A Sampling-Based Model Predictive Control Approach to Motion Planning For Autonomous Underwater Vehicles

Charmane Venda Caldwell
A SAMPLING-BASED MODEL PREDICTIVE CONTROL APPROACH TO MOTION PLANNING FOR AUTONOMOUS UNDERWATER VEHICLES

By
CHARMANE VENDA CALDWELL

A Dissertation submitted to the
Department of Electrical and Computer Engineering
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2011
The members of the committee approve the dissertation of Charmane Venda Caldwell defended on April 21, 2011.

____________________

Emmanuel G. Collins Jr.
Professor Co-Directing Dissertation

____________________

Rodney Roberts
Professor Co-Directing Dissertation

____________________

David Cartes
University Representative

____________________

Linda S. DeBrunner
Committee Member

Approved:

____________________

Simon Y. Foo, Chair, Department of Electrical and Computer Engineering

____________________

John Collier, Interim Dean, FAMU-FSU College of Engineering

The Graduate School has verified and approved the above-named committee members.
This dissertation is dedicated to,
my mother, Petrene,
my godmother, Mary,
and my grandmother, Josephine,
on whose shoulders I stand.
ACKNOWLEDGMENTS

There are a number of people who deserve acknowledgement and my heartfelt gratitude for their encouragement, guidance, advice and support through my life journey that made this dissertation possible.

First and foremost, I would like to give honor and praise to God for allowing me to accomplish my dreams. I especially want to thank my advisor and committee chair Dr. Emmanuel Collins for his guidance, funding, patience and demand for quality work during this process. Thanks to Dr. Rodney Roberts, Dr. Linda DeBrunner and Dr. Dave Cartes for serving on my committee.

I would like to give special thanks to Dr. Stephan von Molnar, for giving me my first research opportunity and encouraging me to pursue a higher degree; Dr. Peter Kalu, for believing that I could obtain a PhD and his willingness to express that to others; Dr. Patricia Stith, for her caring nature during my time at FSU; Dr. Carl Moore, for allowing me to present my research and giving me guidance during his research group meetings; Dr. Makola Abdullah, for playing a crucial role in my development as a researcher, by giving me valuable advice on how to get over a hump to get results; Dr. Simone Peterson-Hruda, for her helpful advice through my PhD journey; and my fourth grade teacher, Mrs. Thomas, for telling me I could go to college.

Beyond individuals who assisted me academically through the process, several fellowships supported me during this research. I am indebted to FGLSAMP, McKnight Doctoral Fellowship and ONR HBECU Future Engineering Faculty Fellowship for not only providing financial resources during my PhD but also networking and professional development
resources. It was during a FGLSAMP expo in undergrad that led me to pursue a PhD; McKnight conferences taught me about the PhD process; and ONR HBECU FEF provided me with valuable internships.

My gratitude goes to the Mechanical Engineering Department, Electrical and Computer Engineering Department and College of Engineering staff for making everything on the administrative side easy. Special thanks go to Mrs. Nelson and Mrs. Jackson for making sure I had all my forms in on time; Mrs. Humose, for making sure I received my checks; Donald, for handling purchasing of equipment, travel and funding; Mrs. Haynes, for moral support and lunch time talks; Mr. Hall, for encouraging me and his funny stories. I will always be grateful to the other graduate students and postdocs in my lab and at the COE for helping me mature as a researcher, especially, Dr. Seleka, for teaching me how to properly read a scientific paper; Damion Dunlap, for his work on SBMPC; and Kevin Brinson, for advising me on how to become a serious engineering student and introducing me to the importance of a higher degree.

During my time working on my dissertation, I have had great friends that made the experience more enjoyable. These friends helped me in life more than words can express. Thanks to Shawn, for helping me when I was struggling with my research by providing me with the steps that lead to my first results, being my study partner during the preliminary exam, conversing with me endless about our research and lending moral support; Bridget, for inviting me into her family, showing me how to be a friend and taking many enjoyable trips with me throughout our PhD process; Rosalyn, for listening to me talk through my hardships and providing words of encouragement; Eno, for giving me courage and confidence when I was at my lowest; Charlie, for all our late night conversations about our shared hardship experiences; Tonia, for being my biggest fan for 23 years, being more excited for me than I am for myself sometimes and keeping me in touch with the real world through all her humor; Mrs. LN, for feeding me and praying with me throughout my college years; John, for making sure I ate; and Lamar, for proofreading my dissertation.
Finally, the support of my family and friends of the family from the beginning of my education all the way through to the completion of my graduate studies must be acknowledged. I am grateful for the sacrifices they made and the environment they created that encouraged me to succeed. This work is just as much a product of their efforts as it is mine. Special thanks to my mother, Petrene, for always encouraging me, supporting me, having unfailing confidence in me and never worrying about how long it took me to finish; my father, Billy, for his unconditional love and the sacrifices he has made for me over the years; my grandparents, Warren and Josephine Alexander, for noticing a spark in me at a young age; my grandparents, Bill and Colean Caldwell, for their love and support; my godparents Paul and Mary Muller, for taking care of me, feeding me, making sure I finished my homework before I went outside, giving me my moral foundation and making sure I knew the Lord...I owe them everything; Aunt Venda Raye, for being an example of a truly independent woman and giving me the strength to not back down to anyone; Aunt Asonia, for being an example of a classy lady with a meek spirit, showing me that the smartest person in the room is not always the one that speaks the loudest but is wise enough not to and being one of the people that made sure I had a spiritual foundation; uncles Warren Jr., Oran and Wayne, for being caring and loving; my cousins, Robin, Bridget, Erika and Ryan, for being more like sisters and a brother; Robin, for sending me money my freshman year in college and giving me my first car; Bridget, for teaching me the ropes and leading the pathway; Erika, for our great conversations and fun trips; Ryan, for always being my partner in crime, protecting me and always coming to my aid; cousin Linda, for stimulating discussions; cousin Junie, for the advice she gave me after my first year of coursework; cousin Candy, for making an awkward looking little girl feel special; cousin Lynne, for my room in the ATL; siblings, Charles and Arnetta, for their unconditional love and retyping my code when my laptop was stolen; and Mrs. Viola, Mrs. Hattie Lee, Mrs. Gloria, Mrs. Rose, the Clarks, cousin JoAnn and cousin Barbara, I cherish the time I spent with you growing up it influenced who I have become as a woman.
All of these people are responsible for equipping me with the tools necessary to be successful in whatever I decide to do in life. I am greatly appreciative.

Charmane V. Caldwell
# Table of Contents

List of Tables ................................................. x
List of Figures ............................................. xi
Abstract ...................................................... xiii

## 1 Introduction
1.1 Background .............................................. 1
1.2 Motivation ............................................... 2
1.3 Literature Review ....................................... 5
  1.3.1 Nonlinear MPC ....................................... 5
  1.3.2 Fast MPC ............................................. 7
  1.3.3 MPC Implemented on Mobile Robots ................. 8
  1.3.4 MPC Implemented on UUVs ......................... 10
  1.3.5 UUV Obstacle Avoidance .......................... 12
  1.3.6 UUV Path Planning .................................. 13
1.4 Research Goals ......................................... 18
1.5 Contributions .......................................... 19
1.6 Dissertation Organization ............................. 19

## 2 Autonomous Underwater Vehicle Models
2.1 General ................................................... 20
2.2 Vehicle Kinematics ...................................... 21
2.3 Vehicle Dynamics ....................................... 25
2.4 Vehicle Decoupled Model ............................... 28
2.5 Inclusion of the Kinematic and Dynamic Models in AUV Motion Planning ........................... 30

## 3 Sampling Based Model Predictive Control
3.1 MPC Overview ........................................... 34
  3.1.1 Receding Horizon Model Predictive Control ...... 34
  3.1.2 Shrinking Horizon Model Predictive Control .... 37
  3.1.3 MPC Advantages ..................................... 38
  3.1.4 MPC Disadvantages .................................. 38
3.2 Evolution of SBMPC ..................................... 39
3.3 SBMPC Overview ....................................... 41
  3.3.1 Sampling Based Motion Planning ................... 41
3.3.1.1 Input Sampling ................................................. 43
3.3.1.2 The Implicit State Grid ................................... 44
3.3.2 Goal Directed Optimization ................................ 47
3.4 SBMPC Algorithm ............................................... 47
3.4.1 SBMPC Variables ............................................. 48
3.4.2 The SBMPC Algorithm ....................................... 49
3.5 Comparison of SBMPC and MPC ............................... 50

4 SIMULATIONS ...................................................... 57
4.1 Simulation Comparison of NMPC and SBMPC Using a 2D Kinematic Model 57
   4.1.1 Random Obstacles in Cluttered Environment ................. 58
   4.1.2 Random Start and Goal Cluttered Environment ............... 61
4.2 SBMPC Using a 3D Kinematic Model ............................. 63
   4.2.1 Steep Hill .................................................. 66
   4.2.2 Cliff ....................................................... 66
   4.2.3 Local Minima Scenario .................................... 66
   4.2.4 Multiple Obstacles ......................................... 68
4.3 SBMPC Using the Decoupled and Full Dynamic Models ........... 69
   4.3.1 Multiple Obstacles ......................................... 71
   4.3.2 Steep Hill .................................................. 73
   4.3.3 Cliff ....................................................... 73
4.4 Tuning Parameters ................................................ 75
   4.4.1 Effect of Sample Number .................................. 75
   4.4.2 Effect of Grid Resolution .................................. 76

5 CONCLUSIONS ..................................................... 80
5.1 Summary ......................................................... 80
5.2 Future Works ................................................... 82

A Appendix .......................................................... 87
Bibliography .......................................................... 89
Biographical Sketch .................................................. 97
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Genetic Algorithm Options</td>
<td>59</td>
</tr>
<tr>
<td>4.2</td>
<td>Simulation Parameters</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>Case 1: Simulation Results for Scenario 1</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>Case 2: Simulation Results for Scenario 2</td>
<td>63</td>
</tr>
<tr>
<td>4.5</td>
<td>Simulation Constraints for the 3D Kinematic Model</td>
<td>66</td>
</tr>
<tr>
<td>4.6</td>
<td>Simulation Constraints for the Decoupled and Full Dynamic Models</td>
<td>70</td>
</tr>
<tr>
<td>4.7</td>
<td>Simulation Parameters for the Decoupled and Full Dynamic Models</td>
<td>70</td>
</tr>
<tr>
<td>4.8</td>
<td>Simulation Results for the Decoupled and Full Dynamic Models</td>
<td>72</td>
</tr>
<tr>
<td>A.1</td>
<td>Simulation Parameters for Dynamic Model</td>
<td>87</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Coordinate frames</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Vehicle angles</td>
<td>23</td>
</tr>
<tr>
<td>2.3</td>
<td>(a) Decoupled model with $n = 500$ and $\delta_s = -15^\circ$, (b) Full dynamic Model with $n = 500$ and $\delta_s = -15^\circ$</td>
<td>32</td>
</tr>
<tr>
<td>2.4</td>
<td>(a) Decoupled model with $n = 1000$, $\delta_s = -15^\circ$, $\delta_r = -15^\circ$, (b) Full dynamic model with $n = 1000$, $\delta_s = -15^\circ$, $\delta_r = -15^\circ$</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>Stages of MPC algorithm</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Receding Horizon Model Predictive Control concept</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Shrinking Horizon Model Predictive Control concept</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>A tree that connects the root with a goal region</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>Illustration of the potential inefficiency of sampling in the output space.</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Illustration of the necessity of an implicit state grid</td>
<td>45</td>
</tr>
<tr>
<td>3.7</td>
<td>Illustration of the necessity of an implicit state grid</td>
<td>46</td>
</tr>
<tr>
<td>3.8</td>
<td>SBMPC optimization phase</td>
<td>53</td>
</tr>
<tr>
<td>3.9</td>
<td>Traditional MPC optimization phase</td>
<td>54</td>
</tr>
<tr>
<td>3.10</td>
<td>Traditional MPC optimization</td>
<td>55</td>
</tr>
<tr>
<td>3.11</td>
<td>Goal-directed optimization</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>Cluttered environment scenario in which $MPC_{SQ}$ fails to enable the AUV to reach the goal.</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>Cluttered environment scenario in which SBMPC yielded the only path to the goal.</td>
<td>62</td>
</tr>
<tr>
<td>4.3</td>
<td>The one evenly spaced obstacle scenario in which SBMPC did not reach goal.</td>
<td>64</td>
</tr>
<tr>
<td>Chapter</td>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.4</td>
<td></td>
<td>Recalculation of the path with number of samples increased to 25.</td>
</tr>
<tr>
<td>4.5</td>
<td></td>
<td>Evenly spaced obstacles scenario for which each planning algorithm reached the goal but $MPC_{SQ}$ determined a more distance optimal path.</td>
</tr>
<tr>
<td>4.6</td>
<td></td>
<td>The one evenly spaced obstacle scenario in which $MPC_{SQ}$ failed to enable the AUV to reach the goal.</td>
</tr>
<tr>
<td>4.7</td>
<td></td>
<td>Steep hill scenario.</td>
</tr>
<tr>
<td>4.8</td>
<td></td>
<td>Cliff scenario.</td>
</tr>
<tr>
<td>4.9</td>
<td></td>
<td>Local minima scenario.</td>
</tr>
<tr>
<td>4.10</td>
<td></td>
<td>Random start, goal and obstacle scenario.</td>
</tr>
<tr>
<td>4.11</td>
<td></td>
<td>Random start, goal and obstacle scenario.</td>
</tr>
<tr>
<td>4.12</td>
<td></td>
<td>Decoupled model random start, goal and obstacle scenario.</td>
</tr>
<tr>
<td>4.13</td>
<td></td>
<td>Full dynamic model random start, goal and obstacle scenario.</td>
</tr>
<tr>
<td>4.14</td>
<td></td>
<td>Decoupled model steep hill scenario.</td>
</tr>
<tr>
<td>4.15</td>
<td></td>
<td>(a) Full dynamic model steep hill scenario. (b) Full dynamic model steep hill scenario with input commands from decoupled model.</td>
</tr>
<tr>
<td>4.16</td>
<td></td>
<td>Decoupled model random start, goal and obstacle scenario.</td>
</tr>
<tr>
<td>4.17</td>
<td></td>
<td>The effect of sample size on computation time.</td>
</tr>
<tr>
<td>4.18</td>
<td></td>
<td>(a) Scenario with sample number = 26, (b) Scenario with sample number = 36</td>
</tr>
<tr>
<td>4.19</td>
<td></td>
<td>The effect of sample size on path cost.</td>
</tr>
<tr>
<td>4.20</td>
<td></td>
<td>The effect of grid size on computation time.</td>
</tr>
<tr>
<td>4.21</td>
<td></td>
<td>The effect of grid size on path cost.</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>SBMPC finds the global solution subject to the sampling used and hence avoids local minima.</td>
</tr>
<tr>
<td>5.2</td>
<td></td>
<td>Closed loop paradigm</td>
</tr>
<tr>
<td>5.3</td>
<td></td>
<td>(a) Example of sampling the control, $u$, (b) Example of sampling the change in control, $\Delta u$.</td>
</tr>
<tr>
<td>5.4</td>
<td></td>
<td>Funnel output constraint, a “virtual obstacle”</td>
</tr>
</tbody>
</table>
In recent years there has been a demand from the commercial, research and military industries to complete tedious and hazardous underwater tasks. This has lead to the use of unmanned vehicles, in particular autonomous underwater vehicles (AUVs). To operate in this environment the vehicle must display kinematically and dynamically feasible trajectories. Kinematic feasibility is important to allow for the limited turn radius of an AUV, while dynamic feasibility can take into consideration limited acceleration and braking capabilities due to actuator limitations and vehicle inertia.

Model Predictive Control (MPC) is a method that has the ability to systematically handle multi-input multi-output (MIMO) control problems subject to constraints. It finds the control input by optimizing a cost function that incorporates a model of the system to predict future outputs subject to the constraints. This makes MPC a candidate method for AUV trajectory generation. However, traditional MPC has difficulties in computing control inputs in real time for processes with fast dynamics.

This research applies a novel MPC approach, called Sampling-Based Model Predictive Control (SBMPC), to generate kinematically or dynamically feasible system trajectories for AUVs. The algorithm combines the benefits of sampling-based motion planning with MPC while avoiding some of the major pitfalls facing both traditional sampling-based planning algorithms and traditional MPC, namely large computation times and local minimum problems. SBMPC is based on sampling (i.e., discretizing) the input space at each sample period and implementing a goal-directed optimization method (e.g., $A^*$) in place of standard nonlinear programming. SBMPC can avoid local minimum, has only two parameters to tune, and has small computational times that allows it to be used online fast systems.
A kinematic model, decoupled dynamic model and full dynamic model are incorporated in SBMPC to generate a kinematic and dynamic feasible 3D path. Simulation results demonstrate the efficacy of SBMPC in guiding an autonomous underwater vehicle from a start position to a goal position in regions populated with various types of obstacles.
CHAPTER 1

INTRODUCTION

1.1 Background

The ocean makes up over 70% of the earth’s surface [3]. However, there are more resources spent to explore space, which is millions of miles away, than to explore the depths of the ocean, which is here on earth. To date the ocean has provided minerals, oil, aquatic life, and alternative energy sources. With less than 10% of the ocean explored by humans there are resources that have yet to be discovered [3]. One of the reasons the ocean’s full capabilities have not been uncovered is the technological difficulties in exploring the depths of the sea. However, similar to space exploration, it is possible to employ robots to perform the tasks that are hard for humans to accomplish or would put them in harms way. In particular, an unmanned underwater vehicle (UUV) can be used to traverse the ocean to carry out difficult duties.

An UUV is defined as a, “self-propelled submersible whose operation is either fully autonomous (pre-programmed or real-time adaptive mission control) or under minimal supervisory control and is untethered except, possibly, for data links such as a fiber optic cable” [69]. A more specific type of UUV is an autonomous underwater vehicle (AUV), which is defined as an “unmanned submersible vehicle with onboard systems and subsystems that provide motive power, motion control, navigation, obstacle detection and collision avoidance. To be truly autonomous, the vehicle needs to be able to execute a planned mission by controlling and monitoring the onboard systems without any external input. It can replan
its mission in case of internal anomalies, such as subsystem degradation, and it will have the capability of replanning its path to avoid previously unknown obstacles.” [29]. AUVs are of considerable interest because they reduce the number of humans needed to complete dangerous or tedious tasks. Therefore, for the purpose of this research we will consider AUVs.

1.2 Motivation

There are various AUVs around the world, examples range from the Odyssey II [1] to the REMUS [2], with each vehicle performing different tasks. AUV applications can consist of cable/pipeline inspection, sea floor mapping, oil exploration, geological sampling, deep-sea exploration, environmental monitoring, underwater rescue, wreckage recovery and military missions. In particular, there is an urge for the U.S. Navy to use AUVs to replace humans in completing dangerous military activities; it is easier to replace an AUV than to inform a family member their loved one has died in duty. Below, military mine countermeasure (MCM) missions are used to illustrate various issues related to motion planning for AUVs.

Since the 1950s, naval mines have caused three times the damage to U.S. Fleets than all other threats combined [38]. These damages can cost the lives of sailors and/or damage multi million dollar equipment. A mine is inexpensive to purchase and is readily available. Moreover, a mine has the added benefit they are time consuming to detect and neutralize. All of these factors make the mine a great weapon of choice for coastal lands to use against opposing forces. Hence, the U.S. Navy conducts mine countermeasure missions to clear an area of mines or reroute the vehicles.

There are four primary stages to MCM missions: 1) reconnaissance, 2) identification, 3) reacquisition, and 4) neutralization. In the first stage of a MCM mission, the AUVs search unsecured areas for mine-like objects. The time allowed to investigate the area is limited due to both military concerns and battery life. Upon detection of a mine-like object, it is
necessary to determine whether the object is actually a mine. Therefore, the AUV must carefully examine the recorded area at a standoff distance that is far enough away as not to cause the mine to activate, yet close enough that the object can be identified by a camera or other sensor. Those objects identified as mines can be neutralized or just avoided entirely. If the mine is to be neutralized, there is the added challenge of reaching the target with precision.

Currently, very shallow water mine countermeasure missions are performed by explosive ordinance disposal divers (EOD) and marine animals. In the future, the Navy plans to have UUVs as part of the MCM process [44]. In recent years with the introduction of littoral warfare the Navy has been driven to make an investment in MCM technology, some of which have already been adopted in war situations. For example, in the latest Iraq war, the REMUS AUVs were employed in the Arabian Gulf to aid in the detection and clearing of mines. As part of the Naval Special Clearance Team-One (NSCT-1), which consisted of Navy Seal divers, Explosive Ordinance Disposal (EOD) divers, Marine Corps forces and dolphins, the Navy used the REMUS specifically for searching the area for mine-like objects. It was believed that using the AUVs in this limited role decreased the mission by two days (or 40%). Even though the Navy has had the REMUS since the late 1990s, this marked the first time AUVs had been used during wartime. The Navy’s goal is to increase the intelligence of the vehicle in order to decrease soldier intervention in this dangerous process. Special emphasis is placed on the surf zone (SZ) and very shallow water (VSW) (40 feet or less) where soldiers are most susceptible and underwater communication is challenging [9]. In addition, the sea bottoms are uneven, underwater obstacles, and currents are present. With the increase in UUV technologies it is possible to increase the role AUVs play in military missions and even further decrease the time to complete missions.

As stated previously, for a vehicle to truly be considered an AUV it must have good motion control, obstacle avoidance and path planning capabilities. Standard AUV motion planning first determines a trajectory that the AUV may not be physically able to follow,
then applies a controller that may seek to have the vehicle follow a possibly infeasible trajectory. An approach that can help ensure robust motion planning is to incorporate a model of the AUV when planning the vehicle trajectory since this applies motion constraints that ensure feasible trajectories. In cluttered environments, the use of kinodynamic constraints in motion planning aid in determining a collision free trajectory. The kinematics provides the turn rate constraint and side slip [19], while the dynamics can provide insight into an AUV’s movement and interaction with the water, providing limits on velocities, accelerations and applied forces.

A method called Model Predictive Control (MPC) exhibits the ability to systematically handle multi-input multi-output (MIMO) control and trajectory planning problems subject to the system model and other constraints (e.g., actuator limitations and obstacle avoidance). MPC can be beneficial for AUV motion planning. However, there are some shortcomings in the way MPC is traditionally implemented. The method was developed for slow systems, so there are issues with being able to compute control updates fast enough for vehicles. Additionally, when the system model is nonlinear, such as for the AUV, there is an increase in computation time and problems with convergence to local minima when a traditional gradient based optimization approach is adapted. In order to address these shortcomings a method called Sampling Based Model Predictive Control (SBMPC) was developed [24]. As its name implies, SBMPC is dependent upon the concept of sampling, which has arisen as one of the major paradigms for robotic motion planning [54]. The method samples the inputs to the system, providing a trade-off between performance and computational efficiency. Also, in contrast to previous MPC methods, it does not rely on a linear or nonlinear programming method. Instead SBMPC uses a graph search algorithm derived from $LPA^*$ [51], an incremental $A^*$ algorithm [54]. These concepts produce a method that overcomes some of the problems associated with traditional MPC and allows for a robust AUV motion planner.
1.3 Literature Review

In order to provide a roadmap for this research it is necessary to examine several research areas and describe the methods and techniques that have been executed. Because this research applies concepts from MPC to develop fast kinodynamically feasible motion planning using nonlinear models of AUVs, it is important to review what has been accomplished in nonlinear and fast MPC, how MPC has been applied to vehicles and finally how AUV path planning and obstacle avoidance has been handled in the past.

1.3.1 Nonlinear MPC

A 1996 survey completed by Qin and Badgwell showed that there were MPC vendors that aided in over 2200 control applications for linear systems [75]. However, four years later the 2000 survey by Qin and Badgwell, only identified 88 industrial nonlinear model predictive control (NMPC) applications [76]. Hence, it is apparent that MPC was implemented using linear models and linear constraints the vast majority of the time. Linear MPC with a quadratic objective function is widely used because it leads to a convex optimization problem, which a number of optimization methods can solve efficiently and ensure convergence to a global minimum. On the other hand, once a nonlinear model and/or constraint is inserted, NMPC must solve a nonconvex optimization problem. The problem takes more time to solve, which makes it difficult to solve online in real-time. Also, solution approaches based on standard gradient based methods can converge to local minima. In actuality, the problem is with the optimization methods that are applied in NMPC and not the actual nonlinear model or nonlinear constraints. Since most real world systems and constraints are nonlinear, it would be beneficial to use the nonlinear model when designing the input and output trajectories. Even though it is not as well developed as linear MPC, some researchers have attempted to address the problems that arise in NMPC.

Researchers approach NMPC from two points of view: 1) by reformulating the nonlinear model to a linearized model and 2) use the nonlinear model directly with an enhanced
optimization method.

**Solving NMPC via Linearization**

Zheng determined a linear controller for a given number of regions [103]. The optimization problem is solved online at each time step using one of the controllers developed offline, similar to gain scheduling. Johansen used a piecewise linear function to approximate the solutions of the nonlinear optimization problem offline and store them in a table [43]. The solution was then determined online through a table search. The approach by Seki et al. used successive linearized models of the nonlinear model to prevent having to use a nonlinear programming method [81]. It employed a linear quadratic controller with a feedback controller that modified the control input using quadratic programming. Al-Duwaish and Naeem used a linear model with static nonlinearities instead of the original nonlinear model and then incorporated a genetic algorithm [4]. Long et al. used new variables and constraints to transform the nonconvex nonlinear problem into an equivalent problem with linear terms and simple nonlinear terms then applied a branch and bound method [63].

The above methods are typical of how NMPC is approached in control literature. The problem with these methods are that they are only valid when the system operates in close proximity to the conditions the linear models assume.

**Solving NMPC via New Optimization Methods**

In some cases, NMPC is addressed by developing new optimization methods that actually use the nonlinear model in the MPC process. This tends to be difficult. In the 2000 survey by Qin and Badgwell [76] mentioned earlier, the systems that actually apply an online solution of a nonlinear optimization problem the number goes from 88 to only 5. As an example, Martinsen et al. used a reduced gradient form of sequential quadratic programming on a nonlinear model, which created a reduced space size for independent variables [65].
1.3.2 Fast MPC

One of the weaknesses of MPC has been that computation times tend to be long. Since it was developed for the process industry, which has slow systems, that was originally sufficient. In order to be applicable to problems with fast systems MPC computational times must be substantially decreased. Note that the computation time is dependent on the optimization method, number of control inputs, number of states, control horizon, and prediction horizon. There are various methods that attempt to make sure MPC has fast updates either by performing some calculations offline, making changes to the online optimization method or incorporating a type of control input parametrization. As a result, there is a reduction in computational effort. The researchers of these various papers consider “fast MPC,” the acceptable time to provide control updates for their particular system. However, in this dissertation “fast MPC” corresponds to computation times for which the inverse (essentially the corresponding sampling rates) are at least 5 times faster than the AUV bandwidth. An AUV tends to have a bandwidth in the range [0.1 - 2] Hz, corresponding to periods in the range [0.5 - 10] sec. Hence fast MPC corresponds to computation times faster than some point in the range [0.1 - 2] sec.

Offline Calculations

Bemporad et al. approached fast MPC by using multi-parametric quadratic programming (mp-qp) for linear systems for offline approximation of the controller [12]. It partitions the state space into a given number of convex polyhedral regions and the solution is solved offline for each region. The online computation is merely an evaluation of a piecewise linear function. The downfall of this method is that substantial memory is needed to store the offline solutions and the look up time increases with the dimensions of the state space and control horizon, which greatly limits the size of the system that can use this method. As stated previously in Section 1.3.1, Johansen utilized the mp-qp method for nonlinear models [43]. The approach by Wills et al. reduced the computation time by
computing some of the matrices in the cost function offline since they change infrequently and only stored half of the matrices entries since it is symmetric [98]. Then it implemented an active set method to solve the online portion, leading to a worst case computation time of $29.81 \mu$s. This is a fast time; however, it is for a simple linear model that has one control input and two states.

**Change to Optimization Method**

In the optimization literature, various researchers have developed optimization methods to decrease computation times. Rao et al. developed an interior point method that utilizes block factorization at each iteration to determine the search direction [78]. This method reduces time compared to quadratic programming methods when the prediction horizon is large, but if the prediction horizon is small, the results are similar. Diehl et al. modified a direct multiple shooting method to obtain fast computation times [22]. Ferreau et al. later used this method on a gasoline engine [28]. Tenny et al. employed a feasible sequential quadratic programming method that applied a scaled trust region [90].

**Control Parametrization**

Since the number of control inputs and control horizon effect the computation time, some researchers reduce the control input to a parametrization that allows for a reduced computation time. Canele designed a method that approximates the nonlinear function that relates the state and the control input; then online that nonlinear function is used at each sampling time to calculate the control input [16].

It should be noted that for fast MPC only a few of the results in the section above are for nonlinear systems [43],[22],[90] and [28].

**1.3.3 MPC Implemented on Mobile Robots**

Even though MPC started out in the process industry, many researchers have seen the value of MPC and used it on autonomous mobile robots. In general, the results listed
in this section were for applications with only a few inputs to the system and/or small control horizons, which yield smaller optimization times. In addition, most used MPC for trajectory tracking which, assumes the path the robot must traverse is already known and only the control inputs necessary to follow that path are determined. In this dissertation, the issue of trajectory generation is considered, which is a more difficult problem than trajectory tracking since it must determine the path for the robot along with the control inputs simultaneously. The results in this section also generally used simple linear models, while this dissertation research will use nonlinear models with multiple inputs and outputs.

**Legged Robots**

Researchers have implemented MPC on legged robots. Piovesan and Tanner used an artificial potential function along with randomized sampling of the control input in MPC to generate successful trajectories for a single legged robot [74]. The process generated a specific number of random samples, tested them, and selected the best one. Azevedo et al. utilized MPC with sequential quadratic programming in the optimization phase on a biped walking robot [8]. Dimitrov et al. used MPC to develop a walking robot’s motion [23].

**Unmanned Ground Vehicle**

There have even been cases where MPC was employed on unmanned ground vehicles (UGVs). Gu and Hu trained a wavelet neural network to represent a mobile robot’s kinematics and employed the network in a MPC framework [33]. Then the cost function was optimized using a gradient descent method. Wei et al. exploited direct collocation for optimization of a two wheel, differentially steered vehicle [97]. Peters and Iagnemma used MPC when comparing how three different models work on path following on non-flat terrain [72]. Kanjanawanishkul and Zell applied MPC to an omnidirectional mobile robot for trajectory tracking [46]. Raffo et al. applied MPC on an Ackerman steered vehicle at high speeds [77]. Lee and Yo used MPC on unmanned ground vehicles operating at high speeds [56].
Unmanned Aerial Vehicle

MPC has also been implemented on unmanned aerial vehicles (UAVs). The approach by Wills et al. coupled NMPC with the potential field method to allow the vehicle to maneuver around obstacles and other autonomous vehicles [83]. The use of a potential field is used to trade performance for computational efficiency. Singh and Fuller applied MPC to a helicopter type UAV using a control basis function [85]. Eklund et al. employed MPC on a fixed wing UAV [26]. Yadav et al. used a Grossberg Neural Network to determine the path of a UAV and Generalized Model Predictive Control to obtain a feasible trajectory [99].

Mixed Integer Linear Programming

When there are continuous and discrete variables in the optimization problem it calls for mixed integer programming (MIP). Schouwenaars et al. introduced a method using mixed integer linear programming (MILP) for ground vehicles that determined a trajectory to a final goal [80]. Each vehicle is modeled as a (linear) point mass with linear constraints. This method allows situations where the vehicle can get too close to an object, and collide. Later, Bellingham et al. added a cost estimate field called “cost-to-go” which applies a modified Dijkstra’s algorithm to a visibility graph that corrects for the local minimum problem [11]. In these methods, each vehicle is modeled as a simple linear point mass with constraints on the speed and turn rate; this may not be sufficient for predicting vehicle movement accurately.

1.3.4 MPC Implemented on UUVs

Since the primary focus of this research is implementing MPC on AUVs, this section specifically address cases where MPC is implemented on UUVs. There are several researchers who have simulated or implemented some form of MPC on UUVs in recent years. Oh and Oh proposed utilizing MPC for the homing and docking of an AUV [71]. A separate controller was implemented for each application. Lisboa et al. used MPC with a
Neural Network model to control the depth of an Underwater Robotic Vehicle (URV) for the purpose of hovering [60]. Sutton and Bitmead simulated MPC on an unmanned submarine with an objective of keeping the vehicle 50m above the sea floor with movement only in the X-Z plane [88]. Katebi and Grimble suggested a three layer AUV control architecture, where MPC is the middle layer for guidance and $H_\infty$ control is used in the bottom layer to control the actuators [47].

Some research has considered implementing MPC on AUV hardware. Miotto et al. applied $D^*$ to replan the path in case of obstacles, then B-Spline to smooth the generated lines to give the reference trajectory (guidance) [66]. MPC was then implemented between the guidance layer and autopilot to provide the control command for the Manta test vehicle. Naeem et al. makes use of a Line of Sight (LOS) technique to generate the reference trajectory. MPC is employed in determining the control input needed to keep sufficient heading for following the trajectory [68]. This technique is implemented in real-time on the Hammerhead AUV utilizing a genetic algorithm for optimization.

Other than Katebi and Grimble [47], the previous methods use MPC to find the control effort necessary to follow a predetermined trajectory (i.e. trajectory tracking) for the UUV. In contrast, this research utilizes NMPC to determine the optimal path and find the control effort simultaneously. This research goes one step further than those listed above. Unlike most of the other methods that linearize the model, a nonlinear model is employed. In addition, this research determines a path for a multi-input multi-output model where as some of the previous researchers only explored controlling the depth or heading separately. Other than Miotto et al. [66], the prior mentioned research assumed an ideal situation of a clear area with no obstacles. On the contrary, this research deals with obstacles, which yield additional constraints for the optimization problem.
1.3.5 UUV Obstacle Avoidance

UUVs are usually the last vehicles to be considered when obstacle avoidance and path planning methods are introduced. When implementing path planning methods in the ocean, one has to consider obstacles such as sea buoys, reefs, and unusual underwater terrains. For this dissertation obstacle avoidance refers to methods that avoid an obstacle locally without regard to the global map. In contrast, a path planning method utilizes a global map to determine a path from a start to goal position while avoiding any obstacles that are present.

There are various techniques used to accomplish obstacle avoidance. Demuth and Springsteen [21] utilized neural networks so that when an obstacle is close, the AUV makes a much larger heading change than when the obstacle is further away. Liu et al. used fuzzy logic for the local planner [62]. The input to the planner is the distance between the obstacle and vehicles, and the output is the angle or distance the vehicle should move. Fondrea employed a weighted function that comes from fuzzy logic concepts along with a line of sight controller to adjust heading error of an AUV [30]. In a similar manner, Healey used a weighted function for obstacle avoidance for the pitch of an AUV [35]. Horner et al. avoided obstacle on the bottom of the sea floor by applying an additive Gaussian function to the original fixed altitude [40].

There are other researchers that used multiple subsystems to accomplish obstacle avoidance. Hyland created an obstacle avoidance system that is broken into two parts [41]. The first part determined where the obstacles are located, and the second attempted to avoid an obstacle by introducing a standoff distance and a maneuverability circle. The obstacle avoidance system by Moitie and Seube has four subsystems [67]. They determine the surroundings, select waypoints to get to the goal (long term), and correct the long term subsystem if there is a change (short term); a sonar program maps the environment. They used the kinematic model to determine all viable trajectories to the goal, and the control inputs that get them there. The obstacle avoidance system by Rhee et al. [79] had three parts:
1) the avoidance component calculated a target point to avoid an obstacle; 2) the guidance component determined the reference trajectory to the target point; and 3) the control component used a sliding mode controller that generates control signals to follow the reference commands. Kwon et al. used an outer loop that adopted the concept of virtual force field to avoid obstacles and an inner loop that employed a proportional integral derivative (PID) controller for course keeping [53]. They used fuzzy logic to determine the gains in the outer loop. Kanakakis et al. had a three level collision avoidance system that incorporated fuzzy logic based navigation [45]. Level one determined potential collisions; level two gave a target point to avoid a potential collision; and level three controlled the vehicle to reach the target point.

All of the above methods provide obstacle avoidance. However, they do not attempt to optimize the vehicle path considering a global map.

1.3.6 UUV Path Planning

There are several techniques that have previously been used for UUV path planning. For the most part the path planning methods can be classified as either potential field methods, gradient based optimization, global optimization, graph search algorithms, sampling based methods, and logic or reason based methods. Some methods are a combination of these techniques.

Potential Field Methods

In general, the main idea behind potential field methods is that the robot is considered a point moving in a potential field that treats the goal as an attraction (positive) and obstacles as a repulsion (negative), which leads to a path. The method is not computationally expensive, but it can become stuck in a local minimum instead of finding a path to the goal. Warren’s method started out with a straight line from the start to the goal and deforms this line using an artificial potential field until it is safe for the vehicle to move
around the obstacle [96]. Antonelli et al. implemented a virtual force field (VFF) with geometric consideration to prevent the vehicle from getting stuck in corridors [6]. Barisic et al. implemented a modified virtual potential field that introduces virtual viscose friction to overcome oscillation and a hybrid logic scheme that places a temporary target point to overcome the local minimum problem [10]. Healey et al. implemented a potential field for path generation that is subject to the vehicles mobility. A Gaussian avoidance function is exerted to provide a smooth path [36].

Gradient Based Methods

Gradient based optimization is a method that searches for the minimum by following a search direction computed using the gradient of the cost function. However, a downfall of this method is that it can only guarantee a local minimum with nonlinear functions. Wang and Lane used a modified sequential quadratic programming method (SQP) where the initial configuration was the starting point of SQP and the goal configuration was a unique global minimum of the cost function [95]. The free space and obstacles are represented by inequality constraints. Kruger et al. applied a nonlinear gradient based optimization to determine a path when bidirectional currents are present [52].

Global Optimization Methods

Global optimization techniques are more prone to determine the global minimum of a cost function. However, this comes at the cost of an increase in computation time. Also, this class of methods is not guaranteed to converge to the global minimum in finite time, creating a suboptimal solution.

Sughira and Yuh utilized a genetic algorithm (GA) to determine a path by coding the 3-D space as binary strings that represented the direction and distance [87]. Hong-Jain et al. gridded the output space and used a GA for which an individual in the population represents a path in the vehicles output space [39]. Alvarez et al. employed a GA to
determine a path in the presence of varying ocean currents [5]. Chang et al. utilized a GA to avoid moving obstacles, where the individuals represented the path and the chromosomes are path segments [18]. Zhang combined the octree method with a genetic algorithm [102]. The octree divided the output space of the vehicle into regions and a GA determined the path. Wang et al. introduced an adaptive genetic algorithm that used a GA to identify the global path, and the path is adapted by a local path planner, which incorporated sonar information [94].

Khanmohammdi et al. compared a conjugate gradient penalty method, particle swarm optimization, and genetic algorithm in AUV path planning [86]. The dynamic model is used as an optimization problem constraint. Liu and Dai employed ant colony optimization to determine the AUV path [61].

**Graph Search Algorithms**

Graph search algorithms explore a graph (or cells) consisting of nodes and edges in the free space. It uses various criteria to expand nodes to traverse from a start node to a goal node. These methods do not have the local minimum problem as some other methods. It also always finds feasible paths if one exists. However, the computation cost increases as higher dimensional problems are introduced.

Carroll et al. adopted $A^*$ for path planning considered variable vehicle speeds, ocean currents, and time dependent exclusion zones [17]. Khorrami and Krishnamurthy used a method called game theoretic optimal deformable zone with inertia and local approach (GODZILA) [50]. $A^*$, which determines the trajectory for a mission, and GODZILA, which is a reactive obstacle avoidance, are combined to create a hierarchal path planning method. Shuzong assumed the vehicle can move purely horizontal or purely vertical [84]. He implemented a visibility graph with static and dynamic cutoff features that reduce the combinatorial expansion of the search tree. A heuristic search algorithm is then implemented. Hyland and Fox compared $A^*$ and breadth first dynamic search methods beginning
at the goal [42]. The area traversed is gridded into safe and unsafe regions. There was a node contained in each rectangular grid that gave the status (safe or unsafe), cost and direction to next node. Garau et al. did a comparison of $A^*$ and Dijkstra for path planning in the presences of currents [32].

Eichhorn used the concept of dipoles [25]. The circle obstacles had a positive point charge and the goal was considered to have a negative point charge, which formed a dipole and its flux lines lead from positive to negative charge. The edges are given weights and the Dijkstra search algorithm is used. The entire system is composed of two levels; the upper level determine the route for a vehicle to follow, and the lower level is a reactive controller that guides vehicles when there are new obstacles. Sequeira and Ribeiro proposed a two level hybrid planner [82]. The high level implemented Dijkstra’s algorithm to evaluate nodes and the low level used a modified artificial potential field that gave the exact path between two consecutive points. Arinaga et al. developed a two step motion planner that applies Dijkstra’s algorithm to find the optimal path and a bidirectional local motion planner that produced a smooth path by simulating a real vehicle at one end and a virtual vehicle at the other end [7].

Sampling Based Methods

Sampling based algorithms obtain samples of the output space and then constructs a roadmap that connects the start and the goal. Obviously, a challenge of this method is it’s success is dependent on how well the space is sampled. Petres incorporated fast marching method with $A^*$ to produce $FM^*$ which accounts for the vehicles turning radius [73]. FM is a level set method using a first order numerical approximation of the Eikonal equation. Tan et al. combined a manoeuvre automation, which is a type of finite state machine, and rapidly exploring random tree to determine the trajectory, and then applied a state dependent Ricatti equation to follow the trajectory [89]. Bo et al. implemented a fast marching method to determine the trajectory subject to kinematic constraints and a feedback controller to follow the trajectory [13].
Logic Based Methods

Logic or reason based methods apply logic to handle reasoning. These methods analyze approximate events instead of precise events, which could lead to nonoptimal trajectories. Vasudevan and Ganesan used specific instances of past experiences in a case based reasoning planner [93]. Previous routes were modified utilizing a repair rule to plan for new routes. The best past route is selected by a fuzzy set membership. Lee et al. developed a new heuristic search method based on fuzzy relational products [57]. The 3-D environment was divided into grids and the obstacle’s size were grown to include AUV turn rate and stop distance. Once this is done, four heuristic conditions are adopted to determine the current node’s successor until the goal is reached.

Other Methods

The following two methods do not correspond with either of the prior categories. These researchers have incorporated a model in the motion planning for an AUV. Kawano utilized a 2D probabilistic planner that included a four part system [49]: 1) the offline Markov Decision Process (MDP) module did the motion planning offline; 2) the replanning module determined a path when there are new obstacles; 3) the real-time path tracking module identifies actions that needs to be taken; 4) the feedback control module controlled the vehicles velocity to the target velocity. MDP is computationally expensive, so in order to acquire real-time results all the possible paths are computed offline considering the AUVs kinematics and dynamics. Replanning is accomplished by incorporating a rough target tracking control of the paths determined offline and the geometrical properties of obstacles. This ensures the AUV reaches the goal without getting stuck in a concave obstacle. However, it does not guarantee the path determined in the replanning module is necessarily capable of being tracked by the AUV and avoiding obstacles, so the AUV is controlled to avoid collision.
Yakimenko et al. used the direct method of calculus of variations to generate trajectories for the vehicle to traverse [100]. There are four main blocks: the first generated a candidate trajectory that satisfied boundary conditions and position, velocity and acceleration constraints, the second employed inverse dynamics that determined the states and control inputs necessary to follow it, the third computed states along the reference trajectory over a fixed set of $N$ points, and the fourth optimized the performance index. The method ensures a smooth path that can be found in real-time, but giving up some of the optimality. Also, because the inverse dynamics are used directly, an accurate model is needed.

1.4  Research Goals

This dissertation’s goal is to add to the AUV path planning methods discussed in Section 1.3.6. As discussed in that section, there are currently only a few methods that consider the vehicle’s model during the planning process. However, by employing SBMPC this dissertation seeks to develop a fast motion planner for AUVs. SBMPC draws from traditional MPC and two types of previously discussed path planning methods: sampling based methods and graph search algorithms. Sampling based methods are limited by how well the space is sampled; however, SBMPC’s approach is different from the sampling based methods discussed in Section 1.3.6. SBMPC samples the input space as opposed to the output space, yielding a more efficient search method. Graph search algorithms typically do not include a model of the vehicle in the path planning process. However, unlike the AUV path planning algorithms in Section 1.3.6 that rely on graph search methods, SBMPC incorporates a model of the vehicle to ensure feasible paths.

Similar to the methods [49] and [100], discussed at the end of Section 1.3.6 under “Other Methods”, this research incorporates the model in motion planning. On the other hand, unlike [49], which considers only 2D motion, this research generates a 3D path for the AUV to follow. Also, like [100], optimality is sacrificed to ensure fast computation, but unlike [100] a full dynamic model is not necessary for the method to work.
1.5 Contributions

This dissertation makes the following contributions:

1. **Section 3.5** Conceptual comparison of Sampling Based Model Predictive Control and traditional Model Predictive Control.

2. **Section 4.1** Performance comparison of Sampling Based Model Predictive Control method and traditional Model Predictive Control using a 2D kinematic model of an AUV.

3. **Sections 4.2 - 4.3** Application of Sampling Based Model Predictive Control to an Autonomous Underwater Vehicle using a 3D kinematic model, decoupled dynamic model and full dynamic model.

4. **Section 4.3** Determination of AUV maneuvers for which it is important to consider the vehicle dynamics.

1.6 Dissertation Organization

The chapters are organized as follows:

**Chapter 2** presents the kinematic model for an AUV and the dynamic model for the Naval Postgraduate School (NPS) AUV II.

**Chapter 3** provides an overview of traditional model predictive control (MPC) then describes concerns with MPC. This chapter discusses how SBMPC evolved from concepts in control, robotics and artificial intelligence and describes SBMPC. Lastly, an in depth comparison of SBMPC and traditional MPC will be given.

**Chapter 4** presents simulations of an AUV incorporating both MPC and SBMPC to traverse a cluttered environment, utilizing plans based on the 2D kinematic model. Then presents simulation results of the AUV in various scenarios that identify the benefits of SBMPC. Two varying sea floor bottoms are considered to exhibit the importance of incorporating the model in AUV path planning. Next, a common local minima problem will be simulated. Then the chapter demonstrates how the AUV operates in a cluttered 3D space. Last, it evaluates how SBMPC tuning parameters effect CPU time.

**Chapter 5** concludes the previous chapters and presents recommendations for future work.
CHAPTER 2
AUTONOMOUS UNDERWATER VEHICLE MODELS

In this chapter, Section 2.1 first explains the differences in various autonomous vehicles. Section 2.2 then provides a description of the AUV kinematic model. Next Section 2.3 discusses a general six degree of freedom dynamic model for AUVs, specifically the Naval Postgraduate School (NPS) AUV II. A decoupled dynamic model of the NPS AUV II is then described in Section 2.4. Finally, Section 2.5 discusses why including the kinematics and dynamics is necessary in motion planning.

2.1 General

When modeling a vehicle, it is essential to accurately consider the physical laws that describe how the vehicle moves in its environment [14]. There are four main types of autonomous vehicles: unmanned ground vehicles (UGVs), unmanned aerial vehicles (UAVs), unmanned surface vehicles (USVs) and unmanned underwater vehicles (UUVs). Each requires different constraints because each one traverses a different medium.

UGVs typically operate in a 2-D horizontal plane. Generally, vertical motion is not highly coupled with horizontal motion. In addition, the vehicles are constrained by surface contact that generates normal and frictional forces between the UGV and terrain. Most UGVs are considered nonholonomic. In contrast, UAVs are not constrained to ground contact and are generally holonomic in nature. An UAV moves in the air, which produces
aerodynamic forces that effect the motion. Additionally, the vertical and horizontal motion are coupled. When considering a fixed wing UAV, the forward velocity of the vehicle is so high compared to the low density of air that the “added mass” effects are negligible. USVs are also holonomic, but they move through two different mediums: water and air, which bring additional difficulties. Generally, forces from the water (hydrodynamic) are stronger than forces from the air (aerodynamic); however, both must be considered when determining the model. Also, vertical motion that keeps the USV on the surface of the water is effected by vehicle weight, water displacement, and the center of buoyancy of the vehicle.

Similarly, the complicated nature of an UUV makes it even more difficult to create a dynamic model. The forces and moments have cross-coupled longitudinal, lateral, and vertical motion. There is a similar nonlinear nature of the UUV and UAV. However in contrast to the UAV, the environment (water) that surrounds the vehicle is massive and accelerates along with the vehicle to create an “added mass.” Any change in the water can effect the vehicle’s motion. The problems with detecting and determining the UUV hydrodynamic response makes it harder to model than the prior three vehicles.

2.2 Vehicle Kinematics

An AUV model approximates the vehicle’s movement as a response to specified inputs. The kinematic model is a simple geometric mathematical model that does not take into account the forces that cause the motion. Before the kinematic model is described, it is important to define the coordinate frames. There are two frames shown in Figure 2.1. The first is the world frame $W$ considered at the ocean surface, where North corresponds to the $X$-axis, East corresponds to the $Y$-axis, and the positive $Z$-axis is pointing down to complete the right hand rule. The second frame is referred to as the local body-fixed frame $B$ and moves with the AUV body, where the origin $O$ is usually at the center of buoyancy $CB$. In Figure 2.1 observe that the positive $x$-axis is pointing through the nose along the
direction of forward motion, while the $z$-axis is pointing down through the middle of the vehicle and the $y$-axis completes the right hand rule.

The notation in Figure 2.1 was designated by the Society of Naval Architects and Marine Engineers (SNAME) in 1950 and is typically still used today [31]. The motion in the $x$ direction, $y$ direction and $z$ direction are respectively called surge, sway and heave. Then there is rotation about the $x$, $y$, and $z$ axes, respectively called roll, pitch, and yaw rates, these velocities are depicted by the vector,

$$
\begin{bmatrix}
\dot{u} \\
\dot{v} \\
\dot{w} \\
\dot{p} \\
\dot{q} \\
\dot{r}
\end{bmatrix},
$$

(2.1)

where $u, v, w$ are linear velocities in the local body fixed frame along the $x, y, z$ axes, respectively and $p, q, r$ are the angular velocities in the local body fixed frame along the
$x, y, z$ axes, respectively. If $v_0$ and $\omega_0$ are defined as linear velocities and angular velocities, then (2.1) becomes

$$\dot{q} = \begin{bmatrix} v_0 \\ \omega_0 \end{bmatrix}. \quad (2.2)$$

The AUV posture vector $[x]$ can be defined by six coordinates, three representing the position,

$$\begin{bmatrix} x_1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.3)$$

and three corresponding to the orientation,

$$\begin{bmatrix} x_2 \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad (2.4)$$
all with respect to the world frame. In particular,

\[
\begin{bmatrix}
  x \\
y \\
z \\
\phi \\
\theta \\
\psi
\end{bmatrix}
\]

(2.5)

The Euler angles are shown in Fig. 2.2 from three different viewpoints. Note the reference is the previous posture. The kinematic model plainly demonstrates how the body frames linear and angular velocities relate to the world frame linear velocities and Euler rates. This relationship can be expressed as

\[
\dot{x}_1 = J_1(\phi, \theta, \psi)v_0,
\]

(2.6)

where

\[
J_1(\phi, \theta, \psi) =
\begin{bmatrix}
  \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi - \sin\phi\sin\psi \\
  \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi - \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\
  \sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta
\end{bmatrix}
\]

(2.7)

and

\[
\dot{x}_2 = J_2(\phi, \theta, \psi)\omega_0,
\]

(2.8)

where

\[
J_2(\phi, \theta, \psi) =
\begin{bmatrix}
  1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\
  0 & \cos\phi & -\sin\phi \\
  0 & \sin\phi\sec\theta & \cos\phi\sec\theta
\end{bmatrix}
\]

(2.9)

Note that (2.6) and (2.8) may be expressed compactly as

\[
\begin{bmatrix}
  \dot{x}_1 \\
  \dot{x}_2
\end{bmatrix} =
\begin{bmatrix}
  J_1(x_2) & 0 \\
  0 & J_2(x_2)
\end{bmatrix}
\begin{bmatrix}
  v_0 \\
  \omega_0
\end{bmatrix}
\]

(2.10)

or equivalently,

\[
\dot{x} = J(x)\dot{q}.
\]

(2.11)

Consequently, there are six inputs and six states in the kinematic model. Note there is a singularity in (2.9) at $\theta = \pm 90^\circ$.
2.3 Vehicle Dynamics

An AUV equation of motion can be described using Newton’s second law. The laws are calculated in the body fixed coordinate system, so the forces and moments stay constant to changes of the AUV’s orientation relative to the world reference frame [31]. The dynamics in the compact form are given by

\[ M\ddot{q} + C(q)\dot{q} + D(q)\dot{q} + g(x) = \tau \]  

(2.12)

with kinematics (2.11), where \( M = M_{RB} + M_A \) is the inertia matrix with \( M_{RB} \) the rigid body inertia matrix and \( M_A \) the added mass matrix; \( C = C_{RB} + C_A \) is a matrix of Coriolis and centrifugal terms with \( C_{RB} \) the Coriolis and centrifugal terms due to the rigid body and \( C_A \) the Coriolis and centrifugal terms due to hydrodynamic mass; \( D = D_B + D_L + D_Q + D_v \) is a matrix of dissipative (damping) terms with \( D_B \) a matrix of damping terms due to potential damping, \( D_L \) a matrix of linear damping terms due to skin friction and drag, \( D_Q \) a matrix of quadratic damping terms due to skin friction and drag, and \( D_v \) a matrix of damping terms due to viscous damping; \( g \) is vector of restoring (gravitational and buoyancy) forces and moments; and \( \tau \) represents the external forces and moments acting on the vehicle.

Assuming constant mass, the dynamic equations of motion can be written as

\[
m(\dot{v}_0 + \omega_0 \times v_0 + \dot{\omega}_0 \times r_G + \omega_0 \times (\omega_0 \times r_G)) = f_0, \tag{2.13}
\]

\[
m r_G \times \dot{v}_0 + m r_G \times (\omega_0 \times v_0) + I_0 \dot{\omega}_0 + \omega_0 \times (I_0 \omega_0) = m_0, \tag{2.14}
\]

where

\[
f_0 = [X_0 \ Y_0 \ Z_0]^T,
\]

\[
m_0 = [K_0 \ M_0 \ N_0]^T,
\]

\[
r_G = [x_G \ y_G \ z_G]^T.
\]

The vectors \( f_0 \) and \( m_0 \) are the external forces and external moments acting on the AUV. Each term describes the sum of the hydrodynamic forces, viscous damping, and propulsion
forces. The vector $r_G$ is the distance from the origin to the center of gravity. Equations (2.13) and (2.14) can be separated into the six degrees of freedom of the AUV:

\[
m[\ddot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - r) + z_G(pr + q)] = X_0, \quad (2.15)
\]
\[
m[\ddot{v} + ur - wp + x_G(pq + r) - y_G(p^2 - r^2) + z_G(qr - p)] = Y_0, \quad (2.16)
\]
\[
m[\ddot{w} - uq + vp - x_G(pr - q) + y_G(qr + p) - z_G(p^2 + q^2)] = Z_0, \quad (2.17)
\]
\[
I_x \ddot{p} + (I_x - I_y)qr + I_{xy}(pr - q) - I_{xz}(q^2 - r^2) - I_{xx}(pq + r)
\]
\[
+ m[y_G(\dot{w} - uq + vp) - z_G(\dot{v} + ur - wp)] = K_0, \quad (2.18)
\]
\[
I_y \ddot{q} + (I_x - I_z)prI - xz(qr - \dot{p}) + I_{yz}(qp - r) + I_{xx}(p^2 - r^2)
\]
\[
- m[x_G(\dot{w} - uq + vp) - z_G(\dot{u} - vr + wq)] = M_0, \quad (2.19)
\]
\[
I_z \ddot{r} + (I_y - I_x)pq - I_{xy}(p^2 - q^2) - I_{xz}(pr + q) + I_{xx}(qr - \dot{p})
\]
\[
+ m[x_G(\dot{v} + ur - wp) - y_G(\dot{u} - vr + wq)] = N_0. \quad (2.20)
\]

The right hand sides of (2.15)-(2.20) can vary depending on the vehicle. The majority of this research is focused on the Navy Postgraduate School (NPS) AUV II [37]. Information about the vehicle’s hardware and software can be found in [92]. The NPS AUV II has six control inputs: the rudder $\delta_r$, starboard bow plane $\delta_{bs}$, port bow plane $\delta_{bp}$, stern plane $\delta_s$, propeller rotational rate $n$, and buoyancy adjustment $B$. In this case the dynamic equations (2.15)-(2.20) become the following

**Surge Equation of Motion**

\[
m[\ddot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - r) + z_G(pr + q)] = \frac{\rho}{2} L^4 [X_{pp} p^2 + X_{qq} q^2 + X_{rr} r^2 + X_{pr} pr]
\]
\[
+ \frac{\rho}{2} L^3 [X_{u\dot{u}} + X_{wq} wq + X_{vp} vp + X_{vr} vr + uq(X_{q\delta_s} \delta_s + X_{q\delta_{bs/2} \delta_{bs} + X_{q\delta_{bp/2} \delta_{bp}}) + X_{r\delta_r} ur \delta_r]
\]
\[
+ \frac{\rho}{2} L^2 [X_{vu} v^2 + X_{vu} w^2 + X_{vq} uq \delta_r + uw(X_{q\delta_s} \delta_s + X_{q\delta_{bs/2} \delta_{bs} + X_{q\delta_{bp/2} \delta_{bp}}) + u^2(X_{\delta_s \delta_r} \delta_s^2
\]
\[
+ X_{\delta_s \delta_{bs/2} \delta_{bs}^2 + X_{\delta_s \delta_{bp/2} \delta_{bp}^2} - (W - B) \sin \theta + \frac{\rho}{2} L^3 X_{q\delta_s} uq \delta_s (n) + \frac{\rho}{2} L^2 [X_{u\delta_{sn}} uw \delta_{sn}
\]
\[
+ X_{\delta_s \delta_{sn} \delta_{sn} u^2 \delta_s^2] \cdot \epsilon (n) + \frac{\rho}{2} L^2 u^2 X_{prop} \quad (2.21)
\]
Sway Equation of Motion

\[ m[\dot{v} + ur - wp + x_G(pq + \dot{r}) - y_G(p^2 + r^2) + z_G(qr - \dot{p})] = \frac{\rho}{2} L^4 [Y_p \dot{p} + Y_{tr} \dot{r} + Y_{pq}pq + Y_{qr}qr] \]
\[ + \frac{\rho}{2} L^3 [Y_v \dot{v} + Y_{p}up + Y_r ur + Y_{vp}vq + Y_{wp}wp + Y_{wr}wr] + \frac{\rho}{2} L^2 [Y_s uv + Y_vwv w + Y_{s\delta} u^2 \delta r] \]
\[ - \frac{\rho}{2} \int_{x_{tail}}^{x_{nose}} [C_{dy} h(x)(v + x)² + C_{dz} b(x)(w - x)²] \cdot \frac{(v + x)r}{U_{cf}(x)} dx + (W - B)\cos\theta \sin\phi \]
(2.22)

Heave Equation of Motion

\[ m[\dot{w} - uq + vp - x_G(pr - \dot{q}) - y_G(qr + \dot{p}) - z_G(p^2 + r^2)] = \frac{\rho}{2} L^4 [Z_q \dot{q} + Z_{pp} p^2 + Z_{pr} pr + Z_{rr} r^2] \]
\[ + \frac{\rho}{2} L^3 [Z_v \dot{v} + Z_q uq + Z_{vp} vq + Z_{vr} vr] + \frac{\rho}{2} L^2 [Z_w uw + Z_{vw} v^2 + u^2 (Z_{s\delta} \delta_s + Z_{sb}/2 \delta_b + Z_{hb}/2 \delta_b)] \]
\[ + \frac{\rho}{2} \int_{x_{tail}}^{x_{nose}} [C_{dy} h(x)(v + x)² + C_{dz} b(x)(w - x)²] \frac{(w - x)q}{U_{cf}(x)} dx + (W - B)\cos\theta \cos\phi \]
\[ + \frac{\rho}{2} L^3 Z_{mn} uq\epsilon(n) + \frac{\rho}{2} L^2 [Z_{wn} uw + Z_{sn} u^2 \delta_s] \epsilon(n) \]
(2.23)

Roll Equation of Motion

\[ I_x \dot{p} + (I_z - I_y) qr + I_{xy} (pr - \dot{q}) - I_{yz} (q^2 - r^2) - I_{zx} (pq + \dot{r}) + m[y_G(\dot{w} - uq + vp) \]
\[ - z_G(\dot{v} + ur - wp)] = \frac{\rho}{2} L^5 [K_{p} \dot{p} + K_{tr} \dot{r} + K_{pq}pq + K_{qr}qr] + \frac{\rho}{2} L^4 [K_v \dot{v} + K_{p}up + K_r ur + K_{vq} vq \]
\[ + K_{wp} wp + K_{wr} wr] + \frac{\rho}{2} L^3 [K_{uv} uv + K_{vw} vv + u^2 (K_{s\delta} \delta_s + K_{sb}/2 \delta_b + K_{hb}/2 \delta_b)] + (y_G W - y_B B) \cos\theta \cos\phi \]
\[ - (z_G W - z_B B) \cos\theta \sin\phi + \frac{\rho}{2} L^4 K_{mn} uq\epsilon(n) + \frac{\rho}{2} L^3 u^2 K_{prop} \]
(2.24)

Pitch Equation of Motion

\[ I_y \dot{q} + (I_x - I_z) pr - I_{xy} (qr - \dot{p}) + I_{yz} (pq - \dot{r}) + I_{zx} (p^2 - r^2) - m[x_G(\dot{w} - uq + vp) \]
\[ - z_G(\dot{u} + vr + wq)] = \frac{\rho}{2} L^5 [M_{q} \dot{q} + M_{pp} p^2 + M_{pr} pr + M_{rr} r^2] + \frac{\rho}{2} L^4 [M_v \dot{v} + M_{uq} uq + M_{vp} vp \]
\[ + M_{vr} vr] + \frac{\rho}{2} L^3 [M_{uw} uw + M_{vw} vv + u^2 (M_{s\delta} \delta_s + M_{sb}/2 \delta_b + M_{hb}/2 \delta_b)] - \frac{\rho}{2} \int_{x_{tail}}^{x_{nose}} [C_{dy} h(x) \]
\[ (v + x)r² + C_{dz} b(x)(w - x)²] \frac{(w + x)q}{U_{cf}(x)} dx - (x_G W - x_B B) \cos\theta \cos\phi - (z_G W - z_B B) \sin\theta \]
\[ + \frac{\rho}{2} L^4 M_{mn} uq\epsilon(n) + \frac{\rho}{2} L^3 [M_{wn} uw + M_{sn} u^2 \delta_s] \epsilon(n) \]
(2.25)
Yaw Equation of Motion

\[ I_z \dot{r} + (I_y - I_x)pq - I_{xy}(p^2 - q^2) - I_{yz}(qr - \dot{q}) + I_{xz}(qr - \dot{p}) + m[x_G(\dot{v} + ur - wp) \]
\[-y_G(\dot{u} - vr + wq)] = \frac{\rho}{2} L^5 [N_p \dot{p} + N_r \dot{r} + N_{pq}pq + N_{qr}qr] + \frac{\rho}{2} L^4 [N_p \dot{v} + N_p up + N_r ur + N_{eq}vq + N_{wp}wp + N_{wr}wr] + \frac{\rho}{2} L^3 [N_v uv + N_v vw + N_{\delta r} u^2 \delta r] - \frac{\rho}{2} \int_{x_{nose}}^{x_{tail}} [C_{dy} h(x)(v + xr)^2 + C_{dz} b(x)(w - xq)^2] \frac{(v + xr)}{U_{cf}(x)} x dx + (x_G W - x_B B) \cos \theta \sin \phi + (y_G W - y_B B) \sin \theta \]
\+ \frac{\rho}{2} L^3 u^2 N_{prop}, \quad (2.26)

where

\[ U_{cf}(x) = \sqrt{(v + xr)^2 + (w - xq)^2}, \]
\[ X_{prop} = C_{d0} (|\eta| - 1), \]
\[ \eta = 0.012 n/u, \]
\[ C_{d0} = 0.00385, \]
\[ \epsilon(n) = -1 + \frac{\text{sign}(n)}{\text{sign}(u)} \frac{\sqrt{C_t + 1} - 1}{\sqrt{C_{t1} + 1} - 1}, \]
\[ C_t = |0.008 L^2 \frac{\eta |\eta|}{2}|, \]
\[ C_{t1} = |0.008 \frac{L^2}{2}|. \]

The hydrodynamic coefficients in (2.21)-(2.26) depend on the tow test applied to the NPS AUV II. Appendix A contains the values of the vehicle’s dynamic parameters.

2.4 Vehicle Decoupled Model

In order to work with a simpler dynamic model, especially for control design, it has been the practice to decouple the highly coupled AUV dynamic model, which requires certain assumptions. It is assumed the AUV has noninteracting subsystems of speed, steering, and diving control modes [37]. Note the roll mode is assumed to be passive. The
assumptions allow each subsystem to become a single input, multi-state system as given below.

**Speed**

\[
\dot{u}(t) = -\alpha u|u| + (\alpha\beta)n|n|
\]  

(2.27)

where

\[
\alpha = \frac{\rho L^2 C_d}{2m + \rho L^3 X_\dot{u}},
\]

\[
\beta = \frac{u_0^2}{n_0^2},
\]

\[
C_d = 0.0034,
\]

and the input is the propeller rotational rate and the state is \( u \).

**Steering**

\[
\begin{bmatrix}
\dot{v} \\
\dot{r} \\
\dot{\psi}
\end{bmatrix} = \begin{bmatrix}
m_1 & m_2 \\
m_3 & m_4
\end{bmatrix}^{-1} \begin{bmatrix}
Y_1 & Y_2 \\
N_1 & N_2
\end{bmatrix} \begin{bmatrix}
0 \\
0
\end{bmatrix} \begin{bmatrix}
v \\
r \\
\psi
\end{bmatrix} + \begin{bmatrix}
m_1 & m_2 \\
m_3 & m_4
\end{bmatrix}^{-1} \begin{bmatrix}
Y_3 \\
N_3
\end{bmatrix} \delta_r
\]  

(2.28)

where

\[
Y_1 = \frac{\rho}{2} L^2 u Y_v,
\]

\[
Y_2 = \frac{\rho}{2} L^3 u Y_r,
\]

\[
Y_3 = \frac{\rho}{2} L^2 u^2 Y_{\delta_r},
\]

\[
N_1 = \frac{\rho}{2} L^3 u N_v,
\]

\[
N_2 = \frac{\rho}{2} L^4 u N_r,
\]

\[
N_3 = \frac{\rho}{2} L^3 u^2 N_{\delta_r},
\]

29
\[ m_1 = m - \frac{\rho}{2} L^3 Y_\dot{v}, \]
\[ m_2 = -\frac{\rho}{2} L^4 Y_\dot{r}, \]
\[ m_3 = -\frac{\rho}{2} L^4 N_\dot{v}, \]
\[ m_4 = I_z - \frac{\rho}{2} L^5 N_\dot{r}, \]

the input is the rudder angle, and the states are \( v, r, \psi \).

**Diving**

\[
\begin{bmatrix}
\dot{q} \\
\dot{\theta} \\
\dot{Z}
\end{bmatrix} =
\begin{bmatrix}
-0.7 & -0.03 & 0 \\
1 & 0 & 0 \\
0 & -u & 0
\end{bmatrix}
\begin{bmatrix}
q \\
\theta \\
Z
\end{bmatrix} +
\begin{bmatrix}
-0.035 \\
0 \\
0
\end{bmatrix} \delta_s,
\]

(2.29)

where the input is the stern angle and the states are \( q, \theta, z \). The decoupled model (2.28) - (2.29) allows for a more simplified dynamic model than (2.21) - (2.26). However, it does not incorporate all the dynamic phenomena of the AUV.

### 2.5 Inclusion of the Kinematic and Dynamic Models in AUV Motion Planning

The main rationale for developing an AUV model is to provide the opportunity to predict the vehicle’s motion under a variety of system inputs. This is important since AUVs must maneuver in an underwater area that contain subsea obstacles, irregular seabeds, and marine growth that the vehicle’s path must void. There are three types of models detailed in Sections 2.2- 2.4 that can aide in predicting paths: the kinematic model, the full dynamic model and the decoupled dynamic model. The kinematic model enforces the vehicle kinematic constraints (i.e. turn radius) that limit the output space (configuration space). In planning tasks for scenarios where the AUV operates in calm waters and the vehicle is not pushed to the extreme in free space the kinematic model may provide feasible motion plans.

The dynamic model incorporates more useful information about the vehicle’s motion than the kinematic model. It describes the feasible control inputs, velocities, acceleration
and various underwater phenomena. Therefore missions that will require the vehicle to perform close to its limits will need the dynamic model. Some examples will be given later in this section.

Because of the complexity of the full dynamic model and large effort needed to integrate the dynamic model, the decoupled model which is a simplified dynamic model that assumes the vehicle’s subsystems are not interacting can be used to yield reduced computations. However, it accurately predicts motions only when the AUV is not involved in overly complex maneuvers. The decoupled dynamic model provides a better prediction of the trajectory than the kinematic model, but is not as reliable as the full dynamic model.

Any AUV motion planner should consider the physical limitations of the vehicle given by the inclusion of the kinematic and dynamic model. There are several examples of when the full dynamic model is necessary. Previous research has revealed that when the AUV is underactuated, the dynamic equation should be considered in addition to the kinematic equation in determining the controller [101] and motion plans [49]. This is because, similar to UGVs, when an AUV is underactuated, it introduces nonholonomic constraints. However, unlike the UGV, the AUV has increased complexity since it has to move through water, which introduces hydrodynamic forces and moments that must be considered in the planning. An UUV typically has velocities along each of its axes, so motion planning and control methods that linearize the dynamic model about a single forward velocity are not as effective as they are with UAVs.

Since a simulation of a dynamic model takes into account vehicle inertias, and the maximum forces and moments due to the actuators, it can capture phenomena that cannot be seen using a kinematic model. In particular, a dynamic simulation can provide reliable predictions of the true velocity when actuator saturation occurs. Since a kinematic model does not have this predictive power, it can lead a planner to command unachievable velocities.

The dynamic model is also valuable in motion planning because it can model the efficiency of the control surface (i.e. rudders and stern planes), which are dependent on the
speed of the AUV. This effect is due to the lift force being proportional to the squared velocity of the vehicle. When determining the vehicle’s path, it is important to be able to predict if the control surfaces are not working to their full capability. Fig. 2.3a shows the AUV path simulated by an open loop decoupled model when the stern plane is held constant at $-15^\circ$ and the propeller speed is low. Since it does not consider the full dynamics of the vehicle, it predicts a path that descends straight down. Fig. 2.3b depicts the path resulting from simulation of the dynamic model; it properly predicts how an AUV traveling at low speeds behaves. Notice this path slightly rises instead of descending even though the stern plane is commanded to give the vehicle a downward motion. This shows how the AUV will tend to float because of the lift forces until the vehicle reaches a certain speed and the control surface (i.e. stern plane) takes effect.

An additional example of the importance of the full dynamic model occurs when there are thrusters present on the AUV that have a different direction than the main motion of the vehicle. This phenomena will alter the AUV’s momentum in the water, which will cause the AUV to experience momentum drag when executing a variety of coupled maneuvers.
Figure 2.4: (a) Decoupled model with $n = 1000$, $\delta_s = -15^\circ$, $\delta_r = -15^\circ$, (b) Full dynamic model with $n = 1000$, $\delta_s = -15^\circ$, $\delta_r = -15^\circ$ (i.e. maneuvers that require more than one control surface). In Fig. 2.4 both the rudder and stern plane of the vehicle are commanded to $-15^\circ$, simultaneously. In essence, the heading changes while the vehicle attempts to descend. The dynamic model in Fig. 2.4b captures how the coupled complex movement can produce drag, which slows the vehicle’s descent as opposed to the decoupled model path prediction in Fig. 2.4a.

In order to be truly autonomous the AUV must be capable of operating reliably in underwater environments under a variety of conditions. All of these principles make the use of information from the AUV model valuable in predicting paths in complex AUV maneuvers. For fast planning it is important to employ the simplest model needed for a given scenario.
CHAPTER 3

SAMPLING BASED MODEL PREDICTIVE CONTROL

Section 3.1, first gives an overview of MPC. Next, Section 3.2 describes the evolution of the SBMPC algorithm. Section 3.3 then examines the fundamentals of SBMPC and Section 3.4 presents the SBMPC algorithm. Finally, Section 3.5 provides a conceptual comparison of MPC and SBMPC that highlights the similarities and differences.

3.1 MPC Overview

Introduced to the process industry in the late 1970’s, MPC is a mixture of system theory and optimization [64]. It is a control approach that explicitly incorporates a model of the system to determine the control inputs by optimizing a cost function subject to constraints. The cost function calculates the desired control signal by using a model of the system to predict future system outputs. There are several versions of MPC, but they can all be thought of as either receding horizon or shrinking horizon.

3.1.1 Receding Horizon Model Predictive Control

MPC has three stages: Prediction, Optimization and Control. Fig. 3.1 provides a diagram of the stages implemented in MPC. At every time step $k$, the future system output is predicted for a pre-determined $N$ steps ahead (called the prediction horizon). The predicted outputs are a function of past inputs and outputs in conjunction with future control
inputs. A fundamental part of this method is the optimization problem that obtains future control inputs by minimizing a cost function subject to constraints on the system. Typically, the cost function $J$ consists of the error between the reference trajectory $r(k+i)$ and the predicted outputs $y(k+i)$ in addition to the control effort $u(k+i)$. In particular, the optimization problem is

$$
\min_{\{u(k),...,u(k+M-1)\}} J = \sum_{i=1}^{N} \|r(k+i) - y(k+i)\|_Q^2 + \sum_{i=0}^{M-1} \|u(k+i)\|_S^2 \quad (3.1)
$$

subject to the model constraints,

$$
x(k+i) = f(x(k+i-1), u(k+i-1)), \quad (3.2)
$$

$$
y(k+i) = g(x(k+i)), \quad (3.3)
$$

and the inequality constraints,

$$
Ax \leq b, \quad C(x) \leq 0, \quad u^l \leq u(k+i) \leq u^u, \quad (3.4)
$$

where the prediction and control horizons are $N$ and $M$ respectively, $Q$ and $S$ are respectively the error weights and control effort weights, and $C(x)$ represents the nonlinear
constraints on the states. The control inputs \([u(k+i) \cdots u(k+M-1)]\) are optimized within the control horizon, and the last control input, \(u(k+M-1)\), is held constant until the end of the prediction horizon. Only the first optimal control input calculated in the control sequence is implemented on the plant. Then the process starts over. This method is commonly referred to as Receding Horizon MPC (RHMPC) because the fixed prediction window is constantly moving. The objective is to get the predicted output \(y(k+i)\), to follow the reference trajectory \(r(k+i)\) utilizing the proposed inputs \(u(k+i)\) during the prediction horizon \(N\). To acquire a better understanding of the receding horizon concept refer to Fig. 3.2. Here for example, the prediction horizon, \(N = 3\), is fixed and moves by one sampling interval at each step. This is also sometimes called the moving horizon.
3.1.2 Shrinking Horizon Model Predictive Control

For the AUV applications considered in this research, it is necessary to consider a second viewpoint of MPC called Shrinking Horizon Model Predictive Control (SHMPC) [91]. As illustrated in Fig. 3.3, the horizon of the model prediction decreases as time increases. The horizon window is not fixed; it decreases by one sampling interval at each step. Hence, the horizon ”shrinks” as the end of the mission approaches. Also note in Fig. 3.3 that the end of the prediction horizon reaches the final time step (6) in each sampling interval, unlike RHMPC in Fig. 3.2 where the prediction horizon only reaches the final time in the last sampling interval displayed. In SHMPC, instead of having a pre-determined reference trajectory to attempt to follow, there is a goal (setpoint) that the outputs must reach in a fixed amount of time, yielding the optimization problem,

$$J = \min_{\{u(k),...,u(k+M-1)\}} \sum_{i=1}^{N^* - k} \|G - y(k + i)\|_Q^2 + \sum_{i=0}^{M-1} \|u(k + i)\|_S^2$$ (3.5)

subject to (3.3) and (3.4), where $G$ is the constant output goal (set point) and $N^*$ is the final time step. The first term in the cost function represents the deviation of the predicted output from the set point $G$. The prediction window is decreased by 1 as the time step is increased by 1. This allows the goal point values to be incorporated since there is only an interest in reaching the goal point at the end of the mission. The optimizer computes a sequence of control inputs for each step up to the goal point. SHMPC is able to regulate the control inputs to optimize the objective and simultaneously obtain the trajectory necessary to reach the desired goal point. Note that SHMPC is a special case of RHMPC for which the prediction horizon is long enough to reach a constant reference trajectory in the first time step. This ensures that an entirely new optimization problem does not have to be solved at each time step, which provides a more optimal path to the reference trajectory. It is important to know the fundamental difference between SHMPC and RHMPC, but for the remainder of this research, SHMPC will be referred to as MPC.
3.1.3 MPC Advantages

MPC has several benefits that make it attractive for use on AUVs. It naturally handles MIMO control problems so it can lead to better control of AUVs than classical control methods. Current and future constraints can be systematically incorporated during the design process, unlike conventional control methods, which makes it possible to address the AUV physical kinematic and dynamic limitations. MPC has a look ahead feature that enables the compensation of future errors, whereas a conventional feedback control method can only compensate for an error once it has occurred. Employing MPC enables the vehicle to take into account disturbances and possible constraint violations and to re-examine its status intelligently. It performs this re-examination at every time step and only implements the first control input, making this method robust. Another desirable feature of MPC is that it provides the ability to generate the trajectory and control inputs simultaneously.

3.1.4 MPC Disadvantages

This discussion would not be complete without considering weaknesses of MPC. MPC tuning procedures are not as straightforward as those for classical control methods. This shortcoming is because the relationship between MPC’s tuning parameters and the closed loop behavior is not explicit. Another problem is that an optimal solution may not be computed, depending on the time available to compute a solution. Therefore, only a sub-optimal solution may be obtained in certain cases. Since MPC was developed for the process control industry its computationally slow nature was acceptable because control updates were only necessary on the order of minutes or hours. For vehicles it is necessary to have updates on the order of seconds. This is a much more stringent requirement that brings an added level of complexity to the problem and provides a weakness for problems that have short time step requirements.

It has been stated previously that MPC utilized a model to predict future system outputs. There has to be a balance between a model that describes the system to accurately predict
the future outputs and simple enough to produce fast updates online. As stated previously in Section 1.3.1, past researchers have predominantly used linear models when implementing MPC, even when the physical systems are nonlinear. In some cases, a linearized model does produce sufficient results. However, for highly nonlinear systems such as AUVs, the use of a linearized model may not produce a good outcome. In this case, it is necessary to implement nonlinear MPC (NMPC), which incorporates the nonlinear model and/or nonlinear constraints. NMPC has not been used in the past, because traditional numerical methods applied to solve the optimization problem (3.5) tend to get stuck in local minima and cannot guarantee a solution. In order to develop a fast NMPC algorithm that can escape local minima, a new concept was developed called Sampling Based MPC (SBMPC) [24].

### 3.2 Evolution of SBMPC

The diverse fields of control theory, robotics, and artificial intelligence (AI) are different, but have similarities when it comes to planning [27]. Each possess concepts that can be used synergistically. Control theory has traditionally seen planning as determining the inputs to a system so the system can move from an initial state to a goal state, while considering optimality and stability. The systems are mostly represented by a mathematical model, whereas robotics have an autonomous system that must move from an initial configuration to a goal configuration and traditionally does not consider the system model. Lastly, AI views planning in a discrete space with a finite number of actions for an agent or tasks to move from an initial state to a goal state. Even though each field may use different labels and methods, the desire to plan is shared.

Originally, this research began by applying NMPC, implemented with sequential programming, to generate a path for an autonomous underwater vehicle [15]. The advantages of NMPC outlined in Section 3.1.3 have made planning with NMPC promising, but the weaknesses of Section 3.1.4 had to be addressed. Since the robotics and AI communities had the same goal for planning but have different approaches that tend to yield computation-
ally efficient algorithms, it was decided to incorporate these various concepts to produce an enhanced planner. Instead of utilizing traditional numerical methods for the optimization phase, SBMPC uses $A^\ast$ type optimization from the AI community. In addition, the idea of using sampling to consider only a finite number of solutions comes from robotics. Thus, SBMPC draws from the control theory, robotics and AI communities.

The concept behind SBMPC was first presented in [24]. As its name implies, SBMPC is dependent upon the concept of sampling, which has arisen as one of the major paradigms for the robotic motion planning community [54]. Sampling is the mechanism used to trade performance for computational efficiency. SBMPC employs quasi-random samples of the input space. Properly designed sampling algorithms have the theoretical property that if the sampling is dense enough, the sampling algorithm will find a solution when it exists (i.e., it has some type of completeness [54]). Unlike traditional MPC, which views the system behavior through the system inputs, the vast majority of previously developed sampling methods plan in the output space and attempt to find inputs that connect points in the output space.

However, SBMPC is an algorithm that, like traditional MPC, is based on viewing the system through its inputs. Unlike previous MPC methods, it uses sampling to provide the trade-off between performance and computational efficiency. This enhances the ability to achieve fast computation times. Also, in contrast to previous MPC methods, it does not rely on numerical optimization. Instead, it borrows from the AI community a graph search algorithm derived from $LPA^\ast$ [51], an incremental $A^\ast$ algorithm [54]. An incremental planning algorithm is one that can replan quickly by using information stored from a past planning algorithm. In general, graph search algorithms are efficient for problems with the type of discretization that arises from sampling and guarantee a solution that is globally optimal, subject to the employed sampling. Note that although SBMPC focuses on AUV motion planning in this dissertation, it is generally formulated and can be utilized for any NMPC or linear MPC problem.
3.3 SBMPC Overview

SBMPC is a method that can address some of NMPC weaknesses. It effectively reduces the problem size of MPC by sampling the inputs of the system. The method also replaces the traditional MPC optimization phase with $LPA^*$. The objective is to determine a sequence of control inputs that cause the system to achieve a given goal (set point) while solving the optimization problem (3.5). Section 3.1 describes MPC in some detail. Now it is necessary to provide in detail the other concepts implemented to produce SBMPC.

3.3.1 Sampling Based Motion Planning

Sampling based motion planning algorithms include Rapidly-exploring Random Trees (RRTs) [55], probability roadmaps [48], and randomized $A^*$ algorithms [58]. A common feature of each of these algorithms to date is that they work in the output space of the robot and employ various strategies for generating samples (i.e., random or pseudo-random points). In essence, as shown in Fig. 3.4, sampling based motion planning methods work by using sampling to construct a tree that connects the root (initial state) with a goal region. The general purpose of sampling is to cover the space so that the samples are uniformly distributed, while minimizing gaps and clusters [59]. This can prove to be challenging if an appropriate sampling technique is not implemented. The sampling method used in this research is based on Halton points [34] because samples can be added to existing Halton points while maintaining good uniformity and are thus more suitable for generating algorithms that have the flexibility to add samples when necessary.

Most online sampling based planning algorithms follow this general framework:

1. **Initialize:** Let $G(V; E)$ represent a search graph where $V$ contains at least one vertex (i.e., node), typically the start vertex and $E$ does not contain any edges.

2. **Vertex Selection Method (VSM):** Select a vertex $u$ in $V$ for expansion.

3. **Local Planning Method (LPM):** For some $u_{new} \in C_{free}$ (free states in the configuration space) and attempt to generate a path $\tau_s : [0, 1] \rightarrow: \tau(0) = u$ and $\tau(1) = u_{new}$. 

41
The path must be checked to ensure that no constraints are violated. If the LPM fails, then go back to Step 2.

4. **Insert an Edge in the Graph:** Insert $\tau_s$ into $E$, as an edge from $u$ to $u_{new}$. Insert $u_{new}$ into $V$ if it doesn’t already exist.

5. **Check for a Solution:** Check $G$ for a solution path.

6. **Return to Step 2:** Repeat unless a solution has been found or a failure condition has been met.

For SBMPC, the LPM of sampling based motion planners is simplified from what is effectively a two-point boundary value problem by sampling in the input space and integrating the system model to determine the next output node. Although most sampling based planners use proximity to a random or heuristically biased point as their VSM, the proposed SBMPC method bases the vertex (node) selection on an $A^*$ criterion, which involves computing the cost to the node and an optimistic (i.e., lower bound) prediction of
the cost from the current node to the goal; this eliminates the need for a potentially costly nearest neighbor search while promoting advancement towards the goal.

3.3.1.1 Input Sampling. There are two primary disadvantages to using output (i.e., configuration space) sampling as is commonly done in traditional sampling methods. The first limitation lies within the VSM, where the algorithm must determine the most ideal node to expand. This selection is typically made based on the proximity of nodes in the graph to a sampled output node and involves a potentially costly nearest neighbor search. The LPM presents the second and perhaps more troublesome problem, which is determining an input that connects a newly sampled node to the current node. This problem is essentially a two-point boundary value problem (BVP) that connects one output or state to another. There is no guarantee that such an input exists. Also, for systems with complex dynamics, the search itself can be computationally expensive, which leads to a computationally inefficient planner. In contrast, when the input space is sampled as proposed in this dissertation, the need for a nearest-neighbor search is eliminated, and the LPM is reduced to the integration of a system model, and therefore, only generates outputs that are achievable by the system.

In order to visualize this concept, consider an Ackerman steered vehicle at rest that has position \((x, y)\) and orientation \(\theta\), which are the outputs of the kinematic model. The model restricts the attainable outputs. All the dots in Fig. 3.5 are output nodes obtained from sampling the output space even though only the dots on the mesh surface can physically be obtained by the vehicle. There are a larger number of dots (sampled outputs) in the output space that do not lie in the achievable region (mesh surface). This means those sampled outputs are not physically possible, so traditional sample based methods would have to start the search over. This leads to an inefficient search that can substantially increase the computational time of the planner. The intersection of the grid lines in Fig. 3.5 correspond to the points in output space generated by a uniform sampling of the model inputs, the left and right wheel velocities. In essence, sampling in the input space leads to more efficient
3.3.1.2 The Implicit State Grid. Although extension of several of the existing sampling based paradigms can lead to input sampling algorithms like SBMPC [54], input sampling has not been used in most planning research. This is most likely due to the fact that input sampling is seen as being inefficient because it can result in highly dense samples in the output space since input sampling does not inherently lead to a uniformly discretized output space, such as a uniform grid. This problem is especially evident when encountering a local minimum problem associated with the $A^*$ algorithm, which can occur when planning in the presence of a large concave obstacle while the goal is on the other side of the obstacle. This situation is considered in depth for discretized 2D path planning in the work of [58], which discusses that the $A^*$ algorithm must explore all the states in the neighborhood of the local minimum, shown as the shaded region of Fig. 3.6, before progressing to the final solution. The issue that this presents to input sampling methods is that the number
of states within the local minimum is infinite because of the lack of a discretized output space.

Figure 3.6: Illustration of the necessity of an implicit state grid.

The second challenge resulting from the nature of input sampling as well as the lack of a grid is that the likelihood of two outputs (states) being identical is extremely small. All $A^*$-like algorithms utilize Bellman’s optimality principle to improve the path to a particular output by updating the paths through that output when a lower cost alternative is found. This feature is essential to the proper functioning of the algorithm and requires a mechanism to identify when outputs (states) are close enough to be considered the same. The scenario presented in Fig. 3.7 is a situation for which the lack of this mechanism would generate an inefficient path. In this situation, node $v_1$ is selected for expansion after which the lowest cost node is $v_3$. The implicit state grid then recognizes that $v_2$ and $v_3$ are close enough to be considered the same and updates the path to their grid cell to be path $c$ since $c < a + b$.

The concept of an implicit state grid [27] is introduced as a solution to both of the
challenges generated by input sampling. The implicit grid ensures that the graph generated by the SBMPC algorithm is constructed such that only one active output (state) exists in each grid cell, limiting the number of nodes that can exist within any finite region of the output space. In essence, the *implicit state grid* provides a discretized output space. It also allows for the efficient storage of potentially infinite grids by only storing the grid cells that contain nodes, which is increasingly important for higher dimensional problems [27]. The resolution of the grid is a significant factor in determining the performance of the algorithm with more fine grids in general requiring more computation time, due to the increased number of outputs, with the benefit being a more optimal solution. Therefore, the grid resolution is a useful tuning tool that enables SBMPC to effectively make the trade off between solution quality and computational performance.
3.3.2 Goal Directed Optimization

There is a class of discrete optimization techniques that have their origin in graph theory and have been further developed in the path planning literature. In this study these techniques will be called *goal-directed optimization* and refer to graph search algorithms such as Dijkstra’s algorithm and the $A^*$, $D^*$, and $LPA^*$ algorithms [54, 51]. Given a graph, these algorithms find a path that optimizes some cost of moving from a start node to some given goal. In contrast to discrete optimization algorithms such as branch-and-bound optimization [70], which “relaxes” continuous optimization problems, the goal-directed optimization methods are inherently discrete, and have often been used for real-time path planning.

Although not commonly recognized, goal-directed optimization approaches are capable of solving control theory problems for which the ultimate objective is to plan an optimal trajectory and control inputs to reach a goal (or set point) while optimizing a cost function. Hence, graph search algorithms can be applied to terminal constraint optimization problems and set point control problems. To observe this, consider the tree graph of Fig. 3.4. Each node of this tree can correspond to a system state and the entire tree may be generated by integrating sampled inputs to a system model. Assume that the cost of a trajectory is given by the sum of the cost of the corresponding edges (i.e., branches), where the cost of each edge is dependent not only on the states it connects but also the inputs that are used to connect those states. The use of the system model can be viewed simply as a means to generate the directed graph and associated edge costs.

3.4 SBMPC Algorithm

This section provides the SBMPC Algorithm implemented in this dissertation. However, first the variables used in the algorithm are described. The SBMPO algorithm and terms follow closely with $LPA^*$ [51], however the variation is in the Generate Neighbor
algorithm which generates the next state by integrating the model and considering constraint violations.

### 3.4.1 SBMPC Variables

SBMPC operates on a dynamic directed graph $G$ which is a set of all nodes and edges currently in the graph. $SUCC(v)$ represents the set of successors (children) of node $v \in G$ while $PRED(v)$ denotes the set of all predecessors (parents) of node $v \in G$. The cost of traversing from node $v$ to node $v' \in SUCC(v)$ is denoted by $c(v, v')$, where $0 < c(v, v') < \infty$. The optimization component of SBMPC is called Sampling Based Model Predictive Optimization (SBMPO) and is an algorithm that determines the optimal cost (i.e. shortest path, shortest time, least energy etc.) from a start node $v_{start} \in G$ to a goal node $v_{goal} \in G$.

The start distance of node $v \in G$ is given by $g^*(v)$ which is the cost of the optimal path from the given start node $v_{start}$ to the current node $v$. Note SBMPC can optimize various metrics, but this dissertation will focus on determining the shortest path.

SBMPC maintains two estimates of $g^*(v)$. The first estimate $g(v)$ is essentially the current cost from $v_{start}$ to the node $v$ while the second estimate, $rhs(v)$ is a one-step lookahead estimate based on $g(v')$ for $v' \in PRED(v)$ and provides more information than the estimate $g(v)$. The $rhs(v)$ value satisfies

$$rhs(v) = \begin{cases} 0, & \text{if } v = v_{start} \\ \min_{v' \in PRED(v)} (g(v') + c(v', v)), & \text{otherwise}. \end{cases} \quad (3.6)$$

A node $v$ is locally consistent iff $g(v) = rhs(v)$ and locally inconsistent iff $g(v) \neq rhs(v)$. If all nodes are locally consistent then $g(v)$ satisfies (3.6) for all $v \in G$ and is therefore equal to the start distance. This enables the ability to trace the shortest path from $v_{start}$ to any node $v$ by starting at $v$ and traversing to any predecessor $v'$ that minimizes $g(v') + c(v', v)$ until $v_{start}$ is reached.

To facilitate fast re-planning SBMPO does not make every node locally consistent after an edge cost change and instead uses a heuristic function $h(v, v_{goal})$ to focus the
search so that it only updates \( g(v) \) for nodes necessary to obtain the shortest path. The heuristic is used to approximate the goal distances and must follow the triangle inequality: 
\[
h(v_{goal}, v_{goal}) = 0 \text{ and } h(v, v_{goal}) \leq c(v, v') + h(v', v_{goal}) \text{ for all nodes } v \in G \text{ and } v' \in SUCC(s).
\]
SBMPO employs the heuristic function along with the start distance estimates to rank the priority queue containing the locally inconsistent nodes and thus all the nodes that need to be updated in order for them to be locally consistent. The priority of a node is determined by a two component key vector:

\[
key(v) = \begin{pmatrix} k_1(v) \\ k_2(v) \end{pmatrix} = \begin{pmatrix} \min(g(v), rhs(v)) + h(v, v_{goal}) \\ \min(g(v), rhs(v)) \end{pmatrix}
\]

where the keys are ordered lexicographically with the smaller key values having a higher priority.

### 3.4.2 The SBMPC Algorithm

The SBMPC algorithm has three main functions: Sampling Based Model Predictive Control, Sampling Based Model Predictive Optimization (SBMPO) and Generate Neighbor. The main SBMPC algorithm follows the general structure of MPC where SBMPO repeatedly computes the optimal path between the current state \( x_{current} \) and the goal state \( x_{goal} \). After a single path is generated \( x_{current} \) is updated to reflect the implementation of the first control input and the graph \( G \) is updated to reflect any system changes. These steps are repeated until the goal state is reached.

The second algorithm SBMPO repeatedly generates the neighbors of locally inconsistent nodes until \( v_{goal} \) is locally consistent or the key of the next node in the priority que is not smaller than \( key(v_{goal}) \). This follows closely with the ComputeShortestPath algorithm of \( LPA^* \) [51]. The node, \( v_{best} \), with the highest priority (lowest key value) is on top of the priority que. The algorithm then deals with two potential cases based on the consistency of the expanded node \( v_{best} \). If the node is locally overconsistent, \( g(v) > rhs(v) \), the \( g \)-value is set to \( rhs(v) \) making the node locally consistent. The successors of \( v \) are then updated.
The update node process includes recalculating $\text{rhs}(v)$ and key values, checking for local consistency and either adding or removing the node from the priority queue accordingly. For the case when the node is locally underconsistent, $g(v) < \text{rhs}(v)$, the $g$-value is set to $\infty$ making the node either locally consistent or overconsistent. This change can affect the node along with its successors which then go through the node update process.

The Generate Neighbor algorithm determines the successor nodes of the current node. In the input space, a set of quasi-random samples are generated that are then used with a model of the system to predict a set of paths to a new set of outputs (states) with $x_{\text{current}}$ being the initial condition. The branching factor $B$ determines the number of paths that will be generated. The path is represented by a sequence of states $x(t)$ for $t = t_1, t_1 + \Delta t, \ldots, t_2$, where $\Delta t$ is the model step size. The set of states that do not violate any state or obstacle constraints is called $X_{\text{free}}$. If $x(t) \in X_{\text{free}}$ then the new neighbor node $x_{\text{new}}$ and the connecting edge can be added to the graph. If $x_{\text{new}} \in STATE.GRID$, then the node currently exists in the graph and only the new path to get to the existing node needs to be added.

**Algorithm 1 Sampling Based Model Predictive Control**

1: $x_{\text{current}} \leftarrow \text{start}$
2: repeat
3: SBMPO ()
4: Update system state, $x_{\text{current}}$
5: Update graph, $G$
6: until the goal state is achieved

### 3.5 Comparison of SBMPC and MPC

SBMPC was developed out of the concept of traditional MPC, i.e., MPC approaches that use more standard numerical optimization to solve the control update online. SBMPC and MPC share similarities, but there are some major differences that should be highlighted. Both methods are established through the inputs of the system. This saves SBMPC some
Algorithm 2 SBMPO()

1: while PRIORITY.TopKey() < v_goal.key || v_goal.rhs ≠ v_goal.g do
2: \( v_{best} \leftarrow PRIORITY.Top() \)
3: Generate_Neighbors \((v_{best}, B)\)
4: if \( v_{best}.g > v_{best}.rhs \) then
5: \( v_{best}.g = v_{best}.rhs \)
6: for all \( v \in SUCC(v_{best}) \) do
7: \( \) Update the node, \( v \)
8: end for
9: else
10: \( v_{best}.g = \infty \)
11: for all \( v \in SUCC(v_{best}) \cup v_{best} \) do
12: \( \) Update the node, \( v \)
13: end for
14: end if
15: end while

Algorithm 3 Generate_Neighbors (Vertex v, Branching B)

1: for \( i = 0 \) to B do
2: Generate sampled input, \( u \in \mathbb{R}^u \cap U_{free} \)
3: for \( t = t_1 : dt_{integ} : t_2 \) do
4: Evaluate model: \( x(t) = f(v.x, u) \)
5: if \( x(t) \notin X_{free}(t) \) then
6: Break
7: end if
8: end for
9: \( x_{new} = x(t_2) \)
10: if \( x_{new} \in STATE\_GRID \) and \( x_{new} \in X_{free} \) then
11: Add Edge\((v.x, x_{new})\) to graph, \( G \)
12: else if \( x_{new} \in X_{free} \) then
13: Add Vertex\((x_{new})\) to graph, \( G \)
14: Add Edge\((v.x, x_{new})\) to graph, \( G \)
15: end if
16: end for
computation time because it does not misuse time with attempting to connect output nodes via an input that may not exist. Another concept that the two methods share is the use of models to predict future outputs. By incorporating this concept SBMPC reduces the nearest neighbor calculation that most sampling based motion planners encounter.

To understand the relationship between sampling based algorithms and MPC optimization, it is essential to pose sampling based motion planning as an optimization problem. To illustrate this point, note that, subject to the constraints of the sampling, a graph search algorithm, can effectively solve the mixed integer nonlinear optimization problem,

$$\min_{\{u(k),\ldots,u(k+N-1)\}} J = \sum_{i=0}^{N} \| y(k+i+1) - y(k+i) \|_{Q(i)} + \sum_{i=0}^{N-1} \| \Delta u(k+i) \|_{S(i)}$$

(3.8)

subject to the system equations,

$$x(k+i) = f(x(k+i-1), u(k+i-1)),$$

(3.9)

$$y(k) = g(x(k)),$$

(3.10)

and the constraints,

$$\| y(k+N) - G \| \leq \epsilon,$$

(3.11)

$$x(k+i) \in X_{free} \forall \ i \leq N,$$

(3.12)

$$u(k+i) \in U_{free} \forall \ i \leq N,$$

(3.13)

where \( \Delta u(k+i) = u(k+i) - u(k+i-1) \), \( Q(i) \geq 0 \), \( S(i) \geq 0 \), and \( X_{free} \) and \( U_{free} \) represent the states and inputs respectively that do not violate any of the problem constraints. The term \( \| y(k+i+1) - y(k+i) \|_{Q(i)} + \| \Delta u(k+i) \|_{S(i)} \) represents the edge cost of the path between the current predicted output \( y(k+i) \) and the next predicted output \( y(k+i+1) \). The goal state \( G \) is represented as a terminal constraint as opposed to being explicitly incorporated into the cost function. Although goal-directed optimization methods implicitly consider the goal through the use of a function that computes a rigorous
Figure 3.8: SBMPC optimization phase

lower bound of the cost from a particular state to $G$, this function, often referred to as an “admissible heuristic” in the robotics literature, is eventually replaced by actual cost values based on the predictions and therefore does not appear in the final cost function. The cost function can be modified to minimize any metric as long as it can be computed as the sum of edge costs. Note that the SBMPC algorithm formation (3.8) - (3.13) is similar to the traditional MPC equations (3.1) - (3.4).

SBMPC and MPC both rely on a similar decomposition of the cost function $f(v) = g(v) + h(v)$, where $v$ represents the current node (or output) in the optimization phase. This function is used differently in the optimization phase, but the main concept is the same for both methods. Referring to Fig. 3.8, in general, SBMPC’s $g(v)$ is the actual cost from the start node to the next node as depicted by the red solid line. Referring to Fig. 3.9, $g(v)$ in traditional MPC is the cost from the current output to the end of the control horizon as shown by the solid yellow line. As depicted in Fig. 3.8, $h(v)$ in SBMPC represents an
optimistic estimate (or heuristic) of the cost from the next node to the goal node, meaning that the estimate is less than but close to the true optimal cost path to the goal as shown by the dashed yellow line. MPC uses a similar concept when the control horizon is not equal to the prediction horizon. As represented by the dashed yellow line in Fig.3.9 the control input is held constant from the end of the control horizon (solid yellow line in Fig. 3.9) to the end of the prediction horizon. This provides a realistic estimate of the cost to the goal sometimes called the “cost-to-go” which is implemented when it is too computationally expensive to actually calculate the cost to the goal. This makes for a rough estimate since it is assumed the control input does not change even though it may change in the future. Although SBMPC and MPC rely on a similar decomposition of $f(v)$, the way the two methods use the cost function $f(v)$ is different as discussed below.

SBMPC uses $f(v)$ to determine the top node in the priority queue. This allows SBMPC optimization to perform in a way that can revisit nodes if a more optimal path is determined.
Figure 3.10: Traditional MPC optimization

Figure 3.11: Goal-directed optimization
when a new node is evaluated on the priority. However, MPC uses $f(v)$ to determine the entire control trajectory at every iteration. These differences are due to the different optimization methods used in SBMPC and MPC.

Goal-directed optimization approaches operate differently than traditional optimization methods utilized in MPC as illustrated by contrasting Figs. 3.10 and 3.11, which contain the term $u^{(i)}(k)$, where $u$ denotes the control input vector (the optimization parameter), $i$ denotes the $i^{th}$ iteration, and $k$ denotes the predicted time step. At each current time step, the optimization method must determine the future control trajectory. Then a new optimization problem is started at the next time step. Referring to Fig. 3.10, at the current time step standard MPC optimization determines an initial sequence $\{u^{(0)}(k)\}$ of predicted control inputs (denoted by the left array) and iterates (denoted by the arrows) until it calculates the optimal predicted control trajectory $\{u^*(k)\}$ for the entire path (denoted by the right-most array). Referring to Fig. 3.11, goal-directed optimization at the current time step, in contrast to traditional MPC optimization, computes each predicted control input separately and backs up when needed as illustrated by the iterations corresponding to the 3rd, 4th, and 5th arrays. This feature enables it to avoid local minima. It converges to the optimal predicted control sequence $\{u^*(k)\}$ (denoted by the right-most array), which is the optimal solution subject to the sampling, whereas a nonlinear programming method may get stuck at a local minimum.

Another difference is the number of the tuning parameters. Traditional MPC has the prediction horizon, control horizon, and weights for each input and output as tuning parameters, which are not straightforward. However, SBMPC only has the grid size (resolution) and sample number (branching factor), which are a fewer number and different type of tuning parameters.
CHAPTER 4

SIMULATIONS

This chapter provides simulation results for the SBMPC algorithm applied to AUVs. First, Section 4.1, presents simulations that compare traditional NMPC and SBMPC. Section 4.2 utilizes several different ocean scenarios and presents simulations for AUV’s kinematic. Then, Section 4.3 presents simulations that show benefits of utilizing the decoupled and full dynamic models in motion planning. Finally, Section 4.4 evaluates the effect of the tuning parameters on computation time and cost.

A 2.93 GHz Intel Core 2 Duo desktop was used for all simulations in this chapter. The models were discretized by implementing Euler’s method ($\dot{x} = \frac{x(k+1) - x(k)}{T}$). In addition, all obstacles were modeled as circles or spheres due to the temporary limitations of the SBMPC software used to generate the simulation results.

4.1 Simulation Comparison of NMPC and SBMPC Using a 2D Kinematic Model

The results presented in this section have two purposes: 1) to compare SBMPC with NMPC on an AUV moving in cluttered environments with random obstacle location, 2) to compare how SBMPC and NMPC algorithms behave when the start and goal locations vary. The first simulations are difficult to perform because the vehicle must successfully maneuver through complex environments where local minimum are present. The second simulations test how varying the start and goal locations can effect the various methods.
from determining an optimal path. It is assumed the obstacle information is available to the algorithms. The problems associated with the uncertainty in sensing obstacles are beyond the scope of this research. However, it must be addressed for real world implementation on AUVs.

### 4.1.1 Random Obstacles in Cluttered Environment

Motion planning for an AUV moving in the horizontal plane is developed using a reduced 2-D kinematic model of (2.10), given by

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\psi}
\end{bmatrix} =
\begin{bmatrix}
\cos \psi & 0 & u \\
\sin \psi & 0 & r \\
0 & 1 & 0
\end{bmatrix}.
\]

Consequently, there are two inputs and three states in this model. The velocity \( u \) was constrained to lie in the interval \([0 2]\) m/s, and the steering rate \( r \) was constrained to the interval \([-15 15]\) deg/s.

In order to evaluate SBMPC it is compared with \( \text{MPC}_{\text{SQ}} \), a nonlinear MPC implementation that employs a MATLAB implementation\(^1\) of sequential quadratic programming (SQP) using an active set strategy, and \( \text{MPC}_{\text{GA}} \), a NMPC implementation which uses a MATLAB implementation\(^2\) of a genetic algorithm (GA) to determine the system inputs. Genetic algorithms, though relatively slow, are considered here because \( \text{MPC}_{\text{SQ}} \) and other MPC implementations built on Newton-type algorithms tend to fail in cluttered environment planning because of convergence to one of the many local minima. GA is a global search algorithm inspired by evolutionary biology, which produces an algorithm that is less likely to get trapped in local minima.

The population is the number of individuals that span the search space in each generation. The best individuals are selected to produce children in the next generation. The reproduction of children can occur through crossover or mutation. Crossover will recombine the parent individuals from the previous generation to make a child individual in the

---

\(^1\) MATLAB Optimization Toolbox
\(^2\) MATLAB Genetic Algorithm and Direct Search Toolbox
Table 4.1: Genetic Algorithm Options

<table>
<thead>
<tr>
<th></th>
<th>MPC\textsubscript{GA1}</th>
<th>MPC\textsubscript{GA2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>No. of Generations</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>Creation Function</td>
<td>Random</td>
<td>Random</td>
</tr>
<tr>
<td>Scaling Function</td>
<td>Rank</td>
<td>Rank</td>
</tr>
<tr>
<td>Selection Function</td>
<td>Stochastic Uniform</td>
<td>Stochastic Uniform</td>
</tr>
<tr>
<td>Mutation Function</td>
<td>Adaptive Random</td>
<td>Adaptive Random</td>
</tr>
<tr>
<td>Crossover Fraction</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Crossover Function</td>
<td>Scattered</td>
<td>Scattered</td>
</tr>
<tr>
<td>Time Limit</td>
<td>0.7\text{s}</td>
<td>1\text{s}</td>
</tr>
</tbody>
</table>

Table 4.2: Simulation Parameters

<table>
<thead>
<tr>
<th></th>
<th>MPC\textsubscript{SQ}</th>
<th>MPC\textsubscript{GA1}</th>
<th>MPC\textsubscript{GA2}</th>
<th>SBMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Time Step ($T_s$)</td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
</tr>
<tr>
<td>Control Update Period ($T_c$)</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>Prediction Horizon ($N$)</td>
<td>[19,57]</td>
<td>[19,57]</td>
<td>[19,57]</td>
<td>[19,57]</td>
</tr>
<tr>
<td>Control Horizon ($M$)</td>
<td>1</td>
<td>1</td>
<td>N/2</td>
<td>$N$</td>
</tr>
<tr>
<td>No. of Input Samples</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>10</td>
</tr>
<tr>
<td>Cost Function Weight</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>$f = 0.8g(v) + h(v)$</td>
</tr>
</tbody>
</table>

next generation, while mutation will augment an individual from the previous generation. There are various methods that can be employed for any of these steps. The general options used for GA in this dissertation are found in Table 4.1. Due to the fundamental differences in SBMPC and traditional MPC, two different parameter sets were used for MPC\textsubscript{GA} and the corresponding simulations are denoted by MPC\textsubscript{GA,1} and MPC\textsubscript{GA,2}. The parameters for the simulations are shown in Table 4.2.

The parameters for MPC\textsubscript{GA1} were chosen to enable MPC to have greater convergence speed, while using the same control update period as SBMPC. Table 4.2 indicates that SBMPC has a control horizon, $M = N$. This is to ensure SBMPC searches for an optimal path. In Table 4.2 the prediction horizon varies because each random scenario requires a different number of steps to traverse from the start position to the goal position. SBMPC was used to solve the optimization problem (3.8) - (3.13) with $Q(i) = I$ and $S(i) = 0$, $f = 0.8g(v) + h(v)$.
Table 4.3: Case 1: Simulation Results for Scenario 1

<table>
<thead>
<tr>
<th></th>
<th>MPC\textsubscript{SQ}</th>
<th>MPC\textsubscript{GA,1}</th>
<th>MPC\textsubscript{GA,2}</th>
<th>SBMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean First Update CPU Time</td>
<td>3.76s</td>
<td>1.89s</td>
<td>4.64s</td>
<td>1.59s</td>
</tr>
<tr>
<td>Median First Update CPU Time</td>
<td>1.99s</td>
<td>1.69s</td>
<td>4.21s</td>
<td>0.11s</td>
</tr>
<tr>
<td>Mean Distance \textsuperscript{a}</td>
<td>30.24m</td>
<td>33.04m</td>
<td>38.95m</td>
<td>30.62m</td>
</tr>
<tr>
<td>Success Rate</td>
<td>7%</td>
<td>73%</td>
<td>74%</td>
<td>100%</td>
</tr>
</tbody>
</table>

\textsuperscript{a}Distance only includes successful trajectories.

which determines the optimal number of steps $N^*$ in addition to the control input sequence. For simulations purposes, MPC\textsubscript{SQ} and MPC\textsubscript{GA} used $N^*$ determined from SBMPC. MPC can also include $N$ as an optimization variable to solve an optimization problem similar to (3.8). However, this has primarily been accomplished using mixed integer linear programming \cite{11}, which is applicable only to LMPC. To accomplish this for NMPC requires mixed integer nonlinear programming (MINLP), a more complex optimization problem that is difficult to implement. In each simulation the implicit state grid resolution was 0.1m.

The basic problem is to use the reduced kinematic model (4.1) to plan a minimum-distance trajectory for the AUV from a start posture $(0m, 0m, 0^\circ)$ to a goal point $(20m, 20m)$ while avoiding the numerous obstacles in a cluttered environment. There were 100 simulations in which 30 obstacles of various sizes and locations were randomly generated to produce different scenarios. The results of the simulations are shown in Table 4.3. Representative results from the 100 scenarios are shown in Fig. 4.1 and Fig. 4.2. SBMPC outperforms both MPC\textsubscript{GA,1} and MPC\textsubscript{GA,2} in terms of optimality (i.e., the mean distance traveled from start to goal) and computation time as measured by either the mean or median. However, MPC\textsubscript{SQ} has a slightly lower cost path than SBMPC for the paths that reach the goal. Note that this comes at a price since SBMPC reaches the goal 100\% of the time, whereas MPC\textsubscript{SQ} reaches the goal only 7\% of the time. These statistics are important since true autonomy requires a vehicle to be able to successfully complete a mission on its own. There is an order of magnitude difference in SBMPC’s mean CPU time and median CPU
Figure 4.1: Cluttered environment scenario in which $MPC_{SQ}$ fails to enable the AUV to reach the goal.

time in Table 4.3 because a few of the randomly generated scenarios had obstacles clustered to form larger obstacles, which creates computationally intensive planning problems. Fig. 4.1 is typical of a scenario in which SBMPC, $MPC_{GA1}$ and $MPC_{GA2}$ simulations reach the goal, but $MPC_{SQ}$ converges to a local minimum located behind a cluster of obstacles. This shows the disadvantage of using a traditional gradient based optimization method when working in complex environments. The scenario of Fig. 4.2 demonstrates the ability of SBMPC to navigate and determine a path when the other methods cannot reach the goal. Note that lower costs paths for $MPC_{GA}$ can be achieved with an increase in $M$, population size and/or number of generations, but this would increase the computation time.

4.1.2 Random Start and Goal Cluttered Environment

For these simulations, the obstacles are uniformly distributed and the start and goal locations are randomly generated. The starting point of the optimization can effect the algorithm’s ability to determine an optimal solution. There were 100 random simulations
where $X, Y$ and $\psi$ for the start posture were chosen randomly in the respective ranges $[0, 20]m$, $[0, 1]m$, $[30^\circ, 150^\circ]$ assuming a uniform distribution, and $X$ and $Y$ for the goal position were chosen randomly in the respective ranges $[0, 20]m$ and $[19, 20]m$. The parameters for the simulations are the same as Table 4.2 except the prediction horizon now varies in the range $[11, 122]$.

The results for these randomly generated simulations are shown in Table 4.4. The results are similar to the previous scenario, but the mean computation times of $\text{MPC}_{SQ}$, $\text{MPC}_{GA1}$, and $\text{MPC}_{GA2}$ increased while the computation time for $\text{SBMPC}$ decreased. Even though $\text{SBMPC}$ had a better success rate than the other methods, it failed once. The case when $\text{SBMPC}$ failed is depicted in Fig. 4.3. This case shows the result when the sample number is set too small, such that the space is not sampled properly. The goal for this case was $(4.28, 19.21)m$. $\text{SBMPC}$ reaches $(4.57, 20.95)m$ which is close but not within the $0.5m$ radius goal region that would prompt the algorithm to stop. Since the vehicle does not arrive at the goal region within the threshold, $\text{SBMPC}$ continues to search; but the vehicle

Figure 4.2: Cluttered environment scenario in which $\text{SBMPC}$ yielded the only path to the goal.
Table 4.4: Case 2: Simulation Results for Scenario 2

<table>
<thead>
<tr>
<th></th>
<th>MPC\textsubscript{SQ}</th>
<th>MPC\textsubscript{GA,1}</th>
<th>MPC\textsubscript{GA,2}</th>
<th>SBMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean First Update CPU Time</td>
<td>2.43s</td>
<td>3.85s</td>
<td>6.91s</td>
<td>1.02s</td>
</tr>
<tr>
<td>Median First Update CPU Time</td>
<td>2.24s</td>
<td>3.42s</td>
<td>4.77s</td>
<td>1.8e-02s</td>
</tr>
<tr>
<td>Mean Distance \textsuperscript{a}</td>
<td>20.57m</td>
<td>23.24m</td>
<td>25.09m</td>
<td>22.20</td>
</tr>
<tr>
<td>Success Rate</td>
<td>9%</td>
<td>90%</td>
<td>81%</td>
<td>99%</td>
</tr>
</tbody>
</table>

\textsuperscript{a}Distance only includes successful trajectories.

cannot physically reach the goal because of the kinematic and sampling limitations, and hence bypasses the goal region. The problem is easily resolved by increasing the number of samples. Fig. 4.4 shows a repeat run of SBMPC when the samples are increased from 10 to 25. In the simulation run, the AUV is capable of arriving at the goal. Fig. 4.5 shows a case when all four methods arrive at the goal, but MPC\textsubscript{SQ} takes a more optimal route. MPC\textsubscript{SQ} rarely reaches the goal. However, when it does, it is understandable that MPC\textsubscript{SQ} would choose a more optimal route than, MPC\textsubscript{GA}, which does not allow the full time to find the optimal solution and SBMPC whose optimality is subject to the sampling. Even in the scenarios where there are random start and random goal regions, MPC\textsubscript{SQ} often has problems reaching the goal, as illustrated in Fig. 4.6.

4.2 SBMPC Using a 3D Kinematic Model

A future vision of the U.S. Navy for mine countermeasures is to have modular unmanned vehicles that can handle different missions in underwater environments [69]. This requires an AUV that can operate in any condition that may occur. The following set of simulations represents various sea floor bottoms and obstacle clutter that an AUV may encounter during a typical underwater mission. The vehicle must be able to avoid obstacles and create a trajectory for the vehicle to operate successfully in various missions. Simulating three types of obstacles provide a glimpse into how SBMPC performs. Fig. 4.7 through 4.9 show that SBMPC can successfully traverse the area with these various types
Figure 4.3: The one evenly spaced obstacle scenario in which SBMPC did not reach goal.

Figure 4.4: Recalculation of the path with number of samples increased to 25.
Figure 4.5: Evenly spaced obstacles scenario for which each planning algorithm reached the goal but $MPC_{SQ}$ determined a more distance optimal path.

Figure 4.6: The one evenly spaced obstacle scenario in which $MPC_{SQ}$ failed to enable the AUV to reach the goal.
Table 4.5: Simulation Constraints for the 3D Kinematic Model

<table>
<thead>
<tr>
<th>Inputs</th>
<th>min</th>
<th>max</th>
<th>States</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>0 m/s</td>
<td>2 m/s</td>
<td>x</td>
<td>-5 m</td>
<td>30 m</td>
</tr>
<tr>
<td>v</td>
<td>-0.1 m/s</td>
<td>0.1 m/s</td>
<td>y</td>
<td>-5 m</td>
<td>30 m</td>
</tr>
<tr>
<td>w</td>
<td>-0.1 m/s</td>
<td>0.1 m/s</td>
<td>z</td>
<td>-20 m</td>
<td>0 m</td>
</tr>
<tr>
<td>p</td>
<td>$-5^\circ/s$</td>
<td>$5^\circ/s$</td>
<td>$\phi$</td>
<td>$-15^\circ$</td>
<td>$15^\circ$</td>
</tr>
<tr>
<td>q</td>
<td>$-5^\circ/s$</td>
<td>$5^\circ/s$</td>
<td>$\theta$</td>
<td>$-15^\circ$</td>
<td>$15^\circ$</td>
</tr>
<tr>
<td>r</td>
<td>$-15^\circ/s$</td>
<td>$15^\circ/s$</td>
<td>$\psi$</td>
<td>$-360^\circ$</td>
<td>$360^\circ$</td>
</tr>
</tbody>
</table>

of obstacles present. The simulations are run with a sampling size of 25 and a grid size of 0.1 m. The 3D kinematic model (2.10) is employed for these simulations. The constraints are given in Table 4.5.

4.2.1 Steep Hill

As depicted in Fig. 4.7, the steep hill is a slightly higher uprise than a mount with its highest point 8 m. The steep hill was constructed utilizing six sphere obstacle constraints. The AUV’s start location was ($-4m \ 17m \ -18m \ 0^\circ \ 0^\circ \ 0^\circ$) and the AUV successfully reached the goal location ($19m \ 7m \ -14m$) with a computation time of 11.38 s.

4.2.2 Cliff

The cliff was constructed using 39 sphere obstacle constraints. The start and goal were ($-4m \ 7m \ -17m \ 0^\circ \ 0^\circ \ 0^\circ$) and ($25m \ 14m \ -5m$), respectively. In order to better span the input space, the sample number was increased to 35. The computation time was extremely high at 117.02 s. This comes from the large number of compact clustered obstacles the vehicle must avoid.

4.2.3 Local Minima Scenario

A common local minima problem and possible scenario is to have a set of concave obstacles between the start and goal. There are some path planning methods that would normally get stuck in the local minima and could not traverse to the goal. However, as
Figure 4.7: Steep hill scenario.

Figure 4.8: Cliff scenario.
shown in Fig. 4.9, SBMPC does not get stuck and has a computation time of 0.59s. Note that whenever the vehicle is behind an obstacle or a group of obstacles and has to increase its distance from the goal to achieve the goal it is in a local minima position. In addition, these results show that SBMPC’s implementation of the implicit state grid helps prevent the issues with input sampling discussed in Section 3.3.1.2.

4.2.4 Multiple Obstacles

Some underwater environments will require the AUV to navigate around multiple obstacles. These simulations assumed there were random start, goal and obstacle locations. The start locations $X, Y, Z$ and $\psi$ were chosen randomly in the respective ranges $[0 20]m, [0 1]m, [-12 -8]m, [30° 150°]$, and the goal was chosen randomly in the respective ranges $[0 20]m, [19 20]m, [-12 -8]m$. In addition, there were 40 randomly generated obstacles. There were 100 simulation runs with a branching factor of 25 to represent different multiple obstacle underwater environment configurations. Fig. 4.10 and Fig. 4.11 exemplify two
random scenarios generated. In both scenarios SBMPC was capable of allowing the AUV to maneuver in the cluttered environment. Out of the 100 runs, SBMPC was successful 100% of the time with a mean computation time of 0.43s and median computation time of 0.15s.

4.3 SBMPC Using the Decoupled and Full Dynamic Models

The simulations in this section are implemented using the decoupled model (2.27) - (2.29), which is a simplified dynamic model that assumes the vehicle is lightly coupled and the full dynamic model (2.21) - (2.26). The state and control constraints for these simulations are shown in Table 4.6. The simulation parameters are given in Table 4.7. The parameters were tuned to find high quality solutions at fast control updates.
Table 4.6: Simulation Constraints for the Decoupled and Full Dynamic Models

<table>
<thead>
<tr>
<th>States</th>
<th>min</th>
<th>max</th>
<th>States</th>
<th>min</th>
<th>max</th>
<th>Inputs</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>-30 m</td>
<td>130 m</td>
<td>u</td>
<td>0 m/s</td>
<td>2 m/s</td>
<td>δ_r</td>
<td>-22°</td>
<td>22°</td>
</tr>
<tr>
<td>y</td>
<td>-30 m/s</td>
<td>130 m</td>
<td>v</td>
<td>-2 m/s</td>
<td>2 m/s</td>
<td>δ_s</td>
<td>-22°</td>
<td>22°</td>
</tr>
<tr>
<td>z</td>
<td>-20 m</td>
<td>5 m</td>
<td>w</td>
<td>-2 m/s</td>
<td>2 m/s</td>
<td>δ_b</td>
<td>-22°</td>
<td>22°</td>
</tr>
<tr>
<td>φ</td>
<td>-15°</td>
<td>15°</td>
<td>p</td>
<td>-5°/s</td>
<td>5°/s</td>
<td>δ_bp</td>
<td>-22°</td>
<td>22°</td>
</tr>
<tr>
<td>θ</td>
<td>-85°</td>
<td>85°</td>
<td>q</td>
<td>-5°/s</td>
<td>5°/s</td>
<td>δ_θb</td>
<td>-22°</td>
<td>22°</td>
</tr>
<tr>
<td>ψ</td>
<td>-360°</td>
<td>360°</td>
<td>r</td>
<td>-15°/s</td>
<td>15°/s</td>
<td>n</td>
<td>0 rpm</td>
<td>1500 rpm</td>
</tr>
</tbody>
</table>

Table 4.7: Simulation Parameters for the Decoupled and Full Dynamic Models

<table>
<thead>
<tr>
<th></th>
<th>Decoupled</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Time Steps</td>
<td>0.5s</td>
<td>0.5s</td>
</tr>
<tr>
<td>Control updates</td>
<td>10s</td>
<td>10s</td>
</tr>
<tr>
<td>No. of Input Samples</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Grid Resolution</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Cost Function Weight</td>
<td>$f = g(v) + 1.2h(v)$</td>
<td>$f = g(v) + 1.2h(v)$</td>
</tr>
</tbody>
</table>
4.3.1 Multiple Obstacles

Similar to Section 4.2.4, the simulations in this section are completed with random start, goal and obstacle locations. The start locations \( X, Y, Z \) and \( \psi \) were chosen randomly in the respective ranges \([0 \ 50] m, [0 \ 5] m, [-13 \ -3] m, [30^\circ \ 150^\circ] \), and the goal was chosen randomly in the respective ranges \([0 \ 50] m, [95 \ 100] m, [-13 \ -3] m \). In each scenario 60 obstacles were chosen randomly. Figs. 4.12 and 4.13 show an example of a path generated in the random scenario, based on respectively the decoupled model and full dynamic model. Just as the scenarios used for planning with the kinematic model, SBMPC was successful in reaching the goal 100% of the time in the presence of multiple nonlinear constraints. The computational time results are given in Table 4.8. The CPU time for the full dynamic model is larger than the decoupled model, however this is understandable since integrating this model is more computationally intensive than the decoupled model. Note the full dynamic model still allows for fast computation times.
Figure 4.13: Full dynamic model random start, goal and obstacle scenario.

Table 4.8: Simulation Results for the Decoupled and Full Dynamic Models

<table>
<thead>
<tr>
<th></th>
<th>Decoupled Dynamic</th>
<th>Full Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completion</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Mean CPU Time</td>
<td>0.40s</td>
<td>1.02s</td>
</tr>
<tr>
<td>Median CPU Time</td>
<td>0.13s</td>
<td>0.46s</td>
</tr>
</tbody>
</table>
4.3.2 Steep Hill

Both the decoupled model and full dynamic model were simulated to determine the path over a steep hill. Each model predicts different paths to the goal. In Fig. 4.14, the decoupled model successfully reaches the goal with a computation time of 55.2s. However, the full dynamic model was not able to determine a path because the vehicle was too close to the steep hill. As depicted in Fig. 4.15, the dynamic model was able to predict that there was not enough momentum to overcome the hill in such a short distance. However, because the decoupled model is a simplified dynamic model it was not able to capture this issue.

In order to verify that the reason a path was not determined in Fig 4.15a, was due to the dynamic model prediction and not the SBMPC algorithm, the control commands generated from the decoupled model that successfully gave a path to the goal were applied as the inputs to the dynamic model. As shown in Fig. 4.15b, the path produced was still not capable of overcoming the steep hill and reaching the goal. This showed that indeed the reason was because the vehicle was not physically capable of achieving this type of movement and not a failure in SBMPC.

4.3.3 Cliff

In section 4.2.2, the kinematic model was used to simulate the AUV movement over a cliff. Fig. 4.8 shows that the use of the kinematic model produced a successful path to the goal. As stated previously the kinematic model does not account for the forces or control surface limitations (or saturation). When the decoupled and dynamic model are used to predict the path a successful path could not be determined, because of the vehicle dynamic limitations. If just the kinematic model had been used to determine the trajectory then it would have been declared that the AUV could have produced a path that the vehicle could not traverse when starting at the given distance away from the cliff. This shows that even though the decoupled model does not include the full details about the vehicle, it does
Figure 4.14: Decoupled model steep hill scenario.

Figure 4.15: (a) Full dynamic model steep hill scenario. (b) Full dynamic model steep hill scenario with input commands from decoupled model.
provide a better prediction of the AUV movement than a kinematic model when the motion pushes the vehicle to an extreme.

4.4 Tuning Parameters

As discussed in Section 3.5, SBMPC has two main tuning parameters, the sampling number (branching factor) and grid resolution (size). Each tuning parameter has an effect on the computation time and cost. In this section one of the random scenarios from Section 4.2.4 was investigated. The 3D kinematic model (2.11) was used to determine the path of an AUV around 40 obstacles and the effects of the tuning parameters were examined.

4.4.1 Effect of Sample Number

The sample number which is the number of samples that cover the input space has an effect on the computation time of SBMPC. The sample number was tested at 10 and increased by two up to 40. The grid resolution was held constant at 0.1m. As shown in Fig. 4.17, the
effect of the sampling number is nonlinear, so there is no direct relationship between the sample size and computation time. The reason for the nonlinear trend is threefold. First as shown in Fig. 4.17 by samples 10 and 12, when there are not enough samples to span the space it can also increase the CPU time, because it takes more iterations (i.e. steps) to determine the solution. A good tuning of the parameter occurs at 14 samples which results in a smaller computation time. The second trend, as shown in Fig. 4.17 between samples 14 and 22, is that after a good tuning of the parameter, increasing the number of samples also increases the computation times. Lastly, a factor that contributes to the computation time is the path produced by SBMPC. It is possible for a larger sample number to have a lower computation time when the path it generates to the goal encounters a smaller cluster of obstacles. Figs. 4.18a and 4.18b show the paths generated respectively using 26 and 36 samples. The path of Fig. 4.18a takes the AUV through a cluster of obstacles, whereas the path of Fig. 4.18b takes a path that largely avoids the obstacles. Even though Fig. 4.18a corresponds to a sample number of 36, referring to Fig. 4.17, its computation time of 0.29s is smaller than that for Fig. 4.18b, which corresponds to a sample number of 26 and has a computation time of 0.5s.

Fig. 4.19 shows how varying the sample number affects the cost (i.e. distance). The cost is larger in the smaller sample numbers 10 and 12. Afterwards, the variation in the cost is small.

4.4.2 Effect of Grid Resolution

The grid size, which is the resolution of the implicit state grid, is a tuning parameter that can also affect the computation time. The sampling number was held constant at 25. Again this tuning parameter is not monotonic with respect to the computation time as depicted in Fig. 4.20. The larger the grid size the higher the possibility that two nodes are considered as the same state, which leads to more sampling of the input space and an increased computation time. When choosing the grid resolution, it is important to recognize
Figure 4.17: The effect of sample size on computation time.

Figure 4.18: (a) Scenario with sample number = 26, (b) Scenario with sample number = 36
Figure 4.19: The effect of sample size on path cost.

that increasing the grid size tends to lead to higher cost solutions as depicted in Fig. 4.21.
Figure 4.20: The effect of grid size on computation time.

Figure 4.21: The effect of grid size on path cost.
CHAPTER 5
CONCLUSIONS

Section 5.1 provides the dissertation conclusion and ideas for future work are given in Section 5.2.

5.1 Summary

Sampling Based Model Predictive Control has been shown to effectively generate a path for an AUV in the presence of a number of nonlinear constraints. This new method can expand the AUV path planning literature to include a method that can determine a 3D path employing a model of the vehicle. Even though SBMPC was applied to AUVs in this research it can also add to the NMPC literature since it is a general fast NMPC method that is useful for nonlinear systems and/or nonlinear constraints. SBMPC is a NMPC method that exploits sampling-based concepts from the robotics literature along with the $LPA^*$ incremental optimization algorithm from the AI literature to achieve the goal of quickly and simultaneously determining the control updates and paths while avoiding local minima. The SBMPC solution is globally optimal subject to the sampling . As shown in Fig. 5.1, SBMPC may not determine the true global minimum however it can come close and not get stuck in the local minima as do NMPC methods that use gradient-based optimization.

SBMPC and traditional NMPC were shown to have several similarities and differences. Their similarities include the fact that for both methods the system inputs are the primary optimization variables, a model is used to predict future outputs, and they both employ a
Figure 5.1: SBMPC finds the global solution subject to the sampling used and hence avoids local minima.

cost function based on the cost of past trajectories and an estimate of future trajectories to the goal. The differences are the method each uses in the optimization phase, the use of sampling, the way each incorporates the cost function to determine the control input, and the number and type of parameters used as tuning parameters.

SBMPC was compared to traditional NMPC in a cluttered environment with many local minima. It was shown that SBMPC had more reliable traversals to the goal in a shorter computation time than traditional NMPC. Traditional NMPC based on nonlinear programming sometimes yielded shorter distance paths (i.e. lower costs), but this comes at the price of a much higher CPU time and less than 10% success rate to the goal. Out of all the simulation runs there was only one case when SBMPC was not successful at reaching the goal, which proved the importance of properly tuning SBMPC. There are two main tuning parameters in SBMPC, the sampling number and the grid resolution. Each has a nonlinear trend with respect to computation time. The best value of the tuning parameters depend on the area the vehicle is traversing. As a result, the tuning parameters can effect
the CPU time and the optimality of the path. In general, the computation time is effected by the sampling number, the grid resolution and the number of obstacles close to the path chosen by SBMPC.

Since an AUV may have to encounter various scenarios while completing underwater missions, the 3D kinematic model was used to successfully determine paths to traverse random underwater areas. Then the NPS AUV II decoupled model and full dynamic model was applied to show how a different path is determined because the dynamic model considered different aspects than the kinematic and decoupled model. Even in the presence of several nonlinear constraints SBMPC with a decoupled and full dynamic model was capable of determining a feasible path in the presence of multiple obstacles. In the steep hill scenario, the decoupled model determined a path, however because of the AUVs physical limitation the dynamic model was able to determine that the vehicle was too close to gain the momentum needed to climb the hill. All in all, both the decoupled model and full dynamic model provide paths in relatively small computation times (e.g., [0.13 0.46]s).

5.2 Future Works

There are several future goals of this research. The first future work is to develop the replanning feature of SBMPC. The ability to replan is essential in complex underwater environments. SBMPC utilizes $LPA^+$ which allows quick replanning of the path without having to completely restart the planning process when new information is obtained or changes in the environment occur. Only the nodes that are affected by a change in the environment must be reevaluated. This reduces the computation time and aids the method in achieving fast computation times. Once the replanning feature of SBMPC is put into place, scenarios that include disturbance, model mismatch, unknown obstacles and moving obstacles can be examined to test more real life underwater situations. The algorithm will also be in a framework to be more comparable with traditional NMPC that only takes the first input and replans at every time step to create a more robust controller.
The second future goal is to incorporate the closed loop model of the AUV in SBMPC as the planning model. A closed loop model incorporates the control system of the vehicle. This produces a more robust system that is less sensitive to model mismatch or disturbances. For this research, as shown in Fig. 5.2, the sliding mode controller (SMC) will be implemented. SMC is commonly used with AUVs because of its ability to handle the highly nonlinear nature of the vehicle. Fig. 5.2 depicts the closed loop paradigm.

In order to illustrate why the closed loop paradigm is less sensitive to uncertainty, assume SMC provides a gain $K$ and the AUV is a linear system represented by $G$ and some uncertainty in the system $\Delta G$. Then the closed loop paradigm yields

$$
y = (G + \Delta G)f + (G + \Delta G)u
$$

$$
y = (G + \Delta G)f + (G + \Delta G)[K(r - y)]
$$

$$
y = (G + \Delta G)f + (G + \Delta G)Kr - (G + \Delta G)Ky
$$

$$
[1 + (G + \Delta G)]Ky = (G + \Delta G)f + (G + \Delta G)Kr
$$

$$
y = \frac{(G + \Delta G)f}{[1 + (G + \Delta G)K]} + \frac{(G + \Delta G)Kr}{[1 + (G + \Delta G)K]}
$$

(5.1)

In (5.1), if $K$ is large, the right term goes to 1 and the left term goes to $\frac{1}{K}$. This reduces the...
controller sensitivity to uncertainty in the AUV system, $\Delta G$, and now it is more dependent on the gain $K$.

In this dissertation, either a kinematic model, decoupled dynamic model or full dynamic model was used to predict the trajectory to the goal. However in the future, a combination of the models can be applied to further reduce the computation time. For example, the full dynamic model can be used to predict the AUVs motion a preset distance in the future and the kinematic model is used from that distance to the goal.

The fourth concept that needs to be addressed in the future is the generalization of the SBMPC code. Once SBMPC was initially developed and tested several features to be added or modified were identified. Presently, obstacles are depicted as circles or spheres, which limits the scenarios that can be tested. It is desired to expand the collision detection feature to test for polygons, ellipses, etc., so more situations can be addressed. The large SBMPC computation time in Section 4.2.2 could be avoided if only one obstacle was used to represent the cliff. However, due to the limitation of how the obstacles are represented in the current version of the SBMPC algorithm, it was necessary to use 39 obstacles.

Currently, there is a major difference between the control update selected from one time step to the next time step which produces up and down control movement as shown in Fig 5.3a. In real hardware implementation this can cause wear and possible breakage of the actuators. To remedy this there has to be a feature to limit the control value from one time step to the next. It is desired to restrict the change in control $\Delta u$ instead of simply the control $u$. Consequently, $\Delta u$ becomes the input sampled. Restricting the range of the samples of $\Delta u$ will avoid big jumps in the control values as depicted in Fig 5.3b.

Another future goal is considering the orientation of the vehicle in the goal region. Currently, SBMPC is only concerned with the position with respect to reaching the goal. However, after the AUV returns from certain missions homing and docking maneuvers are necessary and the posture of the vehicle is relevant. One way to account for this is to create a funnel output constraint or “virtual obstacle” as shown in Fig. 5.4 that forces the vehicle
Figure 5.3: (a) Example of sampling the control, $u$, (b) Example of sampling the change in control, $\Delta u$

to have the desired orientation when it reaches the goal region.

In addition, there is no feature in place that guarantees the vehicle will come to a complete stop once it reaches the goal region. A funnel output constraint can also help with this limitation of SBMPC. However, another method [20] has been successfully used with robot manipulators. It utilizes a heuristic based on the solution of a minimum time control problem and ensures an efficient computation of trajectories that end at zero velocity.
Figure 5.4: Funnel output constraint, a “virtual obstacle”
# APPENDIX A

## APPENDIX

Table A.1: Simulation Parameters for Dynamic Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( W = 53.4 , \text{kN} )</td>
<td>( B = 53.4 , \text{kN} )</td>
</tr>
<tr>
<td>( L = 5.3 , \text{m} )</td>
<td>( I_y = 13587 , \text{Nm}^2 )</td>
</tr>
<tr>
<td>( x_B = 0.0 )</td>
<td>( y_B = 0.0 )</td>
</tr>
<tr>
<td>( m = 5454.54 , \text{kg} )</td>
<td>( X_{pp} = 7.0 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( X_{pr} = 7.5 , \text{e}^{-4} )</td>
<td>( X_{qg} = -1.5 , \text{e}^{-2} )</td>
</tr>
<tr>
<td>( X_{r} = 1.2 , \text{e}^{-3} )</td>
<td>( Y_{pq} = 4.0 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( Z_{q} = -6.8 , \text{e}^{-3} )</td>
<td>( Z_{pp} = 1.3 , \text{e}^{-4} )</td>
</tr>
<tr>
<td>( Z_{w} = -2.4 , \text{e}^{-1} )</td>
<td>( Z_{q} = -1.4 , \text{e}^{-1} )</td>
</tr>
<tr>
<td>( Z_{w} = -3.0 , \text{e}^{-13} )</td>
<td>( Z_{w} = -6.8 , \text{e}^{-2} )</td>
</tr>
<tr>
<td>( Z_{qn} = -2.9 , \text{e}^{-3} )</td>
<td>( Z_{wn} = -5.1 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( K_{r} = -3.4 , \text{e}^{-5} )</td>
<td>( K_{pq} = -6.9 , \text{e}^{-5} )</td>
</tr>
<tr>
<td>( K_{r} = -1.1 , \text{e}^{-2} )</td>
<td>( K_{r} = -8.4 , \text{e}^{-4} )</td>
</tr>
<tr>
<td>( K_{r} = 1.4 , \text{e}^{-2} )</td>
<td>( K_{r} = 3.1 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( K_{pp} = -5.7 , \text{e}^{-4} )</td>
<td>( K_{pp} = 0.0 )</td>
</tr>
<tr>
<td>( M_{rr} = 5.0 , \text{e}^{-3} )</td>
<td>( M_{rr} = -2.9 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( M_{rr} = 1.2 , \text{e}^{-3} )</td>
<td>( M_{r} = -3.4 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( M_{rr} = 5.0 , \text{e}^{-3} )</td>
<td>( M_{rr} = -2.9 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( M_{rr} = 1.2 , \text{e}^{-3} )</td>
<td>( M_{rr} = -3.4 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( M_{rr} = 5.0 , \text{e}^{-3} )</td>
<td>( M_{rr} = -2.9 , \text{e}^{-3} )</td>
</tr>
<tr>
<td>( M_{rr} = 1.2 , \text{e}^{-3} )</td>
<td>( M_{rr} = -3.4 , \text{e}^{-3} )</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>$N_{evr}$</th>
<th>$N_{dr}$</th>
<th>$N_{prop}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-2.7 \times 10^{-2}$</td>
<td>$-1.3 \times 10^{-2}$</td>
<td>0.0</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


94


BIOGRAPHICAL SKETCH

Charmane Venda Caldwell was born in Jacksonville, Florida. She graduated from Stanton College Preparatory High School on June 1994. Charmane entered Florida State University the Fall of 1994 where she declared Electrical Engineering as her major. She received her Bachelors of Science Degree in Spring of 1999. She continued her education at Florida State University, later receiving her Masters of Science Degree in Electrical Engineering the Summer of 2002 and Philosophy Doctorate in Electrical Engineering the Summer of 2011.

Throughout her duration at the FAMU-FSU College of Engineering, she was involved in the National Society of Black Engineers and held several positions such as Chapter Membership Chair, President, Regional Public Relations Chair and Pre-College Initiative Chair. She currently serves as Director of ProAMS (Proficiency in and Appreciation of Mathematics and Science), a K-12 outreach program of the FSU Research Center, CISCOR (Center for Intelligent Systems Control and Robotics). The objective is to expose K-12 students, especially those with disadvantaged backgrounds, to hands-on activities related to math and science and presentations from real-life scientists, mathematicians and engineers.