% of the mean tangent length. We can see that initially, the curve increases in super-linear fashion and then begins to saturate as we reach higher percentage of the mean tangent length. At the mean tangent length

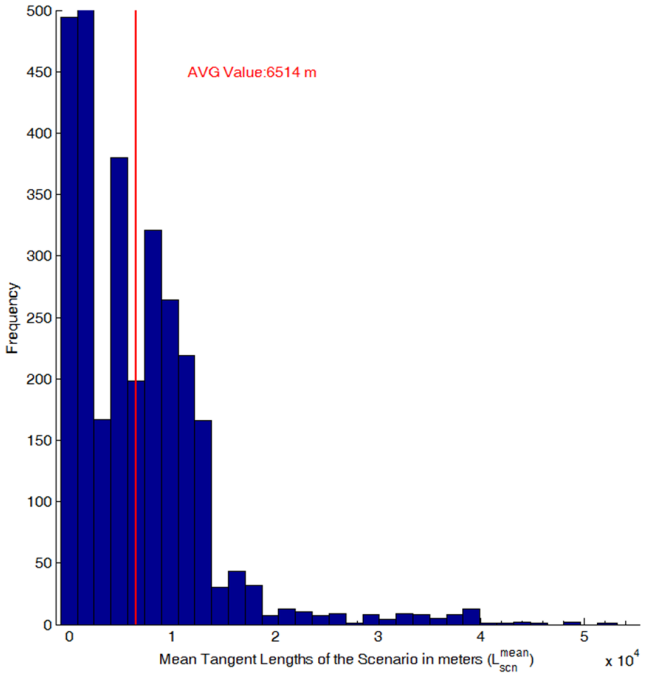


(a)

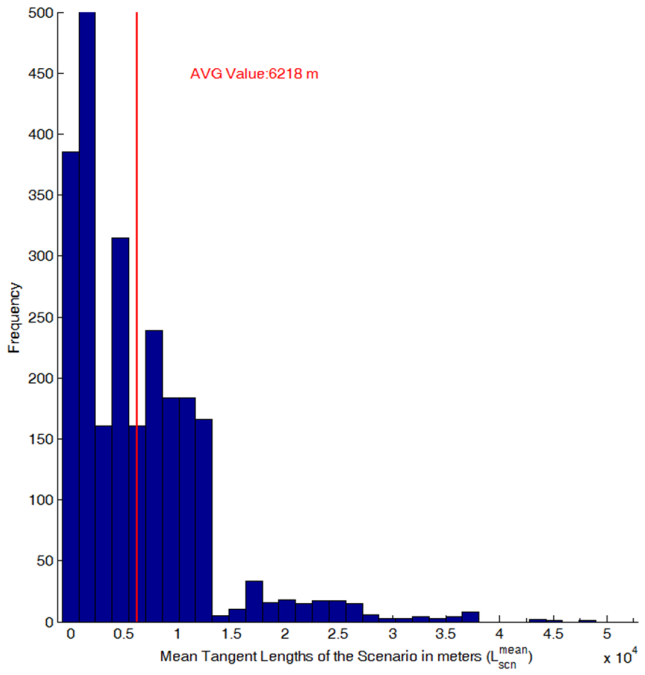


(b)

Figure 4.11: *Distribution of distances between the islands in four out of ten randomly generated scenarios.*



(c)



(d)

Figure 4.11: *Distribution of distances between the islands in four out of ten randomly generated scenarios.*

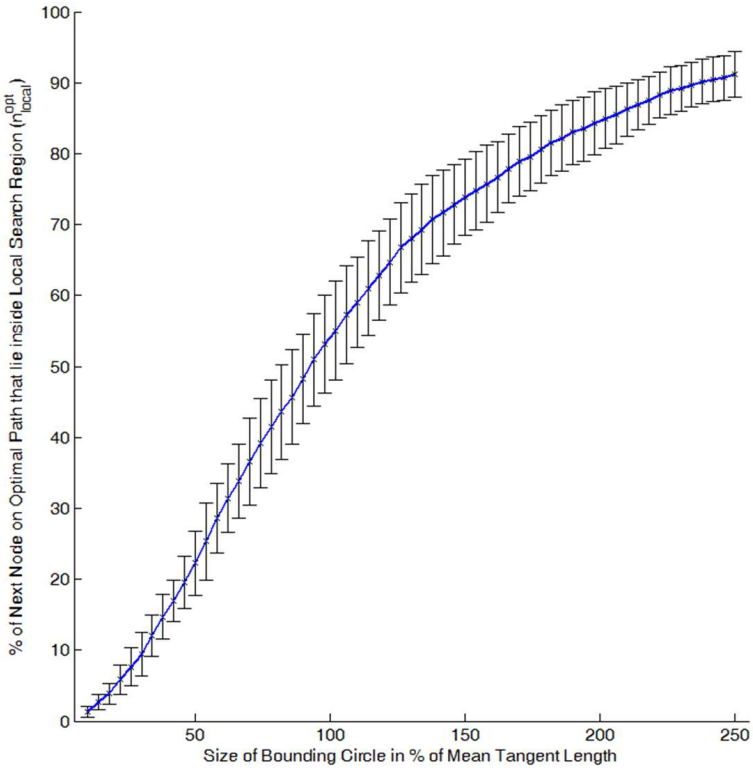


Figure 4.12: *Absolute percentage of next node on the optimal path that lie inside the local search region.*

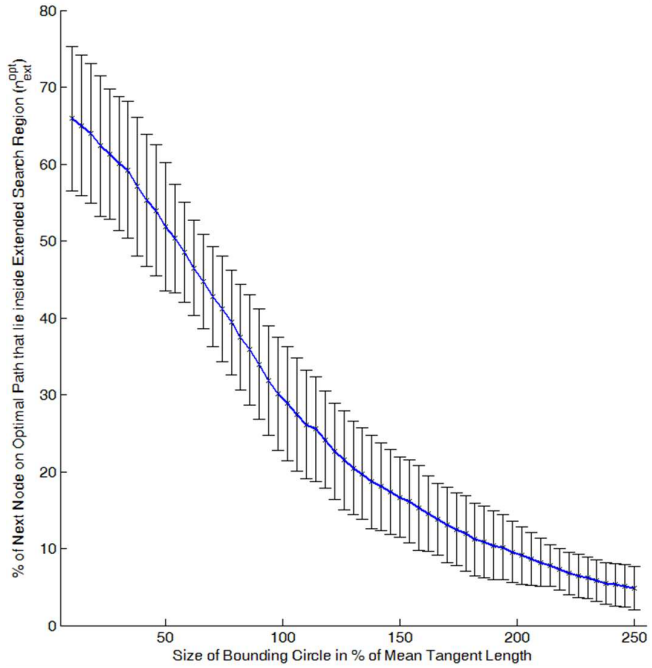
(i.e. 100 %) approximately 55-60% of the next node on the optimal path lies inside the local search region.

The plots in Figure 4.13(a) shows the absolute percentage of next node on the optimal path that lies inside extended search region ($\mathbf{n}_{ext}^{opt}$). We see the curve dropping as the size of the local search region is increased from 10 to 250% of the mean tangent length. This reflects the fact that the nodes lying on the nearest island intersecting the current-to-goal node line falls inside the local search region and are thus eliminated from the extended search region. The plots in Figure 4.13(b) shows the proportional number of the "next node" on the optimal path that falls inside

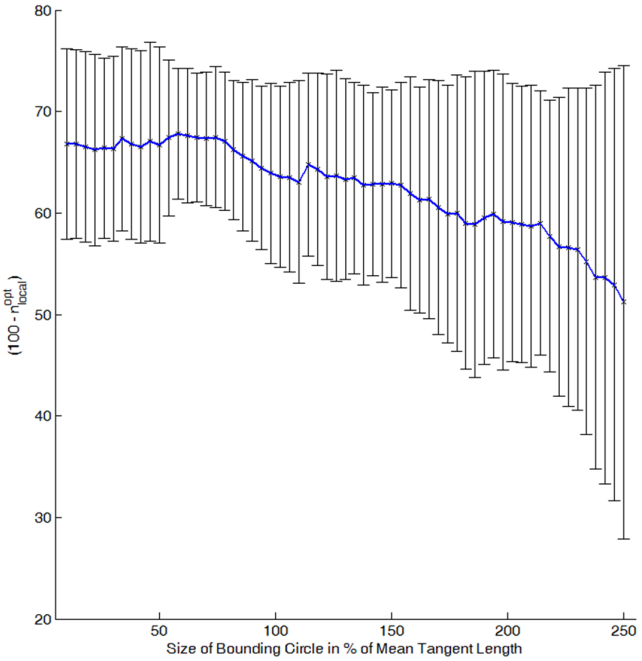
the extended search region i.e. , the percentage of remaining nodes on optimal path after eliminating the nodes that lie inside the local search region $(100 - \mathbf{n}_{local}^{opt})$. We see that from the remaining nodes the approximately 65% of the nodes lie inside the extended search region. At mean tangent length (100%), approximately 65% of the remaining nodes lie inside extended search region. This further reduces the chances of not finding the next node on the optimal path.

We can see the plot in Figure 4.14(a) shows the absolute percentage of the next nodes on the optimal path that are discovered using the back connection approach $(\mathbf{n}_{back}^{opt})$. We can see the plot increasing until the size of the local search region reaches 50% of the mean tangent length. This behavior is due to the fact that when the size of local search region is small, less nodes lie inside the region and there are more nodes on the optimal path yet to be discovered. However, when the size of the local search region increases from 50 250% the curve starts to decay, primarily because there is a low number of undiscovered nodes.

Figure 4.14(b) shows the proportional number of the next nodes on the optimal path that are discovered by the back connection approach i.e. , the percentage of nodes from the remaining undiscovered nodes left after eliminating nodes that lie in the local and extended search region. In contrast to the previous plot, we can see that the curve increases up to 150% and saturates from 150 250% of the mean tangent length. With the increase in size of local search region, the distance between the child nodes and the current node increases. In the back connection approach, we check for connection between the grandparent and the children of the current node. This helps us to discover the nodes that lie at a distance greater than the size

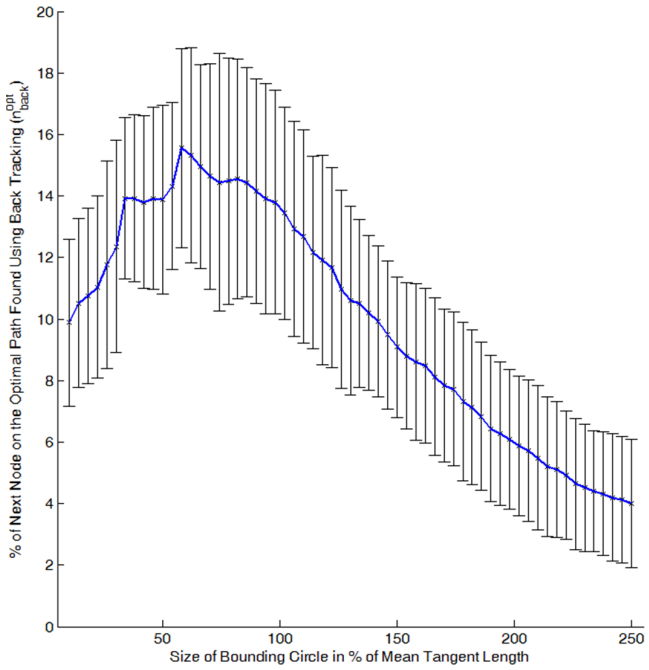


(a)

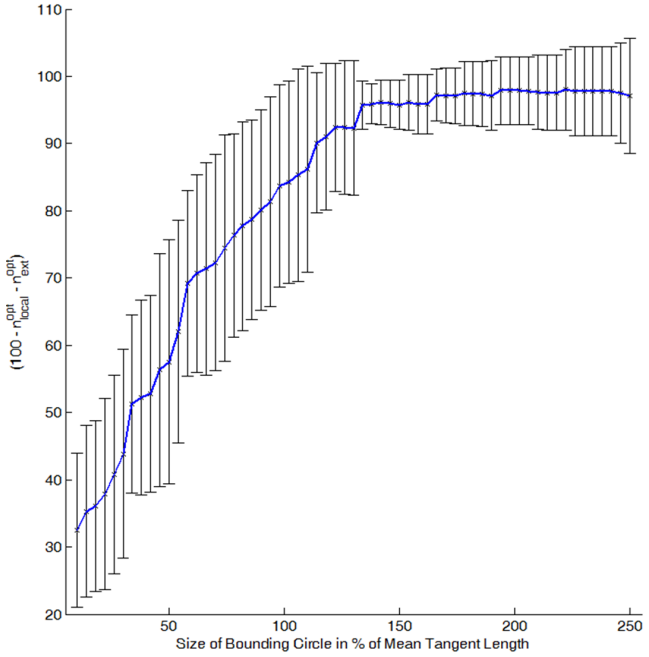


(b)

Figure 4.13: *(a)Absolute percentage, and (b) Proportional percentage of next node on the optimal path that lie inside the extended search region.*



(a)



(b)

Figure 4.14: *(a)Absolute percentage, and (b) Proportional percentage of next node on the optimal path that are discovered by the back connection.*

of the local search region but smaller than twice the size of local search region.

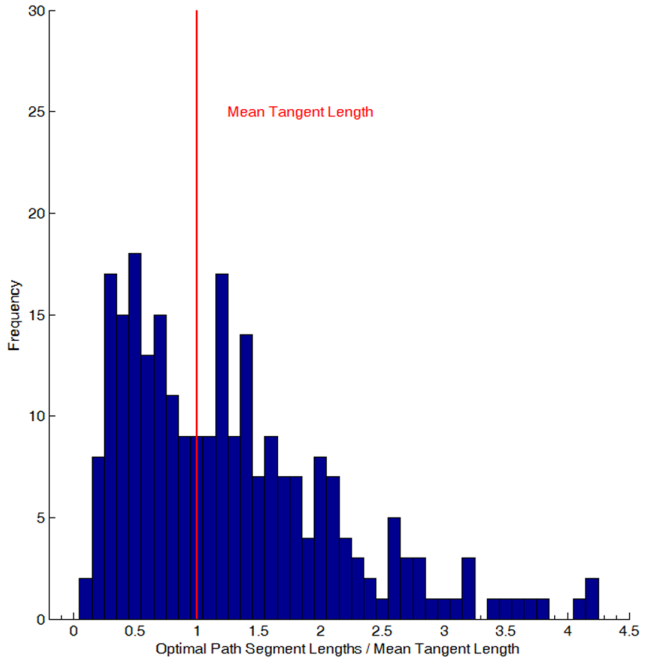
Figure 4.15(a-d) shows the histogram of the ratio of optimal path segment lengths v/s mean tangent lengths of few scenarios from the experimental data. We can see that still there is significant number of optimal path segments that lie between the mean tangent length and the twice mean tangent length of the scenario. These larger segments of optimal path are discovered by the back connection approach until the size of the local search region increases up to 150%. At the mean tangent length (100%), we see that approximately 80% of the remaining undiscovered next nodes that lie on the optimal path are discovered.

Finally, the Figure 4.16 shows the absolute percentage of the next nodes on the optimal path do not lie in the local and extended search region and are not discovered by the back connection approach. We see that the curve decays drastically and almost reaches 0% when the size of the local search region is 125% of the mean tangent length.

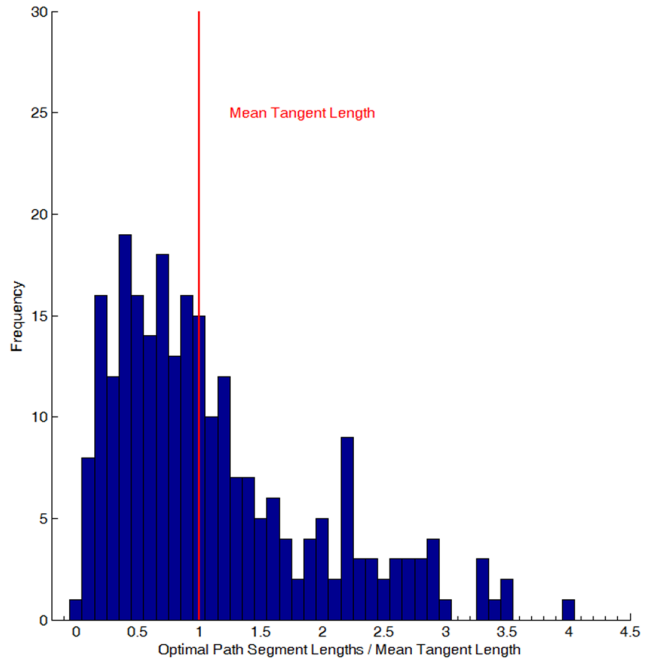
Thus, by selecting a local search region size of approximately one and half times the mean tangent length ensures that the probability of not finding the optimal path is below 1%.

4.7 Handling Time Varying Free Space

In marine environment, the available free space varies over time as a result of ocean current and tides, environmental restrictions and weather. For example, areas where low tides reduces the depth of the water are inaccessible for certain type of marine

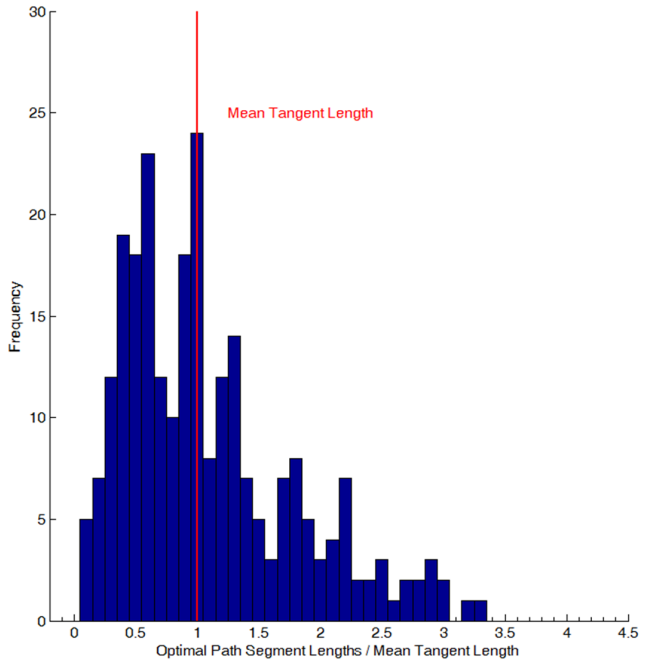


(a)

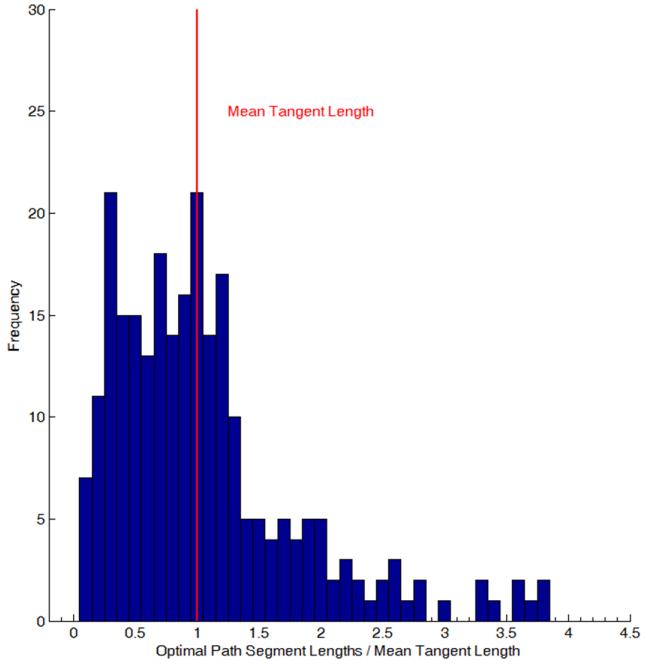


(b)

Figure 4.15: *Distribution of length of line segments on the optimal path in four out of ten randomly generated scenarios.*



(c)



(d)

Figure 4.15: *Distribution of length of line segments on the optimal path in four out of ten randomly generated scenarios.*

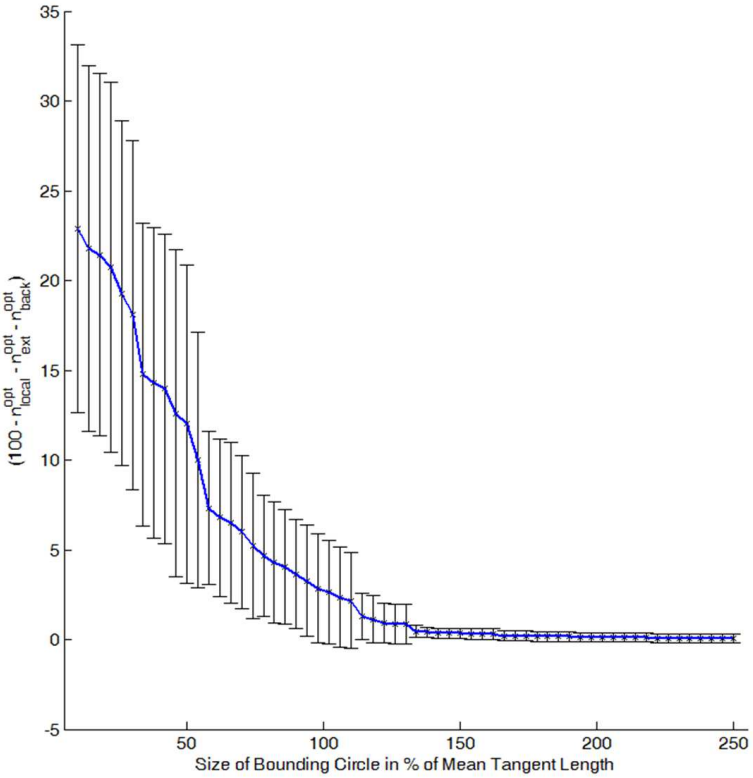
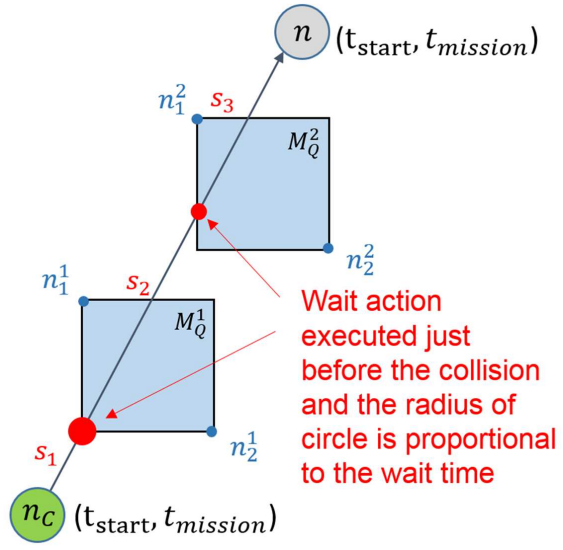


Figure 4.16: *Absolute percentage of next nodes on the optimal path do not lie in the local and extended search region and are not discovered by the back connection approach.*

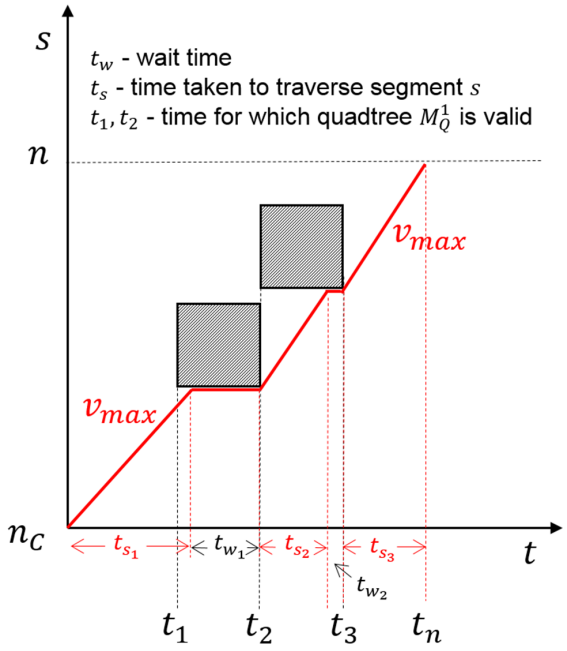
vessels. Also, inclement weather increases the sea-state (or tide) level, making narrow passages in the environment extremely risky to travel. The path planner have to account for the changing free-space during the computation of long distance paths for a USV.

Previously, in 4.2, we have described the procedure to compute a reduced visibility graph given the quadtree representation of the nautical chart data corresponding to the area of the mission. In order to capture the time-varying nature of obstacles (or the free space) of a given geographical area, we need to consider the nautical chart data for the duration of the mission ($t_{mission}$). We can compute discrete quadtree representation of the nautical chart data observed over mission duration. Each quadtree representation $\mathcal{M}_Q^i$ is valid form t_i to t_{i+1}. We compute a reduced visibility graph $\mathcal{V}_t^i$ structure for each quadtree representation having time $0 \geq t_i, t_{i+1} < t_{mission}$. The nodes of this reduced visibility graph $\mathcal{V}_t^i$ are denoted by $\mathbf{n}_j^i$. During the computation of the collision free path, the nodes in $\mathcal{V}_t^i$ are only available for time duration of the quadtree i.e., t_i to t_{i+1}. However, there will be set of nodes that will be available throughout the duration of the mission and will be part of all the reduced visibility graphs. For example, the initial node $\mathbf{n_I}$ and the goal node $\mathbf{n_G}$.

Let us consider the example shown in Figure 4.17. The blue squares $\mathcal{M}_Q^1$ and $\mathcal{M}_Q^2$ are the quadtree representation of the obstacles. The nodes on $\mathcal{M}_Q^1$ denoted by $\mathbf{n}_1^1$ and $\mathbf{n}_2^1$ are the nodes of the visibility graph $\mathcal{V}_t^1$ that are valid for time duration t_1 to t_2. There are several ways we can traverse from the current node $\mathbf{n}_c$ to the next node $\mathbf{n}$. One of the option is to traverse via $\mathbf{n}_1^1$, $\mathbf{n}_1^2$, and $\mathbf{n}$. A second option is to



(a)



(b)

Figure 4.17: *(a)Example illustrating the computation of travel cost from the current node* $\mathbf{n_c}$ *to node* $\mathbf{n}$ *in an environment with two quadtrees* $\mathcal{M}_Q^1$ *and* $\mathcal{M}_Q^2$*.(b) Graph of time v/s distance shows the progression of the computed path with traversal time for each segment denoted by* t_s *and wait time by* t_w

.

wait for the duration of the first $\mathcal{M}_Q^1$ and second $\mathcal{M}_Q^2$ quadtree to expire and then directly head to the next node $\mathbf{n}$. The option that reduces the travel time depends upon the time duration of the quadtree i.e. t_1, t_2, t_3 and the travel time required by the USV i.e., the maximum speed of the USV.

In our implementation, we have decided to compute the time required by the USV to travel to the next node $\mathbf{n}$ by including the wait time (see Figure 4.17 (b)). The direct line from $\mathbf{n_c}$ to $\mathbf{n}$ collides with, first with the obstacle in $\mathcal{M}_Q^1$ and then with the obstacle in $\mathcal{M}_Q^2$. These collisions help us to break down the segment connecting $\mathbf{n_c}$ and $\mathbf{n}$ into 3 segments labeled as s_1, s_2, and s_3. Figure 4.17 (b) shows that the USV traverses the segment s_1 at the maximum velocity v_{max} up to the point immediately prior to collision with obstacles in $\mathcal{M}_Q^1$ and waits for duration t_{w_1} before it starts traveling along segment s_2. However, in the execution phase the USV can prefer to go at lower speeds to avoid waiting. The path planner decides to wait at the point just before the collision takes place rather than waiting at the current node $\mathbf{n}_c$. This eliminates the possibility of a new obstacle appearing between the current node and the point of collision when the duration of the quadtree expires (at which collision occurred).

We have performed simulated experiments to evaluate the performance our path planner in the presence of time-varying obstacles. In contrast to the path planning algorithm that is used to computes paths in the static environment this algorithm optimizes time required by the USV to travel from initial node to the goal node including the wait time.

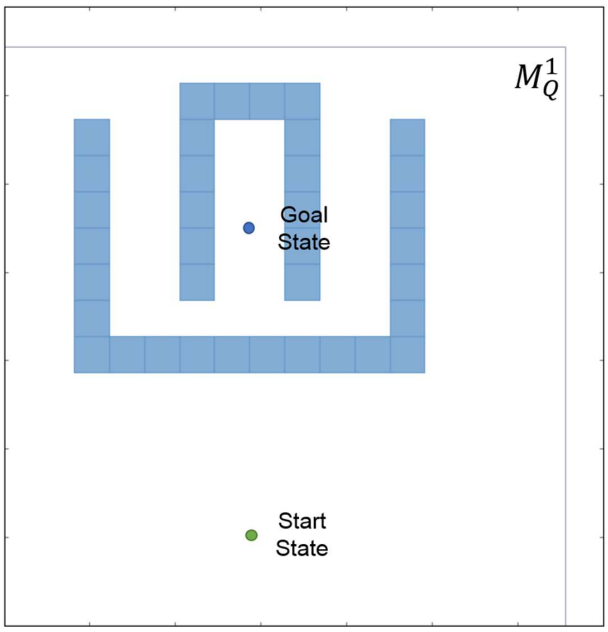
Figure 4.18, shows the layout of two quadtrees $\mathcal{M}_Q^1$ and $\mathcal{M}_Q^2$. Quadtree $\mathcal{M}_Q^1$

is valid from time $t = 0$ to $t = t_1$ and $\mathcal{M}_Q^2$ is valid from time $t = t_1$ to $t = t_{mission}$, where $t_{mission}$ is the upper bound on time for the USV to reach the goal state. The scenarios are designed such that there is a small channel which appears at time $t = t_1$ that USV can exploit to reach the goal. By varying the time t_1 we can see different types of plans computed by the developed path planner in Figure 4.19. In the Figure 4.19(a), the planner produces long serpentine path from start to goal state because the time t_1 is larger than the traversal time required by the USV to follow the long path at 10 m/s. If we reduce the value of t_1 (see Figure 4.19(b) and (c)), then the planner produces a path which involves waiting for duration t_w at the point immediately prior to collision before traversing to the goal.

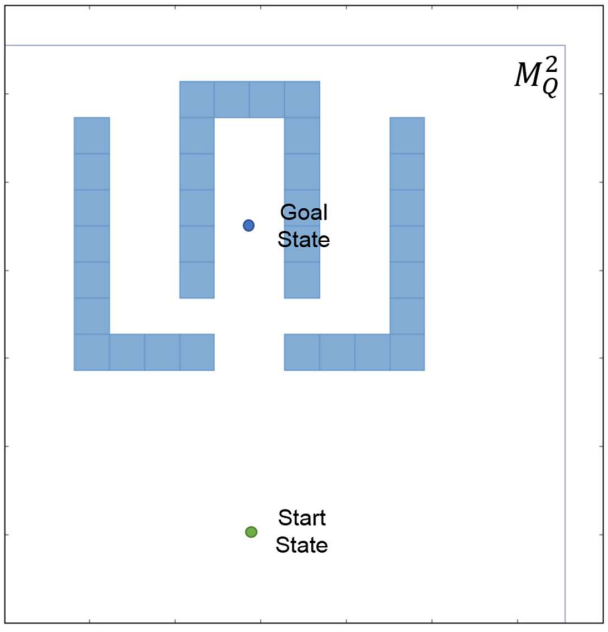
Similar to the previous scenario, Figure 4.20 shows the layout of the 3 quadtrees where the second quadtree opening the channel from north. The first and third quadtree are same as Figure 4.18 but with different time limits. Quadtree $\mathcal{M}_Q^1$ is valid from time $t = 0$ to $t = t_1$ $\mathcal{M}_Q^2$ is valid from time $t = t_1$ to $t = t_2$, and $\mathcal{M}_Q^2$ is valid from time $t = t_1$ to $t = t_{mission}$. Figure 4.21 shows the paths computed by the planner at different values of t_1 and t_2.

4.8 Results and Discussion

We compare the performance of the developed algorithm with the any-angle path planning algorithm Theta* [131]. The implementation of Theta* used for all the simulation experiments is taken from [162]. The scenario shown in Figure 4.23 (quadtree representation) is used to demonstrate the scaling of the developed tangent

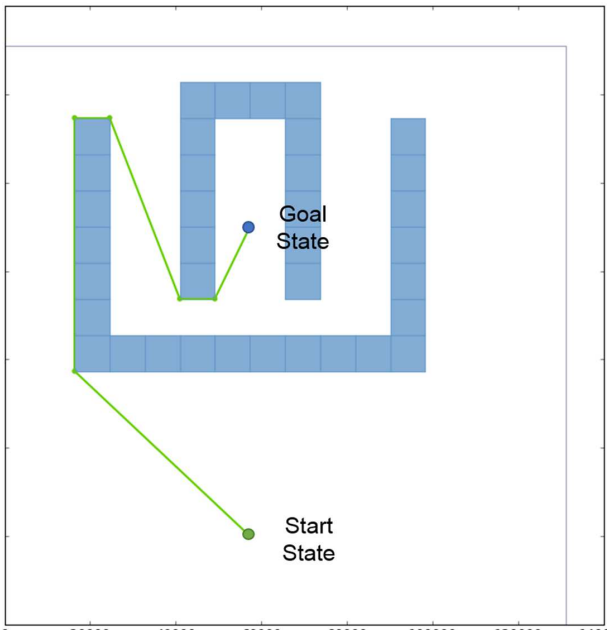


(a)

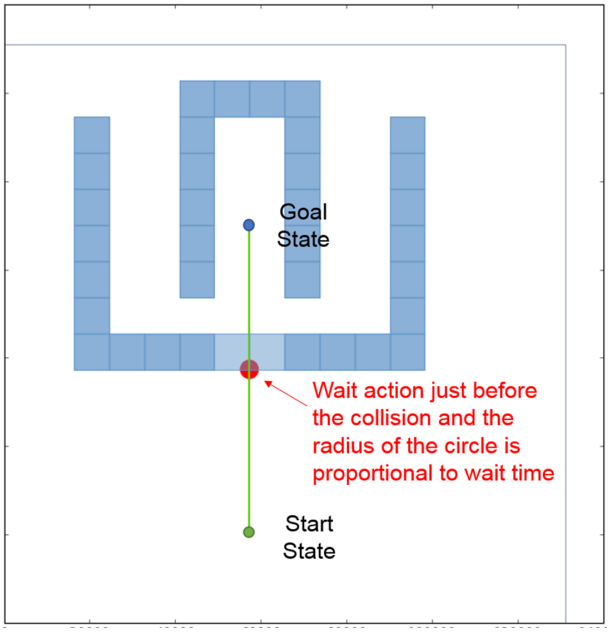


(b)

Figure 4.18: *Experimental scene with two quadtrees, start and goal nodes. (a) Quadtree* $\mathcal{M}_Q^1$*. (b) Quadtree* $\mathcal{M}_Q^2$

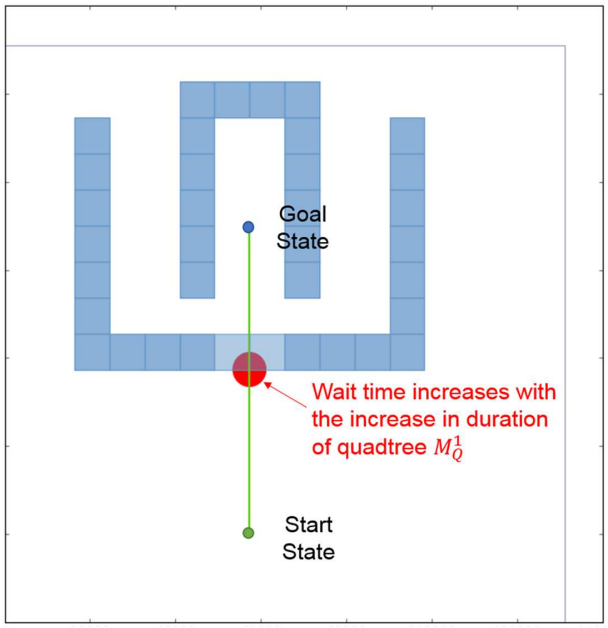


(a)



(b)

Figure 4.19: *Path computed by the planner at different values of time* t_1.

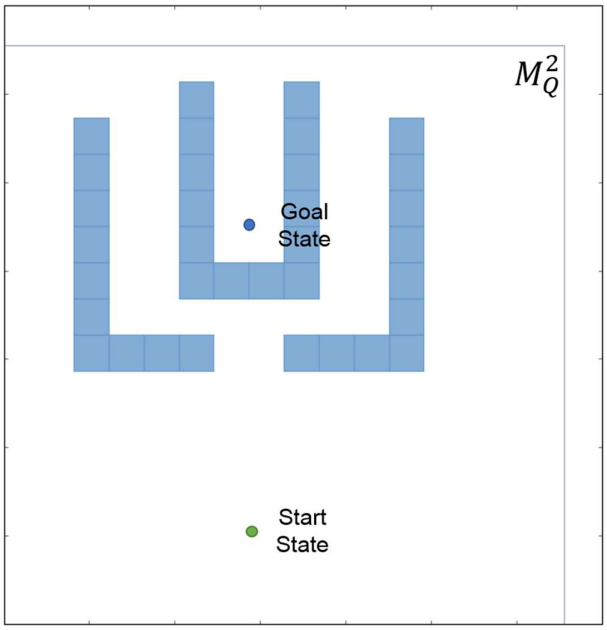


(c)

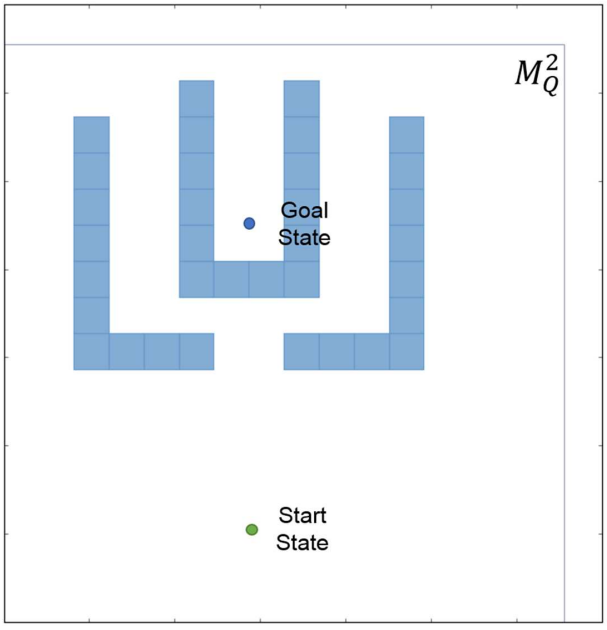
Figure 4.19: *Path computed by the planner at different values of time* t_1.

graph approach with improved heuristics against Theta*. Table 4.1 compares the computation time of Theta* and our approach. We can see that the computation time of Theta* drastically increases with the increase in pixels (or minimum grid size) used to represent the scene. On the other hand, the computation time of our approach TG+HEU, marginally increases primarily because of the Bresenham's collision test algorithm [161]. In other words, the developed tangent graph approach is resolution independent and does not depend on the grid size of the scene.

The simulation setup consisted of a randomly generated quadtree for the area of size 100 x 100 km (see Figure 4.22). The maximum depth of the quadtree was kept at $d_{max} = 13$ i.e. the finest resolution will be $100000/2^{13} = 12.21$ meters. The start and the goal nodes were kept constant at $\mathbf{n_I} = [2000, 2000]^T$ and $\mathbf{n_G} =$

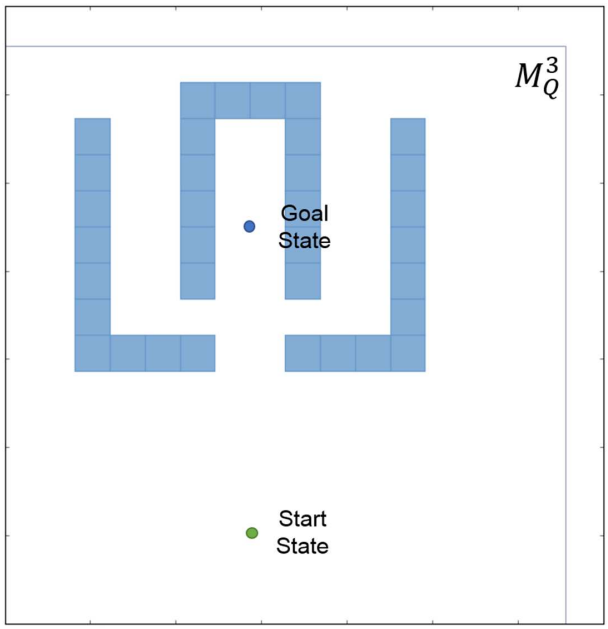


(a)



(b)

Figure 4.20: *Experimental scene with two quadtrees, start and goal nodes. (a) Quadtree $\mathcal{M}_Q^1$. (b) Quadtree $\mathcal{M}_Q^2$. (c) Quadtree $\mathcal{M}_Q^3$.*

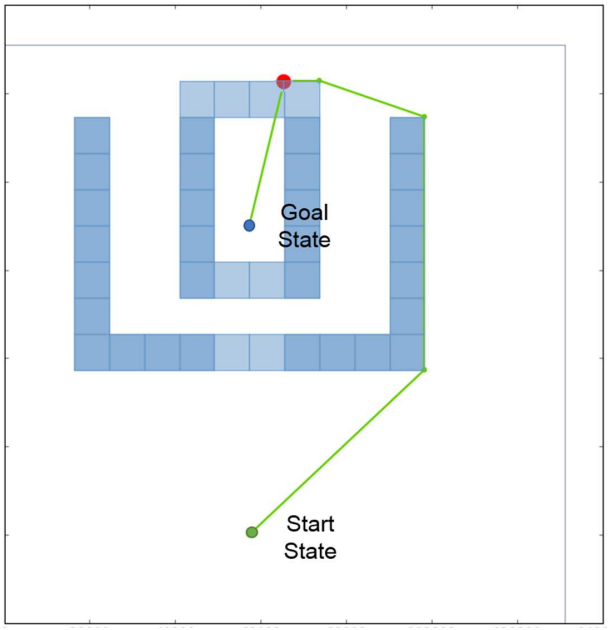


(c)

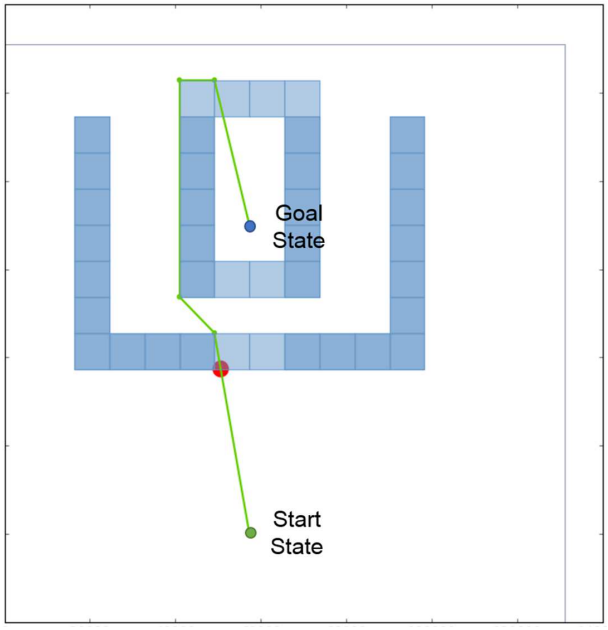
Figure 4.20: *Experimental scene with two quadtrees, start and goal nodes. (a) Quadtree $\mathcal{M}_Q^1$. (b) Quadtree $\mathcal{M}_Q^2$. (c) Quadtree $\mathcal{M}_Q^3$.*

Table 4.1: *Computational results for Theta* and tangent graph with the developed new heuristic (see Section 4.4) (TG+HEU) in the same scenario with different grid sizes.*

Map Size (pixels)	Computation Time (in sec)	
	Theta*	TG+HEU
1024 x 1024	1.87	0.19
2048 x 2048	16.69	0.22
4096 x 4096	90.784	0.24

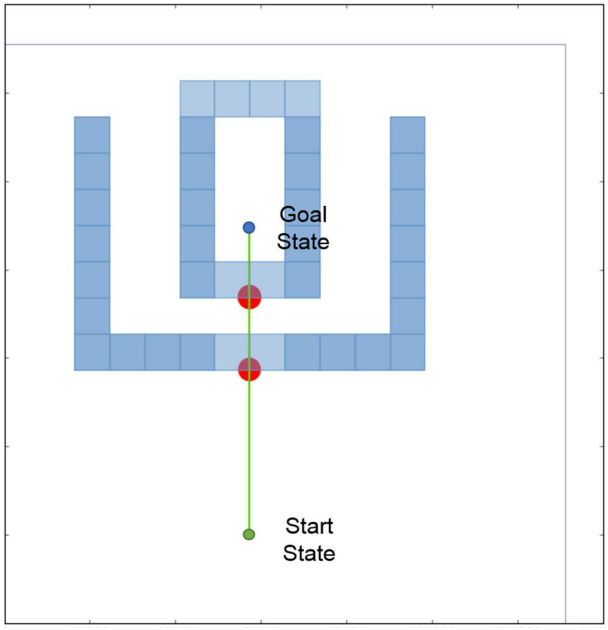


(a)



(b)

Figure 4.21: *Path computed by the planner at different values of time t_1 and t_2.*



(c)

Figure 4.21: *Path computed by the planner at different values of time* t_1 *and* t_2.

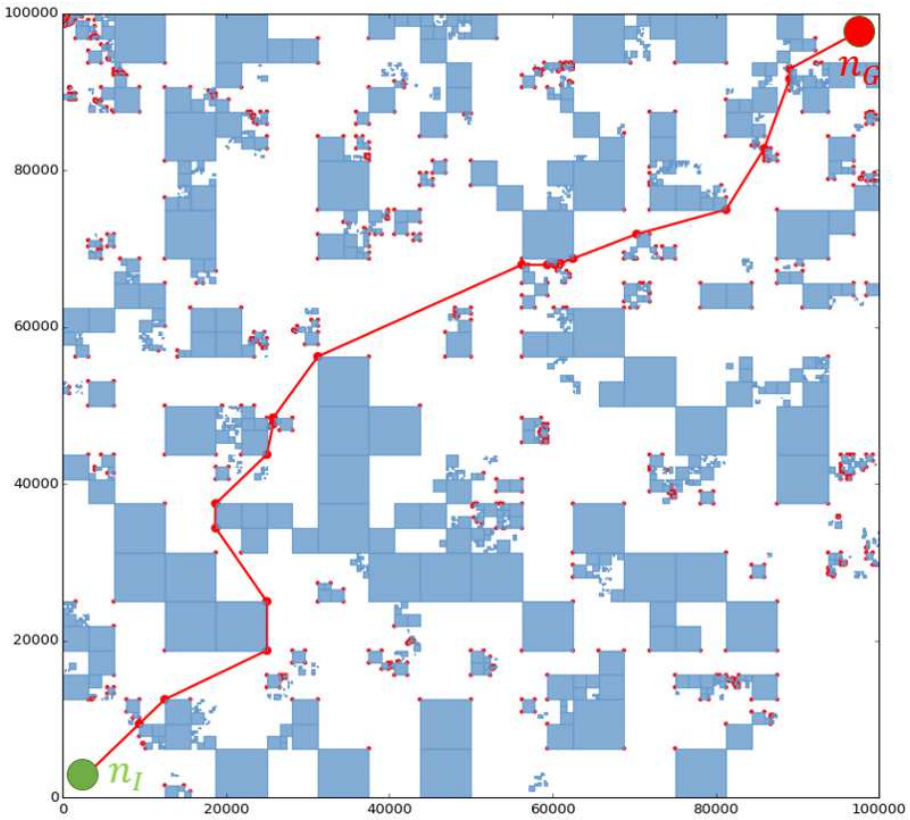


Figure 4.22: *Experimental setup and sample any-angle path from the start node* $\mathbf{n_I}$ *to the goal node* $\mathbf{n_G}$.

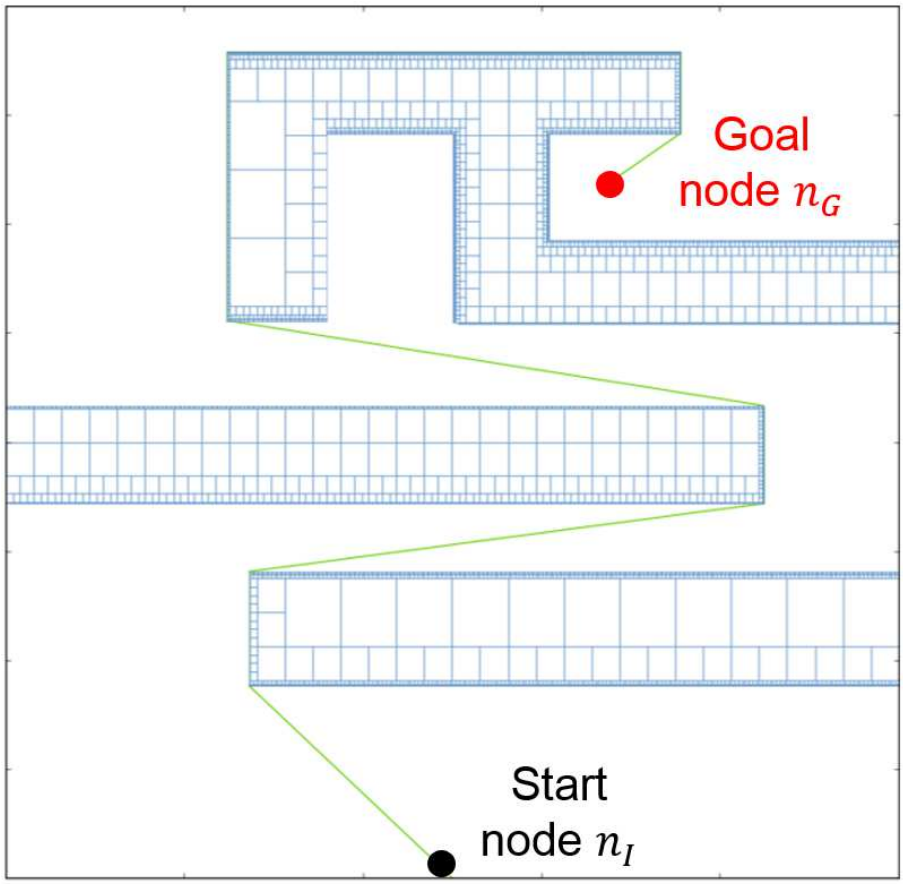


Figure 4.23: *Example scenario to compare the scaling between Theta* and our approach.*

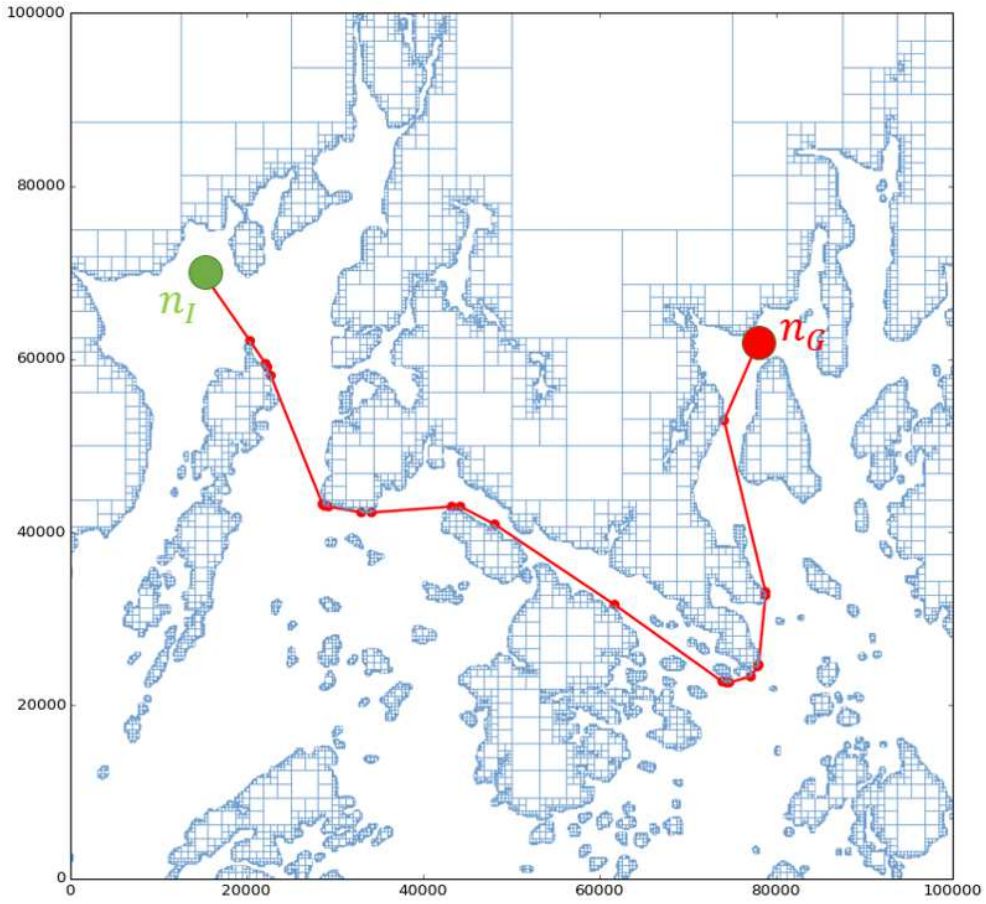


Figure 4.24: *Computed path on a real world scenario.*

$[98000, 98000]^T$ (in meters) respectively. The algorithm is written in Python 2.7 and computed on a Intel(R) Core(IM) i7-2600 CPU @ 3.4 GHz machine with 8GB RAM.

The results presented in Table 4.2 shows the computational performance of the developed approaches in randomly generated quadtree maps. The size of the quadtree map is varied from 5000 to 100000 leaf nodes by varying the depth of the quadtree $d_{max} = 10$ to 13 and the occupancy of the map. The tangent graph approach combined with the improved heuristics (TG+HEU) enhances the computational performance and reduces the number of expanded states as compared to

Table 4.2: *Comparison between different variants of developed visibility graphs-based algorithms on scenarios with a varying number of quadtree nodes. VG+ECU: Visibility graph with Euclidean distance as heuristic, TG+HEU: Tangent graph with the developed new heuristic (see Section 4.4), and FS+HEU: Focused search in tangent graph with the developed heuristic.*

# of Quad Nodes	# of Nodes in RVG	Computation Time (in sec)			% reduction in computation time by FS + HEU over VG + ECU	% increase in path length by FS + HEU
		ECU+ VG	TG + HEU	FS + HEU		
6446	919	21.60	10.17	4.92	77.21	0.00
6578	1272	19.79	10.98	3.11	84.30	0.00
7622	1044	16.58	5.78	2.98	82.05	0.00
8689	823	14.96	5.87	2.10	85.97	0.00
20605	1865	121.21	54.81	18.54	84.71	0.00
21075	2678	171.10	79.76	14.86	91.31	0.00
24951	3255	55.80	19.77	1.87	96.64	0.00
23567	1832	131.85	79.10	16.60	87.41	0.00
50534	9553	771.84	416.21	128.37	83.37	0.24
55180	9785	902.62	573.32	242.18	73.17	0.13
54961	11574	686.67	491.36	247.51	63.96	0.10
45082	6652	243.30	123.07	52.97	78.23	0.00
75578	22571	2440.61	1191.60	547.11	77.58	0.06
76507	19837	1209.50	948.21	476.37	60.61	0.00
75017	20427	856.65	475.82	182.92	78.65	0.33
74375	19340	1005.09	562.49	281.43	72.00	0.05
96480	14924	2638.41	1378.00	497.72	81.14	0.01
100657	16144	1339.26	1123.13	551.64	58.81	0.31
96630	14422	1106.41	852.66	368.99	66.65	0.00
101056	17055	1555.00	766.36	377.17	75.74	0.01

the visibility graph using the Euclidean distance as heuristics (VG+ECU). This is primarily due to a low branching factor of (TG) as it just examines the tangents of the islands while adding visible edges to the graph. The branching factor is further reduced by focusing the A* search on a tangent graph (FS) which restricts the search for the possible visible edges in the local vicinity and in the line-of-sight to the goal. In our experiments, the local vicinity of FS is determined by a circle of constant radius $r_{foc} = 10$ km. However, the improved computational performance of FS comes at the cost of loss of optimality and increased path length by a maximum of 0.33%.

The computation time and path lengths for Theta* on randomly generated scenarios with nodes fewer than 10000 quadtree nodes are comparable to that of FS+HEU. However, Theta* is unable to compute paths in several scenarios having more than 10000 quadtree leaf nodes (i.e. quadtrees of depth $d_{max} \geq 12$ which give maps that are 4096 x 4096 pixels in size) because the planner becomes memory intensive and the computer which we used to perform our simulation experiments cannot handle it. In the scenario above 10000 nodes where the Theta* is able to generate paths, we see a significant increase in computation time and path length.

The quadtree of the real marine environment (see Figure 4.24) is computed using the nautical chart data available from NOAA. Nautical chart data of a 100 x 100 km region is exported into shapefile (.shp) format. We have created a framework where we can read shapefiles of a region and generate the corresponding quadtree of a desired maximum depth (d_{max}). Using this framework we processed the land-regions from the shapefiles to extract the data represented in the form of polygons.

The quadtree of depth $d_{max} = 13$ is computed from the extracted polygons and outputted to a text file.

The extracted polygons are then processed to compute the quadtree of depth $d_{max} = 13$ and output it to a text file.

Figure 4.24 shows the computed path (in red) from the start node $\mathbf{n_I}$ to the goal node $\mathbf{n_G}$ in a real marine environment shown in Figure 4.1 . The size of the map is 100 x 100 km and the number of quadtree nodes is 66425. The total number of candidate nodes in the tangent graph is 2732. The number of nodes expanded by the path planner FS+HEU is 711 and the computation time is 12.19 seconds. The computation time for Theta* is 96.32 sec and the computed path is the same as that computed by FS+HEU.

4.9 Summary

This chapter presents an approach for computing paths on large marine domains. The approach presented in this chapter demonstrates that it is feasible to compute optimal paths using an A* search on visibility graphs defined over quadtrees. Experimental results indicate that optimal paths can be computed in a reasonable amount of time over a 100 km by 100 km area with a 10 m feature resolution. This was made feasible by developing methods to efficiently compute tangent edges in visibility graphs using quadtree data structure. There can be cases where the branching factor is large during the search over the visibility graph due to the large size of the region. To deal with these cases, we introduced the idea of focusing the search by

limiting the child nodes to be in certain regions of the workspace. Our results show that this idea speeds up the computation time significantly without compromising the quality of the path in a significant way. We also developed a method to estimate bounds on how far the computed path can be from the optimal path when methods for focusing the search are utilized for speeding up the computation.

The computational time of the developed algorithm will degrade with increasing obstacle density. With a high density of obstacles, most of the attempts to find long distance visibility nodes will result in collision. In complex and highly dense scenario (obstacle occupancy 70 - 80%), the developed planner may consume more computational time to compute optimal paths as compared to grid-based search algorithms. In the future, the computation time can be reduced by employing parallelization techniques and using computers (or cloud networks) with high processing power.

Chapter 5

Path Planning for Unmanned Vehicles Operating in Time-Varying Flow Fields

In this chapter[3], we present a A* based path planning algorithm with newly developed admissible heuristics for unmanned vehicles operating in time-varying flow fields.

5.1 Introduction

Many unmanned vehicles (also called robotic vehicles) interact with the underlying fluid medium in which they operate (see Figure 5.1). The medium may exhibit a significant fluid flow. For example, an aerial vehicle may encounter significant winds and an underwater vehicle may encounter strong water currents. The performance of the vehicle such as its maximum velocity and also the energy consumption per unit distance traveled is affected by the medium flow.

Usually, a global path planner is used for finding the shortest collision-free path to a specified goal location. The waypoints generated by the global planner are often followed using a feedback controller in order to compensate for environmental disturbances. The rejection of the disturbances via feedback controlled actuators may consume significant energy if the vehicle travels against a strong medium flow.

[3] The work in this chapter is derived from the published work in [163]

From the operational efficiency point of view, the vehicle should exploit the fluid flow instead of attempting to overcome it.

Weather forecast reports provide an estimate of the medium flow as a function of time. This information can be exploited to generate low-cost paths that utilize the flow to aid the motion of the vehicle. Such paths can save energy and hence be much lower in cost. Conserving energy has also an indirect benefit of extending the range of operation. This is especially important in missions where long term operation is desired. In many cases, the vehicle has a window of opportunity for completing a given mission and thus the mission manager can select the mission start time such that the vehicle experiences the most favorable medium flow during the operation.

Let us consider the following scenarios to understand the implications of the fluid flow on the path:

- *Scenario 1:* The vehicle takes a straight line path to the goal and does not account for the influence of the medium flow. It encounters a strong medium flow during the travel that impedes its motion. The vehicle then needs to use a significant energy to traverse the path against the flow, which increases the cost of the path.

- *Scenario 2:* The vehicle waits for the flow to become more favorable. Once the flow is in a favorable direction, the vehicle utilizes it to advance itself and thus uses less energy as it does not need to generate thrust to propel forward. The vehicle only dissipates energy when performing minor corrective

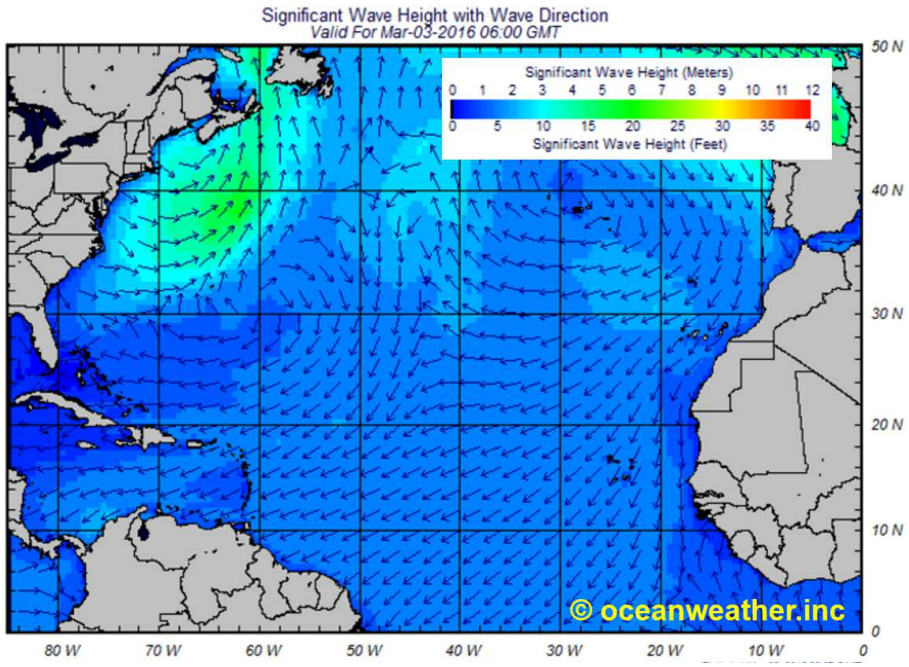


Figure 5.1: *Surface currents in the Atlantic ocean.*

actions to mitigate spatial disturbances. The path of the vehicle exploiting the medium flow may be curved and longer than a straight line path. The flow velocity is small compared to the velocity of the vehicle that uses its thrusters to propel forward, so the vehicle takes much longer time to reach the goal compared to the first scenario. Despite the longer path length and travel time, the vehicle consumes significantly less energy as compared to the first scenario. Therefore, the cost of travel is much lower. The decrease in the energy consumption means that the vehicle can do several more missions without the need for refueling.

This chapter deals with the global path planning under the influence of medium flows. The main contribution of this chapter is the incorporation of the influence of the flow field into the search for an optimal path.

Path planning for vehicles operating in the presence of flow fields has been previously studied in [164, 165]. Computing energy-efficient paths for a vehicle operating in a large environment with flow fields present several challenges such as:

- Utilizing a model of a time-varying flow field during path planning to predict the impact of the flow on the motion of the vehicle and hence compute paths that are energy-efficient as well as compliant with the vehicle's dynamics.
- Integrating the uncertainty in the prediction model of the flow field and the vehicles' spatio-temporal uncertainty arising due to its interaction with obstacles.
- Navigating around complex obstacles in the environment during the computation of an optimal collision-free path.

A majority of previous approaches attempted to address the aforementioned challenges separately. For example, graph search-based algorithms are efficient at computing paths in complex obstacle fields. Model predictive control-based techniques [166] are good at computing paths in the presence of flow fields. Stochastic mathematical programming-based techniques [167, 168, 169] are good at computing paths in the presence of uncertainty.

We believe that an integrated approach that can address all the aforementioned challenges consists of two steps. First, a discrete graph search technique is needed to compute a global path that not only avoids obstacles but also exploits the medium flow. Second, a stochastic mathematical programming-based techniques [149] or

model predictive control-based techniques [170] are needed to compute trajectories between intermediate goals lying on the computed global path.

In this chapter, we present a heuristic-based search technique for computing an energy-efficient global path for a vehicle moving in a flow field. The technique is relatively easy to implement, can incorporate intention models (if available) of dynamic obstacles, and enforce safety constraints [1, 38, 159]. The proposed algorithm can also determine the cost optimal start time of a given mission based on the model of the fluid flow. The computed path can be further locally altered by the mathematical programming or MPC-based techniques to account for spatial uncertainties.

Traditional distance and time-based admissible heuristics, used for estimating cost-to-go by discrete graph search algorithms, are not suitable for this domain because curved paths are needed to exploit available flows. Moreover, using the medium flow to propel the vehicle forward often requires a longer time to reach the goal. Hence, we have developed new admissible heuristics for estimating cost-to-go while taking into account the medium fluid flow.

5.2 Problem Formulation

5.2.1 Terminology

The continuous state space $\mathcal{X} = \mathcal{X}_\eta \times \mathcal{X}_\nu \times \mathcal{T}$ consists of states $\mathbf{x} = [\eta^T, \nu^T, t]^T \in \mathcal{X}$, where $\eta = [x, y, \psi]^T \in \mathcal{X}_\eta \subset \mathbb{R}^2 \times \mathbb{S}^1$ is the vehicle's pose, $\nu = [u, v, r]^T \in \mathcal{X}_\nu \subset \mathbb{R}^3$ is the vehicle's velocity consisting of the surge speed u, sway speed v, and angular

speed r about the z axis, and t is the time. The approximated lower dimensional, discrete 4D version of the continuous state space $\mathcal{X}$ is represented by $\mathcal{S}$, where each state $\mathbf{s} = [x, y, u, t]^T$ contains position, surge speed, and time variables.

The continuous, state-dependent motion primitive space of the vehicle is defined as $\mathcal{U} = \{\mathcal{U}_a, \mathbf{u_f}\} \subset \mathbb{R}^2 \times \mathbb{S}^1$. Here, $\mathcal{U}_a$ is the set of the vehicle's thrust-producing actions in which each action $\mathbf{u_a} = [u_d, \psi_d, \delta t]^T$ consists of the desired surge speed u_d, the desired heading ψ_d, and the execution time δt. The velocity vector of the vehicle at state $\mathbf{x}$ is given by $\mathbf{v^r}_\mathbf{x} = [u_d, \psi_d]^T$. A special free-flow action is defined as $\mathbf{u}_f = [u_m, \psi_m, \delta t]$ allows the vehicle to travel freely with the flowing medium along the current flow vector $\mathbf{v^m}_\mathbf{x}$ for the time interval δt (see Section 5.2.2). The discrete set of vehicle's thrust-producing actions is given by $\mathcal{U}_{a,d}$. Each discrete thrust-producing action of the vehicle is given by $\mathbf{u}_{a,d}$.

5.2.2 Medium Flow Model

In a real world scenario, it is difficult to have a continuous forecast of natural phenomena (e.g., wind, ocean currents, etc.). The available forecast is usually discrete and it is assumed to hold for a specific interval of time. On similar lines, we have modeled the medium flow in the environment to be discrete and is assumed to vary temporally but not spatially. In other words, the flowing medium for any discrete time t is constant for the time interval δt (see Figure 5.2). The simulation time interval of motion primitives $\mathcal{U}_d$ is kept to be the same as the discrete time interval δt of the medium flow model.

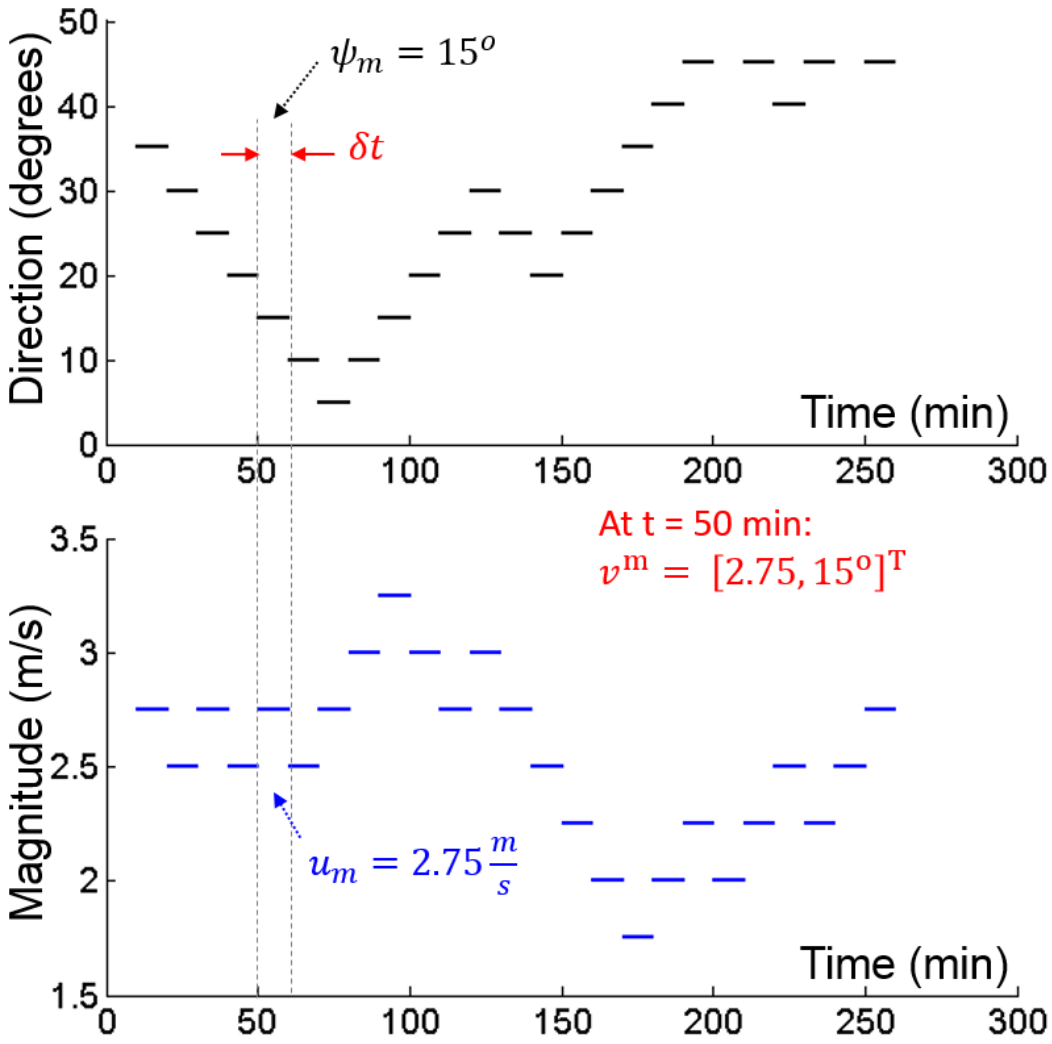


Figure 5.2: *Model of the flowing medium.*

The time-varying model m_f of the flowing medium outputs a velocity vector $\mathbf{v^m_s} = [u_m, \psi_m]^T$ at every state $\mathbf{s} \in \mathcal{S}$. Here, u_m is the magnitude and ψ_m is the direction of the flowing medium.

5.2.3 Motion Model

The motion of the vehicle in an environment with a flowing medium is dependent upon the direction and the magnitude of the flow. The transition of the vehicle

from the current state $\mathbf{s}$ to the next state $\mathbf{s}'$ is determined by the vehicle's thrust-producing action $\mathbf{u}_{a,d} \in \mathcal{U}_{a,d}$ and the velocity vector $\mathbf{v^m_s}$ of the flowing medium at the current state $\mathbf{s}$. We assume that the low level controller of the vehicle is capable of maintaining its heading along the direction ψ_d of the thrust-producing action $\mathbf{u}_{a,d}$.

Depending upon the medium flow, the forward velocity of the vehicle may be boosted or hindered. The magnitude of the forward velocity $|\mathbf{v}^f_\mathbf{s}|$ is determined by the vector sum of $\mathbf{v}^m_\mathbf{s}$ and $\mathbf{v}^r_\mathbf{s}$, with an assumption of the vehicle being a point mass (see Figure 5.3). The velocity of the vehicle $\mathbf{v}^r_\mathbf{s}$ can be resolved into two components: the magnitude of the component in the direction orthogonal to the desired direction $|\mathbf{v}^{r,O}_\mathbf{s}| = |\mathbf{v}^m_\mathbf{s}|sin(\psi_e)$ and the magnitude of the component along the desired direction $|\mathbf{v}^{r,D}_\mathbf{s}| = \sqrt{|\mathbf{v}^r_\mathbf{s}|^2 - |\mathbf{v}^{r,O}_\mathbf{s}|^2}$, where $\psi_e = \psi_d - \psi_m$, and ψ_m is the orientation of the flowing medium. Thus, the resultant forward velocity of the agent along the direction of the thrust-producing action $\mathbf{u}_{a,d}$ is given by $|\mathbf{v}^f_\mathbf{s}| = |\mathbf{v}^{r,D}_\mathbf{s}| + |\mathbf{v}^m_\mathbf{s}|cos(\psi_e)$. The position of the next state $\mathbf{s}'$ generated by the thrust-producing action $\mathbf{u}_{a,d}$ is determined by $[x', y']^T = [x, y]^T + |\mathbf{v}^f_\mathbf{s}| \cdot \delta t \cdot [cos(\psi_d), sin(\psi_d)]^T$. Similarly, the position of the next state $\mathbf{s}'$ generated by the free-flow $\mathbf{u}_{f,d}$ action is determined by $[x', y']^T = [x, y]^T + |\mathbf{v}^m_\mathbf{s}| \cdot \delta t \cdot [cos(\psi_m), sin(\psi_m)]^T$.

5.2.4 Cost Model

Let the cost of executing a vehicle's thrust-producing action per unit time be given by C^t_a and the cost of the special free-flow action per unit time be given by C^t_m, where $C^t_m < C^t_a$. The values of C^t_m and C^t_a are constant and provided by the user.

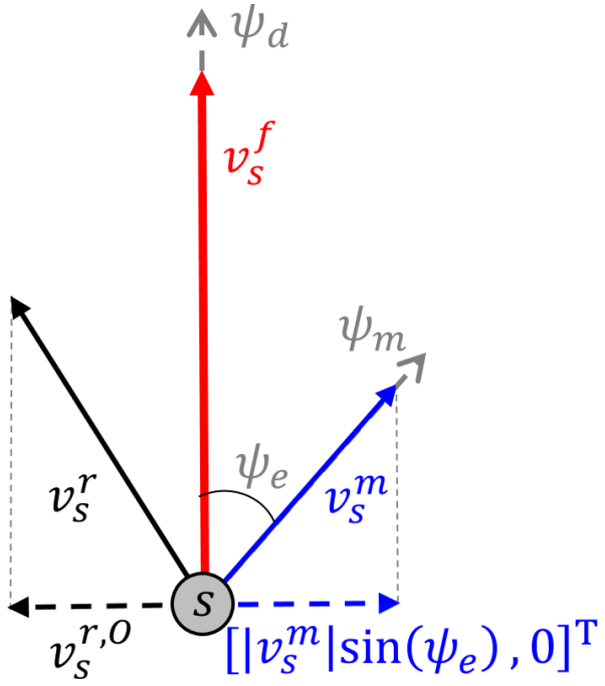


Figure 5.3: *Computation of vehicle's forward velocity under medium flow.*

Let $c(\mathbf{s}, \mathbf{s}')$ denote the traversal cost from the state $\mathbf{s}$ to state $\mathbf{s}'$. The traversal cost $c(\mathbf{s}, \mathbf{s}')$ equals $C_a^t \delta t$ if the vehicle arrives at state $\mathbf{s}'$ by using the thrust-producing action $\mathbf{u}_{a,d} \in \mathcal{U}_{a,d}$. Similarly, the traversal cost $c(\mathbf{s}, \mathbf{s}')$ equals $C_m^t \delta t$ if the vehicle arrives at state $\mathbf{s}'$ by using the free-flowing action $\mathbf{u}_{f,d}$. Finally, let the optimal cost of the computed path τ from the initial state $\mathbf{s_I}$ to the goal state $\mathbf{s_G}$ at start time t_{start} be denoted by $c^*_{start}(\mathbf{s_I}, \mathbf{s_G})$ (see Section 5.3.3). Here t_{start} is the time when the vehicle starts the mission, i.e., leaves from the initial state $\mathbf{s_I}$ and proceed towards the goal state $\mathbf{s_G}$.

5.2.5 Problem Statement

We are interested in designing an energy-efficient path planning algorithm for computation of collision-free paths between the initial and the goal states of a vehicle

operating in an environment with a flowing medium. The developed planner searches for a path that minimizes the energy cost by exploiting the medium flow.

Given,

- the discrete state space $\mathcal{S}$ of the vehicle,
- the initial $\mathbf{s_I}$ and the goal $\mathbf{s_G}$ states of the vehicle,
- the discrete model of the medium flow m_f,
- the cost C_a^t of the vehicle's thrust-producing action per unit time and the cost C_m^t of the free-flowing action per unit time,
- the map of the environment with the geometric regions occupied by static obstacles $\mathcal{O}_s = \bigcup_{k=1}^{K} o_{s,k} \subset \mathbb{R}^2$, and
- the maximum time duration $t_{mission}$ in which the vehicle should complete the current mission and reach the goal $\mathbf{s}_G$.

Compute:

- The start time of the mission $t_{start} < t_{mission}$ that minimizes the cost incurred by the vehicle to travel from $\mathbf{s_I}$ to $\mathbf{s_G}$.
- A collision-free, dynamically feasible trajectory $\tau : [t_{start}, t_{finish}] \rightarrow \mathcal{S}$ such that $\tau(t_{start}) = \mathbf{s_I}$, $\tau(t_{finish}) = \mathbf{s_G}$ and its travel cost is minimized. The value of t_{finish} should not exceed $t_{mission}$ and $\mathbf{s}(t) \notin \mathcal{O}_s$

  Each state $\mathbf{s}(t)$ along τ belongs to the free state space.

In this chapter, we assume that the actuators of the vehicle are able to overcome the medium flow and the velocity of the agent $\mathbf{v^r_s}$ is greater than the medium velocity $\mathbf{v^m_s}$.

5.3 Approach

5.3.1 Overview

The deliberative path planner described in Section 5.3.2 searches in a discrete 4D state space for a collision free, lattice-based path $\tau : [0, t_{finish}] \rightarrow \mathcal{S}$ from a given initial state $\mathbf{s_I}$ to a goal state $\mathbf{s_G}$. The path is optimized not only with respect to its travel cost but also with respect to the vehicle's start time t_{start} (see Section 5.3.3).

The medium flow forecast may have uncertainty associated with it. The A* algorithm does not handle this uncertainty. This uncertainty can be handled by refining the path by using forward value iteration-based stochastic dynamic programming in the vicinity of the computed path [42]. This post-processing can refine the paths to reduce the probability of collision with obstacles due to the uncertainty in the medium flow by optimizing the expected costs of paths. This step is outside the scope of this chapter and will not be discussed further.

5.3.2 Path Planning

The global path planner is designed based on the lattice-based A* heuristic search [33]. The search for a path τ with the minimum cost is performed by expanding states in the least-cost fashion according to the cost function $f(\mathbf{s}) = g(\mathbf{s}) + h(\mathbf{s})$,

where $g(\mathbf{s})$ is the cost-to-come at state $\mathbf{s}$ from the initial state $\mathbf{s_I}$, and $h(\mathbf{s})$ is the cost-to-go from the state $\mathbf{s}$ to the goal state $\mathbf{s_G}$. The cost-to-come $g(\mathbf{s})$ is computed by summing a traversal cost of each action (see Section 5.2.4) executed to reach the current state $\mathbf{s}$ from the initial state $\mathbf{s_I}$. In Section 5.4, we compute three different types of the cost-to-go (h-cost) to be used in the cost function described above. During the search, the neighboring state $\mathbf{s}$ is determined by the motion model described in Section 5.2.3. We have defined a desired goal state region S_G in close proximity around the goal state $\mathbf{s_G}$. The search is terminated when any of the expanded states $\mathbf{s}$ lies in S_G.

5.3.3 Start Time Optimization

The optimal cost $c^*_{start}(\mathbf{s_I}, \mathbf{s_G})$ of the path $\tau : [t_{start}, t_{finish}] \rightarrow \mathcal{S}$ computed by the path planner is highly dependent on the medium flow encountered by the vehicle and the start time t_{start}. In some situations, it is beneficial for the vehicle to wait at the initial location $\mathbf{s_I}$ and start its journey only when the medium flow becomes favorable. The maximum time $t_{mission}$ the vehicle is allowed to wait and complete its mission is predefined by the user.

We find the resolution-optimal start time t_{start} of the path between $\mathbf{s_I}$ to $\mathbf{s_G}$ by adaptively sampling the time interval $\{0, t_{mission}\}$. We iteratively call the deliberative path planner to compute the cost $c^*_{start}(\mathbf{s_I}, \mathbf{s_G})$ of a path for different values of t_{start}. The bounding constraints for t_{start} are given by $0 < t_{finish} \leq t_{mission}$, where $t_{finish} = t_{start} + t_{execution}$ and $t_{execution}$ is time taken by the vehicle to execute the

path. The sampling method begins with large interval steps and adaptively reduces the time step in the promising regions.

5.4 Design of Heuristics

5.4.1 Heuristic #1

Let $\mathbf{s}$ be the current state. We are interested in estimating the cost to reach the goal state $\mathbf{s}_G$ from the current state $\mathbf{s}$. Let $c^*(\mathbf{s}, \mathbf{s}_G)$ be the optimal cost of reaching $\mathbf{s}_G$ from $\mathbf{s}$. We will refer to this cost as optimal cost-to-go. Let heuristic $h(\mathbf{s})$ be a function that provides an estimate of $c^*(\mathbf{s}, \mathbf{s}_G)$. $h(\mathbf{s})$ will be called *admissible heuristic* if, $c^*(\mathbf{s}, \mathbf{s}_G) \geq h(\mathbf{s})$.

A simple way to compute $h(\mathbf{s})$ would be to assume that the flow will be most favorable during the the vehicle operation. This will enable us to achieve the smallest possible cost per unit distance traveled. Hence estimate of cost-to-go $h(\mathbf{s})$ cannot exceed the actual optimal cost to goal state $\mathbf{s}_G$.

Let $dist(\mathbf{s}, \mathbf{s_G})$ be the Euclidean distance between states $\mathbf{s}$ and $\mathbf{s_G}$. If the flow is assumed to be at maximum velocity ${\mathbf{v^m}}_{max}$ and directly flowing towards the goal, then the total cost of travel using free flow is given by Equation 5.1.

$$C^1 = \frac{dist(\mathbf{s}, \mathbf{s_G}) \cdot C^t_m}{|\mathbf{v^m}|_{max}} \tag{5.1}$$

If the vehicle uses its actuators and travels at maximum vehicle velocity ${\mathbf{v^r}}_{max}$ in addition to taking the advantage of the flow, then the total cost of travel is given

by Equation 5.2.

$$C^2 = \frac{dist(\mathbf{s}, \mathbf{s_G}) \cdot C_a^t}{|\mathbf{v^m}|_{max} + |\mathbf{v^r}|_{max}} \tag{5.2}$$

Minimum of the two estimates C^1 and C^2 can be determined to calculate $h(\mathbf{s})$.

$$h(\mathbf{s}) = min(C^1, C^2) \tag{5.3}$$

5.4.2 Heuristic #2

Although admissible, the heuristic presented in Section 5.4.1 significantly underestimates the cost, so we have designed a better heuristic that utilizes the medium flow information. In order to utilize the flow information, we need to first determine the relevant time window. We do this by first estimating the upper bound t_{bound} on the time associated with the optimal path. Any medium flow available at time greater than t_{bound} will not be available during the execution of the path.

We compute the cost $C_{straight}$ incurred by the vehicle to travel in a straight path to the goal state $\mathbf{s_G}$ using its thrust-producing action in the presence of medium flow. $C_{straight}$ is the upper bound on the optimal cost. Let C' be the cost of any arbitrary path to the goal state $\mathbf{s_G}$, and can be represented by $C' = t_f \cdot C_m^t + t_a \cdot C_a^t$, where t_f is the total time consumed by the free flow action and t_a is the total time consumed by the vehicle's thrust-producing action. We are only interested in paths $C_{straight} \geq t_f \cdot C_m^t + t_a \cdot C_a^t$. We can compute t_{bound} by maximizing the objective function $t_f + t_a$, where $t_f = (C_{straight} - t_a \cdot C_a^t)/C_m^t$. We assume that $C_m^t < C_a^t$. Hence, we can maximize total time by selecting $t_a = 0$ and $t_f = C_{straight}/C_m^t$. Thus,

the upper bound on time is given by $t_{bound} = C_{straight}/C_m^t$. The value of $C_{straight}$ will change with the scenarios having different medium flows.

As mentioned in Section 5.2.2, the estimated flow conditions are available as an ordered sequence. Each flow condition holds for a time interval of δt. In each time interval δt, we have the direction and the magnitude of the flowing medium in terms of velocity vector $\mathbf{v^m_s}$.

We can view each flow condition as a performance altering condition that lasts for a duration of δt. We are interested in exploiting these conditions that lower the cost of travel. For each flow condition that we plan to utilize, we need to make a decision to either execute the free-flow action or use the thrust-producing action. In computation of the heuristic cost $h(\mathbf{s})$, we select the action that has the lower cost incurred per unit distance advancement towards the goal.

The cost incurred per unit projected distance traveled using the free-flowing action $\mathbf{u}_{f,d}$ is calculated using Equation 5.4.

$$C_f^l = \frac{C_m^t}{|\mathbf{v^m_s} \cdot |cos(\psi_e)} \tag{5.4}$$

The cost incurred per unit projected distance traveled using the vehicle's thrust-producing action $\mathbf{u}_{a,d} \in \mathcal{U}_{a,d}$ is calculated using Equation 5.5.

$$C_a^l = \frac{C_a^t}{|\mathbf{v^r_s}| + |\mathbf{v^m_s}| \cdot cos(\psi_e)} \tag{5.5}$$

In Equations 5.4 and 5.5, the angle ψ_e is the angle between the desired direction ψ_g to the goal state $\mathbf{s_G}$ from the current state $\mathbf{s}$ and the direction of the flowing medium ψ_m (see Figure 5.4). We can choose the appropriate action for each discrete

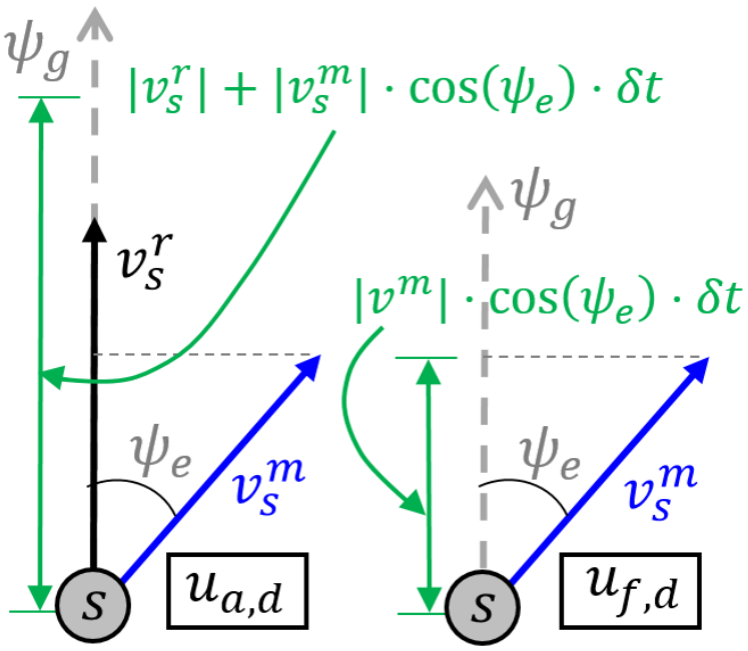


Figure 5.4: *Calculation of heuristic #2.*

time interval δt starting from the current time t to t_{bound}. The selected actions for each discrete time interval $\delta t \in \{t, t_{bound}\}$ are sorted and stored into the priority queue E_O according to the cost incurred per unit distance advancement towards the goal with the least cost action on the top. The actions are sequentially popped out of the priority queue E_O and are integrated for time interval δt, until the summation of the projected distance traveled by all actions is equal to the projected distance to the goal $\mathbf{s_G}$ from the state $\mathbf{s}$.

This heuristic uses actions that have the lowest per unit length cost towards the goal from the available time window. It selects actions without requiring them to be contiguous in time. The optimal path will have either the same action as used by the heuristic or will be forced to use actions that have higher per unit length cost towards the goal. Therefore, it is not possible for the optimal path to exceed

the cost estimated by this heuristic. Therefore, this heuristic is admissible.

5.4.3 Heuristic #3

In Heuristic #2 described above, each selected action is assigned a cost-to-go based on the cost incurred per unit projected distance traveled towards the goal. This is a tight lower bound on cost for thrust producing actions. However, when the free flow action is used, unless $\psi_e = 0$, the vehicle does not go directly towards the goal (see Figure 5.5). If free flow conditions do not exist within the time bound that can provide ψ_e of opposite sign, a thrust-producing action is needed to bring the vehicle towards the goal. If using the thrust-producing action becomes necessary in conjunction with the free flow action, then the lower bound computed on the cost in heuristic #2 significantly underestimates the cost and can lead to expansion of a large number of states. We have devised an improvement over heuristic #2, by increasing the per unit length cost associated with free flow actions to account for use of the thrust producing actions. Let us consider a free flow action shown in Figure 5.5. Without the loss of generality, let us assume that ψ_e is positive and no free flow action is available with negative value of ψ_e until time t_{bound}. We will, therefore, have to use a thrust-producing action to bring the vehicle towards the goal.

In the Figure 5.5, the free flowing action $\mathbf{u}_{f,d}(s')$ along the flowing medium with velocity $\mathbf{v^m}$ makes an angle ψ_e with the desired direction of motion . Now, the corrective distance the vehicle has to travel to get back to the desired path is

Algorithm 3 ComputeHeuristic2$(\mathbf{s}, t, \psi_g, t_{bound}, m_f)$

Input: The current node $\mathbf{s}$, current time of arrival t, the desired direction ψ_g from the current state $\mathbf{s}$ to the goal state $\mathbf{s_G}$, the maximum bound on travel time t_{bound}, and the model of medium flow m_f.

Output: An estimated cost-to-go $h(\mathbf{s})$ from current state $\mathbf{s}$ to goal state $\mathbf{s}_G$.

Let $t_i = t$ be the forward simulation time and δt be the simulation time step.

Let $\mathbf{v}_i$ be the velocity vector along the desired direction achieved by executing action $\mathbf{u}_i$ at time $t_i \in \{t, t_{bound}\}$.

Let E_O be a priority queue containing selected actions $\mathbf{u}_i$ at each discrete time $t_i \in \{t, t_{bound}\}$

while $t_i \leq t_{bound}$ **do**

    Let the current velocity vector of the medium flow at state $\mathbf{s}_i$ be denoted by $\mathbf{v^m}_{\mathbf{s}_i}$

    Cost incurred by per unit length advanced towards the goal while executing free-flow action $\mathbf{u}_{f,d}$ and thrust-producing action $\mathbf{u}_{a,d}$ are given by Equation 5.4 and 5.5 and denoted as C_f^l and C_a^l respectively.

    if $C_f^l < C_a^l$ **then**

        $\mathbf{v}_i = |\mathbf{v^m}_{\mathbf{s}_i}| cos(\psi_e)$ and $C_i^l = C_f^l$

    else

        $\mathbf{v}_i = |\mathbf{v^r}_{\mathbf{s}_i}| + |\mathbf{v^m}_{\mathbf{s}_i}| cos(\psi_e)$ and $C_i^l = C_a^l$

    end if

    Insert vector $[\mathbf{v}_i, C_i^l]^T$ into/in E_O

    $t_i = t_i + \delta t$

end while

Let $d_G = dist(\mathbf{s}, \mathbf{s_G})$ be the distance of the state $\mathbf{s}$ from the goal state $\mathbf{s_G}$.

$d_{travel} = 0$ and $C_{incur} = 0$

while E_O not empty **do**

    $[\mathbf{v}_i, C_i^l] \leftarrow E_O.First()$

    $d_{travel} = d_{travel} + |\mathbf{v}_i| \cdot \delta t$

    $C_{incur} = C_{incur} + C_i^l \cdot d_{travel}$

    if $d_{travel} \geq d_G \cdot cos(\psi_g)$ **then**

        $h(\mathbf{s}) \leftarrow C_{incur}$

        return $h(\mathbf{s})$

    end if

end while

return $h(\mathbf{s}) = \infty$ (not enough time to reach the goal state, thus the node $\mathbf{s}'$ does not lie on optimal path τ^*.

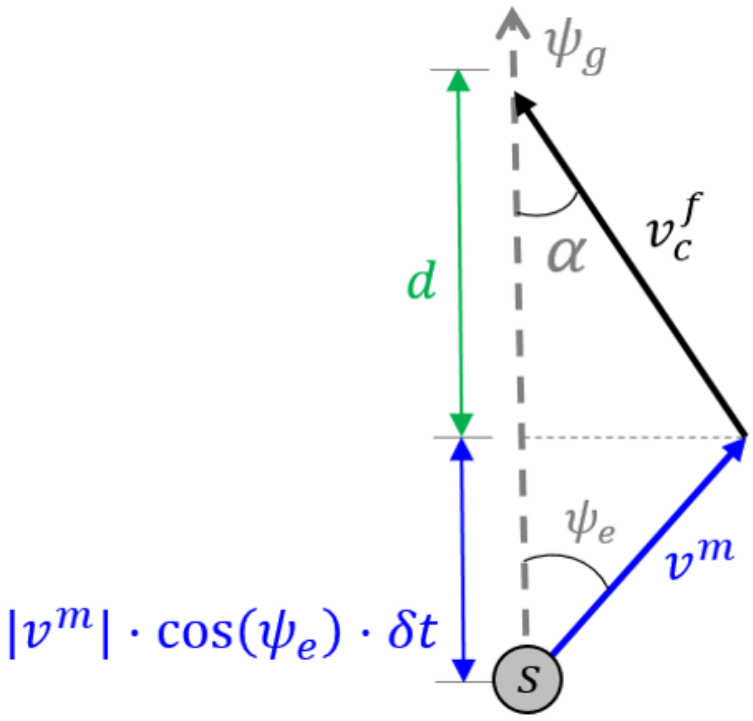


Figure 5.5: *Calculation of additional compensation cost-to-go for free-flowing action* $\mathbf{u}_{f,d}(\mathbf{s})$.

given by $d_c = \sqrt{d^2 + (|\mathbf{v^m}| \cdot sin(\psi_e) \cdot \delta t)^2}$, where d is the projected distance traveled along the desired direction towards the goal.

To compute the lower bound on the cost, we want to use the fastest possible velocity for the vehicle. Let us assume that there will be flow available that will provide the maximum possible assistance to the vehicle. We will only apply this correction if there is no flow available with negative ψ_e. The best that we can hope for is that the flow is going along ψ_g as shown in Figure 5.5. Let us assume that the magnitude of the flow velocity is the maximum possible $|\mathbf{v^m}|_{max}$ within the available time window. Under these conditions the vehicle's forward velocity while performing the corrective action can be calculated as:

$$\begin{gathered} |\mathbf{v^f}_c| = |\mathbf{v^r}| + |\mathbf{v^m}|_{max} \cdot cos(\alpha), \\ \text{where } \alpha = \tan^{-1}[(|\mathbf{v^m}|_{max} \cdot sin(\psi_e) \cdot \delta t)/d] \end{gathered} \tag{5.6}$$

The time taken to perform the corrective action can be calculated as $t_c = d_c/|\mathbf{v^f}_c|$. Thus, the cost incurred per unit distance advanced towards the goal by using a combination of free-flow and thrust-producing action can be given by:

$$C^l_a = \frac{t_c \cdot C^t_a + C^t_m \cdot \delta t}{d + |\mathbf{v^m}| \cdot cos(\psi_e) \cdot \delta t}. \tag{5.7}$$

In order to compute the lower bound on the cost given in Equation 5.7, we need to select d so that the C^l is minimized. Solving the above function analytically is not possible and requires application of numerical techniques. Please note that this function depends on $|\mathbf{v^m}|$, $|\mathbf{v^m}|_{max}$, and ψ_e. We have optimized the above function for different combinations of these values using off-line computation. A meta-model (e.g., lookup table) has been developed that allows us to quickly access the lower

bound on the value of C^l for free flow actions. Please note that these optimized values of C^l are usually higher compared to the values provided by Equation 5.4.

If there is no free flow available within the available time window with the opposite sign of ψ_e that will take the vehicle back towards the goal, then we use the modified value of C^l in line 6 of Alg. 3. The use of this value is expected to produce a much better estimate of the cost-to-go and hence improve the computational performance.

5.5 Results and Discussion

5.5.1 Simulation Setup

We chose an action set comprising of seventeen actions, out of which 16 actions are thrust-producing $\mathbf{u}_{a,d} = [u_d, \psi_d, \delta t]$ having desired direction ψ_d equally spaced from 0 to 360 degrees and constant surge speed of 10 m/s with respect to the medium. We assume that the maximum magnitude of the medium flow is 6 m/s. We assigned the cost of executing each thrust-producing action to be $C^t_a = 6$ per minute while the cost of executing a free-flow action to be $C^t_m = 1.2$ per minute. We discretized the time with 10 min intervals, i.e., $\delta t = 10$ min, which means that a motion primitive is executed for δt duration before another motion primitive can be commanded. This is mainly because the weather predictions available in practice are seldom more frequent than $\delta t = 10$ min. The medium flow model as described in Section 5.2.2 has a discrete magnitude and direction profile. The designed scenarios used for performance evaluation of the developed heuristics (see Section 5.4) have medium

profiles that either vary in magnitude, direction or both.

The first set of scenarios uses medium with a constant magnitude profile. The magnitude of the medium flow is held constant at 6 m/s. Specific test cases are:

- Constant flow directions along 30^o and 90^o.
- Rotating medium flow at the rate of 0.1^o per minute with initial direction of 330^o and rotating medium flow of 0.2^o per minute with initial direction of 310^o.

The second set of scenarios uses medium with a randomly generated magnitude profile. The magnitudes are randomly generated in a range of 0 to 6 m/s with the rate of change of 0.5 m/s between two consecutive discrete time steps. Specific test cases are:

- Constant flow direction of 30^o and 90^o
- Randomly generated direction profile changes by 5^o in each discrete time step. We use two scenarios having initial medium flow directions of 5^o and 45^o
- Rotating medium flow at the rate of 0.1^o per minute with initial direction of 330^o and rotating medium flow of 0.2^o per minute with initial direction of 310^o

5.5.2 Comparison of Heuristics

The results presented in Table 5.1(a) compares the performance of all the three heuristics in test scenarios having medium flow of constant magnitude. The reduction in number of states by heuristic #2 with respect to heuristic #1 is lower for

Table 5.1: *Comparison of the number of states expanded by the path planner using the heuristic #2 and #3 with respect to the heuristic #1 in scenario having medium flows with (a) constant magnitude and (b) random magnitude.*

Scenarios with Constant Magnitude of Flow		# States Expanded (in order of 1000s)			% Reduction in States Expanded	
		# 1	# 2	# 3	# 2 with # 1	# 3 with # 1
Constant Direction	30°	196.32	71.32	1.51	63.67	99.23
	170°	459.86	9.04	8.80	98.03	98.09
Rotating Direction	Rate 1°/s	48.86	15.59	5.40	68.10	88.96
	Rate 2°/s	58.89	14.65	5.05	75.13	91.43

(a)

Scenarios with Random Magnitude		# Avg. States Expanded (in order of 1000s)			% Reduction in States Expanded			
					# 2 with # 1		# 3 with # 1	
		# 1	# 2	# 3	Mean	Std Dev	Mean	Std Dev
Constant Direction	30°	186.62	56.59	3.83	59.33	23.13	92.17	15.45
	170°	363.71	17.73	7.14	95.13	13.84	98.04	12.83
Random Direction	Init at 0°	323.08	53.75	32.37	81.03	19.40	91.19	11.62
	Init at 45°	270.45	48.63	37.23	87.94	12.33	89.61	11.28
Rotating Direction	Rate 1°/s	336.75	52.01	24.13	84.54	16.27	92.84	12.68
	Rate 2°/s	215.75	28.45	13.32	86.81	17.28	93.82	16.85

(b)

scenarios with constant and rotating medium flow with lower values of ψ_e (i.e., favorable medium flows) because it does not account for the cost of thrust-producing action to reach the goal after executing the free-flow action. Heuristics #3 corrects for this problem.

The results presented in Table 5.1(b) shows the performance of all the three heuristics in test scenarios having medium flows of random magnitude. The performance of heuristic #3 is lower in the scenario having random direction, because in this case it uses best case correction for the deviations caused by the free flow actions.

Table 5.2 shows the ratio of the cost C_1 to C_2, where C_1 is the cost incurred by the vehicle while using the shortest distance path, and C_2 is the cost incurred by the vehicle while using the developed path planner and the vehicle starts its mission at time $t_{start} = 0$. Higher the ratio of C_1/C_2, the vehicle saves more energy by using the developed planner as compared to the shortest distance-based path planner. The results in Table 5.2 are computed by randomly generating 100 scenarios for each occupancy value ranging from 10-40%. The results show that with the increase in occupancy of the scenario, the performance of the developed path planner degrades. The primary reason for this decline is the lack of free space for executing long free-flowing actions. Secondly, the vehicle has to execute its thrust-producing action to overcome large number of obstacles in the environment.

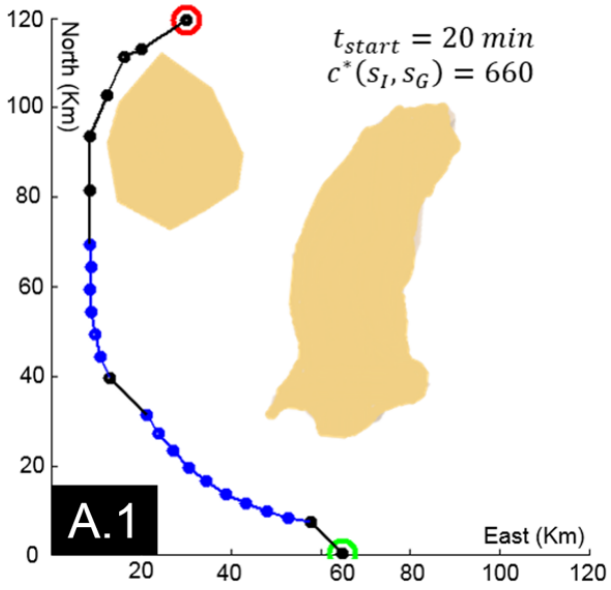
Table 5.2: *Performance of the developed energy-efficient planner in randomly generated scenarios with varying occupancy. Cost C_1 is the cost incurred while using the shortest distance path planner and cost C_2 is the cost incurred while using the developed path planner at time $t_{start} = 0$.*

Occupancy (in %)	Ratio of C_1 and C_2	
	Mean	Std Dev
10.00	1.94	0.15
20.00	1.72	0.21
30.00	1.42	0.30
40.00	1.12	0.37

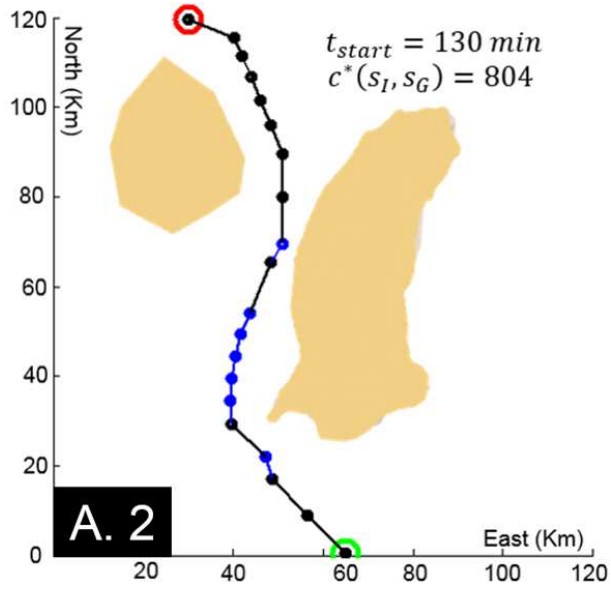
5.5.3 Results on Example Scenarios

The results presented in Figure 5.6 show the paths generated by the deliberative path planner in the scenario A at different start times of the mission. The scenario presented in the figure has a medium flow of constant magnitude, rotating clockwise at the rate of 0.2^o/min. The initial direction of the medium flow at $t_{start} = 0$ is pointing towards the west (i.e., 270^o). The blue actions are the free-flowing actions and the black actions are the thrust-producing action. Also, the green circle represents the initial location and the red circle indicates the goal location of the vehicle.

Now, if the vehicle decides to start the mission early at $t_{start} = 20$ min (see Figure 5.6(a.1)), the planner generates the path by initially using the free-flow action

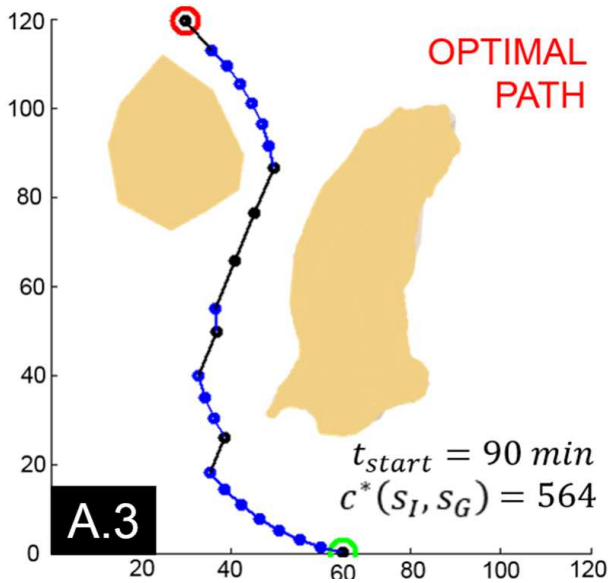


(a)



(b)

Figure 5.6: *Comparison of paths for the scenario A with different start times. The green circle represents the initial location and the red circle represents the goal location of the vehicle. Each blue segment is a free-flow action and each black segment is a thrust-producing action.*



(c)

Figure 5.6: *Comparison of paths for the scenario A with different start times. The green circle represents the initial location and the red circle represents the goal location of the vehicle. Each blue segment is a free-flow action and each black segment is a thrust-producing action.*

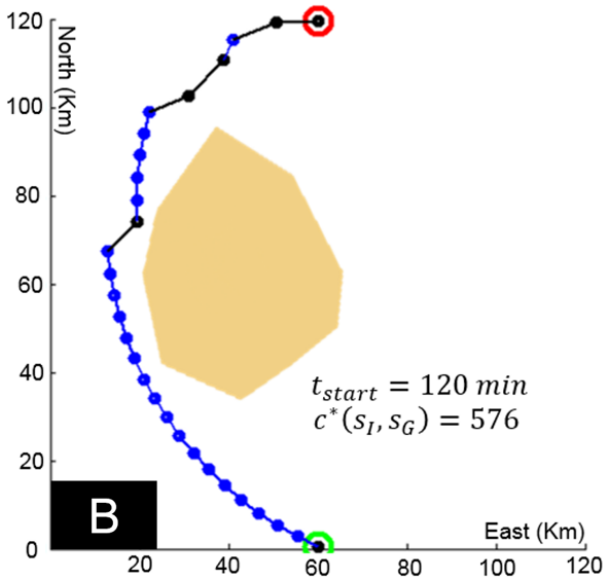
along the medium direction and moves the vehicle far west. In the latter half of the path, the vehicle has to use its thrust-producing action to avoid the obstacle and to reach the goal. On the other hand, if the vehicle prefers to start the mission late (see Figure 5.6(b)), then it can just use the free-flow action in the middle portion of the path. Finally, Figure 5.6(c) shows the lowest-cost path produced by the path planner when the vehicle decides to start the mission at the optimal start time.

The paths shown for scenarios B, C and D in Figure 5.7, are computed at the optimal start time produced by the optimizer. Scenario B has medium flows similar to scenario A, but rotating at the rate 0.4^o/min. Scenario C and D have the same medium flow with constant magnitude of 6 m/s, rotating counterclockwise at the rate of 0.6^o per minute. The initial direction of the medium flow at start time $t_{start} = 0$ is pointing east (i.e., 90^o).

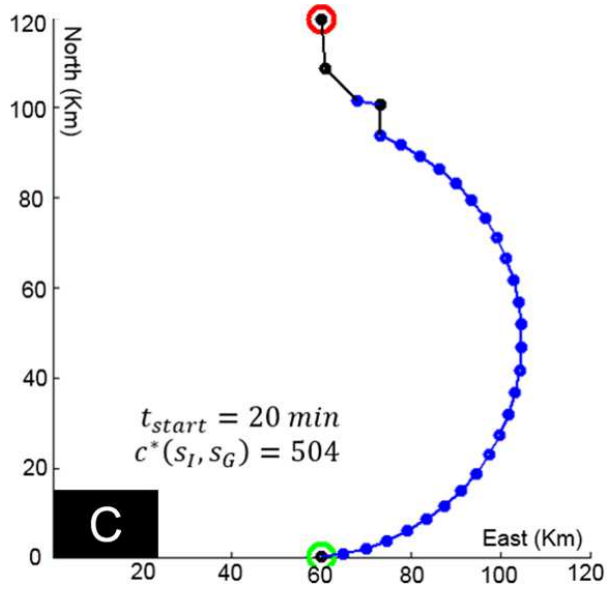
Similar to the results shown in Table 5.2, Table 5.3 shows the comparison between cost ratio C_1/C_2 and C_1/C_3 in example scenarios (A-D). Here, the costs C_1, C_2 are the same as described in Section 5.5.2, cost C_3 is the cost incurred by the vehicle while using the developed energy-efficient path planner and the vehicle starts its mission at optimal time time $t_{start} = t_{optimal}$. The table compares the energy efficiency of the developed algorithm with and without start time optimization.

5.6 Summary

This chapter presents a new approach for generating paths for unmanned vehicles in time-varying flow fields. Generated paths show significant improvement in terms

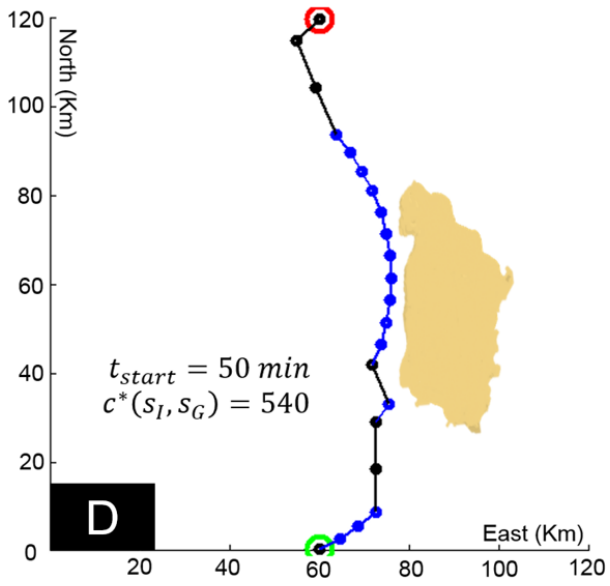


(a)



(b)

Figure 5.7: *Optimal path produced by the path planner for scenarios B, C, and D at optimal start times produced by the optimizer.*



(c)

Figure 5.7: *Optimal path produced by the path planner for scenarios B, C, and D at optimal start times produced by the optimizer.*

Table 5.3: *Comparison between the energy-efficiency provided by the developed path planner without start time optimization (i.e., ratio C_1/C_2) and with start time optimization (i.e., ratio C_1/C_3). Cost C_1 is the cost incurred while using the shortest distance path planner, cost C_2 is the cost incurred while using the developed path planner at time $t_{start} = 0$, and cost C_3 is the cost incurred while using the developed path planner at time $t_{start} = t_{optimal}$.*

Scenarios	Start at t = 0 min	Start at optimal time
	Ratio of C_1 and C_2	Ratio of C_1 and C_3
A	1.38	1.39
B	1.11	1.67
C	1.35	1.55
D	1.21	1.33

of energy cost compared to the shortest distance paths. This has been accomplished by selecting an optimal start time to exploit the flow conditions and using free flow actions that propel the vehicle forward instead of using thrust produced by the actuators. We have developed new admissible heuristics to estimate the cost-to-go in the A* algorithm. These heuristics work effectively to reduce the number of expanded states in a wide variety of flow conditions.

The computational time of the developed algorithm and the quality of the computed path is highly dependent upon the resolution of the state space discretization, thus making the planner difficult to scale with map size. The developed algorithm assumes that the flow-fields do not vary spatially and are independent of the obstacles present in the region. However, this assumption is violated in the real-world scenarios. Incorporating the spatially varying flow-field in the current version of the developed planner will degrade the performance of the heuristic, resulting in significant computation time. In the future, it will be beneficial to extend the current algorithm and heuristic to incorporate the spatially varying flow fields and its interaction with the obstacles present in the region.

Chapter 6

Conclusions

This chapter presents the expected intellectual contributions and anticipated benefits from the work proposed in this dissertation.

6.1 Intellectual Contributions

The tasks listed in Chapter 1 broadly aims towards the development of planning algorithms for autonomous operation of USVs. The following are some of the key contributions:

6.1.1 Risk-Aware Trajectory Planning in Congested Civilian Traffic

This dissertation introduces a novel lattice-based, 5D trajectory planner for an unmanned surface vehicle (USV) operating in an environment with civilian traffic in Chapter 3. The planner: 1) estimates the trajectory's collision risk and reasons about the availability of contingency maneuvers to counteract unpredictable behaviors of civilian vessels; 2) incorporates the avoidance behaviors of civilian vessels into the search to minimize collision risk; 3) considers the USV's dynamics; and 4) dynamically scales control action primitives based on the congestion of state space regions to maximize search performance. This dissertation also introduces a novel congestion metric that ranks the complexity of different marine scenarios by taking

into account the topography of the scene, maneuverability and the relative distances of all the vehicles in the scenario.

6.1.2 Path Planning over Long Distances

The topography of the marine environment significantly varies in terms of free and traversable spaces. Also, traversable space changes over time as a result of tides, environmental restrictions, and weather. Standard grid-based methods with constant grid size are inefficient and computationally inefficient. Sampling-based methods produce non-optimal paths, with the computation time dependent on map specific parameter like d_{min} (in RRT* [76]). This dissertation introduces a resolution independent path planning algorithm that computes optimal paths using a A* search on visibility graphs defined over quadtrees in Chapter 4. This dissertation also provides an admissible heuristic that accounts for large islands while estimating the cost-to-go and provides lower bound superior to a Euclidean distance-based heuristics. We have created a framework where we can read nautical chart data (i.e. shapefiles) and generates the corresponding quadtree of desired maximum depth.

6.1.3 Trajectory Planning in Time-Varying Flow Fields

The maximum velocity and energy consumption of the USVs are significantly influenced by the the medium flows (i.e. ocean currents). This dissertation introduces an A* based path planning algorithm that generates low-cost paths by incorporating the weather forecast models and exploiting the medium flow to aid vehicle's motion

and conserve energy. Traditional admissible heuristics that are based on shortest distance or time are not suitable for this approach as the exploitation of the medium flow often requires longer time or distance to reach the goal. This research work also presents novel admissible heuristics for estimating cost-to-go by taking into account flow considerations.

6.2 Anticipated Benefits

This dissertation introduces a trajectory and path planning algorithm that enables the autonomous operation of USVs in large maritime environment and the execution of missions with long time horizon. A team of autonomous USVs can enhance the safety and operations of the port by patrol and guiding the marine vessels in the busy ports and congested marine environment. Furthermore, the use of USVs allows the Coast Guard to perform finer-scale search in an rescue operation using data obtained solely from aerial vehicles. The USVs can also be used to reduce time, money and risk to life during the offshore extraction of both fossil fuels, renewable energies, and other resources that are often found in harsh environments. Finally, a team of USVs can be used in exploration and sensing of the marine environment that are unexplored and provide large risk to human life.

6.3 Future Directions

This dissertation provides a solid foundation for path and trajectory planners required for autonomous navigation of USVs. The approaches discussed here can

be extended in the following directions to more completely realize the autonomous operation of the USVs in marine environment.

1. *Integration of Perception and Planning for Long Term Operation of Unmanned Surface Vehicles*:

   In all the developed approaches, our autonomous USVs had complete and perfect knowledge about static obstacles and dynamic moving civilian vessels. However, this assumption stands void in real world missions and requires more detailed consideration of perception problem. Perception has always been a challenging problem in robotics and automation, the performance of any perception system varies significantly during missions performed by the USV due to weather conditions, the motion of the vehicle itself, occlusions caused by waves and splashing water on the sensors. This makes real-world planning for autonomous operations of USVs a challenging problem. Performance of the planning system is highly influenced by accuracy of data provided by perception system.

   Planning system for USVs need to plan trajectories to minimize the limitations and maximize the utility of the perception system. Hence, planning has to be done not only for specific mission but also to overcome the limitations of the perception system, sensing and motion uncertainty. In the process of developing this approach (which combines planning and perception), characterizing and modeling the behavior of actual physical sensors on the USV plays an important role. Use of a sensor model in conjunction with filtering techniques

will allow the planning system to predict and minimize the measurement errors and enhances the planning capabilities of the autonomous USV. For example: a camera is blinded by directly facing the sun, which severely degrades its performance. These limitations of the perception system can be addressed by intelligent planning, leading to better performance for long-term autonomous operations of USVs.

2. *Modeling of Civilian Vessels in the Environment for Risk-aware Trajectory Planning*:

   In Chapter 3, we model civilian vessels using a simplistic velocity obstacle (VO) based model. Additionally, the uncertainty in the position of the civilian vessels is approximated by normal distribution with variance increasing in time (computed using Monte Carlo simulations). We did not considered the physical parameters such as weight, actuation type, etc. while predicting the motion of the civilian vessels. We have not accounted for COLREGs-based avoidance strategies and behaviors exhibited by the sailors while avoiding oncoming vessels (e.g. risk taking versus conservative nature). Finally, in the real world scenarios USV will need to use a perception system to determine the state of the civilian vessels. The uncertainty in the perception system must be accounted during the computation of risk-aware trajectory in congested environment.

3. *Incorporation of uncertainty in the path planner for USVs operating in time varying flow fields*:

The lattice-based path planner developed in Chapter 5 presents a new approach to generate energy efficient paths, providing a significant improvement in terms of energy cost as compared to shortest distance-based path planner. However, the developed planner does not consider the temporal and spatial uncertainty in USV's state.

Spatial uncertainty in USV's position arises due to its interaction with the flow-fields (i.e. winds and/or currents). For example, if a USV is trying to maintain its straight line course towards the goal,any significant flow field will hinder USV's motion, causing deviation from the planned straight-line path. Considering spatial uncertainty in USV's position during the computation of the path helps the planner to more accurately determine the USV's risk of collision with the obstacles.

Secondly, temporal uncertainty arises due to delays caused during the execution of the USV's motion primitive by its interaction with the flow-fields. For example, a USV is predicted to reach at the intermediate waypoint at t sec and it actually reaches at $t+x$ sec, where x is uncertain. In this case, the plan must account for the delay and consider the flow field model beginning from $t+x$ sec and not t sec all while starting from the intermediate waypoint. Considering the temporal uncertainty in planning will help the USV to effectively exploit the flow fields.

Bibliography

[1] Petr Švec, Brual C Shah, Ivan R. Bertaska, Jose Alvarez, Armando J. Sinisterra, Karl von Ellenrieder, Manhar Dhanak, and Satyandra K Gupta. Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13)*, 2013.

[2] S.J. Corfield and J.M. Young. Unmanned surface vehicles–game changing technology for naval operations. *Advances in unmanned marine vehicles*, pages 311–328, 2006.

[3] Michael R Benjamin, Joseph A Curcio, John J Leonard, and Paul M Newman. Navigation of unmanned marine vehicles in accordance with the rules of the road. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3581–3587. IEEE, 2006.

[4] Justin E Manley. Unmanned surface vehicles, 15 years of development. In *OCEANS 2008*, pages 1–4. IEEE, 2008.

[5] Hashem Ashrafiuon, Kenneth R Muske, Lucas C McNinch, and Reza A Soltan. Sliding-mode tracking control of surface vessels. *Industrial Electronics, IEEE Transactions on*, 55(11):4004–4012, 2008.

[6] Robin R Murphy, Eric Steimle, Chandler Griffin, Charlie Cullins, Mike Hall, and Kevin Pratt. Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane wilma. *Journal of Field Robotics*, 25(3):164–180, 2008.

[7] John M Dolan, Gregg W Podnar, Alberto Elfes, Stephen Stancliff, Ellie Lin, John Higinbotham, Jeffrey C Hosler, John Moisan, and Tiffany A Moisan. Smart ocean sensing using the telesupervised adaptive ocean sensor fleet. 2008.

[8] Gabriel Hugh Elkaim. System identification-based control of an unmanned autonomous wind- propelled catamaran. *Control Engineering Practice*, 17(1):158–169, 2009.

[9] Brian S Bingham, Eric F Prechtl, and Richard A Wilson. Design requirements for autonomous multivehicle surface-underwater operations. *Marine Technology Society Journal*, 43(2):61–72, 2009.

[10] Patrick F Rynne and Karl D von Ellenrieder. Unmanned autonomous sailing: Current status and future role in sustained ocean observations. *Marine Technology Society Journal*, 43(1):21–30, 2009.

[11] Les Elkins, Drew Sellers, and W Reynolds Monach. The autonomous maritime navigation (amn) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles. *Journal of Field Robotics*, 27(6):790–818, 2010.

[12] Thomas Pastore and Vladimir Djapic. Improving autonomy and control of autonomous surface vehicles in port protection and mine countermeasure scenarios. *Journal of Field Robotics*, 27(6):903–914, 2010.

[13] Reza A Soltan, Hashem Ashrafiuon, and Kenneth R Muske. Ode-based obstacle avoidance and trajectory planning for unmanned surface vessels. *Robotica*, 29(05):691–703, 2011.

[14] Terry Huntsberger, Hrand Aghazarian, Andrew Howard, and David C Trotz. Stereo vision–based navigation for autonomous surface vessels. *Journal of Field Robotics*, 28(1):3–18, 2011.

[15] Vincent Howard, Jonathon Mefford, Lee Arnold, Brian Bingham, and R Camilli. The unmanned port security vessel: an autonomous platform for monitoring ports and harbors. In *OCEANS 2011*, pages 1–8. IEEE, 2011.

[16] Justin Manley, Graham Hine, et al. Persistent unmanned surface vehicles for subsea support. In *Offshore Technology Conference*. Offshore Technology Conference, 2011.

[17] Wolfgang Fink, Markus Tuller, Alexander Jacobs, Ramaprasad Kulkarni, Mark A Tarbell, Roberto Furfaro, and Victor R Baker. Robotic lake lander test bed for autonomous surface and subsurface exploration of titan lakes. In *Aerospace Conference, 2012 IEEE*, pages 1–12. IEEE, 2012.

[18] Petr Švec and Satyandra K Gupta. Automated synthesis of action selection policies for unmanned vehicles operating in adverse environments. *Autonomous Robots*, 32(2):149–164, 2012.

[19] Christopher Kitts, Paul Mahacek, Thomas Adamek, Ketan Rasal, Vincent Howard, Steve Li, Alexi Badaoui, William Kirkwood, Geoffrey Wheat, and Sam Hulme. Field operation of a robotic small waterplane area twin hull boat for shallow-water bathymetric characterization. *Journal of Field Robotics*, 29(6):924–938, 2012.

[20] A Gadre, Shu Du, and D Stilwell. A topological map based approach to long range operation of an unmanned surface vehicle. In *American Control Conference, June*, pages 5401–5407, 2012.

[21] Aditya S Gadre, Christian Sonnenburg, Shu Du, Daniel J Stilwell, and Craig Woolsey. Guidance and control of an unmanned surface vehicle exhibiting sternward motion. In *OCEANS, 2012*, pages 1–9. IEEE, 2012.

[22] Christian R Sonnenburg and Craig A Woolsey. Modeling, identification, and control of an unmanned surface vehicle. *Journal of Field Robotics*, 30(3):371–398, 2013.

[23] JG Marquardt, J Alvarez, and KD von Ellenrieder. Characterization and system identification of an unmanned amphibious tracked vehicle. 2013.

[24] Maurice F Fallon, Hordur Johannsson, Michael Kaess, John Folkesson, Hunter McClelland, Brendan J Englot, Franz S Hover, and John J Leonard. Simul-

taneous localization and mapping in marine environments. In *Marine Robot Autonomy*, pages 329–372. Springer, 2013.

[25] Y. Kuwata, M.T. Wolf, D. Zarzhitsky, and T.L. Huntsberger. Safe maritime autonomous navigation with colregs, using velocity obstacles. *Oceanic Engineering, IEEE Journal of*, 39(1):110–119, Jan 2014.

[26] Eric T Steimle and Michael L Hall. Unmanned surface vehicles as environmental monitoring and assessment tools. In *OCEANS 2006*, pages 1–5. IEEE, 2006.

[27] 2012 marine accident study - uk ports, port skills and safety.

[28] UCG Commandant. International regulations for prevention of collisions at sea, 1972 (72 COLREGs). *US Department of Transportation, US Coast Guard, COMMANDANT INSTRUCTION M*, 16672, 1999.

[29] Thierry Fraichard and Hajime Asama. Inevitable collision states-a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[30] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[31] M. Greytak and F. Hover. Motion planning with an analytic risk cost for holonomic vehicles. In *IEEE Conference on Decision and Control (CDC/CCC'09)*, pages 5655–5660. IEEE, 2009.

[32] P. Švec, M. Schwartz, A. Thakur, and S. K. Gupta. Trajectory planning with look-ahead for unmanned sea surface vehicles to handle environmental disturbances. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, September 2011.

[33] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[34] Sven Koenig and Maxim Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.

[35] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, pages 262–271, 2005.

[36] Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[37] Brual Shah and Satyandra K. Gupta. Speeding up A* search on visibility graphs defined over quadtrees to enable long distance path planning for unmanned surface vehicles. In *Internation Conference on Automated Planning and Scheduling (ICAPS' 16), London, UK, June 12 - 17, 2016*, 2016.

[38] Brual C Shah, Petr Švec, Ivan R. Bertaska, Wilhelm Klinger, Armando J. Sinisterra, Karl von Ellenrieder, Manhar Dhanak, and Satyandra K Gupta. Trajectory planning with adaptive control primitives for autonomous surface

vehicles operating in congested civilian traffic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'14)*, 2014.

[39] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[40] Luis Martinez-Gomez and Thierry Fraichard. Collision avoidance in dynamic environments: an ICS-based solution and its comparative evaluation. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 100–105. IEEE, 2009.

[41] Petr Švec, Brual C Shah, Ivan R. Bertaska, Wilhelm Klinger, Armando J. Sinisterra, Karl von Ellenrieder, Manhar Dhanak, and Satyandra K Gupta. Adaptive sampling based COLREGS-compliant obstacle avoidance for autonomous surface vehicles. In *Workshop on Persistent Autonomy for Marine Robotics (PAMR '14) held at International Conference on Robotics and Automation (ICRA), Hong Kong, China, June 2014*, 2016.

[42] S. M. LaValle. *Planning algorithms.* Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu.

[43] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[44] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[45] Bernard Faverjon and Pierre Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1152–1159. IEEE, 1987.

[46] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3375–3382. IEEE, 1996.

[47] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, 1991.

[48] Fumio Kanehiro, Florent Lamiraux, Oussama Kanoun, Eiichi Yoshida, and Jean-Paul Laumond. A local collision avoidance method for non-strictly convex polyhedra. *Proceedings of robotics: science and systems IV*, 2008.

[49] James Gil de Lamadrid. Avoidance of obstacles with unknown trajectories: Locally optimal paths and periodic sensor readings. *The International Journal of Robotics Research*, 13(6):496–507, 1994.

[50] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[51] Stephane Petti and Thierry Fraichard. Safe motion planning in dynamic environments. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2210–2215. IEEE, 2005.

[52] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1610–1616. IEEE, 2007.

[53] Matthew Zucker, James Kuffner, and Michael Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1603–1609. IEEE, 2007.

[54] Antoine Bautin, Luis Martinez-Gomez, and Thierry Fraichard. Inevitable collision states: a probabilistic perspective. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4022–4027. IEEE, 2010.

[55] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Probabilistic motion planning among moving obstacles following typical motion patterns. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4027–4033. IEEE, 2009.

[56] Dizan Vasquez, Frédéric Large, Thierry Fraichard, and Christian Laugier. High-speed autonomous navigation with motion prediction for unknown moving obstacles. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceed-*

ings. 2004 IEEE/RSJ International Conference on, volume 1, pages 82–87. IEEE, 2004.

[57] Amalia F Foka and Panos E Trahanias. Predictive autonomous robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 490–495. IEEE, 2002.

[58] Chiara Fulgenzi, Anne Spalanzani, Christian Laugier, Christopher Tay, et al. Risk based motion planning and navigation in uncertain dynamic environment. 2010.

[59] Daniel Althoff, James J Kuffner, Dirk Wollherr, and Martin Buss. Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. *Autonomous Robots*, 32(3):285–302, 2012.

[60] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928–1935. IEEE, 2008.

[61] Jamie Snape, Jur Van den Berg, Stephen J Guy, and Dinesh Manocha. The hybrid reciprocal velocity obstacle. *Robotics, IEEE Transactions on*, 27(4):696–706, 2011.

[62] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[63] Boris Kluge and Erwin Prassler. Reflective navigation: Individual behaviors and group behaviors. In *IEEE International Conference on Robotics and Automation*, pages 4172–4177, 2004.

[64] Javier Alonso-Mora, Andreas Breitenmoser, Paul Beardsley, and Roland Siegwart. Reciprocal collision avoidance for multiple car-like robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 360–366. IEEE, 2012.

[65] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. *Optimal reciprocal collision avoidance for multiple non-holonomic robots*. Springer, 2013.

[66] Daman Bareiss and Jur Van den Berg. Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3847–3853. IEEE, 2013.

[67] Jur van den Berg, David Wilkie, Stephen J Guy, Marc Niethammer, and Dinesh Manocha. Lqg-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 346–353. IEEE, 2012.

[68] Martin Rufli, Javier Alonso-Mora, and Roland Siegwart. Reciprocal collision avoidance with motion continuity constraints. *Robotics, IEEE Transactions*

on, 29(4):899–912, 2013.

[69] Bin Xu, Andrew Kurdila, and Daniel J Stilwell. A hybrid receding horizon control method for path planning in uncertain environments. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4887–4892. IEEE, 2009.

[70] Jacoby Larson, Michael Bruch, and John Ebken. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. Technical report, DTIC Document, 2006.

[71] Oivind Loe. Collision avoidance concepts for marine surface craft. *Trondheim, December*, 19:111, 2007.

[72] Antonio Sgorbissa and Renato Zaccaria. Planning and obstacle avoidance in mobile robotics. *Robotics and Autonomous Systems*, 60(4):628–638, 2012.

[73] Thomas M Howard, Colin J Green, Alonzo Kelly, and Dave Ferguson. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325–345, 2008.

[74] Cyrill Stachniss and Wolfram Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 508–513. IEEE, 2002.

[75] Marija Seder, Kristijan Macek, and Ivan Petrovic. An integrated approach to real-time mobile robot control in partially known indoor environments. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6–pp. IEEE, 2005.

[76] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *arXiv preprint arXiv:1005.0416*, 2010.

[77] Aleksandr Kushleyev and Maxim Likhachev. Time-bounded lattice for efficient planning in dynamic environments. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1662–1668. IEEE, 2009.

[78] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5628–5635. IEEE, 2011.

[79] Venkatraman Narayanan, Mike Phillips, and Maxim Likhachev. Anytime safe interval path planning for dynamic environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4708–4715. IEEE, 2012.

[80] S Campbell, W Naeem, and GW Irwin. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control*, 2012.

[81] Wasif Naeem and George W Irwin. An automatic collision avoidance strategy for unmanned surface vehicles. In *Life System Modeling and Intelligent Computing*, pages 184–191. Springer, 2010.

[82] A Tan, Wong Chee Wee, and TJ Tan. Criteria and rule based obstacle avoidance for usvs. In *Waterside Security Conference (WSS), 2010 International*, pages 1–6. IEEE, 2010.

[83] Jacoby Larson, Michael Bruch, Ryan Halterman, John Rogers, and Robert Webster. Advances in autonomous obstacle avoidance for unmanned surface vehicles. Technical report, DTIC Document, 2007.

[84] Wasif Naeem, George W Irwin, and Aolei Yang. COLREGs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics*, 22(6):669–678, 2012.

[85] Ken Teo, Kai Wei Ong, and Hoe Chee Lai. Obstacle detection, avoidance and anti collision for meredith auv. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–10. IEEE, 2009.

[86] Y. Kuwata, M.T. Wolf, D. Zarzhitsky, and T.L. Huntsberger. Safe maritime autonomous navigation with COLREGs, using velocity obstacles. *Oceanic Engineering, IEEE Journal of*, 39(1):110–119, Jan 2014.

[87] Terry Huntsberger and Gail Woodward. Intelligent autonomy for unmanned surface and underwater vehicles. In *OCEANS 2011*, pages 1–10. IEEE, 2011.

[88] Michael R Benjamin, John J Leonard, Joseph A Curcio, and Paul M Newman. A method for protocol-based collision avoidance between autonomous marine surface craft. *Journal of Field Robotics*, 23(5):333–346, 2006.

[89] Michael R Benjamin and Joseph A Curcio. COLREGs-based navigation of autonomous marine vehicles. *Proceedings of Autonomous Underwater Vehicles*, 2004.

[90] Sang-Min Lee, Kyung-Yub Kwon, and Joongseon Joh. A fuzzy logic for autonomous navigation of marine vehicles satisfying COLREG guidelines. *International Journal of Control Automation and Systems*, 2:171–181, 2004.

[91] Y Xue, BS Lee, and D Han. Automatic collision avoidance of ships. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 223(1):33–46, 2009.

[92] LP Perera, JP Carvalho, and C Guedes Soares. Autonomous guidance and navigation based on the colregs rules and regulations of collision avoidance. In *In Proceedings of the International Workshop Advanced Ship Design for Pollution Prevention*, pages 205–216, 2009.

[93] LP Perera, JP Carvalho, and C Guedes Soares. Fuzzy logic based decision making system for collision avoidance of ocean navigation under critical collision conditions. *Journal of marine science and technology*, 16(1):84–99, 2011.

[94] LP Perera, JP Carvalho, and C Guedes Soares. Intelligent ocean navigation and fuzzy-bayesian decision/action formulation. *Oceanic Engineering, IEEE Journal of*, 37(2):204–219, 2012.

[95] James Colito. *Autonomous mission planning and execution for unmanned surface vehicles in compliance with the marine rules of the road*. PhD thesis, University of Washington, 2007.

[96] Wasif Naeem and George W Irwin. Evasive decision making in uninhabited maritime vehicles. In *Proceedings IFAC World Congress, Milan, Italy, August*, pages 12833–12838, 2011.

[97] CheeKuang Tam and Richard Bucknall. Path-planning algorithm for ships in close-range encounters. *Journal of marine science and technology*, 15(4):395–407, 2010.

[98] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[99] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[100] Anthony Stentz, John Bares, Thomas Pilarski, and David Stager. The crusher system for autonomous navigation. *AUVSIs Unmanned Systems North America*, 3, 2007.

[101] P Švec, A Thakur, E Raboin, B. C. Shah, and S. K. Gupta. Target following with motion prediction for unmanned surface vehicle operating in cluttered environments. *Autonomous Robots*, 36:383–405, 2014.

[102] Alonzo Kelly, Anthony Stentz, Omead Amidi, Mike Bode, David Bradley, Antonio Diaz-Calderon, Mike Happold, Herman Herman, Robert Mandelbaum, Tom Pilarski, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006.

[103] Yanbo Li and Jing Xiao. On-line planning of nonholonomic trajectories in crowded and geometrically unknown environments. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3230–3236. IEEE, 2009.

[104] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[105] David Šišlák, Přemysl Volf, and Michal Pěchouček. Accelerated A* path planning. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1133–1134. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[106] David Šišlák, Premysl Volf, and Michal Pechoucek. Accelerated A* trajectory planning: Grid-based path planning comparison. In *Proceedings of the*

19th International Conference on Automated Planning & Scheduling (ICAPS), pages 74–81. Citeseer, 2009.

[107] Peter Yap, Neil Burch, Robert C Holte, and Jonathan Schaeffer. Block a*: Database-driven search with applications in any-angle path-planning. In *AAAI*, 2011.

[108] Peter Kai Yue Yap, Neil Burch, Robert C Holte, and Jonathan Schaeffer. Any-angle path planning for computer games. In *AIIDE*, 2011.

[109] Jérôme Barraquand and Jean-Claude Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2-4):121–155, 1993.

[110] Stephen R Lindemann and Steven M LaValle. Multiresolution approach for motion planning under differential constraints. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 139–144. IEEE, 2006.

[111] Benjamin J Cohen, Gokul Subramanian, Sachin Chitta, and Maxim Likhachev. Planning for manipulation with adaptive motion primitives. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5478–5485. IEEE, 2011.

[112] Kalin Gochev, Benjamin Cohen, Jonathan Butzke, Alla Safonova, and Maxim Likhachev. Path planning with adaptive dimensionality. In *Fourth Annual Symposium on Combinatorial Search*, 2011.

[113] Kalin Gochev, Alla Safonova, and Maxim Likhachev. Planning with adaptive dimensionality for mobile manipulation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2944–2951. IEEE, 2012.

[114] Kalin Gochev, Alla Safonova, and Maxim Likhachev. Incremental planning with adaptive dimensionality. In *Twenty-Third International Conference on Automated Planning and Scheduling*, 2013.

[115] Michael Hoy, Alexey S Matveev, and Andrey V Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03):463–497, 2015.

[116] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317. IEEE, 1994.

[117] H Wang and SJ Julier. Path planning in partially known environments. 2011.

[118] Ryan Luna, Morteza Lahijanian, Mark Moll, and Lydia E Kavraki. Optimal and efficient stochastic motion planning in partially-known environments. In *AAAI Conf. on Artificial Intelligence*, 2014.

[119] Navid Dadkhah and Bérénice Mettler. Survey of motion planning literature in the presence of uncertainty: considerations for uav guidance. *Journal of Intelligent & Robotic Systems*, 65(1-4):233–246, 2012.

[120] Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm:follow the gap method. *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012.

[121] Zhenyu Wu and Lin Feng. Obstacle prediction-based dynamic path planning for a mobile robot. *International Journal of Advancements in Computing Technology*, 4(3), 2012.

[122] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[123] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[124] Franz Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[125] Priyadarshi Bhattacharya and Marina L Gavrilova. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.

[126] Peter Yap. Grid-based path-finding. In *Advances in Artificial Intelligence*, pages 44–55. Springer, 2002.

[127] Der-Tsai Lee. Proximity and reachability in the plane. Technical report, DTIC Document, 1978.

[128] Tomás Lozano-Pérez and Michael A Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[129] Abdulmuttalib Turky Rashid, Abduladhem Abdulkareem Ali, Mattia Frasca, and Luigi Fortuna. Path planning with obstacle avoidance based on visibility binary tree algorithm. *Robotics and Autonomous Systems*, 61(12):1440–1449, 2013.

[130] Alex Nash, Sven Koenig, and Maxim Likhachev. Incremental phi*: Incremental any-angle path planning on grids. 2009.

[131] Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, pages 533–579, 2010.

[132] Alex Nash, Sven Koenig, and Craig Tovey. Lazy theta*: Any-angle path planning and path length analysis in 3d. In *Third Annual Symposium on Combinatorial Search*, 2010.

[133] Tansel Uras and Sven Koenig. Speeding-up any-angle pathplanning on grids. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2015.

[134] James Bailey Craig Tovey, Tansel Uras Sven Koenig, and Alex Nash. Path planning on grids: The effect of vertex placement on path length. 2015.

[135] Dave Ferguson and Anthony Stentz. Using interpolation to improve path planning: The field d* algorithm. *Journal of Field Robotics*, 23(2):79–101, 2006.

[136] Alex Yahja, Anthony Stentz, Sanjiv Singh, and Barry L Brumitt. Framed-quadtree path planning for mobile robots operating in sparse environments. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 650–655. IEEE, 1998.

[137] Clement Petres, Yan Pailhas, Pedro Patron, Yvan Petillot, Jonathan Evans, and David Lane. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23(2):331–341, 2007.

[138] Qi Zhang, Jiachen Ma, and Qiang Liu. Path planning based quadtree representation for mobile robot using hybrid-simulated annealing and ant colony optimization algorithm. In *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, pages 2537–2542. IEEE, 2012.

[139] Dov Kruger, Rustam Stolkin, Aaron Blum, and Joseph Briganti. Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. In *IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 4265–4270, 2007.

[140] Jonas Witt and Matthew Dunbabin. Go with the flow: Optimal AUV path planning in coastal environments. In *Australian Conference on Robotics and Automation*, 2008.

[141] David R Thompson, Steve Chien, Yi Chao, Peggy Li, Bronwyn Cahill, Julia Levin, Oscar Schofield, Arjuna Balasuriya, Stephanie Petillo, Matt Arrott, et al. Spatio-temporal path planning in strong, dynamic, uncertain currents.

In *IEEE International Conference on Robotics and Automation (ICRA'10)*, pages 4778–4783. IEEE, 2010.

[142] T Lolla, MP Ueckermann, K Yigit, PJ Haley Jr, and Pierre FJ Lermusiaux. Path planning in time dependent flow fields using level set methods. In *IEEE International Conference on Robotics and Automation (ICRA'12)*, pages 166–173, 2012.

[143] Michael Soulignac. Feasible and optimal path planning in strong current fields. *IEEE Transactions on Robotics*, 27(1):89–98, 2011.

[144] Bartolome Garau, Alberto Alvarez, and Gabriel Oliver. Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach. In *IEEE International Conference on Robotics and Automation*, pages 194–198. IEEE, 2005.

[145] José Isern-González, Daniel Hernández-Sosa, Enrique Fernández-Perdomo, Jorge Cabrera-Gámez, Antonio Carlos Domínguez-Brito, and Víctor Prieto-Marañón. Obstacle avoidance in underwater glider path planning. *Journal of Physical Agents*, 6(1):11–20, 2012.

[146] Wesam H Al-Sabban, Luis F Gonzalez, Ryan N Smith, and Gordon F Wyeth. Wind-energy based path planning for unmanned aerial vehicles using markov decision processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13)*, pages 784–789, 2013.

[147] Wesam H Al-Sabban, Luis F Gonzalez, and Ryan N Smith. Extending persistent monitoring by combining ocean models and markov decision processes. In *Oceans*, pages 1–10. IEEE, 2012.

[148] Dushyant Rao and Stefan B Williams. Large-scale path planning for underwater gliders in ocean currents. In *Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[149] Nanaz Fathpour, Lars Blackmore, Yoshiaki Kuwata, Christopher Assad, Michael T Wolf, Claire Newman, Alberto Elfes, and Kim Reh. Feasibility studies on guidance and global path planning for wind-assisted montgolfière in titan. *IEEE Systems Journal*, 8(4):1112–1125, 2014.

[150] Yoshiaki Kuwata, Lars Blackmore, Michael Wolf, Nanaz Fathpour, Claire Newman, and Alberto Elfes. Decomposition algorithm for global reachability analysis on a time-varying graph with an application to planetary exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pages 3955–3960, 2009.

[151] Brual C Shah, Petr Švec, Ivan R. Bertaska, Wilhelm Klinger, Armando J. Sinisterra, Karl von Ellenrieder, Manhar Dhanak, and Satyandra K Gupta. Resolution-adaptive risk-aware trajectory planning for surface vehicles operating in congested civilian traffic. *Autonomous Robots*, 2015.

[152] Thierry Fraichard. A short paper about motion safety. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1140–1145. IEEE,

2007.

[153] T.I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. Wiley, 2011.

[154] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[155] Ivan R Bertaska, Brual Shah, Karl von Ellenrieder, Petr Švec, Wilhelm Klinger, Armando J Sinisterra, Manhar Dhanak, and Satyandra K Gupta. Experimental evaluation of automatically-generated behaviors for usv operations. *Ocean Engineering*, 106:496–514, 2015.

[156] Nicholas Chan, James Kuffner, and Matthew Zucker. Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM symposium on robot design, dynamics, and control*, pages 103–114, 2008.

[157] Yun-Hui Liu and Suguru Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles communication. *The International Journal of Robotics Research*, 11(4):376–382, 1992.

[158] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[159] Petr Švec, Atul Thakur, Eric Raboin, Brual C. Shah, and Satyandra K. Gupta. Target following with motion prediction for unmanned surface vehicle operating in cluttered environments. *Autonomous Robots*, 36(4):383–405, 2014.

[160] I.R. Bertaska, J. Alvarez, S. Armando, K. D. von Ellenrieder, M. Dhanak, B. Shah, P. Švec, and S. K. Gupta. Experimental evaluation of approach behavior for autonomous surface vehicles. In *ASME Dynamic Systems and Control Conference (DSCC'13)*, Stanford University, Palo Alto, CA, October 21-23 2013.

[161] Sunglok Choi, Jae-Yeong Lee, and Wonpil Yu. Fast any-angle path planning on grid maps with non-collision pruning. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 1051–1056. IEEE, 2010.

[162] T. Uras and S. Koenig. An empirical comparison of any-angle path-planning algorithms. In *Proceedings of the 8th Annual Symposium on Combinatorial Search*, 2015. Code available at: http://idm-lab.org/anyangle.

[163] Brual Shah, Atul Thakur, Petr Švec, and Satyandra K. Gupta. Path Planning for Unmanned Vehicles Operating in Time-Varying Flow Fields. In *Workshop on Planning and Robotics (PlanRob), held at Internation Conference on Automated Planning and Scheduling (ICAPS' 16), London, UK, June 12 - 17, 2016*, 2016.

[164] John H Reif and Zheng Sun. Movement planning in the presence of flows. *Algorithmica*, 39(2):127–153, 2004.

[165] Nicola Ceccarelli, John J Enright, Emilio Frazzoli, Steven J Rasmussen, and Corey J Schumacher. Micro UAV path planning for reconnaissance in wind. In *American Control Conference (ACC '07)*, pages 5310–5315. IEEE, 2007.

[166] Ryan N Smith and Van T Huynh. Controlling buoyancy-driven profiling floats for applications in ocean observation. *IEEE Journal of Oceanic Engineering*, 39(3):571–586, 2014.

[167] Stuart Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson, 1995.

[168] Maxim Likhachev, Sebastian Thrun, and Geoffrey J Gordon. Planning for Markov decision processes with sparse stochasticity. In *Advances in neural information processing systems*, pages 785–792, 2004.

[169] Scott Sanner, Robby Goetschalckx, Kurt Driessens, Guy Shani, et al. Bayesian real-time dynamic programming. In *International Joint Conference on Artificial Intelligence (IJCAI '09)*, pages 1784–1789, 2009.

[170] Van T Huynh, Matthew Dunbabin, and Ryan N Smith. Predictive motion planning for AUVs subject to strong time-varying currents and forecasting uncertainties. In *IEEE International Conference on Robotics and Automation (ICRA '15)*, pages 1144–1151, 2015.