Spring 2021

PERFORMANCE EVALUATION AND REVIEW FRAMEWORK OF ROBOTIC MISSIONS (PERFORM): AUTONOMOUS PATH PLANNING AND AUTONOMY PERFORMANCE EVALUATION

Allisa Dalpe
University of New Hampshire, Durham

PERFORMANCE EVALUATION AND REVIEW FRAMEWORK OF ROBOTIC MISSIONS (PERFORM): AUTONOMOUS PATH PLANNING AND AUTONOMY PERFORMANCE EVALUATION

BY

ALLISA J. DALPE

B.A. Physics, Connecticut College, 2016

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

in

Ocean Engineering

May 2021

This dissertation has been examined and approved in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Ocean Engineering by:

Dissertation Director, Dr. May-Win Thein,
Associate Professor of Mechanical
and Ocean Engineering

Dr. Martin Renken,
Naval Undersea Warfare Center
Division Keyport

Dr. Yuri Rzhanov,
Research Professor of Ocean Engineering

Dr. Michael Carter,
Associate Professor of Electrical
and Computer Engineering

Dr. Firat Eren,
Artificial Intelligence Dept. Manager
Cezeri Artificial Intelligence and Robotics

on 4/9/2021.

Original approval signatures are on file with the University of New Hampshire Graduate School.

This dissertation is dedicated to my family.

ACKNOWLEDGMENTS

First, I would like to thank my Ph.D. adviser and committee chair, Dr. May-Win Thein. Her endless guidance and support throughout this process is very much appreciated. Thank you for taking the chance on an unknown physics student.

I'd also like to thank my committee members Dr. Martin Renken, Dr. Yuri Rzhanov, Dr. Michael Carter, and Dr. Firat Eren for their time and efforts in providing feedback and advice towards my dissertation. Additional thanks to Dr. Renken for the always interesting conversations on autonomy and for hosting me at NUWC-Keyport for the summer.

Thank you to lab group members Sital Khatiwada and John McCormack for being the first friendly faces I met at UNH. Additional thanks go to lab group members Greg Hatfield, Hannah Arnholt, Ozzy Oruc, and Alex Cook. All of these people played a significant role in the everyday life of a graduate student and the much needed humor and support to accompany that time. I would also like to give extra thanks to Alex for his programming and testing help.

Many thanks to each of the UNH ASV and ROV team members over the past five years for their help and memorable trips to Keyport, WA.

This work was made possible from the help of several funding sources: the Naval Engineering Education Consortium (NEEC), Naval Sea Systems Command (NAVSEA), UNH Graduate School Dissertation Year Fellowship, UNH Summer Teaching Assistant Fellowship, and the UNH Mechanical and Ocean Engineering Department for additional support through Teaching Assistant appointments.

Lastly, I would like to thank my friends and family. Thank you to my brother, Marcus, for your support. I am so proud of all the work you have done towards your degrees and the person you have become. My parents, Mark and Sally, deserve all the credit for the sacrifices they made so I could have the best education possible. Without their unwavering love and support on this journey I would not be where I am today.

TABLE OF CONTENTS

NOMENCLATURE

Abbreviations

A*	A-star
AGV	Autonomous Ground Vehicle
AHP	Analytic Hierarchy Process
ALFUS	Autonomy Levels for Unmanned Systems
ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
AV	Autonomous Vehicle
CAC	Contextual Autonomous Capability
CORA	Core Ontology for Robotics and Automation
CPA	Closest Point of Approach
FIS	Fuzzy Inference System
FL	Fuzzy Logic
FOU	Field of Uncertainty
GT2-FL	General Type-2 Fuzzy Logic
IRC	Intersection Rule Configuration
IT2-FL	Interval Type-2 Fuzzy Logic
JHU	Johns Hopkins University
LMF	Lower Membership Function
MDPs	Markov Decision Processes
MF	Membership Function
MOOS-IvP	Mission Oriented Operating Suite - Interval Programming

MPP	Mission Performance Potential
NCAP	Non-Contextual Autonomous Potential
ORA	Ontologies for Robotics and Automation
PERFORM	Performance Evaluation and Review Framework Of Robotic Missions
PFM	Potential Field Method
POMDPs	Partially Observable Markov Decision Processes
PRM	Probabilistic Road Map
RAPT	Range Adversarial Planning Tool
RHIB	Rigid Hull Inflatable Boat
ROS	Robotic Operating System
RRT	Rapidly Exploring Random Tree
SISO	Single Input Single Output
SoS	System-of-Systems
SSH	Secure Shell
T1-FL	Type-1 Fuzzy Logic
T&E	Test and Evaluation
TUPPs	Testing Unmanned Performance Platforms
UMF	Upper Membership Function
UMS	Unmanned System
UNH	University of New Hampshire
URC	Union Rule Configuration
USBL	Ultra Short Base Line

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

Performance Evaluation and Review Framework Of Robotic Missions (PERFORM): Autonomous Path Planning and Autonomy Performance Evaluation

by

Allisa J. Dalpe

University of New Hampshire, May, 2021

The scope of this work spans two main areas of autonomy research 1) autonomous path planning and 2) test and evaluation of autonomous systems. Path planning is an integral part of autonomous decision-making, and a deep understanding in this area provides valuable perspective on approaching the problem of how to effectively evaluate vehicle behavior.

Autonomous decision-making capabilities must include reliability, robustness, and trustworthiness in a real-world environment. A major component of robot decision-making lies in intelligent path-planning. Serving as the brains of an autonomous system, an efficient and reliable path planner is crucial to mission success and overall safety. A hybrid global and local planner is implemented using a combination of the Potential Field Method (PFM) and A-star (A*) algorithms. Created using a layered vector field strategy, this allows for flexibility along with the ability to add and remove layers to take into account other parameters such as currents, wind, dynamics, and the International Regulations for Preventing Collisions at Sea (COLGREGS). Different weights can be attributed to each layer based on the determined level of importance in a hierarchical manner. Different obstacle scenarios are shown in simulation, and proof-of-concept validation of the path-planning algorithms on an actual ASV is accomplished in an indoor environment. Results show that the combination of PFM and A* complement each other to generate a successfully planned path to goal that alleviates local minima and entrapment issues. Additionally, the planner demonstrates the ability to update for new obstacles in real time using an obstacle detection sensor.

Regarding test and evaluation of autonomous vehicles, trust and confidence in autonomous behavior is required to send autonomous vehicles into operational missions. The author introduces

the Performance Evaluation and Review Framework Of Robotic Missions (PERFORM), a framework for which to enable a rigorous and replicable autonomy test environment, thereby filling the void between that of merely simulating autonomy and that of completing true field missions. A generic architecture for defining the missions under test is proposed and a unique Interval Type-2 Fuzzy Logic approach is used as the foundation for the mathematically rigorous autonomy evaluation framework. The test environment is designed to aid in (1) new technology development (i.e. providing direct comparisons and quantitative evaluations of varying autonomy algorithms), (2) the validation of the performance of specific autonomous platforms, and (3) the selection of the appropriate robotic platform(s) for a given mission type (e.g. for surveying, surveillance, search and rescue). Several case studies are presented to apply the metric to various test scenarios. Results demonstrate the flexibility of the technique with the ability to tailor tests to the user's design requirements accounting for different priorities related to acceptable risks and goals of a given mission.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Broader Impacts

With an increasingly automated world, Autonomous Vehicles (AVs) in water, land, and air, have become a major area of study. With applications spanning science, commercial, and military interests, billions of dollars have been invested in developing technology to build and deploy safe, reliable, and practical vehicles. Current research still primarily consists of simulations and proof-of-concept vehicles tested only in controlled laboratory or field environments due to the lack of reliability in autonomous decision-making [75][116][43]. To transition these autonomous systems into operational missions and public spaces, a quantifiable level of trust and confidence in vehicle behavior requires validation.

Much like how the car industry forever changed the way people go about their lives, autonomous vehicles have the potential to be just as revolutionary with improvements to safety and efficiency [54]. With the benefits of new technology, however, also comes new issues and risks [72]. The automobile industry has spent decades of time and resources to understand and minimize risk factors for human driving behavior [1]. This need for reliability is accentuated by a 2015 National Highway Traffic Safety Administration report which found that 94 percent of traffic accidents and 89–96 percent of ship collisions occur as a result of human error [114] [11]. Autonomous vehicles present new safety threats, but they also show significant promise towards reducing the number of accidents once the technology in this area has matured [72].

While the marine environment may not have the same vehicle density as that of a city street, the ocean poses other unique challenges for AVs. Due to varying sea states and tidal currents and marine vehicles often being underactuated, navigation and control quickly become complex and

sometimes unstable. Obtaining a completely accurate model of this complex marine environment is not only nearly impossible but also bears a high computational cost in the attempt, especially to take into account the excessively high number of variables. Additionally, below the surface, if working with underwater vehicles, the environment requires different approaches than land, ocean surface, and air based vehicles due to the properties of water. Sensors such as Global Positioning System (GPS) and Radar (Radio Detection and Ranging) have signals that are highly attenuated in water, deeming them unusable.

Research cruises are costly (often tens of thousands of dollars per day), hard to reserve, and have limited endurance capacity. These challenges have resulted in less than 15 percent of the ocean being explored. A fleet of autonomous vehicles will improve scientific sampling efficiency for increased data collection and take on longer duration missions. The vehicles can perform tasks dangerous to human involvement, conduct search and rescue operations, enhance national security, and provide shipping support as 90 percent of the world's trade is carried by sea. Although this research focuses on improving evaluation of autonomous technology in the ocean environment, this work can also translate to land and air based applications.

The question now resides in how one may earn public trust in and acceptance of these vehicles in day-to-day society and how should legislators form policy and regulations regarding these vehicles operating without direct human control [72]. The answer lies in the new autonomy standards and test protocols that must be created and put in place to assure AV capabilities and minimum expected performances. No agreed-upon standardized metric yet exists to measure and rationalize robotic decision-making in unconstrained environments. Autonomous ground vehicle companies resort to driving millions of miles to perform validation tests which are generally economically impractical [9][93][56]. Simulations, on the other hand, have the capability of testing a high quantity of scenarios, but lack the rigorous high-integrity integration of all onboard system sensors and hardware. They also have issues providing an environment with the requisite resolution, detail, noise, randomness and other factors that are found in real-world environments, and this in par-

ticular interferes with perception sensors. This is a key matter to note, as perception is often the beginning point of failure across all known autonomous vehicles.

This work introduces the Performance Evaluation and Review Framework of Robotic Missions (PERFORM) as a start towards standardizing autonomy evaluation with platforms of any domain by providing a foundational and generic structure for constructing and evaluating autonomy test missions. PERFORM demonstrates the feasibility of applying fuzzy logic, namely an Interval Type-2 Fuzzy Logic (IT2-FL) strategy, as an effective and efficient evaluation framework for assessing autonomous vehicles in a scalable testbed environment with the ability to further generalize the methods for different mission types and scenarios.

A limiting factor in deploying these vehicles is reliable autonomous decision-making due to the challenge of hierarchically prioritizing tasks and accounting for risk-assessment in an unpredictable marine environment [116][43]. A major component of this decision-making process lies in autonomous path planning. Intelligent path planning is crucial to autonomous mission success. It makes use of all available sensor measurements of the surrounding environment and uses its AV "perception" to determine the AV's appropriate steps (and order thereof) to successfully accomplish a given mission. As such, intelligent path planning serves as the central neural core of an autonomous system. Currently, most autonomous vehicles operate using scripted routines [72]. Improvements in self-learning and decision-making using Artificial Intelligence (AI) and Machine Learning (ML) are gaining traction due to recent developments in available computational power but still have not reached robust enough levels for reliable day-to-day use [72]. Autonomy is not binary and, on the contrary, must be described using a spectrum based upon the amount and level of the AV decision-making abilities. It is critical for testing protocols to acknowledge this spectrum and, as such, must be designed accordingly.

For integration of autonomous vehicles in society, confidence and trust must be established in this autonomous decision-making. To start, this work explores global path planning strategies and builds with this foundation a novel global and local autonomous path planning approach. This work also aims to develop methods for quantification of mission-specific performance parameters to

provide an overall grade of autonomous mission execution in marine testbed environments with the introduction of PERFORM. The existing gap between technological advancement and the effective testing and evaluation of these systems [97] must be closed to make autonomous vehicles practical for field use and not simply for an academic exercise.

1.2 Research Scope and Contributions

The scope of this work extends to two main areas of autonomy research 1) autonomous path planning and 2) test and evaluation of autonomous systems. Path planning is an integral part of autonomous decision-making, and a deep understanding in this area provides valuable perspective on approaching the problem of how to evaluate vehicle behavior. This knowledge is used in the construction of the PERFORM framework for evaluating autonomous performance. Research contributions are as follows and notated as research area (1) or (2):

1. **Exploration of 3 different global path planners (1):** Different preexisting global path planners are explored, compared, and analyzed via numerical simulations and experimental testing to determine the best fit for integration with a local planner.

2. **Development of a novel hybrid (global/local) path planner (1):** Using a multi-layered vector-field approach, A-star (A*) algorithm is integrated with the Potential Field Method (PFM) and tested both in simulation and experimentally. Particular attention is given to analyzing the approach for feasibility on live platforms in areas such as computational load, update rates, and handling sensor data.

3. **Defining a generic mission architecture (2):** To generalize the autonomy evaluation framework, a generic architecture for defining the missions under test is first created. With a mathematical representation, set operations may then be used to compare various generic missions and tasks with the intention of limiting test redundancy and reducing the number of test missions to a constrained number of critical tasks.

4. **Creating a generalized, flexible framework for evaluating autonomy (2):** Using an Interval Type-2 Fuzzy Logic approach, a novel design procedure is developed that is flexible, scaleable, and incorporates sensor uncertainty. Named the Performance Evaluation and Review Framework Of Robotic Missions (PERFORM), the test environment is designed to aid in (1) new technology development (i.e. providing direct comparisons and quantitative evaluations of varying autonomy algorithms), (2) the validation of the performance of specific autonomous platforms, and (3) the selection of the appropriate robotic platform(s) for a given mission type (e.g. for surveying, surveillance, search and rescue).

5. **Testing and analysis of the developed autonomy evaluation metrics (2):** The procedures and metrics are tested and analyzed in several case studies. These case studies show the ability for the evaluation methodology to provide a high-level external view of the autonomous system with which to measure and validate system proficiency (with respect to user-specified mission tasks) and to analyze overall autonomous vehicle behavior.

1.3 Dissertation Organization

The dissertation is organized as follows:

- **Chapter 2** presents general background on path planning as a research area. Three global path planners are explored (A-star (A*), Rapidly Exploring Random Tree (RRT), and Probabilistic Roadmap (PRM)) in both simulation and on an experimental platform to compare planner behavior, computational load, and practicality for real-world application and integration with a local planner.

- **Chapter 3** provides the methodology and application of the chosen A* and Potential Field Method (PFM) path planners to use in development of a hybrid planner using a multi-layered strategy. Simulations and experimental results are presented verifying the approach. The developmental process of a small scale experimental platform is also explained.

- **Chapter 4** addresses the definition of autonomy, mission design, autonomous system architecture, and current test and evaluation strategies. Current limitations in the verification and validation of autonomous systems is discussed. The last part of the chapter introduces the proposed mission design and definitions used for the evaluation framework in the following chapters.

- **Chapter 5** provides background on fuzzy logic and why it is chosen for the given application. Type-1 and Type-2 Fuzzy Logic Systems are explained and a detailed explanation is given for the selection of Interval Type-2 Fuzzy Logic specifically.

- **Chapter 6** focuses on the design of the Interval Type-2 Fuzzy Logic framework for evaluating autonomous performance. A detailed generalized procedure is outlined in addition to techniques for choosing performance parameters, modeling membership functions, and creating a rule base.

- **Chapter 7** presents selected case studies to demonstrate the capabilities of IT2-FL as a strategy for assembling important metrics identified with respect to the mission framework and the inherent tasks. These case studies focus on individual tasks and indicate the applicability of PERFORM for comparing path planning algorithms.

- **Chapter 8** presents three additional case studies to demonstrate the capabilities of PERFORM towards evaluating full missions. Analysis on the potential inferences that may be stated from test results are examined along with the potential scope of applications.

CHAPTER 2

COMPARISON OF GLOBAL PATH PLANNERS

2.1 Path Planning Introduction

The navigation of autonomous vehicles relies on robust and high-performance path planning. Several strategies exist to approach this problem and may be classified into two groups based on the type and amount of information that is made available to the vehicle [109]. *Global path planning* refers to situations where complete information of the environment is available. *Local path planning*, on the other hand, requiring no memory, relies solely on reactive methods and is utilized in unknown environments. Ideal AV applications use a hybrid (global/local) approach dependent on such things as available memory, computing power and sensors, to name a few. This hybrid, hierarchical paradigm fuses sensor observations into one global data structure, commonly referred to as a "world model" [98]. Applications such as manufacturing may only need global methods if the task is merely repetitive in nature and if the robot is not required to take into account unpredictable situations. Figures 2.1 - 2.3 display the system architecture of these intelligent robot paradigms.

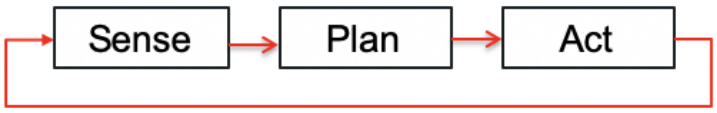


Figure 2.1. Global Path Planning Architecture

Figure 2.2. Local Path Planning Architecture

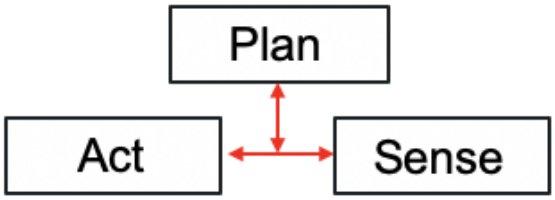


Figure 2.3. Hybrid Path Planning Architecture

In describing the characteristics of a path planning algorithm, common terms include optimality, completeness, and complexity. *Optimality* refers to minimizing a specified condition. This condition, for instance, could be total distance, battery usage, or required vehicle turning to name a few. An algorithm is *complete* if for all instances a solution is found (if any exist) and otherwise returns failure. This property describes whether or not the algorithm is dependable and will work as expected given any set of inputs. *Complexity*, often referred to in order of magnitude (i.e., $O(n)$), describes the performance of an algorithm based on the amount of resources required. Time complexity is often measured, although space and communication workloads are sometimes also analyzed. The smallest execution time is highly desirable in real-time applications.

For the purposes of this chapter, *obstacle avoidance* is in reference to static obstacles within an environment, while *collision avoidance* refers to dynamic obstacles which are defined as a function of space and time. It is assumed that all obstacles are closed and untraversable. Also, while this chapter focuses on autonomous path planning, the scope of this work does not address motion planning. Here, the distinction lies in the fact that path planning does not take vehicle dynamics into account, while motion planning takes the solution of the path planning algorithm and converts it into a feasible trajectory [131]. Motion planning weighs more heavily on the controls implementation. Note that Rapidly Exploring Random Tree (RRT) and Probabilistic Road Map (PRM) can also be considered motion planning algorithms but are treated as path planners in the context of this chapter.

The following sections focus on obstacle avoidance strategies beginning with the analysis of three different global path planners 1) A star (A*) 2) Rapidly Exploring Random Tree (RRT) and

3) Probabilistic Roadmap (PRM). The results of this analysis are used in the development of a novel hybrid path planner which uses both A* and the Potential Field Method (PFM) [31] [30], which is presented in Chapter 3.

2.2 Algorithm Background

Well known algorithms in the global path planning category include A star (A*) [48], Rapidly Exploring Random Tree (RRT) [67] and Probabilistic Road Map (PRM) [61] [60]. A* is a node-based optimal algorithm while PRM and RRT are sampling-based techniques. These algorithms are chosen for comparison in this study due to their established literature, simulated studies, and practicality for experimental testing. It is noted here that not a significant amount of work has been published regarding field testing of autonomous surface vehicles [75][116][69]. The majority of vehicles in the literature were built for specific research applications and, therefore, did not necessarily undergo robust testing of various path planning algorithms.

One example of experimental verification was provided by Kim et al.[63] using an angular rate constrained Theta* method. Other studies combined various planners into a hybrid form, such as the one by Loe [76], which merged the A* and RRT global planners. Bertaska et al. [16] evaluated the performance of four automatically generated path-planning behaviors via field-testing by using a global lattice-based trajectory planner technique [121]. Field trials were accomplished by Song et al. [117] to demonstrate a smoothed A* algorithm approach where a pre-generated trajectory was calculated offline before the start of the test mission.

In this study, observations are made with regards to both transferability and practicality from simulation to real-world application. Global path planner traits are also explored for future integration with local methods. In an environment with rapidly changing factors such as wind, waves, and currents, this work considers approximate global planning solution as appropriate for practical use. Different planner parameters, furthermore, are tested to include such factors as the number of nodes, number of iterations, and maximum connection distances to observe the balance between efficiency and feasibility in reaching a desired waypoint. System modularity and low cost com-

ponents are a point of emphasis and are placed as a secondary goal in this research. Establishing a reliable and robust surface vehicle autonomy system is the foundational initial stage of a UNH research effort to establish a collaborative network of autonomous surface and underwater vehicles. In this regard, modularity is considered a crucial component to facilitate expanding current research to multiple platforms.

Variations of each of these classical path planning algorithms have been developed to address inherent limitations. Details on each algorithm and discussion on related variations are provided here. Generic and base forms of the algorithms are used for general comparison in this study, as other specific variations thereof may always be chosen at a later time if it is so determined that such modifications would further aid in local planner integration.

2.2.1 A*

The A star (A*) method [48] aims to minimize the total path cost using features from "uniform cost search" with "best-first search." Therefore, the algorithm converges to an optimal solution quickly but has a sizeable memory requirement. The cost function is represented as:

$$F(n) = G(n) + H(n) \tag{2.1}$$

Here, $F(n)$ represents the total cost, $G(n)$ the cost from the start node to the current node, and $H(n)$ the estimated cost from the current node to the goal node. The cost $G(n)$ typically represents the distance traveled. Algorithm 1 displays pseudocode for A* path searching. Parameters such as connection distance and method of distance calculation (e.g. Manhattan or Euclidian distance) may be interchanged to find a balance of between optimization and efficient computing. In this research, linear distance is used for numerical simulations. A* has space complexity of $O(b^{(d+1)})$ and time complexity of $O(b^d)$ where b and d denote the average branching factor and tree depth, respectively.

Daniel et al. introduced a variant of A*, referred to as Theta*, that does not constrain paths to grid edges and therefore may plan an any-angle path [32]. The benefit of this strategy is the

Algorithm 1 A* algorithm pseudocode

```
ClosedList = {}
OpenList = {Start}
while OpenList ≠ ∅ do
    CurrentNode = OpenListNode with smallest F(n) score
    Remove CurrentNode from OpenList
    Calculate neighbors of CurrentNode
    for each neighbor of CurrentNode do
        if Neighbor = Goal then
            return Reconstruct Path
        else
            Neighbor.G = Distance Start to CurrentNode
            Neighbor.H = Distance CurrentNode to GoalNode
            Neighbor.F = TotalCost
            if (Neighbor ∈ OpenList) ∧ (Neighbor.F > OpenList F score) then
                Ignore Neighbor
            else if (Neighbor ∈ ClosedList) ∧ (Neighbor.F > ClosedList F score) then
                Ignore Neighbor
            else
                Add Neighbor to OpenList
            end if
            CurrentNode ∈ ClosedList
        end if
    end for
    return Reconstruct Path
end while
```

potential to find the true minimum path of travel. In classic implementations, A* grid paths are artificially constrained to be integer multiples of a pre-determined angle (say, 45 degrees) with each step, limiting the ability of the algorithm to find, let alone traverse, the shortest possible distance. Choi and Yu [25] extended Theta* to work with non-uniform costmaps. Non-uniformity allows for modeling grid cell cost on characteristics such as terrain, obstacles, and other environmental attributes.

2.2.2 Rapidly Exploring Random Tree

The Rapidly Exploring Random Tree (RRT) method [67] randomly samples the configuration space and finds a path from the nodes and connections generated. The maximum number of samples and connection distances that the algorithm searches for are user-defined. RRTs are generally found to bias largely unexplored portions of the state space but demonstrate consistent behavior. This is because the number of distributed vertices approaches that of the sampling distribution [67]. As a single query algorithm, RRTs incrementally query the travel space of interest, so the number of samples does not necessarily need to be determined a priori [57]. This method may also be applied to online planning cases. The pseudocode and an example of the algorithm node expansion are given in Algorithm 2 and Figure 2.4, respectively. $RANDOM\_STATE()$, $NEAREST\_NEIGHBOR()$, and $STEER()$ refer to functions that select a random configuration, returns the nearest vertex, and calculates a new configuration for the direction of q_{rand}, respectively.

Kuffner and Lavalle [68] showed probabilistic completeness for the RRT algorithm. Typically, RRTs work well in situations of high-dimensionality, differential constraints, and nonlinear dynamics [57] [68] [40]. The requirement of few parameters and heuristics also make RRT an attractive choice. However, it also has drawbacks in its sub-optimality, its exploration of space uniformly potentially leading to slow convergence towards the goal and possibly even entrapment, and its high computational costs as a result of the set of nodes expanding with each iteration [45].

RRT has space complexity of $O(n)$ with time complexity of $O(nlog(n))$ for the processing stage and $O(n)$ for the query stage.

Algorithm 2 RRT algorithm pseudocode (q_{start}, q_{goal}, $maxVertices$)

```
Initialize graph G(V, E) with start configuration q_start
for i = 1 to maxVertices do
    q_rand ← RANDOM_STATE()
    q_near ← NEAREST_NEIGHBOR(q_rand, G)
    q_new ← STEER(q_near, q_rand)
    if ObstacleFree(q_near, q_new) then
        V ← V ∪ {q_new}; E ← {(q_near, q_new)};
    end if
end for
return G(V, E)
```

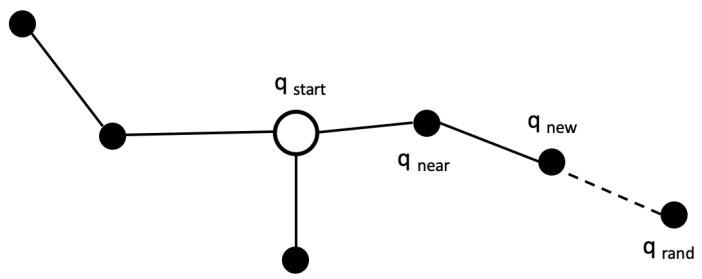


Figure 2.4. Example of RRT node expansion

Karaman and Frazzoli [57] contributed an updated version of RRT, referred to as RRT*, which is asymptotically optimal while maintaining the computational complexity to within a given constant factor of the classic RRT algorithm. Since RRT* searches for the optimal path, this variation often shows slow convergence rates and memory issues when searching large spaces [45]. Another variant, RRT*-FN [10], addresses some of these concerns by restricting the total allowable number of expanded nodes in the RRT* algorithm.

2.2.3 Probabilistic Roadmap

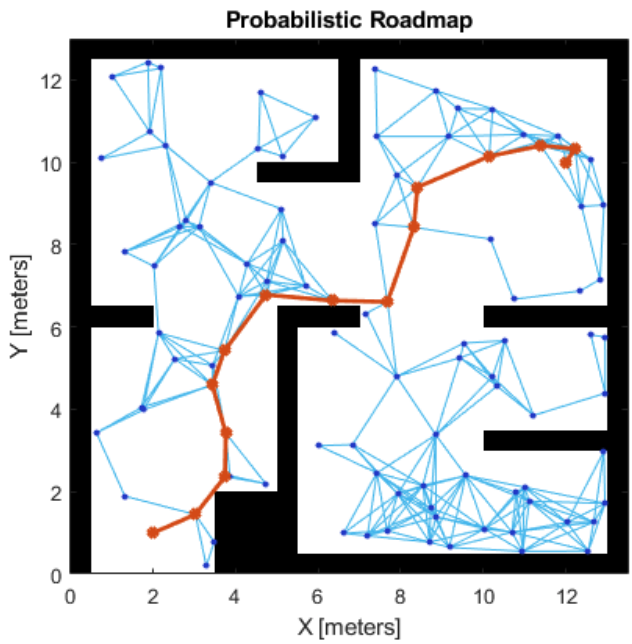
Probabilistic Road Map (PRM) methods are similar to that of the RRT method, as it randomly samples the configuration space but differs in how the node graph is constructed [57]. The user

selects the number of samples and connection distance. This method utilizes a multi-query strategy with a learning and query phase [61]. The learning phase samples the points and connects the nodes, while the query phase determines feasible paths from a given origin to the goal waypoint. The shortest path is then calculated. As nodes are increased, the probability of failure to find a path decreases exponentially to zero [57] and and ensures that the algorithm is probabilistically complete. Figure 2.5 provides an example of how the number of nodes affects the search path. PRMs tend to perform best in highly structured and well-known environments [57]. PRM has space complexity of $O(n)$ with time complexity of $O(nlog(n))$ for both the processing stage and the query stage. The pseudocode is provided in Algorithm 3 where $NEAR()$ refers to all nodes within a specified radius, r.

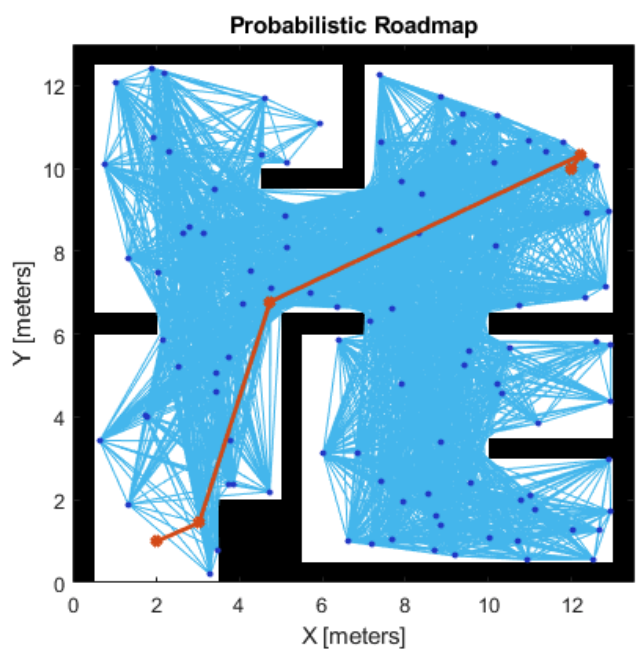
Algorithm 3 PRM algorithm pseudocode (pre-processing phase) [57]

```
Initialize graph G with start configuration q_start
V ← ∅
E ← ∅
for i = 1 to maxVertices do
  q_rand ← a randomly chosen free configuration
  U ← NEAR(G, q_rand, r)
  V ← V ∪ {q_rand}
  for each u ∈ U in increasing order of ||u − q_rand|| do
    if q_rand and u are not in the same connected component of G then
      if CollisionFree(q_rand, u) then
        E ← E ∪ {(q_rand, u), (u, q_rand)}
      end if
    end if
  end for
end for
return G(V, E)
```

Similar to RRT*, PRM has a modified algorithm, PRM* [57], which addresses some of the classic PRM algorithm disadvantages. Although this strategy increases the computational complexity of the procedure, PRM* creates a dense graph where the number of edges approaches the maximum possible in the graph, resulting in smoother planned paths.



(a)



(b)

Figure 2.5. Example of the PRM algorithm with (a) a few nodes (b) many nodes. (Courtesy of MathWorks Matlab documentation [82])

2.3 Methods

A binary occupancy grid is used to differentiate between free space from obstacle locations in the workspace. Grid resolution is set to $1m^2$ with a bounded operation area of 50x100m. The three planners (A*, RRT, and PRM) are simulated under the same test conditions. 2-D movement is assumed with the vehicle treated as a point mass. Three obstacle configurations are tested: a two obstacle configuration, a U-shaped configuration, and a three obstacle U-shape configuration with gaps. These test scenarios are chosen to analyze situations such as obstacles with local minima that may confuse the algorithms. A 2m buffer zone surrounding each obstacle is applied for simulation and field testing.

A West Marine 8ft. Rigid Hull Inflatable Boat (RHIB) is used as the experimental platform (Figure 2.6). Navigational sensors include an Inertial Measurement Unit (IMU) and Global Positioning System (GPS). These sensors are connected to a laptop running Mission Oriented Operating Suite - Interval Programming (MOOS-IvP) as the autonomy software [5]. Two Arduino microcontrollers are used to control steering and thrust of a DC brushless trolling motor. The electronics package is shown in Figure 2.7.

Figure 2.6. Platform ASV4 ready for testing

Figure 2.7. ASV4 Electronics

MOOS-IvP, comprised of the MOOS-DataBase (MOOSDB) and the Interval Programming (IvP) Helm, is an open source software platform that operates with a centralized database to publish and subscribe to data. Its "decisions" are based on its autonomy IvP helm. Several built-in MOOS-IvP features and applications are used in addition to applications designed for an Arduino microcontroller and a Razor Sparkfun IMU [18] [14]. These applications and corresponding software architecture are shown in Figure 2.8. Waypoints are generated during numerical simulations and input to MOOS-IvP for experimental verification. Open source code for each algorithm [10] [122] [82] is modified for integration in this specific system. Offline waypoint generation is performed a priori for the sake of computational efficiency and with the anticipation of incorporating real-time on-board waypoint-to-waypoint generation in future work.

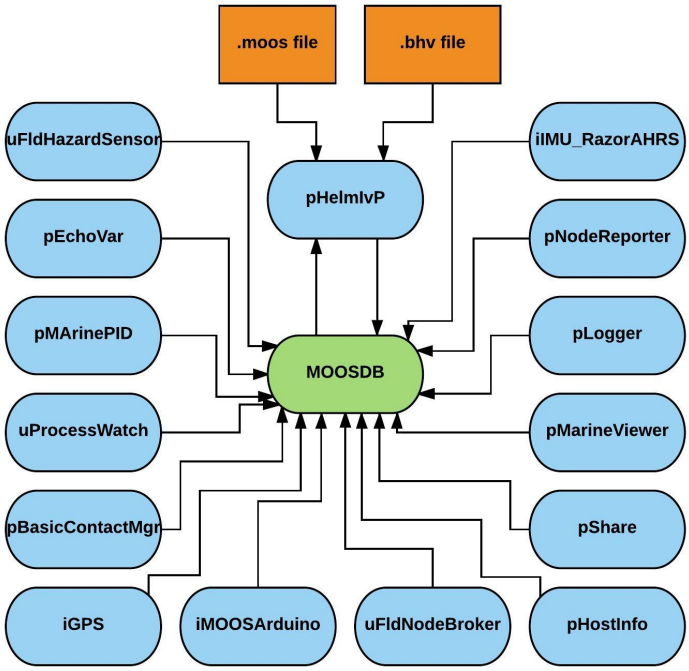


Figure 2.8. MOOS-IvP architecture implemented on ASV4

2.4 Results

2.4.1 Analytical Simulations

The test conditions previously described are applied in analytical tests using numerical simulation software. Figures 2.9 - 2.11 shows the simulation results of each algorithm for each obstacle configuration. The start and goal coordinates are at (0,0) and (49,99), respectively. It should be noted that multiple simulation runs are performed for the PRM simulations to show variance in path planning inherent to the algorithm using a changing random seed value. A* consistently generates identical paths for each simulation run, since the algorithm is based on lowest cost. As such, multiple runs are not shown. The RRT implementation utilized the same random seed, so the random nodes generated are the same for all simulations, noting that RRT would also produce different configurations, however, with varying random seed values. Simulation parameters are

Table 2.1. Path Planning Testing Values

Algorithm	Connection Distance (m)	Number of Nodes/Iterations
A*	10	N/A
PRM	10	300
RRT	5	3000

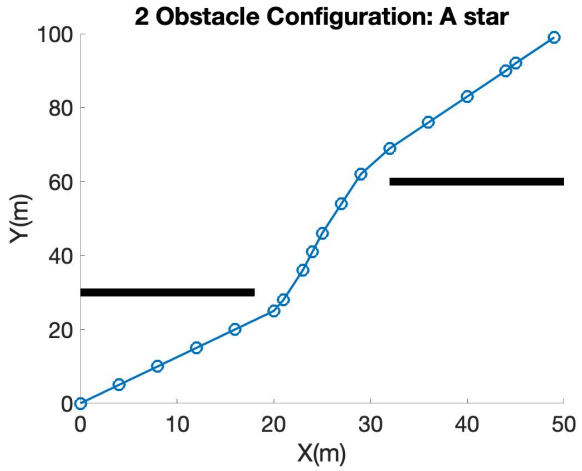
chosen based on achieving a compromise between path speed and the ability to find a path and are provided in Table 2.1.

These three figures (Figures 2.9 - 2.11) display the characteristics of each algorithm as expected. In all three obstacle configurations, A* provides a more direct route. PRM and RRT demonstrate their sampling strategies as the calculated paths show evidence of how they sample the space with less direct and suboptimal routing. These characteristics could be beneficial in specific scenarios.

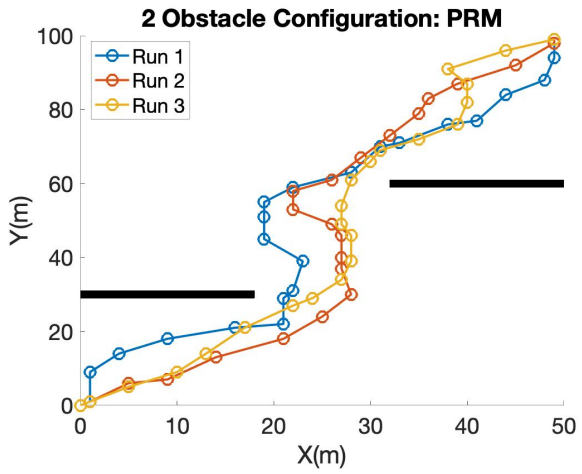
Figure 2.12 displays paths generated for each algorithm incorporating varying connection distances for each algorithm. 600 and 6000 iterations are used for PRM and RRT, respectively. A* calculated a more direct route as the connection distance increased, however this might not be the case in all obstacle configurations as the step size may not be of a small enough resolution to handle several obstacles, especially in close proximity. PRM also displayed smoother path with a higher connection distance, but would also have the same issues in tight spaces. RRT generated the most indirect paths of the algorithms with all connection distances providing poor routing.

Figure 2.13 shows how the path changes as iterations increase for PRM. RRT is not included for the iteration test because it generates the same path and is independent of iteration number as long as there are enough iterations to determine a feasible path. With more iterations the algorithm can improve sampling of the space as is shown with the path improvements as the iterations increase in Figure 2.13

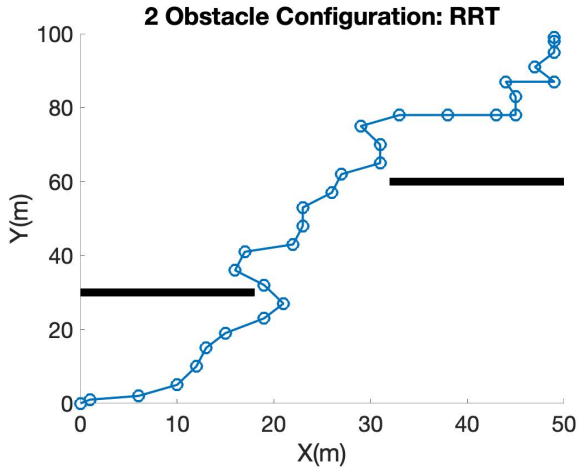
Figures 2.14 - 2.17 compares time and number of iterations for different connecting distances and obstacle configurations. The number of iterations or nodes tested for PRM and RRT differ by a factor of 10 due to what is determined as a practical range of values for each specific algorithm through trial and error testing. As a result, direct comparisons of the number of iterations is not



(a)

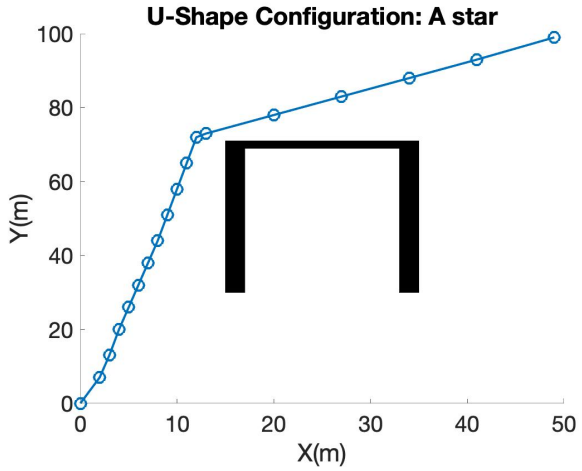


(b)

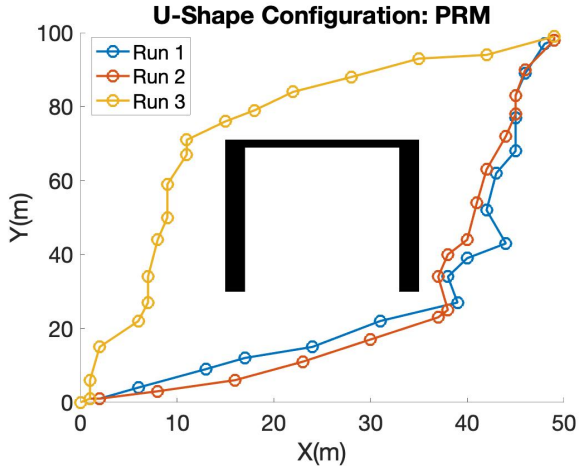


(c)

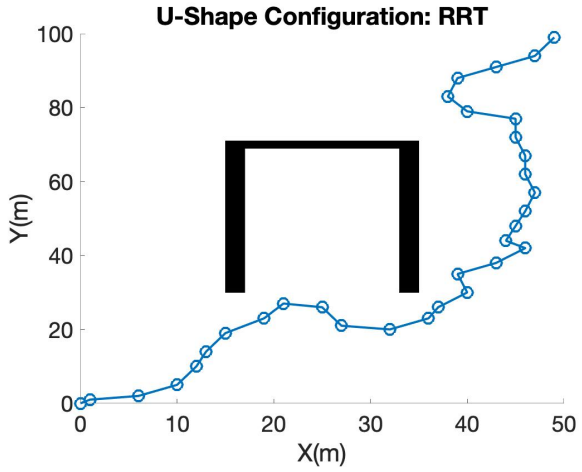
Figure 2.9. Simulation results for the 2 obstacle configuration



(a)

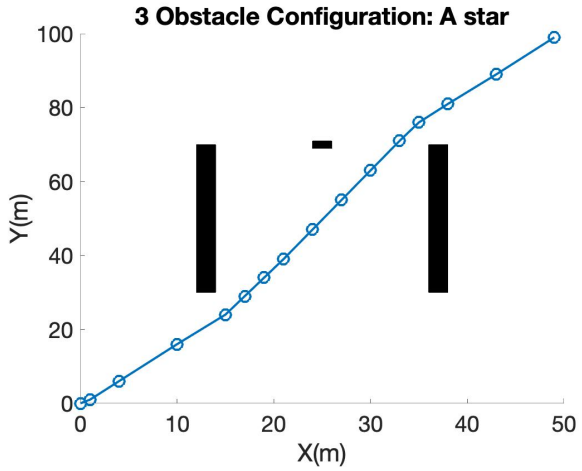


(b)

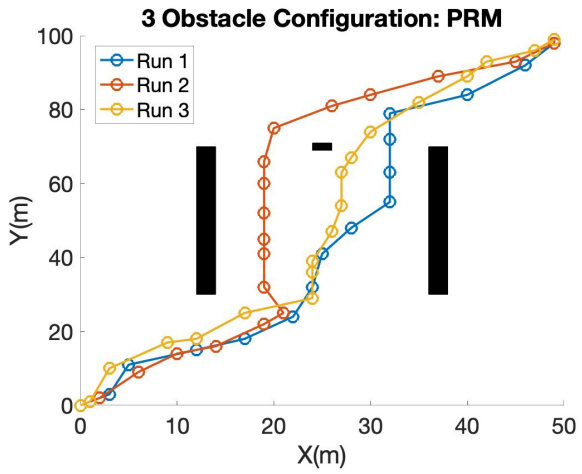


(c)

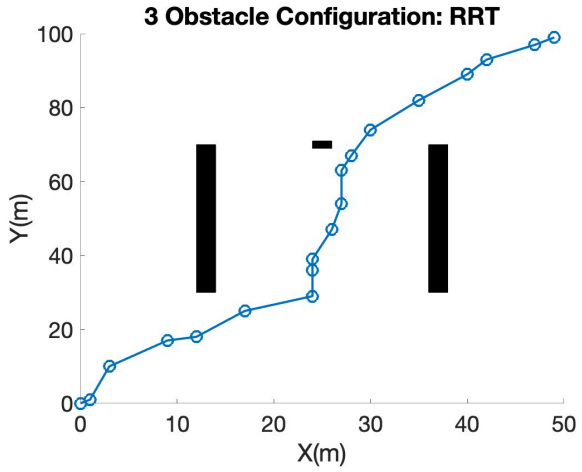
Figure 2.10. Simulation results for the U-shaped obstacle configuration



(a)

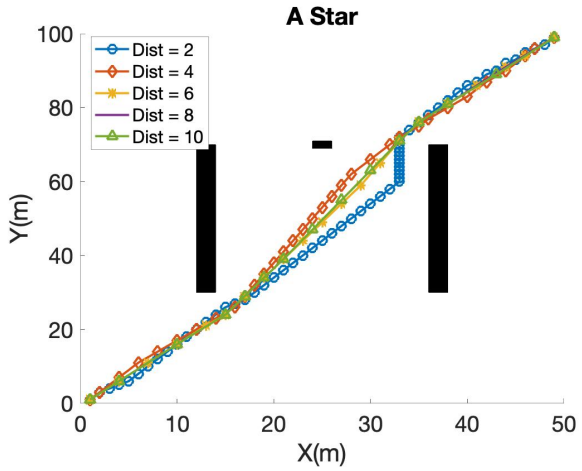


(b)

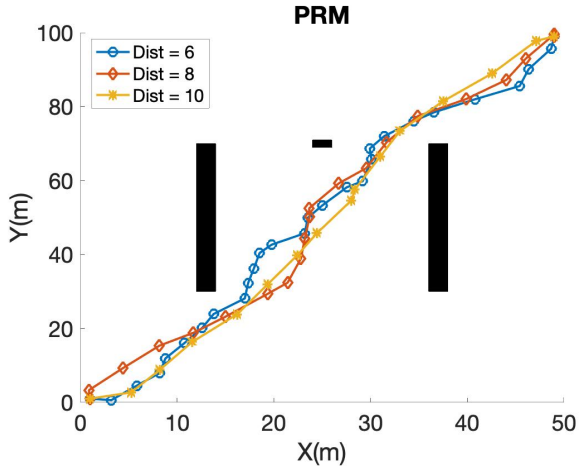


(c)

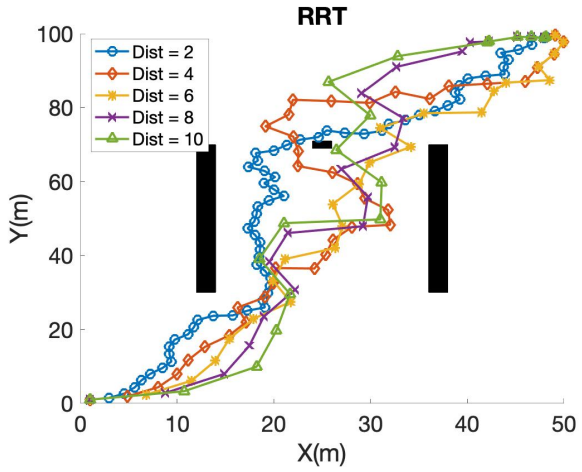
Figure 2.11. Simulation results for the 3 obstacle configuration



(a)



(b)



(c)

Figure 2.12. Simulation results for varying connection distances

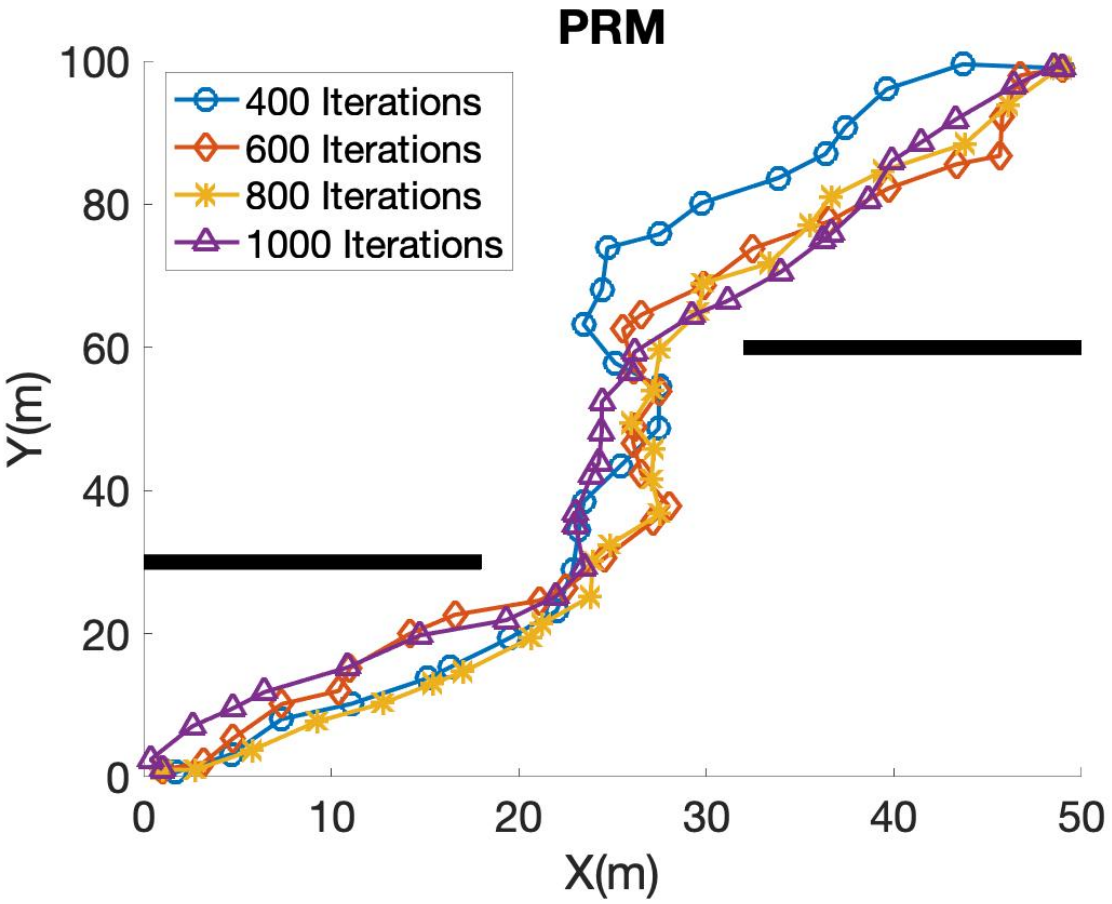


Figure 2.13. Simulation results from varying iteration values with the PRM algorithm

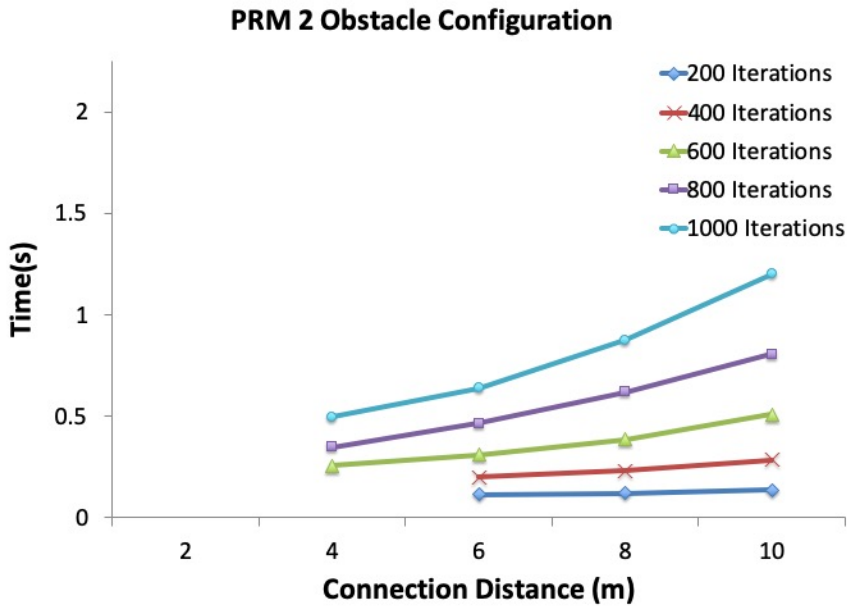
possible, but algorithm qualities and speed for practical use may still be analyzed. The average time is calculated for 10 simulation runs. The PRM algorithm begins to have positive slope as iterations and connection distance increase. The RRT algorithm increases computation time as the number of iterations increase as expected. Time did not significantly increase with increasing connection distance. Computing time for A* increased slightly at higher connection distances, but still remained faster than the other two algorithms.

To summarize, as expected, A* produces the smallest path times overall and is always successful in finding a path. Average calculation time decreases with smaller connection distances. PRM is able to find a path more quickly at lower connection distances. However, the lower iteration threshold requires higher connection distances to find this path. RRT does not appear to depend on connection distance, and its calculation time steadily increases as the number of iterations increase.

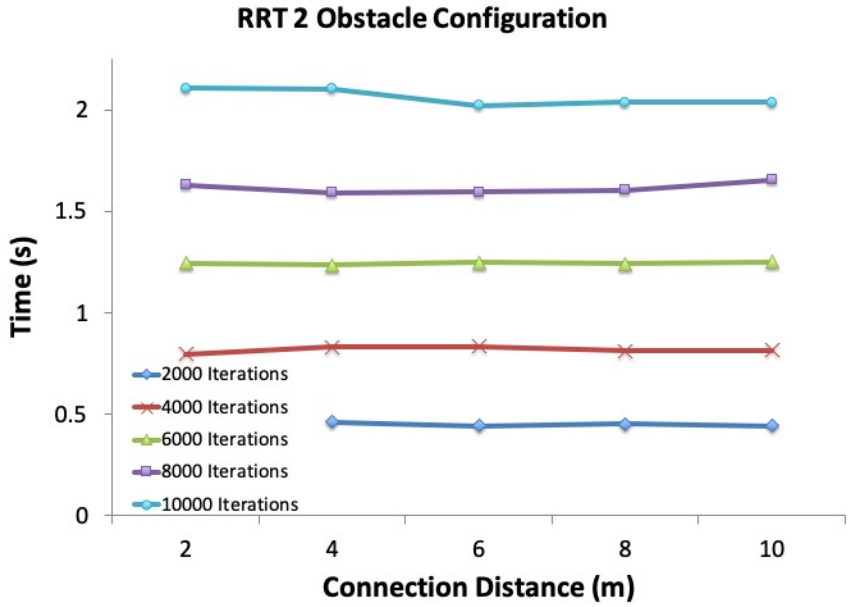
2.4.2 Experimental Results

Experimental field tests are used to observe the practicality of each of the path planning algorithms. ASV4 is used as the test platform representing a generic ASV craft that is only minimally tuned for speed and heading control, so as to mimic non-ideal mission conditions (e.g. poorly controlled craft due to currents, winds, and high sea states). Due to time constraints, one set of path planning parameters are used for each algorithm for experimental tests. Three field tests are performed for each path planner per obstacle configuration. The same values of connection distances and number of nodes/iteration used for analytical simulations (those provided in Table 2.1) are used for these experimental tests and are determined by selecting a mix of roughly tuned values for calculation speeds and nodes.

Constant vehicle speed is maintained throughout field testing, as the main objective of the experiments is to analyze the performance of the global planners during waypoint-following missions. As previously noted, waypoints are generated offline a priori, and the same points are used as those in Figures 2.9 - 2.11. Since the map is known ahead of time, obstacles are placed virtu-

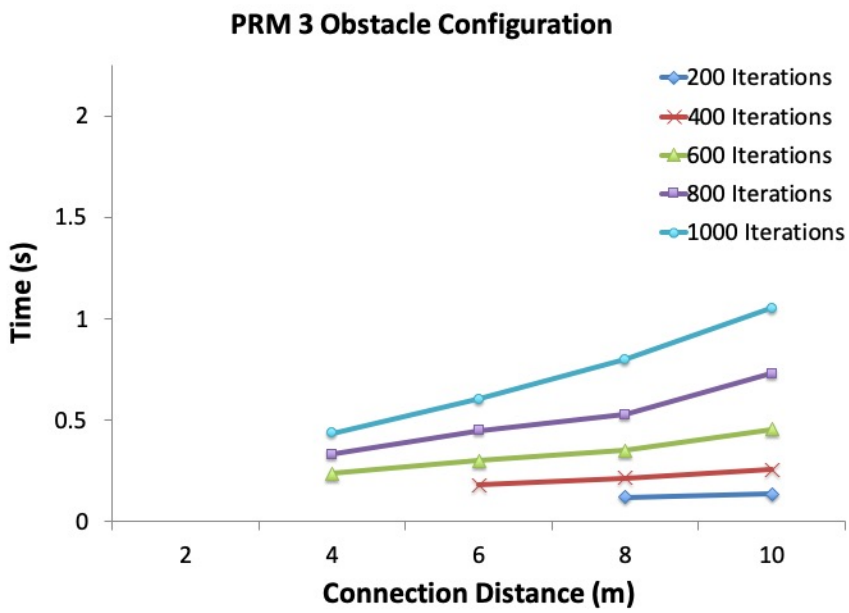


(a)

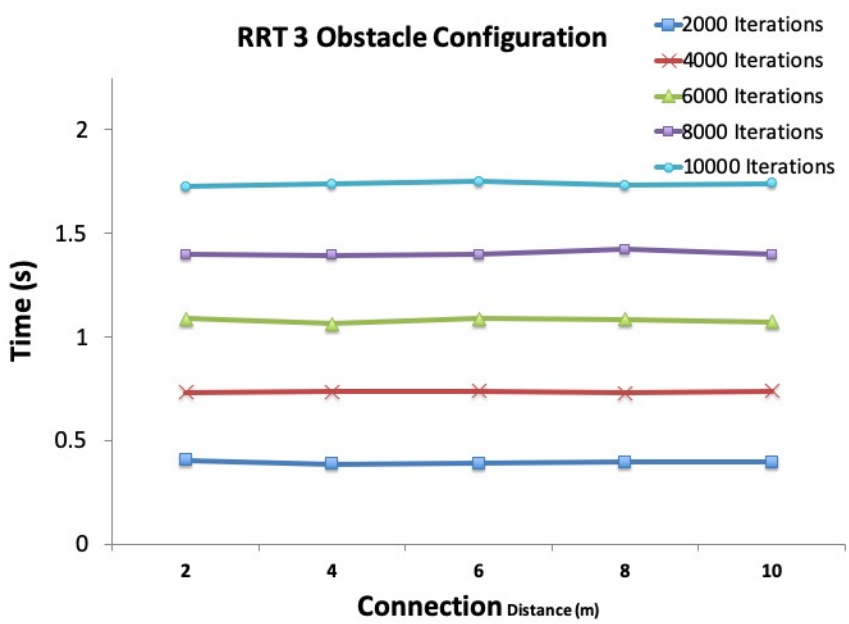


(b)

Figure 2.14. Simulation results for computation time with the 2 obstacle configuration

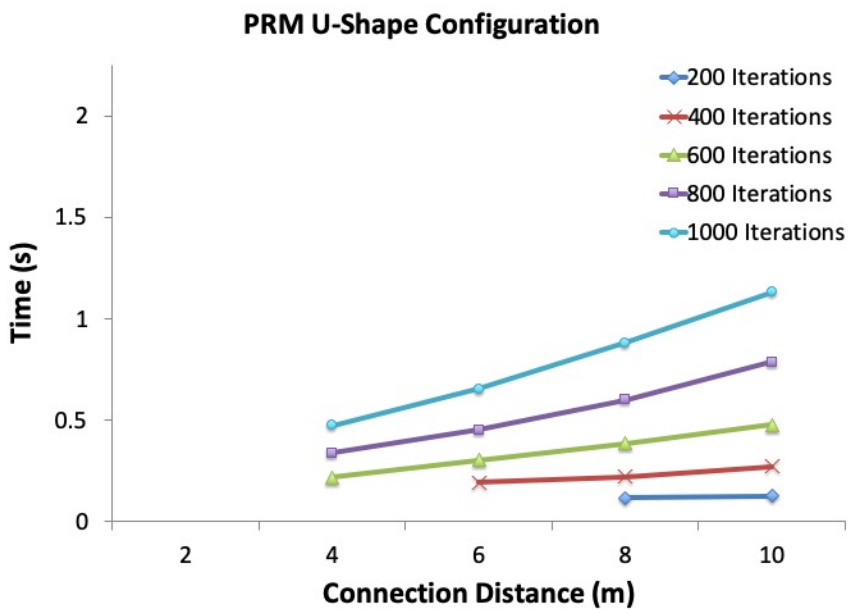


(a)

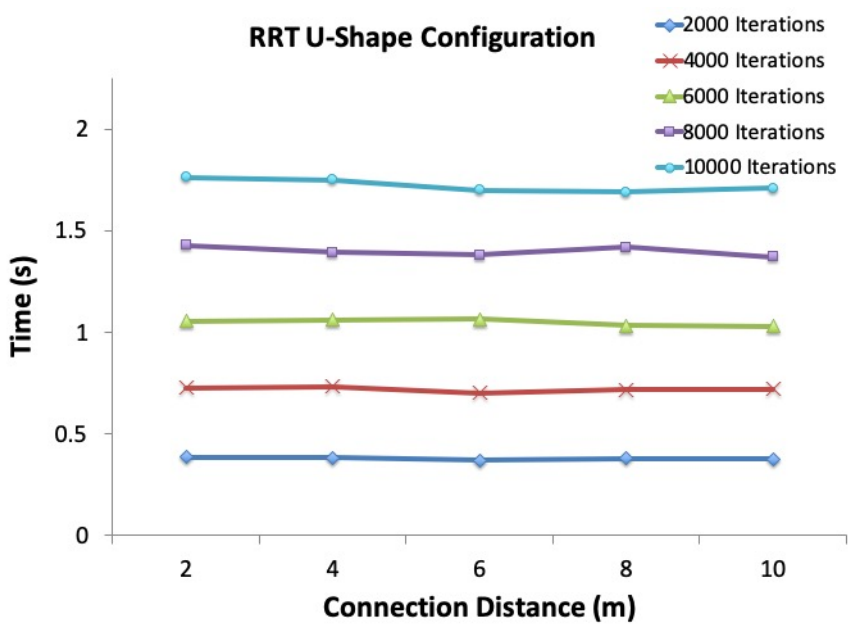


(b)

Figure 2.15. Simulation results for computation time with the 3 obstacle configuration



(a)



(b)

Figure 2.16. Simulation results for computation time with the U-shape obstacle configuration for (a) PRM and (b) RRT

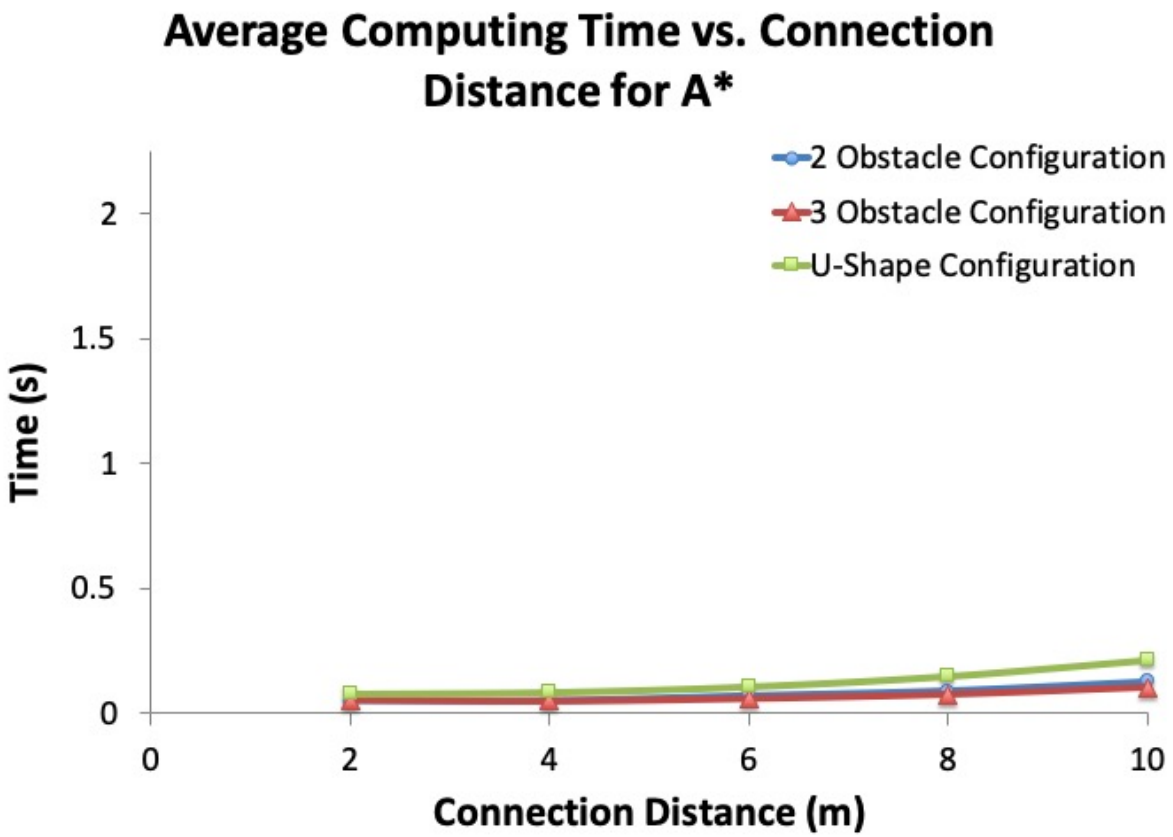


Figure 2.17. Simulation results for each obstacle configuration generated by the A* algorithm

ally in MOOS to represent their location, and are not physically in the testing course. Testing is performed at Swains Lake in Barrington, NH.

Experimental results show successful navigation for each algorithm about each type of obstacle configuration (Figure 2.18). Smoothest trajectories are accomplished by the A* paths as the track displays a fairly consistent wave pattern. PRM also remains close to the intended waypoints, but the vehicle is observed to steer mostly on one side of the path. For the RRT method, the vehicle is unable to follow the sharply placed waypoints precisely, but still displays proficiency in navigating the path in a collision-free manner.

2.5 Discussion

Simulation and experimental testing compared fairly well. As expected, vehicle dynamics and environmental factors caused variance in following planned paths. However, proof-of-concept was still shown for path-following robustness with a roughly tuned controller on an underactuated vehicle. An algorithm that embraces change in dynamics more readily and can show adaptability for different platforms is critical for a modular system. Ideally, parameter changes are minimized transferring, so to speak, the autonomous system from one platform to another. Vehicle dynamics, of course, must be taken into account, but by providing wide margins for control gains, the controller performance burden is eased. Adding a path smoothing function via trajectory planning would help create feasible waypoint navigation as well.

Longer connection distances will give straighter paths overall, which is ideal for maneuverability but must also be balanced with computation time. Out of the three path planning algorithms in the study, A* is found to be overall the most effective in determining optimal paths followed by PRM. RRT consistently calculates more jagged paths (i.e. least direct paths) even with higher numbers of iterations and connection distances. A* has a heavy memory requirement, but when used as a global planner, this method does not require frequent updates. A* may offer additional ease of use since the connection distance is the only parameter that requires tuning.

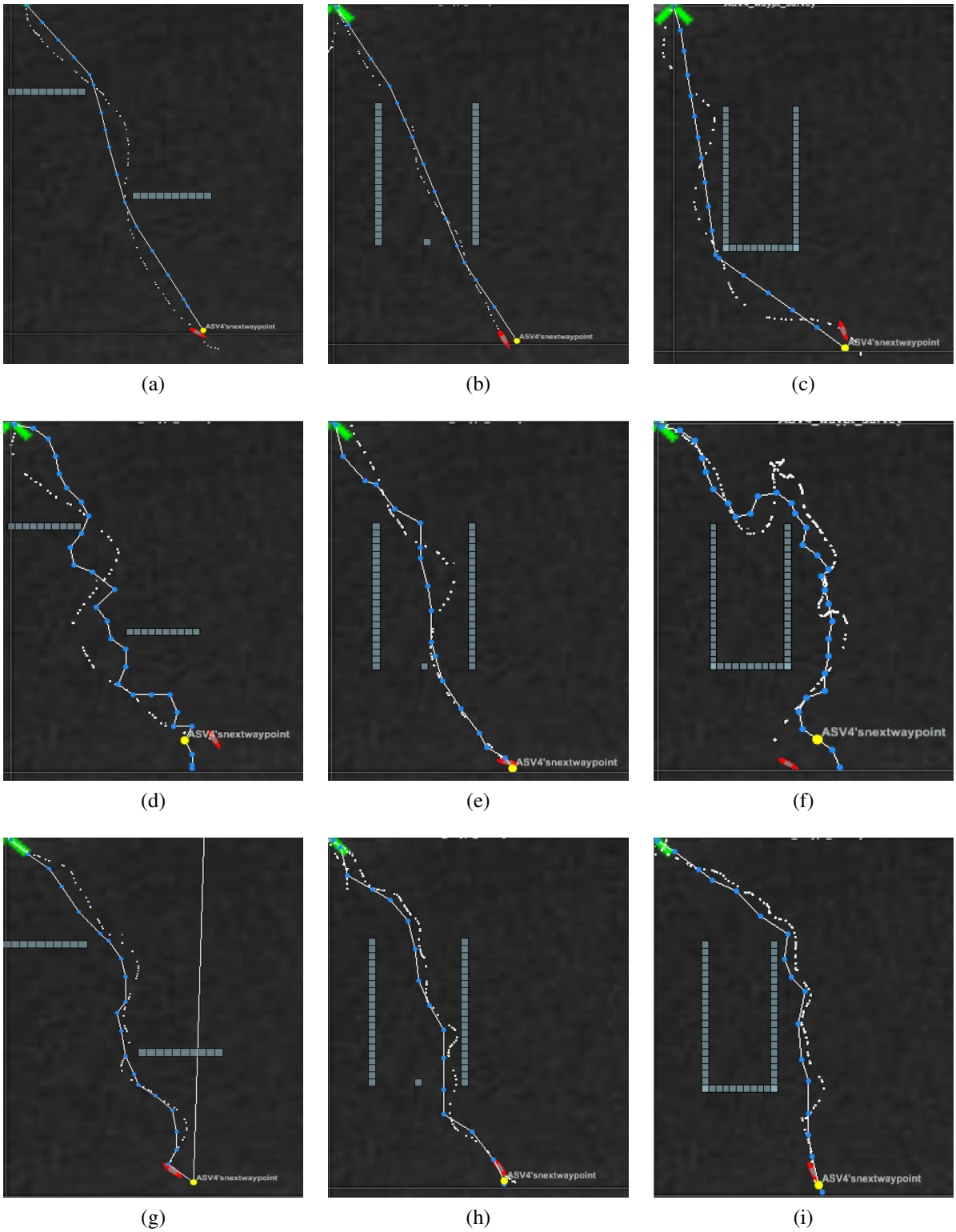


Figure 2.18. Experimental data from test runs at Swains Lake in Barrington, NH of the three different algorithms for each obstacle configuration. Figures (a)-(c) show A star results, Figures (d)-(f) show RRT results, and Figures (g)-(i) show PRM results. The blue dots represent the planner generated waypoints and the white dots are the vehicle's position during the test run.

2.6 Conclusion

A comparative analysis of three different global planners (A*, RRT, and PRM) were explored via simulation and experimental testing to investigate planner tendencies and readiness for real-world scenarios. Three different obstacle configurations were analyzed for situational performance. Numerical simulations and field testing demonstrated proof-of-concept of path following robustness with a generic ASV with a roughly tuned steering control system. Given appropriate path planning parameters (i.e. connection distance, number of nodes, and iterations), numerical simulations show that all three algorithms were able to find feasible paths. Each set of waypoints generated were successfully transited by the ASV platform during field testing. A* and PRM methods calculated paths inherently easier to follow, resulting in smoother trajectories from the start to goal coordinates than that of the RRT method. As a result, A* and PRM displayed the most promising potential for integration with a local planner, particularly to take into account new obstacles and any resulting deviations from the initial globally planned path.

A* is ultimately chosen and utilized for the development of a hybrid planner (presented in the next chapter). A graph search algorithm instead of a sampling based planner was chosen as desirable due to how the author intends to build the environment using occupancy grids. With the expectation that the test environment may be large, discretizing the space is advantageous. Assuming the vehicle is constrained to the surface (i.e. 2-D space), high dimensions do not need to be taken into account which are strengths of PRM, RRT, and sampling algorithms in general. With fewer tuning parameters, more predictable output (due to no random sampling), and cost-based functionality, A* is decidedly the best fit for the research goals in this study, which will be discussed in more detail in the following chapters.

CHAPTER 3

DEVELOPMENT OF A HYBRID GLOBAL LOCAL PLANNER

3.1 Background

Significant work has been done using A* [48](as discussed in Section 2.2) and the Potential Field Method (PFM) [62] separately in the field of robotics. First introduced by Khatib [62] in 1986, the PFM has been applied and further developed with various mobile robots since that time. When applied to the ocean environment, several other challenges must be taken into account. Waves, currents, and other environmental factors lead to difficulty in vehicle control. A path planner requires robustness against changing conditions and new hazards. Additionally, vessels must obey the International Regulations for Preventing Collisions at Sea (COLREGS). The PFM can provide real time obstacle avoidance for local planning, but it can also act as a global planner.

Song et. al. [116] used weighted attractive and repulsive field construction and a Multi-layered Fast Marching (MFM) method to account for currents and wind in a comprehensive environmental framework. A paper authored by Wang et. al [124] also looked at environmental factors using potential field techniques to improve energy consumption for USVs. Xue et al. [130] applied the PFM to find safe passage for ships in possible collision situations. COLREGS constraints are added via a repulsive potential field function in work presented by Lyu and Yin [80]. To date, minimal work has been done and applied comprehensively in experimental testing in addition to accounting for practicality of implementation on real platforms [43].

The foundational concept of the PFM relies on attractive and repulsive forces, much like the concept of electrical charges. In PFM, obstacles and hazards act as a repelling force, while goal locations act as an attractive force. Potential energy increases with proximity to obstacles and decreases near goal locations. Therefore, in this artificially constructed field, the resulting force

causes the robotic platform to navigate from high potential energy to low potential energy [132]. A major advantage of the PFM is its low computational cost even while using complex potential functions [105].

The sum of the attractive and repulsive potentials gives the total potential of the robot where *U(q)* denotes the total potential, $U_{att}(q)$ the attractive potential, and $U_{rep}(q)$ the repulsive potential such that

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{3.1}$$

$$U_{att}(q) = \frac{1}{2}\alpha d_g^2(q) \tag{3.2}$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\beta(\frac{1}{d_o(q)} - \frac{1}{s})^2 & \text{if } d_o \leq s \\ 0 & \text{if } d_o > s \end{cases} \tag{3.3}$$

A parabolic well is chosen as the attractive and repulsive potential function given in (3.2) and (3.3), respectively, where α and β are user-defined design parameters. The attractive equation is a function of the distance, $d_g(q)$, and angle to the goal location. The repulsive equation is dependent upon the distance, $d_0(q)$, and angle between the robot and the obstacle. s refers to the range of influence of the repulsive field has around the obstacle. Outside of this range, the repulsive value is considered to be zero. In practical implementation, the range of influence is equal to the maximum reliable range of the obstacle detection sensors in use.

By taking the negative gradient of the potential $U(q)$, the resulting force is:

$$\vec{F}(q) = -\Delta U_{att}(q) - \Delta U_{rep}(q)$$

$$= \vec{F}_{att}(q) + \vec{F}_{rep}(q) \tag{3.4}$$

$$\vec{F}_{att}(q) = -\alpha d_g \hat{d}_g \tag{3.5}$$

$$\vec{F}_{rep}(q) = \begin{cases} \beta(\frac{1}{d_o(q)} - \frac{1}{s})(\frac{1}{d_o^2(q)})\hat{d}_o & \text{if } d_o \leq s \\ 0 & \text{if } d_o > s \end{cases} \tag{3.6}$$

Here, $\vec{F}(q)$ is a vector representing a direction and magnitude of force felt by the robotic platform. $\hat{d}_g$ and $\hat{d}_o$ are unit vectors pointing in the direction of the goal and obstacle, respectively. The attractive and repulsive components of the force correspond to $\vec{F}_{att}(q)$ and $\vec{F}_{rep}(q)$. $\vec{F}_{att}(q)$ and $\vec{F}_{rep}(q)$ can be calculated for each location in an occupancy grid to create a vector field map, or in the case of local planning, only for the current position of the vehicle. The resulting occupancy grid is a discretized grid space where each cell in the grid is either occupied (contains an obstacle or hazard) or unoccupied (free space). Motion of the robot is produced in small steps in the direction and magnitude of the force dependent on the vehicle's current location. The user may adjust the attractive and repulsive forces (or potentials) accordingly (i.e., being more conservative or aggressive) to take into account any existing unmodeled disturbances/uncertainties by adjusting alpha and beta appropriately.

3.2 Methods

3.2.1 Proposed Hybrid Implementation

A* and the PFM are stand-alone path planners. In this study, they are used in a hybrid implementation. The goal is to utilize both planners' strengths, while reducing weaknesses inherent to each. The vector field capability of the PFM is the basis for integrating the two algorithms. This allows for adding several "layers" of vector field maps that are combined using vector summation. Other layers could include various knowledge about the vehicle's environment such as wind and currents, various classifications of obstacles, COLREGS, etc. Figure 3.1 demonstrates a visual representation of stacked two-dimensional vector fields.

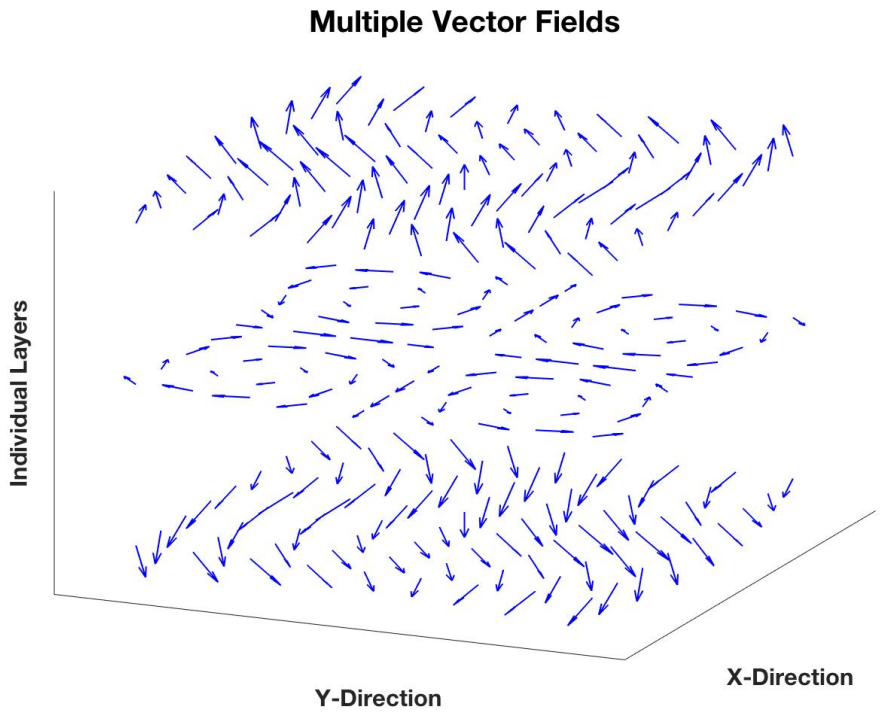


Figure 3.1. Visual representation of multiple two-dimensional vector fields

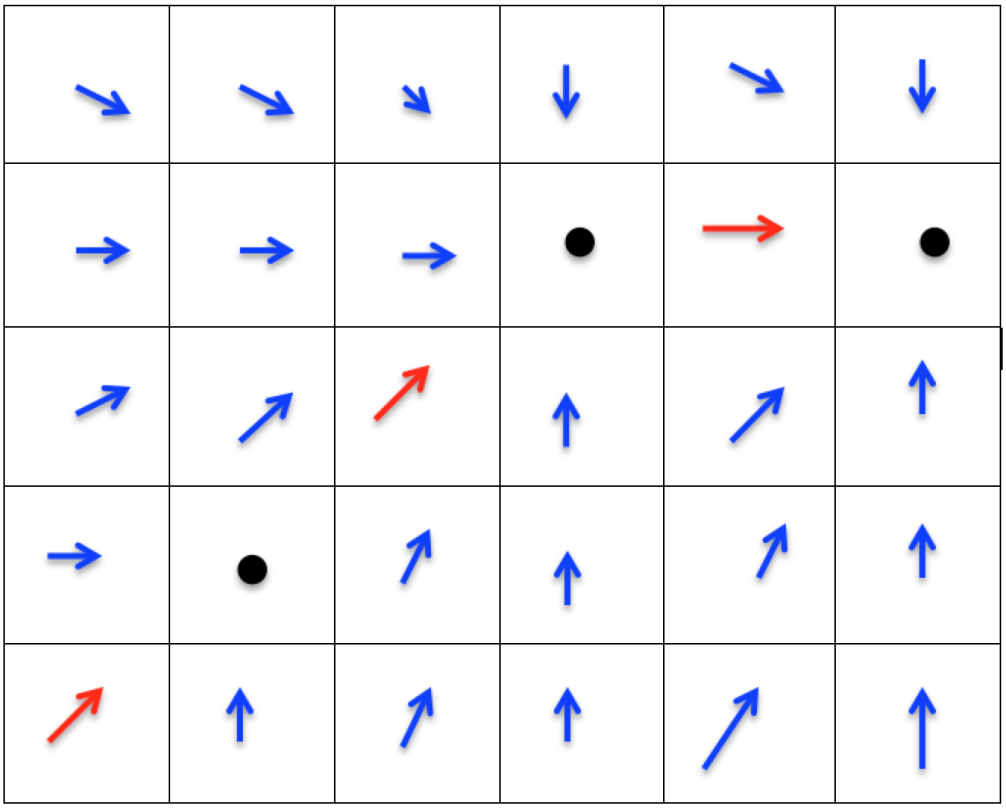


Figure 3.2. Example of the field generated from the A* planned path: waypoints (black circles), vectors along the path chosen by A* (red arrow), and the vectors attracted to the A* path (blue arrows)

This layered method allows for the flexibility to balance and weight the relative importance of each layer in relation to one another. This weighted hierarchy is critical, as in some scenarios, such as when another vessel disobeys COLREGS, there is no concrete “correct" decision and a combination of factors must be accounted for. The final trajectory update must be based on current knowledge of the vehicle’s “world” and the hierarchy of priorities accompanying the vessel to provide an acceptable path-planning decision.

A* is traditionally known as a computationally costly offline (global) planner. However, since the global map is generated a priori, computation time and memory usage is less critical. As a result, a matrix of heading and speed values are produced for use on the vehicle platform and no further computations (or memory space for computation) are required. Unless obstacles are inflated

in the configuration space, A*-generated paths typically tightly follow the obstacle perimeter. Due to the use of Euclidean distance to calculate obstacle distance (to reduce computational complexity) and the nature of the A* algorithm, paths with sharp corners between waypoints are often generated depending on the connection distance chosen.

Two classic problems with PFM occur with U-shaped and narrow-passage type obstacle configurations, as both configurations often suffer from local minima issues. The PFM is unable to guarantee that its minimum found is the global minimum, thus leading to the possible entrapment of the robot due to the attractive and repulsive forces effectively canceling each other [64]. In narrow-passages (i.e. a river environment), PFM also tends to create an oscillatory trajectory as the robot “bounces" from side to side [64].

Past studies and methods have proposed approaches to solve these issues. One method uses Laplace’s Equation to impose constraints on the functions used in order to avoid spontaneous creation of local minima [29]. Guerra et. al took a controls theory perspective, using an established Input-to-State Stability (ISS) property to avoid unstable equilibria [44]. Another example uses a “virtual hill” concept to escape local minima situations by repelling the robotic platform from the minima locations [101].

In this study, an implementation of the hybrid planner includes both the global and local planning components. Here, the operation area is subdivided into a binary occupancy grid. This will still allow for probabilistic methods to be applied in the future for determining obstacles and incorporating sensor models with the flexibility to switch to a probabilistic occupancy grid. The combined A* and PFM planner uses previous knowledge of an area to plan an initial path. This path is then updated in real time for newly discovered obstacles by adding new repulsive forces to the current vector field.

The A* vector field is created by first producing waypoints to the goal location based on A* planning. A vector field is then calculated that is attracted towards the generated path:

$$F_{A^*} = -k_A d_{wpt} \hat{d}_{wpt} \tag{3.7}$$

Each grid location is attracted to the next closest waypoint. F_{A^*} is the force produced from the A* layer, k_A is a design parameter, d_{wpt} is the distance between the current grid location and the next closest waypoint, and $\hat{d}_{wpt}$ is a unit vector pointing in the direction of the next closest waypoint. Fig. 3.2 is an example of the generated A* field.

The proposed method is designed so that with the added bias towards the A* path, the combined planner will alleviate common local minima issues. In addition, the PFM will generate paths with "smoother corners," creating an easier trajectory to follow than that of A* alone.

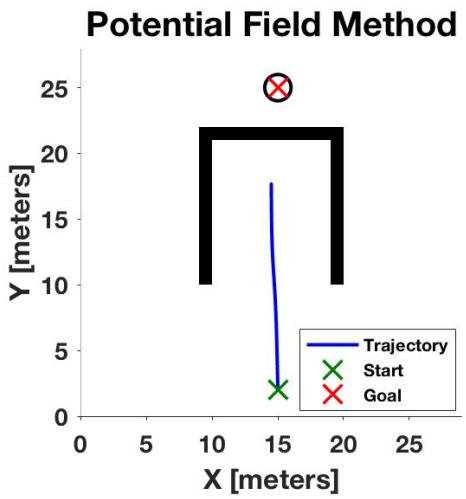
3.3 Simulation Results

Four different obstacle scenarios are simulated: 1) U-shaped obstacle 2) narrow-passageway 3) application to Portsmouth Harbor (NH) and 4) obstacle with different starting locations. The design parameters (α, β, and k_A) used in the proposed path planning method for each scenario are provided in Table 3.1. These values are chosen through trial and error as there are currently no techniques to optimize value selection.

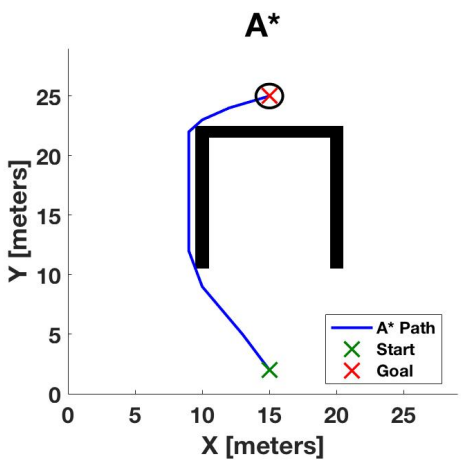
Table 3.1. Gain Values for Each Simulation Scenario

Obstacle Configuration	α	β	k_A
U-Shaped	10	1,000	600
Narrow Passageway	10	1,000	600
Real Environment	100	20,000	1000
Path Convergence	100	15,000	600

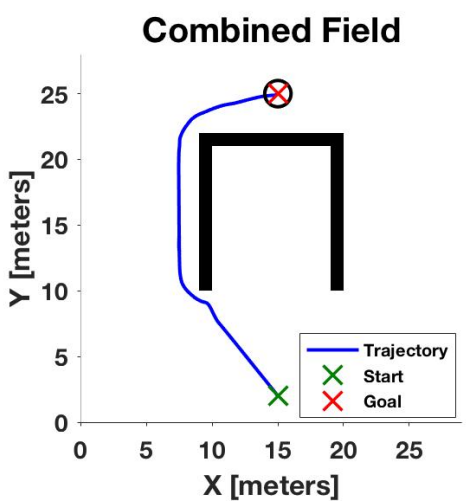
In Scenario 1, the results of the Potential Field Method (PFM), the A* method, and the proposed hybrid path planner applied to a U-shaped obstacle configuration are shown in Figure 3.3. As expected, the PFM encounters a local minima and becomes trapped. However, with the addition of the A* generated vector field, the combined layers of the proposed hybrid method allows the vehicle to successfully reach the desired waypoint. This combined trajectory takes a more conservative path around the obstacle and also results in a smoother given trajectory.



(a)

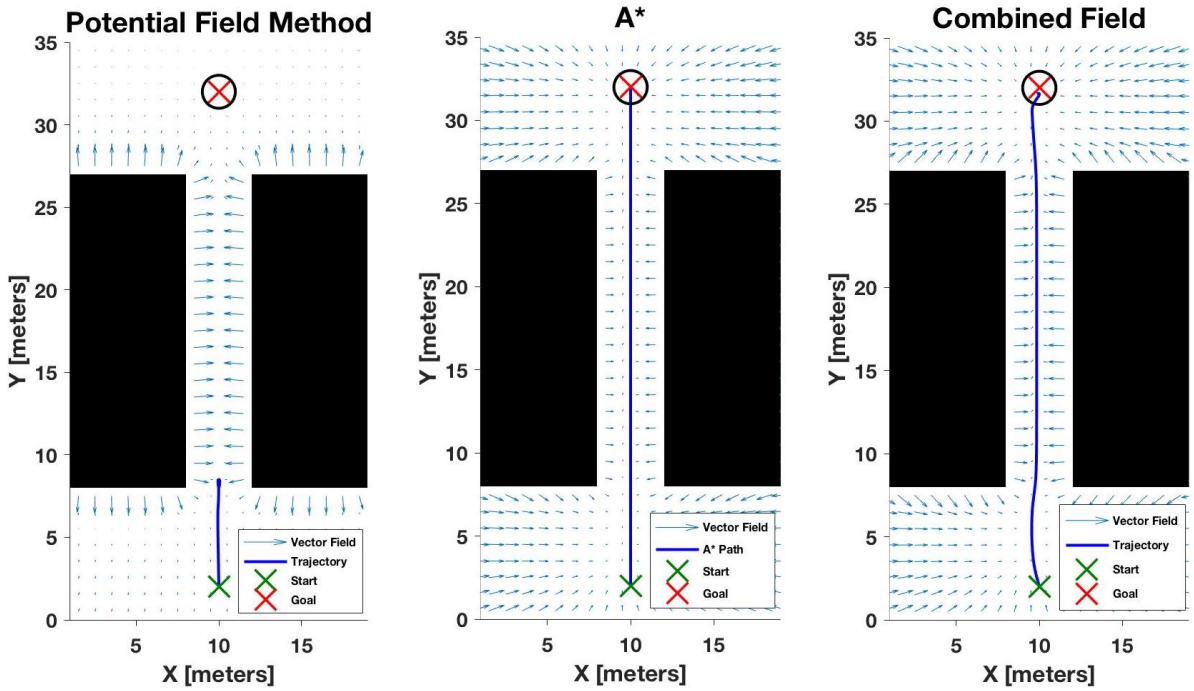


(b)



(c)

Figure 3.3. A simulation of the U-shape scenario (Scenario 1)



(a) (b) (c)

Figure 3.4. A simulation of the narrow passageway scenario (Scenario 2)

The path generation for the narrow-passageway configuration of Scenario 2 is applied and the results of all three path planners are shown in Figure 3.4. Similar to the results of Scenario 1, using the PFM alone the vehicle platform suffers from entrapment in a local minimum upon the entrance to the passageway is approached using the PFM alone. With the aid of A*, a path is created that is biased towards the optimal A* algorithm waypoints and helps guide the robot to the goal circle of acceptance.

In Scenario 3, the Portsmouth Harbor environment is shown in Figure 3.5, where a Google Map image (Figure 3.5a) is converted to a binary occupancy grid to differentiate between occupied and free space. A start point and desired waypoint are determined such that the planner must provide a safe route without a direct line of sight as it navigates around the land mass. Figure 3.5(b)-(d) presents the results of all three respective path planners. Again, due to entrapment, the PFM struggles to exit the cove. A* generates a path to the desired waypoint, but calculates a route

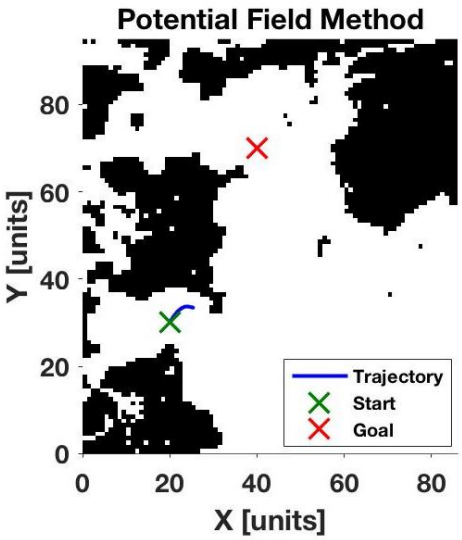
very close to shore. By vector summation of the A* vector field with the PFM generated vector field, a path is found consisting of a smooth path that takes a conservative route around the coast.

The hybrid A* and PFM strategy is also compared to PFM individually to explore different trajectories and path convergence from each grid location on the map with results given in Figure 3.6. It is noted that A* individually is omitted as it does not provide any additional knowledge to the comparison. As presented in Figure 3.2, the A* vector field layer is attracted to the generated A* path. Exploring path convergence, it is found that, generally speaking, the PFM and the combined fields produce similar trajectories from the starting location in this obstacle configuration. However, the difference between the PFM and the hybrid planner lies in scenarios where the ASV finds itself separated from this assumed initial starting location. This behavior would be beneficial in scenarios such a seafloor mapping where deviations from the desired path could cause gaps in coverage.

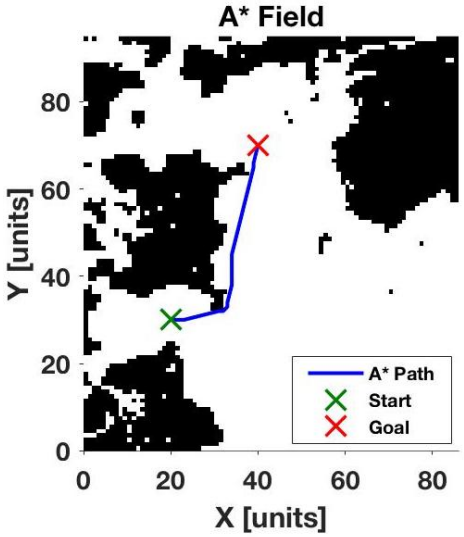
3.4 Experimental Platform Development

For experimental implementation, a small-scale ASV, “TUPPS" (Testing Unmanned Performance PlatformS), is used (Figure 3.7). Two thrusters are used for steering and are controlled with an Arduino microcontroller. Custom designed and built at UNH, this platform went through multiple design iterations and sensor upgrades as research progressed. Platform development is presented first and the corresponding test results are introduced in the next section. Vehicle design goals are centered around an easy to build and replicate platform utilizing low cost components and developing software modularity for rapid testing of different autonomies and sensors. Experiments are performed in the indoor facilities at the Jere A. Chase Ocean Engineering Laboratory at the University of New Hampshire (UNH)(Figure 3.8).

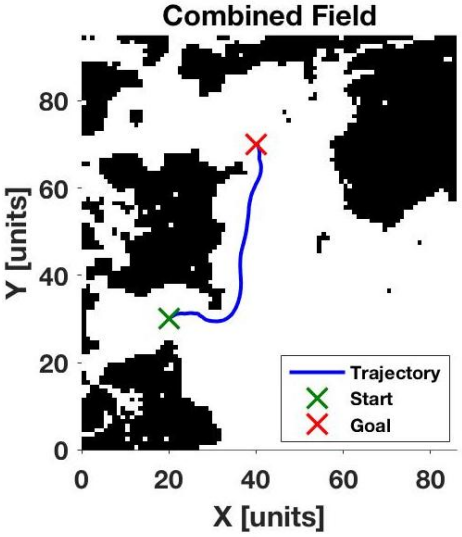
(a) A Google Map image of the entrance to Portsmouth Harbor.



(b) Trajectory generated by PFM

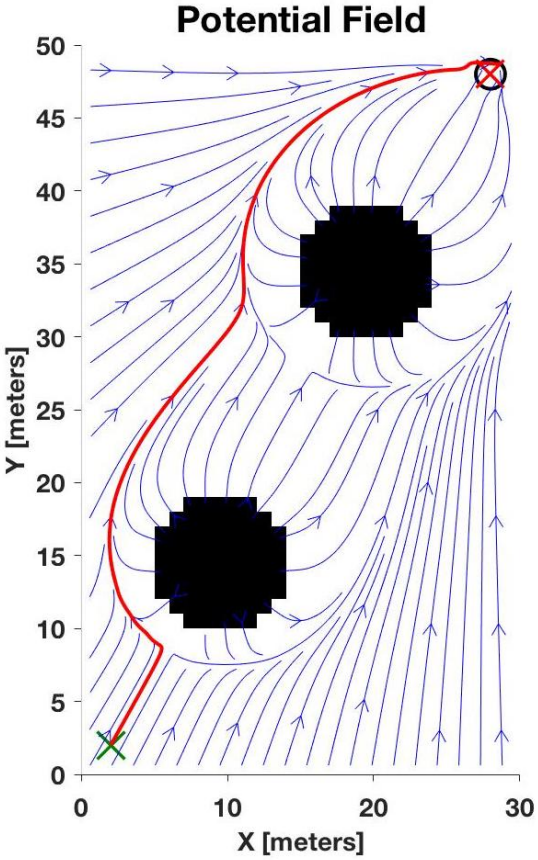


(c) Path generated by A*

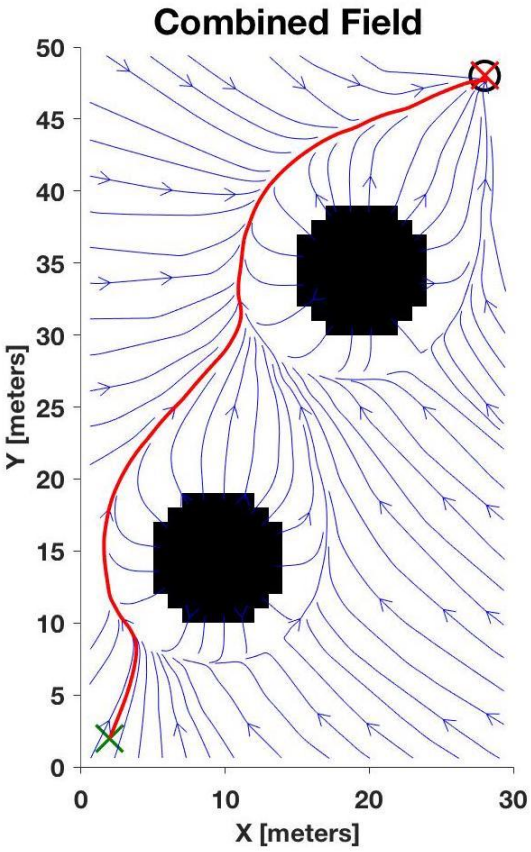


(d) Trajectory generated by combining the PFM and A* layers

Figure 3.5. A simulation of the entrance to Portsmouth Harbor (Scenario 3)



(a) Path trajectories produced from PFM



(b) Path trajectories produced from the summation of the PFM and A* layers

Figure 3.6. Exploring path trajectories from different locations (Scenario 4)

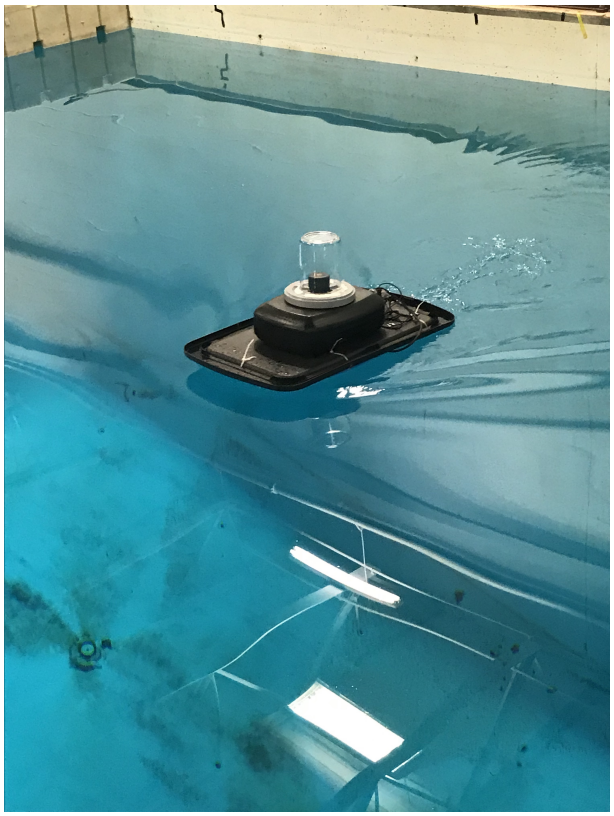


Figure 3.7. TUPPS, a small-scale ASV used for experimental testing

Figure 3.8. The UNH Jere A. Chase Engineering Tank

3.4.1 TUPPS Version 1.0

For the first version of TUPPS [1], electronics include a Scanse Sweep Lidar for obstacle detection, an Adafruit LSM9DS0 Inertial Measurement Unit (IMU) for heading control, and a Raspberry Pi for computation (Figure 3.7). Digi Xbees are used for wireless communication between the Raspberry Pi and an external laptop. Constant magnitude of the thrust output is assumed for experimental implementation and the test runs were run as discrete impulses (with delay and drift).

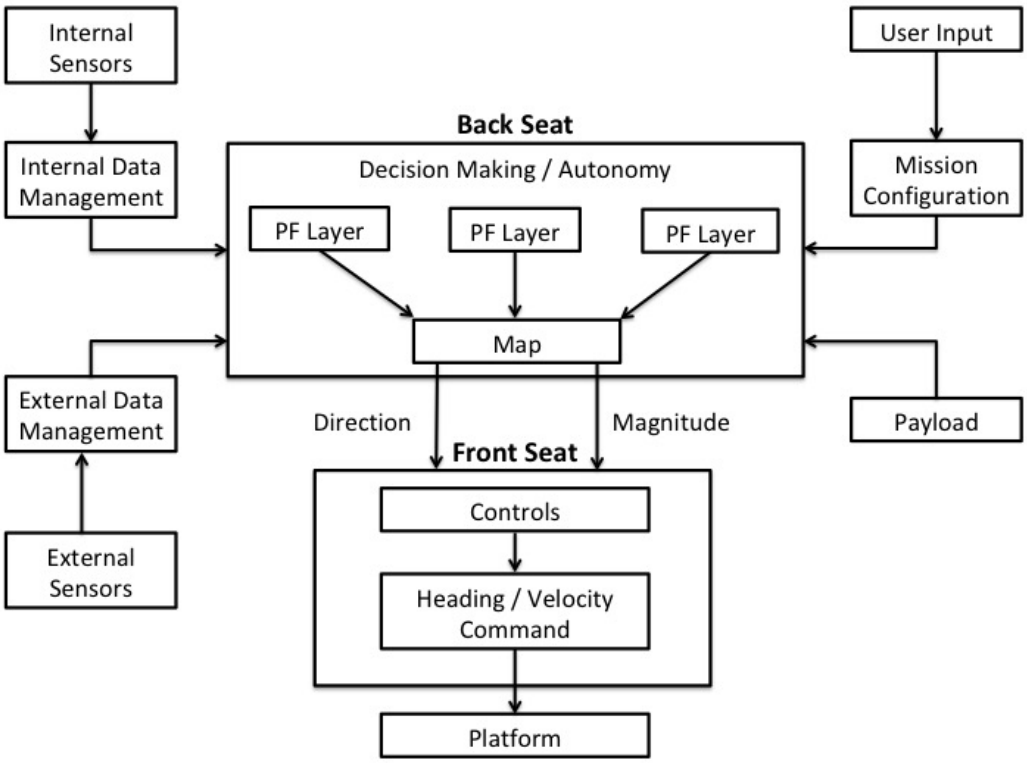


Figure 3.9. Software architecture implemented on the TUPPs platforms

A diagram describing the software architecture is given in Figure 3.9. The Raspberry Pi with custom C++ programming performs the decision-making (Back Seat) duties of the autonomy, while the Arduino is responsible for the controls, and thus the Front Seat module of the autonomous system. Applications are created to parse the raw Lidar data and provide information

[1]Developed with Alexander Cook, UNH Ph.D. Student in Systems Engineering

Figure 3.10. Indoor GPS beacons [2]

to the Raspberry Pi on obstacle distance and relative angle to the ASV. This is classified as external data management (i.e., perception and world model building). The IMU (and/or any other self-localization sensors) and associated processing are organized in the internal data management module.

For implementing this on the experimental platform, a matrix containing heading values corresponding to each grid cell of size $1m^2$ is generated. The platform retrieves a desired heading by indexing into this matrix based on the ASV's current location. The vehicle receives sensory feedback from the IMU in order to determine its current heading and this information is used to output a thrust command using bang-bang control to correct the heading error.

3.4.2 TUPPS Version 2.0

The next iteration of the experimental platform utilizes a new indoor positioning system allowing for better position feedback and data post-processing. This system, developed by Marvelmind Robotics [2](Figure 3.10), uses 4 ultrasonic stationary beacons which are placed on each side of the engineering tank (Figure 3.8), and a mobile beacon placed on the robot. With +/- 2cm precision, the mobile beacon's location is calculated based on a propagation delay of ultrasonic pulses (Time-Of-Flight or TOF) between stationary and mobile beacons using a trilateration algorithm [2].

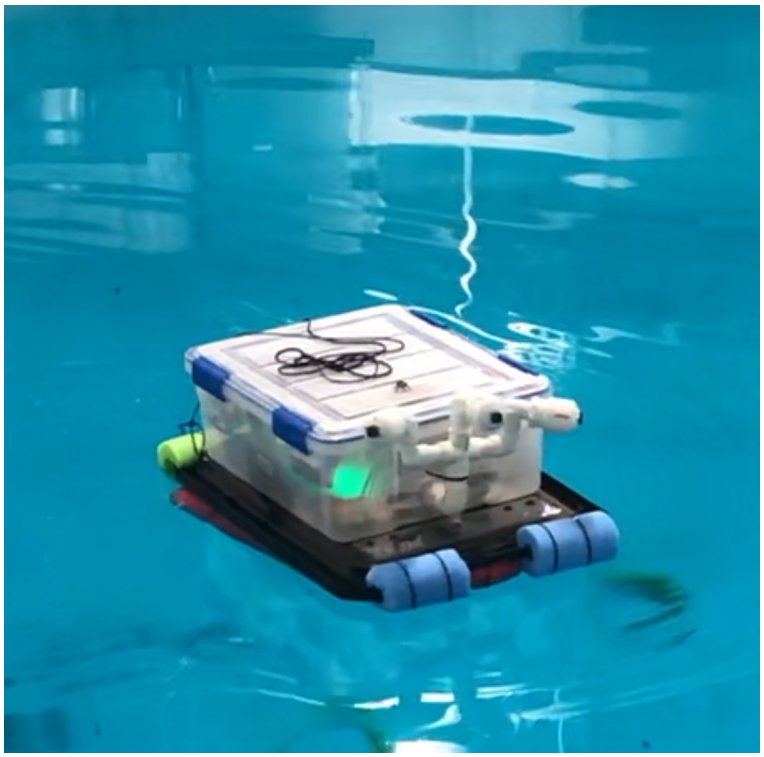


Figure 3.11. TUPPS 2.0

For obstacle detection, the Scanse Sweep Lidar is replaced by 3 ultrasonic range finder sensors facing forward (0°), left (330°), and right (30°). This change was made due to unreliable data generated by the lidar. Simple filters were added to try and improve the data consistency and noisiness, however alternative solutions were eventually needed. The range finder sensors are chosen as the alternative based on documented precision range-finding, low-voltage operation, and low-cost, in addition to reducing data points and therefore processing as well. Due to the beam pattern of the ultrasonic sensors, obstacle detection coverage spanned approximately +/-60° from the centerline of the vehicle (Figure 3.12). This was considered sufficient for the purposes of testing. Additional sensors may be added if a larger field of view is needed. Max range for the range finder sensors is 5m, however this value was cut to 2m to minimize noise.

Other modifications include a small laptop to replace the Raspberry Pi for debugging convenience when testing, Xbees replaced by using Secure Shell (SSH) from the shore station laptop for better communication, and a second Arduino for processing the range sensor data (Figure 3.13).

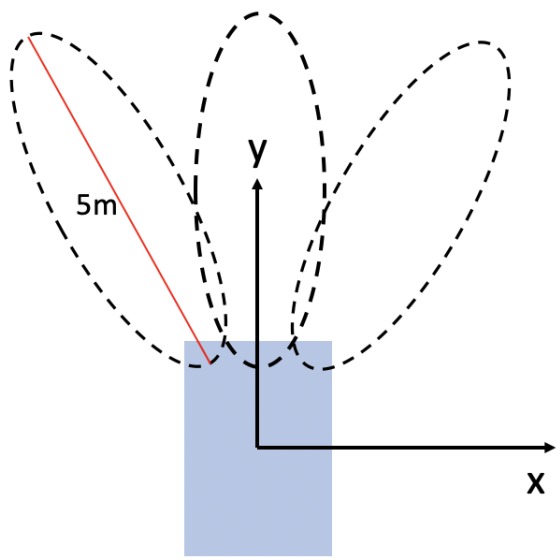


Figure 3.12. Visual representation of the beam pattern for the ultrasonic range finder. Figure not drawn to scale.

Previous code structure is converted to the Robotic Operating System (ROS) for increased code modularity and to act as middleware for generalizing the process of integrating new sensors.

3.4.3 TUPPS Version 3.0

The next iteration of the platform involves minor electrical improvements. Previously, each thruster used a separate battery for power. The setup is reduced to a single battery by using a terminal block and the overall wiring is also consolidated. The IMU is replaced by using 2 indoor GPS beacons spaced 0.2m apart on the vehicle. The amount of rebar in the Chase Ocean Engineering Tank was enough to cause a significant amount of magnetic interference and, in turn, affected IMU reliability.

A major change to the system results from acquiring a Velodyne VLP-16 lidar, a commercial-grade sensing device. A custom 3D printed mount is designed [2] and integrated with the electronics box (Figures 3.14-3.15). Utilizing visualization tools from ROS, example lidar data is given in

[2] With the help of Michael Jenness, UNH ASV Team Member

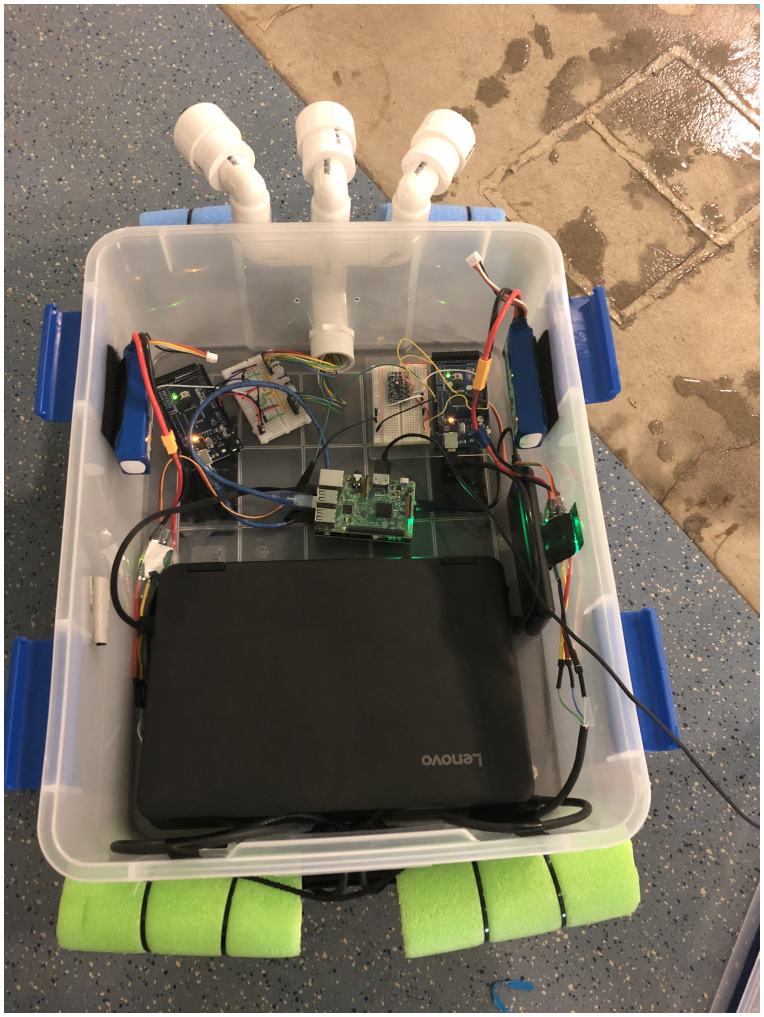


Figure 3.13. TUPPS 2.0 electronics box

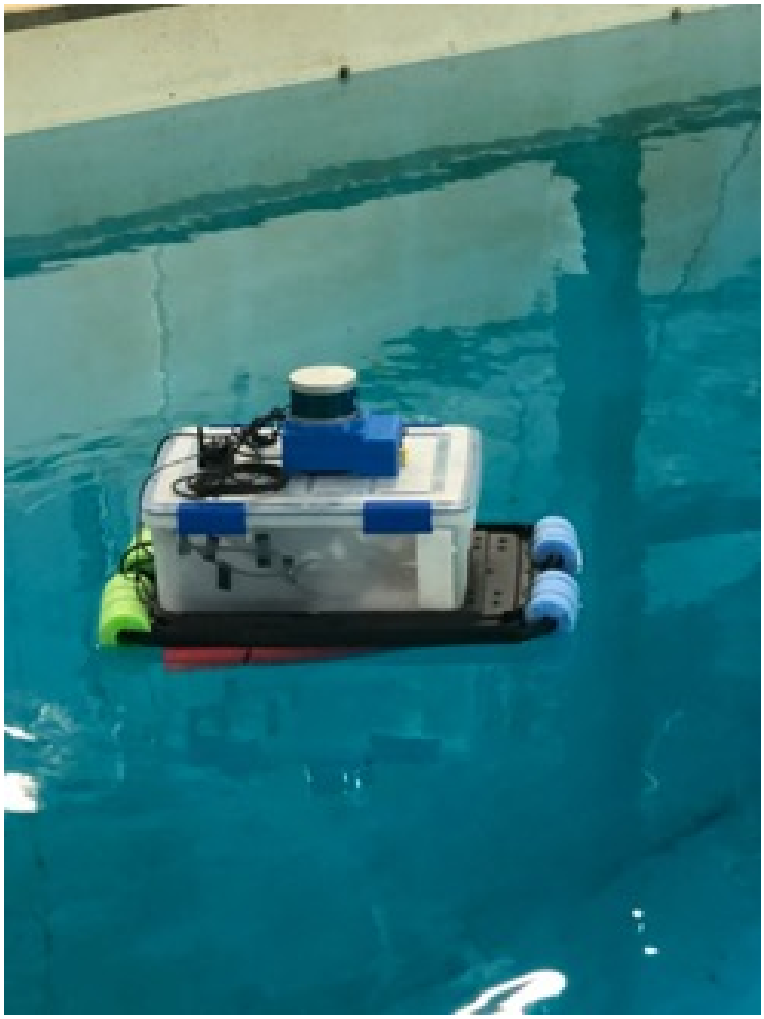


Figure 3.14. TUPPS 3.0

Figure 3.16. Several features can noted from the UNH engineering tank with a 360 degree horizontal field of view and 15 degree vertical field of view. Data below 0 degree vertical field of view is filtered out to reduce noise from the water surface. This noise became a common issue with the range finder sensor on TUPPs 2.0.

To build the occupancy grid, a static and dynamic layer (Figures 3.17 - 3.18) are used and can also be visualized using ROS tools. Here, the static layer contains the a priori knowledge of the area (i.e. the empty tank) and the dynamic layer provides the real time information. Figure 3.18 shows, via the black cells, the occupied regions of the tank ($1m^2$ resolution). In this example,

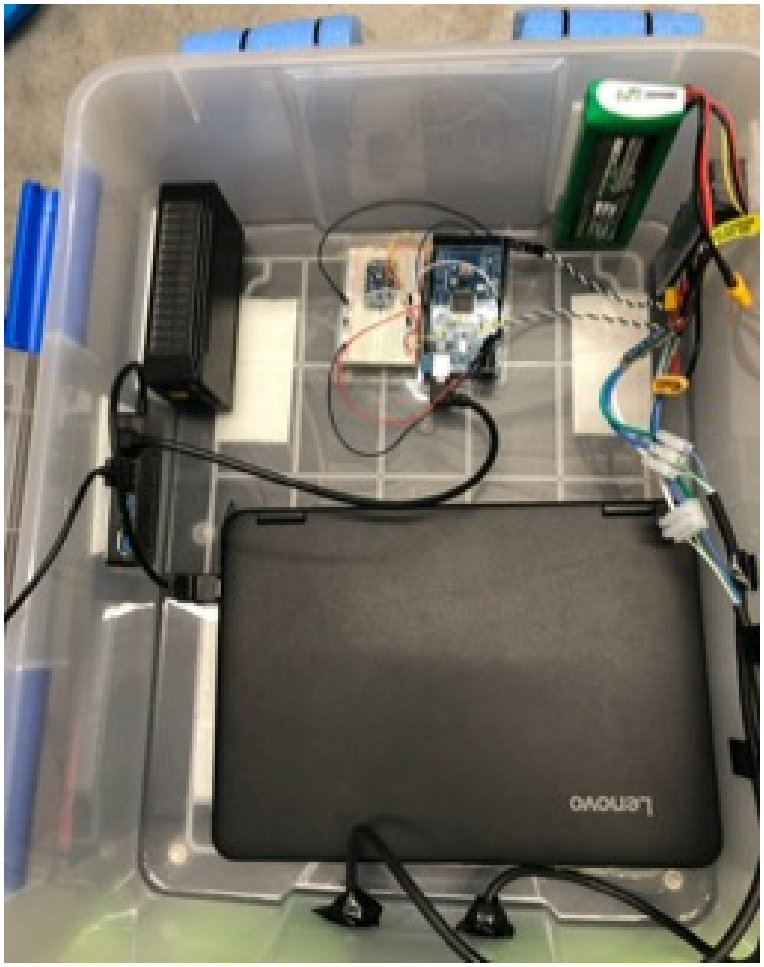


Figure 3.15. TUPPS 3.0 electronics box

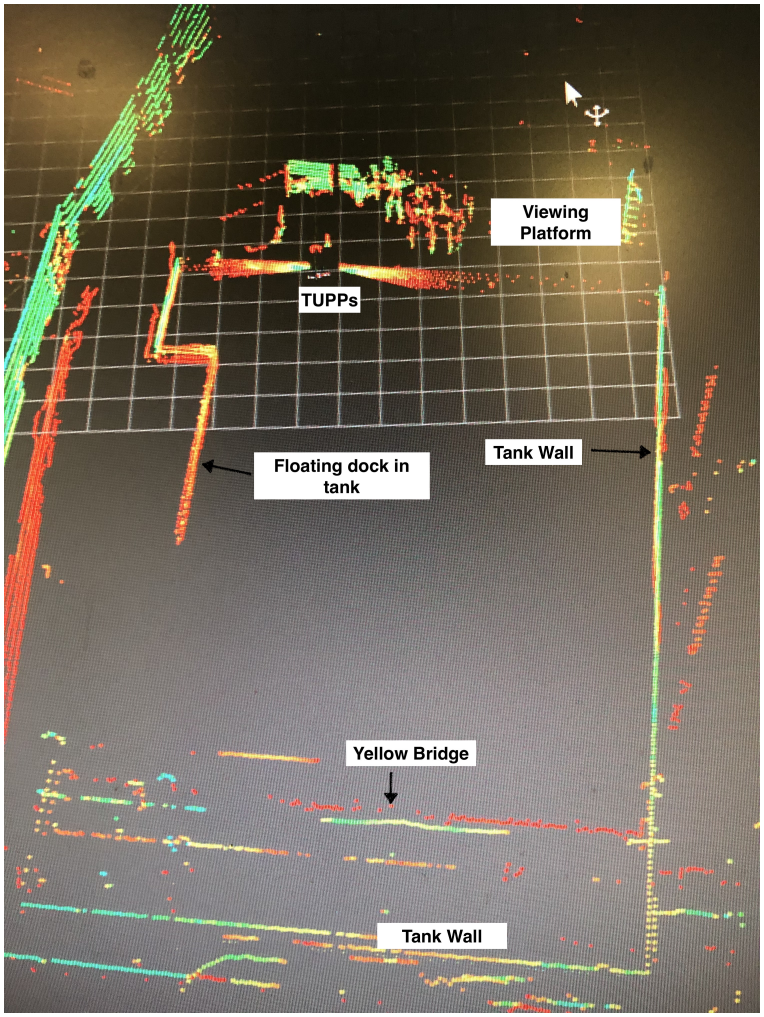


Figure 3.16. Velodyne example data of the engineering tank

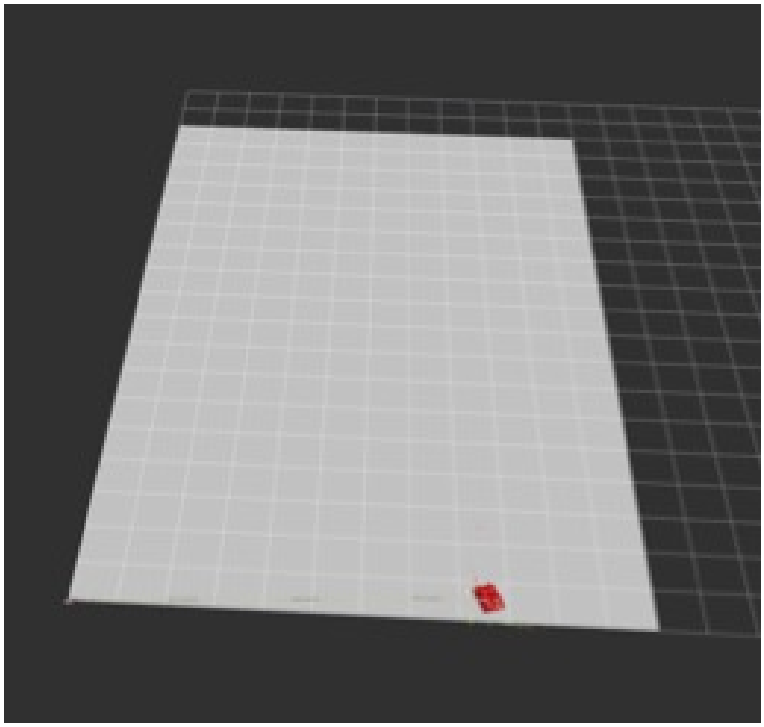


Figure 3.17. Cost Map static

except for the bottom right corner of the image, the occupied regions correspond with the walls of the tank. The bump out in the bottom right corner is from 2 side by side floating docks.

Several parameters may be modified to improve occupancy grid performance. A crucial parameter is the maximum update range, which is the maximum distance from the vehicle that the lidar can "see." When set too high, the data processing becomes too much of a burden for the laptop to compute and provide coordinate transforms in a timely manner. It is found through testing that for a Lenovo N23 laptop with 4GB of RAM and 1.6GHz of processing speed, this value is 8-10m (the true maximum range for the Velodyne VLP-16 is 100m).

The transform tree allows for referencing vehicle and obstacle locations in the world frame (Figure 3.19). The first transform converts the vehicle frame of reference to the world frame. This is notated as T_0^1, the transformation matrix from reference frame 1 (vehicle frame) to reference frame 0 (world frame). The transformation matrix is made up of a rotation matrix, R_0^1 appended with a distance vector, D_0^1:

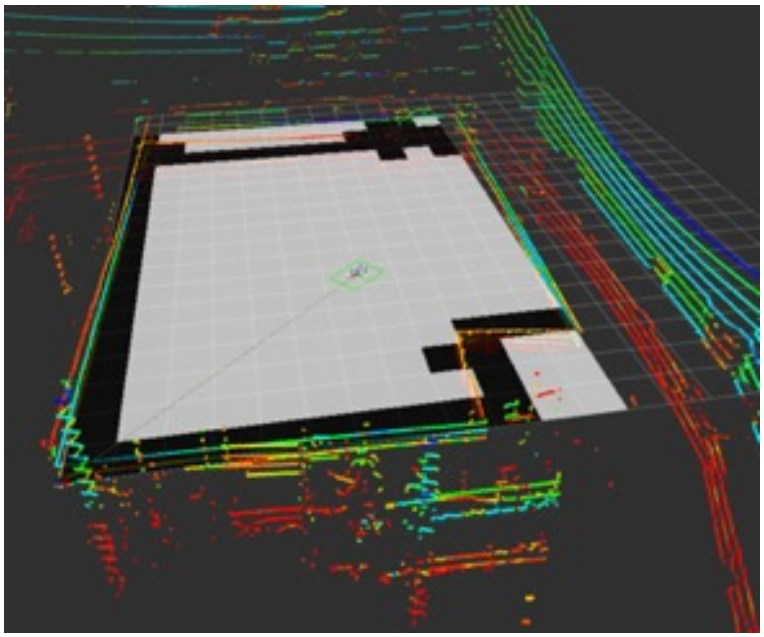


Figure 3.18. Cost Map dynamic

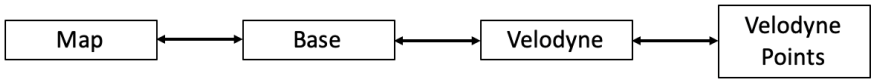


Figure 3.19. Diagram of the transformation tree for experimental testing

$$T_0^1 = \begin{bmatrix} R_0^1 & D_0^1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & x_v \\ sin(\theta) & cos(\theta) & y_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

θ is the angle between the x_0 axis and the x_1 axis (Figure 3.20) which is found using a heading sensor (i.e. IMU). x_v and y_v are the x and y coordinates of the vehicle given from the indoor GPS. A second transformation is needed to account for the Velodyne sensor position on the vehicle. The sensor is centered on the y_1 axis translated 0.2m in the positive y_1 direction. Since there is no rotation and the transformation is static, the T_1^2 matrix, from the sensor frame (2) to the vehicle frame (1), reduces to:

$$T_1^2 = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.2 \\ 1 \end{bmatrix} \tag{3.9}$$

where p_x and p_y are the x and y position of the sensor in the vehicle reference frame. The last transformation is the obstacle location relative to the Velodyne sensor. This becomes another positional vector:

$$P_2 = \begin{bmatrix} ob_x \\ ob_y \\ 1 \end{bmatrix} \tag{3.10}$$

where ob_x and ob_y are the x and y positions of the obstacle relative to the Velodyne sensor. These values are deduced from the range and angle data generated from the lidar. Combined, the equation to transform the obstacle data to the world frame becomes:

$$T_0^1 T_1^2 P_2 = \begin{bmatrix} cos(\theta) & -sin(\theta) & x_v \\ sin(\theta) & cos(\theta) & y_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0.2 \\ 1 \end{bmatrix} \begin{bmatrix} ob_x \\ ob_y \\ 1 \end{bmatrix} \tag{3.11}$$

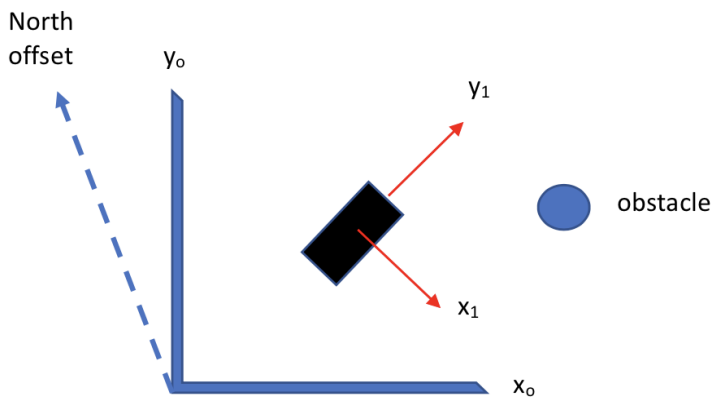


Figure 3.20. Coordinate transforms from the vehicle (local) frame to the map (world) frame

Table 3.2. Gain Values for Experimental Testing

α	β	k_A
100	10,000	200

A Unified Robot Description Format (URDF) file is created for visualizing the platform and its transform components (Figure 3.21). URDF is an XML format for representing a robot model. A custom tool is also created to help visualize and analyze the PFM/A* vector field (Figure 3.22).

3.5 Experimental Results

Gain values used for testing are given in Table 3.2. A path is first calculated assuming no obstacles are in the operation area (Figure 3.23). To test local replanning for new, unknown obstacles, an obstacle is placed in the testing area. Here, the vehicle is required to detect the obstacle, calculate a new heading based on Equation 3.6, update the map, and then steer in the new direction. For the radius of obstacle influence, s, a value of 2m is chosen due to the determined best signal to noise ratio results within the specifications of the Lidar sensor used. Figure 3.24 compares the precomputed trajectory (Figure 3.23) under the assumption no obstacle was present against the simulated and experimentally tested paths generated after obstacle detection and updating the map. This testing was performed with TUPPs 1.0.

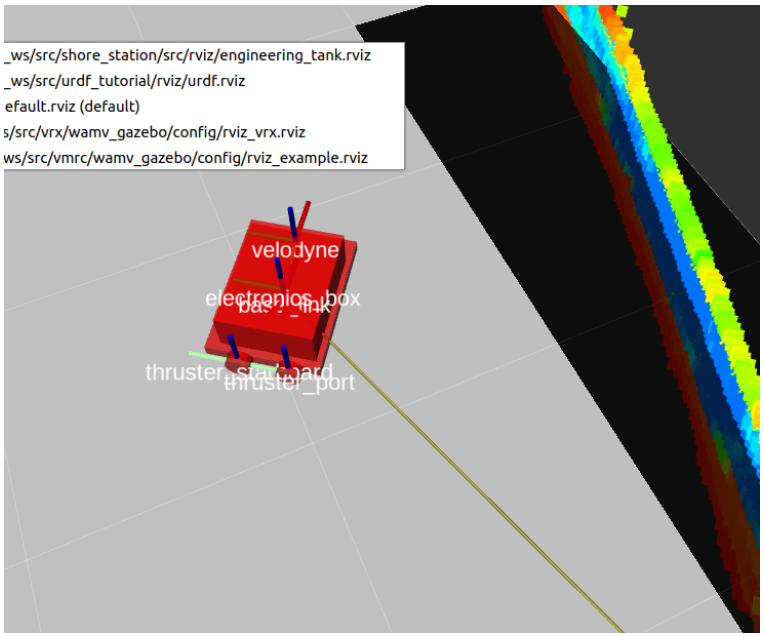


Figure 3.21. Custom URDF file for visualizing the platform and its transform components

Although resolution for the path traveled could only be resolved to $1m^2$ due to lack of an indoor GPS signal and sensor constraints, simulation and experimental trajectories matched fairly well. Jagged paths are produced in data post-processing due to the low resolution and quantity of data points, however it is noted by observation that the vehicle responded to heading controller commands in a timely fashion and in the correct direction based on its current position and matching matrix index value. The vehicle possessed the capability of following the path with minimal control implementation, thus potentially lessening the necessary controller tuning when moving the system to different platforms.

Next, experimental testing focused on the convergence aspect of the algorithm and used TUPPs 2.0. This is first explored via simulation in Figure 3.6. In this scenario, experimental validation is sought to confirm that at different start locations, the path will converge to the intended path starting at (10, 1)(Figure 3.25). This is intended to replicate situations where the ASV is required to stay as close to the intended path as possible while responding to perturbations due to obstacles or environmental factors. Seafloor mapping is one instance that will necessitate this approach. When mapping, track lines must be followed closely to minimize data gaps.

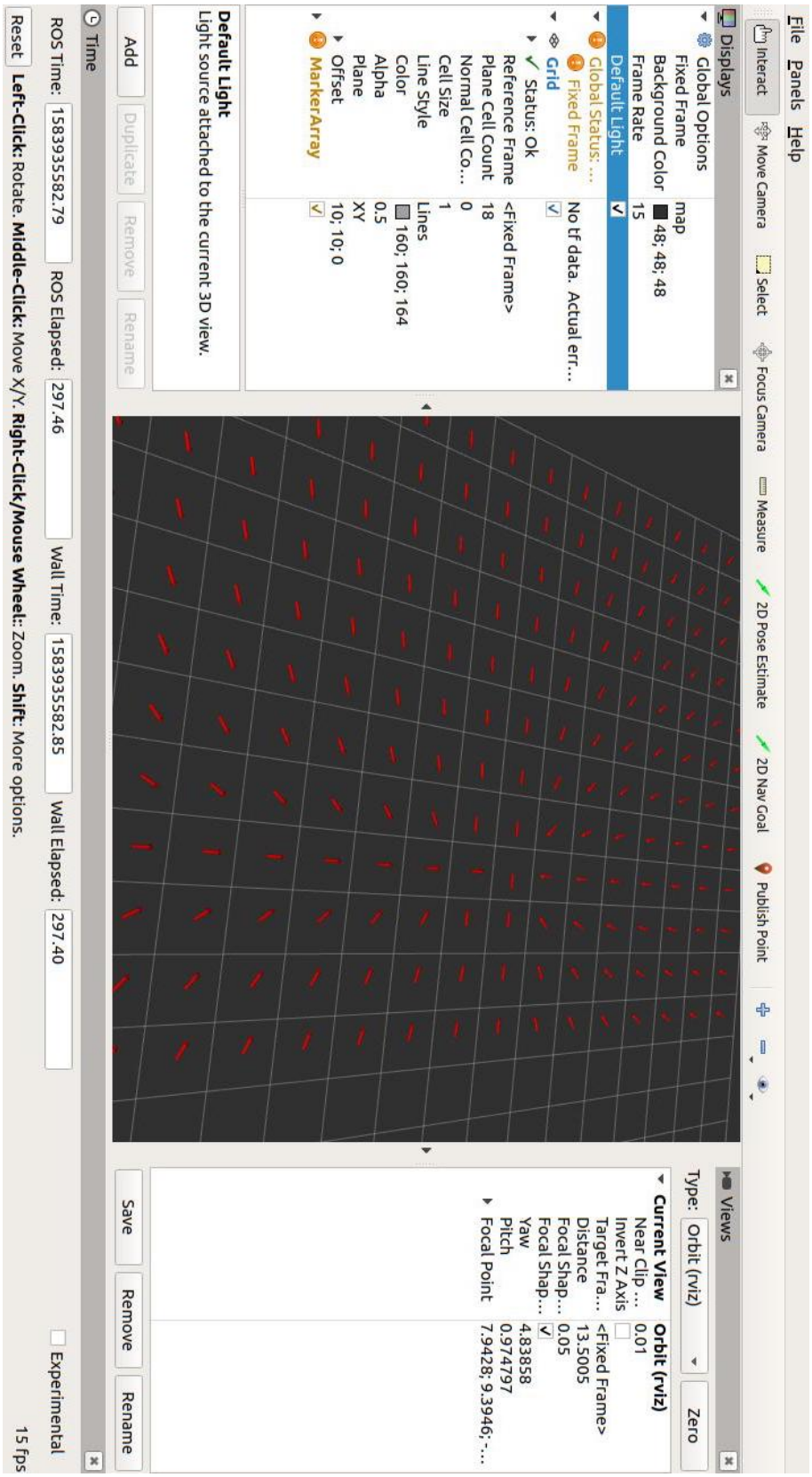


Figure 3.22. Custom ROS tool developed for visualizing the PFM/A* vector field

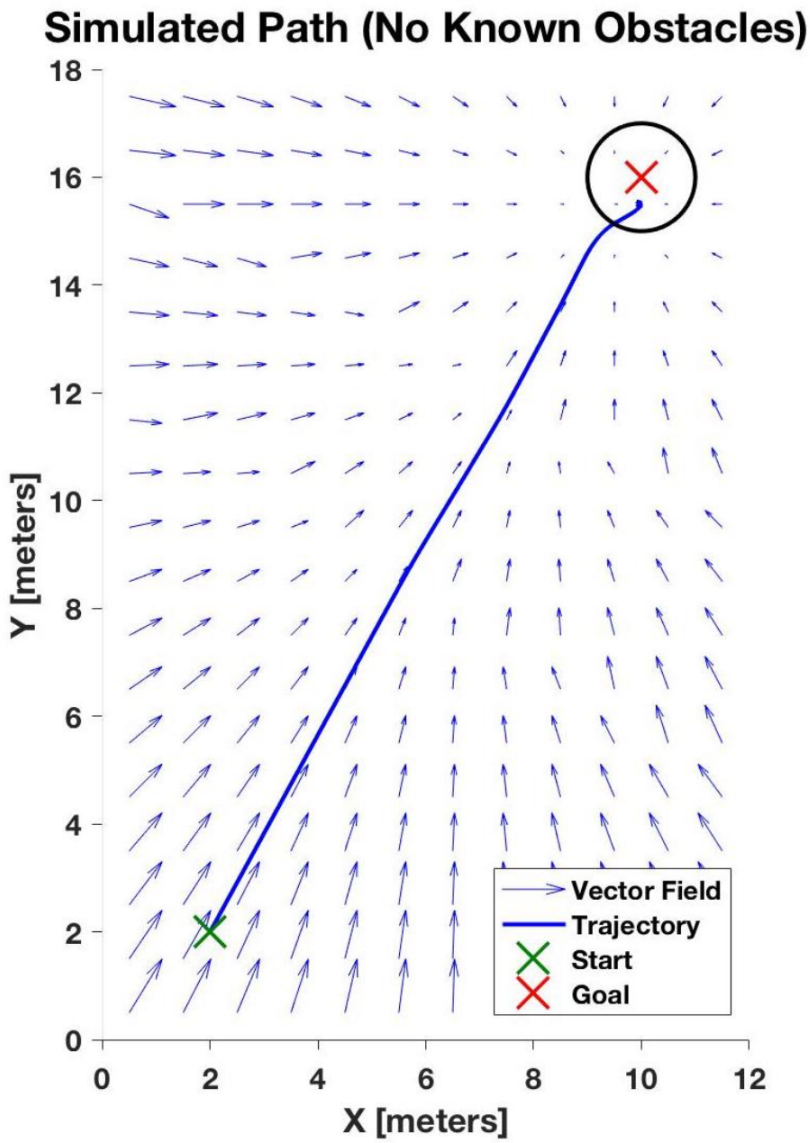


Figure 3.23. The path generated before obstacle placement

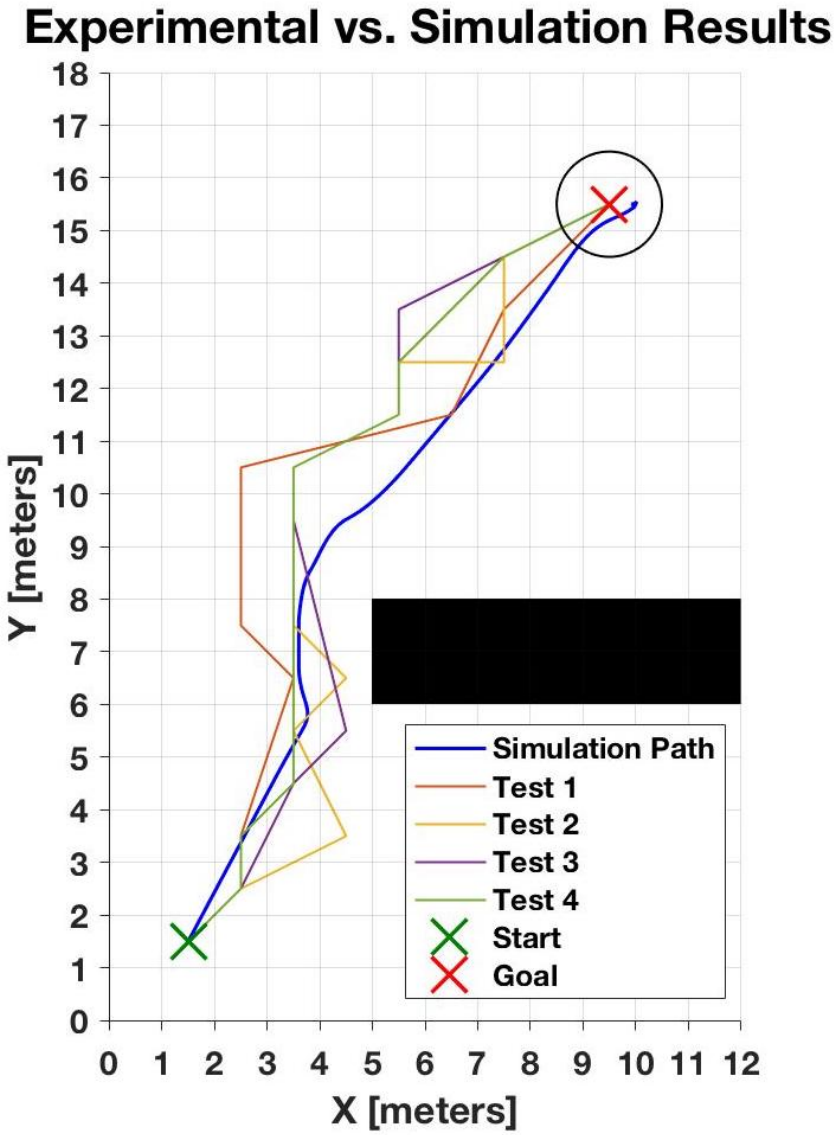


Figure 3.24. Experimental testing results. The region covered by the obstacle is denoted in black. The simulation path is based on the ideal scenario where the obstacle's location is known a priori and precomputed.

Results show consistency between simulation and experimental testing. Figure 3.25d follows the streamlines provided in Figure 3.25. The vehicle also safely navigates around the obstacle when necessary. The indoor GPS displays clear improvement in the data collected. Increased resolution and update rate provides a smoother representation of the vehicle trajectory. A more sophisticated controller would help smooth the tracklines further.

3.6 Discussion

The simulations produced in Figures 3.3-3.5 indicate that when implemented as additive vector fields, A* and PFM complement each other and can construct a successfully planned trajectory to goal. This lends to the potential to layer more fields to include other critical knowledge and situational awareness influences to build a tailored decision-making hierarchy compatible to the user's mission goals.

Referring to Figure 3.6, the combined planner is influenced by the A* optimally found path, and therefore is partial towards returning to this path. This may lead to more predictable autonomy as assumptions can then be made that if vehicle strays from this path due to new obstacles or strong current, it will be incentivized to either remain on or return to the originally planned path. This will also minimize the amount of updating necessary when new obstacles appear by only needing to change the values within the obstacle's region of influence.

Results show that the planner displayed the ability in real time to avoid the newly detected obstacle. It is additionally noted that computation and communication speeds are found to be feasible for practical implementation.

It is found that the hybrid approach presented is highly dependent on the gains chosen. Further work is necessary to normalize and optimize the gain values and find an approach accounting for percentage of area covered by obstacles, the number of layers used, and the balance of importance between the layers.

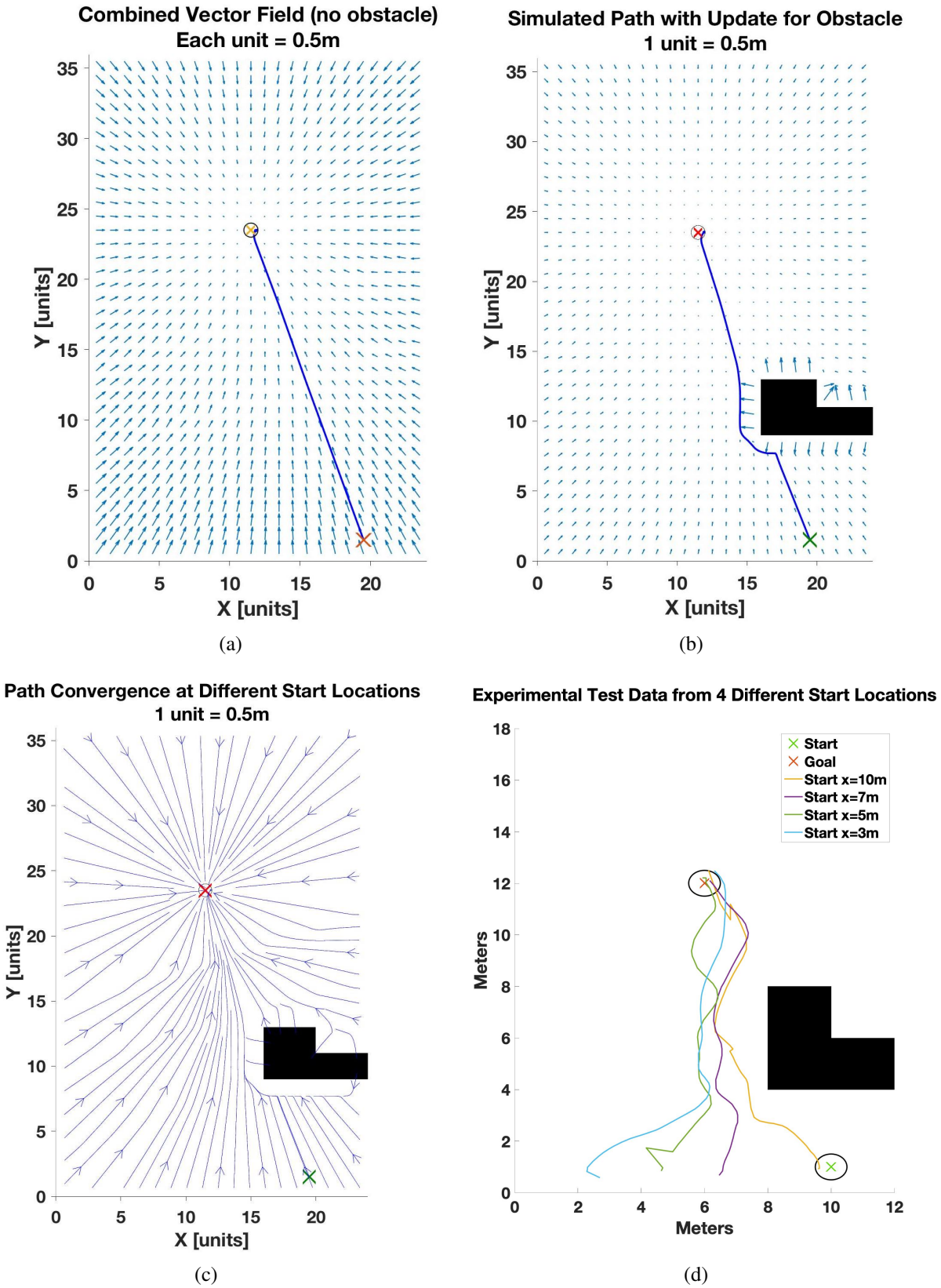


Figure 3.25. Simulation vs. Experimental Results for Analyzing Convergence

3.7 Conclusion

Proof-of-concept testing of a hybrid A* and the PFM path planner was accomplished via simulation and the approach was validated with an experimental platform. Algorithms show promise for use in generating a pre-mission global path, accounting for different decision-making behaviors, and updating the path (if needed) in real-time. Future work will involve further testing in an open ocean environment with a large-scale ASV, adding more layers for different behaviors and environmental influences, and extending the algorithm to three dimensions for the underwater domain. Work will also include exploring ways to optimize the gains for each potential field layer and accounting for vehicle dynamics.

CHAPTER 4

MISSION PLANNING AND AUTONOMOUS SYSTEMS

4.1 Autonomous Systems

4.1.1 Defining Autonomy

In previous chapters, path planning, as a subcomponent of an autonomous system, was discussed. This will now be placed in context of an autonomous system as a whole so that approaches and challenges to validating these systems may be considered. First, to understand the test and evaluation problem, the word "autonomy" should be defined. With origins from the Greek language, *autos* meaning "self" and *nomos* meaning "law", something autonomous essentially means to make its own laws, or, in other words, having independence from outside control [3]. Applied to the realm of autonomous vehicles, this definition more narrowly means that a vehicle is navigated by a computer without a need for human control or intervention under a range of driving situations and conditions. While some subsystems of a vehicle could be considered "autonomous" (for instance, cruise control), this study considers autonomy in the higher-level context of vehicle decision-making. As such, path planning is an aspect of the decision-making process. The ability to plan, update, adapt, and execute a route based on internal and external factors demonstrates a certain level of decision-making proficiency.

True autonomy does not yet (and arguably cannot) exist [51][137], so the actual definition becomes a broader (and more complex) scope of abilities. Thus, "autonomy" in the vehicle domain exists as a spectrum [55]. Levels of autonomy have been previously defined in various ways. However, at this point in time, no consensus on consistent terminology has thus far been accepted [55][26][39][70][46]. Disagreements and challenges are posed in literature, some of which include

how to characterize human/robot-coordinated operations, if and how to include automated robots, and how many levels of autonomy should be included.

Efforts have been made in defining a core ontology for robotics and automation to help develop a common language. An *IEEE* Working Group was formed, referred to as *Ontologies for Robotics and Automation (ORA)*, with the goal of standardizing knowledge representation in the Robotics and Automation field [104]. This work, proposed as the Core Ontology for Robotics and Automation (CORA) in [104] and further extended in [39], defines a standard set of ontologies related to robotics and automation and created the first-ever standard for the IEEE Robotics and Automation Society [39][100]. While these ontologies make progress on defining a vocabulary and formally specifying key concepts and relationships of robotics and automation, they still note difficulty in formally defining degrees of autonomous capability precisely [104][39]. In the CORA framework [39] (Table 4.1), their interpretation of autonomy aligns with the Autonomy Levels for Unmanned Systems (ALFUS) framework [52][53], with the key difference being that CORA does not try to characterize absolute levels of autonomy [39].

The ALFUS framework, designed to improve communication on operational and development issues, analyze mission requirements, and evaluate the capabilities of unmanned systems [53], is one of the most extensive and commonly referenced frameworks presented to date with involvement from several government organizations and over seventeen workshops completed [36][37]. These methods use a contextual model that assesses performance as a function of autonomy level [36]. ALFUS uses a 3-axis approach where levels of autonomy are decomposed into categories of Mission Complexity (MC), Environmental Difficulty (ED), and Human Independence (HI) to specify a Contextual Autonomous Capability (CAC) (Figure 4.1). This model captures requirements, capability, and complexity, in addition to characterizing the autonomous operating modes of the system [53]. Autonomy is defined in this context as the ability of an unmanned system to sense, perceive, analyze, communicate, plan, make decisions and act to achieve its defined goal [53][46]. On the contrary, other definitions in literature sometimes focus, instead, on the amount of human supervision as the main variable [46].

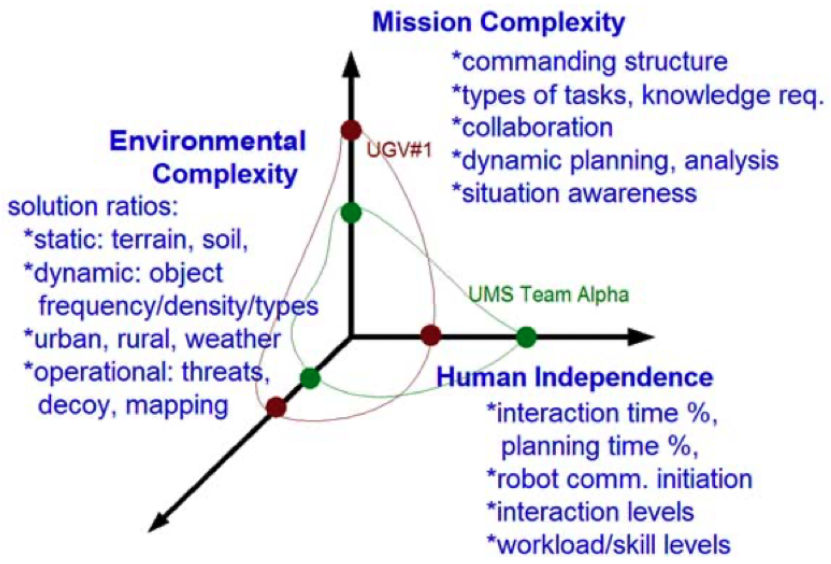


Figure 4.1. ALFUS framework concept [53]

While this framework provides a comprehensive approach toward how several different test metrics may be combined to generate an autonomy level [36], it does not specify direct performance measures for experimental test runs. It is also noted that ALFUS interchangeably uses the term "modes of operation" and "level of autonomy" which, according to Gyagenda et. al is a common misconception [46]. Authors such as Clothier et al. argue that multiple levels of autonomy can constitute a single mode of operation, thereby giving each of these terms distinct meanings [26][46]. It is noted that to the writer's knowledge, no further publicly available publications on the ALFUS framework has been made since 2007.

Other autonomy level frameworks have been put forward, such as the Non-Contextual Autonomy Potential (NCAP) introduced by the Army Corps of Engineers [35], in response to ALFUS. This strategy differs from other methods by treating autonomy level and autonomous performance separately [36]. It is referred to as non-contextual since the autonomy level is measured without taking into account mission and environmental specifics. The ultimate goal is "to provide a means of combining component and engineering level tests into a predictive measure of UMS autonomous performance [36]." Again, however, this framework does not provide an evaluation of an UMS's

Table 4.1. Levels of autonomy defined by CORA [39]

Autonomy Level	**Role Definition**
Fully Autonomous Robots	The role for a robot performing a given task, in which the robot solves the task without human intervention, while adapting to operational and environmental conditions.
Semi-Autonomous Robot	The role for a robot performing a given task, in which the robot and a human operator jointly plan and conduct the task, requiring various levels of human interaction.
Teleoperated Robot	A human operator either directly controls the actuators using sensory feedback, or assigns incremental goals on a continuous basis. A tele-operated robot will complete its last command after the operator stops sending commands, even if that command is complex and time-consuming.
Remote Controlled Robot	The role for a robot performing a given task, in which the human operator controls the robot on a continuous basis, from a location off the robot via only her/his direct observation. In this mode, the robot takes no initiative, and relies on continuous (or nearly continuous) input from the human operator.
Automated Robot	The robot acts as an automaton, following pre-defined (scripted) plans, not adapting to changes in the environment.

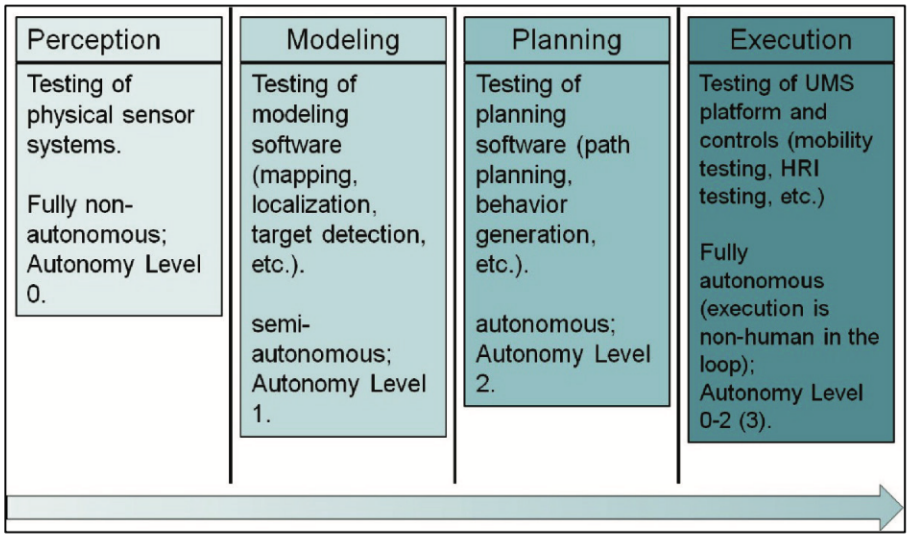


Figure 4.2. NCAP autonomy level architecture [36]

actual autonomous performance, it considers the system's *potential* to operate autonomously [36]. This is a subtle point, autonomous potential versus actual performance executing a task or mission, but one that must be made clear when comparing other methods, proposing new frameworks, and deciding on the goals of evaluating autonomy.

NCAP defines four autonomy levels in their framework. Ranging in discrete values from 0 to 3, the levels are fully non-autonomous (0), semi-autonomous (1), autonomous (2), and fully autonomous (3). Intelligence level is used as the autonomy level benchmark. So, distinction between perception, creating a world model, forming a plan of action from the world model, and both planning and performing an action without human input are the markers for determining the autonomy level classification [36] (Figure 4.2).

The U.S Army's Mission Performance Potential (MPP) framework proposed in [37] uses a hybrid approach by combining contextual and non-contextual performance assessment methods. This framework builds upon the previous work developed by both the ALFUS and NCAP frameworks and focuses on the impact of an increase or decrease in autonomous capability on performance specific to a mission. Durst et. al propose a metric for measuring the mission-specific fitness of a UMS that is a function of the UMS's level of autonomy [37]. Five levels of autonomy are defined

for the MPP (Table 4.2). Three types of input data are required for this methodology: platform physical parameters and sensing capabilities, the platform's decision-making abilities, and environmental conditions [37]. Due to the varying information types, a fuzzy inference technique is used for the MPP calculation. Specifics are not given on how the rule base and the membership functions are constructed. The predicted MPP score is compared to actual performance during a field exercise to validate the methodology, but the authors state that field testing is not necessary for this metric [37]. As discussed with NCAP, this is also looking at potential, with potential in this particular case relative to mission specific performance, but not an assessment of system performance. To the writer's knowledge, no further publications have been made regarding the MPP since this 2014 conference proceeding.

4.1.2 Autonomous System Architecture

In general, for a system to operate autonomously, sensors, actuators, and CPUs are required to work together in a Sense-Plan-Act loop (Section 2.1). Design of these systems is highly driven by deployment context as the majority of these systems are built for specific mission types and environments. There is very little consensus on standard autonomous system design, limiting the reusability and modularity of these systems [113]. A high level system diagram is shown in Figure 4.3 providing an overview of commonly found modules [108].

There have been some strides made from software middleware. Robotic Operating System (ROS) is one notable highlight [8]. As an open source tool for robotics, it has been increasingly accepted across robotic communities for land, sea, and air applications and environments. It's flexible framework allows for highly customizable implementations while encouraging community collaboration. ROS's strength lies in its communication and messaging structure that standardizes interfaces between nodes and packages.

The perception module is often made up of various *exteroceptive* sensors (i.e. sensors focusing on the external vehicle environment) such as depth cameras, Lidars, and sonars that allow the vehicle to "see" and "understand" its surroundings. *Proprioceptive* sensors (i.e., sensors focusing

Table 4.2. Levels of autonomy defined by the MPP [37]

Autonomy Level	Definition
Higher Intelligence	The operator is provided with the vehicle's relevant information for decision making and tactical planning. The operator does not need access to full vehicle's sensor readings or navigation sensors, and instead focuses on the mission-sensitive data collection.
Adaptive Autonomy	The operator is provided with a method for accepting the vehicle-initiated changes to the initial task, path, or goal. The vehicle is capable of suggesting, changing, or overriding previous operator commands, based on new situational awareness. It is at the operator's discretion to manage the decision-making process in the UMS.
Supervised Autonomy	The operator is provided with a method of controlling the vehicle's general behaviors. It is assumed the operator can maintain communications with the vehicle for task reallocation. This autonomy level includes waypoint control, goal-based control, and scenario-based control.
Tele-operation	The operator is provided with a method of indirectly controlling the actuators on the vehicle, through control-by-wire or rates' control. They are also informed of the vehicle's status through communication subsystems and data visualization techniques, i.e., visual animated gauges, maps, arrows, or heads-up- displays.
Radio Control	The operator is provided with a method of controlling the actuators of the vehicle directly. Sensory feedback is through human senses (limited by visual range and noise).

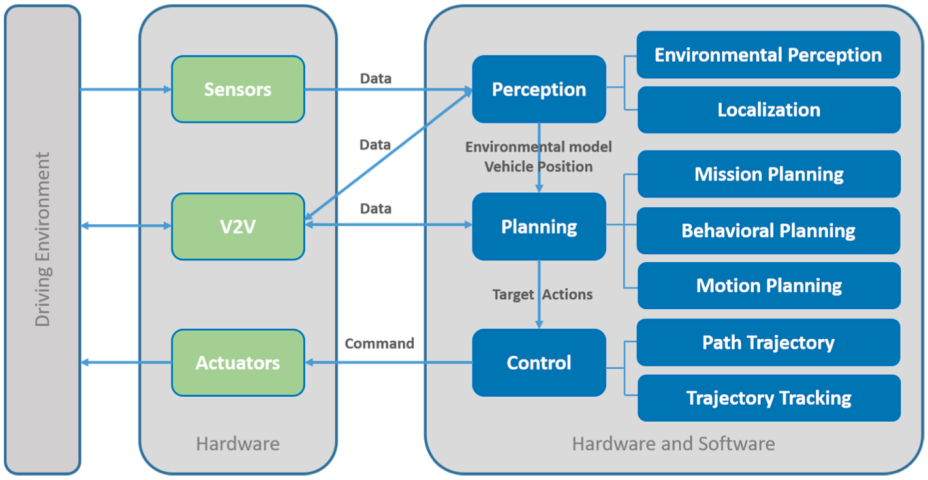


Figure 4.3. High-level system diagram of a generic autonomous system. V2V stands for vehicle-to-vehicle communications [108]

on the vehicle itself) give additional information about the internal states of the robotic platform, such as the vehicle battery level, heading, and position. Increasingly sophisticated systems fuse several sensors together to create a "world" model that helps localize the vehicle and make informed decisions based on external hazards and points of interest. Data from multiple sensors improve redundancy and fault tolerance [108]. These sensor fusion techniques generally fall under three categories: estimation based methods (e.g., Kalman filters), probabilistic calculations (e.g., Bayesian methodologies), and machine learning algorithms (e.g., un/supervised and reinforcement learning) [108]. The capability for perception and situational awareness directly influences the level of decision-making proficiency.

Processing this data, as one can imagine, can quickly become computationally expensive and even impractical. To make real-time decisions, a compromise must be reached between the associated computational requirements and the required data resolution and density necessary for acceptable real-time decision performance. Path planning, navigation, and motion control techniques also depend on the available data quality and require its own portion of the available on-board computational power. Environments also play a major role in processing data. For example, a busy harbor will require much higher sensing capabilities than that of an open ocean environment.

To take into account the existing processing limitations, some systems use a layered hierarchical software architecture which sometimes involve "deliberative" and "reactive" layers (synonymous with global/local methods as discussed in Section 2.1) and incorporates multiple control loops. In this architecture, the platform has the ability to make quick decisions via the reactive layer, while requiring no mission memory or previous knowledge necessary and being able to take on more complex and heavy processing tasks (e.g. route planning) performed in the background as the deliberative layer. Approaching the architecture in this manner results in software practices where sequences may not be sequential and where asynchronous processing techniques (multitasking, parallel threading, etc.) are used [98]. As will be alluded to in the following sections, this type of hierarchical software architecture introduces significantly higher complexity to predicting system behavior and results in added challenges with regards to testing and evaluating internal autonomous processes.

One example of an established autonomy architecture using separate control loops is Mission Oriented Operating Suite - Interval Programming (MOOS-IvP) [5]. Using a "backseat driver" paradigm, vehicle autonomy ("backseat") and vehicle control ("frontseat") are separated. Each uses a distinct processor. However, safety rules can trigger the front seat to perform the protocol in place (e.g., mission abort) and override the backseat commands in a reactive scenario. The autonomous system architecture for the TUPPS also uses this approach and utilizes the backseat/frontseat architecture to decouple vehicle autonomy and vehicle control. The "world" model is built using occupancy grids. The decision output is a desired speed and heading sent to the front seat for execution. The TUPPs front seat controller is responsible for converting the desired speeds and heading values into appropriate thrust values for the port and starboard motors.

4.1.3 Decision-Making Strategies

A well-known way to implement autonomy is using a *behavior-based* approach. Behaviors are self-contained modules that are activated under specifically defined conditions. For example, "Avoid Collision" or "Station-Keep" are behaviors that an ASV may select during a mission. An

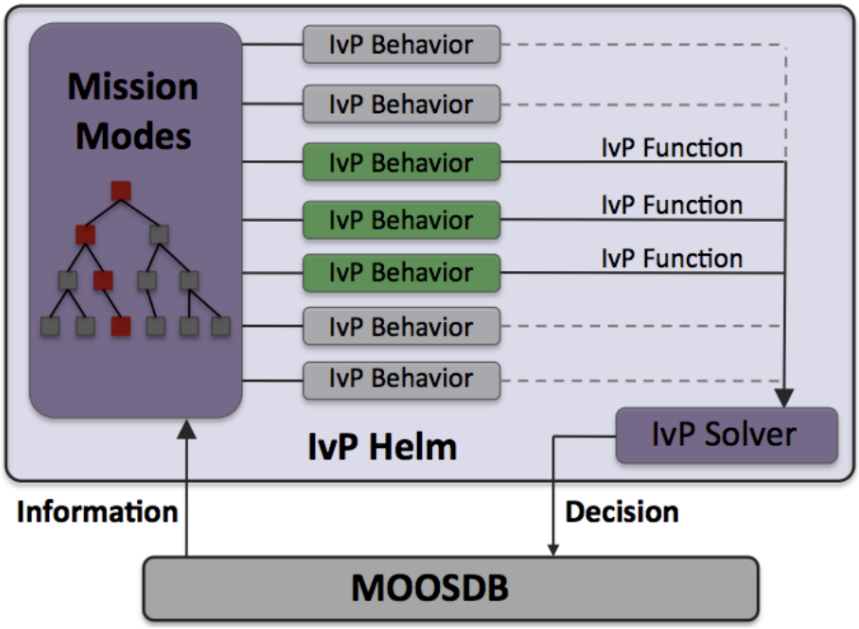


Figure 4.4. A graphical representation of the MOOS-IvP decision-making flow [4]

example of a system that uses this paradigm is Mission Oriented Operating Suite - Interval Programming (MOOS-IvP), an open source set of C++ modules for providing autonomy on robotic platforms [5]. This approach uses multi-objective optimization for selecting behaviors and subsequent decision-making. An objective function maps a domain to a range where the domain in this case refers to a "decision". The "decision-space" outputs a desired heading, speed, and depth. Since there are several objective functions defined for this domain, techniques are then used to determine the location in the decision space that optimizes all functions simultaneously [6]. For a solution to be calculated, users must assign weights to each function. This then reduces the multi-objective optimization problem to that of a single objective optimization problem [6]:

$$\vec{x}^* = \arg\max_{\vec{x}} \sum_{i=0}^{k-1} w_i f_i(\vec{x}) \tag{4.1}$$

where k corresponds to the number of functions and w_i is the weight of the ith function. Figure 4.4 gives an overview of the behavior-based decision architecture for MOOS-IvP.

Markov Decision Processes (MDPs) are another technique for solving optimal decision-making problems using a probabilistic approach [69]. MDPs assume that the world consists of a finite number of discrete states [113]. Since the state of the vehicle is not perfectly observable due to imperfect measurements, MDPs are generalized to a partially observable MDP (POMDP) and are dependent on state probability distributions [71]. POMDP was first presented in 1965 [13], but had limitations in solving for real-time applications. Recently, new developments with computational methods to approximate solutions has increased their potential [111][49]. Though theoretically sound, there are some noted limitations when this approach is scaled to real-world scenarios [41]. As such, research is ongoing to further improve the efficiency of calculations. The PODMP is mathematically represented by a tuple (S, A, T, O Z, R, B) [113][111] where members are defined as:

- $s \in S$: The set of all possible states
- $a \in A$: The set of all possible actions
- $o \in O$: The set of all possible observations
- $T(s', s, a) = P(s'|s, a)$: The transition probability of reaching state s' by action a in state s
- $R(s, a)$: The reward function for choosing an action a in state s
- $b \in B$: The belief estimate of the true state

and an optimal policy, π, may be calculated. This policy maximizes the expected sum of rewards for an initial belief b_0 [111]. And optimal value function $V_\pi(b_0)$ can then be defined such that

$$V_\pi(b_0) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t))\right] \tag{4.2}$$

where γ is a discount factor to incentivize early decisions and V denotes the optimal value function. By using this probabilistic approach, events can be "anticipated" by the vehicle. For instance, Schörner et al. [111] apply this technique to road vehicle intersection scenarios to demonstrate

its validity. Other lanes can not be observed completely oftentimes due to buildings and other vehicles. Using a POMDP, decisions can be made that include consideration of future observations and uncertainty of hidden hazards in the process [111].

Generally, rule-based approaches consist of decision-making logic explicitly specified in advance [95]. A system of rules is built a priori and then used as a database for online decision-making. While perhaps sufficient for simple cases, issues arise when scaling to larger, more complex systems and generalizing scenarios as all potential scenarios must be considered when constructing the system [95]. Accounting for every possible situation also quickly increases the complexity in both the application and management of the system [95]. Learning-based methods have been shown to outperform a majority of rule-based systems [95], but struggle with generalizing to variations of a scenario and necessitating training data to take into account all possible patterns, resulting in both time and robustness issues [41]. Autonomous road vehicles do have a potential advantage in this regard over marine vehicles in that ground vehicles have much more structured traffic patterns, established roads, and the ability to implement surrounding sensing infrastructure.

Many of these approaches incorporate predictive modeling, as in the examples above, as this is considered the key to handling uncertain actions and evaluating any potential consequences [41]. For full reasoning, adaptability, and risk assessment capabilities of autonomous decision-making, uncertainty must not only be taking into account for the vehicle itself but in the intentions of other vehicles, pedestrians, animals, and moving objects. This is an area where autonomous vehicles have not reached the level of human decision-making [112].

Ethical considerations are also warranted. In model based implementations, the human user is responsible for creating the model and thereby developing the system's "values," a system which can still produce unintended consequences. What is often described as a "gut feeling" decision by a human does not involve definable calculations and are typically motivated by a mixture of previous experience, cultural norms, among other factors [103]. AI algorithms present additional complications to decision-making on attempting to obtain appropriate, unbiased, training data.

As with strategies such as neural networks, full transparency is difficult and the networks reach computational levels that make human tracing unfeasible [103]. Much work still lies ahead towards formulating ethical values as quantifiable parameters that can be translated to machines [103][20]. Brutzman et al. state that human accountability is required to preserve ethical boundaries [20]. The operator must have a level of understanding of high-level mission flow, mission descriptions, and that these missions only consist of trusted behaviors [20].

4.2 Mission Planning and Design

Mission planning for autonomous vehicles involves interactions from both human and robot. The amount of freedom the robot is allowed to exercise largely depends on the level of abstraction (or lack thereof) described in the mission plan in addition to a given prerogative to perform any task prioritization and sequencing during a mission. At the highest hierarchical level, the mission has a desired outcome which is then decomposed into an arbitrary number of tasks and low-level actions. Figure 4.5 places mission planning in context of the overall robot planning hierarchy and the corresponding required high-to-low level decision-making required. In Figure 4.5, *Route Planning* refers to a physical sequence of locations to visit, *Path Planning* denotes the spatial path generated for the vehicle to follow, and *Motion Planning* takes into account vehicle dynamics and converts the planned path into a feasible trajectory. These tasks can have various attributing characteristics such as priority value, level of risk, and time constraints [137]. Many systems still rely on preprogrammed commands with minimal decision-making authority causing limited adaptability to situations that arise during mission run time [137]. AUVs, in particular, with the communication challenges of the undersea environment, could benefit greatly from becoming independent of relying on operator commands.

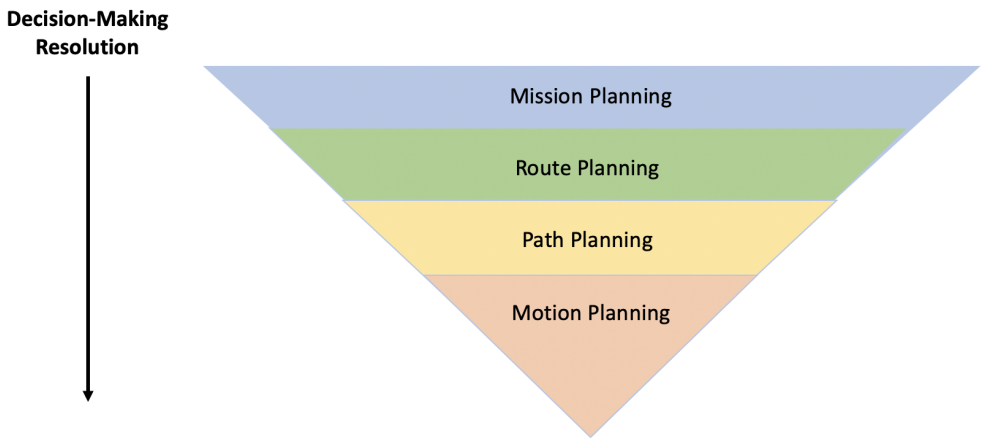


Figure 4.5. Robot Planning Hierarchy

Ideally, through a user interface, a mission planner should reduce operator input to higher-level tasks in order to minimize planning errors [84]. Thus, a common "language" is needed so that human language can be translated to robot actions [84][107]. Several works have proposed frameworks to standardize vocabulary and mission terminology although none have been commonly accepted, especially across all domains of land, air, and sea ([104][21][65][84][102]). The taxonomy developed by Gerkey and Mataric̀ [42] is considered to be the most widely accepted classification method for multi-robot task allocation (MRTA) [99]. Their work defines a common vocabulary for MRTA problems, but is constrained to systems where tasks are independent. Korsah [65] built on this work by proposing another taxonomy, *iTax*, which extends the task allocation space to include interrelated tasks. In a later work, Nunes et al. later proposed a categorized taxonomical approach to the task allocation problem with temporal and ordering constraints [99].

Research in this area commonly draws from combinatorial optimization and operations research with mission definitions providing temporal and ordering constraints [65]. As such, Zadeh [137] treats mission planning as a both a Traveling Salesman (TSP) and a Binary Knapsack (BKP) problem with scheduling split into vehicle task priority assignment and vehicle routing problem, respectively. Task priority assignment is concerned with the combination of solutions as opposed to the vehicle routing problem which is concerned with the order and sequence of solutions [137].

This approach considers the effective management of resources (e.g., battery power) as the main role for the mission planner. Mission success is directly dependent on operation productivity. So, under the assumption that each task is distributed over a specific operation area, Zadeh designed the scheduler as a network of waypoints allowing for the utilization of graph theory techniques. Other graph theory approaches specific to task assignment include algorithms for Tabu search [50], graph matching [66], and branch and cut [79], with extensibility for distributing tasks among several agents [137].

4.3 Current Test and Evaluation Techniques for Autonomous Vehicles

Integration of autonomous vehicles into society will require Test and Evaluation (T&E) processes that produce a quantifiable level of trust and confidence in robot actions [19][38] [108]. Current research still primarily consists of simulations and proof-of-concept vehicles tested only in controlled laboratory or field environments due to the lack of reliable autonomous decision-making performance [19][75][116][43]. One of the main barriers of autonomous systems in civilian applications lies within questions of how safe these systems are [46][108], as studies have shown that public acceptance of autonomous vehicles is low [138]. The existing gap between technological advancement and the effective testing and evaluation of these systems must be bridged to make autonomous vehicles increasingly practical and accepted for field use.

4.3.1 Experimental Testing

Current experimental test procedures for autonomous vehicles are usually conducted on a case-by-case basis without mathematical rigor partly due to the lack of agreed upon standards and definitions for autonomous systems and related performance metrics [100][51][36][43][75] (as discussed in Section 4.1.1). Autonomous ground vehicle companies resort to driving millions of miles to perform validation tests which is generally economically impractical [9][93][56]. Other test range methods have been developed in response to ground vehicle competitions such as the DARPA Grand Challenge [119][129]. These methods have applied a Type-1 Fuzzy Inference System (T1-FL) using an analytic hierarchy process (AHP) for weight distribution. T1-FL systems,

however, do not take into account additional uncertainty in its given membership functions and, therefore, have limited capabilities in minimizing the effects of such uncertainties [85]. These uncertainties originate from such sources as (but not limited to) noisy measurements, the chosen linguistic terms, and the user-defined rule-base, among other variables. These competitions typically drive innovation and push technological boundaries, but do not necessarily push the boundaries of certifying and assuring acceptable vehicle behavior at parallel speeds [106].

4.3.2 Modelling and Simulation

In terms of simulation based studies, standard T&E techniques (e.g., design of experiments and Monte Carlo analysis) are unusable due to the excessive number of variables inherent to an autonomous system [17]. One technique developed by the Johns Hopkins Applied Physics Laboratory (APL) uses modeling and simulation to perform many iterations of particular scenarios and then form scores to provide some means for evaluating a statistically large (or very large) number of runs. This tool, referred to as the Range Adversarial Planning Tool (RAPT), searches for boundaries in capabilities to identify the most critical tests for test range operations [97]. Other work has been done with developing methods using model-checking, finite state machines, and process algebras, but these techniques require a model that completely describes the autonomy [97]. Due to proprietary intellectual property with software and the complex nature of autonomous systems, these modeling strategies have limited applicable use [97][106]. Additionally, these strategies test the robustness of the software but do not give information about how the system will perform while executing the mission (i.e. whether the vehicle will navigate to the left or right of an obstacle) [96].

4.3.3 Validation and Verification

T&E procedures usually include validation and verification of systems. *Validation* analyzes whether or not the system meets the intended design objectives and desired uses (i.e., "Are we building the correct product?"), while *verification* addresses whether the design specifications are correctly implemented by the system (i.e., "Are we building the product correctly?"). These classic techniques do not transition well for autonomy testing as these systems contain numerous interact-

ing behaviors, many of which are not deterministic [106]. Autonomy development aims to eventually have vehicles handle situations that are unknown and not explicitly defined a priori [106]. This makes current verification strategies that require complete knowledge of how the system is expected to behave in all situations unfeasible [106]. Additionally, by defining the requirements on lower levels, it leads to the autonomy being designed as well, thereby limiting the system to actually act autonomously [106]. Due to the challenges of proving reliability with exhaustive verification and scenario-based testing, there has been a push to develop techniques where generalizations can be made, an appropriate subset of conditions can be tested, and trusted conclusions and inferences on autonomous behavior can be produced.

4.4 Discussion

The sections above speak to some of the varying approaches towards defining autonomy, system architecture, decision-making, mission planning, in addition to test and evaluation challenges of these autonomous systems. Common themes include a lack of accepted definitions and standards, trust in safe operation, and managing uncertainty in an unpredictable world. All of these topics of this chapter depend on one another to progress technologically. Effective decision-making, mission planning, etc. can only advance with improvements to situational awareness and understanding the environment as well. This chapter has covered a broad range of autonomous system topics to demonstrate these deeper-level intricacies and the vast complexity that requires rigorous vehicle and autonomy testing before field deployment can be made practical.

It is clear that autonomy cannot be tested as other systems are. The nondeterministic nature of the algorithms causes increased difficulties localizing and diagnosing defects with unreproducible sequencing. Ocean environments, especially, are inherently unpredictable and rarely produce identical conditions. Consider a driving license test where one scenario is typically tested. Not a perfect measure by any means, but it comes with what one perceives as an “acceptable” level of risk. What is acceptable risk for an autonomous vehicle? The answer is, of course, context-dependent but one must realize that the minimum benchmark must be improved human performance (and/or safety).

A large portion of the work accomplished thus far focuses on the tools to measure the vehicle autonomy level and not the level of vehicle performance while attempting to accomplish its pre-determined tasks [106]. Predicted capability does not necessarily signify that the vehicle can actually perform up to those standards [46]. These techniques (as discussed in Section 4.1.1) predict expected performance for a mission set and level of autonomy but it does not directly compare the performances of varying types of autonomy algorithms, nor does it provide a direct measure of vehicle performance for individual iterations of a given mission scenario. Additionally, these methods do not account for system faults and sensor perception issues.

The contributions presented here and in the following chapters approach autonomous system evaluation differently from the stated norm. The proposed test and evaluation methodology in the following chapters, referred to as the Performance Evaluation and Review Framework Of Robotic Missions (PERFORM), addresses the above challenges by:

- **Modular Scenario-Based Testing:** Mission and task decomposition strategies with analysis towards extrapolation to other similar tasks helps maximize data from each test. By minimizing the scenarios necessary to test, the approach increases efficiency to the T&E process and is easily scalable. Modularizing mission components adds flexibility to examine not only individual tasks and behaviors, but to also examine overall missions.

- **Incorporating Uncertainty:** Acknowledging the uncertainty of a test run projects a measure of confidence in the determined level and repeatability of performance a vehicle has demonstrated during testing. Variations in environmental conditions, trajectories of other potential hazards (i.e. other vehicles), and perception of these situations will always exist. Providing a quantitative view of test quality and limitations allows for better risk assessment before field use.

- **Performance Based Assessment:** The proposed framework is a performance based assessment. Evaluation, in this context, refers to measuring the ability of the vehicle to complete the given mission and not its total (highest possible level) abilities. The metric for perfor-

mance is the ability to successfully achieve the desire goal which is independent of autonomy level [46]. Human and robot performance may be tested in the same framework. However, the scope of this work does not include human/robot interactions. The end goal is to provide a vehicle performance score for the test engineer to make an informed decision on a vehicle's fitness for a specific mission, but *without* predicting mission success (which is outside the scope of this work).

- **Independent of Internal Autonomy Architectures and Level of Autonomy:** Maintaining independence as an external observer from "black box" systems is critical for maintaining flexibility of the framework and in ensuring its global applicability across varying types of autonomous vehicles. Using system performance and autonomy as distinct measures permits testing performance without requiring a priori knowledge of the Level of Autonomy (LOA), thus, simplifying the performance evaluation problem. If a lower cost and simpler platform can perform the mission on par with a more expensive and complex option, that provides an important piece of information with cost and operational benefits.

- **Real-Time Testbed Environment:** While it is understood that live field testing is restricted to far fewer test runs than that for simulation studies, it is still critical to confirm that hardware, software, and the overall system-of-systems (SoS) are working cohesively. The testbed acts as a "dress rehearsal" space providing, improvements to the end of the T&E spectrum.

4.5 Proposed Mission Design for Evaluating Autonomous Vehicles in a Testbed Environment

4.5.1 Defining a Test Mission

For the purpose of generalizing the autonomy evaluation procedure, a generic architecture for defining the test missions is proposed. In this work, it is assumed that any autonomous *mission* may be subdivided into a set of generic *(sub)tasks* that autonomous missions often incorporate (e.g., transit, conduct survey, station keeping, etc.). (For the remaining document, *task* will refer

to both tasks and subtasks). These tasks are separated by time and order via the mission-scripted programming but also by *events* which represent time-based changes in the environment (e.g., a newly discovered obstacle, a sudden change in sea state, etc.) or within the autonomous platform itself (e.g., low battery, faulty on-board sensor, etc.). Depending upon how/when these events occur during the tasks being performed, specific mission *behaviors* should (or must) take place in order to complete a specific task (or mission) or to prevent catastrophic system failure. In other words, behaviors consist of actions and/or reactions conducted by the autonomous vehicle during the execution of a task to aid in completion of the task. An important aspect of decomposing a mission into tasks and events is that at the lowest level of the decomposition, there could be significant overlap of tasks and reactions to events for a wide variety of missions. This enables one to make inferences about future autonomous capability from the evaluation performed in this environment for a set of canonical missions. Flexibility in mission definitions to capture a range of operational scenarios (as opposed to testing individual scenarios) during evaluation is highly desirable [107]. Describing a mission with tasks and associated events form the foundations from which corresponding scenarios and metrics may be created.

In the proposed architecture, a *mission* (M) is defined as a set of tasks that must be completed to realize a specific goal or goals. *Tasks* (T) are a set of data structures that define quantifiable starting and ending states (as a function of time and space) and that can be combined with other tasks to realize a mission goal. Tasks should be described through simple language (e.g., travel to a desired waypoint), and they are conditionally dependent within a specific mission such that relations between the order of tasks can be represented as a directed graph. *Events* (E) are defined as a set of possible time-based changes (both internal and external to the given autonomous vehicle) that occur during and/or as a result of a specific task. *Behaviors* (B) are defined as the actions necessary to perform a given task (e.g., avoid known obstacle, no event occurring, etc.) or occur in response to an event (e.g., avoid new obstacle, perform station keeping, etc.). Example parameters used for performance evaluation may include elements such as the vehicle's total time and distance traveled to complete a task, the closest point of approach (CPA), and the total energy consumed.

Scenarios become instances of a given mission and is defined such that S denotes a space of all possible scenarios that can be "realized" with respect to the mission and the platform being used to accomplish the mission. Each element $S_i \in S$ is selected by choosing values for a set of scenario parameters that are derived from key aspects of the mission description. Scenario parameters may encompass items such as survey areas (box size and location), launch/recovery points, time of day, and starting battery charge. A metric function, β, maps elements of the scenario space S_i to $\Re$ values:

$$\forall S_{ik} \in S_i \, \exists \, \beta : \beta(S_{ik}) \in \Re \tag{4.3}$$

thereby combining many aspects for evaluating performance of the autonomous vehicle as it reacts to events and accomplishes tasks as it executes the Scenario. Essentially, this metric produces a score for every scenario (a realized instance of the mission). Each Scenario should be considered an instance of the execution of the mission by the given platform. In practicality, there are only a finite number of scenarios that are actually executed, either in a simulation/modeling environment or as live field tests. However, because of the way the scenario parameters can be specified, some can take on a range of values, which allows the possibility of an infinite number of scenarios to take into account the entire interval of values that could possibly be assigned to a particular parameter.

In set notation, the mission space with the associated T, E, B, and S may be represented as follows:

$$M(S_i) = \begin{cases} M_{1i} = \left\{T_{11}(t,x,y,z,B,E), T_{12}(t,x,y,z,B,E), ..., T_{1n}(t,x,y,z,B,E)\right\} \\ M_{2i} = \left\{T_{21}(t,x,y,z,B,E), T_{22}(t,x,y,z,B,E), ..., T_{2n}(t,x,y,z,B,E)\right\} \\ \vdots \\ M_{mi} = \left\{T_{m1}(t,x,y,z,B,E), T_{m2}(t,x,y,z,B,E), ..., T_{mn}(t,x,y,z,B,E)\right\} \end{cases} \tag{4.4}$$

where m, n, and i are mission number, task number, and scenario number, respectively. Figure 4.6 gives an overview of an example Autonomous Surface Vehicle (ASV) seafloor mapping mission with the proposed architecture.

By representing a mission with this framework, set operations may then be used to compare various generic missions and tasks with the intention of limiting test redundancy and reducing the number of test missions to that of the most critical scenarios. The main goal here is to alleviate the need to determine the exhaustive list of all possible scenarios and combinations of mission tasks, events, and behaviors for which the performance of the vehicle autonomy must be tested against. Instead, this mission framework provides an analysis platform with which to test/observe platform autonomy and which is fully modular, versatile, and scalable.

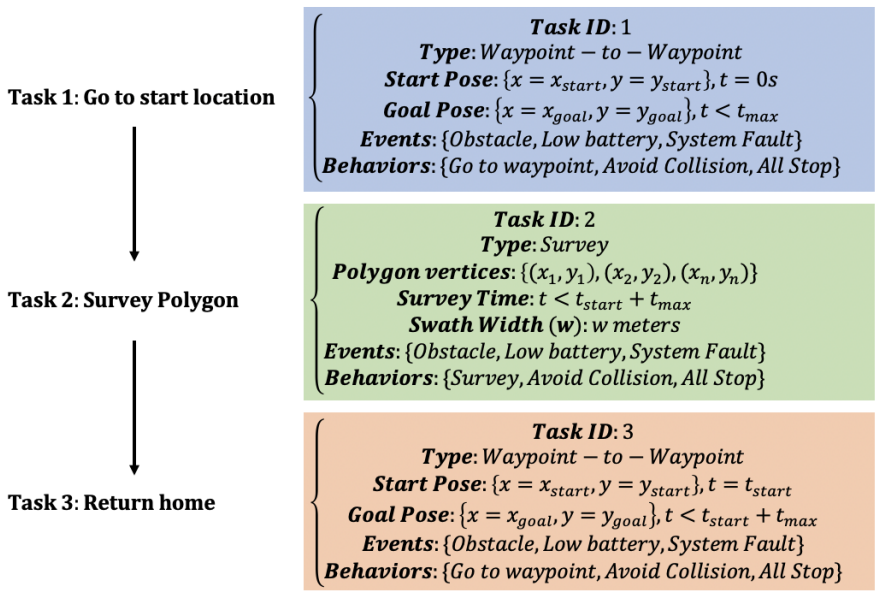


Figure 4.6. Example seafloor mapping mission with associated tasks and behaviors

Another useful representation is expressing the mission as a directed graph such that M is defined as the graph, T is the subgraph partitioned by tasks containing a set of vertices, V_G, and a set of ordered edges, E_G (Equation 4.5). The subscript G differentiates the graph representation from that of the set notation of Equation 4.4. Mission M may then be represented as:

$$M(T_1(V_G, E_G), T_2(V_G, E_G), ..., T_n(V_G, E_G)) \tag{4.5}$$

Figure 4.7 shows an example of this graph structure with the same mission as defined in Figure 4.6. In Figure 4.7, insight on relationships between tasks and possible events causing transitions between tasks is observed. Here, the vertices correspond to the possible events and the edges are the possible transitions between events. Behaviors, B, represent the underlying status of the vehicle in response to events and are not shown explicitly in Figure 4.7 and Equation 4.5. Figure 4.8 shows a task and event path for an instance of mission failure versus mission success.

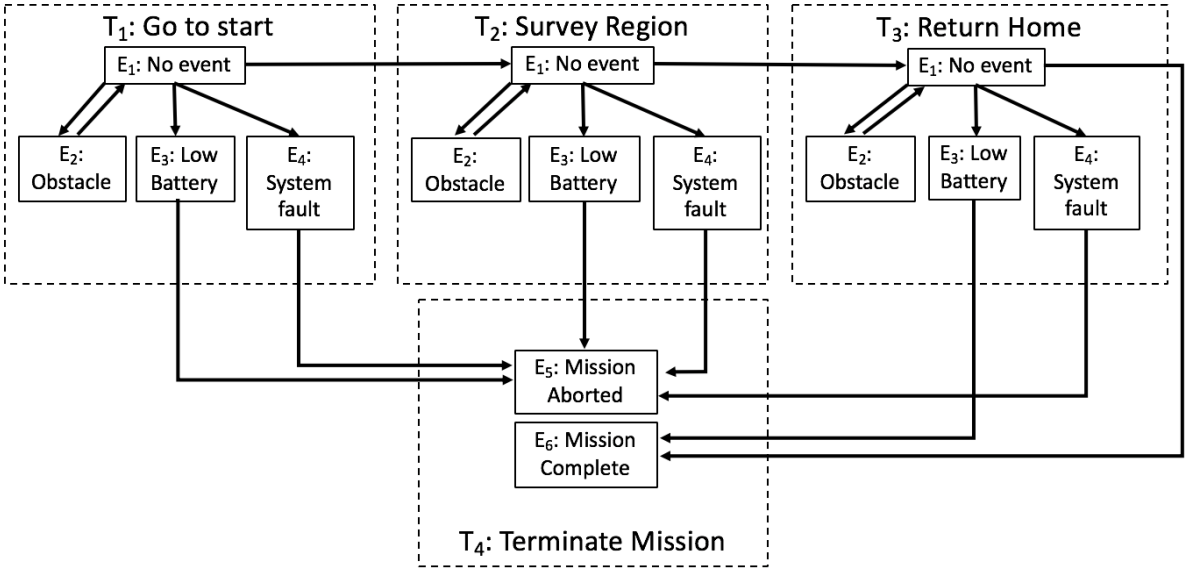


Figure 4.7. Graphical overview of the example mission for a given scenario

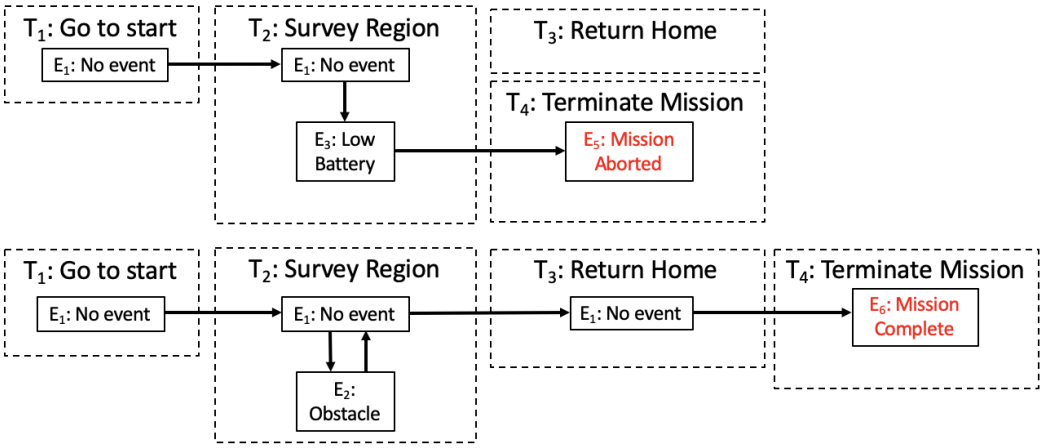


Figure 4.8. Example of a task and event flow for mission failure (top) vs. mission success (bottom)

The concept of a *Capability Space* is also introduced here. Notated as $C_{ik} \in C_i$, each element is defined through a one-to-one mapping of S_i to C_i where each element is assigned a score $\beta(S_{ik})$. The *Capability Space*, which contains different capability regions, is used to identify which scenarios are considered successful, which are considered failures, and the potential boundaries between these regions. The implementation of PERFORM is intended to act as this scoring metric, β. This metric serves to provide a single score for each Scenario that takes on a range of values so that different scenarios can be more easily compared to evaluate relative success/failure. The following chapters will introduce and apply this proposed metric with the help of an Interval Type-2 Fuzzy Logic approach.

CHAPTER 5

FUZZY LOGIC THEORY

5.1 Overview

Fuzzy Logic (FL) systems and theory, a subset of AI, is employed as the basis for quantifying parameters that describe the performance of a task. FL is the chosen approach for this work due to its ability to provide a strict mathematical framework in which vague and uncertain phenomena can be precisely and rigorously studied [135][139]. A complete description of a real system requires complete knowledge of the mission which is often not known a priori [139]. Due to the infinite number of possible scenarios in a given mission, the impracticality of constraining the system, and the prevalence of proprietary software, the approach for this research remains independent of internal autonomy architectures [97][17] and defines boundaries on the scope of this work without loss of generality. Additionally, this FL strategy allows flexibility for various definitions of "success" between users (e.g. a defense employee as opposed to a scientist) by weighting performance qualities the user deems important.

Unlike traditional binary logic (i.e., either "0" or "1"), FL utilizes the concept of partial truth (i.e. values ranging between 0 and 1). This logic more closely resembles how the human brain processes information [135]. The general FL procedure takes *fuzzified* input data (via linguistic variables and user-determined membership functions) and processes them through an "if-then" framed rule base, producing a set of fuzzy outputs. These outputs are then combined/weighted into a single output, which is then *defuzzified* (via another set of membership functions) to produce the final crisp result. First presented by Zadeh [134], this strategy is often employed in other sectors such as medicine, manufacturing, and business with applications to control systems, decision-making, evaluation, and management.

5.2 Type-2 Fuzzy Logic

Type-2 Fuzzy Logic (T2-FL) was first defined in [135], but has gained significant research interest in recent years [127][23]. T2-FL is also referred to as "General T2-FL (GT2-FL)." A special case of GT2-FL systems is Interval T2-FL (IT2-FL). Both categories of T2-FL are parametric models with additional design degrees-of-freedom to that of a T1-FL system [87] [22] and are useful in situations where determining a definitive MF is difficult and where dynamic uncertainties in unstructured environments must be taken into account [135][58][127][86]. With this additional design degree-of-freedom, not only can the degree of membership of a given linguistic set be modeled, but the uncertainty in the degree of membership as well. Basic terminology is reviewed in this chapter to rationalize the use of Interval Type-2 Fuzzy Logic (IT2-FL) over that of other forms of FL.

5.2.1 Membership Functions

In traditional T1-FL theory, a characteristic function allows for various degrees of membership for the elements of a given set. A T1 fuzzy set, A, with a collection of objects X (also referred to as the "universe of discourse") and elements x, is defined as:

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{5.1}$$

where μ_A is the degree of membership of an element x in the set A [139]. These *membership functions* (MFs) are user-defined, often (but not exclusively) incorporating triangular, trapezoidal, and gaussian functions, among others. The MF itself is an arbitrary curve that is chosen for simplicity, convenience, speed, and efficiency. For this application, the fuzzy sets for mission parameters is assumed to have defined minimum and maximum thresholds based on the testbed space.

A T2-FL set is denoted as $\widetilde{A}$. As opposed to T1 systems, additional Lower Membership Functions (LMFs) and Upper Membership Functions (UMFs) must also be defined for T2-FL. The corresponding MFs are denoted as $\mu^L_{\widetilde{A}}(x)$ and $\mu^U_{\widetilde{A}}(x)$ for lower and upper bounds, respectively. X

now refers to the primary domain, and J_x is now defined as the secondary domain. Rewriting Equation 5.1 as a General T2 system, $\widetilde{A}$ can be defined as (as shown in [23][115]):

$$\widetilde{A} = \{((x,u), \mu_{\widetilde{A}}(x,u)) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{5.2}$$

What differentiates IT2-FL from GT2-FL is that the IT2-FL uses a uniform secondary MF (i.e., $\mu_{\widetilde{A}}(x,u)$ = 1), whereas the MF of GT2-FL varies with its secondary membership. As such, the resulting IT2-FL reduces the GT2-FL of Equation 5.2 to Equation 5.3 as exemplified in Figure 5.1, and $\widetilde{A}$ reduces to:

$$\widetilde{A} = \{((x,u), 1) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{5.3}$$

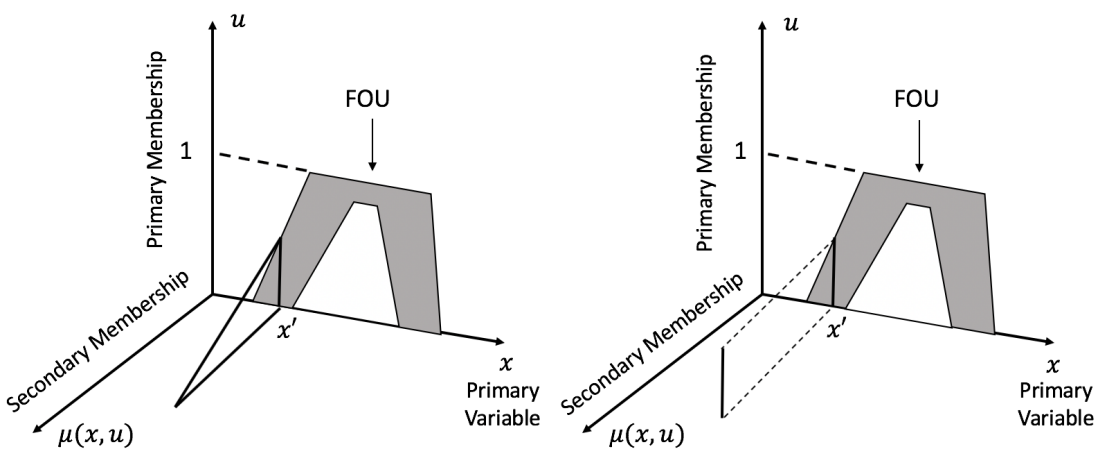


Figure 5.1. Example of a GT2-FL (left) vs. IT2-FL (right) Fuzzy Set

Here, since the third dimensional value of the IT2-FL membership is constant, it can be more conveniently represented as a reduced two-dimensional Field of Uncertainty (FOU). For added accuracy and versatility in dealing with MF uncertainties (e.g., vehicle platform uncertainties and sensor noise), the author chooses to implement T2-FL (over T1-FL), and selects IT2-FL (over GT2-FL) in anticipation of possible data overload burden to maintain computational feasibility, as the GT2-FL approach has been previously shown to introduce design issues and results in high computational costs [87].

As variance quantifies the uncertainty of a variable's mean in probability theory, the use of IT2-FL enables the quantification of existing uncertainties within a membership function. The FOU is defined as the area bounded by the Lower Membership Function (LMF) and the Upper Membership Function (UMF) and is depicted in the example MF in Figure 5.2, where the IT2-FL MF is also differentiated from that of a T1-FL. A T2-FL design constraint is the LMF $\leq$ UMF for the entire domain. The IT2-FL FOU may be expressed as the union of all primary memberships such that

$$FOU(\widetilde{A}) = \bigcup_{\forall x \in X} (\mu^L_{\widetilde{A}}(x), \mu^U_{\widetilde{A}}(x)) \tag{5.4}$$

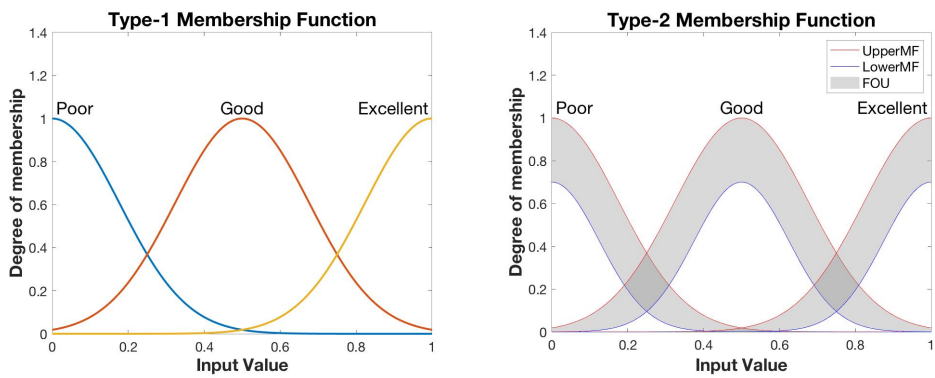


Figure 5.2. Example membership functions for T1 (left) and T2 (right) Fuzzy Systems

5.2.2 Implication

The mapping of the input space to that of the output space is the result of processing the input through a set of "if-then" linguistic rules (e.g. "If x is A, then u is B"). The Fuzzy Inference System (FIS) begins with fuzzifying the crisp input values by using the LMFs and UMFs of the rule antecedent to determine the corresponding degree of membership in terms of linguistic metrics. This step produces two fuzzy values for each IT2-FL MF from the membership value of $x = x'$ at the UMF and LMF of the activated fuzzy sets: $f_U(x')$ and $f_L(x')$ (Figure 5.3)[89]. With application of the fuzzy operator *min* to all $f_U(x')$ and all $f_L(x')$, this creates a *firing interval* such that

$$f'_U = min(f_{U_1}(x'_1), f_{U_2}(x'_2))$$
$$f'_L = min(f_{L_1}(x'_1), f_{L_2}(x'_2)) \tag{5.5}$$

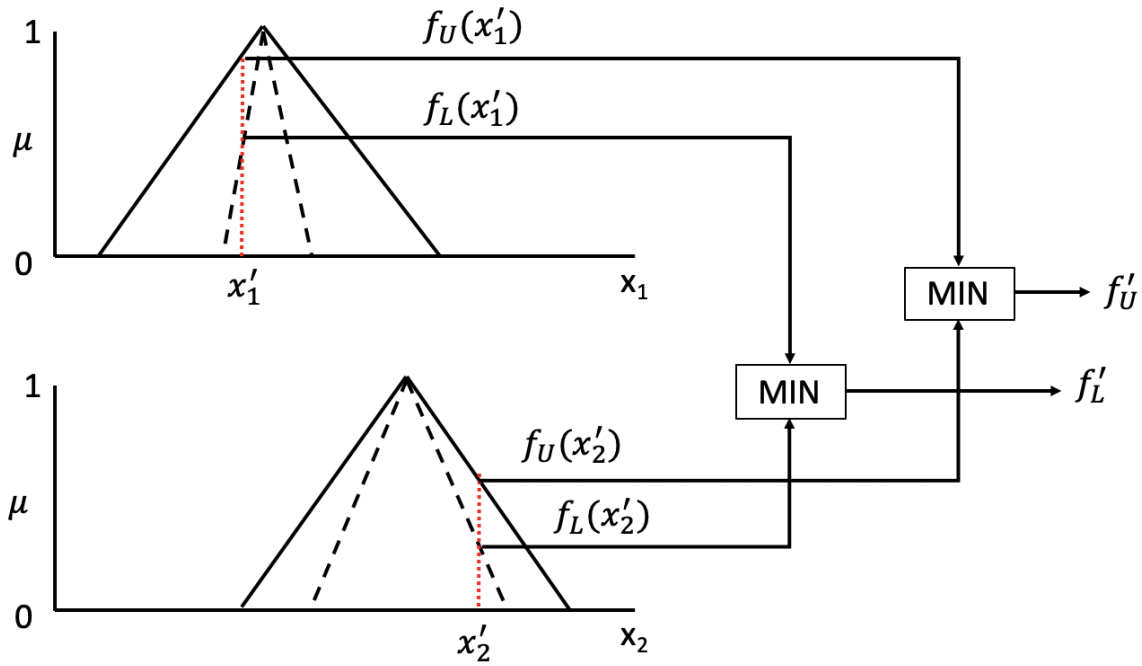


Figure 5.3. Firing interval of the IT2-FL system for two antecedents

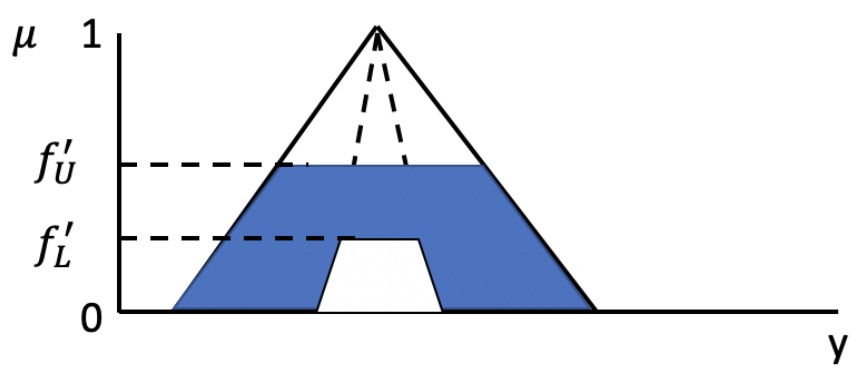


Figure 5.4. Output FOU for the consequent fuzzy set after implication

Here, the interval has a maximum value of f'_U and a minimum value of f'_L, and Figure 5.3 demonstrates this two antecedent case.

From the firing interval for each rule, the output (i.e. the consequent of the rule) becomes an IT2 fuzzy set (Figure 5.4). Mendel [89] stated that this fuzzy set is obtained because the firing set is a continuous set of points between the UMF and LMF. The *min* implication method trims this fuzzy set using the firing range interval.

Inference mechanisms in FL are divided between two well established methods: those from Mamdani [81] and Sugeno [118]. The main difference between the techniques lies in the output generation process [47]. Sugeno FIS relies on a weighted average for output computation as opposed to the defuzzification step utilized for a Mamdani FIS to obtain a crisp output. As such, the Sugeno method outputs either constant or linear equations instead of a fuzzy set.

The Mamdani technique is selected for this work due to its demonstrated advantages over that of Sugeno in intuitiveness, rule base interpretability, and when taking into account human input and rules created from a priori knowledge and experience [83]. It is noted that the Sugeno method is more computationally efficient. However, as stated previously, computational efficiency is not considered critical since the intended use is for pre/post-processing and not for real-time application.

5.2.3 Aggregation

In this step, the output fuzzy sets from applying the rule base is combined into a single T2 fuzzy set. There are several different aggregation methods, but the "maximum" union operator will be the assumed method for this work as it is the most commonly used. As an example, the output set corresponding to a second rule is added to that in Figure 5.4 and is shown in Figure 5.5.

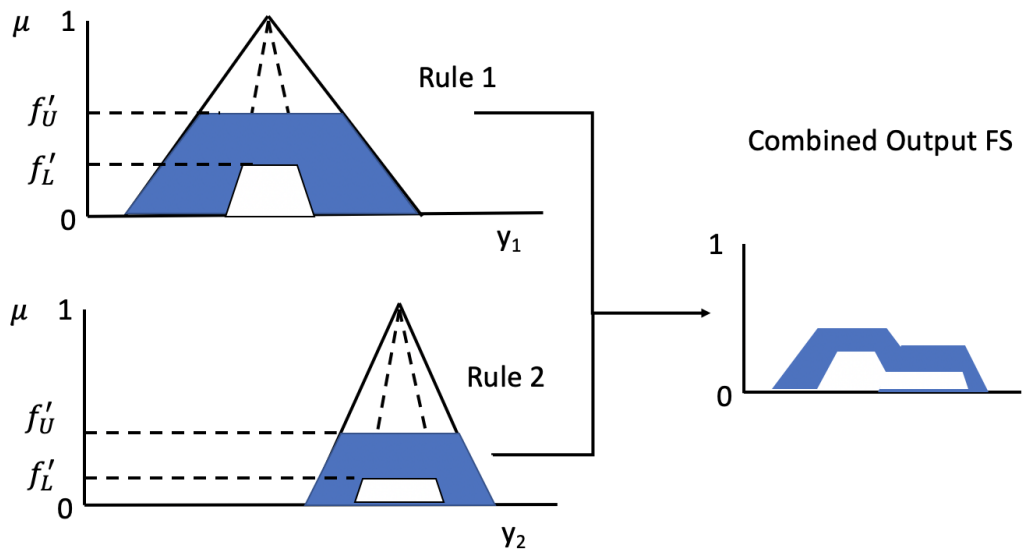


Figure 5.5. Combined T2 output fuzzy set for two fired rules

5.2.4 Defuzzification

Under the assumption in using a Mamdani-based system, defuzzification is necessary to convert the fuzzy output set to a crisp output value. The T2 output fuzzy set is reduced (via a "type reducer") to an Interval T1 fuzzy set resulting in a range (with c_L being the lower limit and c_R the upper limit) and which is considered the centroid of the T2 fuzzy set. This refers to the average of the centroids of all the Type-1 fuzzy sets embedded in the Type-2 fuzzy set. The centroid values are calculated iteratively due to the inability to compute exact values for c_L and c_R. Methods developed by Karnik and Mendel [59][88] are commonly used with the approximations for c_L and c_R given as

$$c_L \approx \frac{\sum_{i=1}^{L} x_i \mu_{\tilde{A}}^{U}(x_i) + \sum_{i=L+1}^{N} x_i \mu_{\tilde{A}}^{L}(x_i)}{\sum_{i=1}^{L} \mu_{\tilde{A}}^{U}(x_i) + \sum_{i=L+1}^{N} \mu_{\tilde{A}}^{L}(x_i)}$$

$$c_R \approx \frac{\sum_{i=1}^{R} x_i \mu_{\tilde{A}}^{U}(x_i) + \sum_{i=R+1}^{N} x_i \mu_{\tilde{A}}^{L}(x_i)}{\sum_{i=1}^{R} \mu_{\tilde{A}}^{U}(x_i) + \sum_{i=R+1}^{N} \mu_{\tilde{A}}^{L}(x_i)} \tag{5.6}$$

where N denotes the number of samples, x_i is the ith output value sample, and L and R are the left and right estimated switch points, respectively. The defuzzified crisp output value, y, is determined by averaging the two centroid values such that

$$y = \frac{c_L + c_R}{2} \tag{5.7}$$

The reader is referred to [88] for further detail on FIS. An overview of the architecture of T1 and T2 Fuzzy systems is provided in Figure 5.6 and Figure 5.7, respectively.

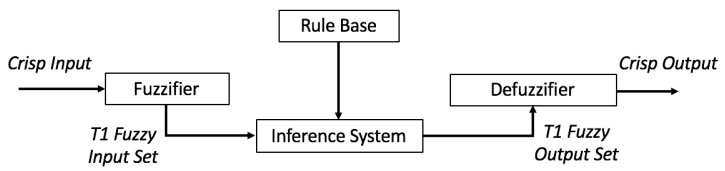


Figure 5.6. T1-FL System

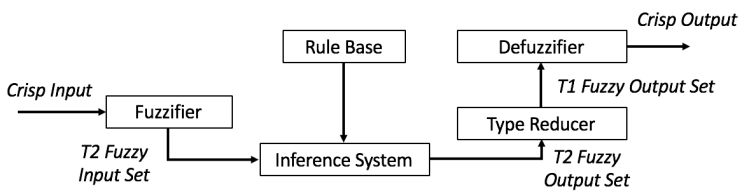


Figure 5.7. T2-FL System

In literature, studies have been performed comparing T1 and T2 systems. Castillo et al. [23] completed a comparative study of T1-FL, IT2-FL, and GT2-FL systems in control problems. They conclude through simulation that the IT2-FL controller showed marked improvement over the T1-FL controller with and without generated noise perturbations. Linda and Manic [73] also analyzed T1-FL and IT2-FL systems using fuzzy control in the context of learning behaviors for mobile robotics. The IT2-FL membership functions were constructed by blurring the membership functions of the original T1 system [73]. Findings included that the IT2-FL system out performed the T1 system when handling noisy inputs, but a reduction in speed by the former. Since the proposed application of IT2-FL for autonomous vehicle performance evaluation occurs during the

post-processing stage, slightly longer calculation time is not of concern. The IT2-FL provides a convenient compromise between design complexity and computational burden.

CHAPTER 6

PERFORM: PERFORMANCE EVALUATION AND REVIEW FRAMEWORK OF ROBOTIC MISSIONS

6.1 Background

Specific to performance evaluation studies, fuzzy logic has been utilized in selected applications. Surya et al. [120] investigated the use of T1-FL for grading student performance. They argued that classical methods inefficiently describe a student's skill level. Through direct comparison of the classical and proposed fuzzy approach, the flexibility and potential improvements of the fuzzy methods were shown. FL techniques have also been applied to measure cloud computing performance by defining different cloud performance parameters [110]. Saxena and Nanath chose FL for that application specifically for its ability to describe complex systems with linguistic descriptions [110]. Since the evaluation framework was constructed to help the users evaluate the performance of the cloud, the linguistic descriptions provided an intuitive interface.

A study by Debnath et al. [33] performed air quality assessment using a weighted IT2-FL inference system. Air quality assessments traditionally use an index with distinct boundaries, and thus do not accurately depict existing boundary haziness [33]. Debnath et al. considered evaluation of air quality as a problem of the *degree* of pollution, thereby pointing towards the use of fuzzy logic techniques as an appropriate tool. They incorporate both Interval Type-2 and Analytic Hierarchy Process (AHP) to provide additional uncertainty and parameter weighting management, respectively.

Sun et al. [119] deployed T1-FL tactics to evaluate the DARPA Grand Challenge AGV competition. Combined with AHP for multilevel evaluation, this work analyzed five evaluation elements: vehicle control behavior, basic driving behavior, basic traffic behavior, advanced driving behavior,

and advanced traffic behavior [119]. The membership grade was determined by the evaluation of 10 experts on their assessment of task completion quality, rather than the input being dependent upon data collection. Another robotic application by Chen and Zhang [24] combined FL, evidence theory, and fuzzy neural networks to create an evaluation method for unmanned automotive robots. This methodology used both subjective and objective evaluation measures to train the fuzzy neural network model.

6.2 Performance Parameters

The performance parameters for the proposed framework in this work consist of quantitative characteristics that describe aspects of a mission and/or task. Spatial measures, such as total distance travelled by the vehicle between two waypoints, for instance, would provide a value that analyzes both the path planner and the control system. Since the evaluation concern is on the execution level (i.e., the actual performance), it is not necessarily important to make the performance parameters distinct between the path planning and control subsystems. The proposed framework, instead, provides a comprehensive assessment that asks the question, "Can this vehicle adequately perform this task and/or respond to a particular unexpected event?" As the mission architecture in Section 4.5.1 presents, the tasks may be decomposed into the mission's description and potential scenarios to the largest set of individual tasks that correspond to different behaviors and/or modules within the autonomy software *at the level it is being tested*. If, for instance, a waypoint-to-waypoint behavior is under test, then path planning, obstacle detection, and heading/speed control may all be applicable subtasks. If, perhaps, the confidence in the platform's control system is already established, the task decomposition may simply address path planning and obstacle detection.

Decision-making may be measured in various ways depending on the context. An approach to measure obstacle avoidance performance is to consider the Closest Point of Approach (CPA). For example, vehicle approach and close proximity to an obstacle would demonstrate poor situational awareness, and an incorrect decision for the calculation of a safe trajectory could result. On a higher mission level, decision-making could also be considered from a mission planning perspec-

tive, where the sequencing of tasks is the decision-making component under test. Here, selecting the order of tasks for the best mission productivity could be measured by the number of tasks completed within a given time constraint.

Due to the inability to completely control environmental test conditions, fuzzy logic may be used to calculate a separate reference score in order to provide additional context to a particular test run. This score could consist of some combination of Beaufort force, current speeds, water turbidity, fog conditions, etc. The level of environmentally related challenges may change between test runs, so this could be an additional quantitative value to differentiate performance scores.

6.2.1 Inputs

Each parameter, p_i, are elements of the set p on the universe of discourse U_i and serve as the inputs of the FIS:

$$p = [p_1, p_2, ..., p_i] \tag{6.1}$$

where the domain of each member of p is bounded such that

$$p_i \subset [x_{min} \quad x_{max}] \tag{6.2}$$

The corresponding range is the fuzzy membership value, μ, which is bounded such that $p_i \subset [0 \quad 1]$. The limits of p_i are determined by the test engineer using expected or required results. Single-valued inputs are assumed, meaning that some data may need to be a priori. Considering average speed, for instance, one would first compile the logged speed data and then compute the mean to use as the final input. The determination of performance is not a "per time step" calculation as with that of FL-based control system type implementations. Figure 6.1 provides a visual representation of a generic input parameter with associated triangular fuzzy sets $\widetilde{A}$.

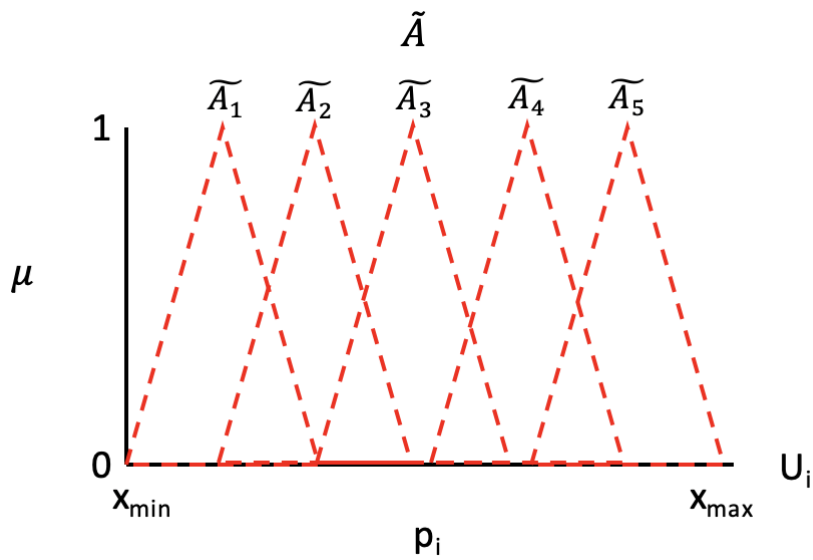


Figure 6.1. Example input parameter with 5 MFs (represented by a T1 system for visual simplicity)

Input parameters, p, may be assigned different weights for different priority levels as desired. These levels of priority may also be modeled using the rule base (which shall be further discussed). Assigning input parameter weights (as opposed to incorporating the weights into the rule base) may be a more straightforward task, as the rule base becomes large and complex with increasing number of inputs and correlating multiple antecedents. In assigning weights, w_i, the combined weights from input variables are such that they sum to 1 [123]:

$$\sum_{i=1}^{i} w_{p_i} = 1 \tag{6.3}$$

Here, each weight is used to distribute priority for each membership value used to determine the antecedent score a for the rule base such that

$$a = w_{p_1} * \mu_{p_1} + w_{p_2} * \mu_{p_2} + ... + w_{p_i} * \mu_{p_i} \tag{6.4}$$

6.2.2 Output

The performance score, $\mathbb{P} \subset [\, y_{min} \; y_{max} \,]$, is a crisp value defined on a universe of discourse V that is the output of the FIS. $\mathbb{P}$ can be numeric (e.g., ranging from 0 to 100) or mapped to a linguistic term (e.g., the letter grades, A, B, C,...) and contains an arbitrary number of fuzzy sets $\widetilde{C}_k$. Figure 6.2 shows an example output μ as a function of $\mathbb{P}$ using five MFs.

The Performance Evaluation and Review Framework of Robotic Missions (PERFORM) defines performance in this context as the ability of the platform under test to achieve mission or task objectives using quantified measures. These measures determine fitness for real mission deployment through satisfying defined standards or with demonstrated improvements in the case of comparing platforms. For additional degrees of uncertainty, IT2-FL (via its FOU) is flexible enough to take into account: (1) uncertainty directly associated with the input and (2) uncertainty from linguistic vagueness in defining performance. This topic will be discussed further in Section 6.5.

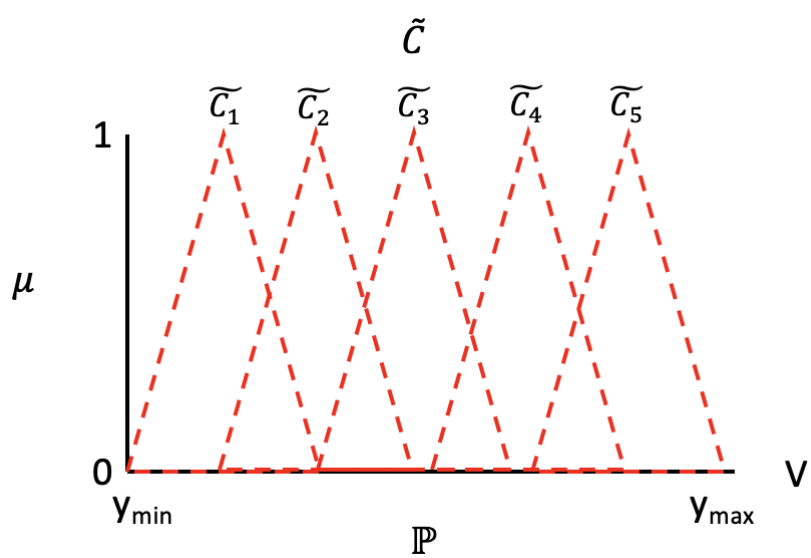


Figure 6.2. Generic output parameter with 5 MFs (represented by a T1 system for visual simplicity)

6.2.3 Data Collection

Ideally, data for the input variables is collected using an external suite of sensors, so as to maximize data consistency between platforms and minimize dependency on the platforms themselves. For underwater vehicles, some test ranges may have established underwater acoustic arrays or access to ship-based Ultra Short Base Line (USBL) systems to determine local/inertial coordinates and other navigational data. However, this may not always be possible for some desired data (i.e. battery levels) or if the vehicle does not have the capacity to carry additional sensors. In addition, sensor sampling rates should be consistent between test runs and set to an appropriate value that achieves a balance between required resolution and data size.

6.3 Membership Function Construction

In a generic fuzzy system, the three main design considerations are: (1) the membership functions themselves, (2) the number and nature of the selected rules, and (3) the inference technique [77]. Lotfi and Tsoi [78] found that adjusting the membership functions has a dominant effect over that of the other two design considerations. Wu and Mendel suggested using no more than seven MFs for each input to facilitate interpretation [128]. For the majority of FISs there are two strategies to construct MFs: model-driven strategies and knowledge-driven strategies [126]. Model-driven design usually occurs in the context of control systems where optimization algorithms to tune parameters may be used under the assumption that the system plant is known (or that at least a nominal model is known) [126]. The knowledge-driven approach utilizes an "expert" to design appropriate parameter values (i.e., from prior insight and experience). PERFORM has the ability to use either approach or a combination of the two. If simulations are available, that data can be incorporated by providing expected parameter ranges. Having clearly defined mission specifications and testing goals will help direct the test engineer in the knowledge-driven approach. With the flexibility of MF construction, one is able to model the quality of the performance, not merely whether or not the mission was a success or failure.

Although flexibility is provided in the implementation of PERFORM to account for various definitions of success, the author argues that the lowest threshold for measurement is the completion of a task. In other words, the evaluation system is not intended to scale for partial completion on a task level. To confidently deploy a vehicle for field operation, the vehicle must first demonstrate it can, at a minimum, complete the required task(s). As such, the performance measure then becomes the measure of the *degree* of success as defined and modeled via the input parameters and MFs. If desired, the task can be further decomposed into smaller, more manageable, components (i.e. subtasks) as necessary.

While any function may be implemented, this study will limit the MF shapes to triangular, trapezoidal, and Gaussian for the sake of simplicity and without loss of generality. The membership grades of p_i on A_i (i.e., $\mu_{A_i}(x)$) for each shape are described as in Table 6.1,

	MF $\mu_A(x)$	UMF $\mu_A^U(x)$	LMF $\mu_A^L(x)$
Triangular	$\begin{cases} 1-\lvert x\rvert & x<1 \\ 0 & otherwise \end{cases}$	$\begin{cases} 1-\lvert x_U\rvert & x_U<1 \\ 0 & otherwise \end{cases}$	$\begin{cases} 1-\lvert x_L\rvert & x_L<1 \\ 0 & otherwise \end{cases}$
Trapezoidal	$\begin{cases} \frac{x-a}{b-a} & a<x<b \\ h & b\leq x\leq c \\ \frac{d-x}{d-c} & c<x<d \\ 0 & otherwise \end{cases}$	$\begin{cases} \frac{x_U-a_U}{b_U-a_U} & a_U<x_U<b_U \\ 1 & b_U\leq x_U\leq c_U \\ \frac{d_U-x_U}{d_U-c_U} & c_U<x_U<d_U \\ 0 & otherwise \end{cases}$	$\begin{cases} \frac{x_L-a_L}{b_L-a_L} & a_L<x_B<b_L \\ e & b_L\leq x_L\leq c_L \\ \frac{d_L-x_L}{d_L-c_L} & c_L<x_L<d_L \\ 0 & otherwise \end{cases}$
Gaussian	$e^{\frac{-(x-m)^2}{2\sigma^2}}$	$\begin{cases} e^{\frac{(x_L-m_1)^2}{2\sigma^2}} & x_L<m_1 \\ 1 & m_1\leq x_L\leq m_2 \\ e^{\frac{(x_L-m_2)^2}{2\sigma^2}} & x_L>m_2 \end{cases}$	$argmin(e^{\frac{(x_U-m_1)^2}{2\sigma^2}}, e^{\frac{(x_U-m_2)^2}{2\sigma^2}})$

Table 6.1. Membership function definitions for triangular, trapezoidal, and Gaussian functions

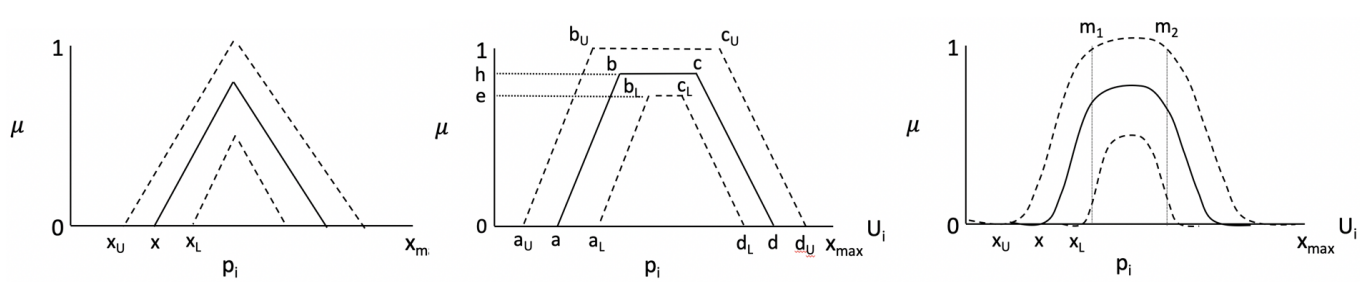


Figure 6.3. Three common MF shapes with associated UMFs and LMFs

and the associated plots are shown in Figure 6.3. Note that the Gaussian shaped function is drawn according to the mean as a design parameter, however the UMF and LMF may also be constructed by using an interval for the standard deviation. While there is no standard overlap value, sufficient overlap of the MFs is advised as it results in smoother functions, which are preferred over hard nonlinearities. As such, 25% to 50% overlap is considered common. Figure 6.4 exemplifies the effects of overlap (and lack thereof) on input-output curves. For this study, it is assumed each input parameter has an equal number of a fuzzy sets to simplify the rule generation process [123]. Also, for consistency, using identical MF shapes for each input and output is recommended [12].

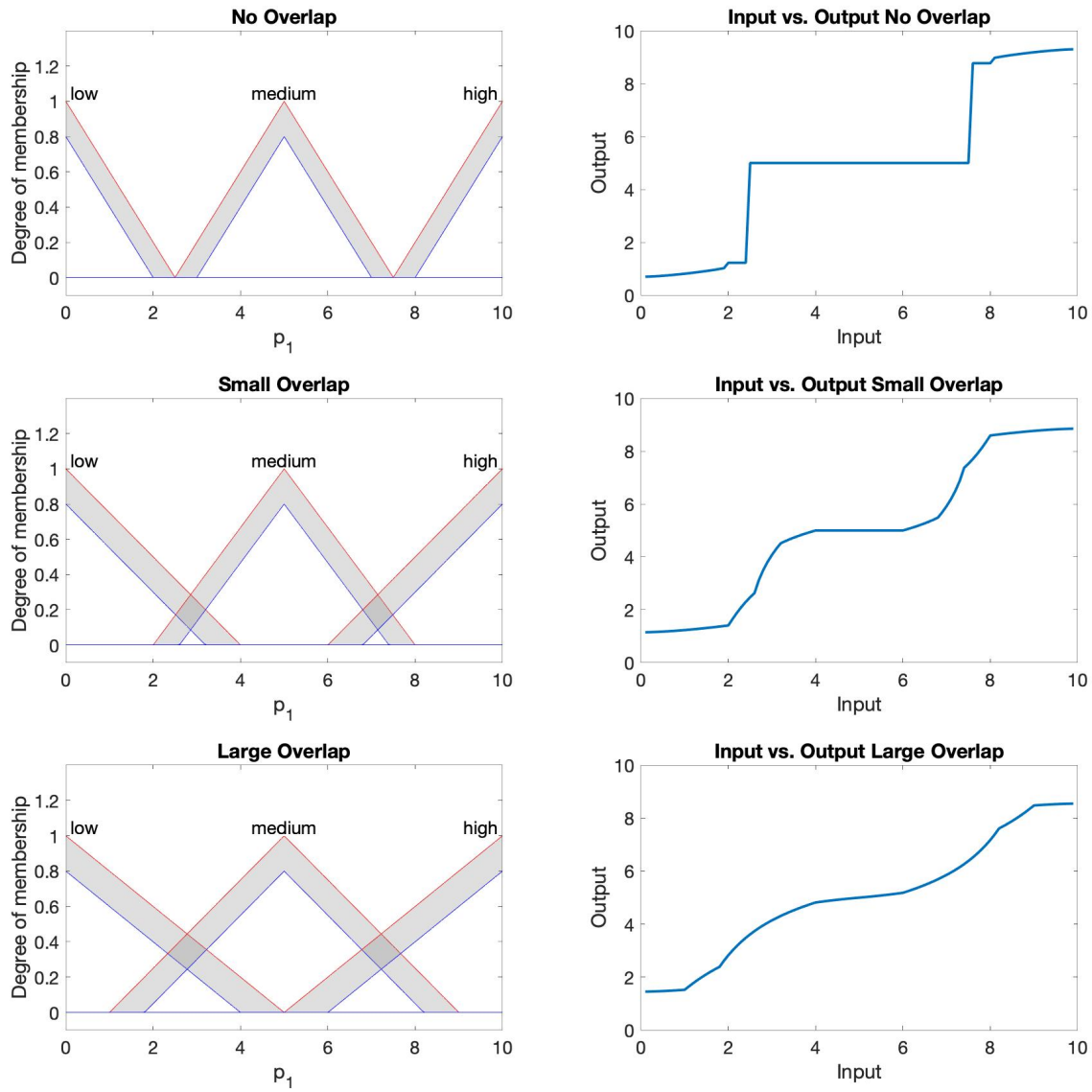


Figure 6.4. Various MF overlapping scenarios and the effect on the input-output relationship

6.4 Rule Base

Many FL systems use an Intersection Rule Configuration (IRC) [125]. This configuration uses multi-antecedent rules with a single consequent subset, as they incorporate an expert's perceived correlation between antecedents (Equation 6.5)[125]:

$$\text{If } p_1 \text{ is } \widetilde{A}_i \text{ and } p_2 \text{ is } \widetilde{B}_j, \text{ then } \mathbb{P} \text{ is } \widetilde{C}_k \tag{6.5}$$

where $\widetilde{A}$ and $\widetilde{B}$ denote the set of MFs associated with the input parameters p_1 and p_2, respectively, and $\widetilde{C}$ represents the set of MFs associated with the performance score, $\mathbb{P}$. If there are p input parameters, each with n MFs (under the assumption that each input is partitioned into the same number of MFs), a complete system would contain n^p rules. This exponential growth can lead to what is referred to as a "combinatorial rule explosion" where the number of rules and corresponding rule matrix can grow quickly and, as a result, implementation becomes impractical [28][94]. Genetic algorithms and other machine learning techniques have been developed to address the issue of rule base reduction with optimization [133]. However, these methods are not appropriate for this intended application, as the intention is to keep the human operator in the loop for test design.

Combs and Andrews [28] proposed an alternative rule configuration, called the Union Rule Configuration (URC), to address the aforementioned issue. Based on the proposition where s and q are the antecedents and r is the consequent:

$$[(s \cap q) \Rightarrow r] \Leftrightarrow [(s \Rightarrow r) \cup (q \Rightarrow r)] \tag{6.6}$$

Combs and Andrews argued that a series of single antecedent and single consequent rule relations provide comparable functionality (Equation 6.7), thereby increasing the rule base size linearly instead of exponentially.

$$\begin{aligned} &\text{If } p_1 \text{ is } \widetilde{A_i}, \text{ then } \mathbb{P} \text{ is } \widetilde{C_k} \\ &\text{If } p_2 \text{ is } \widetilde{B_j}, \text{then } \mathbb{P} \text{ is } \widetilde{C_k} \end{aligned} \tag{6.7}$$

There was significant discussion among the FL community about the validity of the approach due to equivalence (or lackthereof) between IRC and URC and whether the method could still model an expert's perceived correlation between antecedents. The method was eventually successfully defended and accepted as a suitable technique [28][90][34][27][125].

As an example, one may consider the following linguistic identifiers given to p_1, p_2, and $\mathbb{P}$ in Table 6.2.

p_1	
Very Fast	VF
Fast	F
Average	Av
Slow	S
Very Slow	VS

p_2	
Very Accurate	VA
Accurate	A
Satisfactory	S
Inaccurate	I
Very Inaccurate	VI

$\mathbb{P}$	
Very Good	VG
Good	G
Satisfactory	S
Poor	P
Very Poor	VP

Table 6.2. Linguistic identifiers and corresponding notation for p_1, p_2, and $\mathbb{P}$

Using the IRC, each specific component of this 2-input and 1-output system contains 5 MFs. The resulting symmetric rule matrix is of dimension 5x5 with 25 rules as shown in Table 6.3

	VF	**F**	**Av**	**S**	**VS**
VA	VG	VG	G	G	S
A	VG	G	G	S	P
S	G	G	S	P	P
I	G	S	P	P	VP
VI	S	P	P	VP	VP

Table 6.3. $p_1 - p_2$ rule matrix using the IRC. The horizontal axis refers to the p_1 linguistic identifiers and the vertical axis refers to the p_2 linguistic identifiers.

as opposed to that URC which only necessitates 10 rules with two 1x5 matrices (shown in Table 6.4) using the following propositional rule structure 6.8 [27]:

$$\begin{gathered}[(s \Rightarrow r) \cup (q \Rightarrow r)] \\ [(p_1 \Rightarrow \mathbb{P}) \cup (p_2 \Rightarrow \mathbb{P})]\end{gathered} \tag{6.8}$$

p_1

VF	**F**	**Av**	**S**	**VS**
VG	G	S	P	VP

p_2

VA	**A**	**S**	**I**	**VI**
VG	G	S	P	VP

Table 6.4. Rule matrix using the URC

The two inputs are now considered as two single-input and single-output rules coupled by union. It is noted that provided the aggregation method is commutative, the rules may be executed in any order.

The performance evaluation application proposed can consider each input independently. In other words, each parameter is an independent measure of performance and each antecedent has a direct relationship with the consequent. As such, the URC is a valid selection under the constraint of additive separability. By maintaining distinct input parameters, modularity of the framework is supported and allows for the reuse of input parameter design for other tests. It is further recommended that the URC be used when scalability is necessary and when the number of inputs exceeds three variables.

6.5 Incorporating Uncertainty

A major reason why IT2-FL was selected for the evaluation framework was for its ability to take uncertainty into account. Two types of uncertainties are considered in the design of the FOU: measurement uncertainties (from the testbed and data logging sensors) and linguistic uncertainties (subjective views regarding performance and the linguistic definitions).

For this application, each testbed sensor used for the input parameters of the performance evaluation includes fuzzy MFs to account for specific sensor characteristics and uncertainties. For example, total distance traveled is determined using GPS. The uncertainty in the GPS measurement, found experimentally or provided by product specifications/documentation, is integrated into the MF FOU. Sensor measurement quality can also be affected by sources such as high noise levels and changing environmental conditions (e.g., humidity, rain, etc.) [88][15]. If desired, these methods may also benefit from added knowledge from a priori simulations giving further context to suitable measurement ranges/bandwidths and corresponding MF intervals for a given testbed.

Linda and Manic incorporated experimentally measured input uncertainty into the design of IT2-FL systems for a fuzzy controller [74]. Most implementations interpret the output uncertainty with the geometrical properties of the output centroid. However, Linda and Manic noted that

although there is correlation between the input and output uncertainty distribution, oftentimes the distribution is biased by the geometry of the MFs and not derived from the input data [74]. While it may not be practical or feasible to calibrate every sensor for every testbed, this consideration is an option if higher levels of accuracy are required using their design method.

For designing these uncertainty bounds related to linguistic terms, one strategy is to allow for lower uncertainty at end/decisive points. The linguistic term "low," for instance, if mapped to a domain $[0\ 10]$, would have the highest certainty at $x = 0$. As x increases, the less the certainty is associated with "low." In other words, the agreement of the description "low" between individual users decreases with increasing x. The designed FOU would then increase accordingly. One way to address the disparity in descriptive linguistics is to poll experts and other relevant parties regarding their opinion of the association between select words and corresponding values. These polls could, then, provide a basis with which to construct the MFs.

6.5.1 Fuzzy Logic vs. Probability Theory

A source of confusion, debate, and controversy is the linkage between FL and traditional probability theory (PT) [139]. The topic is worth acknowledging to provide clear reasoning for selecting FL over PT and the specific type of uncertainty represented by FL. Zadeh [136] captures the divergence succinctly:

> Viewed through the prism of partiality, probability theory is, in essence, a theory of partial certainty and random behavior. What it does not address—at least not explicitly—is partial truth, partial precision and partial possibility—facets which are distinct from partial certainty and fall within the province of fuzzy logic (FL).

Generally, FL is used to manage uncertainty problems related to vagueness while PT is used to measure uncertainty due to randomness [136].

Performance, as defined and implemented in this study, is largely a construct of perception-based information. While the thresholds for "good" performance may be numeric and data-based, the decision in choosing the actual threshold values originate from human perception of acceptable

risk, behavior, and required capabilities. Probability may predict the chance of completing a mission or task. However, due to its limitation of bivalent logic where every proposition is either true or false, probability is not able to provide the degree of quality from which to make well-informed judgements. The predictions based upon probability would also require numerous assumptions on precisely known values and information of these same probabilities, neither of which are practical for an unpredictable real-world environment [136].

6.5.2 Field of Uncertainty Design

Linda and Manic discuss the relationship between input and output uncertainty in FISs [74]. Generally, for IT2-FL systems, input uncertainties have been implemented as analogous with the input membership function FOU width [74]. From this information, Linda and Manic state the assumption that the " IT2 FLS also acts as a functional mapping between the system input and output uncertainty. The application of such functional mapping is the presence of a correct uncertainty measure in the output of the IT2 FLS, which constitutes additional and very valuable information."

The principal MF is centered between the bounds of the FOU. The uncertainty bandwidth, μ_{bw}, is the the distance limited by $[\mu_{bw}^L \, \mu_{bw}^U]$ at $x = x_o$ (Figure 6.5). Here, x_o is a point on the universe of discourse, and $[\mu_{bw}^L \, \mu_{bw}^U]$ denote the corresponding locations on the lower and upper membership functions, respectively [12]. Oftentimes in literature, μ_{bw} is the value used in reference to FOU size. A second bandwidth value is defined here, x_{bw}, however, as this is considered more appropriate for defining FOU size related to sensor uncertainty.

Increasing the area of the FOU corresponds to an increased uncertainty. In the two-step output processing stage, type reduction is a mapping from a T2 fuzzy set to that of a T1 fuzzy set. Defuzzification then maps the T1 set into a crisp value. Hence, the type-reduced set provides a measure of the uncertainty similar (but not equal to [59]) a confidence interval in statistics where the type reduced set and defuzzified value is analogous to that of the standard deviation and mean of a random variable, respectively [89].

AND operator method	min
OR operator method	max
Implication Method	min
Aggregation Method	max
Defuzzification Method	centroid

Table 6.5. Matlab FIS settings

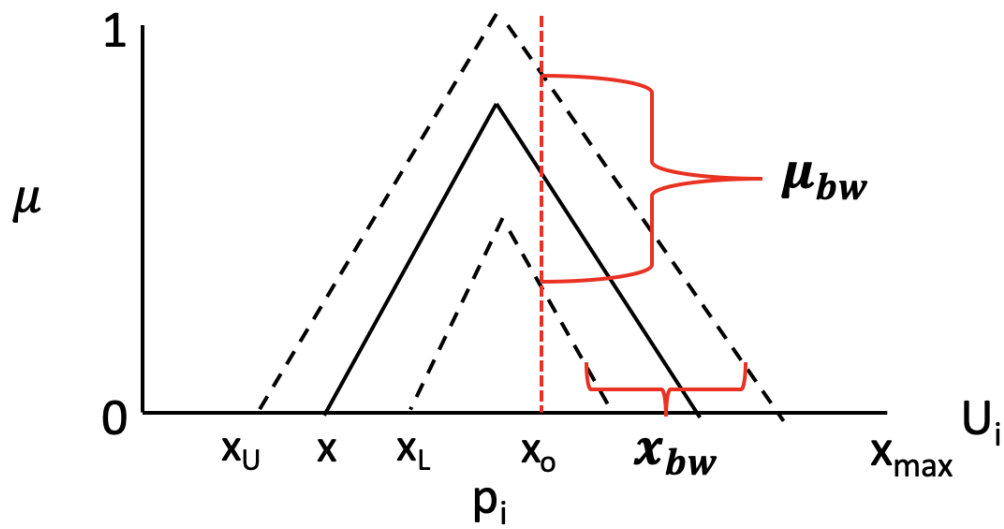


Figure 6.5. Visual representation of μ_{bw} and x_{bw}

6.5.2.1 Simulations

To demonstrate the effect of FOU size on the FIS output, several simulations are provided in Figure 6.6. The figure displays three example scenarios for a given SISO system: a small FOU (top), a medium FOU (middle), and a large FOU (bottom). All three FOUs have a constant μ_{bw}. The sub-figures in the left column represent the input MFs for a generic parameter, p_1. The output MFs (not shown here) are the same shape and size as that of the input MFs for each respective FOU. The value $p_1 = 6$ for this example demonstrates a point which lies on two overlapping MFs. The right column is the output aggregation with the associated defuzzified crisp output. The FIS settings used with Matlab software for all simulations in the section are given in Table 6.5.

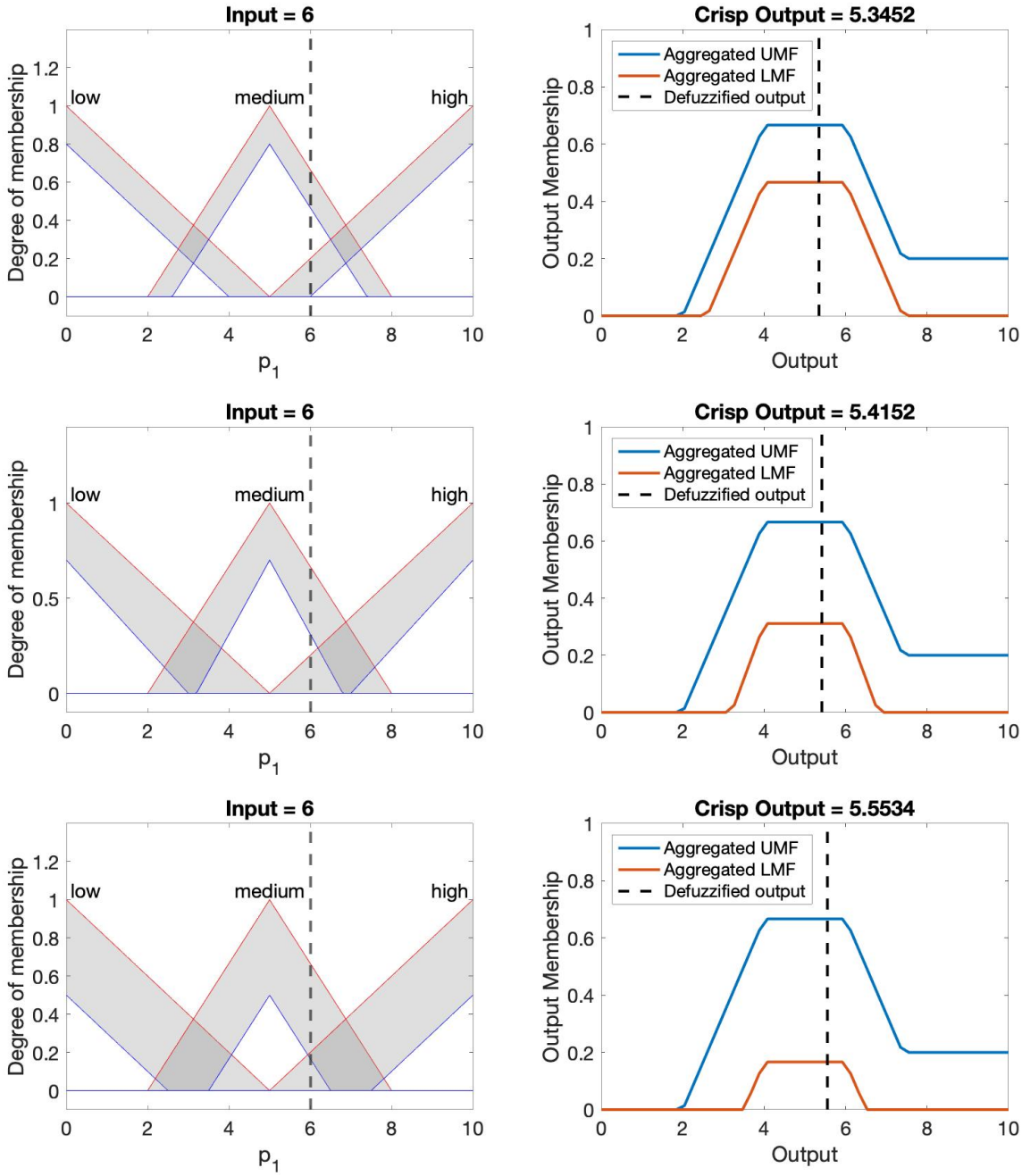


Figure 6.6. FOU and corresponding FIS output for constant μ_{bw} for $p_1 = 6$: small FOU (top), medium FOU (middle), large FOU (bottom)

Figure 6.7 uses the input MFs from the example in Figure 6.6 and shows the output response of the output for each location, x_o, in the universe of discourse. The simulation uses a resolution of 0.1 units.

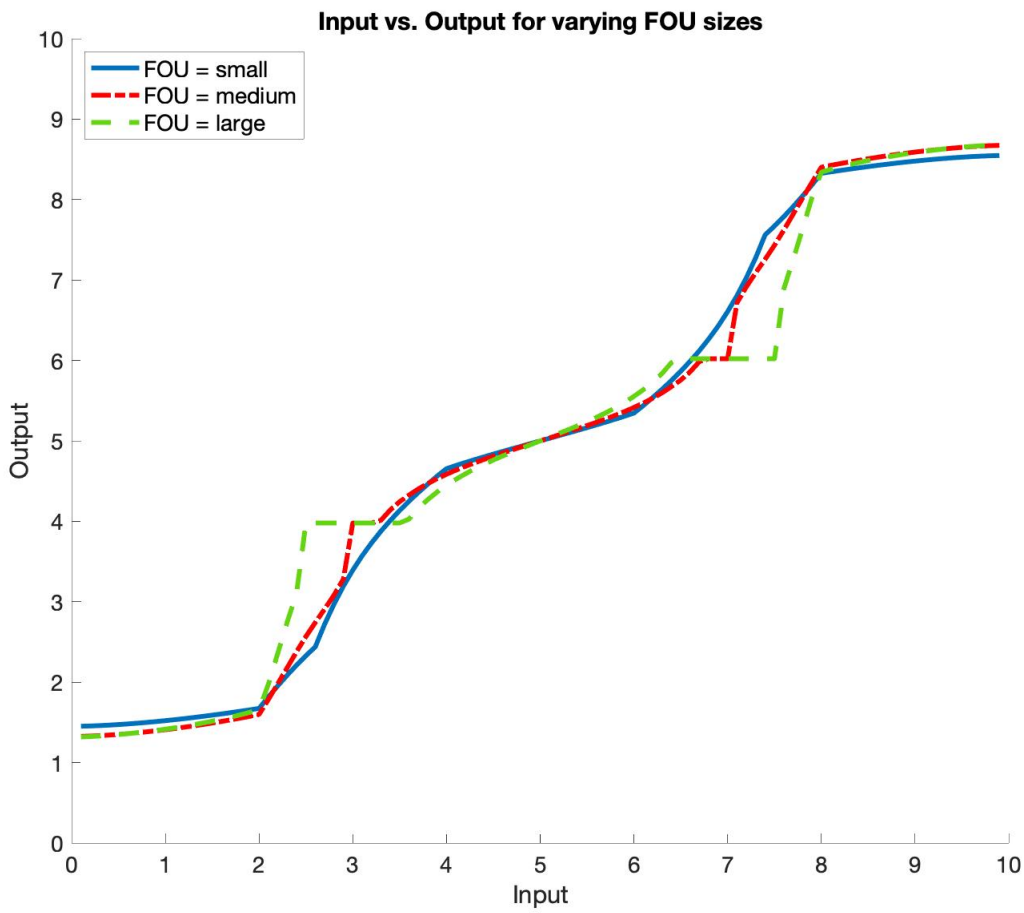


Figure 6.7. FIS output-input relationship for varying FOU sizes and constant μ_{bw}

Monte Carlo analysis is also performed to observe the FIS response to varying FOU sizes and the results are provided in Figure 6.8. N inputs are randomly generated between 6 and 7 (arbitrarily chosen interval that contains 2 MF's with sufficient overlap) with a uniform distribution to observe output perturbations. The top, the middle, and the bottom plot correspond with the small, medium, and large FOUs, respectively. The associated variance between the randomly generated input and the output value is provided in Table 6.6.

FOU size	Variance
Small FOU	0.0057
Medium FOU	0.0092
Large FOU	0.0316

Table 6.6. Variance associated with each FOU size for a constant μ_{bw}

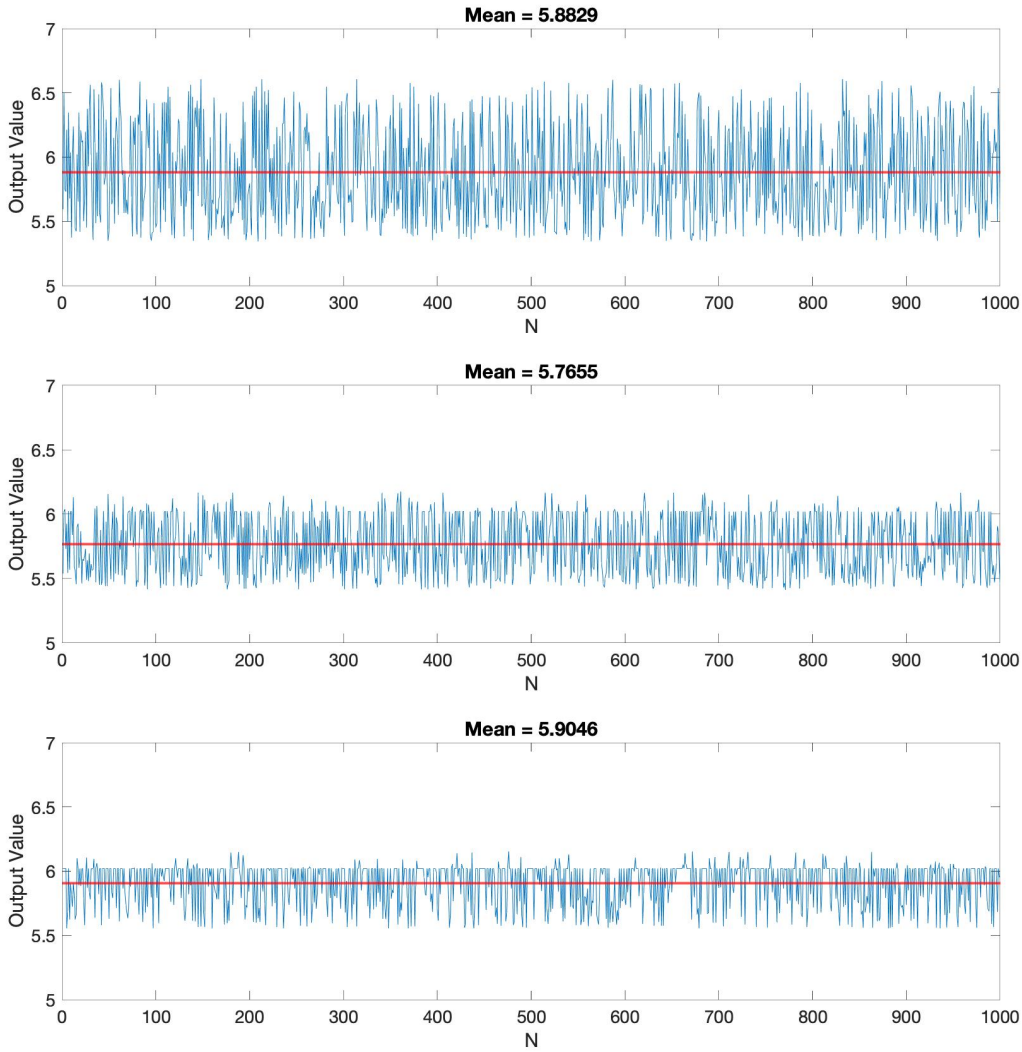


Figure 6.8. Monte Carlo results for small (top), medium (middle), and large (bottom) FOUs using uniformly distributed random values between 6 and 7 and for a constant μ_{bw}

The same set of simulations are performed for input MFs with increasing μ_{bw} (Figures 6.9 - 6.11). Table 6.7 provides the resulting variance values for Figure 6.11.

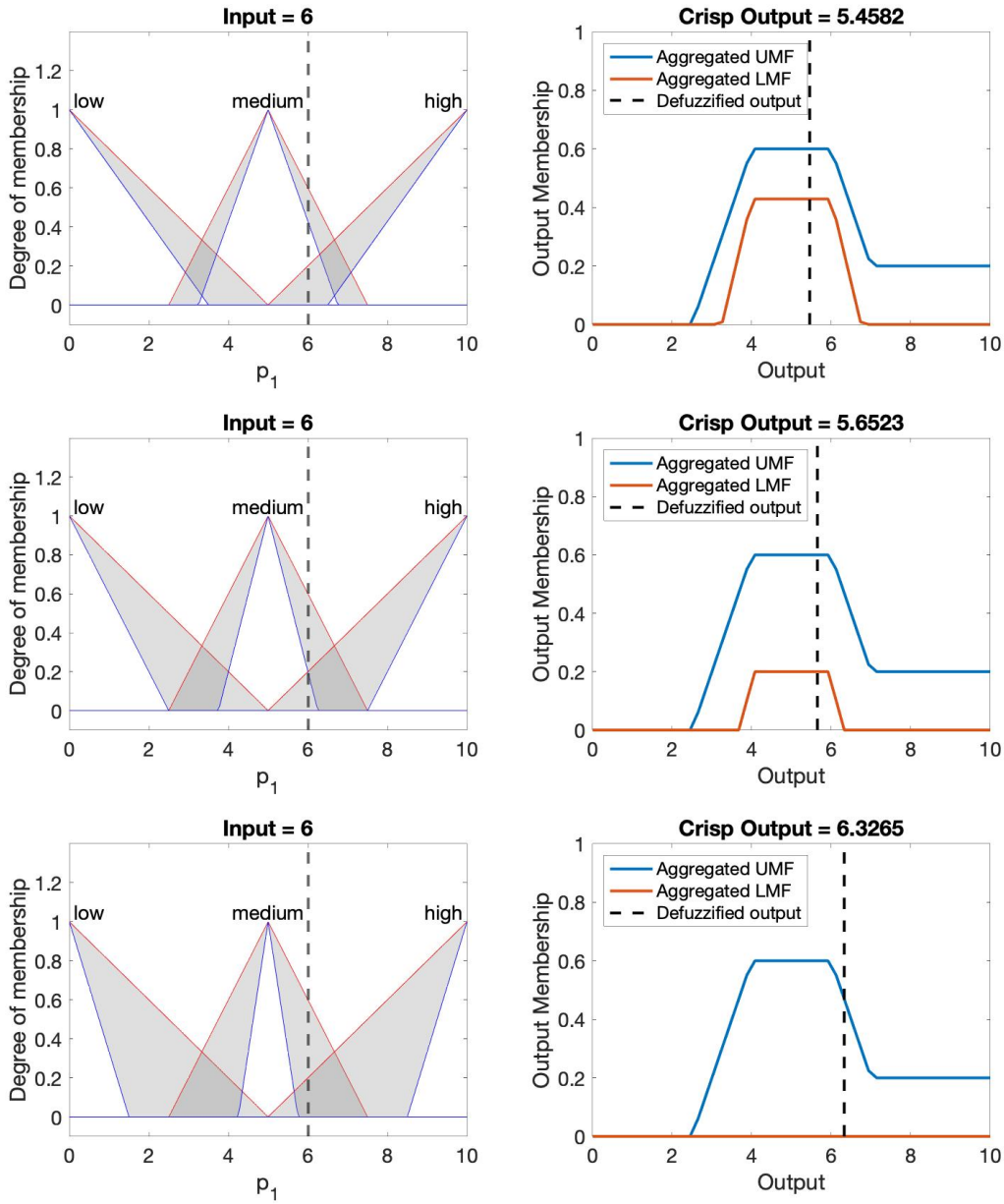


Figure 6.9. FOU and corresponding FIS output for increasing μ_{bw} for $p_1 = 6$: small FOU (top), medium FOU (middle), large FOU (bottom)

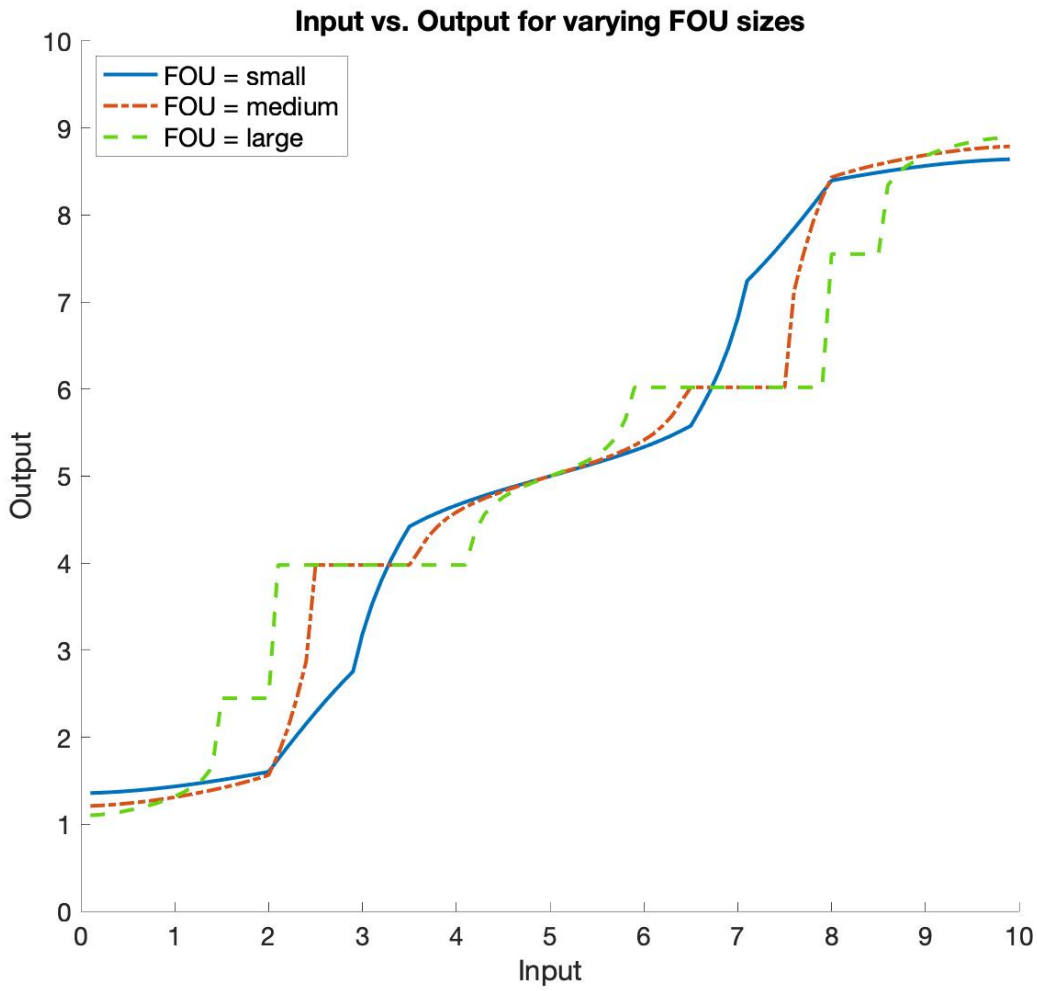


Figure 6.10. FIS output-input relationship for varying FOU sizes and increasing μ_{bw}

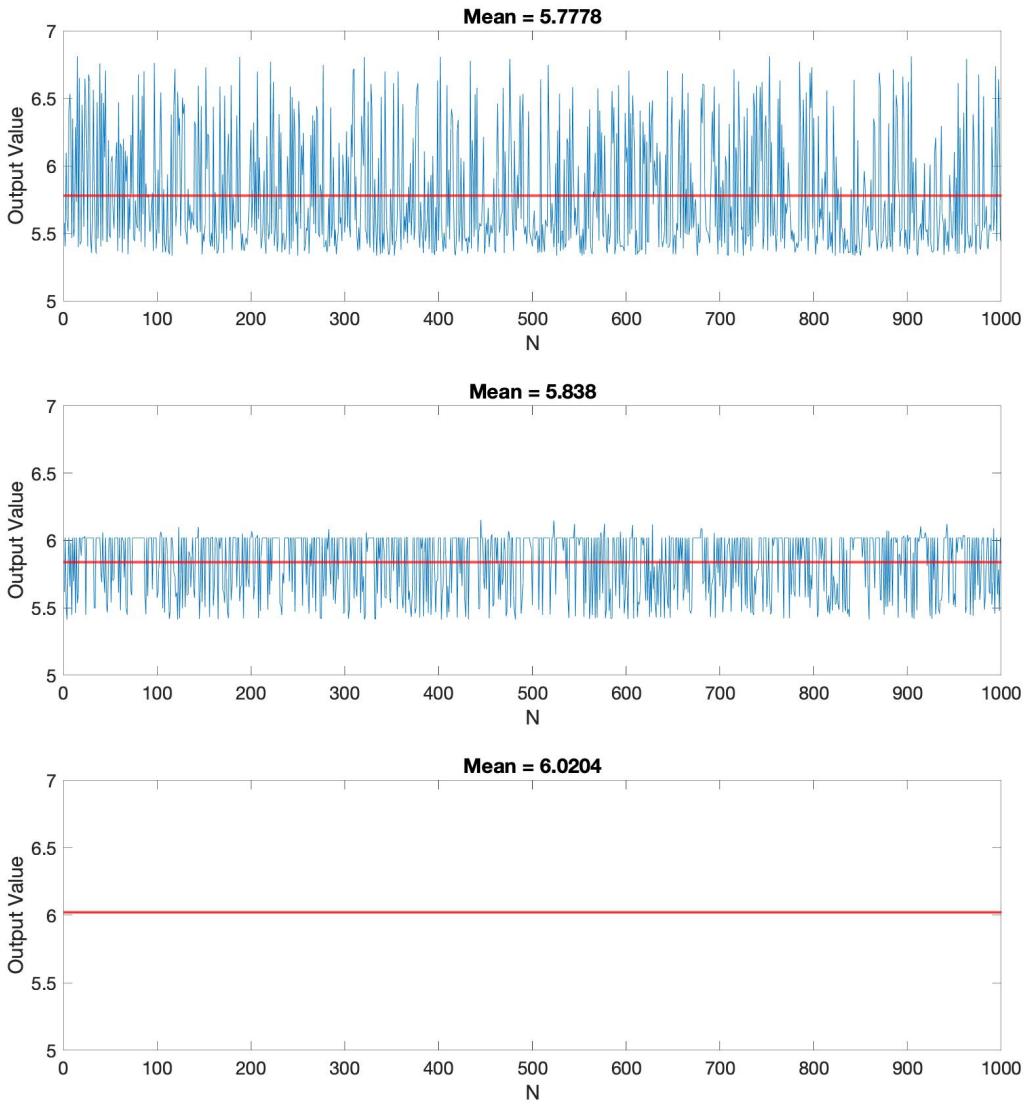


Figure 6.11. Monte Carlo simulation for a small (top), medium (middle), and large (bottom) FOU using uniformly distributed random values between 6 and 7 for an increasing μ_{bw}

6.5.2.2 Analysis

Since no closed form solutions exist to calculate c_R and c_L (noting that the Karnik-Mendel algorithm discussed in Section 5.2 provides only an approximation), it is difficult to characterize exactly how these end-points are affected by the geometric properties of the FOU [92][91][12].

FOU size	Variance
Small FOU	0.0327
Medium FOU	0.0188
Large FOU	0.0827

Table 6.7. Variance associated with each FOU size for an increasing μ_{bw}

Little research exists on the topic. But, while it is generally accepted that the area of the FOU, in addition to the upper and lower MF centroids of the FOU, correlate to the degree of uncertainty, the behavior of the system in response to different levels of uncertainty has not yet been generalized.

Some characteristics to consider include noting that the input and output mapping is continuous when the input MFs fully span the domain space [128], as demonstrated in Figures 6.7 and 6.10. Additionally, if the lower MF does not fully span the domain, the input-output mapping may have "jump" discontinuities (i.e., hard nonlinearities) [128]. This behavior is demonstrated for the large FOU in Figure 6.10. Jumps are noted at an input value of approximately 2 and 8 and correlate with the gaps in the lower MF of the large FOU in Figure 6.6.

In Figure 6.6 & 6.9, it is observed that the spacing between the aggregated UMF and LMF increases with increasing FOU size. This agrees with current literature, where the output uncertainty is associated with a correct mapping of the input uncertainty. For constant μ_{bw}, the crisp output is biased to the left of input value 6 with decreasing FOU size. In the increasing μ_{bw} scenario, the same is true for the small and medium FOU. However, for the large FOU the output is biased towards the right of the input. This may be due the lack of a LMF from both the *medium* and *high* MF in the vertical slice at $p_1 = 6$.

Observing the input vs. output relationships for constant μ_{bw}, the values for each FOU size match fairly well, except for regions with MF overlap. The smallest FOU produces the smoothest output. The largest FOU has regions of steep gradients and sudden plateaus. Increasing μ_{bw} generates significantly more variation between FOU sizes. It is seen that the smallest FOU produces the smoothest output in this case as well.

The Monte Carlo simulations performed for constant μ_{bw}(Figure 6.8) show that variance increases with increasing FOU size. In all cases, the output was reduced to roughly a range of 5.5 to 6.5 with inputs between 6 and 7. For the medium and large FOUs this range is decreased further to a range roughly between 5.5 and 6.0. While the small FOU appears to have the most fluctuation in values, this is most likely due to the output value having a smaller difference with the corresponding input value as a result of having less uncertainty, as supported by the variance values in Table 6.6.

Analyzing the Monte Carlo results for increasing μ_{bw} (Figure 6.11), variance does not appear to increase with increasing FOU size. The most obvious difference occurs in the bottom plot where the output is constant for inputs between 6 and 7. This result matches the corresponding region in Figure 6.10. For the small FOU, the output extends further toward 7 than that in the case of a constant μ_{bw}. The medium FOU output is roughly bounded between 5.5 and 6.0 similar to the constant μ_{bw} output.

From the gathering of published research and the simulation studies presented above, it is recommended to design the FOU using a trial-and-error method. The appropriate input-output mapping is expected to be context-dependent and is left to the user's discretion according to his/her desired modeling characteristics. Simulations as the above may be performed and manipulated to help analyze and shape the output space.

6.6 Task Weighting

To weight the scoring parameters, an assessment is made as to the importance and the specific aspect of each task used to generate a particular score to the overall successful completion of a mission. Greater weights are allocated to tasks that hold higher importance with respect to mission success. Analytic Hierarchy Process (AHP) is a method that could be used to rank and prioritize (sub)tasks and is well established in literature. Even so, the process in determining specific weights for each (sub)task is considered to be beyond the scope of this research.

In this work, a normalized weighted mean is used to calculate a score for each level of (sub)task, producing a final single score $\mathbb{P}_{total}$ given individual scores $\mathbb{P}_i^k$ such that

$$\mathbb{P}_{total} = \sum_{n=1}^{N} w_n \mathbb{P}_n^1 \tag{6.9}$$

$$\mathbb{P}_{task} = \sum_{n=1}^{N} w_n \mathbb{P}_n^k \tag{6.10}$$

noting that this weight is the relative importance level of the task to the overall mission, not the weighting for the MF input parameters. Here, N denotes the total number of scores in the weighted sum, n refers to the (sub)task number, w is the weight associated with the specific score, and k specifies the (sub)task level of the decomposed tree. Figure 6.12 presents an example flow chart of the calculated scores. The fuzzy inference process used in the proposed autonomy evaluation framework in this research (discussed in Chapter 7) occurs at the lowest level of the tree for each respective (sub)task. In Figure 6.12, this translates into *Task 1* and *Task 3* of Level 1 implementing a fuzzy process at Level 2, while *Task 2* (as it is not further decomposed) would implement the fuzzy process at Level 1.

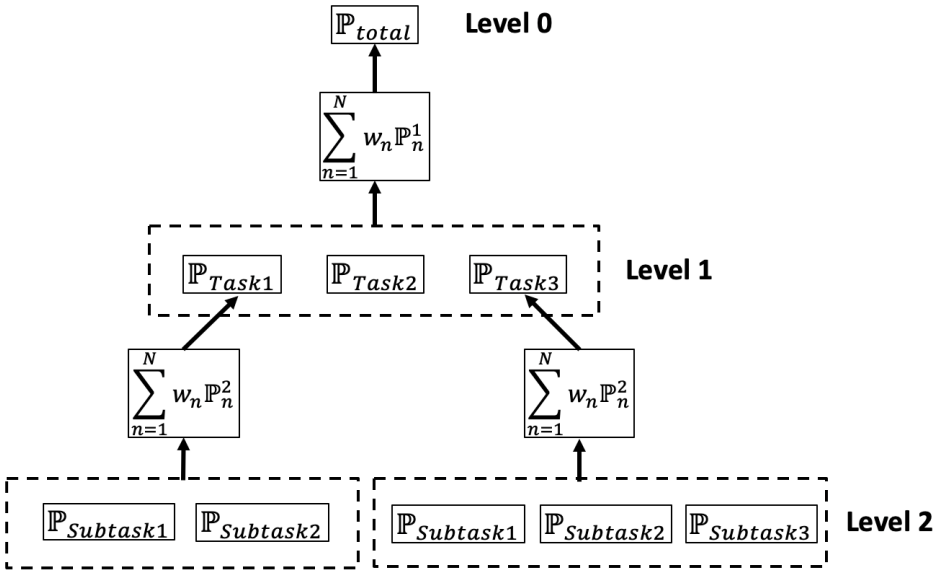


Figure 6.12. Generic example of performance score calculations decomposed by task

6.7 Generalized Design Procedure

For the proposed FL-based evaluation framework and given mission performance criteria, the Performance Evaluation and Review Framework of Robotic Missions (PERFORM) autonomy evaluation procedure is summarized as follows:

1. *Determine user-specified testbed parameters (e.g., the size of the test area and its location, choosing between a two or three-dimensional environment).*

2. *Select the input performance parameters (e.g., total distance, time, etc.) of interest.*

3. *Determine appropriate performance measurement tools/criteria (i.e., sensors) and corresponding uncertainty levels (e.g., GPS accuracy limits for measuring navigational coordinates).*

4. *(Optional) Use simulations (performed a priori) to provide further context to choosing suitable MF intervals and insight into expected parameter values.*

5. *Generate appropriate MFs using insight gathered from (3) and (4).*

6. *Construct a rule base to reflect the desired input-output mapping.*

7. *Perform autonomy test missions.*

8. *Gather and post-process data from (7) to use as inputs to the IT2-FL Fuzzy Inference System (FIS).*

9. *Obtain overall performance score(s) from IT2-FL evaluation method for final evaluation/comparison of autonomous platform(s)/engine(s).*

It is of note that each parameter (performance criteria) may be individually analyzed, in addition to the overall autonomy performance score represented by the final FIS crisp output. The autonomy performance testing method proposed in this work is intentionally modular and scaleable in design,

allowing for testing as simple (or as complex) as the test objectives warrant. The following chapter will present a series of case studies to demonstrate various test scenarios as a proof of concept for PERFORM.

CHAPTER 7

APPLICATION OF PERFORM VIA SELECTED TASK-BASED CASE STUDIES

7.1 Case Study Overview

For demonstration purposes (and without loss of generality), the Performance Evaluation and Review Framework Of Robotic Missions (PERFORM) incorporates two different autonomous path planning techniques for evaluation and direct comparison: (1) a multi-layered Potential Field Method / A-Star (PFM/A*) approach [30] and (2) a Probabilistic Roadmap (PRM) method [61]. Details on both algorithms may be found in Chapters 2 & 3. The outline below provides the title of each case study and the corresponding goal of each validation test for PERFORM (from a test engineer's point-of-view and without loss of generality), and the specific criteria for performance parameters. The overall performance score of the path-planning autonomy is the output for all 3 cases.

- **Case Study I: Waypoint navigation with single obstacle**
  - Evaluate a vehicle's ability to detect an unknown stationary obstacle (if any).
  - Re-plan a path to safely avoid any such newly discovered obstacles.
  - Reach the goal waypoint(s) with an acceptable path length
  - **Input Parameters:** the vehicle's total distance traveled and the closest point of approach (CPA) to any existing obstacle
- **Case Study II: Waypoint navigation with multiple obstacles**
  - Evaluate a vehicle's ability to detect multiple unknown stationary obstacles.

Table 7.1. Summary of gain values used for simulations

Case Study	*I*	*II*	*III*
α	100	200	100
β	10,000	9,000	10,000
k_A	300	100	300

  - Re-plan a path to safely avoid any such newly discovered obstacles with more emphasis placed on a conservative trajectory than that in Case Study I.
  - Reach the goal waypoint(s) with an acceptable path length.
  - **Input Parameters:** the vehicle's total distance traveled and the closest point of approach (CPA) to any existing obstacle

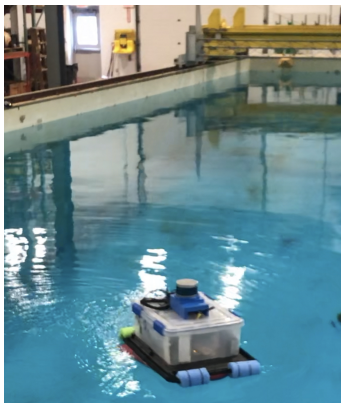
- **Case Study III: Area survey (i.e. lawnmower path survey)**
  - Evaluate how efficiently the autonomy engine is able to complete an area survey mission.
  - Complete a lawnmower path while minimizing gaps in data coverage.
  - Maintain an appropriate speed for high quality data collection.
  - **Input Parameters:** path percent error and average speed

For the given test scenario, an analytical binary occupancy grid is generated a priori and provided to the path planners to allow them to differentiate between free and occupied space. It should be noted that the path planners are deliberately left untuned to generate non-optimal paths to better simulate experimental test data and provide higher-integrity data for observing the efficacy of the proof-of-concept IT2-FL autonomy testing and evaluation framework. Routes given are not intended to represent planner capabilities, but to give a reasonable representation of the actual path a vehicle might take given commands generated from the path planners. The specific gain values used in the simulations for the PFM/A* algorithm in each case study are given in Table 7.1. Specific tuning values for PRM used for the case study simulations are given in Table 7.2.

Table 7.2. Summary of tuning values used for simulations

Case Study	*I*	*II*	*III*
Number of Nodes	1000	2000	1000
Maximum Neighbor Distance (m)	1	1	1
Maximum Number of Neighbors	3	3	3

(a)



(b)

Figure 7.1. (a) The Chase Engineering Tank located at the University of New Hampshire (b) Small-Scale ASV Experimental Platform. Simulations are based upon the laboratory equipment shown here.

Laboratory autonomy testing is performed at the Jere A. Chase Ocean Engineering Laboratory (Figure 7.1a) at the University of New Hampshire (UNH). With dimensions of 18m x 12m x 6m, the UNH Engineering Tank allows for rapid, multi-seasonal testing with both surface and underwater vehicles. The experimental platforms used for this research are small-scale, differential thrust Autonomous Surface Vehicles (ASV), referred to as Testing Unmanned Performance PlatformS (TUPPS). The testbed vehicle has a base width of 0.6m and a length of 0.9m and is outfitted with a Velodyne VLP-16 lidar for obstacle detection (Figure 7.1b). This laboratory test environment is used as the basis for the simulations in this Chapter to first demonstrate proof-of-concept of PERFORM. It is emphasized that all test data generated for this study are via analytical simulations mimicking test data obtained from the UNH Engineering Tank.

7.1.1 Testbed Data Collection

It is assumed that the sensors and vehicle are identical for all simulations so as to enable the direct comparison of the path planning algorithms. The positioning system used for this testbed is a Marvelmind HW v4.9-NIA indoor positioning system. By taking into account the measurement noise produced from this signal, improvements are expected for the IT2-FL techniques [85]. Marvelmind company documentation gives a measurement uncertainty value of +/- $0.02m$ [2]. From the obstacle configuration given in Figure 7.7, the shortest path to the goal location (including the obstacle) based on Euclidean distance is used as the minimum of the input range for total distance. The maximum distance used in the MF is arbitrarily defined to be twice the minimum distance and can be adjusted depending on the user's acceptable tolerance. Any test run value greater than the maximum will automatically default to this saturated maximum value.

The "total distance" in this work is calculated by determining the overall sum of the changes in position such that

$$\sum_{k=1}^{n} d_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \tag{7.1}$$

where k, n, and d represent the measurement number, total number of measurements, and corresponding distance, respectively. To determine an appropriate uncertainty value, the combined uncertainty, u_c, is calculated using an l^2-norm such that

$$u_c = \sqrt{u^2(x_1) + u^2(x_2) + u^2(x_3) + ... + u^2(x_n)} \tag{7.2}$$

where $\mu(x_i)$ represents the uncertainty at position measurement x_i.

The number of measurements for a given test run is dependent on two factors: the sampling rate and total test time. It is assumed here that the sampling rate for the vehicle's position is the same for each specific mission or task. A value of +/- $0.75m$ is determined as a reasonable value for this case study based on previously observed test platform speeds and time ranges (results not shown here). Since determining the vehicle's CPA from the obstacle relies on a singular measurement,

the uncertainty value of +/-0.02m from the company's documentation is used as the FOU width for CPA-related MFs. Matlab settings for computing the FIS in this chapter are as follows:

Table 7.3. Matlab FIS settings

AND operator method	min
OR operator method	max
Implication Method	min
Aggregation Method	max
Defuzzification Method	centroid

7.1.2 Input Membership Function Construction

A summary of the evaluation parameter ranges and uncertainties for both total distance and CPA are provided in Table 7.4. From these values, the input MF's for total distance and CPA are given in Figure 7.2. The same input MF's are used for Case Study I and II (with II using only the 5-MF system).

Table 7.4. Range of values for construction of Case Study I and II MFs

Evaluation Parameter	*Total Distance (m)*	*CPA (m)*
Range	16.00 - 32.00	0 - 4.00
Uncertainty	+/- 0.75	+/- 0.02

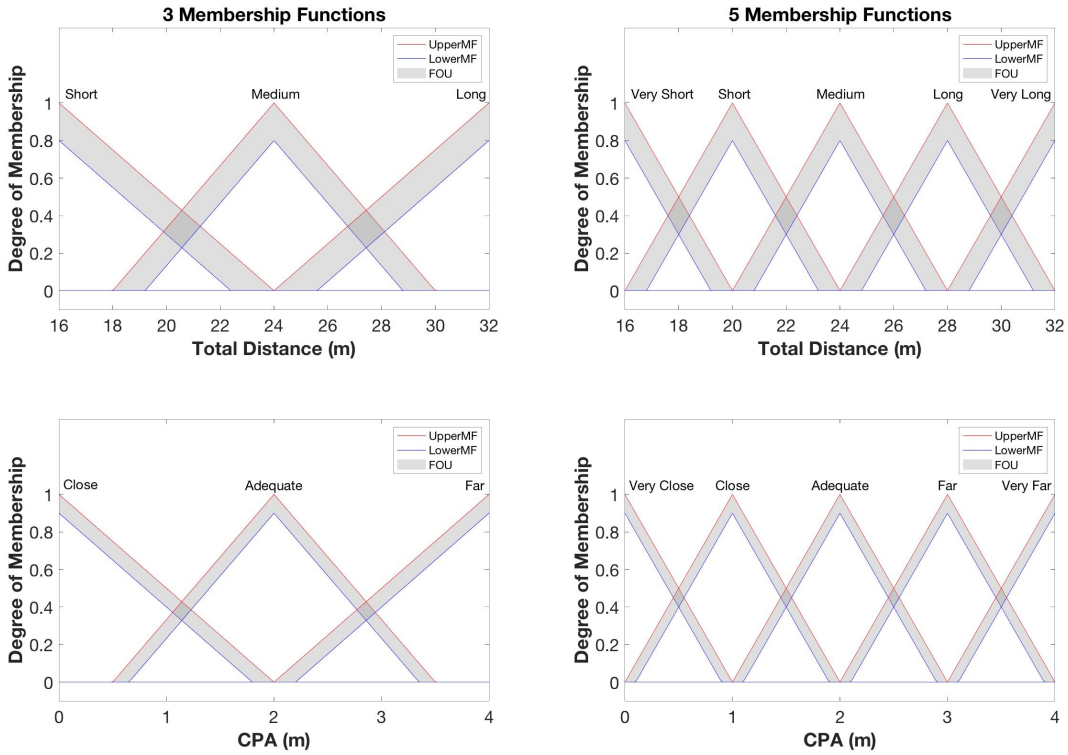


Figure 7.2. Membership functions for total distance and CPA

For Case Study III, the MFs are modified to represent the new parameters (path percent error and average speed) under this test environment and is shown in Figure 7.3. Both parameters use data from the testbed GPS unit, so the uncertainty bounds are designed taking this into account. 0.50% and 0.25m/s for path percent error and average speed, respectively, are deemed appropriate for the level of accuracy warranted in this case study. Variable ranges for Case III are summarized in Table 7.5.

Table 7.5. Range of values for construction of Case Study III MFs

Evaluation Parameter	*Path Percent Error*	*Average Speed (m/s)*
Range	0 - 10.0	0 - 4.0
Uncertainty	+/- 0.50	+/- 0.25

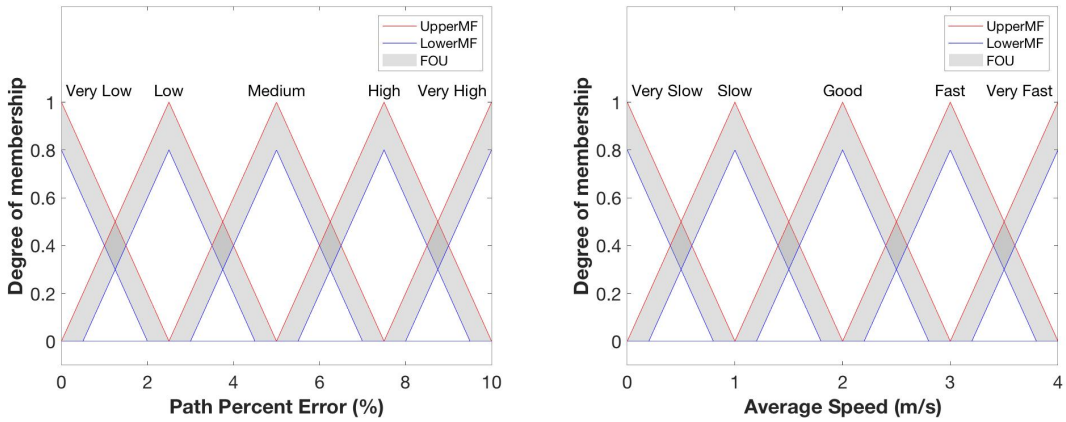


Figure 7.3. Membership functions for Case Study III to observe path percent error and average vehicle speed.

7.1.3 Performance Score Poll Data & Membership Functions

For the construction of the performance score MFs, a different approach is used to show the variety of factors that may be incorporated. The same performance MFs are used for Case Studies I, II, and III. The uncertainty in these MF's originate from the uncertainty in the linguistic terms. Linguistic terms tend to be subjective, or rather that these terms have different meanings to different individuals and thereby create yet another level of vagueness [88]. To address this, a separate study was performed where several individuals (members of the University of New Hampshire marine robotics teams) were polled to provide their opinions regarding the relations of linguistic terms to numerical values in order to construct the MFs. The resulting MFs are shown in Figure 7.6.

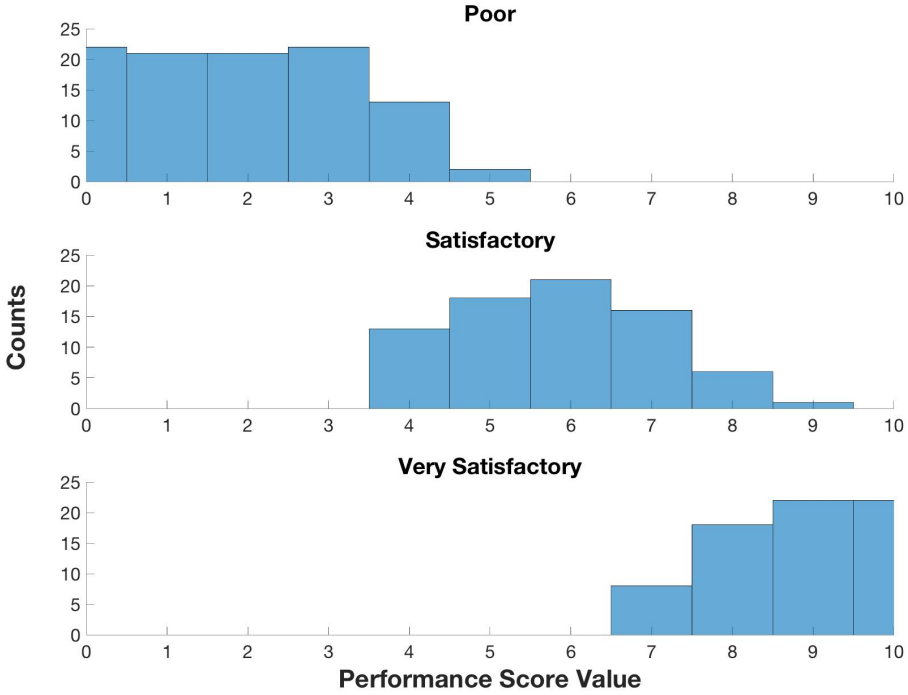


Figure 7.4. Histogram of polled data associating linguistic terms, relating overall performance scores with numerical values for a 3-MF system

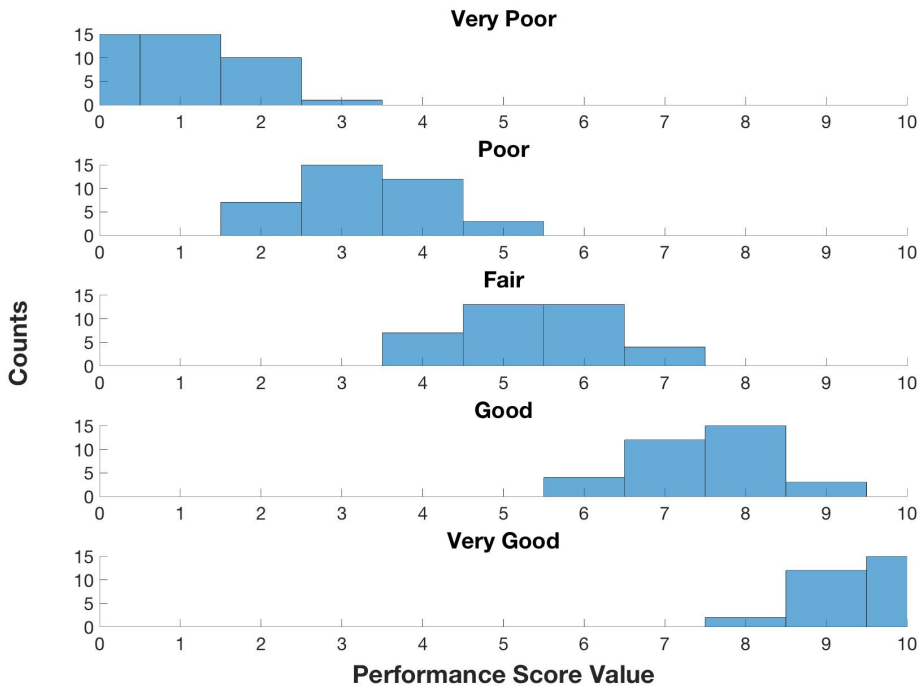


Figure 7.5. Histogram of polled data associating linguistic terms relating overall performance scores with numerical values for a 5-MF system

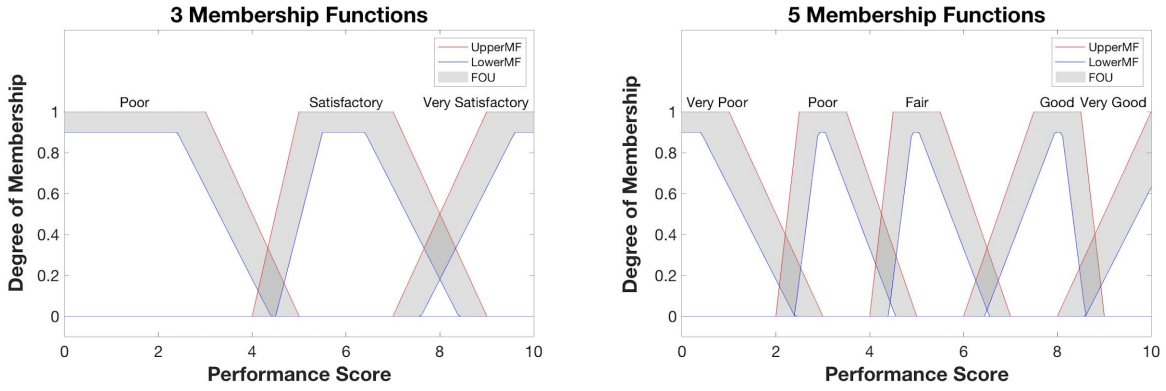


Figure 7.6. Membership functions for the performance score output

7.2 Case Study I: Waypoint navigation with single obstacle

As the first case study, an obstacle configuration given in Figure 7.7 is simulated to generate experimental test data and analyze the evaluation framework.

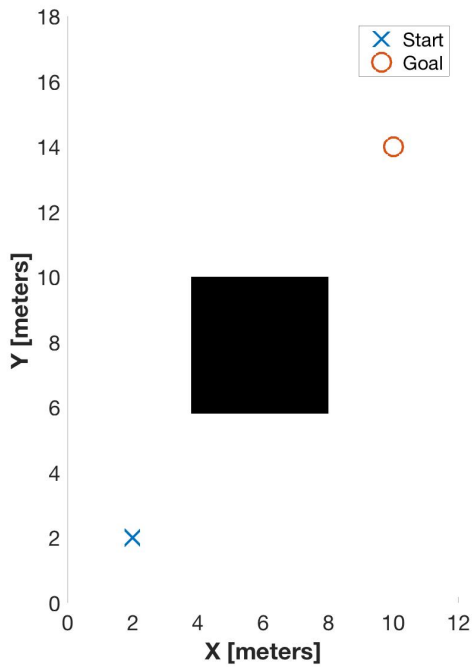


Figure 7.7. Testbed for Case I (obstacle represented by the black box)

Two different FIS are analyzed (a 3-MF system and 5-MF system) to observe what effects may result (if any) from varying the number of MF's.

The rule bases for this mission evaluation (arbitrarily chosen and without loss of generality) are summarized in Tables 7.6 and 7.7 for a 3-MF system and 5-MF system, respectively. The IRC is used since the number of inputs is small. Here, the linguistic input terms are italicized with CPA on the horizontal axis and path length given on the vertical axis. The linguistic pattern of Table 7.6 reads in the following manner:

1. *If the total distance is short and the CPA is Close, then the performance score is Satisfactory.*

2. *If the total distance is short and the CPA is Adequate, then the performance score is Very Satisfactory.*

3. etc.

Table 7.6. Summary of rules used for a 3-MF system

Linguistic Term	*Close*	*Adequate*	*Far*
Short	Satisfactory	Very Satisfactory	Satisfactory
Medium	Satisfactory	Very Satisfactory	Satisfactory
Long	Poor	Satisfactory	Poor

Table 7.7. Summary of rules used in the example for a 5-MF system

Linguistic Term	*Very Close*	*Close*	*Adequate*	*Far*	*Very Far*
Very Short	Fair	Good	Very Good	Good	Fair
Short	Fair	Good	Very Good	Good	Fair
Medium	Poor	Fair	Good	Fair	Poor
Long	Very Poor	Poor	Fair	Poor	Very Poor
Very Long	Very Poor	Poor	Fair	Poor	Very Poor

To encourage the vehicle to remain a safe distance away from an obstacle, the linguistic term "adequate" is mapped to a better performance score. The ideal scenario is to have the vehicle remain a safe distance away from any obstacles while also maintaining the shortest possible path to a given destination within the given test environment.

7.2.1 Results

The MF relationships between the input and output variables can be visualized as a 3D plot (Figure 7.8). This view shows the ideal parameters value, designated in yellow on the color scale and displayed as the maximum value on the z-axis. The objective for the vehicle in this case study, reflected in the rule base and defined numerically in the MFs, is to maintain a safe distance (determined to be $2m$) while minimizing the total distance to the goal (i.e., desired waypoint). As shown, the yellow portion of the plot corresponds to these goals spanning the area defined by a 2m CPA and the region extending from $0m$ to roughly $24m$ of total distance traveled.

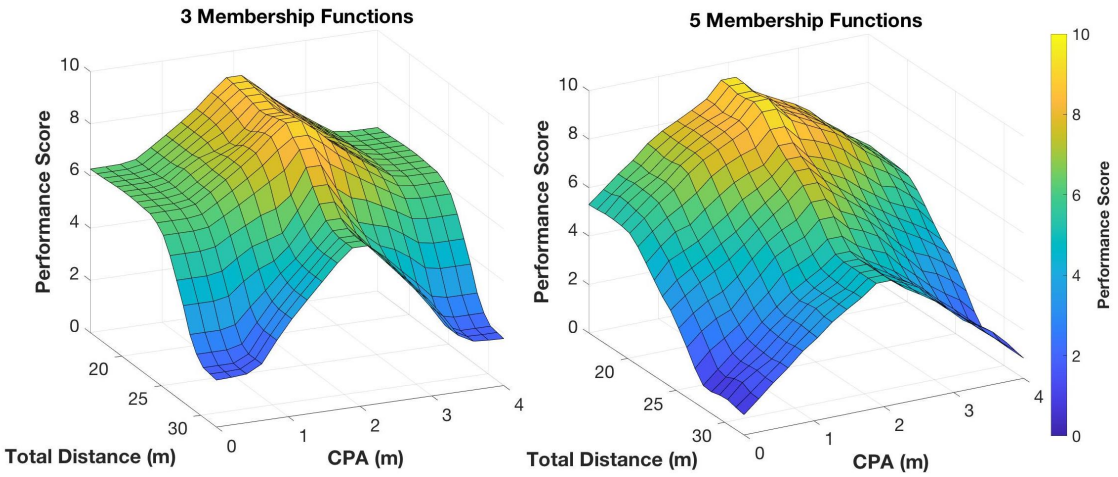


Figure 7.8. 3-D MFs plots relating input variables (total distance and CPA) to corresponding output variables (performance scores)

Figure 7.9 provides a side-by-side comparison between the PFM/A* path planning method (left) and that of the PRM method (right). Table 7.8 summarizes the simulated performance parameter values. An additional route is given in Figure 7.10 for additional context to demonstrate a path that would generate a poor score. This route (referred to as a "Generic Planner") is significantly more circuitous than either of the PFM/A* and PRM methods and also traverses much closer to the obstacle than is desired.

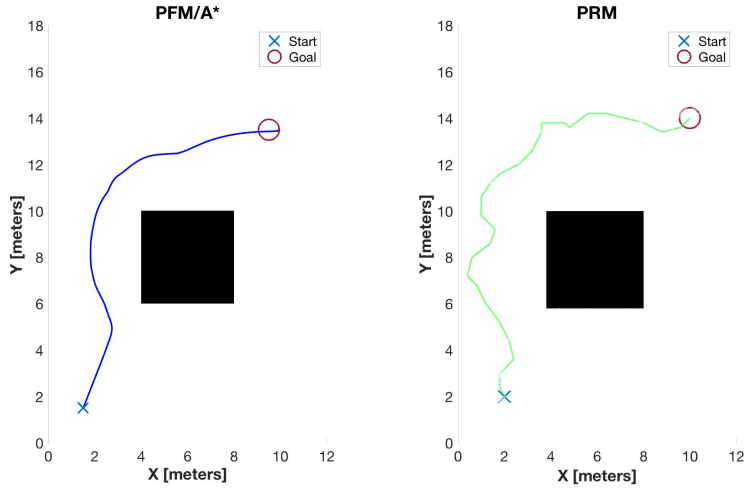


Figure 7.9. PFM/A* generated path (left), PRM generated path (right)

Table 7.8. Summary of simulated performance values for Case Study I

Evaluation Parameter	*Total Distance (m)*	*CPA (m)*
PFM/A*	19.28	1.50
PRM	21.52	2.48
Generic Path	31.92	1.00

Performance score results from applying the FIS to the input values in Table 7.8 are shown in Table 7.9. The resulting output fuzzy set overlaid by the defuzzified outputs given in Table 7.9 are shown in Figures 7.11 - 7.12 for both 3-MF and 5-MF based systems for the PRM and PFM/A* planners, respectively. Visual observation shows that the PFM/A* route has the best performance given the mission goals. As expected, the generic planner scores the lowest. The difference in scores between the PRM and PFM/A* autonomy methods are negligible in the 3-MF case (0.03) and more obvious in the 5-MF case (0.81).

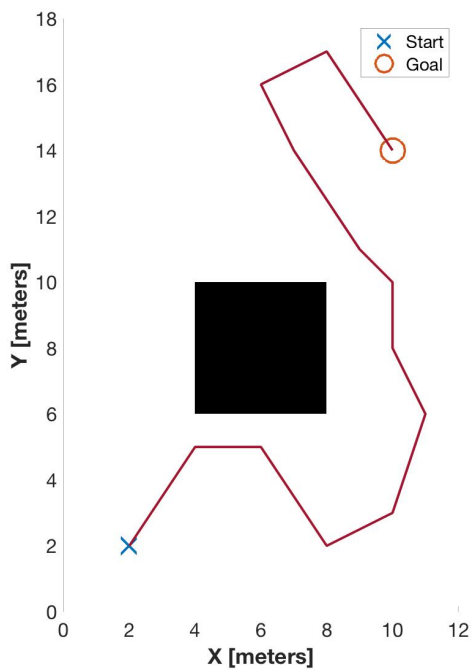


Figure 7.10. Generic Planner: An example of a poorly traveled path

Table 7.9. Performance Score Output for Case Study I

Path Planning Algorithm	*3 Membership Functions*	*5 Membership Functions*
PFM/A*	7.79	8.24
PRM	7.82	7.43
Generic Planner	3.67	3.30

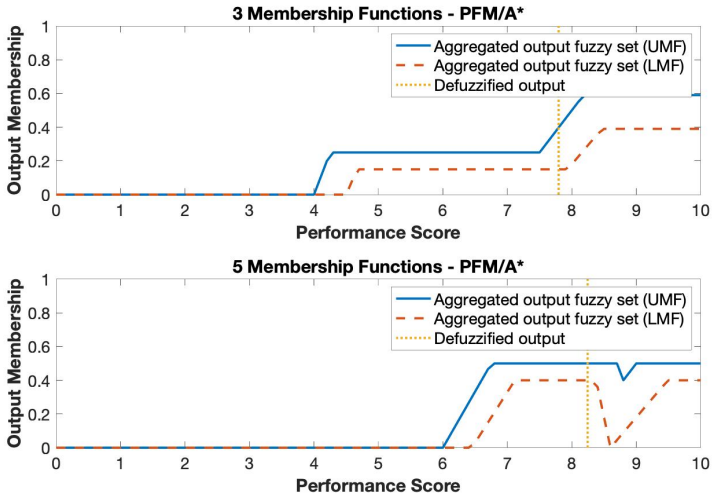


Figure 7.11. Output set for the PFM/A* algorithm

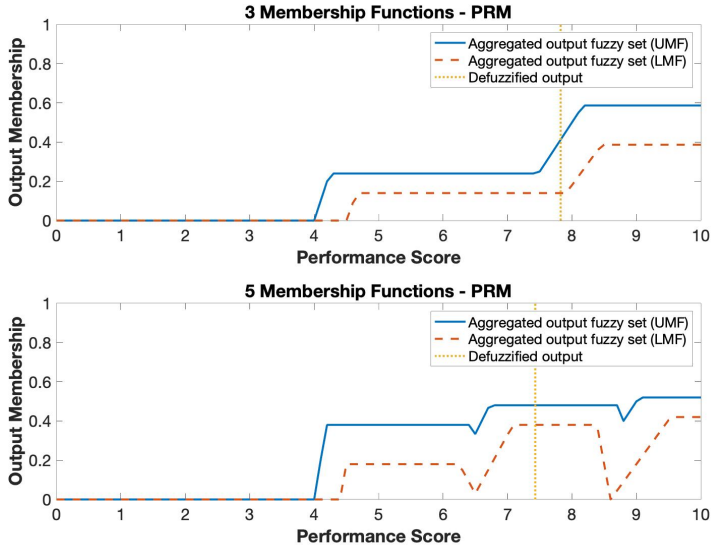


Figure 7.12. Output set for the PRM algorithm

7.3 Case Study II: Waypoint navigation with multiple obstacles

Similar to Case Study I, the second case study observes a waypoint-to-waypoint task (as shown in Figure 7.13). In this scenario, two obstacles are present and positioned to analyze the decision-making autonomy of the vehicle with regards to weighting the total distance traveled against vehicle safety (as determined by CPA). To reach the goal, the autonomy engine must decide between maneuvering between the two obstacles and increasing its safety risk or increasing the total distance traveled by taking a more conservative path to avoid the obstacles. Dependent on the intended use of the vehicle, the scoring in the IT2-FL techniques may be modeled to reflect these test goals with appropriate modifications to the rule base. CPA is chosen to be the minimum of the shortest distance between the vehicle and each obstacle. The reader should note that this approach is flexible in its application to an environment with varying numbers of obstacles and types of configurations. For example, an average value calculated from the CPA to each hazard, for instance, may be used instead.

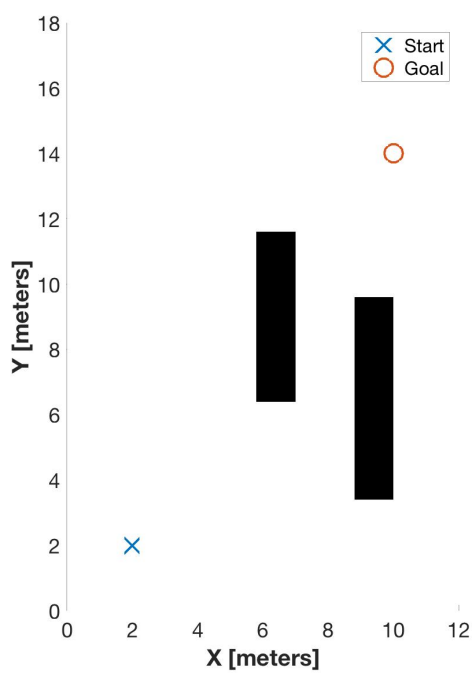


Figure 7.13. Testbed for Case Study II (obstacles represented by black boxes)

For Case Study II, the rule base (as shown in Table 7.10) is adjusted to incentivize a heavier weighting of the safety of the vehicle. Here, a higher CPA score corresponds to the linguistic term "far."

Table 7.10. Summary of rules used for Case Study II.

Linguistic Term	*Very Close*	*Close*	*Adequate*	*Far*	*Very Far*
Very Short	Poor	Fair	Good	Very Good	Good
Short	Poor	Fair	Good	Very Good	Good
Medium	Poor	Poor	Fair	Good	Fair
Long	Very Poor	Very Poor	Poor	Fair	Poor
Very Long	Very Poor	Very Poor	Poor	Fair	Poor

7.3.1 Results

The 5-MF colormap for Case Study II is provided in Figure 7.14. In accordance with the test goals for this case study, the MFs incentivize the more conservative CPA scores (roughly 3m distance) while again minimizing the total distance covered by the vehicle. The highest performance score values are designated in yellow. The resulting simulated paths are shown in Figure 7.15. The corresponding table, Table 7.11, presents the parameters values from the test run.

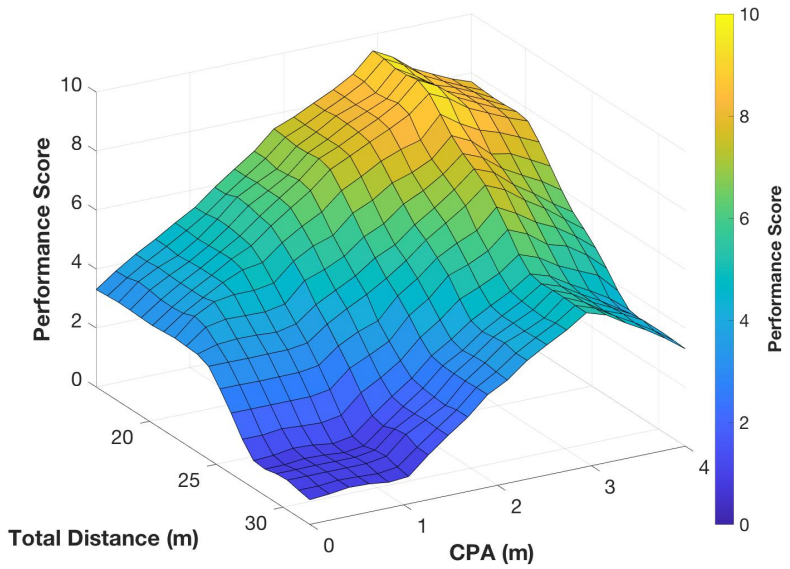


Figure 7.14. Case Study II 5-MF 3D plot relating the input variables (total distance and CPA) to corresponding output variables (performance scores)

Table 7.11. Summary of simulated performance values for Case Study II

Evaluation Parameter	*Total Distance (m)*	*CPA (m)*
PFM/A*	21.16	1.47
PRM	16.90	0.81

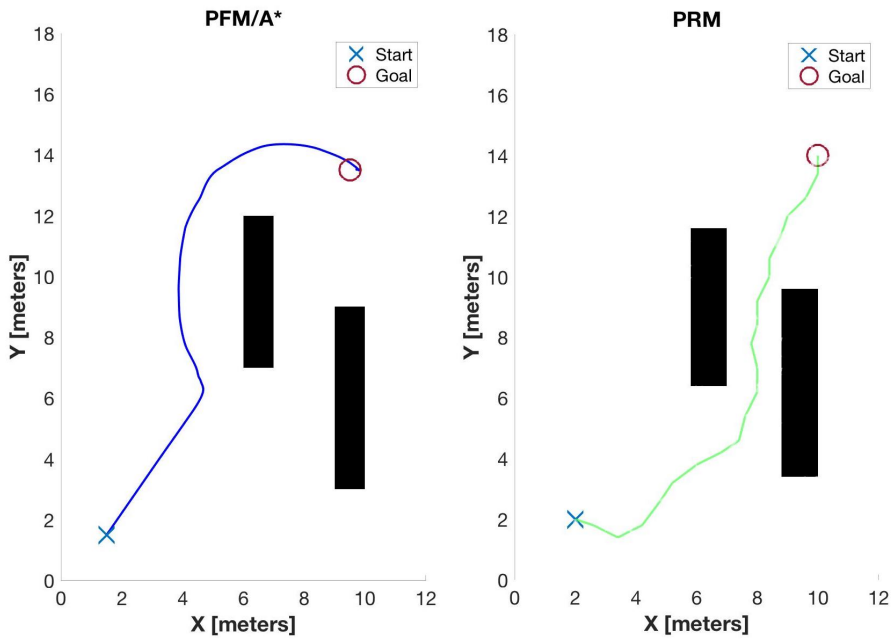


Figure 7.15. Case Study II: PFM/A* generated path (left), PRM generated path (right)

After applying the rule base and MF's using the test run values given in Table 7.11, the evaluation results are given in Table 7.12. The output fuzzy set with the defuzzified value is shown in Figure 7.16. While both planners succeed at avoiding the obstacle and arriving at the desired waypoint, the more conservative route taken by the PFM/A* vehicle (although corresponding to a longer distance traveled than that of the PRM vehicle) is given the better score which is consistent with the modeled rule base MFs and mission objectives.

Table 7.12. Performance Score Output for Case Study II

Path Planning Algorithm	Performance Score
PFM/A*	5.90
PRM	4.96

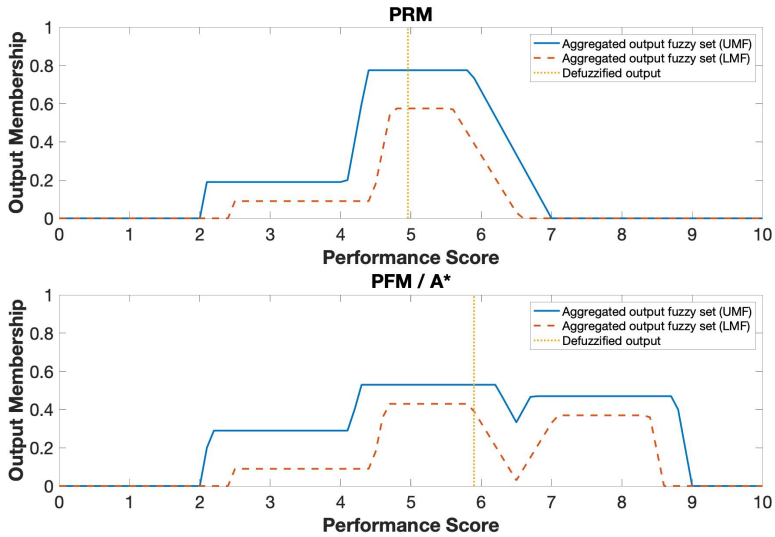


Figure 7.16. Output set for Case Study II

7.4 Case Study III: Area survey (i.e. lawnmower path survey)

Case Study III analyzes a survey task using a lawnmower pattern (as shown in Figure 7.17), common for seafloor mapping operations. Different parameters are used to reflect the desired evaluation attributes. Due to mapping operations relying on swath widths based on depth to minimize gaps in coverage, path percent error, the percent error between the actual route the vehicle takes compared to the desired lawnmower pattern, is one of the parameters measured. The second parameter is average vehicle speed. A consistent vehicle speed within the optimal range of the sonar is also important for high quality data.

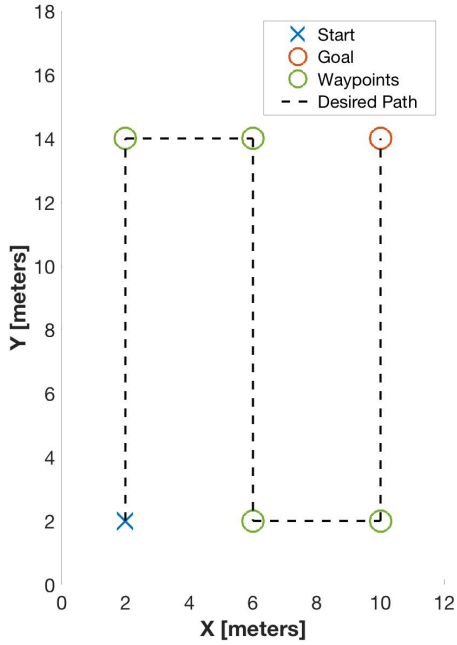


Figure 7.17. Testbed for Case Study III: A lawnmower pattern to perform seafloor mapping operations

The rule base for Case III is constructed to reflect a narrower region for acceptable performance and shown in Table 7.13. To receive a higher performance score, the vehicle's average speed should remain in the "good" range window while maintaining low error over the path taken.

Table 7.13. Summary of rules used for Case Study III

Linguistic Term	*Very Slow*	*Slow*	*Good*	*Fast*	*Very Fast*
Very Low	Poor	Fair	Very Good	Fair	Poor
Low	Poor	Fair	Very Good	Fair	Poor
Medium	Poor	Poor	Fair	Poor	Poor
High	Very Poor	Very Poor	Very Poor	Very Poor	Very Poor
Very High	Very Poor	Very Poor	Very Poor	Very Poor	Very Poor

7.4.1 Results

The resulting MF 3D plot for Case Study III depicts the correlations between the evaluation parameters and the performance score (Figure 7.18). Noted in the colormap is the smaller, more distinct area corresponding with the best performance scores (in yellow) meant to model stricter

standards for performance. With the simulated path outputs given in Figure 7.19 and the resulting parameters values (Table 7.14), the final scores are presented in Table 7.15 and the full output fuzzy set is given in Figure 7.20. Both routes successfully complete the given seafloor mapping mission. However, it is clear (as seen in Figure 24 and numerically indicated in Table 7.14) that the PFM/A* vehicle takes a smoother and more direct path, resulting in a lower path percent error.

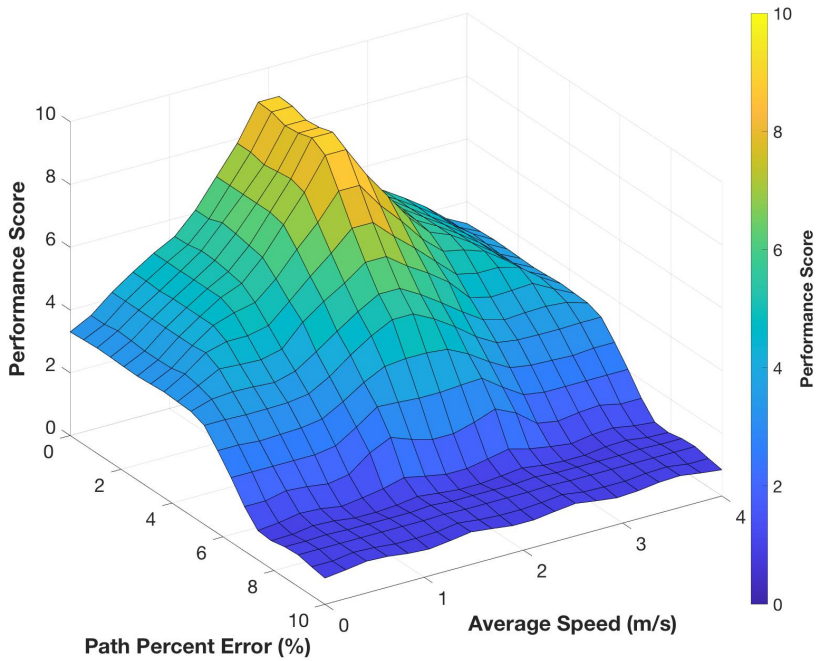


Figure 7.18. Case Study III 5-MF 3D plot relating the input variables (path percent error and average vehicle speed) to the corresponding performance scores

Table 7.14. Summary of simulated performance values for Case Study III

Evaluation Parameter	*Path Percent Error*	*Average Speed (m/s)*
PFM/A*	1.43	1.50
PRM	6.71	2.50

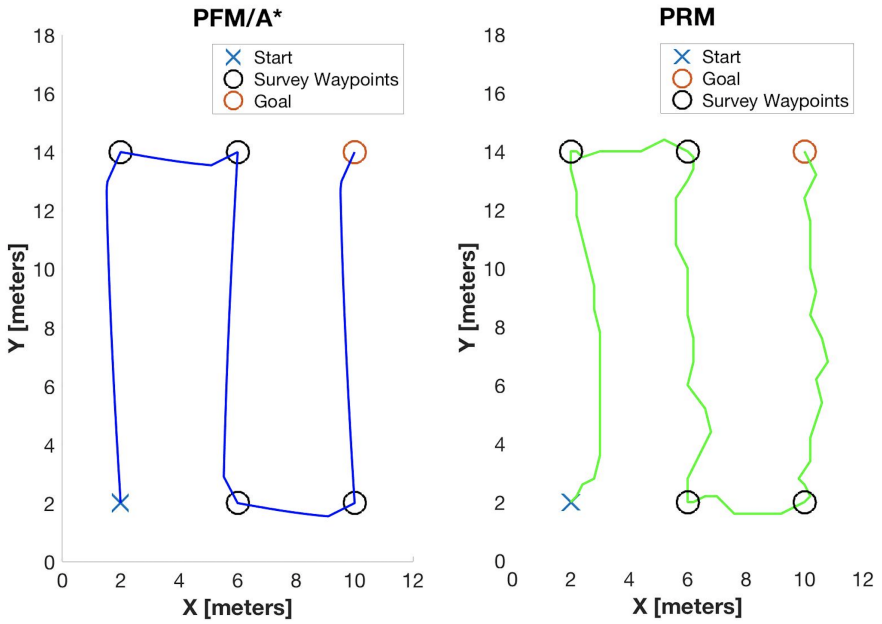


Figure 7.19. Case Study III simulated path output

Table 7.15. Performance Score Output for Case Study III

Path Planning Algorithm	Performance Score
*PFM/A**	6.94
PRM	2.78

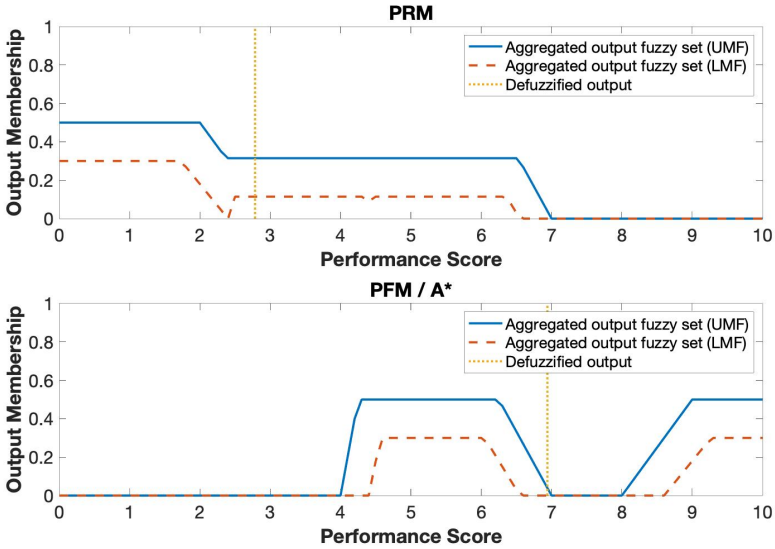


Figure 7.20. Output set for Case Study III

7.5 Discussion

This study presented the Performance Evaluation and Review Framework of Robotic Missions (PERFORM), a flexible but rigorous method to validate autonomous vehicles in testbed environments using a novel IT2-FL performance evaluation framework. The use of FL allows for test parameters that are tailored to the user's design requirements and can account for different priorities related to acceptable risks and goals of a given mission. The 3D renderings presented show parameter values for a specific mission across the space of reasonable test results in relation to a performance score output. This, in addition to the decomposition of a mission, M, into tasks, T (Equation 4.4), reduces the number of test runs necessary with the ability to analyze different scenarios taking on a range of values. Translating into both a time and cost savings, this may limit the need to perform a large number of sets of simulations which can have difficulty modeling the ocean environment accurately, testing sensor perception capabilities, and predicting other hardware issues. Results indicate that these methods aid in direct comparison of path planning algorithms as presented in the case studies with broader applications to other high-level validation test objectives such as autonomous behavior analysis.

In viewing the results for Case Study I, the 3-MF system results in a negligible difference in scores. The 5-MF system, however, allows for an easier differentiation of scores due to increased value sensitivity, producing a more substantial difference in value. One should weigh the number of included MFs based on a balance of available test design time and desired score resolution. A quick and simple study could utilize a 3-MF system, while a higher-MF system (in this case, a 5-MF system) can be designed if increased complexity is needed.

In Case Study II, analyzing behavior regarding balancing risk with efficiency, the PFM/A* path taken to avoid the obstacles receives a better score than that of the PRM path. This was predicted as the FIS was intentionally modeled to incentivize conservative decisions by the vehicle in the final performance score. During general implementation, there may be test cases where evaluation parameters will appear to have conflicting goals such as in this scenario, where one weighs safety against efficiency. The user, then, must determine the acceptable risk and hierarchy of performance priorities and reflect this in the modeling of the system.

The final scenario, Case Study III, takes a stricter approach to a vehicle receiving a desirable score. As shown in Figure 7.18, there is a steeper decline to poorer performance values. In some applications, there is a strict "cutoff" of acceptable performance. The National Oceanic and Atmospheric Administration (NOAA), for instance, has set hydrographic surveying standards that are required to be met for data to be utilized by the organization [7]. With this in mind, the evaluation model has the ability to incorporate these standards in the design of both the MFs and the rule base.

PERFORM becomes streamlined once a testbed environment is established and common measurement sensors are incorporated and calibrated. Many testing scenarios can be evaluated with the incorporation of minor changes, which would depend on given specific testing goals. This research envisions a library (expanded upon over time) of testing parameters with the x-axis MF range being the only necessary change based on the specific task under test. Once the MFs are created and test foundation constructed by a test engineer, the linguistic aspect of FL may be more approachable for various parties to understand and to set vehicle autonomy expectations in order

to show vehicle performance and "success." The linguistic base encourages a common language between engineers, operators, researchers, and program managers that all disciplines understand.

In perspective of other studies in the area of autonomous system test and evaluation, PERFORM may be useful as an extension of the JHU simulations [97] to define boundary cases. These boundary cases are a subset of all possible scenarios and could provide a feasible number of cases to undergo experimental testing.

CHAPTER 8

APPLICATION OF PERFORM TO BUILDING MISSIONS

8.1 Case Study Overview

This chapter presents three additional case studies that build upon the task-based examples provided in the previous chapter. These case studies take the decomposed tasks and reconstruct them into full missions to demonstrate the ability of PERFORM to serve as the framework for the metric function, β, that provides a mapping from the scenario space to the capability space as introduced in Section 4.5.1. The outline below provides the title of each case study and the corresponding goal of each validation test for PERFORM (from a test engineer's point-of-view and without loss of generality).

- **Case Study IV: Combining Canonical Tasks (Extension of Case Studies I and III)**
  - Compare scores from testing individual tasks with a multi-task mission while varying scenario parameters.
  - Analyze viability of decomposition methods and the ability to build missions from individual tasks.
- **Case Study V: Testing Endurance with Multiple Surveys**
  - Evaluate a vehicle's mission planning capabilities at the highest level by testing operation efficiency.
  - Complete as many surveys as possible while reaching the recovery point with greater than 10% battery charge.
- **Case Study VI: Full Mission with Multiple Platforms**

- Analyze a scenario with multiple tasks to gain insight on feasibility of the test methods as the number of inputs increases.
- Construct a study with more than one platform.

8.2 Case Study IV: Combining Canonical Tasks (Extension of Case Studies I and III)

Using the results from Case Studies I and III, Case Study IV combines these tasks together (i.e., waypoint navigation with obstacle and a survey). The goal is to observe if there is a substantially different score between that of summing the results of the testing from each individual task and that of testing the mission as a whole. Additionally, this will also investigate if changing the scenario parameters significantly changes the score. The following scenario parameters are changed for the combined mission: the number of obstacles, the location of the obstacles, and the survey size and spacing.

The same vehicle and simulation setup is assumed as that from Chapter 7. The simulated test location is a larger area (80m x 120m) to allow for testing with the increase of the scenario parameters. The configuration for the mission is given in Figure 8.1. Three tasks are required: the transit to the survey location (Task 1), the surveying of the area (Task 2), and the transit back to the recovery point (Task 3). PFM/A* and PRM algorithms are again used to provide a source of comparison and so that scores can be compared with results from that of Case Studies I and III. For PFM/A*, the same gain values are used as that in Case Study I (Table 7.1). The values for PRM are increased, however, due to the size of the area. Here, the number of nodes is set to 8000, the maximum neighbor distance is set to 10m, and the maximum number of neighbors remains at 3.

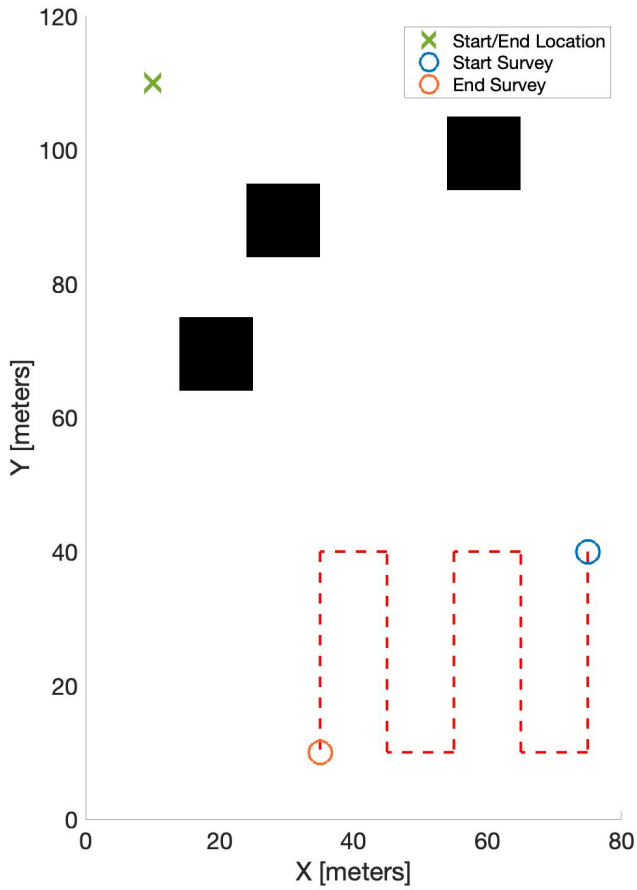


Figure 8.1. Case Study IV simulated environment configuration

The total distance input parameter is re-scaled to account for the larger area. As in Chapter 7, the domain for the MFs will range from the shortest possible distance to twice that amount (as shown in Figures 8.2 and 8.3). The same rule base is used as that in Case Study I, and trajectories around obstacles that balance safety and efficiency are more highly prioritized (Table 7.7, 5-MF version). For the mapping portion, path percent error and average speed are again used. As such, the same MFs and rules apply (Figures 7.3 & Table 7.13). The average speed input is arbitrarily given the same values for each algorithm as in Chapter 7. The output performance score parameter is also the same as that defined in Chapter 7 (Figure 7.6, 5-MF version). The 3-D plot for Task 1 is given in Figure 8.4 to show the similarity with Case Study I.

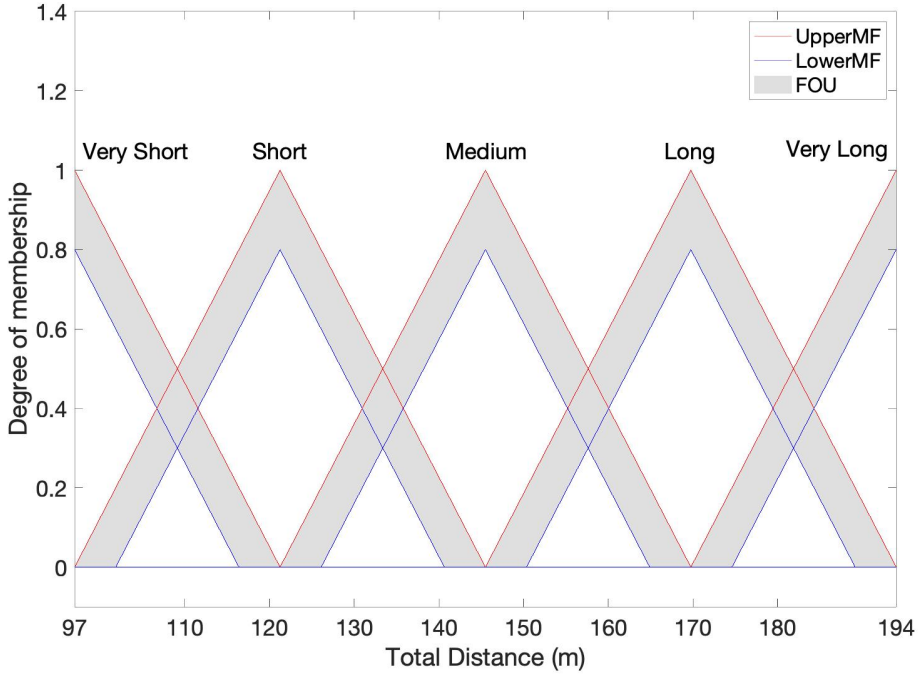


Figure 8.2. Transit 1 MFs

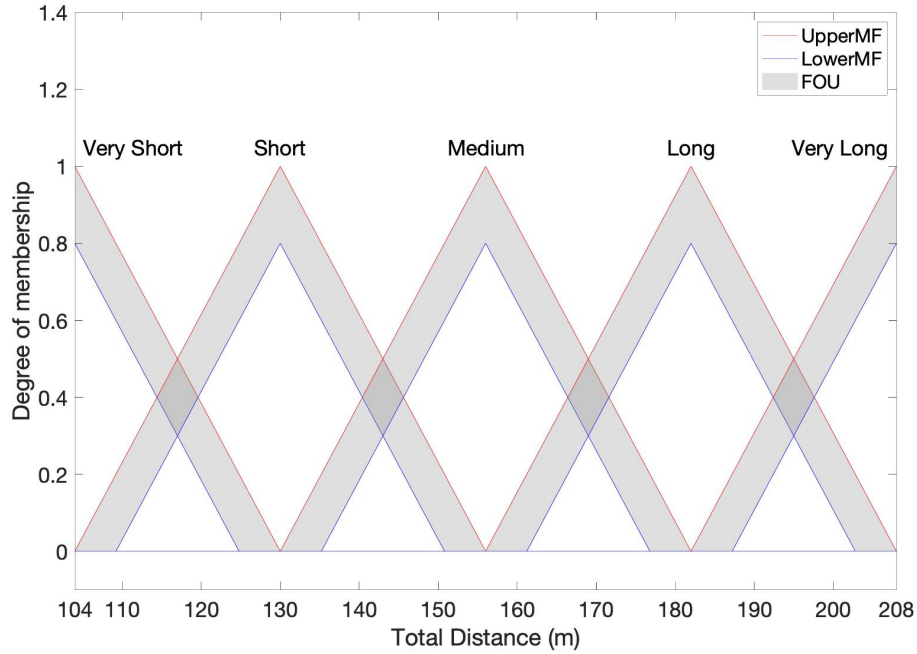


Figure 8.3. Transit 2 MFs

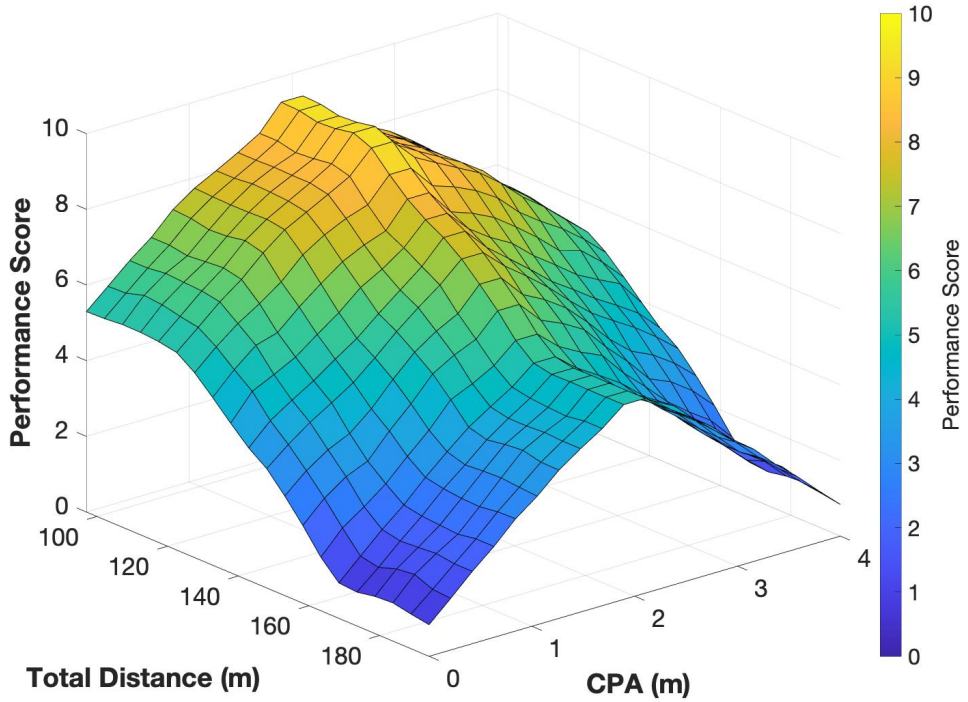


Figure 8.4. 3-D plot of Task 1 input parameters vs. the output performance score, similar to the corresponding plot for Case Study I (Figure 7.8) with only a change in axis values

8.2.1 Results

The simulated results of the full mission are provided in Figure 8.5. The simulated travel path is clockwise. The calculated input parameter values for each task are given in Tables 8.1, 8.2, and 8.3.

Table 8.1. Summary of simulated performance values for Task 1, Case Study IV

Evaluation Parameter	*Total Distance (m)*	*CPA (m)*
PFM/A*	98.95	1.05
PRM	101.09	2.06

Table 8.2. Summary of simulated performance values for Task 2, Case Study IV

Evaluation Parameter	*Path Percent Error*	*Average Speed (m/s)*
PFM/A*	0.67	1.50
PRM	8.06	2.50

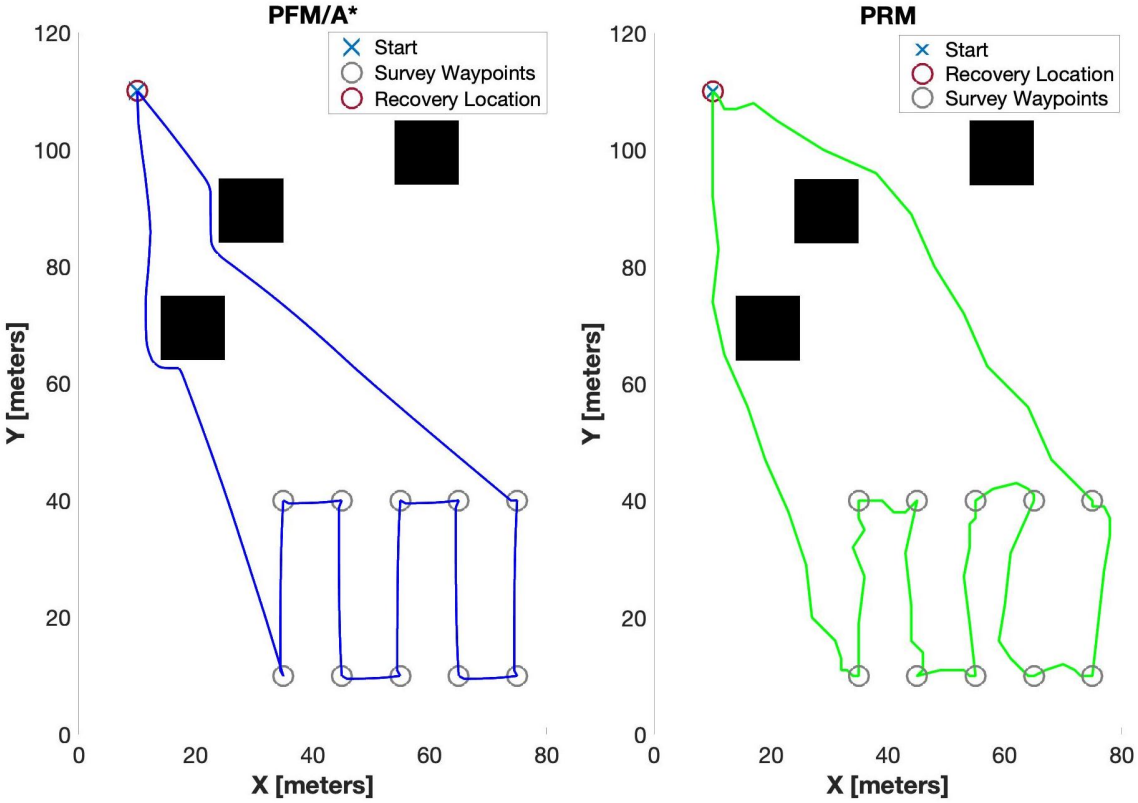


Figure 8.5. Simulated mission results for each algorithm.

Table 8.3. Summary of simulated performance values for Task 3, Case Study IV

Evaluation Parameter	*Total Distance (m)*	*CPA (m)*
PFM/A*	106.96	1.06
PRM	107.29	1.49

Table 8.4. Performance Score Output for Case Study IV compared with the analogous tasks in Chapter 7. Task 1, 3, and Case Study 1 are the waypoint-to-waypoint tasks (gray columns) while Task 2 and Case Study 3 refer to the survey tasks

Path Planning Algorithm	Task 1	Task 2	Task 3	Case Study 1	Case Study 3
PFM/A*	7.72	6.94	7.72	8.24	6.94
PRM	9.28	1.14	8.23	7.43	2.78

To compute the total mission score, Equation 6.9 is used. For this example, equal weights (0.33) are used for each task. Equations 8.2 and 8.3 provide the final score for the PFM/A* and PRM algorithms in this case study, respectively. These mission performance scores are compared with calculating the mission performance score from the individually tested tasks in Chapter 7 in Table 8.5 using the same equations. (It is noted that in compiling the scores from the individual tasks, the score from Case Study I is used twice to account for the two waypoint-to-waypoint tasks.)

$$\mathbb{P}_{total} = w_1\mathbb{P}_{Task1} + w_2\mathbb{P}_{Task2} + w_3\mathbb{P}_{Task3} \tag{8.1}$$

$$\mathbb{P}_{PFM/A*} = 0.33(7.72) + 0.33(6.94) + 0.33(7.72) = 7.38 \tag{8.2}$$

$$\mathbb{P}_{PRM} = 0.33(9.28) + 0.33(1.14) + 0.33(8.23) = 6.15 \tag{8.3}$$

Table 8.5. Comparison of the final mission score from testing individual tasks (Chapter 7) and from testing the entire mission (Chapter 8).

Path Planning Algorithm	*Chapter 7 Combined Score*	*Chapter 8 Mission Score*	*Difference (%)*
PFM/A*	7.73	7.38	4.63%
PRM	5.82	6.15	5.51%

Results support the hypothesis that for testing purposes, missions can be decomposed into individual tasks. The task-by-task score of the mission for PFM/A* is very similar to that in Chapter 7. This is partly due to the nature of the algorithm and how it calculates the path consistently each iteration. It is noted that some natural variations will be seen with full vehicle dynamics and live

field testing. While still reasonably similar, the PRM algorithm demonstrates more variability due to the nature of how it explores a given search space. The randomness inherent with the algorithm in addition to the intentional lack of vehicle control parameter-tuning allows the PERFORM process to show its robustness to these variations. The other critical result supports the fact that one does not need to simulate all possible scenarios and combinations of test scenarios. On the contrary, changing the scenario parameters does not significantly change the performance scores, and similar tasks also produced similar scores.

8.2.2 Discussion

Testing with PERFORM, as shown in Case Study IV, reduces redundancy by eliminating potential overlaps of tasks. Here, for example, Task 1 & 3 are the same type of task. As such, the tested mission could have removed Task 3 from the start, since it was shown that similar scores were produced. One may question if adding an obstacle to the survey task would require changes to the test. It can be argued that this addition would merely result in testing redundancy of obstacle avoidance in the waypoint tasks (Task 1 & 3). If the vehicle already has proven obstacle avoidance capability in the other tasks, this additional task would not necessarily provide any new information to the survey task, especially since the survey task is already essentially waypoint-to-waypoint navigation.

Another question that may be posed in regards to decomposing missions is whether or not one can ignore the transitions between consecutive tasks. It is argued that since these transitions are essentially software based and are not actually a physical parameter, they do not have a direct effect on physical performance. These transitions are also verifiable via simulations and hardware-in-the-loop testing due to the reducibility of the task into two components: the acknowledgement that a task has been completed and the ability to transition to the next task/behavior.

The MFs are easily modified for slightly varying mission and scenario parameters (Figures 8.2 & 8.3), if modification is even necessary. As seen from the aforementioned simulations, there were no changes in the CPA and path percent error parameters. The ability to find boundary cases

(as in the JHU RAPT framework) in conjunction with the PERFORM framework introduced in this work is designed to serve as a powerful combination of determining the capability limits of changing the scenario parameters as well as identifying the mission-critical scenarios for which field tests must be performed. Live testing, in turn, contributes additional information from which a higher-integrity simulation environment may be established.

The sensitivity of the score can also be changed through the modeling component of PERFORM. Visually, the 3-D surface plots that have less steep gradients will be more robust in the output to fluctuating inputs as they are less sensitive to changing inputs. As shown in Case Study I with the comparison between 3-MF and 5-MF systems, the number of MFs can be changed to reflect the desired resolution. For instance, we see Task 2 and Case Study III produced the same individual task performance scores (6.94) for the PFM/A* algorithm even though Task 2 had a path percent error of 0.67 while Case Study III has a value of 1.43. In Figure 7.18, those input scores are located at the yellow ridge of the plot and at the same z-axis coordinate. Since the average speed value is the same for both cases, the same score would be produced for a path percent error value ranging from 0% to approximately 2% as in this case. This modeling capability could also prove useful for cost benefit analysis. If an acceptable range is known for a scoring parameter (e.g., maintaining a CPA distance no closer than 5m) it could provide a comparable score where perhaps a lower cost platform can perform adequately for the defined standards of the mission. Independent from the platform itself, the scoring mechanism produces similar scores regardless of the platform's hardware and specifications.

8.3 Case Study V: Testing Endurance with Multiple Surveys

This case study examines the PERFORM process for evaluating a vehicle's decision-making process towards efficient mission planning where the goal is to maximize productivity within a time and battery level constraint. The time is bounded by the vehicle's endurance (i.e., battery capacity). Mission criteria involves conducting multiple surveys while monitoring remaining power, using as much battery charge as possible, and still achieving safe recovery. While the mission

can be further decomposed and evaluated by (sub)task, this case study remains at a high-level of evaluation focused on mission planning. This also demonstrates the flexibility of the process by decomposing the mission's description and potential scenarios to the largest set of individual tasks that correspond to different behaviors and/or modules of the autonomy software at the desired level of autonomy that is being tested. For this case study, a REMUS 100 AUV with a generic sonar payload is considered. (A specific vehicle is chosen so as to use given manufacturing specifications for designing the test parameters.) The simulated test range is chosen to be at the Naval Undersea Warfare Center - Division Keyport (NUWC-Keyport). As shown in Figure 8.6, the north operation boundary of the testing area is 600m, the east is 3500m, the south is 1960m, and the west is 3000m.

The REMUS is provided with three survey regions and the corresponding sizes and locations (also shown in Figure 8.6). Survey Region 1 and 3 have identical dimensions (approximately 525m x 850m), and Survey Region 3 is approximately 650m x 950m. The battery is assumed to be fully charged at the start of mission, the approximate endurance time of the vehicle is known (approximately 24 hours for a REMUS 100), and the desired lawnmower path spacing is determined a priori to be 20m. Tasks must be completed in sequential order for this particular test, and the vehicle is to return to the recovery point before the battery level reaches 10% capacity. Here, the REMUS is permitted to stop a survey at any time as appropriate in order to return to the recovery point safely. Failure of the vehicle to return to the recovery point or to reach the recovery point before the 10% battery threshold is reached is considered a mission failure.

The two input parameters, battery level and survey completion percentage, are considered independently. Therefore, the REMUS not completing the entire survey does not necessarily indicate mission failure, so long as the vehicle returns safely. In fact, the test setup is intended to push the boundary between completion of the surveys and the available endurance of the vehicle to gauge the safe decision-making ability of the autonomy. Survey Regions 1 and 3 require 26 tracklines and Survey Region 2 requires 33 tracklines with 20m spacing. Assuming a desired average speed of 1 m/s, 19.3 hours is the required time to complete the surveys (not including the turnaround distance once a trackline is complete), in addition to the extra hour to complete the transits.

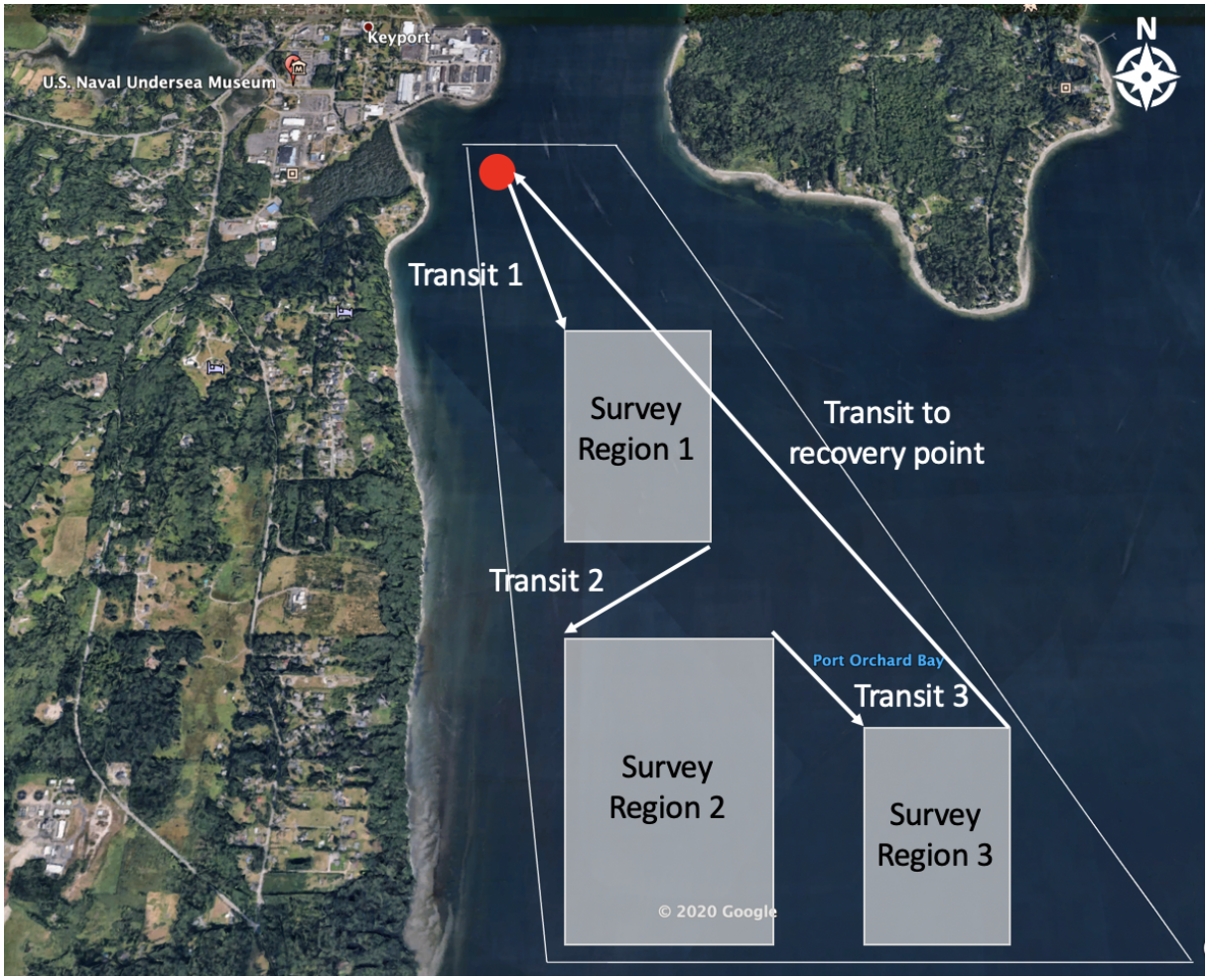


Figure 8.6. NUWC-Keyport test range: the operation area (thin white line), the launch/recovery point (solid red circle), and the transit lines (solid white arrows).

The scenario parameters are as follows:

- Launch/Recover Point (x,y)
- Survey Region 1 (Box Size, Location) (x,y), (x,y)
- Survey Region 2 (Box Size, Location) (x,y), (x,y)
- Survey Region 3 (Box Size, Location) (x,y), (x,y)
- Starting Battery Charge

- Environmental Factors (Currents, Time of Day, etc.)

and the set of possible tasks, T, are defined as:

$$\begin{aligned} T = \{ & \text{ Transit to Region 1}, \\ & \text{Survey Region 1}, \\ & \text{Transit to Region 2}, \\ & \text{Survey Region 2}, \\ & \text{Transit to Region 3}, \\ & \text{Survey Region 3}, \\ & \text{Transit to Recovery Point}\} \end{aligned} \tag{8.4}$$

where the set of possible events, E, are:

$$\begin{aligned} E = \{ & \text{ Launch}, \\ & \text{Arrive Survey Region 1}, \\ & \text{Finish Survey}, \\ & \text{Arrive Survey Region 2}, \\ & \text{Arrive Survey Region 3}, \\ & \text{Detect Battery Charge Warning}, \\ & \text{System Fault}, \\ & \text{Arrive Recovery Point}\} \end{aligned} \tag{8.5}$$

The structure of this mission is such that it is to observe the performance from the perspective of Level 0, as shown in Figure 8.7. Here, the structure of this mission uses one *Task* (which, in this case, is the overall mission objective) without loss of generality. However, the structure may

be further decomposed into multiple *(Sub)tasks* (Level 1) as necessary. Since the purpose of the simulation is to observe the the overall endurance of the vehicle, the implementation of the fuzzy process to evaluate the REMUS autonomy performance occurs at Level 0.

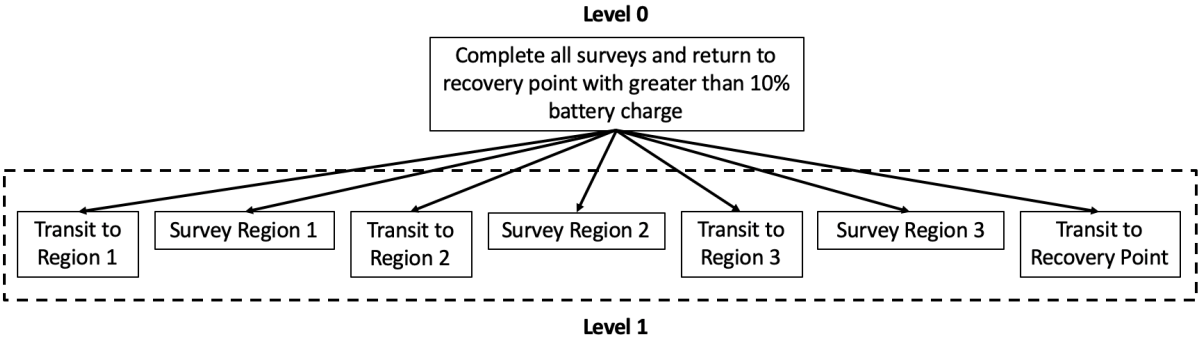


Figure 8.7. Task decomposition structure for Case Study V

8.3.1 Metric Design: Case Study V

As stated previously, two inputs to PERFORM are considered here: battery charge percentage and the percentage of surveys completed. The percentage of surveys completed is determined by the number of completed tracklines with respect to the total number of possible tracklines for all three surveys, which is 59 in this case. The transits are not directly evaluated via modeling a specific performance parameter, but do have some influence in the amount of battery charge left.

To create the MFs, the FOU size for this example is dictated by linguistic vagueness, as sensor uncertainty does not have a direct influence on the performance score for this case study. The number of completed track lines may be obtained from the onboard data logging system after the test run. Updates may also be relayed to the shore station during surfacing events. The MFs for the performance score output used for this case study are the same as that in Chapter 7. The MFs are provided again here in Figure 8.8 for ease of reference.

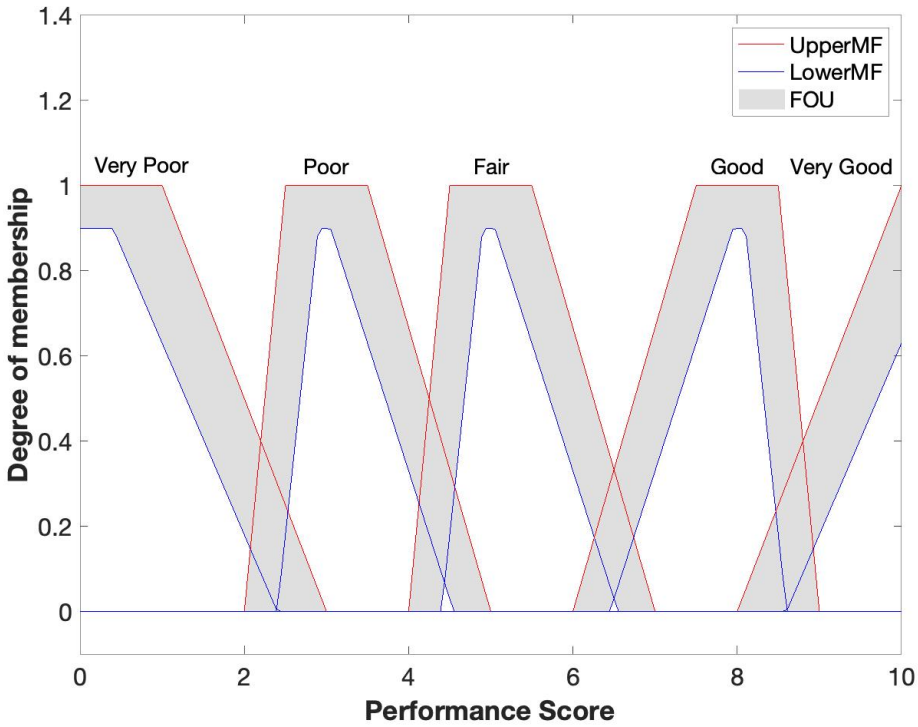


Figure 8.8. Performance Score Membership Functions

The survey completion input parameter is shown in Figure 8.9 and is modeled with equally spaced fuzzy sets except for that of "Very Low." A trapezoid MF is used so that a higher degree of membership can be extended to 20% of survey completion. Anything below this threshold of completion is considered unacceptable, but this MF can be extended out further depending on the level of performance needed. This narrows the available range for modeling the other four MFs and, thus, results in a steeper gradient for the performance score mapping within that region.

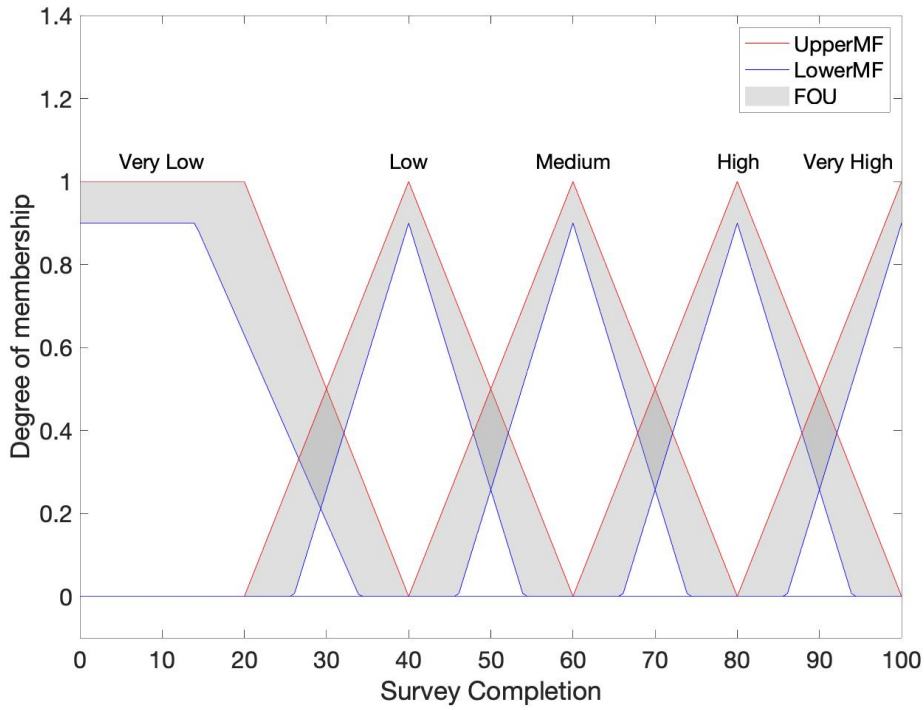


Figure 8.9. Membership functions for the survey completion parameter

For the remaining battery charge MFs, one may notice the binary nature of the function associated with "Very Low" (Figure 8.10). In the initial mission description, the objective is to return to the recovery point before the battery charge reaches 10%. This function describes this threshold and has no uncertainty associated with it due to the strict threshold value. As such, this is a T1 fuzzy set. Since all T2 fuzzy sets reduce to a T1 fuzzy set as uncertainty decreases to zero (Chapter 5), using a combination of T1 and T2 sets is valid and does not change the calculations. The fuzzy set for "Low" has a narrower base with which to capture the expected final battery level to complete all of the surveys and therefore reserves the best performance scores for only a small range of values.

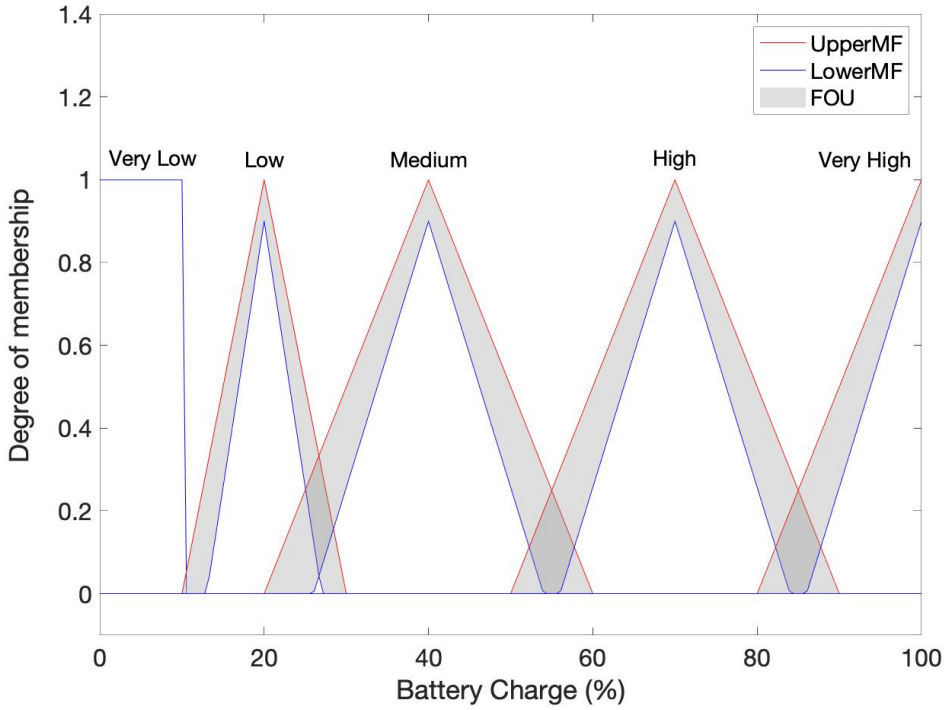


Figure 8.10. Membership functions for the battery charge parameter

The URC is used in this case study with the rule base for each input given in Table 8.6. The rule matrix may seem counterintuitive for the battery charge level. However, if a vehicle terminates a mission with a high to very high battery level, this infers that the mission ended prematurely due to a system fault or other such issue. Again, one should note that the mission objective is to use as much battery as safely possible to complete as much of the surveys as possible but still be able to return "home". With knowledge of the expected battery performance, completion of all surveys should fall between the range of $10 - 30\%$. The FOU bandwidth for each input increases slightly as the function moves away from the center point of the MF in order to model the increasing uncertainty between linguistic terms.

Battery Charge

VL	L	M	H	VH
VP	VG	G	P	VP

Survey Completed Percentage

VL	L	M	H	VH
VP	P	F	G	VG

Table 8.6. Rule matrix for the two input parameters: Battery Charge (left) and Survey Completed Percentage (right). VL=very low; L=low; M=medium; H=high; VH=very high; VP=very poor; P=poor; F=fair; G=good; VG=very good

The 3-D plot in Figure 8.11 shows the interaction between the input parameters and output performance score. As intended, there is a steep gradient descent at the 10% battery level to provide a definitive boundary between safe and unsafe operation. The highest scores correspond to a battery level of $10 - 30\%$ with all surveys completed. Lower scores are also associated with higher battery charge but with a low percentage of surveys completed.

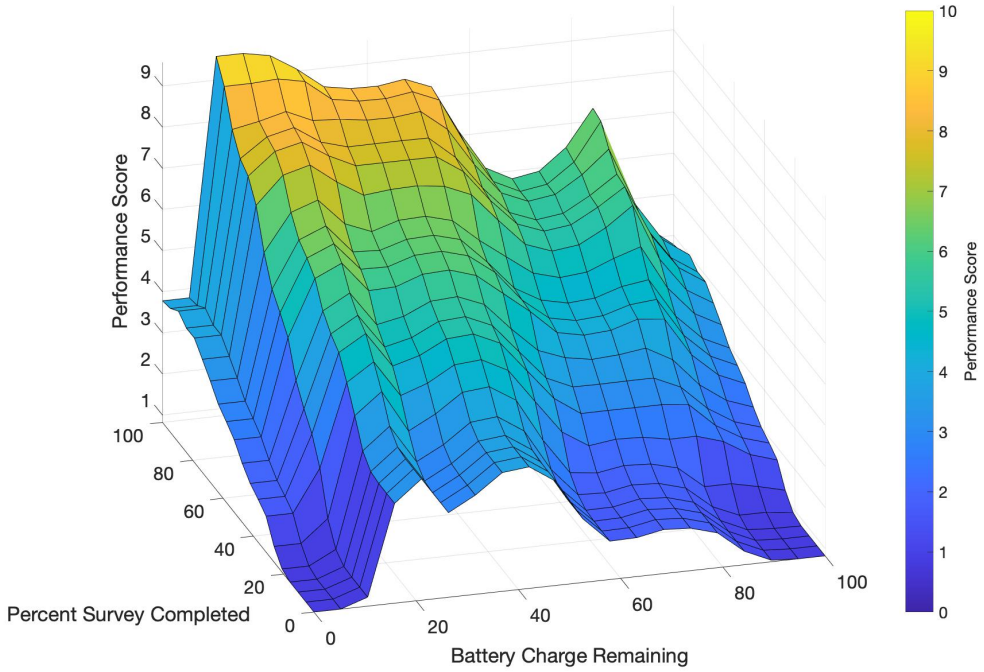


Figure 8.11. 3D plot for the input parameters of Case Study V

8.3.2 Discussion: Case Study V

Through this case study, PERFORM demonstrates its ability to take into account endurance-based missions scenarios and top-level mission planning behaviors. To increase the level of autonomous capability at which the autonomy is tested, this scenario could, instead, provide survey locations without a set of tracklines. In this case, the vehicle would need to make decisions on the most efficient way to complete the survey depending on tide, currents, and any other relevant weather conditions, in addition to appropriate trackline spacing. Another type of task that would require a high-level evaluation is task scheduling, say, if the vehicle were also tasked with choosing the order of the surveys.

Regarding the structure of the mission, if further mission decomposition is desired, say, due to the test engineer's concerns about the autonomy's ability to carry out a survey, the fuzzy process could instead be implemented at Level 1 of Figure 8.7. To avoid task redundancy at that level (with 3 existing surveys), one could isolate a single survey to not only reduce redundancy but to also focus on that specific capability. The mission could also be decomposed into an additional level to, for example, analyze the vehicle control system or perhaps the motion planner and its ability to generate feasible turns to move to the next trackline. Also, since 24 hours is a lengthy duration for testing, to shorten the duration, the vehicle could start with a half-full battery charge and a scaled-down survey size.

A mission with temporal constraints would be approached in the same way, where hard temporal deadlines can be modeled using a T1-MF (with binary functions as necessary), as was exemplified by "Very Low" in Figure 8.10. Softer temporal deadlines (i.e. having a range with increasing penalty) can easily be modeled using the T2-MFs. Each task with a temporal constraint would have a time-based input parameter.

8.4 Case Study VI: Full Mission with Multiple Platforms

To show the PERFORM process in another context and to demonstrate scalability, Case Study VI compiles multiple tasks into a full multi-platform mission. While the complexity of a mis-

sion increases as a whole, the methodology still allows for the analytical decoupling of tasks. As the size of the multi-platform system increases, it is helpful to organize the subsystems using a systems engineering approach. Most missions have overlap (e.g., several waypoint-to-waypoint tasks), leaving opportunity for the reduction of required test scenarios. The tested mission and the actual mission do not need to be identical. This aspect is critical for practical implementation, as mission time scales are sometimes measured in days, even weeks, usually making full-scale tests impractical if not impossible. Testing individual tasks (i.e., Case Studies I-III) gives insight to specific critical capabilities of the vehicle which can later be combined with higher level testing (i.e., Case Study V).

Figure 8.12 shows an example of a mission categorized according to time-dependent mission evaluation levels. At the mission planning level, the mission planner spans the entire mission. At the task level, on the other hand, each task spans a subset of the mission length, and corresponding subtasks occur within the timeframe of their associated task. Subtasks may be performed concurrently. For instance, the path planner and heading control are both needed simultaneously. The monitoring of vehicle status (e.g., energy, conditions, system health, etc.) also spans the entire mission and would also be placed at the highest level of the evaluation tree.

Logic dictates that, in the evaluation of the mission, the decomposition categorizes aspects of the system that span the length of the mission as "top level," as exemplified in Figure 8.13, where mission objectives refer to the set of tasks that are necessary for mission completion (i.e., transit, obstacle avoidance, surveying, etc.). The testing performed in Case Study V would fall under the "Mission" planning category.

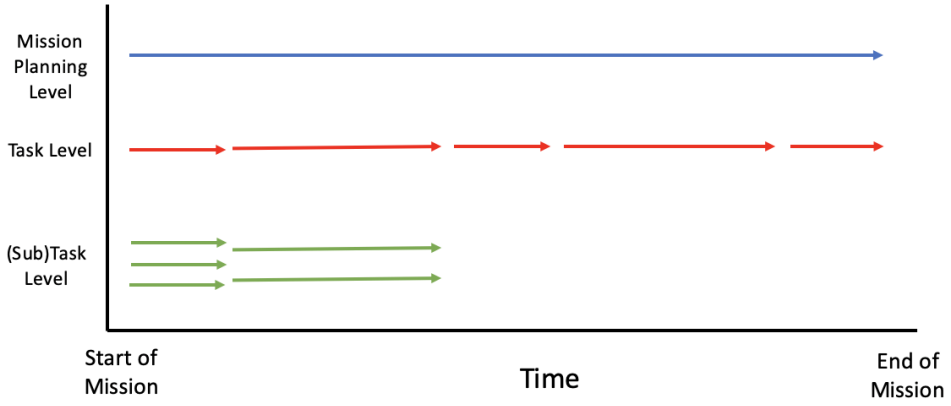


Figure 8.12. Visual representation of the temporal constraints for various mission levels

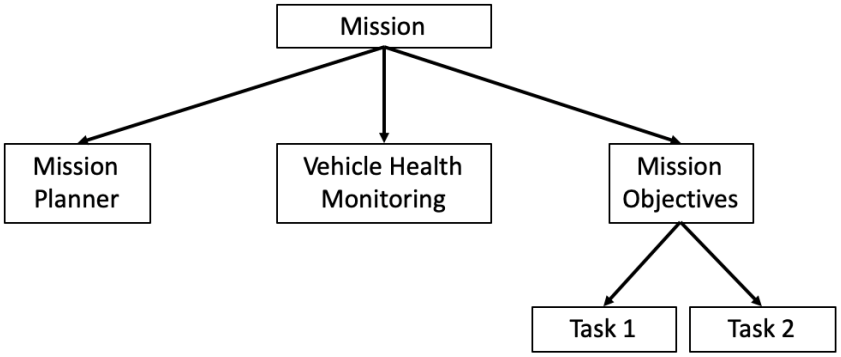


Figure 8.13. Recommended structure for computing a single-mission performance score with activities spanning the entire mission being placed at the top level

In Case Study VI, a multi-platform mission is investigated to validate the scalability and flexibility of the PERFORM methodology. An ASV is tasked with finding an object of interest and then deploying a small-scale AUV to visually document the object. While the sum of the capabilities needed to complete this mission is complex, testing with respect to specific decomposed components simplifies and reduces the necessary field testing. From Case Study IV, it is shown that decomposition is a valid approach. The decomposition of tasks for each vehicle in this case study is given in Figure 8.14.

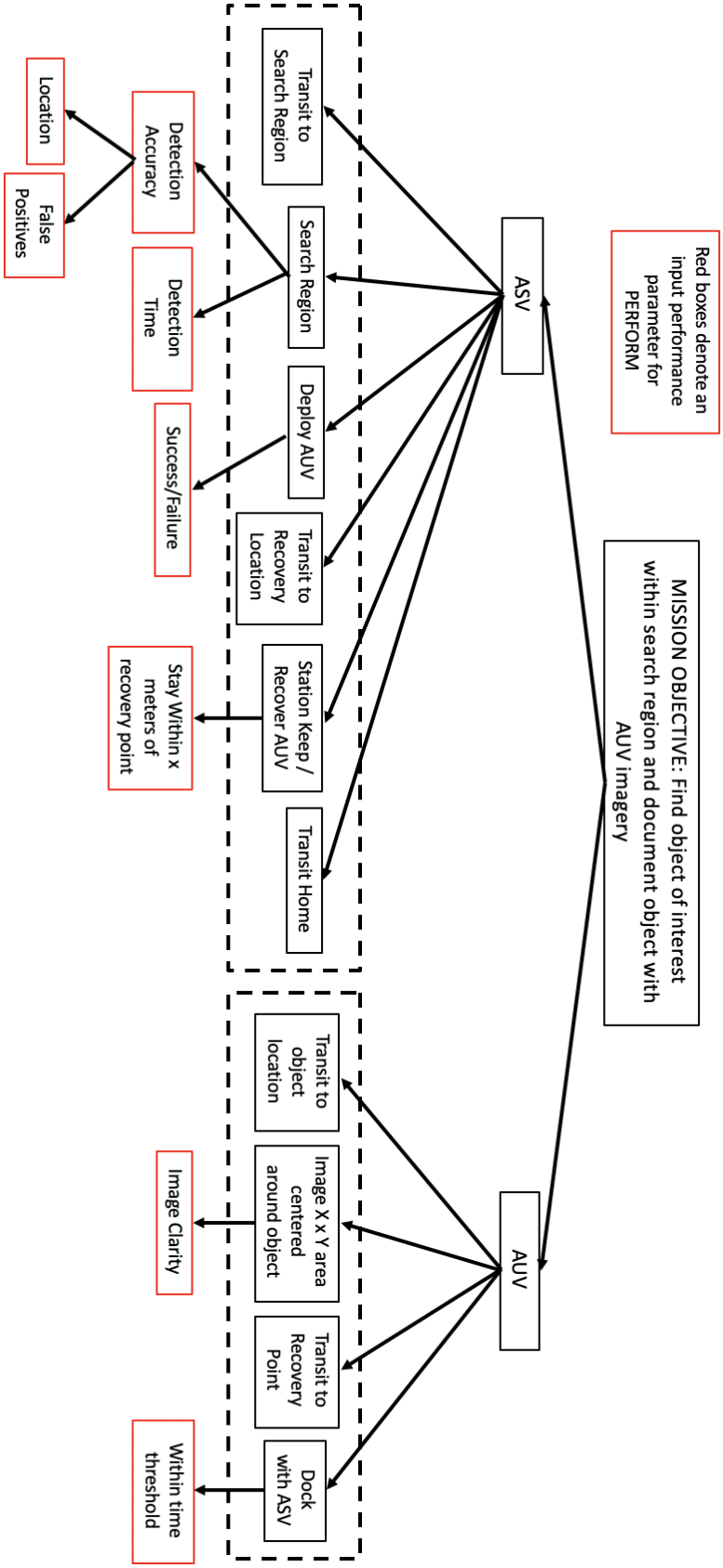


Figure 8.14. Case Study VI: Overview of task decomposition

The mission is not simulated and arbitrary values are given for each task since the focus of this case study is on the PERFORM calculation step and the structure of the mission decomposition. The scenario parameters are as follows:

- Launch/Recover Point (*x,y*)
- Survey Region (Box Size and Location) (*x,y*), (*x,y*)
- Obstacle(s) (*x, y, vx, vy*)
- Object of Interest Location (*x, y*)
- Starting Battery Charge
- Environmental Factors (Currents, Time of Day, etc.)

The set of possible tasks for the ASV and the AUV are denoted as T_{ASV} and T_{AUV}, respectively, and are defined as:

$$\begin{aligned} T_{ASV} = \{ & \text{Transit to Search Region,} \\ & \text{Search Region,} \\ & \text{Deploy AUV,} \\ & \text{Transit to Recovery Location,} \\ & \text{Station Keep / Recover AUV,} \\ & \text{Transit Home}\} \end{aligned} \tag{8.6}$$

$$\begin{aligned} T_{AUV} = \{ & \text{Transit to Object Location,} \\ & \text{Image Area,} \\ & \text{Transit to Recovery Point,} \\ & \text{Dock with ASV}\} \end{aligned} \tag{8.7}$$

where the set of possible events, E, are:

$$
\begin{aligned}
E = \{ & \text{ Launch,} \\
& \text{Arrive at Survey Region,} \\
& \text{Obstacle on Surface,} \\
& \text{Detect Object of Interest,} \\
& \text{Detect end of Mission Condition,} \\
& \text{Detect Battery Charge Warning,} \\
& \text{Arrive Recovery Point,} \\
& \text{AUV Launched,} \\
& \text{AUV Recovered}\}
\end{aligned}
\tag{8.8}
$$

For the purposes of the example, it is assumed that the test engineer has confidence in the ASV's ability to perform transit tasks (i.e., waypoint-to-waypoint navigation) proven through previous testing. For this particular example study, one may consider the case for these offline tests produced, say, an average PERFORM score of 8.5 based on the output MF's defined in Figure 8.8. This reduces the ASV test design for this mission to the following tasks "Search Region," "Deploy AUV," and "Station Keep / Recover AUV." The AUV is also assumed to have established waypoint-to-waypoint capabilities. For this study, the AUV has earned a transit-task PERFORM score of 8.2. It is noted here that testing for area imaging and AUV-ASV docking are still needed.

8.4.1 Metric Design: Case Study VI

8.4.1.1 ASV

Two of the tasks under test for the ASV use the IT2-FL approach: "Search Region" and "Station Keep / Recover AUV." "Deploy AUV," however, is considered a binary parameter. (As an AUV cannot be partially deployed successfully, it is either task success or task failure.) With

the performance score range set between 0-10, a zero is assigned to failure and a score of ten is assigned for success.

The scoring of the "Search Region" task is compiled from two input performance parameters: detection accuracy and detection time. A low detection time is desirable, although not at the expense of accuracy. Detection accuracy is further decomposed into two scoring parameters, location and false positives. Location refers to the accuracy with which the platform determines the position of the object and false positives refer to the ranking of potential objects in terms of detection probability. Data is extracted for a sorted list compiled by the autonomy platform that contains the detected location, the probability of detection, and a timestamp. An example of how the detection location input parameter may be designed is shown in Figure 8.15. Detection time can also be modeled in a similar fashion, where time is the abscissa unit. A few approaches may be appropriate for the false positive metric. One strategy is to design a score based on where the actual object is ranked in a sorted list by detection probability calculated by the vehicle's autonomy. If the correct object is first on the list, that would have the highest score, with the score decreasing for each subsequent spot on the list.

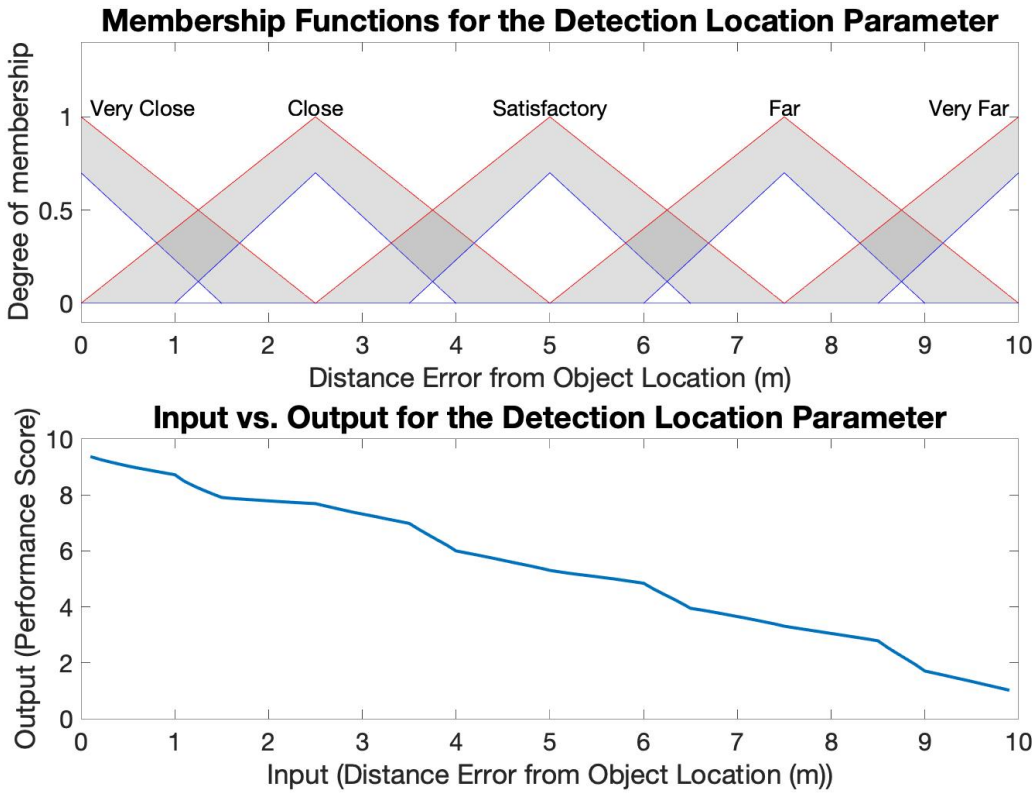


Figure 8.15. Example MF for the detection location parameter

For the "Station Keep / Recover AUV" task, the corresponding input performance parameter is maintaining a location within an acceptable radius of the AUV recovery point. Data is analyzed once the vehicle enters a specified radius of the recovery point, and the input value corresponds to, without loss of generality, the furthest marked point of the vehicle during the task. (The mean distance or Root Mean Square Error could also be used for the input value). As a location-based parameter, GPS is a suitable data source. A generic GPS system with specifications of $1m$ accuracy is used in the modeling of the example MFs. Due to the GPS accuracy limitations and expected ASV maneuverability, the FOU width is set to $1m$ and domain is set to 0-20m, respectively. Figure 8.16 displays the MFs and the corresponding input/output relationship based on the rule base presented in Table 8.7.

Table 8.7. Station-keeping task rule base mapping: input linguistic terms (italicized) to the corresponding output performance linguistic terms

Very Close	*Close*	*Satisfactory*	*Far*	*Very Far*
VG	G	F	P	VP

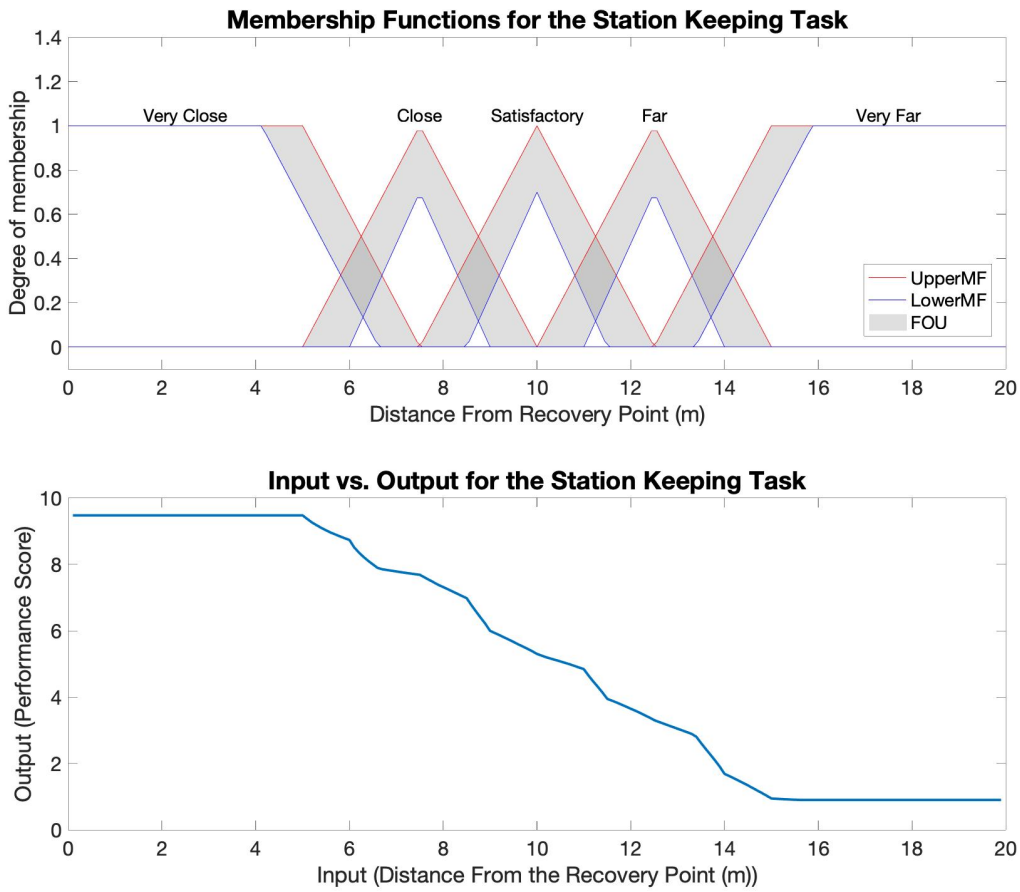


Figure 8.16. Example membership functions for the station keeping task

8.4.1.2 AUV

Image clarity is a parameter that may be used to determine the performance of the vehicle's imaging capabilities. As a subjective and inexact parameter, FL allows a human-determined input via a linguistic term to be mathematically translated into a compatible format with the other input parameters and overall scoring framework. The test engineer has the ability to rank the AUV object

image based upon five classifications: Very Clear, Clear, Satisfactory, Unclear, and Very Unclear. Here, a guideline is required to define the classifications, an example of which is provided in Table 8.8.

Table 8.8. Example image classifications and their corresponding descriptions

Image Classification	*Description*
Very Clear	Object easily identified / no distortion of image
Clear	Object easily identified / minor distortion and imperfections of image
Satisfactory	Outline of object visible / some blurring of image
Unclear	Object barely visible / substantial blurring of image
Very Unclear	No object visible / major image distortion

For docking the AUV to the ASV, a time threshold is used as the input performance parameter. Time starts when the AUV enters within a specified radius of and depth under the ASV and ends when the AUV is successfully docked. A set of MFs for this input parameter may be designed as in Figure 8.17. Since the time measurement is fairly certain, the FOU is modeled to account for the linguistic uncertainty of defining the scoring regions. The FOU bandwidth takes into account this linguistic uncertainty, as it increases as the docking time deviates from the MF median value.

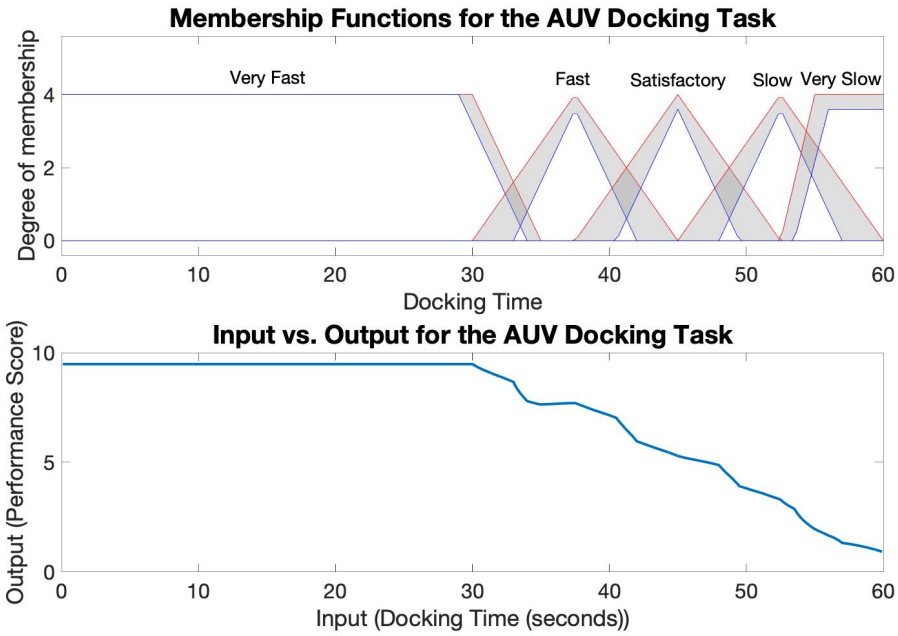


Figure 8.17. Example AUV docking task MFs

8.4.2 Scoring the Full Mission

Arbitrary values are given to each task (without loss of generality) to demonstrate the calculation process for the full mission and are provided in Table 8.9. The weighting terms, w_i, all share the same value. That is, each task is weighted equally. It is noted that the score for the "Search Region" is the result of two input parameters, so it would have an additional calculation involved such that

$$\begin{aligned} \mathbb{P}_{Task2} &= w_1\mathbb{P}_{DetectionAccuracy} + w_2\mathbb{P}_{DetectionTime} \\ \mathbb{P}_{DetectionAccuracy} &= w_1\mathbb{P}_{DetectionLocation} + w_2\mathbb{P}_{FalsePositives} \end{aligned} \tag{8.9}$$

The resulting ASV and AUV performance scores may be calculated, respectively, as

$$\begin{aligned} \mathbb{P}_{ASV} &= w_1\mathbb{P}_{Task1} + w_2\mathbb{P}_{Task2} + w_3\mathbb{P}_{Task3} + w_4\mathbb{P}_{Task4} + w_5\mathbb{P}_{Task5} + w_6\mathbb{P}_{Task6} \\ &= (0.167)8.5 + (0.167)7.5 + (0.167)10 + (0.167)8.5 + (0.167)7.0 + (0.167)8.5 \\ &= 8.35 \end{aligned} \tag{8.10}$$

$$\begin{aligned} \mathbb{P}_{AUV} &= w_1\mathbb{P}_{Task1} + w_2\mathbb{P}_{Task2} + w_3\mathbb{P}_{Task3} + w_4\mathbb{P}_{Task4} \\ &= (0.25)8.2 + (0.25)6.3 + (0.25)8.2 + (0.25)7.6 \\ &= 7.58 \end{aligned} \tag{8.11}$$

$$\begin{aligned} \mathbb{P}_{total} &= w_1\mathbb{P}_{ASV} + w_2\mathbb{P}_{AUV} \\ &= (0.5)8.35 + (0.5)7.58 \\ &= 7.96 \end{aligned} \tag{8.12}$$

ASV Task Scores

Task	Task ID	$\mathbb{P}$
Transit to Search Region	1	8.5
Search Region	2	7.5
Deploy AUV	3	10
Transit to Recovery Location	4	8.5
Station Keep / Recover AUV	5	7.0
Transit Home	6	8.5

AUV Task Scores

Task	Task ID	$\mathbb{P}$
Transit to Object Location	1	8.2
Image Object	2	6.3
Transit to Recovery Point	3	8.2
Dock with ASV	4	7.6

Table 8.9. Performance score values given for each task to demonstrate the process of calculating a single mission score

As shown, the end result is a simple calculation that provides a single evaluation score built from layered metrics. This score is the foundation for defining β in future work as the mapping from the scenario space to the capability space (as shown in Chapter 4).

8.4.3 Discussion: Case Study VI

Case Study VI demonstrates the ability of PERFORM to build and score full missions in addition to being able to accommodate missions involving multiple platforms. Other potential input parameters, such as image quality and object detection accuracy, are also introduced. As shown, some tasks are more appropriately modeled as binary terms (e.g., "turn off payload"), while other tasks are more appropriately modeled with fuzzy sets. The PERFORM methodology is versatile enough to accommodate both data types and, as such, results in increased flexibility in test design.

In modeling and simulation, simplifications and assumptions are acceptable in areas that are not deemed critical. These decisions are made based on some function of available time, resources, and system requirements. Similarly, PERFORM is adaptable to the level of detail warranted. Of course, since this involves live testing, safety becomes the most important factor. As a complement to simulation-based testing, where it is easier to simulate and verify components at lower task levels, PERFORM provides data for higher level and system of systems (SoS) testing which is difficult to achieve with simulation environments. As robotic systems become more and more complex, methods that can scale to the evaluation of the systems coordination is critical to re-

veal otherwise unexpected inter-system interactions. Again, it must be noted that clear guidelines of what constitutes mission success (or failure) and a user's level of acceptable risk both play a significant role in the interpretation of performance scores.

With the modular structure of PERFORM, full-mission testing is no longer needed to update (sub)task scores. For example, if a new path planner needs to be analyzed and compared to that of the planner currently in use, one may simply isolate a transit task and implement the new algorithm. Since the input parameter is decoupled from other input parameters, the score for that specific input can be compared and updated to reflect the "better" path planner. This also applies to extrapolating data for mission variations. Tasks may be added or removed without necessitating a full mission re-run, while noting systems that span the entire mission, i.e. the mission planner, would, of course, require testing a larger subset of tasks or full mission test.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

9.1 Summary

Through this work, contributions are made in the areas of autonomous path planning and the evaluation of autonomous systems. Three global path planners (A*, RRT, and PRM) are analyzed. They are used as the basis from which a novel hybrid path planner is developed that combines A* and PFM using a multi-layered vector-field approach. The effectiveness of this proposed multi-layered path planning method is confirmed via both numerical simulations and experimental testing. Results demonstrated improved routing compared to that using A* or PFM separately. Additionally, in experimental validation testing, the algorithm showed its ability to perform real-time path-planning updates to take into account newly discovered obstacles.

In developing the autonomous path planning methods, small-scale ASV platforms, referred to as Testing Unmanned Performance PlatformS (TUPPS) were designed and manufactured to serve as testbed platforms for rapid algorithm prototyping. A software architecture was constructed and Robotic Operating System (ROS) was implemented as the software middleware framework for the platforms. The software architecture was designed for the ability to interchange autonomy modules. This interchange was accomplished by creating a standardized autonomy message passed between the autonomy module and the "frontseat."

This research introduces a generic architecture for mission design and definitions with the enabling concept being the decomposition of missions into associated mission tasks, behaviors, and events. It also introduces the concept of scenario and capability spaces. These fundamental principles and definitions are the building blocks for which the Performance Evaluation and Review

Framework Of Robotic Missions (PERFORM), the underlying autonomy metrics and evaluation framework, is established.

The PERFORM process incorporates AI-based evaluation methods. Here, the use of Interval Type-2 Fuzzy Logic (IT2-FL) satisfies several criteria including: objectively taking into account subjective data, providing flexibility with regards to defining testing goals, enabling scalability of mission and task types, incorporating uncertainty, managing many different data types, and maintaining independence from internal autonomy architectures – all while maintaining a mathematically rigorous (and objective) structure.

Specific recommendations for the PERFORM design methodology, input parameter selection, membership function construction, rule base designations and FOU design are provided. Here, special care was given to observe the specific relationship between FOU size and output behavior. The end result is a procedure for calculating an overall autonomy mission performance score based upon decomposed mission sub/tasks and respective behaviors and events.

Several case studies are presented to demonstrate PERFORM's efficacy, modularity, versatility, scalability, and overall versatility. The test cases specifically provide detailed examples of PERFORM test design and modeling parameter customization to suit both user needs and his/her corresponding priorities regarding mission goals and acceptable risks. PERFORM is shown to also be applicable to high-level elements, such as in mission planning and for constructing full missions from multiple tasks. The mission decomposition strategy (i.e., principle of superposition) was shown to be valid and effective, as simulation scores between the individual tasks and that of the full mission were very similar.

9.2 Future Work

The future work resulting from this research can be summarized into the following investigations:

1. **PFM/A* Path-Planning Techniques**

   The proposed path-planning algorithm would benefit greatly from an optimization technique for α and β, the attractive and repulsive gain, respectively. In addition, a strategy is needed

to fully integrate (1) the motion planning strategy that computes feasible trajectories for the vehicle-based control system and (2) the path-planning vector field output. The incorporation of additional layers to account for other forces (e.g., currents, wind, shallow areas) would further contribute to the efficacy, versatility, and overall performance of the multi-layered path-planning method. Here, one would also require a methodical process with which to determine the relative "layer weighting" (i.e., to prioritize the effects of one layer over that of another).

Further experimental testing is the next step to progress this research work. Additional testing is necessary to investigate the Velodyne VLP-16 Lidar (or other applied sensor) parameter specifications (i.e., the number of available data points, the range of detection, etc.), so as to determine the appropriate balance/compromise between map resolution and resulting computational requirements.

2. **PERFORM**

Current PERFORM strategies dictate the need for a better understanding for and methods of quantifying the relationship between the FOU input and output parameters. This problem is currently an emerging area of interest within Fuzzy Logic research. In addition, a software tool (i.e., user interface) with which to implement PERFORM would allow for efficient MF design and would enable the construction of an input parameter library to further reduce excessive repetition. Database capability to streamline performance score tracking and test data would also increase functionality and help efficiently compare autonomy systems. One may also opt to incorporate stochastic techniques and, therefore, would require a large dataset to apply the PERFORM process for appropriate statistical analysis. Regardless, comparing actual field test results with that of mission simulations would most certainly confirm the validity and efficacy of PERFORM, in addition to improving the integrity of simulation models.

9.3 Conclusions

With this research, insight and appreciation was gained for the deep complexity and potential for autonomous vehicles as autonomous systems research pushes forward technologically. First, a novel path planner implementation using a hybrid A* and Potential Field Method algorithm resulted in improved routing compared to the two algorithms alone (i.e., a "best of both worlds" scenario). In addition, broader impacts were shown for using layered vector fields to account for various vehicle information.

To continue to build trust and confidence in these systems, an adaptable methodology is needed to account for increasing system complexity and to adequately "test" the autonomous platforms prior to use in actual missions. PERFORM is a significant step towards standardizing autonomy evaluation for most any type of robotic platform, as PERFORM provides a foundational and generic structure with which to construct and evaluate autonomy test missions. Results demonstrated the viability of IT2-FL for creating the metric functions. The results also showed the benefits of the overall mission decomposition strategy.

BIBLIOGRAPHY

[1] Automobile safety. Available at https://americanhistory.si.edu/america-on-the-move/essays/automobile-safety (Accessed 2020/08/17). National Museum of American History.

[2] Marvelmind robotics. Available at https://marvelmind.com.

[3] Merriam webster dictionary. Available at https://www.merriam-webster.com/dictionary/autonomous.

[4] MOOS-IvP: Design considerations of MOOS-IvP. Available at https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Helm.HelmDesignIntro.

[5] MOOS-IvP home page. Available at https://oceanai.mit.edu/moos-ivp/pmwiki/pmwiki.php?n=Main.HomePage.

[6] MOOS-IvP: Introduction to the ivpbuild toolbox. Available at https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Helm.IvPBuildOverview.

[7] NOAA standards. Available at https://nauticalcharts.noaa.gov/publications/standards-and-requirements.html.

[8] Ros home page. Available at https://www.ros.org.

[9] Safety report. Available at https://waymo.com/safety/ (Accessed 2020/08/17).

[10] O. Adiyatov and H. Varol. Rapidly-exploring random tree based memory efficient motion planning. *2013 IEEE International Conference on Mechatronics and Automation, Mechatronics and Automation (ICMA)*, 2013.

[11] N. F. Al-Shammari and J.-S. Oh. Effects of human error on marine safety: Case study. *Journal of Engineering Research and Application*, 8, 2018.

[12] S. Aminifar and A. Marzuki. Uncertainty in interval type-2 fuzzy systems. *Mathematical Problems in Engineering*, 2013.

[13] K. Astrom. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 1965.

[14] P. Bartz. MOOS-IvP application code for razorahrs.cpp, 2017.

[15] N. Benatar, U. Aickelin, and J. Garibaldi. Performance measurement under increasing environmental uncertainty in the context of interval type-2 fuzzy logic based robotic sailing. *ArXiv*, 2013.

[16] I. R. Bertaska, B. Shah, K. von Ellenrieder, P. Svec, W. Klinger, A. J. Sinisterra, M. Dhanak, and S. K. Gupta. Experimental evaluation of automatically-generated behaviors for usv operations. *Ocean Engineering*, 2015.

[17] K. Betts and M. Petty. Automated search-based robustness testing for autonomous vehicle software. *Modelling and Simulation in Engineering*, 2016.

[18] M. Bogochow, J. Masucci, and C. Noel. MOOS-IvP application code for imoosarduino.cpp, 2017.

[19] A. Bouchard and R. Tatum. Verification of autonomous systems: Challenges of the present and areas for exploration. *Proceedings of OCEANS 2015 - MTS/IEEE, Washington, DC, USA*, 2015.

[20] D. Brutzman, C. Blais, D. Davis, and R. McGhee. Ethical mission definition and execution for maritime robots under human supervision. *IEEE Journal of Oceanic Engineering*, 43, 2018.

[21] H. Cabral, J. Alves, N. Cruz, J. Valente, and D. Lopes. Mpl—a mission planning language for autonomous surface vehicles. *Robotic Sailing*, 2013.

[22] O. Castillo and L. Aguilar. *Type-2 Fuzzy Logic in Control of Nonsmooth systems: Theoretical Concepts and Applications*. Springer International Publishing, 2019.

[23] O. Castillo, L. Aguilar, J. Castro, and M. Garcia-Valdez. A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Information Sciences*, 354, 2016.

[24] G. Chen and W. Zhang. Comprehensive evaluation method for performance of unmanned robot applied to automotive test using fuzzy logic and evidence theory and fnn. *Computers in Industry*, 2018.

[25] S. Choi and W. Yu. Any-angle path planning on non-uniform costmaps. *IEEE International Conference on Robotics and Automation*, 2011.

[26] R. A. Clothier, B. Williams, and T. Perez. A review of the concept of autonomy in the context of the safety regulation of civil unmanned aircraft systems. *Australian System Safety Conference (ASSC)*, 151, 2013.

[27] W. Combs. Authors reply to combinatorial rule explosion eliminated by a fuzzy rule configuration. *IEEE Transactions on Fuzzy Systems*, 7, 1999.

[28] W. Combs and J. Andrews. Combinatorial rule explosion eliminated by a fuzzy rule configuration. *IEEE Transactions on Fuzzy Systems*, 6, 1998.

[29] C. Connolly, J. Burns, and R. Weiss. Path planning using Laplace's equation. *IEEE International Conference on Robotics and Automation*, 1990.

[30] A. Dalpe, A. Cook, M.-W. Thein, and M. Renken. A multi-layered approach to autonomous surface vehicle map-based autonomy. *IEEE OCEANS'18 – Charleston, SC*, 2018.

[31] A. Dalpe and M.-W. Thein. Obstacle avoidance strategies for autonomous surface vehicles. *IEEE OCEANS'17 – Anchorage, AK*, 2017.

[32] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. *The Journal of artificial intelligence research*, 39, 2010.

[33] J. Debnath, D. Majumder, and A. Biswas. Air quality assessments using weighted interval type-2 fuzzy inference system. *Ecological Informatics*, 2018.

[34] S. Dick and A. Kandel. Comments on combinatorial rule explosion eliminated by a fuzzy rule configuration. *IEEE Transactions on Fuzzy Systems*, 7, 1999.

[35] P. Durst. A non-contextual model for evaluating the autonomy level of intelligent unmanned ground vehicles. *Proceedings of the 2011 Ground Vehicle Systems Engineering and Technology Symposium*, 2011.

[36] P. J. Durst and W. Gray. Levels of autonomy and autonomous system performance assessment for intelligent unmanned systems. Technical report, US Army Corps of Engineers - Engineer Research and Development Center, 4 2014.

[37] P. J. Durst, W. Gray, A. Nikitenko, J. Caetano, M. Trentini, and R. King. A framework for predicting the mission-specific performance of autonomous unmanned systems. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, 2014.

[38] R. Dutta, X. Guo, and Y. Jin. Quantifying trust in autonomous system under uncertainties. *Proceedings of the 29th IEEE International System-on-Chip Conference (SOCC), Seattle, WA, USA*, 2016.

[39] S. R. Fiorini, J. L. Carbonera, P. Gonçalves, V. A. Jorge, V. F. Rey, T. Haidegger, M. Abel, S. A. Redfield, S. Balakirsky, V. Ragavan, H. Li, C. Schlenoff, and E. Prestes. Extensions to the core ontology for robotics and automation. *Robotics and Computer-Integrated Manufacturing*, 33, 2015.

[40] E. Frazzoli, M. Dahlel, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Proceedings of the American Control Conference*, 2014.

[41] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41, 2017.

[42] B. Gerkey and M. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 232, 2004.

[43] E. I. Grotli, T. A. Reinen, K. Grythe, A. A. Transeth, M. Vagia, M. C. Bjerkeng, P. Rundtop, E. Svendsen, J. O. Redseth, and G. Eidnes. Seatonomy. *OCEANS 2015 - MTS/IEEE Washington*, 2015.

[44] M. Guerra, D. Efimov, G. Zheng, and W. Perruquetti. Avoiding local minima in the potential field method using input-to- state stability. *Control Engineering Practice*, 55, 2016.

[45] B. C. Guevara. An overview of the class of rapidly-exploring random trees. Bachelor's of Science Thesis, Utrecht University, 2018.

[46] N. Gyagenda, O. Gamal, and R. Hubert. A non-contextual method for determining the degree of autonomy to develop in a mobile robot. *International Federation of Automatic Control*, 50, 2017.

[47] A. Hamam and N. D. Georganas. A comparison of Mamdani and Sugeno fuzzy inference systems for evaluating the quality of experience of hapto-audio-visual applications. *HAVE 2008 – IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2008.

[48] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on systems science and cybernetics*, 4, 1968.

[49] M. Hauskrecht. Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13, 2000.

[50] A. Higgins. A dynamic tabu search for large-scale generalised assignment problems. *Computers and Operations Research*, 28, 2001.

[51] C.-E. Hrabia, N. Masuch, and S. Albayrak. A metrics framework for quantifying autonomy in complex systems. In J. P. Muller, W. Ketter, G. Kaminka, G. Wagner, and N. Bulling, editors, *Multiagent System Technologies*, pages 22–41. Springer International Publishing, 2015.

[52] H. Huang, E. Messina, and J. Albus. Toward a generic model for autonomy levels for unmanned systems (alfus). Technical report, National Institute of Standards and Technology (NIST), 09 2003.

[53] H. Huang, E. Messina, and J. Albus. Autonomy levels for unmanned systems (alfus) framework: Volume ii - framework models. Technical report, National Institute of Standards and Technology (NIST), 12 2007.

[54] M. A. III, D. Mahaffee, M. Vale, J. Kitfield, and H. Renner. The autonomous vehicle revolution: Fostering innovation with smart regulation. Available at `https://www.ftc.gov/system/files/documents/public_comments/2017/03/00002-140353.pdf` (Accessed 2020/08/18), 2017.

[55] C. C. Insaurralde and D. Lane. Autonomy-assessment criteria for underwater vehicles. *IEEE/OES Autonomous Underwater Vehicles*, 2012.

[56] N. Kalra and S. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Available at `https://www.rand.org/pubs/research_reports/RR1478.html` (Accessed 2020/05/11).

[57] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. Available at `http://arxiv.org/abs/1105.1186` (Accessed 2020/09/03), 2011.

[58] N. Karnik and J. Mendel. Type-2 fuzzy logic systems. *IEEE Trans. on Fuzzy Systems*, 7, 1999.

[59] N. Karnik and J. Mendel. Centroid of a type-2 fuzzy set. *Information Sciences*, 132, 2001.

[60] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 14, 1998.

[61] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12, 1996.

[62] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5, 1986.

[63] H. Kim, D. Kim, J. Shin, H. Kim, and H. Myung. Angular rate- constrained path planning algorithm for unmanned surface vehicles. *Ocean Engineering*, 84, 2014.

[64] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *Neoplasia*, 2, 1991.

[65] G. Korsah, A. Stentz, and M. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32, 2013.

[66] K. Kwok, B. D. an C. Philips, and C. Tovey. Analyzing the multiple-target- multiple-agent scenario using optimal assignment algorithms. *Journal of Intelligent and Robotic Systems*, 35, 2002.

[67] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

[68] S. M. Lavalle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20, 2001.

[69] D. Li, P. Wang, and L. Du. Path planning technologies for autonomous underwater vehicles–a review. *IEEE Access*, 7, 2019.

[70] L. Li, W.-L. Huang, N.-N. Zheng, and F.-Y. Wang. Intelligence testing for autonomous vehicles: A new approach. *IEEE Transactions on Intelligent Vehicles*, 1, 2016.

[71] N. Li, A. Girard, and I. Kolmanovsky. Stochastic predictive control for partially observable Markov decision processes with time- joint chance constraints and application to autonomous vehicle control. *Journal of Dynamic Systems, Measurement, and Control*, 141, 2019.

[72] H. Y. Lim. *Autonomous Vehicles and the Law*. Edward Elgar Publishing, Cheltenham, UK, 2018.

[73] O. Linda and M. Manic. Comparative analysis of type-1 and type-2 fuzzy control in context of learning behaviors for mobile robotics. *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society, Glendale, AZ*, 2010.

[74] O. Linda and M. Manic. Uncertainty modeling for interval type-2 fuzzy logic systems based on sensor characteristics. *IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), Paris*, 2011.

[75] Z. Liu, Y. Zhang, X. Yu, and C. Yuan. Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control*, 41, 2016.

[76] O. Loe. Collision avoidance for unmanned surface vehicles. Master's thesis, Norwegian University of Science and Technology, Trondheim, 6 2008.

[77] A. Lotfi, H. Andersen, and A. Tsoi. Matrix formulation of fuzzy rule-based systems. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 26, 1996.

[78] A. Lotfi and A. Tsoi. Importance of membership functions: A comparative study on different learning methods for fuzzy inference systems. *Proceedings of the Third IEEE International Conference of Fuzzy Systems*, 1994.

[79] J. Lysgaard, A. Letchford, and R. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100, 2004.

[80] H. Lyu and Y. Yin. Colregs-constrained real-time path planning for autonomous ships using modified artificial potential fields. *The Journal of Navigation*, 72, 2019.

[81] H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7, 1975.

[82] Mathworks. Probabilistic roadmaps. Available at `https://www.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html` (Accessed 2017), 2017.

[83] Mathworks. Mamdani and Sugeno fuzzy inference systems. Available at `https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html` (Accessed 2020), 2020.

[84] J. McMahon and E. Plaku. Mission and motion planning for autonomous underwater vehicles operating in spatially and temporally complex environments. *IEEE Journal of Oceanic Engineering*, 41, 2016.

[85] J. Mendel. Type-2 fuzzy sets: Some questions and answers. *IEEE Connections: Newsletter of the IEEE Neural Networks Society*, 2003.

[86] J. Mendel. Type-2 fuzzy sets and systems: An overview. *IEEE Computational Intelligence Magazine*, 2007.

[87] J. Mendel. General type-2 fuzzy logic systems made simple: A tutorial. *IEEE Transactions on Fuzzy Systems*, 22, 2014.

[88] J. Mendel, H. Hagras, W. Tan, W. Melek, and H. Ying. *Introduction to Type-2 Fuzzy Logic Control*. IEEE Press, 2014.

[89] J. M. Mendel. Fuzzy sets for words: Why type-2 fuzzy sets should be used and how they can be used. IEEE FUZZ, 2004.

[90] J. M. Mendel and Q. Liang. Comments on combinatorial rule explosion eliminated by a fuzzy rule configuration. *IEEE Transactions on Fuzzy Systems*, 7, 1999.

[91] J. M. Mendel and H. Wu. Centroid uncertainty bounds for interval type-2 fuzzy sets: Forward and inverse problems. *FUZZ-IEEE*, 2004.

[92] J. M. Mendel and H. Wu. New results about the centroid of an interval type-2 fuzzy set, including the centroid of a fuzzy granule. *Information Sciences*, 177, 2007.

[93] T. Menzel, G. Bagschik, and M. Maurer. Scenarios for development, test and validation of automated vehicles. *In the Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[94] O. Milbredt. Parameter weighting for multi-dimensional fuzzy inference system. *International Conference on Control, Automation and Information Sciences (ICCAIS)*, 2016.

[95] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[96] G. E. Mullins, P. G. Stankiewicz, and S. Gupta. Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore*, 2017.

[97] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, J. D. Appler, M. H. Biggins, K. Chiou, M. A. Huntley, J. D. Stewart, and A. S. Watkins. Delivering test and evaluation tools for autonomous unmanned vehicles to the fleet. *Johns Hopkins APL Technical Digest*, 33, 2015.

[98] R. R. Murphy. *Introduction to AI Robotics*. MIT Press, 2000.

[99] E. Nunes, M. Manner, H. Mitiche, and M. Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90, 2017.

[100] J. I. Olszewska, M. Barreto, J. Bermejo-Alonso, J. Carbonera, A. Chibani, S. Fiorini, P. Goncalves, M. Habib, A. Khamis, A. Olivares, E. P. de Freitas, E. Prestes, S. V. Ragavan, S. Redfield, R. Sanz, B. Spencer, , and H. Li. Ontology for autonomous robotics. *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017.

[101] M. Park and M. Lee. Real-time path planning in unknown environment and a virtual hill concept to escape local minima. *30th Annual Conference of IEEE Industrial Electronics Society*, 2004.

[102] A. Pavin and A. Inzartsev. A GeoJSON-based mission planning language for auv. *IEEE OCEANS'18 – Charleston, SC*, 2018.

[103] S. Polonski. Can we teach morality to machines? Three perspectives on ethics for artificial intelligence. Available at `https://medium.com/@drpolonski/can-we-teach-morality-to-machines-three-perspectives-on-ethics-for-artificial-intelligence-64fe479e25d3`.

[104] E. Prestes, J. L. Carbonera, S. R. Fiorini, V. A. M. Jorge, M. Abela, R. Madhavanb, A. Locoroc, P. Goncalves, M. E. Barretof, M. Habibg, A. Chibanih, S. Gérardi, Y. Amirath, and C. Schlenoffj. Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, 61, 2013.

[105] Y. Rasekhipour, A. Khajepor, S.-K. Chen, and B. Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 18, 2017.

[106] S. Redfield and M. Seto. *Autonomy and Artificial Intelligence: A Threat or Savior?*, chapter Verification Challenges for Autonomous Systems. Springer International Publishing, 2017.

[107] W. Roberts, P. Meyer, S. Seifert, E. Evans, M. Steffens, and D. Mavris. A flexible problem definition for event-based missions for evaluation of autonomous system behavior. *OCEANS MTS/IEEE Charleston, SC*, 2018.

[108] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, 2019.

[109] A. V. Savkin, A. S. Matveev, M. Hoy, and C. Wang. *Safe Robot Navigation among Moving and Steady Obstacles*. Elsevier Science and Technology, 2015.

[110] G. Saxena and K. Nanath. Cloud performance evaluation using fuzzy logic. *IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2015.

[111] P. Schorner, L. Tottel, J. Doll, and J. Zollner. Predicted trajectory planning in situations with hidden road users using partially observable Markov decision processes. *IEEE Intelligent Vehicles Symposium*, 2019.

[112] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1, 2018.

[113] D. Seward, C. Pace, and R. Agate. Safe and effective navigation of autonomous robots in hazardous environments. *Autonomous Robot*, 22, 2007.

[114] S. Singh. Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey. Technical report, U.S Department of Transportation – National Highway Traffic Safety Administration, 02 2015.

[115] H. Sola, J. Fernandez, H. Hagras, H. Francisco, M. Pagola, and E. Barrenechea. Interval type-2 fuzzy sets are generalization of interval-valued fuzzy sets: Toward a wider view on their relationship. *IEEE Transactions on Fuzzy Systems*, 2015.

[116] R. Song, Y. Liu, and R. Bucknall. A multi-layered fast marching method for unmanned surface vehicle path planning in a time-variant maritime environment. *Ocean Engineering*, 129, 2017.

[117] R. Song, Y. Liu, and R. Bucknall. Smoothed a* algorithm for practical unmanned surface vehicle path planning. *Applied Ocean Research*, 83, 2019.

[118] M. Sugeno. *Industrial applications of fuzzy control*. Elsevier Science Pub. Co, 1985.

[119] Y. Sun, G. Xiong, W. Song, J. Gong, and H. Chen. Test and evaluation of autonomous ground vehicles. *Advances in Mechanical Engineering*, 2014.

[120] A. A. Surya, M. Kurian, and S. M. Varghese. Overall performance evaluation of engineering students using fuzzy logic. *International Journal on Cybernetics & Informatics*, 2016.

[121] P. Svec, B. Shah, I. Bertaska, J. Alvarez, A. Sinisterra, K. von Ellenrieder, M. Dhanak, and S. Gupta. Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[122] E. Ueland. A star search algorithm. Available at `https://www.mathworks.com/matlabcentral/fileexchange/56877-a---astar---a-star--search-algorithm-easy-to-use` (Accessed 2017), 2017.

[123] C. Wang. A study of membership functions on Mamdani-Type fuzzy inference system for industrial decision-making. Master's thesis, Lehigh University, United States, 1 2015.

[124] S. Wang, M. Fu, Y. Wang, and L. Zhao. A multi-layered potential field method for water-jet propelled unmanned surface vehicle local path planning with minimum energy consumption. *Polish Maritime Research*, 26, 2019.

[125] J. Weinschenk, W. Combs, and R. M. II. Avoidance of rule explosion by mapping fuzzy systems to a union rule configuration. *The IEEE International Conference on Fuzzy Systems*, 2003.

[126] D. Wu. Twelve considerations in choosing between Gaussian and trapezoidal membership functions in interval type-2 fuzzy logic controllers. *IEEE World Congress on Computational Intelligence*, 2012.

[127] D. Wu and J. Mendel. Uncertainty measures for interval type-2 fuzzy sets. *Information Sciences*, 177, 2007.

[128] H. Wu and J. M. Mendel. Recommendations on designing practical interval type-2 fuzzy systems. *CoRR*, abs/1907.01697, 2019.

[129] G. Xiong, X. Zhao, H. Liu, S. Wu, J. Gong, H. Zhang, H. Tan, and H. Chen. Research on the quantitative evaluation system for unmanned ground vehicles. *Proceedings of the 2010 IEEE Intelligent Vehicles Symposium, San Diego, CA, USA*, 2010.

[130] Y. Xue, D. Clelland, B. Lee, and D. Han. Automatic simulation of ship navigation. *Ocean Engineering*, 38, 2011.

[131] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia. Survey of robot 3D path planning algorithms. *Journal of Control Science And Engineering*, 2016.

[132] Q. Yao, Z. Zheng, L. Qi, H. Yuan, X. Guo, M. Zhao, Z. Liu, and T. Yang. Path planning method with improved artificial potential field—a reinforcement learning perspective. *IEEE Access*, 2020.

[133] S. Yeasmin, A. K. Paul, and P. C. Shill. Optimization of interval type-2 fuzzy logic controllers with rule base size reduction using genetic algorithms. *3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, 2016.

[134] L. Zadeh. Fuzzy sets. *Information and Control*, 8, 1965.

[135] L. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8, 1975.

[136] L. Zadeh. Toward a perception-based theory of probabilistic reasoning with imprecise probabilities. *Journal of Statistical Planning and Inference*, 105, 2002.

[137] S. M. Zadeh. *Autonomous Reactive Mission Scheduling and Task-Path Planning Architecture for Autonomous Underwater Vehicle*. PhD thesis, Flinders University, South Australia, 12 2016.

[138] T. Zhang, D. Tao, X. Qu, X. Zhang, R. Lin, and W. Zhang. The roles of initial trust and perceived risk in public's acceptance of automated vehicles. *Transportation Research Part C*, 98, 2019.

[139] H. Zimmermann. *Fuzzy Set Theory and Its Applications*. Springer, 2001.