AUTONOMOUS MOTION PLANNING USING A PREDICTIVE TEMPORAL METHOD

By

ERIC L. THORN, JR.

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

To my parents

ACKNOWLEDGMENTS

I would like to begin by thanking my family for supporting me and being patient through this long process that began almost 10 years ago as I came to the University of Florida as an undergraduate. They never placed any undue expectations on me and kept me motivated by their interest in the projects I worked on.

I would also like to express my gratitude to my advisor Dr. Carl Crane for his guidance and support from the first day I set foot in CIMAR throughout my entire graduate education. Likewise, I would like to thank my committee, Dr. Antonio Arroyo, Dr. Douglas Dankel, Dr. John Schueller, and Dr. Gloria Wiens for their valuable input and guidance the past few years. This research and all of CIMAR's unmanned vehicle work was made possible by the support of the Air Force Research Lab at Tyndall Air Force Base in Panama City, Florida, so I would like to extend a special thanks to the staff that I was able to work with out there.

Finally, I would like to thank all of my co-workers, and more importantly, friends at CIMAR. Nothing brings people together more than shared experiences, and we certainly shared some experiences. Along with the interesting projects I was fortunate enough to have worked on, it was them that made coming to work every day enjoyable.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS 4

LIST OF TABLES 7

LIST OF FIGURES 8

LIST OF ACRONYMS 10

ABSTRACT 12

CHAPTER

1 INTRODUCTION 14

Background 14
Focus 17
Problem Statement 19
Motivation 19

2 REVIEW OF LITERATURE 24

Path Planning in Dynamic Environments 25
Replanning Algorithms 25
Velocity Obstacles 27
Dynamic Window 28
Path-Velocity Decomposition 30
Artificial Potential Fields 32
Probabilistic Roadmaps 33
Path Planning with Moving Obstacle Motion Prediction 36
Regression Methods 36
Bayesian Methods 38
Markov Methods 39
Neural Networks 41
Dynamic Environment Representation 42
State-Time Graph 43
Temporal Geographical Information Systems 45
Temporal Occupancy Grid 46
Autonomous Target Interception 48

3 THEORETICAL APPROACH 50

Motion Planning Overview 51
Temporal Grid 54
Obstacle Motion Prediction 57

Temporal Motion Planning 60

4 IMPLEMENTATION DETAILS 70

Urban NaviGator 70
Architecture 71
Hardware 71
Software 73
Traditional Traversability Grid 74
Temporal Grid Creator 77
Obstacle Position Data Collection 78
Search Parameters 79
Temporal Grid 83
Moving Obstacle Prediction 88
Temporal Motion Planner 90
Target Interception Application 93

5 TESTING AND RESULTS 115

Test Plan 115
Following Behavior 116
Obstacle Field Behavior 117
Target Interception 118
Test Metrics 119
Simulation 121
Following Behavior 121
Obstacle Field Behavior 125
Target Interception 128
Testing Summary 130

6 CONCLUSIONS AND FUTURE WORK 156

Future Work 156
Conclusions 160

LIST OF REFERENCES 162

BIOGRAPHICAL SKETCH 169

LIST OF TABLES

Table page

4-1 Temporal layer size parameters for $N_l = 1 \ldots 10$.....113

4-2 Distance horizon values for each temporal layer for initial temporal grid testing.....113

4-3 Grid creation times (in seconds) for $N_l = 1 \ldots 10$.....113

4-4 Grid cloning times (in seconds) for $N_l = 1 \ldots 10$.....114

4-5 Memory requirements (in bytes) for $N_l = 1 \ldots 10$. 114

5-1 Following behavior test plan.....152

5-2 Obstacle field behavior test plan.....153

5-3 Target Interception test plan. 154

5-4 Sample of predicted positions and velocities for following behavior testing. 155

5-5 Sample of predicted positions and velocities for obstacle field behavior testing. 155

5-6 Prediction times for three test scenarios at various temporal. 155

LIST OF FIGURES

Figure		page
1-1	The Urban NaviGator: Team Gator Nation's Urban Challenge entry.	23
3-1	Breadth-first search state exploration order.	65
3-2	Depth-first search state exploration order.	65
3-3	Three-dimensional temporal grid structure.	66
3-4	Temporal grid tree visualization.	66
3-5	Sample output traversability grid from motion planning.	67
3-6	Discrete exploration steps of motion planner with no obstacle interference.	68
3-7	Discrete exploration steps of motion planner with obstacle interference.	69
4-1	Urban NaviGator system architecture diagram.	95
4-2	Traversability grid representation of robot's environment.	96
4-3	Formation of torus buffer	97
4-4	Motion planning algorithm input traversability grid.	98
4-5	Vehicle kinematic model geometry.	99
4-6	Illustration of search algorithm node expansion showing key search parameters.	99
4-7	Pyramidal-shaped optimized temporal grid structure.	100
4-8	Optimized temporal grid layers showing static obstacle appearing.	101
4-9	Formation of temporal torus buffer.	102
4-10	Creation times for TG, full temporal TG and optimized temporal TG.	103
4-11	Cloning times for TG, full temporal TG and optimized temporal TG.	104
4-12	Memory requirements for TG, full temporal TG and optimized temporal TG.	105
4-13	Prediction visualization showing polynomial curve fit and extrapolated data.	106
4-14	Full temporal grid layers showing predicted positions of obstacle.	107
4-15	Optimized temporal grid layers showing predicted positions of obstacle.	108

4-16 Future predicted positions of obstacle in single traversability grid. 109
4-17 Optimized temporal grid showing simple dilation of predicted obstacle positions.....110
4-18 Full temporal grid layers showing expansion of search tree with obstacle. 111
4-19 Optimized temporal grid layers showing expansion of search tree with obstacle. 112
5-1 Gainesville Raceway pit area with points defining test areas. 133
5-2 Straight road segment selected for following behavior test with defining waypoints. 133
5-3 Open unstructured area with defining waypoints and perimeter points drawn. 134
5-4 Sample following behavior control test results. 135
5-5 Sample obstacle field behavior control test results. 136
5-6 Sample full temporal grid layers with predicted position of followed object.....137
5-7 Sample optimized temporal layers of following behavior test temporal grid. 138
5-8 Sample output temporal grid from following behavior. 139
5-9 Number of expanded search nodes for following operating behavior tests. 140
5-10 Solution trajectory costs for the following behavior tests. 141
5-11 Steering commands for following behavior tests.....142
5-12 Sample full temporal layers showing predicted positions of obstacle. 143
5-13 Sample optimized temporal grid layers from obstacle field behavior test.....144
5-14 Steering commands for obstacle field behavior tests. 145
5-15 Sample output temporal grid of obstacle field behavior simulation. 146
5-16 Path deviation results for obstacle field behavior tests. 147
5-17 Sample full temporal grid layers showing progression of target object. 148
5-18 Sample optimized temporal grid layers showing progression of target object. 149
5-19 Sample full temporal grid output of motion planner during target interception test. 150
5-20 Target and solution path heading angles for target interception test. 151

LIST OF ACRONYMS

AFRL	Air Force Research Lab
ANN	Artificial neural network
APPE	Adaptive prediction planning and execution
ARM	Autoregressive Model
BOF	Bayesian occupancy filter
CCD	Charged couple device
CIMAR	Center for Intelligent Machines and Robotics
CLF	Control Lyapunov function
DARPA	Defense Advanced Research Projects Agency
DPA	Deterministic prediction algorithm
DUC	DARPA Urban Challenge
FADPRM	Flexible anytime dynamic A* probabilistic roadmap
GIS	Geographic information system
GPOS	Global position and orientation sensor
GPS	Global positioning system
GSL	GNU science library
HMM	Hidden Markov model
KVM	Keyboard-video-mouse
LADAR	Laser detection and ranging
MDF	Mission data file
MO	Moving obstacle sensor
NFM	North finding module
NLVO	Nonlinear velocity obstacle

PD	Primitive driver
PMP	Partial motion planning/planner
POMDP	Partially observable Markov decision process
PRM	Probabilistic roadmap
PTMP	Predictive temporal motion planning/planner
PVO	Probabilistic velocity obstacle
RBFNN	Radial-basis-function neural network
REDCAR	Remote detection challenge and response
RHC	Receding horizon control/controller
RNDF	Route network definition file
RRF	Reconfigurable random forest
RRT	Rapidly-exploring random tree
SARB	Smart Arbiter
SLAM	Simultaneous localization and mapping
SSRMS	Space station remote manipulator system
TG	Traversability grid
TGC	Temporal grid creator
THH	Toyota Highlander Hybrid
TOG	Temporal occupancy grid
UTM	Universal Transverse Mercator
VO	Velocity obstacle

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AUTONOMOUS MOTION PLANNING USING A PREDICTIVE TEMPORAL METHOD

By

Eric L. Thorn, Jr.

December 2009

Chair: Carl Crane
Major: Mechanical Engineering

The introduction of moving obstacles into a robot's environment presents added complexity to the motion planning task. This dissertation examines the need for and development of a representation which incorporates the dynamic nature of the environment and presents a novel motion planning method which utilizes this representation to facilitate the generation of optimal trajectories among moving obstacles, termed the predictive temporal motion planning (PTMP) method. This new method provides an advanced approach to the problem of generating solution trajectories in dynamic environments by elegantly connecting the tasks of obstacle detection and prediction, environment mapping and motion planning.

The dynamic environmental representation takes the form of a typical grid which is extended into the time dimension by adding temporal layers to the grid structure. The layers of this temporal grid represent distinct time-steps into the future. These time-steps are determined by considering how the motion planning algorithm calculates its discrete control commands. Obstacle motion prediction is incorporated into the temporal grid by estimating future positions of moving obstacles and displaying these estimates in the layer of the temporal grid associated with the prediction times.

The new motion planning method then can use this predictive temporal grid to investigate potential control input sequences to generate an optimal trajectory to achieve its goal. As the algorithm evaluates potential control commands at various time-steps in the future, it does so by exploring the various temporal layers of the new grid structure corresponding to these distinct control times. By considering the estimated future motions of any obstacles, the motion planning algorithm can more intelligently calculate its control sequences to avoid these objects in an efficient manner.

The research presented covers the theory of this new method and a specific implementation on an unmanned ground vehicle platform at the University of Florida's Center for Intelligent Machines and Robotics (CIMAR). Results from simulation have shown that the PTMP method is viable and advantageous for the motion planning required by robotic systems in dynamic environments.

CHAPTER 1
INTRODUCTION

The field of robotics, while not so young anymore, is still experiencing rapid growth and development. Advanced areas of research are being presented every year, and robots continue to be utilized in new and novel ways. Unmanned vehicles are one specific area of robotics that is receiving a lot of attention. These intelligent mobile platforms are highly attractive since they can be placed in uncertain or even dangerous environments to accomplish missions that would be unsafe for humans. Each new application of mobile robots typically involves increasingly complex scenarios that require the robot to sense its environment and execute behaviors with more precision and efficiency than before. For unmanned ground robots, navigation through dynamic urban environments represents one of these increasingly complex scenarios. The research presented in this document continues this ongoing trend of advancing the field, specifically in dealing with this issue of navigating in areas with moving obstacles.

Background

Unmanned robots exist in many different shapes and sizes and varying degrees of intelligence and capability. Unmanned ground vehicles can operate on rough and rugged terrain, inside of buildings where hostile conditions may exist, and in tight spaces that would otherwise be inaccessible for humans. Unmanned underwater vehicles can be deployed by the military to sneak undetected under the surface if necessary, or can be used by scientists to analyze the ocean floor mapping or drilling purposes. Unmanned aerial vehicles range from models with one foot wingspans that can be used for urban search and rescue, to the full-sized Predators currently deployed for reconnaissance and precision strikes in Iraq and Afghanistan.

Unmanned vehicles can be further classified as either teleoperated or fully autonomous depending on the level of intelligence. Teleoperated vehicles require human input, but remotely

from a distance. They are regularly used by the military and bomb squads across the country to deal with potential explosive devices. The robot can be remotely driven to a possible unexploded ordinance, and then through the use of onboard mechanisms, can detonate or dispose of the threat. They can also be driven through dangerous rubble to search for survivors of an earthquake or missile attack. On the other hand, fully autonomous robots require no human interaction aside from an initialization process. This type of vehicle is appealing to the military because of the ability of the robot to make decisions and take action on its own keeps soldiers out of harm's way. These robots are incredibly complex and require hundreds or even thousands of components working together flawlessly to be successful. The presented research focuses on these autonomous unmanned vehicles, although it could potentially be used on teleoperated vehicles and other types of robotic systems as well.

Fully autonomous vehicles require significant cognitive abilities. Given only some goal to achieve, the robot must localize itself, put together a representation of its surroundings, plan a course of action through its surroundings to achieve its goal, and then act upon this plan. The problem of localization requires the robot to determine where it is and where its destination is in a particular reference frame. One possible solution is the process known as simultaneous localization and mapping, or SLAM (Leonard & Durrant-Whyte, 1990). SLAM involves creating an environmental map based on sensor data, while concurrently keeping track of the robot's current position. Another common approach is through the use of the Global Positioning System satellite network coupled with an inertial navigation system which can track the motion of the robot in the absence of accurate data from the satellites. This localizes the robot in a global frame of reference using latitudes, longitudes, and elevations, but can be transformed into other frames of reference.

In addition to knowing its location in its reference frame, the robot must know what its surrounding look like. There are a variety of sensors capable of putting together a representation of the environment depending on the application, and usually, a collection of these sensors is utilized. For underwater robotic vehicles, sonar might be the sensor of choice. Cameras and laser scanners are just a few examples of hardware that can be used to classify the terrain in front of and around a ground robot, and to locate and track static and dynamic obstacles near the vehicle. And for unmanned aerial vehicles, radar and thermal cameras may be best suited to provide the required information.

This raw sensor data must then be combined to create a representation that can be useful to the robot. A common type of representation is a grid map, which rasterizes the surrounding environment into cells and classifies each cell to some degree of occupancy according to the sensor data associated with that cell. Vector representation is another way to describe the robot's surroundings. This method is more closely related to the raw data provided by the sensors in that it describes the environment as a series of distances and angles. Regardless of the technique, this knowledge of the surrounding area is crucial for the issue of maneuvering the robot from its current position to the goal.

The determination of deliberative control inputs to accomplish this task of acquiring a goal can be completed by a variety of means and is typically termed path planning or motion planning. Planning algorithms include, but are not limited to, the rather simplistic vector driver algorithms which compute the angle between the current vehicle heading and the heading to the goal and relate it to a steering input to change the heading, complex model predictive control methods which take the kinematics and dynamics of the vehicle into account, and heuristic algorithms which efficiently estimate the optimal path and generate control inputs. Presently,

there are many algorithms in use that are capable of calculating trajectories that are optimal in terms of travel time, travel distance, or some other metric that differentiates one potential path from another. The work to be presented expands on an existing path planning method.

Finally, once a set of control is calculated, the inputs must be implemented through actuation of the vehicle. Typically, the control inputs consist of steering commands to change the heading of the vehicle and acceleration or deceleration commands to change the velocity of the vehicle. These commands are usually implemented through mechanical or electrical means. Steering commands may be instituted as changes in pressure for hydraulically driven vehicles, or position changes of a motor connected to the steering column of an Ackerman-steered vehicle. Velocity changes can be implemented by means of an electrical signal being sent to a computer for vehicles with electronically controlled acceleration and deceleration, or by pushing or pulling on a brake or throttle pedal using a motor.

This sequence of events must be repeatedly carried out as new information about the state of the vehicle and the state of its surroundings changes. Depending on the type of vehicle and application, this could require considerable computational resources to process all the incoming data and run the algorithms necessary to generate the output commands. This brief summary only begins to scratch the surface when describing the complexity required in operating unmanned robotic vehicles.

Focus

Past research indicates many motion planning algorithms are adept at finding optimal paths through static environments. However, the introduction of dynamic objects which have uncertain motions into the environment complicates the planning process. While sensing technology is fairly reliable in terms of detecting and mapping static objects, the issue of detecting and tracking moving objects has only recently begun to be investigated. Furthermore,

the ability of an autonomous robot to plan a trajectory through an environment where dynamic obstacles are present is a very young area of research.

One of the main issues that arise in such a situation is the modeling of a moving object in the environmental representation through which the robot plans. In the case of a grid map, the question of how to describe the motion of the object in the cells of the grid is important. Many existing motion planners run their planning algorithms repeatedly at some regular time interval and treat each instance of a grid as static. As new sensor data is received, the previous map is updated for the next iteration of the algorithm. In this sense, the motion of moving objects can be represented over a number of successive grids, with the objects being treated as static in each individual grid. This will allow the motion planner to find a complete trajectory from the current position to the goal position that avoids object for this instance in time, assuming a complete trajectory is attainable. However, this can be misleading as the planner has no intuition as to what the future motions of the moving object will be, and could lead to the robot planning a path that would ultimately lead to a collision, or planning a highly suboptimal path to avoid the object as time progresses.

The ability to predict the motion of dynamic objects could aid in solving this problem. There are several prediction algorithms that could be used to describe the future states of a moving object in the environment of the robot. If a motion planning algorithm could know how an object's position will change over some period of time in the future with a reasonable degree of certainty, it could make more intelligent choices in terms of steering and velocity commands. And even if the exact future position of the object cannot be precisely defined, a probability distribution of future positions could still be useful.

Problem Statement

After the aforementioned discussion of predicting the future motions of moving obstacles, the question of how to describe the predicted motion of a dynamic object in the representation of the robot's surroundings arises. The introduction of time-varying data in a data structure that represents the robot's environment brings new challenges and new possibilities to the robot motion planning task. Additionally, the development of a motion planner that can plan a trajectory through this type of temporal data structure is desirable. While the development of a prediction algorithm is outside the scope of this study, the research presented in this document addresses the issue of how to use this predicted data to plan an optimal path for an autonomous robot by focusing on the development of a grid map which can take advantage of this data to store occupancy information for a range of time, rather than for a single instant, and the development of a motion planner which can plan an optimal trajectory for the robot using the spatial and time-dependent data contained within this new grid. Even though the central aim of this work was to develop a temporal motion planning algorithm for obstacle avoidance purposes, it also has applications such as target interception, which will be presented, in which an obstacle is treated as a goal to attain rather than an obstacle to avoid.

Motivation

The presented research advances the capabilities of autonomous ground robots that have been developed at the Center for Intelligent Machines and Robotics (CIMAR) at the University of Florida and at the Air Force Research Lab (AFRL) at Tyndall Air Force Base. CIMAR and AFRL have been collaborating on projects for many years and have successfully automated vehicles for such applications as clearing mine fields, detecting unexploded bombs and patrolling the perimeters of military installations. Motion planning development is crucial toward the success of this collaboration and the issue of dynamic obstacles is prevalent in current projects.

At Tyndall Air Force Base, research is conducted with the aim of using ground robots for military base perimeter defense as part of the Remote Detection Challenge and Response (REDCAR) project. A rugged mobile platform has been developed that can be used autonomously or through teleoperation. Upon detection of an intruder while autonomously patrolling a perimeter, the vehicle can switched into teleoperated mode and can be directed to intercept the target and neutralize it by lethal or non-lethal means. With an advanced target tracking system and a reliable target motion predictor coupled with the developed temporal motion planner, the robot would be able to chase down and intercept a target without human intervention. On the other hand, the planner would also avoid any other dynamic objects within the environment that are not identified as a target.

The most recent autonomous vehicle research at CIMAR has focused on a large-scale autonomous vehicle for navigation in an urban environment. The motivation for this work was entry into a robotics competition coordinated by the Defense Advanced Research Projects Agency (DARPA) known as the DARPA Urban Challenge (DUC) and held in southern California in November of 2007. This most recent contest built on the success of the previous robotic challenges in 2004 and 2005. While the previous DARPA competitions were geared toward off-road navigation, the Urban Challenge, as the name suggests, focused on autonomous driving capabilities in urban environments in the presence of other moving vehicles, including other robots. Robots were required to exhibit a long list of complex driving behaviors that are commonplace when a human is driving a vehicle. Being that the competition was a race, competitors had to balance the desire for speed with the necessity of obeying all traffic rules and being safe. Some of the basic navigation and traffic requirements included: remaining within the desired lane, obeying speed limits, maintaining a safe following distance, passing slow moving

or stopped cars, completing a U-turn, and obeying intersection precedence. Required behaviors for advance navigation and traffic included: navigating an obstacle field, parking, merging with moving traffic, and defensive driving.

CIMAR's entry into the challenge was an automated 2007 Toyota Highlander Hybrid, as seen in Figure 1-1. The vehicle was outfitted with a sensor network composed of global positioning system (GPS) receivers coupled with an inertial navigation system, eight laser range scanners, and four cameras. Motors were attached to the steering column and shifting mechanism for autonomous steering and gear shifting. The existing drive-by-wire system for throttle and brake was utilized for autonomous acceleration and deceleration. A custom built computer rack was populated with twelve dual-core computers, of which ten were used.

The motion planner utilized by CIMAR for the DUC is a form of receding horizon controller running a modified A* search algorithm to find the optimal path through a traversability grid of fused sensor data (Crane et al., 2005). The A* algorithm creates a search tree through the grid map, evaluates the cost of the nodes in the tree, and uses these costs heuristically searches through the tree to find a least cost path to a predetermined goal. The nodes of the tree are vehicle state representations which are expanded through a kinematic vehicle model. The cost of a node depends on several factors, including distance from the node to the center of the perceived lane, distance from the node to the goal, and the traversability values associated with the cells between the node of interest and its predecessor node. Being a best-first search algorithm, the A* algorithm keeps track of the costs of all nodes on the tree and selects the lowest cost node for expansion (Hart, Nilsson, & Raphael, 1968).

Several of the requirements of the DUC have motivated the presented research topic. The inevitability of the robot interacting with other moving obstacles, and potentially with other

intelligent vehicles, demonstrates the need for efficient and optimal motion planners that can take the predicted motion of dynamic objects into consideration. The path planning algorithm described previously is not sufficient to deal with dynamic objects in the robot's environment. For most behaviors developed for the DUC, moving obstacles were handled at a higher level in the system. For example, for behaviors associated with road following, intersections, and passing, any moving obstacles were simply removed from the grid map to avoid interfering with the planning algorithm's stability. Changes in velocity were instead commanded by other components of the system to slow the robot to a stop if following too closely to a moving object, or to speed up to pass a moving object. For behaviors associated with obstacle fields and parking lots, the motion planner simply treated moving objects as static for each particular planning instance. With the ability to represent the predicted motion of dynamic objects in a grid map, and with the ability to plan motions through this type of grid, the trajectories planned by the Urban NaviGator will be more optimal with regards to dealing with these moving obstacles. The facilities and resources at CIMAR allowed an in depth study of this claim and improved the chances of this type of technology coming to fruition.

This chapter provides an introduction and some background information to the task of motion planning in dynamic environments that is investigated in this dissertation. It outlines the problem statement in three main areas: generating a temporal representation of the environment, incorporating predictive motion models of moving objects into this temporal representation, and developing a temporal motion planning technique. Lastly, it provides specific motivation for addressing this type of problem in the form of the requirements of autonomous ground vehicle navigation in dynamic environments. The following chapter reviews previous research conducted that deals with the three areas of the problem statement.

Figure 1-1. The Urban NaviGator: Team Gator Nation's Urban Challenge entry.

CHAPTER 2
REVIEW OF LITERATURE

A review of published research was conducted to get an idea of how different areas of this problem have been solved, thereby providing a means of comparing and contrasting the presented work. While the issue of planning trajectories through dynamic environments is a relatively new area of research in the field of autonomous robotics, several different approaches were studied. The review began by investigating how others had approached motion planning in the presence of moving obstacles. Many of the techniques regarded the world as static and projected forbidden regions from the instantaneous location of the objects. Some of them addressed the issue by planning in the velocity space and adjusting angular and linear velocities rather than planning in a spatial domain. The addition of the time dimension into the vehicle's configuration space was also analyzed, but again, in a static representation. Several replanning algorithms which maintain the previous solution path and only alter sections of the path that are affected by changing arc costs due to objects moving in the environment were also studied.

Next, planning methods that utilized motion prediction of dynamic obstacles were investigated. Many of these techniques were only capable of predicting the states of the objects a short time in the future, which may be useful when the robot is close to any of these objects, but it is desirable to know well ahead of time if any evasive action may be required to avoid collisions. Several methods studied required a training period, where the motion patterns of obstacles were observed for a time to aid the prediction algorithm. While this may be useful for highly structured environments such as warehouses or factories where moving objects' paths tend to be repeated, this approach may be misleading or simply not useful when dealing with objects with uncertain motion or other cognitive vehicles.

Time was taken to review ways of representing a robot's environment to see how this could be extended to save temporal data in addition to spatial data. Several methods created a grid structure with a time dimension included, but this temporal dimension represented the history of the environment being described rather than looking in the future. The state-time space of a robot is another representation that was used in a variety of planning algorithms and allows for temporal information to be stored.

Lastly, existing literature dealing with the issue of robotic interception problems was investigated. Many applications of this problem relate to robotic manipulator arms attempting to grasp a moving object where the trajectory of the object is straight-forward and can be easily predicted. The complications associated with the end-effector of the manipulator matching the position, velocity and orientation of the obstacle make this problem difficult to solve. Several techniques associated with mobile ground robotics tracking and catching a moving target were also studied.

Path Planning in Dynamic Environments

Replanning Algorithms

Some motion planning algorithms for autonomous robots are run over and over again at a high rate, planning from scratch each time to take changes in the environment into account. However, several types of algorithms make use of saving the previous solution trajectory from the start point to the goal location and only make minor adjustments to the trajectory as arc costs in the search tree change in the vicinity of the path.

In 1994, Stentz presented a search algorithm that was similar to the A* algorithm, but allowed for arc cost parameters to change during the planning process (Stentz, 1994). The algorithm used "raise states" and "lower states" on the list of nodes waiting for expansion to propagate information about path cost increases and decreases, respectively. When one of these

states was expanded, it passed on the cost changes to its successor nodes. Experimental results showed that the new algorithm, known as the D* algorithm, demonstrated significant speedup compared to a general optimal replanner, and the speedup increased dramatically as the number of cells in the grid increased.

Several extensions of the D* algorithm have been developed over the years. A focusing heuristic was used to effectively narrow the propagation front of the "raise states" and "lower states" in the Focussed D* algorithm, and thus, was shown to be more efficient in replanning (Stentz, 1995). This narrowing occurred as a result of the inclusion of a cost to return to the robot which penalized the wider edges of the propagation front. The modified algorithm proved to be effective when run off-line, resulting in lower planning times than the basic D* algorithm, but had higher planning times when used as an on-line planner.

The Delayed D* algorithm required approximately half the computation of the D* algorithm by including a one-step look-ahead cost, but generated equivalent optimal paths (Ferguson & Stentz, 2005). The look-ahead cost was defined as the minimum of the cost estimates of the successor nodes to the goal. The algorithm was shown to be more efficient since the propagation of some cost changes were ignored as long as possible, while more important changes were propagated.

Finally, the Field D* algorithm addressed the issue of assuming straight-line trajectories from a node to its neighbors (Ferguson & Stentz, 2007). This new algorithm used linear interpolation to produce smoother paths. Simulated robotic experiments showed Field D* generated solutions that were 96% as costly as the original D* algorithm and took 1.7 times as long to calculate.

Velocity Obstacles

Planning in the velocity-space of a robot is a relatively new method of dealing with dynamic obstacles. Just as a robot's environment can be broken up into spatial states for the motion planning algorithm to search through, it can also be discretized into a finite number of linear and angular velocity pairs. But in doing so, the problem of how to facilitate reactive obstacle avoidance during the path planning process arises.

The concept of Velocity Obstacles (VO) was discussed by Fiorini and Shiller as a potential solution to this problem (Fiorini & Shiller, 1998). The process of constructing a VO began by using the relative velocity between the robot and the obstacle to create a collision cone, which was the set of relative velocities that would result in a collision at some time in the future. The VO was then defined as the vector sum of this collision cone with the velocity vector of the obstacle. The VO represented a region in the velocity space of the robot that would lead to a collision with the obstacle within a time horizon. Obstacle avoidance was carried out by creating a set of reachable avoidance velocities defined by the dynamic constraints of the vehicle.

The aforementioned process of creating velocity obstacles assumed the obstacle was moving in a straight line, represented by a constant linear velocity with no angular velocity. However if the obstacle was traveling along an arc, this method could have misrepresented the set of velocities that would result in a collision. This could have led to falsely indicating a collision would occur, when the current trajectory would actually be suitable to avoid the object. This problem was addressed by developing the concept of Non-Linear Velocity Obstacles (NLVO) (Shiller, Large, & Sekhavat, 2001). Again, the NLVO represented the set of robot velocities that resulted in a collision with a particular obstacle, but they took the nonlinear motion of these obstacles into account.

This concept was then implemented in a robot motion planning algorithm and used for obstacle avoidance (Large, Laugier, & Shiller, 2005). As the NLVOs were generated, the algorithm also calculated time-to-collision estimates for each of the NLVOs. An A* algorithm was then run to search the velocity-space of the robot. The two criteria used for sorting and evaluating the set of velocities for the search algorithm were the computed time-to-collision and the time-to-goal.

In 2004, Kluge developed a Probabilistic Velocity Obstacle (PVO) method to take the uncertainty of obstacle shapes and velocities into account (Kluge, 2003). Under the assumption that the obstacles in the environment were also intelligent, a recursive modeling technique was employed to attempt to model the decision making processes of the moving obstacles. With these models, the predicted velocities could then be used to generate the PVOs rather than requiring observation of obstacle velocities.

Dynamic Window

The dynamic window approach was another velocity space search method. It was first developed to control the motion of a robot with a synchro-drive (Fox, Burgard, & Thrun, 1996). The algorithm was well suited for dealing with velocity and acceleration constraints present in all mobile robots. It strived to limit the velocity search space of the motion planner in three steps. The first step reduced the space by allowing only circular trajectories defined by unique translational and rotational velocity pairs. Of these pairs, the second step ensured that only safe velocities were searched by limiting the space to admissible velocities. Specifically, these admissible velocities were velocities that allowed the robot to stop before reaching the closest obstacle on the current path. The last step reduced the space further by creating the dynamic window, which represented the set of admissible velocities that could be reached in a time interval defined by the limited acceleration or deceleration of the robot.

This planning method was then extended in a number of studies to improve its performance. *A priori* maps of the environment were used to aid in avoiding undetectable obstacles that were represented in the maps (Fox, Burgard, Thrun, & Cremers, 1998). Known as the model-based dynamic window approach, this form of the algorithm used metric Markov localization to estimate the robot's position within the map. The reduced dynamic window approach took advantage of the fact that 95% of the time, the highest admissible velocity was selected by only calculating the dynamic window for this speed (Arras, Persson, Tomatis, & Siegwart, 2002). This further diminished the size of the velocity search space, and thus increased the speed and efficiency of the search algorithm. Ogren and Leonard combined the dynamic window approach with Lyapunov nonlinear control theory to generate a convergent dynamic window method (Ogren & Leonard, 2002). A model predictive control law and a control Lyapunov function (CLF) were chosen such that the constrained optimal control problem was satisfied for a short time horizon. This study overcame shortcomings of other dynamic window approaches such as the potential for the solution to diverge or get stuck in a limit cycle.

An elegant combination of the dynamic window approach with the previously discussed Focused D* algorithm was recently developed that generated admissible trajectories for a mobile robot (Seder, Macek, & Petrovic, 2005). This algorithm was implemented on a real mobile robot outfitted with a laser range finder to detect moving obstacles. This method was further extended in such a way that guaranteed solution trajectories that avoided collisions with moving obstacles (Seder & Petrovic, 2007). The concept of "moving cells" was introduced to represent the motion of dynamic objects in the grid map. For each newly occupied grid cell, a motion heading and velocity vector were estimated by some means. A series of potential trajectories were then generated starting at this "moving cell" much in the same way candidate trajectories were

generated for the robot. These "moving cell" trajectories were then checked against the dynamic window trajectories for the robot for collisions.

Path-Velocity Decomposition

The path-velocity decomposition method attempted to solve the motion planning problem in dynamic environments by breaking it down into two sub-problems (Kant & Zucker, 1986). The first sub-problem could be thought of as a traditional path planning process attempting to avoid static obstacles, while ignoring the dynamic ones. The second sub-problem then planned the velocity profile along the solution path of the first sub-problem to deal with moving objects. The path planning process was solved by creating a graph defined by the set of vertices of the static polygonal obstacles and the set of edges among nodes that did not intersect any of the static objects. To find the velocities along this path, moving objects were represented as hyper-volumes that were swept out in space-time. The velocity planner then checked for intersections of these volumes with the path output from the planning problem. Any intersections provided time-varying constraints on the robot's position along the path and allow for a velocity to be selected.

The aforementioned study decomposed the problem to a global path and velocity planner, with no means to consider changes in the environment. Kant and Zucker then extended this concept into a hierarchical planner by adding a local avoidance planner that reacted to changes in the environment detected by sensors (Kant & Zucker, 1988). The constraints resulting from the volumes swept by detected obstacles effectively formed forbidden regions in the path-time space of the robot. The low-level avoidance planner overlaid a repulsive acceleration on the path generated by the global planner that moved the robot away from a detected obstacle. Simulations were run with perfect and imperfect estimation of obstacle velocities and showed

that this hierarchical form of path-velocity decomposition allowed for local avoidance of moving obstacles.

Fraichard and Laugier further developed the method by adding the generation of "adjacent paths" into the path planning process which dealt with cases where a dynamic object followed the robot's nominal path or stopped on top of it (Fraichard & Laugier, 1993). The first step of the path-velocity decomposition algorithm now found a nominal path and calculated a set of paths, which were reachable from the nominal path. These "adjacent paths" were not necessarily collision free with respect to stationary objects. They, therefore, required being checked for collisions, which resulted in a set of "forbidden" intervals along the paths. The second step of the path planning process, denoted as trajectory planning, determined the motion of the robot among the nominal paths and "adjacent paths." The velocity planning stage was carried out in the same manner described above.

While the previous study focused on the spatial planning stage of the path-velocity decomposition, a recent study alternatively focused on the velocity planning stage in a shared workspace where multiple robots may have travelled (Hwang & Ju, 2002). The method classified the moving objects in the environment as controllable or uncontrollable. It was assumed that the global planning problem had been solved for all the controllable robots without any static obstacles interfering with the path. The global path was then decomposed into a series of subtasks, each of which contained a desired goal position and a desired arrival time. Space-time graphs were then generated for each of the subtasks. These space-time graphs were essentially a local mapping of forbidden regions for the controllable objects. An interface propagation method was then used to calculate velocity profiles for the controllable objects to avoid entering forbidden regions.

Artificial Potential Fields

The use of time-varying potential field functions for motion planning was first introduced as an obstacle avoidance technique for robotic manipulators and mobile robots (Khatib, 1986). The method created a virtual force field for the robot to plan through with goal positions being represented by an attractive force and obstacles to be avoided represented as repulsive forces. A resultant force was calculated for the robot with the direction of the force being the desired heading direction for the robot and the magnitude of the force being equivalent to the desired speed. While this original study dealt solely with static obstacles, it was extended to incorporate dynamic ones.

Ko and Lee defined the "avoidability measure" as a way of describing the possibility that the robot collided with an obstacle (Ko & Lee, 1996). This measure was a function of the distance between the robot and the object and the speed of the object relative to the robot. The avoidance measure was inversely proportional to the possibility that the robot and the dynamic obstacle experienced a collision, so it increased as the distance and relative velocity decreased. This study used a virtual distance function, which emphasized the distance metric over the speed, as its "avoidability measure." This function could be tuned so that the robot began avoiding obstacles closer or further away. It was then mapped to a potential force to be used with the traditional potential field method.

This method was extended to account for moving obstacles and moving targets in a number of studies. Ge and Cui defined an attractive potential for the target that took the relative position and velocity of the target with respect to the robot into account (Ge & Cui, 2002). Therefore, as the distance between the robot and target increased, or the target started accelerating away from the robot, the attractive force of the target increased. On the other hand, each moving obstacle was given a repulsive potential that was inversely proportional to the

relative positions and velocities among these obstacles and the robot. This study also addressed local minima issues that arose when an obstacle was between the robot and the target and was moving in the same direction as the two, and when the robot was approaching the goal, but could not reach it because an obstacle was close. Computer simulations and experiments on real mobile robots showed this method to be effective in avoiding moving obstacles while intercepting the target. Another study addressed moving obstacles and targets in a robotic soccer scenario by defining the relative threat function (Cao, Huang, & Zhou, 2006). This threat function was again derived by examining the relative positions and velocities between the robot, target, and moving obstacles. When the robot was within a region of influence of a particular obstacle, the repulsive potential of that particular object acted against the robot. Otherwise, if it is outside the region of influence, the robot ignored the effects of that dynamic obstacle.

Several other methods have been developed to use the potential field method with motion planning among moving obstacles. Poty, Melchior, and Oustaloup used a fractional potential to generate a fractional road to take the danger of each moving object into account (Poty, Melchior, & Oustaloup, 2004). The use of the fractional potential ensured a continuous flow of potential among isolated sources. It then used the method in (Ge & Cui, 2002) to extend the use of fractional potential for motion planning in dynamic environments.

Probabilistic Roadmaps

The concept of probabilistic roadmaps (PRM) was developed by Svestka and Overmars for single car-like robots in a static environment (Svestka & Overmars, 1994) and was later applied to multiple robots (Svestka & Overmars, 1995). An undirected graph was built incrementally by adding random free configurations of the robot to the set of nodes and trying to connect those newly added nodes to a number of chosen nodes already in the existing set. The map was then checked for intersections with known obstacles. Once the roadmap had been generated, it could

be used to solve motion planning problems, referred to as "queries." For the multi-robot situation, a "super-graph" was built from the individual roadmaps of the robots. The areas swept by the robots when moving along a particular edge in their map were checked against each other and disallowed if they intersected. This method was shown to be probabilistically complete, meaning that, given enough time, the planner found a solution to any query if a solution existed.

It is important to note that the previously described studies dealt strictly with static environments and with roadmaps that were generated off-line. This method was extended to dynamic environments, however, by building small roadmaps on-the-fly that connected some initial and final state (Kindel, Hsu, Latombe, & Rock, 2000). This study introduced the use of PRMs in the state-time space of the robot. It also assumed the trajectories of the moving obstacles were known *a priori*. Another study that assumed the motions of the obstacles were known ahead of time used a hierarchical method to search through the roadmap (van den Berg & Overmars, 2005). An A* algorithm was used on the global level to find a trajectory to pass on to the local planner, which used a depth-first search to investigate the edges of the roadmap in state-time space. A geometric property denoted as "expansiveness" was introduced in (Hsu, Kindel, Latombe, & Rock, 2002) and used to show that the probability that this type of planner failed to find a solution trajectory when one existed quickly converged to zero as the number of collision-free samplings of the workspace increased.

An incremental learning approach to the roadmap problem was presented in (Koenig & Likhachev, 2002) to allow on-line management of the roadmap for every planning query. This method differed from traditional probabilistic roadmap methods in that it used a Rapidly-exploring Random Tree (RRT) structure to grow the map. While a traditional RRT algorithm may grow a new RRT for every planning problem, this variation took advantage of

previously “learned” trees for future planning problems. The map structure then became a “forest” of RRTs, termed the Reconfigurable Random Forest (RRF). As multiple planning queries were solved, the algorithm managed the forest by invalidating sections that were affected by moving obstacles and trimming away unnecessary nodes.

A recent study addressed the issue of PRM-based planners spending most of their time checking for collisions when constructing the roadmap (Jaillet & Simeon, 2004). This study took advantage of the fact that moving objects only partially changed the workspace of the robot. It limited the area of the roadmap that was updated to portions that were important to obtaining a solution. If an important area of the roadmap was broken by a moving obstacle, the algorithm employed an RRT planner to attempt to locally reconnect the endpoints of the blocked edges. If this local reconnection solution failed, extra nodes were placed in the map near the broken edges and then connected to allow for more options. This algorithm also stored past positions of moving objects to reduce the number of collision checks required for a particular edge in the map.

A method utilizing a Flexible Anytime Dynamic A* PRM (FADPRM) algorithm was implemented in (Belghith, Kabanza, Hartman, & Nkambou, 2006), allowing the planner to use a roadmap to provide a sub-optimal path quickly. The robot’s workspace was divided into zones based on the different degrees of desirability of traversal. The algorithm then incrementally improved the quality of the solution path if extra time was available. Experiments were conducted by simulating the Space Station Remote Manipulator System (SSRMS). The results of these simulations showed that the FADPRM algorithm initially required more time to replan, but quickly and significantly reduced that amount of time when compared with a traditional PRM replanner.

Path Planning with Moving Obstacle Motion Prediction

The shortcomings of the discussed motion planning algorithms leave much to be desired. Treating dynamic objects as instantaneously static may simplify the motion planning procedure and allow for rudimentary obstacle avoidance; however, when dealing with large autonomous robots moving at fairly high velocity, as was the case during the DUC, rudimentary obstacle avoidance is not sufficient. Avoiding collisions was of the utmost importance and, therefore, requires a more elegant solution. The ability to predict the likely motion of any dynamic obstacles in the environment would allow for motion planners to more effectively find a solution path that would safely and smoothly avoid these obstacles.

Regression Methods

Autoregressive modeling (ARM) is one method of predicting future states of time series data. This form of prediction was implemented in (Kehtarnavaz & Li, 1988) to aid in generating a collision-free path for a simulated robot. This model was a function of a difference equation between successive position measurements and a prediction error estimation term. Accelerations for the obstacles were modeled in a similar fashion. The coefficients of this model were then solved for and used in the prediction function. Several sets of simulations results presented showed that this method was feasible for predicting obstacle positions one time-step in the future for obstacle avoidance purposes.

Another study fused a steering behavior generated from an ARM-based obstacle motion prediction with obstacle avoidance and goal-seeking behaviors to control mobile robot in an environment with dynamic objects (Yung & Ye, 1998). A least squares method was used to form the ARM from previous measurements of an obstacle's positions. The predicted positions coming from the ARM and the current position of the obstacles were then used to form a collision zone, which was used to define fuzzy representations of allowed and disallowed

steering angles. Fuzzy control actions were also generated in an obstacle avoidance behavior and goal-seeking behavior and then fused with the control actions from the predictor.

Elnagar and Gupta used the conditional maximum likelihood technique to estimate the parameters for an ARM for motion prediction of moving obstacles (Elnagar & Gupta, 1998). The study also extended the aforementioned ARMs in that it predicted both translational and rotational motion. The same simulations were carried out as in (Kehtarnavaz & Li, 1988), but the maximum likelihood method produced much more accurate predictions of the obstacle movement. Elnagar further extended this work by considering variable time-steps for predictions (Elnagar & Hussein, 2003). This method also used a quaternion representation rather than an Euler representation. The ARM utilized was almost the same as in other methods mentioned, but the prediction process was performed with variable time-steps. If a prediction was deemed accurate, the time interval was increased before the next reading. This adaptive time-interval feature allowed the algorithm to outperform other ARM methods in terms of computational cost.

Another interesting robotic application of an ARM for object motion prediction deals with robotic manipulators attempting to grasp moving targets. Houshangi began developing this concept using visual feedback of the manipulator and the target object (Houshangi, 1990). Because of the inherent delay in the estimating the current location of the target and the manipulator end-effector, the prediction model was used to control the manipulator arm. The trajectories generated by the planner were implemented by a self-tuning controller to allow the end-effector to track and grasp the target object.

Several other variations of motion planning algorithms which use ARMs to predict future obstacle positions have been developed. Zhuang used an ARM predictor in a polar-coordinate

space based planner (Zhuang, Du, & Wu, 2006), and Yu and Su used an Adapted Regression model based on a polynomial regression coupled with a new method which classified obstacles as triangular, convex, or convex (Yu & Su, 2003). Motion planning in a situation with multiple, cooperative robots was also considered by Pereira (Pereira, Campos, & Aguirre, 2000).

A recent study carried out in CIMAR used a polynomial regression to track and predict the motion of moving obstacles (Kent, 2007). The method presented by Kent first fit a high order polynomial to a time series of position data for an obstacle. The algorithm then ran a statistical analysis of variance to determine which coefficients of the polynomial were irrelevant and disregarded these terms from the model. It repeated the entire procedure until a lowest order polynomial predictor was settled upon. This lowest order polynomial was then used to predict obstacle position, velocity, and heading for time steps of one, three and five seconds in the future. The algorithm also provided an upper and lower confidence interval to account for uncertainty in the model.

Bayesian Methods

Because of their usefulness when dealing with modeling systems based on uncertain data, prediction algorithms utilizing Bayesian probability theory are well suited for solving the presented problem. One such method, known as the Bayesian Occupancy Filter (BOF), was developed for short-term motion prediction. This method draws heavily from the development of Bayesian programming described in (Coue, Fraichard, Bessiere, & Mazer, 2003), which broke the problem down into two complementary stages run recursively. The estimation stage sought to approximate the probability of occupancy for each cell in the described grid by recursively using the sensor observations. The prediction stage then developed an *a priori* description of the occupancy of a particular cell in the grid for the next time step.

The BOF was used as a potential solution to the problems of occlusion and consistent detection of moving obstacles in (Laugier et al., 2005). It was applied to the cells in a four-dimensional occupancy grid which included the velocity dimension. This study in particular applied a wavelet-based model for the BOF. A partial motion planning algorithm (PMP) was then used with the predictions to navigate through the dynamic environment by generating a set of "inevitable collision states," which were robot states which, regardless of the avoidance control input applied, would result in a collision with an obstacle. The PMP searched through the grid and returned the best plan, which may not have reached the final goal point, but which avoided these collision states.

A dynamic Bayesian network was used to model the motions of moving objects in (Rennekamp, Homeier, & Kroeger, 2006). The mapping of this Bayesian network matched that of a planar Voronoi graph. The transition models of the trajectories of the moving objects were learned during a training phase. These models were then projected onto the graph and used to predict the motions of the obstacles. This system was implemented using a vision system to track the motions of humans in an office-type environment.

Markov Methods

Several modeling methods have been attributed to Russian mathematician Andrey Markov. Hidden Markov models (HMM) describe stochastic processes that cannot be observed directly, but rather are observed through sequences of observations generated by another set of stochastic processes (Rabiner & Juang, 1986). Three problems must be solved for an HMM to be useful. The evaluation problem deals with proving a model generated a particular set of observations. The second problem addresses the issue of figuring out the state sequence, which is the hidden part of an HMM. And finally, the training problem attempts to optimize the parameters of the model.

A hidden Markov model was used to predict moving obstacle's motions by Zhu for a visually controlled robot in (Zhu, 1990). The study considered three types of obstacle models. The constant velocity model assumed zero acceleration and served as a basis for more complicated models. The random motion model used a probability distribution function to describe the changes of the obstacles state. Lastly, the intentional motion model considered that the obstacle may have had some predetermined objective or route, which could not simply be described by a probability distribution, but rather required some *a priori* knowledge of the obstacle's intent. The current position and future model-predicted position of the obstacle were used to create "forbidden regions" in the robot's grid. A trajectory-guided motion planning algorithm was used to evaluate a finite set of potential paths which took the HMM's probabilistic evaluations into account. This work was later compared to a deterministic prediction algorithm (DPA) in (Zhu, 1991). It was shown that the HMM method led to a lower collision rate with moving obstacles than did the DPA. However, it also had a higher computation time than the DPA and deviated more from the global path.

Markov chains are the simplest form of hidden Markov models and deal with how the outcome of one process affects the outcome of a subsequent instance of that process. More specifically, given a set of states, Markov chains seek to estimate transition probabilities to predict what the next set of states will be (Grinstead & Snell, 1997). Using Markov chains allows the next state to be predicted based solely on the current state, and not based on previous states.

The dynamics of moving humans were abstracted to Markov chains in (Rohrmuller, Althoff, Wollherr, & Buss, 2008) and used to generate time-dependent occupancy grids for robot navigation. A velocity model and an acceleration model were studied. A set of probabilistic

reachable states, defined as the set of all states that a human can reach from an initial state for all possible control inputs, was generated and was projected in a potential field for the motion planning algorithm. Experimental results showed this method was effective in navigating through areas populated by humans. The results also showed that use of the velocity model allowed for quicker replanning than did use of the acceleration model.

Neural Networks

Development of artificial neural networks (ANN) is growing rapidly due to their many applications. Chang and Song devised a model-free ANN for single time-step prediction of moving obstacles for mobile robot navigation (Chang & Song, 1996). For this early study, obstacles were assumed to follow a rectilinear path with constant velocity. Past sensor readings were the inputs to the ANN and the outputs were the predicted reading at the next time instant for that particular sensor. It was found that the two most recent sensor readings were enough to predict the next one. The ANN was trained off-line by means of back-propagation. This predictor was coupled with a virtual force guidance navigation scheme. The repulsive forces generated by the moving obstacle were generated by a deceleration and push-away layer surrounding the obstacle position. The algorithm was later tested on a mobile robot with human obstacles (Chang & Song, 1997). Because of several shortcomings of the ultrasonic transducers used in (Chang & Song, 1996) and (Chang & Song, 1997), Song and Chang also tested the feasibility of a CCD camera added to the sensor network (Song & Chang, 1999).

An ANN was used to solve the navigation problem in a study which utilized a hierarchical partially observable Markov decision process (POMDP) in (Foka & Trahanias, 2002). The POMDP, which in effect modeled the decision process of the robot, took a state representation of the environment as input and output the optimal control actions. For large search spaces, hierarchical POMDP's are computationally more efficient since they decompose the problem

into multiple linked POMDP's, each with a smaller state space to search. A polynomial ANN was used for short-term prediction and was integrated into the reward function of the POMDP. The coefficients of the polynomial were calculated by training the ANN off-line with an evolutionary method.

A moving obstacle avoidance algorithm was developed by coupling a radial-basis-function neural network (RBFNN) with a rolling planning method in (Li, Li, & Song, 2008). A camera attached to the robot was used to capture the motions of the dynamic obstacles for training the ANN. A rolling window was generated to shrink the search space to the sensor's viewing area. If it was determined in the rolling planning process that the predicted position of a moving obstacle would obstruct the robot, the obstacle was treated as static at that predicted position and an obstacle avoidance maneuver was generated.

Dynamic Environment Representation

Typically, a robot's environment is represented in two spatial dimensions. A common approach to creating this representation is to generate a discretized grid. Each cell in this grid is assigned a value which describes the occupancy of that particular cell. This concept was first introduced for mobile robots by Elfes (Elfes, 1989). The most basic occupancy grids are binary in nature and can only label cells in the grid as "occupied" or "free." Sometimes a more variable representation of the occupancy is desired, such as a probability of occupancy. One method similar to a probabilistic representation was created by CIMAR with the development of the concept of "traversability" and the traversability grid (Crane et al., 2005). This type of grid still can describe cells as "occupied" or "free," but also allows for varying degrees of "traversability" to distinguish areas of the grid that may be more desirable to traverse than others.

These types of grids, coupled with advanced motion planning algorithms have been utilized by many robotics systems in recent history. However, they allow for a strictly static

representation of the environment at any instant in time. Therefore, a moving obstacle is actually represented as a static obstacle in a single occupancy grid. For dynamic environments, it would be advantageous to be able to have a spatial and temporal representation of the surroundings. A few studies have attempted to address this issue, but much room is left for improvement.

State-Time Graph

The concept of adding the time dimension to a robot's state space was first considered by Fujimura and Samet in (Fujimura & Samet, 1989). Moving objects were represented as swept volumes in state-time space. A hierarchical quadtree structure was used to represent this three dimensional space. The state-time space was repeatedly divided until each cell satisfies one of a set of conditions. Vertex cells contained part of a trajectory of a vertex of an obstacle, edge cells contained part of a trajectory of an edge of an obstacle, empty cells did not contain any part of any trajectory, and full cells were completely contained in a trajectory. Cells were then decomposed into L-points associated with the two spatial dimensions and T-points associated with the time dimension. The set of L and T-points were then searched by a heuristic algorithm to find the time-optimal trajectory for the robot.

State-time space was later used for the motion planning problem of a mobile robot following specified lanes in (Fraichard & Laugier, 1992). Points in this state-time space were represented as tuples consisting of the current lane, position, velocity, and time instant. Neighboring points were reached by applying acceleration to the robot for a time-step. This process generated a directed graph in the state-time space for the robot to search through. An A* algorithm was used to facilitate this search to find the time-optimal trajectory for the robot to follow.

The authors further extended this study to take constraints imposed by the dynamics of the robot and moving obstacles into account (Fraichard & Laugier, 1993). A dynamic model of the

robot was presented and constraints associated with engine force, sliding, velocity, and acceleration were considered. Dynamic obstacles were again represented as swept volumes in the robot's state-time space. These volumes, along with the regions represented in the dynamic constraints models, were used to create "forbidden regions" of the state-time space of the robot. The same A* algorithm was then used to search the state-time space of the robot and successfully avoided these "forbidden regions" in simulation.

Rude investigated the issue of collision avoidance for two cooperative robots in state-time space in (Rude, 1997) by means of a space-time "collision vector." These two robots were constantly transmitting their planned trajectory through the joint state-time space to each other, so each knew the intended path of the other. The "collision vector" was then calculated as the shortest magnitude vector between the two planned trajectories. If the magnitude of this vector was less than a distance equivalent to the radii of the two robots, a collision was possible and needed to be avoided. This avoidance maneuver was generated by constructing a displacement vector that would alter the trajectory of one of the robots enough such that the magnitude of the resulting "collision vector" was large enough to evade the collision. Simulations and experiments on robots confirmed that this method was successful in avoiding collisions.

The interface propagation method, a modified form of the path-velocity decomposition, was used for velocity planning through a state-time graph in (Hwang & Ju, 1999). In this case, the state-time graph was two-dimensional, consisting of the time dimension and a single distance dimension to represent the distance traveled along an existing global plan. Forbidden regions were again used to represent the motion of obstacles in the state-time graph. A maximum distance and time interval were calculated for these regions to approximately quantify a potential collision. A "speed zone cone" was created to modify the state-time graph based on the

maximum and minimum speed of the robot, as the two-dimensional graph could equivalently represent a speed graph. The velocity profile was then calculated based on the interface propagation algorithm. This profile facilitated avoiding the "forbidden region" representations of the moving obstacles in the environment.

Temporal Geographical Information Systems

Geographic information systems (GIS) keep track of objects and events and where these objects and events occur or exist (Longley, Goodchild, Maguire, & Rhind, 2001). In addition to these spatial descriptions, it may be desirable to capture and store temporal information about these objects and events, which gives rise to the concept of temporal GIS. This new form of GIS seeks to record historical geographic states, pick out changes and patterns in spatio-temporal data, and predict the future properties of these geographic states.

This multi-dimensional framework gives rise to the potential for one dimension dominating another, which results in the clustering of data being very attractive as discussed in (Langran, 1990). One example of a structure that may experience this dimensional dominance is the temporal grid, which is a common geographic data structure, and can be clustered in a variety of ways. A space-dominated approach provides individual snapshots of the grid over time as described in (Armstrong, 1988) and is fittingly known as the Snapshot Model. Conversely, a time-dominated approach could group values contained within an individual cell over all time-steps, while a spatiotemporal approach, where both the time and space dimensions are weighted approximately equal, could cluster small cubes in space-time. In addition to the Snapshot Model, Langran and Chrisman developed the Space-Time Composite method (Langran & Chrisman, 1988) which builds a view of a geographic area by beginning with a base map, which becomes a temporal composite by considering the accumulated geometric changes in the area. This essentially decomposes the map into smaller pieces as time passes.

Temporal GIS has previously been applied to several different robotic applications. Hatayama and Matsuno describe the use of temporal GIS by robots examining damaged buildings after disasters in (Hatayama & Matsuno, 2008). The KIWI+ format developed in this study allows spatiotemporal objects to be managed by using both the aforementioned Snapshot Model and Space-Time Approach model described in (Worboys, 1994). Trajectories of spatiotemporal objects can be described either as a polyline or polygon with N points and only one temporal factor or as a polyline or polygon where each point has its own instant temporal factor. Using the same format, temporal GIS is used to aid in recognizing vehicles on a road which may be parking in (Ishikawa et al., 2005). "Change regions" are built from the difference between reference GIS data and a proposed omni-directional motion stereo vision system. This method was successfully used to recognize a truck, sedan, and station wagon in the field.

Temporal Occupancy Grid

A recent extension to the traditional occupancy grid added the time dimension and has led to the concept of the Temporal Occupancy Grid (TOG) (Arbuckle, Howard, & Mataric, 2002). This new type of grid allowed for occupancy to be estimated for a number of different time-scales. The grid consisted of a matrix made up of two spatial dimensions, one time dimension, and additional dimensions for the number of time-scales used. Rather than each cell in the grid having one occupancy value, each cell had several different occupancy values corresponding to the different time scales specified. Each cell was further classified based on these numerous occupancy values. Cells containing static obstacles would likely have high probabilities of being occupied on all time scales. On the other hand, cells which had moving obstacles passing through them would have high probabilities of occupancy on short time scales, but low probabilities on long time scales. The time dimension of the TOG could then be collapsed by considering these classifications. Experiments were conducted to validate the

method for static and dynamic environments, and showed that a TOG could effectively estimate historic occupancy of an environment.

While they did not refer to it explicitly as a temporal occupancy grid, Biber and Duckett used a grid which stored occupancy values for multiple time scales in an attempt to map dynamic environments for mobile service robots (Biber & Duckett, 2005) . Each level of the grid associated with a specific time scale was built from a set of sensor samples taken from the previous scale to that particular scale. The levels of the map were updated by randomly removing a number of samples and replacing them with an equivalent number of randomly selected samples from the new sample set. To use the map at a specific time, a normal distribution was estimated from the samples. Experiments were carried out using this method with five time scales in a busy office setting to obtain a short-term and long-term memory map. The results of these experiments showed that this method could effectively describe areas of the environment that are likely static or dynamic.

Another TOG algorithm was developed for mobile robot environment mapping by Mitsou and Tzafestas in (Mitsou & Tzafestas, 2007). In this method, the traditional occupancy grid was again extended through the time dimension. The time dimension was represented by time intervals, and a time index, in the form of a B+ tree, was assigned to each cell in the grid. The probability of occupancy of the cell for a particular time step was stored in these indices. The standard deviation of the various occupancy probabilities for each cell was then used to describe the dynamics of that cell. Cells containing static objects exhibited little to no dynamic activity and had high occupancy probabilities. Dynamic objects were classified as either low dynamic objects or high dynamic objects. Low dynamic objects appeared only in a limited number of places, while high dynamic objects moved arbitrarily and could be found in many places. The

time indices of each cell were searched to aid in detecting the different types of objects. For experimentation, a simulated robot was outfitted with a simulated laser range finder and tasked with mapping an environment with doors that can be opened or closed and humans moving around.

Autonomous Target Interception

The concept of using motion prediction to aid in allowing a robot to navigate to a particular state to intercept some sort of object or target has been explored for a number of applications. Hujic, Croft, Fenton, Mills, and Benhabib developed a strategy entitled Adaptive Prediction Planning and Execution (APPE) in (Hujic et al., 1995) to guide a robotic manipulator arm to a rendezvous point to grasp an object. Once a trajectory was predicted for the target, a one-dimensional search was conducted of the predicted trajectory to find rendezvous-points where the target and manipulator end-effector arrived at the same time and with the same velocity. The minimum time rendezvous point was selected and the joint trajectories were then determined for the manipulator. This strategy was then extended to on-line replanning in another study (Croft, Fenton, & Benhabib, 1998).

A rendezvous-guidance technique was coupled with the velocity obstacle concept in (Kunwar & Benhabib, 2006) to autonomously navigate a mobile robot to intercept a moving target among other moving obstacles. A vision sensing system was used to collect position and velocity data on both the target and all other dynamic objects in the environment. Velocity obstacles were then generated for all objects within a defined time horizon for obstacle avoidance. A parallel-navigation law was then used to guide the robot to the target's location and to match the velocity of the target. This law stated that if the relative velocity between the robot and the target remained parallel to the position vector between the two, the distance

between them would decrease until they "collide." Simulation and robotic experiments were used to test the method with multiple moving obstacles and a single target.

An interception problem for a robotic manipulator on an orbiting spacecraft attempting to grasp free-floating objects was discussed by Robert in (Robert & Sharf, 2007). A Kalman filter was used to propagate the target dynamics to predict the future states. The interception technique was broken down into the approach to the rendezvous point and the capture of the target. The approach trajectory was generated by finding the time-optimal interception point. This optimal trajectory was recalculated as needed when the predicted trajectory of the target changed, thus changing the optimal interception point. For the capture phase, a finer tracking algorithm was used to adjust the manipulator end-effector velocity and twist to match that of the target. Numerical simulations were executed to compare this technique to a more traditional visual-servoing method and showed that the predictive technique resulted in almost negligible joint angle error and a lower interception time.

This chapter provides an overview of many of the common techniques previously studied to address the key elements of the problem statement for the research presented. Specifically, it has focused on motion planning methods for dynamic environments, motion prediction for moving objects in dynamic environments, and dynamic environment representation. The following chapter describes the newly developed predictive temporal motion planning method from a theoretical standpoint.

CHAPTER 3
THEORETICAL APPROACH

Navigating dynamic environments is a complex problem for a robot that requires a coordinated effort among several elements of the system. This dissertation presents a novel approach to this task, which considers how the robot's surroundings changes with time. A predictive temporal motion planning method was developed that utilized the coupling of moving obstacle detection and prediction with the occupancy grid concept to generate a representation of the changing environment, which the motion planning algorithm then used to more intelligently generate motions which avoided any objects in its vicinity. This chapter strives to break this process down into its key components. The first of those components considers the generation of the temporal grid, followed by the inclusion of obstacle prediction information into the temporal grid, and finally, the exploration of the temporal grid by a motion planning algorithm to build optimal trajectories for the robot.

First, an overview of common control techniques and graph search methods is provided to divulge the important parameters that intimately link the motion planning algorithm and the temporal grid concept. The next section then focuses on the construction of the temporal grid itself, followed by a description of the process of predicting the motions of obstacles and the integration of these predictions into the temporal grid. Lastly, the temporal motion planning algorithm is outlined to draw all of the elements presented in the previous sections together. The information in these sections is provided in a generic sense to emphasize that this predictive temporal method is applicable to a variety of robotic systems, including robotic manipulators in addition to unmanned air, ground, and underwater vehicles. The next chapter narrows down the description to an implementation on an unmanned ground vehicle.

Motion Planning Overview

Before outlining the fundamental theory of the temporal grid, it is prudent to discuss the influence of the motion planning algorithm on the creation of the grid. Therefore, a brief overview of discrete-time control methods for robot navigation is provided. The ultimate goal of robot planning is to generate a continuous motion from a start configuration to a goal configuration contained within the robot's configuration space while avoiding all obstacles. Before this process can even begin, sensors must examine the surrounding environment to produce a mapping displaying the areas that are both safe and unsafe for the robot to navigate to reach its goal.

A common approach to this mapping is the generation of a grid map, which decomposes the robot's environment into a series of spatially-defined cells. Each cell in the grid represents a region of the surrounding environment that can be classified as being free and accessible to the robot, or closed due to being currently occupied by an object. These classifications can then be used by a motion planning algorithm to evaluate the grid to determine a sequence of configurations that will successfully move the robot from its initial position and orientation to its goal position and orientation.

The first technique of interest considers the various grid-based or graph-based search methods. These methods overlay the grid map on the systems configuration space, therefore, considering each grid point as a potential configuration, or state. These methods include uninformed techniques, such as breadth-first and depth-first search. Uninformed search methods operate with no special knowledge of the problem other than the start state and the goal state and, therefore, have no information to focus the direction of the search. Instead, they use a brute-force approach to search as many nodes on the graph as possible in hopes of finding a solution trajectory (Nilsson, 1998).

Breadth-first search considers applying all possible operators to a state, followed by applying all possible operators to the successors of that state, and so on. Conversely, depth-first search expands and explores the successors of a single state at a time. Once a successor is generated, one of its own successors is generated, and so on. A depth bound is defined such that a particular branch of the search does not continually expand states without allowing for different branches to be explored. Figures 3-1 and 3-2 display simple representations of breadth-first search and depth-first search state exploration, respectively.

Informed search methods include best-first techniques such as greedy search and A* search. Best-first search algorithms explore a graph by always expanding the most promising state. This is realized by using a rule, or heuristic, to rank states and determine which is best for the next expansion. Heuristics can range from simple functions, such as the Euclidean distance between a state and the goal to more complex evaluations based on a number of different parameters and they serve to estimate the distance, or cost, from a particular state to the goal. Admissible heuristics are required to not overestimate cost to the goal such that:

$$\hat{h}(n^*) \leq h(n^*) \tag{3-1}$$

where $\hat{h}(\cdot)$ is the estimate of the cost from a state n^* on the optimal solution path and $h(\cdot)$ is the actual cost from that same state to the goal. Each time a state is explored and evaluated, it is placed on a priority queue, where the first state on the queue is the most promising.

Receding horizon control (RHC) is another commonly used motion-control strategy that utilizes a discrete set of configuration or state changes that are subject to the dynamics of the system, given simply as:

$$X(t+1) = f(x(t), u(t)) \tag{3-2}$$

where $X(t+1)$ is the set of potential future states that are obtainable from the current state, $f(\cdot)$ is the dynamics function, $x(t)$ is the current state, and $u(t)$ is the current control input. This sequence of state changes is determined by an associated sequence of open-loop plant input commands that are generated out to a finite time horizon T. This sequence is optimized and given as:

$$\pi_t = \{\hat{u}_t, \hat{u}_{t+1}, \hat{u}_{t+2}, \cdots, \hat{u}_{t+T-1}\} \tag{3-3}$$

where $\hat{u}_{t+i}$ are the discrete optimal inputs and T is the finite time horizon. The initial input in the optimal sequence is selected as the commanded control input sent to the plant such that:

$$u(t) = \hat{u}_t. \tag{3-4}$$

At this point, the control is implemented and the optimization repeats to determine the next control input for the state change from the new current state $x(t)$ to a specific future state $x(t+1)$ considering updated vehicle and environmental state information. As evidenced in Equations 3-2 and 3-3 the discrete control inputs and state changes occur at distinct distance-steps or time-steps, as was the case with the grid-search method. Each successive state-change results in a potential configuration that could describe the system at a distinct time in the future. At each of these future-times, it is possible that the composition of the surrounding environment could have changed dramatically, including the location, orientation, and velocity of any obstacles present. It is this capricious nature of a robot's environment that was one of the main factors in the development of the presented temporal motion planning method. The work presented in this dissertation showed that the ability to incorporate information about how the environment changes into a motion planning algorithm that utilizes a strategy such as RHC can improve stability and performance of the system.

The description of these methods, which use discrete state changes, gives rise to essential parameters required for the development of a temporal grid structure. For the graph-search techniques, the exploration of adjacent states represented discrete steps through the configuration space while attempting to reach the goal state. Each of these steps has some physical meaning depending on the state representation, such as a distance or a time-step for example. Likewise, the RHC approach utilized a sequence of optimized control inputs separated by similar time-steps as evident in Equation 3-3. The number of inputs in the optimal sequence was determined by the step-size associated with each state change and the finite time horizon. The time-step among adjacent states is hereafter denoted as Δt and the time horizon is denoted as T_H. These two key parameters played an important role in determining the number of temporal layers to be included in the temporal grid structure. They were also used in determining the appropriate future-times to estimate the positions and velocities of any obstacle present in the robot's environment. The details of these applications are provided in ensuing sections of this chapter.

Temporal Grid

A particularly novel aspect of the presented research was the concept of a predictive temporal grid map that represents a robot's environment at the current time and at distinct times in the future up to a time horizon discussed in the previous section. This section focuses on the temporal aspect of the grid map, while the predictive aspect is discussed in the next section. The novel temporal grid map was an extension of the traditional occupancy grid that is commonly used for robotic environment mapping. Occupancy grid maps, as defined by Thrun in (Thrun, Burgard, & Fox, 2005) seek to rasterize an environment such that:

$$m = \{m_i\} \tag{3-5}$$

where m is the map itself and m_i represents the individual cells that make up the map. The index i is associated with a particular row and column within the grid structure. This value is determined for a distinct row-column pair(r, c) as:

$$i = r \cdot c_{total} + c \tag{3-6}$$

where c_{total} is the total number of columns in the grid map. This convention can be applied regardless of the origin of the grid, as long as it is located in one of the corners. Each cell of m must be assigned an occupancy value based on accumulated sensor data for the region of the environment represented by that cell and the motion of the robot. This value can be estimated for an individual cell as the probably function:

$$p(m_i|z_{1:t}, x_{1:t}) \tag{3-7}$$

where $z_{1:t}$ is the set of all sensor measurements up to time t and $x_{1:t}$ is the sequence of positions of the robot up to time t. This calculation attempts to estimate the likelihood that a cell is occupied by some object at a given instant in time given the accompanying sensor data for that cell. Similarly, this posterior distribution can be built up for an entire map as the product of the individual cells as:

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}). \tag{3-8}$$

The development of the temporal grid required a similar procedure for the assignment of occupancy for all of its cells as well. The structure of the temporal grid differed as subsequent temporal layers were added. This spatio-temporal grid structure can be viewed in Figure 3-3 as essentially a three-dimensional array of grid cells with two spatial dimensions and the single time dimension. The inclusion of the time dimension in the grid allowed it to contain information depicting how the environment changed as time progresses. A specific grid cell may, therefore, be thought of as possessing multiple occupancy values, each associated with a time that is farther

in the future than the last. Figure 3-4 shows another visualization of these temporal layers of a group of grid cells. This image shows a representation of a single row of grid cells with several attached temporal branches. Each branch coming from the original cell represents that same cell spatially, but at a distinct time in the future. The identifier for each cell takes the form $(row, column, time)$.

By adding this temporal dimension to the grid structure, the indexing procedure for accessing a particular cell changed slightly. Whereas the index for a cell in the spatial grid relied only on the row and column value of the cell as in Equation 3-5, this index term must now account for the temporal layer in which the cell of interest m_i resides. The new indexing scheme for a row-column-time triple (r, c, t) can be shown as:

$$i = t \cdot r_{total} \cdot c_{total} + r \cdot c_{total} + c \tag{3-9}$$

where t is the temporal layer in which the cell resides, r_{total} and c_{total} are the total numbers of rows and columns in a layer of the grid, and r and c are the row and column value of cell m_i.

Referring to the previous section on motion planning algorithms, the values of discrete control time-step Δt and finite time horizon T_H were described as crucial components of the construction of the temporal grid structure. The time-step determined the instants in time represented in each of the successive temporal layers of the grid and, along with the time horizon, determined the total number of temporal layers in the grid to be generated. The number of layers N_l was calculated simply as:

$$N_l = 1 + \frac{T_H}{\Delta t}. \tag{3-10}$$

This calculation assumed that the time-step and time-horizon were both constant values, which is shown to not necessarily be true in the next chapter. An optimization scheme is also presented in the next chapter which succeeded in reducing the resources necessary to build the temporal grid

structure. This method also altered the indexing scheme presented in Equation 3-9. Now that the basic theory behind the temporal grid structure has been presented, the inclusion of obstacle motion prediction information in the grid is discussed.

Obstacle Motion Prediction

The introduction of moving obstacles into the robot's environment presents additional challenges to the motion planning problem. The use of an existing occupancy grid described in the previous section requires the motion planner to generate its sequence of control inputs out to the time horizon at the edge of the grid while assuming all objects are static during that time. Figure 3-5 shows a form of occupancy grid that was modified to show the output of a motion planning algorithm of an unmanned ground vehicle that was attempting to generate its sequence of controls. Detected obstacles are painted yellow and remained in the same position in the grid, even though they may have been moving. This provides a good example of how a motion planning algorithm expands out its potential future states all the way to the time horizon at the edge of the grid, which may represent ten to fifteen seconds or more in the future, while all the detected obstacles remain in the same position, regardless of whether they are stationary or moving.

Using a planning strategy such as RHC described previously, if one of the obstacles displayed were actually moving along a trajectory that would cross the robot's desired path in the future, the robot would not be able to react until the object actually obstructed its path and affected the control input evaluation. This could ultimately be fatal if the robot did not have enough time to stop or plan a path that avoids the obstacle. On the other hand, if the obstacle were crossing the robot's desired path, but would be well out of the way by the time the robot reached that position, the planning algorithm would have no reason to attempt to react to navigate around the obstacle. However, using the described methods, the motion planner could

incorrectly determine that it was necessary to alter its trajectory to avoid the obstacle. For these reasons, it was helpful to be able to estimate the position of the obstacle in the future and to be able to represent these predicted future positions in the grid. To accomplish this, a prediction algorithm was required to provide this information for both the position and velocity of all obstacles detected in the robot's vicinity. A general prediction method is first discussed to estimate the future positions and velocities, and is followed by a description of how these estimates were represented in the grid.

To facilitate this estimation, any of a number of prediction algorithms such as a polynomial regression predictor, similar to the one discussed in (Kent, 2007), can be used. This algorithm sought to fit curves to time-series of position and velocity data for each obstacle provided by a laser-based moving obstacle sensor. It accomplished this by tracking the global position of an obstacle and maintaining a recorded data set of size $n\ x\ 2$ as:

$$p = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}. \tag{3-11}$$

For a given data set such as in Equation 3-11, a polynomial was fit to the data that possesses the form:

$$y = \beta_{y_0} + \beta_{y_1} \cdot t + \beta_{y_2} \cdot t^2 + \cdots + \beta_{y_n} \cdot t^k \tag{3-12}$$

where k is the maximum order of the polynomial. Therefore, the creation of this prediction model depended on solving for the coefficients of the polynomial. The polynomial can also be represented in matrix form as:

$$\hat{y} = T\hat{\beta}_y \tag{3-13}$$

where: $\hat{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ $\hat{\beta}_y = \begin{bmatrix} \beta_{y_0} \\ \beta_{y_1} \\ \vdots \\ \beta_{y_k} \end{bmatrix}$ $T = \begin{bmatrix} 1 & t_1 & \dots & t_1^k \\ 1 & t_2 & \dots & t_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \dots & t_n^k \end{bmatrix}$.

In this matrix form, the coefficient vector $\hat{\beta}$ was solved for by pre-multiplying Equation 3-13 by T^T and solving the modified equation as:

$$\hat{\beta}_y = (T^T T)^{-1} T^T \hat{y} \tag{3-14}$$

Similarly, a polynomial of the form:

$$x = \beta_{x_0} + \beta_{x_1} \cdot t + \beta_{x_2} \cdot t^2 + \cdots + \beta_{x_n} \cdot t^k \tag{3-15}$$

can be fit and evaluated for the estimation of future x positions by solving for an equivalent coefficients vector $\hat{\beta}_x$.

Upon solving for the coefficients of the polynomials as in Equation 3-14, future estimates for x and y can be calculated for various values of t. Referring to the previous section on the creation of the temporal grid structure, each temporal layer was separated from adjacent layers by a distinct time-step determined by the motion planning algorithm time-step parameter Δt. These distinct time-steps between temporal layers were used to determine the time values for which the future positions of all obstacles needed to be predicted. A new predicted position was required for each successive group of potential state changes investigated by the motion planning algorithm, where the time-step associated with the group of state changes t_i was calculated as:

$$t_i = \{i \cdot \Delta t, i = 0,1 \dots n : \ t_i \leq T_H\}. \tag{3-16}$$

These values of t_i were then used in Equations 3-12 and 3-15 to solve for the estimated future values of x_i and y_i.

After estimating the future positions of any detected obstacles at the distinct times presented in Equation 3-16, it was necessary to convert these positions into the temporal grid

frame of reference, namely row and column values, to be represented in their respective temporal layer. The row and column values for a particular temporal layer were calculated from the predicted *x* and *y* positions at that same future-time value t_i as follows:

$$row_i = \frac{y_{t_i} + (GSM/2.0)}{R_g} \tag{3-17}$$

$$column_i = \frac{x_{t_i} + (GSM/2.0)}{R_g} \tag{3-18}$$

where x_{t_i} and y_{t_i} are the predicted *x* and *y* positions of an obstacle at time t_i, *GSM* is the grid size (length in units equal to that of x_{t_i} and y_{t_i}), and R_g is the resolution of each grid cell (in the same units). At this point, the occupancy of these row-column pairs was set as occupied in the appropriate temporal layer to represent where the obstacle would likely be at that future time.

Temporal Motion Planning

With the inclusion of obstacle motion prediction information in the temporal grid structure, the motion planning algorithm used this predictive temporal grid to consider how its environment changes as time progresses and to more intelligently generate its sequence of control commands according to these changes. As outlined in the previous sections covering the creation of the predictive temporal grid, each temporal layer corresponded to a representation of the environment at a distinct time-step in the future. These time-steps coincided with the future times at which the planning algorithm was attempting to determine the appropriate control to institute the desired change in state.

The robot motion planning problem can generally be viewed in its optimal control problem formulation, which can be given as:

$$J\big(x(t)\big) = \Phi\big(x(t_0), t_o, x\big(t_f\big), t_f\big) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt \tag{3-19}$$

$$\begin{aligned} subject\ to: \quad & \dot{x} = f(x(t), u(t), t) \\ & \phi\big(x(t_0), t_0, x(t_0), t_f\big) = 0 \\ & c(x(t), u(t), t) \leq 0 \end{aligned}$$

where $J(x(t))$ is the performance measure used to evaluate the possible trajectories the robot can follow, $\Phi(\cdot)$ is the endpoint cost associated with the final state of the robot, $L(\cdot)$ is the intermediate cost functional associated with each of the intermediate states along a trajectory, $x(\tau)$ is a state of the robot at time τ, $u(\tau)$ is the control instituted at time τ, and t_0 and t_f are the initial and final time of the motion, respectively. The constraints placed upon the system include the dynamics function $\dot{x} = f(\cdot)$, the endpoint constraints defining where the system must begin and end its motion $\phi(\cdot)$, and the path constraints of the environment defining physical constraints that may limit the possible trajectories $c(\cdot)$. The second element of Equation 3-19, representing the sum of the intermediate state costs, can be split up by discretizing the time interval between t_0 and t_f such as

$$\boldsymbol{t} = [t_0 \quad t_1 \quad \cdots \quad t_f]. \tag{3-20}$$

where the important motion planning parameters discussed previously can be related as

$$\Delta t = t_i - t_{i-1}\ i = 1 \ldots f \tag{3-21}$$

$$T_H = t_f.$$

By utilizing this method, this discrete set of evaluation times coincided with the distinct time-steps associated with the temporal layers of the temporal grid described previously. Therefore, the temporal layer associated with each of these intermediate times was used in the calculation of each of the intermediate costs defined by the evaluation functional $L(\cdot)$.

The intermediate cost function can take many forms, but ultimately will consider the conditions of the physical region of the environment which coincides with the potential state to be evaluated. Using an occupancy grid map as an example representation of the environment, this would be equivalent to using the occupancy value of a cell in which a potential state resides to evaluate the appeal of that state. To expand on this idea, consider the motion planning

algorithm evaluating its first set of potential state changes, occurring at time-step t_1. Jumping ahead a few steps in the evaluation process, perhaps a grid cell that will contain a future potential state at time t_j is unoccupied and available for the robot to traverse. Now consider the situation that, in the time between evaluating the first set of potential states and evaluating the future state in the cell at time t_j, the cell becomes occupied by an object. It is then likely that the motion planning algorithm will incorrectly evaluate that state by believing that cell to still be unoccupied. This error could lead to this state being selected to determine the control input sequence defined in Equation 3-3 and could ultimately lead to the robot colliding with the obstacle.

By replacing the occupancy grid in the previous example with the new predictive temporal grid map described in this chapter, the intermediate cost evaluation function did not simply use the spatially-defined region of the environment to evaluate a particular state, but also considered the estimated future conditions of that region on a time-dependent basis. Returning to the previous example, if the prediction algorithm correctly estimated the future states of the obstacle, the cells in the grid that contain this estimated future obstacle state were correctly modified from being considered free to being considered occupied on the time-scale associated with that step in the state-evaluation process. Therefore, as the motion planning algorithm evaluated that time-dependent state, it correctly assessed that the state was not feasible as it could lead to a collision with the object.

Figure 3-6 displays a series of successive steps in the proposed temporal motion planning process showing the coarse exploration of potential states as the predicted positions of a moving object pass in front of the vehicle. Each black circle represents a potential future vehicle state, with the connecting arcs and straight lines representing the trajectory taken between these

various potential states. The goal state is represented by a green circle and the associated goal region is shown as a dashed circle around the goal state. Figure 3-6(a) represents the initial exploration step which uses the information in the initial temporal layer of the grid for evaluation. Successive generations step further out in time as the motion planner accesses their associated layers of the temporal grid, which also step further out in time out until it reaches the goal as in Figure 3-6(f). These images depict a case where the object passes before the robot would reach the crossing point of the objects path with its own desired path. Figure 3-7 displays a similar series of images representing a case where the motion planner is required to divert from its nominal path to avoid the object at its predicted position in shown in Figure 3-7(b).

This chapter has provided a general overview of the predictive temporal motion planning method. It has presented the concept in a fashion that could lead to the method being implemented on a variety of robotic applications. The description of this new approach has been broken into its main components, those being the temporal grid, the inclusion of obstacle motion prediction, and the temporal motion planner. Each of these components was intimately related by the temporal elements of the discrete nature of common motion planning algorithms. The next chapter outlines the specific implementation of this new motion planning method that was developed and analyzed for the presented research.

This chapter outlines the new predictive temporal motion planning method in terms of the theory of the key parameters used in common discrete motion planning techniques, the temporal grid concept, motion prediction models, and temporal motion planning. These concepts addressed the main elements of the problem statement provided in Chapter 1. The following chapter details the specific implementation of these concepts for a large unmanned ground

vehicle which was the basis for this study. It focuses on the important software components that previously existed and those that were created for this research.

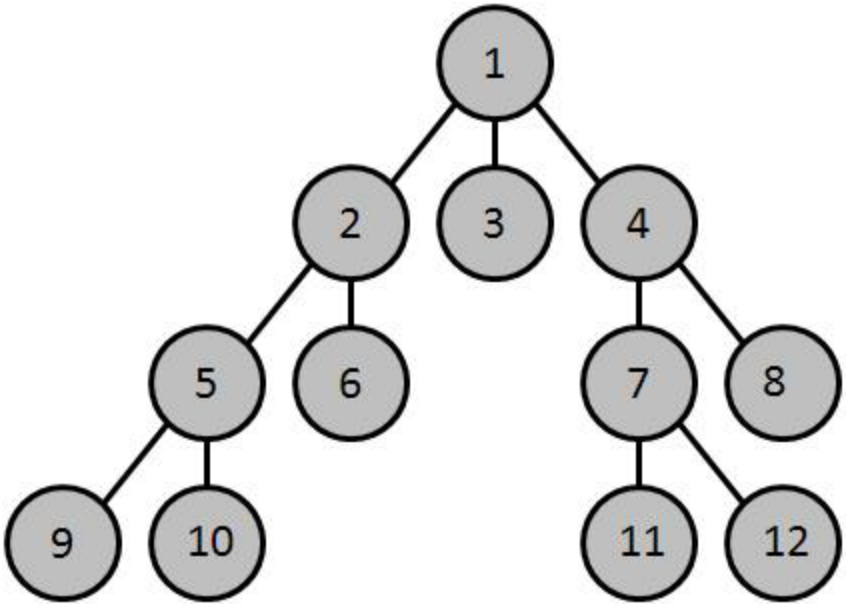

Figure 3-1. Breadth-first search state exploration order.

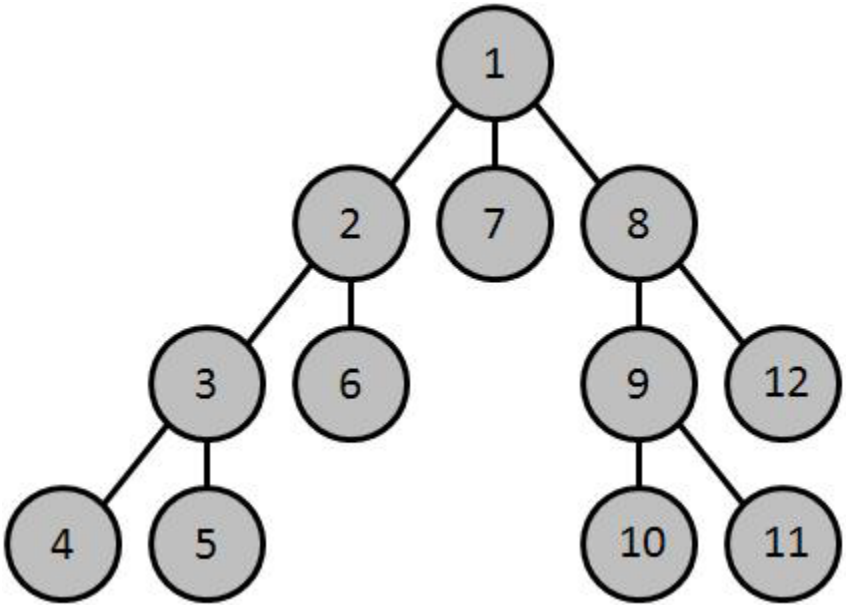

Figure 3-2. Depth-first search state exploration order.

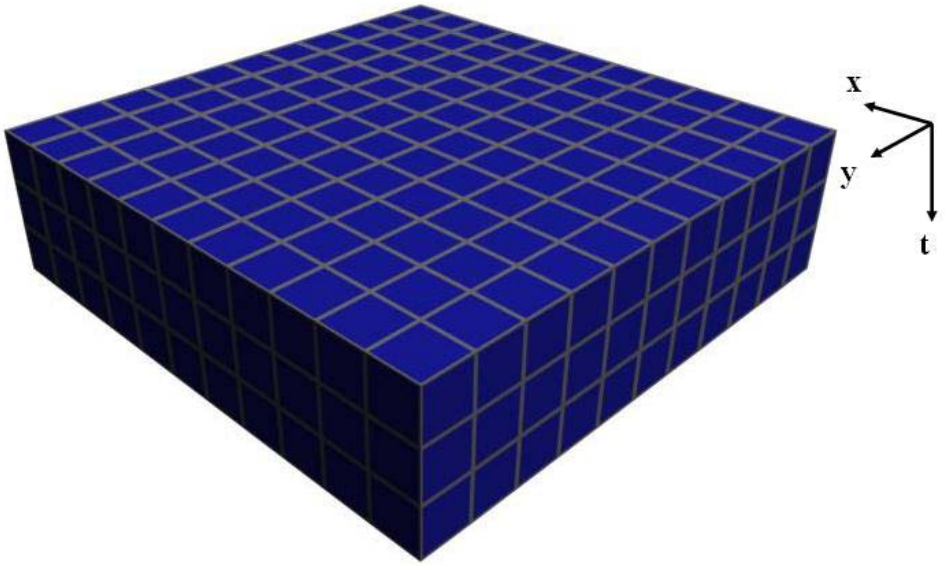

Figure 3-3. Three-dimensional temporal grid structure.

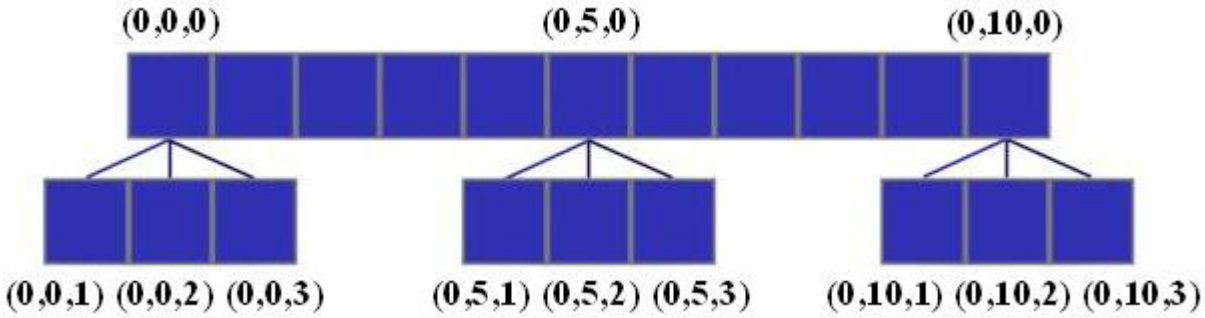

Figure 3-4. Temporal grid tree visualization.

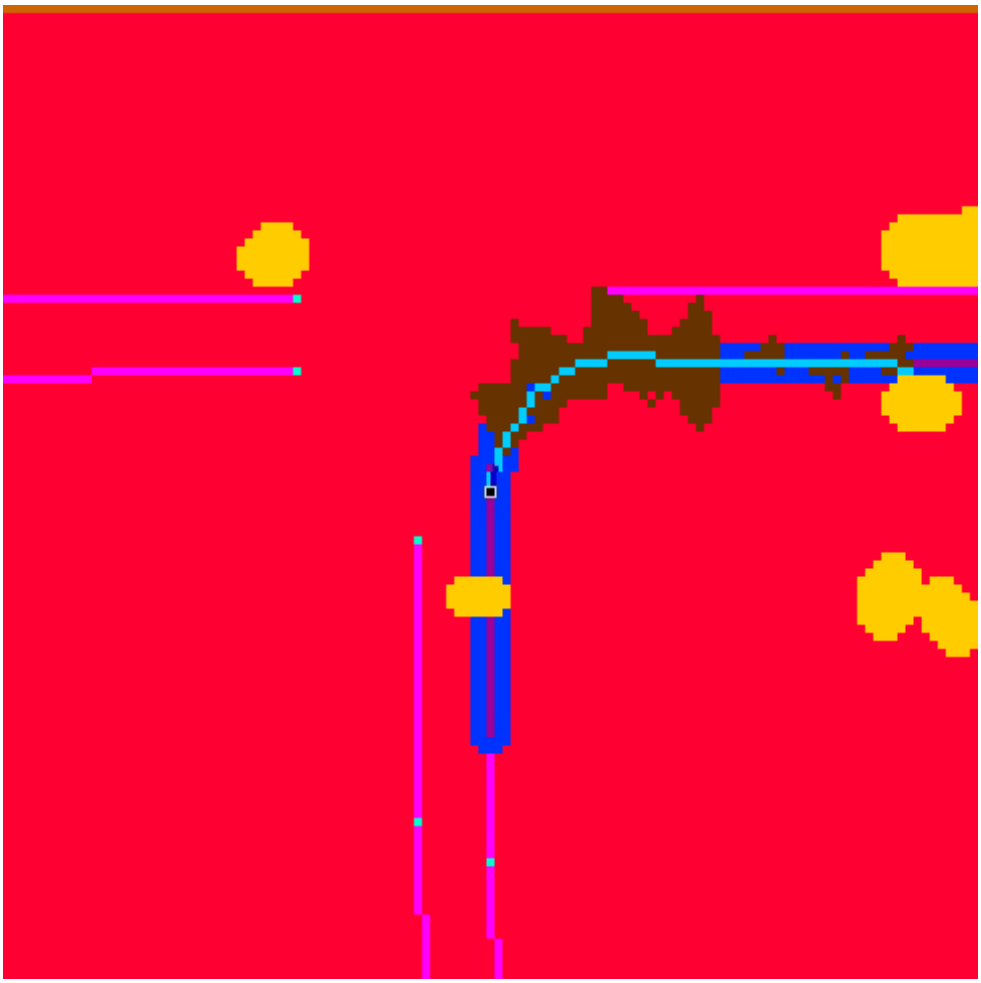

Figure 3-5. Sample output traversability grid from motion planning.

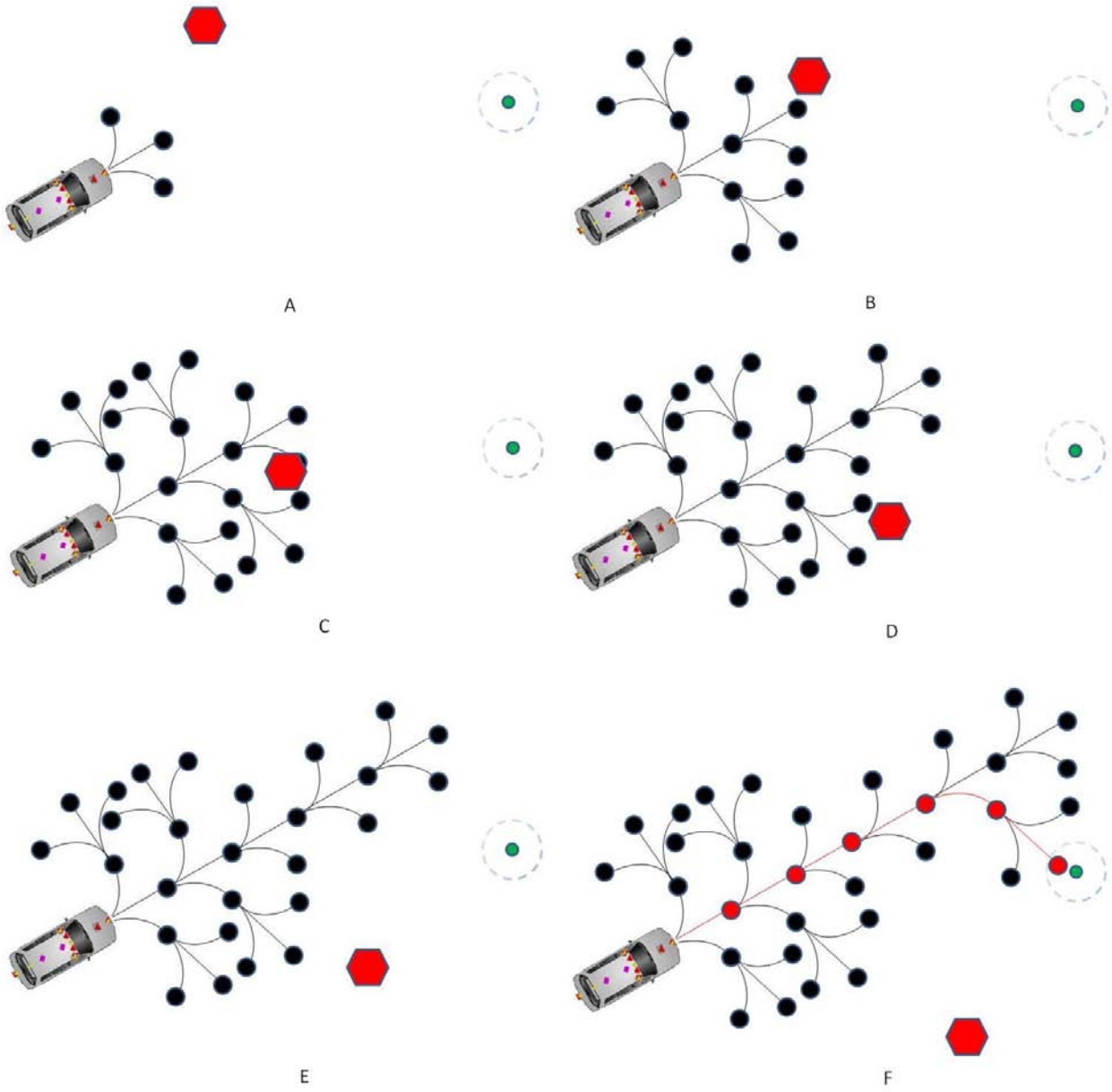

Figure 3-6. Discrete exploration steps of motion planner with no obstacle interference.

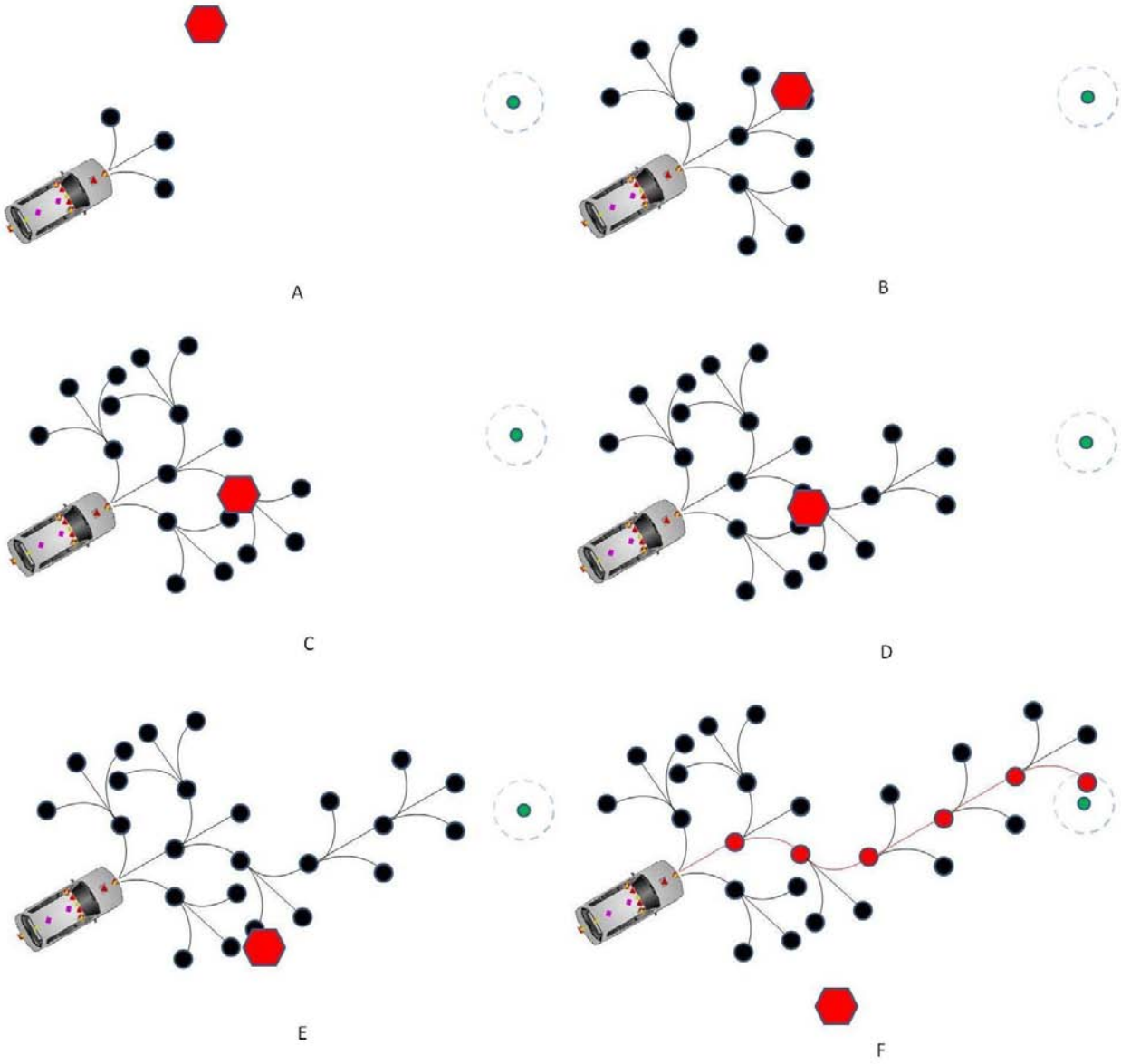

Figure 3-7. Discrete exploration steps of motion planner with obstacle interference.

CHAPTER 4
IMPLEMENTATION DETAILS

This chapter outlines the technical details of the existing implementation of the various components of the PTMP method used for the research presented in this dissertation. The ultimate objective of the presented research was the actualization of the temporal method on a real autonomous robot. The resources present at CIMAR made this goal a reality in terms of software development and the ability to test the method on a reliable robotic test platform. Firstly, one of the robotic ground vehicle platforms designed and built at CIMAR, the Urban NaviGator is described, followed by a brief background of the traversability grid (TG) developed by CIMAR in (Crane et al., 2005) as a starting point from which the temporal grid was built. Next, the software component responsible for the generation of the predictive temporal grid is discussed in detail. The temporal motion planning algorithm is then outlined in detail to show how the temporal grid was used to effectively generate trajectories for the robotic vehicle. Lastly, a unique application of the PTMP method is presented that allowed a robot to autonomously follow and intercept a target object.

Urban NaviGator

The Urban NaviGator, shown in Figure 1-1, is a fully-autonomous sport utility vehicle that was designed and developed at CIMAR for the 2007 DARPA Urban Challenge robotics competition and served as the robotic test-platform for the implementation and testing of the PTMP method presented in the previous chapter. The robot was built on a 2006 Toyota Highlander Hybrid (THH) chassis that was heavily modified to meet the requirements of autonomous navigation. The hybrid nature of the vehicle allows it to run on its internal electric power train and/or its internal combustion engine. The relevant hardware present on the vehicle

is now described, followed by an overview of the system architecture. Lastly, the software components of the architecture that are critical to the PTMP method are outlined.

Architecture

The system architecture of the Urban NaviGator is outlined in Figure 4-1 and was comprised of four main elements. The Perception Element contained all the sensor hardware and software components that provided data about the surrounding environment to the rest of the system. This included a network of three GPS units, eight laser range finders, four color cameras, and a variety of software components that use the aforementioned hardware to carry out such tasks as line-finding, moving obstacle detection, and terrain estimation. The Planning Element was responsible for using a Mission Data File (MDF), which defined the locations the robot was required to visit and the order in which the robot was required to visit them, to plan high-level and mid-level trajectories through the Route Network Definition File (RNDF), which defined the entire drivable environment as a set of GPS-based waypoints and checkpoints. The mid-level plans were provided to the low-level motion planning algorithm to calculate the goal point to which it generated its trajectory. The Intelligence Element sought to collect and analyze information about the vehicle, the environment, the status of the mission, and the available operating behaviors to best choose the next course of action for the vehicle. Lastly, the Control Element was responsible for executing low-level motion planning, which generated the control inputs implemented by the vehicle to navigate through its environment and avoid static and dynamic obstacles.

Hardware

The Urban NaviGator possessed a comprehensive sensor package used for localization, terrain estimation, and obstacle detection. This package included six SICK LMS-291 LADAR sensors and two SICK LD-LRS1000 long range LADAR sensors used for obstacle detection and

terrain classification. Also included were four Matrix Vision BlueFox USB color cameras, which were used for lane line detection and path-finding. Three GPS units were combined with a GE Aviation North-finding-module (NFM) for vehicle localization. The NFM maintained Kalman filter estimates of the position and orientation of the vehicle in its global frame of reference.

A distributed computing package replaced the rear row of seats in the Urban NaviGator for all data processing needs. The vehicle supported up to twelve ATX motherboards, of which ten were populated. The motherboards contained AMD X2 4600 processors and storage resources included four gigabyte compact flash cards and eighty gigabyte hard drives. The computers ran multiple operating systems including Ubuntu 8.04 Linux and Windows XP. Two gigabit Ethernet network switches were available for high-speed data transfer among the various computers on the robot. Lastly, in-car development was facilitated by means of a dual-head keyboard-video-mouse (KVM) switch, which allowed access to any of the ten populated computers from either of two rear-seat workstations.

Actuation of the Urban NaviGator was facilitated by a variety of means. For steering automation, an Animatics SmartMotor was attached to the steering column and received position commands that were associated with the steering commands coming from the motion planning algorithm. Throttle and brake automation was enabled through use of the existing drive-by-wire system already implemented on the vehicle. A custom controller was designed and incorporated to pass throttle and brake commands to the vehicle when it was in an autonomous state. This autonomous state was reached through a multiple-step process, at which point the vehicle was remotely activated to begin a mission.

Software

All of the various elements making up the system architecture described in the previous section were supported by multiple software components that were responsible for the various tasks associated with each architecture element. Two of the components that are critical to the success of the implemented PTMP method are now briefly described. As part of the Perception Element, the Moving Obstacle (MO) sensor was responsible for detecting, localizing, classifying, and tracking objects of interest in the robot's environment. This software component accomplished these tasks by fusing data from several of the LADARs on the vehicle. The use of these four strategically placed lasers allowed for a data field that encompassed the entire area around the vehicle, and the two long-range scanners allowed for early detection of objects on the front and sides of the vehicle. The raw range data provided by the various lasers was fused and evaluated for objects of interest that roughly match a model of a vehicle. Once one of these objects of interest was identified, its location, defined by its estimated centroid, and its velocity was calculated. As MO completed this task for the entire sensor data series, a list of potential dynamic objects was created and was made available to the rest of the system. This list of moving obstacles was used in the PTMP method to build a dataset of obstacle positions and velocities to be used by the prediction algorithm to estimate future object positions and velocities.

Other elements of the Perception element were responsible for producing traversability grids according to their respective function. These multiple TGs then had to be fused into a single, usable grid, which was the charge of the Smart Arbiter (SARB) component. The concept of traversability is explained shortly in the following section, but may be used loosely to describe the function of the SARB. After collecting the various TGs from the different components, the SARB executed an arbitration algorithm which analyzed the traversability values for a particular

cell in each of its incoming grids and selected the lowest traversability value to assign to that cell in the output TG. The lowest value was selected as a matter of safety to accommodate different sensors having different views of the environment and providing different traversability values for a particular cell. The SARB could also intelligently change how certain regions, such as the desired lane of travel, were represented in the grid as a means of aiding motion planning during different vehicle operating behaviors. The last step in the fusion algorithm was to dilate any obstacles by a size equivalent to the width of the robot, plus a buffer region, to allow for safe avoidance of these obstacles. This output grid was used as the baseline for the creation of the predictive temporal grid.

Traditional Traversability Grid

The first step in the development of the predictive temporal planning method considered building a temporal grid structure from an existing spatial grid. The spatial TG developed by CIMAR sought to create a rasterized representation of the environment around a robot similar to an occupancy grid. Rather than simply classifying cells as occupied or free, the grid allowed for variability in the level of "occupancy" by assigning each cell a value which represented how traversable the cell was. For example, a cell containing a modest incline was assigned a lower traversability value than a cell containing completely flat terrain, although a vehicle would still be able to traverse the former. Figure 4-2 displays a visualization of how the immediate environment could be viewed by a human in the top image and by a robot as a TG in the bottom image. All cells considered out of bounds are painted red and assigned the lowest possible traversability value, while the drivable corridor is painted green and assigned a high traversability value. Cells containing trees or other impassable objects are also painted red and assigned low traversability values and cells representing terrain that experiences significant changes in elevation are painted yellow and assigned intermediate traversability values

signifying that these cells are not as traversable as their neighboring cells that contain flat terrain. A cell's traversability value was determined by the fusion of data coming from the robot's sensor network. Cells containing obstacles, or which were deemed out of bounds, were assigned low traversability values, while cells that were unoccupied, flat, and in the robot's desired direction of travel were assigned high traversability values.

The TG design used during the presented research called for 121 rows (0 to 120) and 121 columns (0 to 120) with the vehicle located in the center cell at location (60, 60). Each cell represented an area of 0.5 m by 0.5 m square, resulting in a grid of size 60 m by 60 m square. With the vehicle located in the center of the grid, this grid provided data for a range of approximately 30 m around the vehicle. The traversability values of each of the 14,641 cells of the grid were stored as unsigned character data types, resulting in each grid requiring just less than 15 kilobytes of memory. The grid coordinate system was aligned with the Universal Transverse Mercator coordinate system such that the vertical axis of the grid represented the Northing direction and the horizontal axis represented the Easting direction. As the grid coordinate frame was fixed, the environment in turn rotated around the origin according to the vehicle heading direction.

A special data structure, termed the torus buffer, was previously implemented to store the data associated with the traditional TG. This structure, while essentially a linear data array, allowed for easy shifting of the grid's traversability values as the vehicle moved by connecting the last element of each row with the first element and the last element of each column with the first. By comparing the current global location of the grid center with the current global position of the vehicle, row and column shift values could be calculated and the corresponding number of rows and columns at the edge of the grid could be re-initialized as the vehicle moves.

Considering a single column of twenty grid cells in Figure 4-3(a), attaching the last cell to the first cell resulted in the ring buffer data structure depicted in Figure 4-3(b). Extending this out to twenty columns resulted in the linear array of ring buffers shown in Figure 4-3(c). And finally, Figure 4-3(d) shows a visualization of the torus buffer after the last ring buffer was connected to the first ring buffer in the array.

The simple nature of the traditional TG made it possible for each sensing component on the robot to produce their own TG, which reported their specific view of the environment. For example, a camera-based vision component could be used to detect the lines on the edges of the desired lane of travel and paint them in a TG, while a laser-based sensor could concentrate on terrain estimation and report a TG based on its findings. With multiple grids coming from the various components, it became necessary to fuse them into a single output grid. In general, the arbitration algorithm observed the multiple traversability values for a given cell in each of the grids provided by the individual sensor components and assigned the lowest value as the traversability of that particular cell in the output TG, as described in the previous section.

The arbitrated TG was then used by the motion planning algorithm to generate optimal control inputs to the vehicle. Figure 4-4 shows an example of an input TG with the center of the desired lane of travel painted in green and assigned the best traversability value possible, and the extent of the desired lane painted in blue with a slightly lower traversability value. All regions of the grid outside of the desired lane of travel are painted in red with a low traversability value. The yellow cells surrounded by green represent an obstacle and are assigned untraversable values. Referring back to Figure 3-3 provides an example of an output TG from the motion planning algorithm. The image depicts the vehicle navigating a sharp right turn through an

intersection. The areas painted brown represent the areas of the grid explored by the motion planning algorithm, while the light blue line represents the selected optimal solution path.

A shortcoming of this traditional TG, which was one of the motivating factors behind the development of this new temporal method, must be addressed. This shortcoming dealt with the fact that the grid represented an instantaneous snapshot of the robot's environment where all objects were treated as static. This snapshot was used by the motion planning algorithm to generate a sequence of control inputs to navigate the robot from its current state to its desired goal state, which was likely at the edge of the TG. Each of these control inputs were calculated at discrete times in the future until the goal state was achieved. This sequence of inputs was then traced back to the first input, which was the instantaneous control implemented by the vehicle. The final step in this sequence, which achieved the goal state, could have represented a control input required at a time falling ten to fifteen seconds or more in the future, depending on the speed of the vehicle. This practice of planning a path through a dynamic environment that was being artificially treated as static could have led to instabilities in the control inputs and could have been fatal when considering the close proximity of the robot to moving obstacles. The presented research provides a new approach that addresses this drawback to the use of the traditional TG when trying to plan optimal paths through environments with dynamic objects.

Temporal Grid Creator

The Temporal Grid Creator (TGC) component was responsible for generating the predictive temporal grid structure that was the backbone of the presented motion planning method. Inputs included vehicle position and velocity measurements, the fused traversability grid coming from the SARB, the list of moving objects in the environment coming from MO and the values of the time-dependent search parameters that are described in Chapter 3 coming from the motion planning algorithm. The structure of the component was separated into multiple

processor threads based on its three main functions to allow for concurrent execution: the collection of moving obstacle data, the prediction of future moving obstacle positions, and, finally, the creation of the predictive temporal grid. Each of these functions are outlined in detail, along with a discussion of the specific motion planning algorithm search parameters that are described in general terms in the previous chapter and how they were used in the creation of the temporal grid structure.

Obstacle Position Data Collection

As mentioned previously, one of the inputs of the TGC component was the list of detected moving obstacles reported by MO. Information in this list included a unique identifier, an obstacle classifier declaring an object as a car, bus, tree, or the like, an x velocity and a y velocity, an array of x and y positions that defined the left-most, right-most, and closest points of the obstacle, and a time-stamp for the data for each obstacle. MO provided this information for all objects within a range of approximately 150 m around the vehicle in every direction. However, as the maximum dimensions of the new temporal grid structure were set to 30 m around the vehicle to match those of the traditional TG, the TGC only maintained a list of the objects within a range of approximately this size and disregarded all objects that were further away.

The TGC used this information to build a time-stamped data set for each obstacle present in the environment. While an obstacle could be defined by MO as multiple points at the extents of its visibility, the TGC calculated and assigned the obstacle position as the centroid of these extents. As it was building these data sets, the algorithm shifted the actual time-stamp values so that the oldest data point was associated with an initial zero-time. The shifted time-stamps of the subsequent data points were determined by calculating the difference between the un-shifted time-stamp value of a particular data point and that of the oldest data point.

The number of data points to be saved was an adjustable parameter which was set to 100 samples for this implementation. This value was selected to improve the accuracy of the prediction model for each obstacle, while being small enough to still allow for fast convergence of the model. When the number of total data points reached this limit, the oldest data point already in the collection was removed to make room for an additional data point to be added. Assuming that the algorithm update rate was equal to or near the desired rate of 40 Hz, this data collection represented a time-history of approximately 2.5 seconds for each obstacle. This data collection with shifted timestamps could then be used to estimate a prediction model for the future motions of the obstacles.

Search Parameters

To reiterate, the developed temporal grid extended the spatial TG into the time dimension by adding layers to the grid structure. Each temporal layer of the new grid represented how the robot's environment would have likely appeared at a distinct time-step in the future, according to the obstacle prediction models described in the previous section. The number of temporal layers in the new grid and the size of the time-steps between layers were determined by several parameters associated with the motion planner's search algorithm. Therefore, it is again prudent to provide a brief overview of the motion planning algorithm implemented on the Urban NaviGator and these key parameters in particular. The algorithm utilized for the development of this temporal method combined a receding horizon control strategy with an A* heuristic search algorithm to expand out a tree of search nodes, which represented potential vehicle states that could be reached in the future according to specific control inputs. The state of the robot was defined as:

$$s = \begin{bmatrix} x \\ y \\ \psi \\ v \\ \phi \end{bmatrix} \tag{4-1}$$

where x and y are the position of the state in the vehicle frame of reference, ψ is the vehicle heading angle, v is the desired vehicle velocity and ϕ is the desired steering effort for the given state. The steering effort was defined for a range from -100% to 100%, representing the steering wheel achieving full-left and full-right soft-lock, respectively. These components of the potential states represented in Equation 4-1 were estimated from a kinematic model of the robot and were generated according to prospective steering commands, which were one of the control inputs to the vehicle. The geometry used in the model is based on a bicycle (two-wheeled) model and is shown in Figure 4-5 and the kinematic equations of motion are given as:

$$\Delta x = \frac{\cos\psi_0 \cdot (1-\cos\Delta\psi) + \sin\psi_0 \cdot \sin\Delta\psi}{k_{avg}} \tag{4-2}$$

$$\Delta y = \frac{\cos\psi_0 \cdot \sin\Delta\psi - \sin\psi_0 \cdot (1-\cos\Delta\psi)}{k_{avg}} \tag{4-3}$$

$$\Delta\psi = \Delta d \cdot k_{avg} \tag{4-4}$$

where ψ_0 is the current vehicle heading angle, $\Delta\psi$ is the change in vehicle heading angle which is instituted by the steering effort,ϕ, Δd is the distance-step equivalent of the generation time-step, Δx and Δy are the change in vehicle position in the vehicle frame of reference, and $k_{avg}$is the average curvature of the path travelled by the vehicle for a commanded steering angle of ϕ and a vehicle wheelbase of length l_w, and is defined as:

$$k_{avg} = \frac{\tan\phi}{l_w}. \tag{4-5}$$

The arc length S from Figure 4-5 represents the time-step (or equivalent distance-step) between the current vehicle state and one of the potential future vehicle states that could be reached from this current state as expanded by the search algorithm. This step size was one of

the key parameters used in determining the number of layers that the temporal grid contained. It should be noted that the step size did not need to be constant for every generation of search nodes expanded. It was determined during previous development of the motion planning algorithm that a unique step size for the initial expansion of search nodes was beneficial, followed by a consistent step-size for all subsequent generations. This was caused by the fact that the initial generation of expanded search nodes had the most impact on the selected control input for the vehicle. This standard was adopted for the development of the temporal grid structure for the presented research. The search time horizon was the other important parameter and represented the maximum future-time to which the motion planning algorithm expanded search nodes before terminating. The goal state for the planning algorithm was placed at this time horizon, which typically corresponded to the edge of the TG.

The number of layers for the temporal grid was calculated by determining the number of steps required to reach the goal at this time horizon. A visualization of the expansion of the prospective future vehicles states is provided in Figure 4-6 with the initial node expansion time-step Δt_0 normal node expansion time-step Δt_n and the search time horizon T_H shown. It is important to emphasize that these time-steps and time horizons had equivalent distance-steps Δd_0 and Δd_n and distance-horizon D_H that could be used in their place. These values could be used interchangeably by considering the velocity of the vehicle in the x-direction v_x of the vehicle coordinate system, which pointed in the direction of travel. The conversion was calculated as:

$$\Delta d_0 = \Delta t_0 \cdot v_x \quad (4\text{-}6)$$
$$\Delta d_n = \Delta t_n \cdot v_x \quad (4\text{-}7)$$
$$D_H = T_H \cdot v_x \quad (4\text{-}8)$$

The vehicle is depicted with the goal state painted green at the edge of the image in Figure 4-6. Each of the black circles represents a potential vehicle state that was explored by the planning algorithm, while the red circles and lines represent the solution sequence of state changes which will bring the vehicle from its current state to the goal state. The curving, dashed line represents the search time horizon, with the initial and normal time-steps displayed as well. The number of temporal layers N_l was calculated as:

$$N_l = 1 + (\frac{T_H - \Delta t_0}{\Delta t_n}). \tag{4-9}$$

In the event that the calculated number of temporal layers coming from Equation 4-9 did not result in an integer, the value was rounded to the next integer.

An obvious concern with this approach was that this value for the number of temporal layers was not necessarily constant during a mission as the time horizon or node expansion time-step size could change during runtime as the robot's behavior or speed changed. For this reason, the planning algorithm was required to report any changes to these key search parameters so the temporal grid structure could be updated. Another concern arose as this method assumed that the number of expanded nodes along the solution path was the minimum number required to reach the goal state at the time horizon. While this assumption may not have been realistic for environments heavily occupied by obstacles, either static or dynamic, that the motion planning algorithm will have to avoid, it was acceptable for the initial development, testing, and validation of the temporal planning method for representing simple dynamic environments with perhaps a single obstacle. It was also acceptable because the remote state changes approaching the time horizon had less impact on the implemented control than the initial state changes. For the initial development, any expanded search node generations in excess of N_l relied on the last available temporal layer in the grid.

Temporal Grid

With the designation of the aforementioned search parameters, the temporal grid structure could then be generated. The simple approach would have been to build an entire new TG of size 121 rows by 121 columns for each new temporal layer and to have modified each layer as needed. By fully populating each temporal layer of the grid, this method would have required large amounts of memory and required significant amounts of time to create and copy the grid structures as evidenced by the preliminary grid test results shown in Tables 4-3, 4-4, and 4-5. The selected solution to this problem arose from considering the same key search parameters used to construct the temporal grid.

The initial generation of search nodes expanded by the A* algorithm extended out a time in the future according to the defined step-size, as previously described. As a best-first search procedure, the A* algorithm then evaluated these nodes and assigned them a cost to organize them in a way such that the lowest-cost node was determined for the next round of expansion. The motion planning algorithm used for the presented research estimated these costs based on the distance from each node to the goal state and the traversability values of the grid cells connecting the node and its predecessor node. It then became obvious that the grid cells at the search time horizon had no affect on the evaluation of these initially expanded search nodes. Therefore, for this first temporal layer associated with the initial expansion of search nodes, the grid only needed to be constructed out to the time-step corresponding to the node expansion. That is, the grid layer associated with a particular generation of search node expansion only needed to be built far enough to include the cells within which the nodes of that generation resided so that they could be evaluated by the A* algorithm.

The size of the temporal grid structure was minimized by varying the size of each temporal layer according to the time-step parameters provided by the motion planning algorithm. Each

temporal layer was built-up so that the cells containing the search nodes for that particular generation were present. For each successive step, the layer became larger and larger up to the time horizon, which would have most likely resulted in a fully-populated grid layer of 121 rows by 121 columns. The number of rows and columns in a particular temporal layer n_{rc} were calculated as follows:

$$n_{rc} = rint(\frac{2\cdot(\Delta t_0 + n_t \cdot \Delta t_n)}{R_g}) \tag{4-10}$$

where Δt_0 and Δt_n have been defined in the previous section as the initial and normal node expansion time-steps,n_t is the temporal layer number, R_g is the grid resolution, and $rint()$ is a function that rounds the result to the nearest integer value according to the prevailing rounding rule. This calculation worked for determining either rows or columns since the values were equal for a square grid. Table 4-1 displays the initial row and column and final row and column for each layer of a sample temporal grid. The sample grid was built from values of Δd_0 = 6.0 m, Δd_n = 4.0 m, and D_H = 42.0 m. From the seventh temporal layer until the last, the layers are fully populated with 121 rows and 121 columns. This comes about as the total distance exceeded the maximum possible dimensions of the grid.

This method of optimization gave the temporal grid structure a pyramidal shape. Figure 4-7 shows a representation of this pyramidal-shaped grid structure and Figure 4-8 shows the profile of each of the layers of an optimized temporal grid. The initial layer in Figure 4-8(a) represents the environment out to the initial time-step, while the layer depicted in Figure 4-8(g) represents the entire environment known to the robot out to the search time horizon. A static obstacle begins appearing in Figure 4-8(d) and comes into full view in Figure 4-8(f). This serves to reemphasize that the nodes expanded in the first generation were not impacted by objects further out in the environment. Because the obstacle was not relevant to the evaluation of the first few

generations of expanded search nodes, it was not necessary to include that area in their associated temporal layers.

As the torus buffer data structure was used to store and modify the traditional TG, an amended structure was required to store and modify the data for the temporal grid structure. The temporal torus buffer was used and behaved much the same as the traditional torus buffer structure. With the inclusion of additional temporal layers in the temporal grid, the last element of each row and column was to be connected to the first element of each row and column for each temporal layer in the grid.

The images in Figure 4-9 show conceptually the similarities between the progression from a simple single column of grid cells to the torus buffer and the progression from the single temporal column of cells to the temporal torus buffer described above. The column of cells is extended into the time-dimension in Figure 4-9(a) and assumes that each temporal layer contains the same number of cells, while the temporal ring buffer is displayed in Figure 4-9(b). The outer-ring represents the first temporal layer and each subsequent inner-ring represents the next layer. The linear array of temporal ring buffers is displayed in Figure 4-9(c) followed by a segmented view of the temporal torus buffer in Figure 4-9(d). This broken view serves to show the temporal layers within the torus buffer structure. These images represent a grid containing three temporal layers.

Even considering the views of the temporal torus buffer discussed above, the grid data was still stored as a linear array of traversability values. Careful attention was taken to guarantee that the correct element in the array was being set or modified when altering the grid. For the temporal torus buffer the correct element of the array n_{cell} was calculated as:

$$n_{cell} = \sum_{i=0}^{n_l-1}(n_{row}[i] \cdot n_{col}[i]) + (r - r_{start}[n_l]) \cdot n_{col}[n_l] + (c - c_{start}[n_l]) \qquad (4\text{-}11)$$

where n_l is the temporal layer in which the desired cell resides, $n_{row}[\cdot]$ and $n_{col}[\cdot]$ are the number of rows and columns in a particular temporal layer, r and c are the row and column number of the desired cell, and $r_{start}[\cdot]$ and $c_{start}[\cdot]$ are the starting row and column number for a particular temporal layer. This calculation was used in multiple applications, including setting and retrieving the traversability value of a temporal grid cell.

Initial testing was conducted to study several important values associated with grid generation. The creation times t_{create}, cloning times t_{clone}, and memory requirements n_{bytes} were recorded and compared for the traditional TG, fully-populated temporal grid and optimized temporal grid structures. These tests were carried out for temporal grids with the number of temporal layers ranging from one to ten. The search parameters utilized for the initial testing consisted of an initial distance-step of $\Delta d_0 = 6.0$ m and a normal time-step of $\Delta d_n = 4.0$ m. The search distance-horizon was varied from $D_H = 6.0$ m to $D_H = 42.0$ m to allow for the appropriate number of temporal layers for the initial tests. The actual values are displayed in Table 4-2 for $N_l = 1 \ldots 10$.

The creation times for the three grid types are plotted in Figure 4-10 and listed in Table 4-3. The creation time for the traditional TG remained approximately constant as would be expected since this grid always contained only a single grid layer, with an average time of t_{create} = 0.1866 ms. The creation times for the fully-populated temporal grid increased linearly from an initial time of t_{create} = 0.2223 ms for a single grid layer to t_{create} = 2.322 ms for ten temporal layers. Lastly, the creation times for the optimized temporal grid followed an approximate second order increase from t_{create} = 0.01490 ms for a single temporal layer to t_{create} = 1.423 ms for all ten layers. Of interest was the smaller creation time for the optimized temporal grid than the traditional TG. This arose because the total number of cells in the

temporal grid did not exceed that of the traditional TG until the fourth temporal layer was added. The creation time for an optimized temporal grid with ten temporal layers represented a 540% increase from the single layered traditional TG; however, the creation time for an optimized temporal grid with only seven layers of t_{create} = 0.7579 ms resulted in only a 241% increase and represented a 112% reduction of the creation time of the fully-populated temporal grid with the same number of temporal layers. This seven-layered temporal grid coincided with the selected search time-step parameters that were used for this study and thus provide the most relevant comparison.

The second test studied the cloning times of the three different grid structures. Specifically, this test measured the time required to access the traversability value of each cell in one of the grids and set the traversability value of the same cell to be equal to the assessed value in a separate grid of the same type. Figure 4-11 displays the trends for the three grids and Table 4-4 lists the actual recorded times. As with the creation times, the cloning times of the traditional TG remained approximately constant, with an average of t_{clone} = 0.07160 ms, while those of the fully-populated and optimized temporal grids followed an approximate second order curve. The fully-populated temporal grid's clone times ranged from t_{clone} = 0.03662 ms for a single layer to t_{clone} = 1.1405 ms for ten layers. Likewise, the cloning times of the optimized temporal grid ranged from t_{clone} = 0.03740 ms to t_{clone} = 0.6715 ms. The cloning time for a temporal grid with seven temporal layers of t_{clone} = 0.2690 ms represented a 275% increase over that of the traditional TG, but was a 53% reduction when compared to that of the fully-populated temporal TG.

Lastly, the memory usage was recorded for all three grid types for the same number of temporal layers as was used in the other tests. While the issue of storage space was not a

significant problem, the sizes of these grids were still of interest because of the high rate of transmission of these grids among various software components on an unmanned vehicle. Table 4-5 shows the actual recorded size of the grids versus the number of temporal layers, while Figure 4-12 displays the trends in the data. The traditional TG required 14,641 bytes of memory, which obviously remained constant. The fully-populated temporal grid's memory requirement increased linearly from 14,641 bytes for a single layer to 146,410 bytes for ten layers. And finally, the optimized temporal grid only required 961 bytes for a single layer and 93,290 bytes for ten temporal layers. For an optimized temporal grid with seven temporal layers, the memory requirement of 49,367 bytes represented a 237% increase over that of the traditional TG, but was a 51% reduction in usage compared to the fully-populated temporal TG.

Moving Obstacle Prediction

Since the prediction of the moving objects was not the focus of the presented research, a modified version of the statistics-based polynomial predictor discussed in (Kent, 2007) was used to fit curves to the time-series of position and velocity data for each obstacle reported by MO. A second-order polynomial was chosen for simplicity and ease of use and its coefficients were calculated by using the GNU Science Library's (GSL) regression tools (GNU, 2008). A more complex fit could have been chosen to provide a more accurate estimation of the future positions and velocities, but as the development of a prediction algorithm was not the focus of the research, a simple model was sufficient. The data sets described in the previous section were parsed into vectors of x positions, y positions, x velocities, y velocities, and a matrix of shifted time-stamps. The GSL function gsl_multifit_linear() took these vectors and matrices and returned the solution coefficient vector for fitting the second-order polynomial to the data. From this polynomial, data points were extrapolated into the future to provide the predicted positions of the obstacles to be painted in the temporal grid. Figure 4-13 shows a visualization of the fitted

polynomial to the time-series of data for an obstacle and the extrapolated future position estimates, along with upper and lower confidence intervals.

For each obstacle, this algorithm estimated the predicted x_t and y_t position at future time t according to:

$$x_t = \beta_{x_0} + \beta_{x_1} \cdot t + \beta_{x_2} \cdot t^2 \quad (4\text{-}12)$$

$$y_t = \beta_{y_0} + \beta_{y_1} \cdot t + \beta_{y_2} \cdot t^2 \quad (4\text{-}13)$$

where β_{x_i} and β_{y_i} are the coefficients of the polynomial curve fits, which are calculated by the GSL tools. The prediction times used to determine the values of x_t and y_t again came from the search parameters which were used to build the temporal grid structure, namely the generational time-steps of search node expansion and the search time horizon. For each generation's time-step, the position of each obstacle was estimated using Equations 4-12 and 4-13. The predicted obstacle positions were then painted in the layer of the temporal grid associated with that prediction time by converting the predicted position into a row and column pair by using Equations 3-16 and 3-17 from the previous chapter.

Figure 4-14 shows the results of the initial, fully-populated temporal grid construction with the predicted positions of a simulated obstacle painted as it moved perpendicular to and crossed the robot's desired lane of travel. The vehicle's direction of motion, or heading angle, is labeled as ψ in this figure as well as all subsequent grid images. Figure 4-14(a) displays the representation of the predicted environment after the initial time-step of the search algorithm, while Figure 4-14(g) represents the final prediction of the environment as the search algorithm reached its time horizon. This figure serves to show the successful mapping of the predicted motion of a moving obstacle. A sample of the optimized temporal grid, which was provided to the motion planning algorithm, is shown in Figure 4-15 with the robot's heading facing toward

the left of the image pointing down the desired lane of travel painted in blue. In both instances, the obstacle was simulated with a velocity of 2.2352 mps (5 mph) in the vertical (Y) direction.

The obstacles were painted in the temporal grid according to their exact predicted positions. The cell represented by the row-column pair calculated from Equations 3-16 and 3-17 was assigned the lowest traversability value possible to show the obstacle. As a result of the vehicle being treated as a point, the predicted position of the obstacle was then dilated by an area equivalent to the width of the robot. Therefore, the surrounding cells were also assigned the lowest traversability value possible out to an area the size of the vehicle, plus a variable buffer region to allow for a safe avoidance distance. Figure 4-16 shows a single grid image with all the predicted positions of a moving obstacle painted with only the dilation described above to show the progression of the object across the lane. The layers of an optimized temporal grid are shown in Figure 4-17 with the individual predicted positions painted to show what the motion planning algorithm was provided with to attempt to generate a plan with the moving obstacle present with predicted positions painted.

Temporal Motion Planner

The culmination of the presented research came as the motion planning algorithm generated its sequence of control input steering commands by searching through the temporal grid described in this chapter. The motion planning algorithm developed in (Galluzzo, 2006) provided the backbone for the development of this new temporal planning method. This algorithm combined a receding horizon control strategy with the A* algorithm to simultaneously generate steering, throttle, and brake commands for an unmanned ground robot. This algorithm allowed the robot to successfully navigate a road course and avoid static obstacles in a fairly unstructured environment. The temporal method presented extended the functionality of this

original algorithm to deal with both static and dynamic obstacles in structured and unstructured environments.

Referring to the previous discussion of the search algorithm, a tree of search nodes representing potential future vehicle states was expanded, with each of these nodes falling within a cell in the grid. Costs were then assigned to each node based on a number of factors. The A* search algorithm called for a two-part cost taking the form:

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \tag{4-14}$$

where $\hat{f}(n)$ is the estimated total cost of node n, $\hat{g}(n)$ is the actual cost to get from the initial node n_0 to node n, and $\hat{h}(n)$ is the estimated cost to get from node n to the goal. The original algorithm from (Galluzzo, 2006) and the presented temporal algorithm both estimated these costs based on the traversability values of the grid cells between a particular node and its predecessor node and the distance from the node to the goal point. Therefore, it was necessary to access these values in the torus buffer or temporal torus buffer data structures.

For the temporal method, this was made possible by utilizing special functions to which the node generation and row and column of the cell of interest were provided. These values were then used to access the grid cell in the linear data array of the temporal torus buffer according to Equation 4-11. The node generation was used to determine the correct layer of the temporal grid structure to access. For example, a search node generated in the initial expansion required the traversability value of its containing cell in the initial temporal layer to correctly calculate its associated cost, while a node generated in the second expansion of nodes required the value of its containing cell in the next temporal layer, and so on. Therefore, with each new generation of search nodes expanded, the temporal planning algorithm was required to search deeper in the temporal grid structure, or equivalently further out in time, to calculate the cost of each node.

This ability to access the traversability values of any given cell allowed the first component of a node's estimated cost $\hat{g}(n)$ to be calculated as:

$$\hat{g}(n) = \frac{\sum_{i=1}^{n_g}\sum_{j=1}^{n_c}(2^{63-T_v[j]})\cdot\Delta d}{n_{c_0}\cdot C_{MPC}} \tag{4-15}$$

where n_g is the generation number of node n, n_c is the number of grid cells traversed between node n and its predecessor node, $T_v[\cdot]$ is the traversability value associated with a particular grid cell, Δd is the distance step between node n and its predecessor, C_{MPC} is a constant representing the number of meters traversed per cell (0.5 m), and n_{c_0} is the number of grid cells traversed from the start node n_0 to node n. Similarly, the second component of the node's estimated cost $\hat{h}(n)$ was determined as:

$$\hat{h}(n) = \frac{Cost_{avg}\cdot d_g}{C_{MPC}} \tag{4-16}$$

where $Cost_{avg}$ is the average cost per grid cell from the start node to the node n and d_g is the distance from the node n to the goal. The total cost resulting from Equation 4-14 had abstract, dimensionless cost units which were compared for the various search nodes.

Upon finding a successful path from the start node to the final goal state, the solution path was traced back to the initially expanded node. The desired steering command associated with this initial search node on the solution path was then selected as the control input to the vehicle. Figure 4-18 displays the temporal layers of a single, fully-populated temporal grid, which show the expansion of the search tree as an obstacle crossed the desired path of the vehicle. Each black line represents an expanded search node, with the solution path painted in grey. Figure 4-19 shows the same situation in the temporal layers of an optimized grid.

Target Interception Application

All previous sections of this chapter discuss the development of the temporal motion planning method in terms of avoiding moving obstacles in the robot's environment. However, another application, which presented itself during the development and testing process, considered the opposite goal. Rather than treating an object as an obstacle, it was advantageous to treat this object of interest as a target to be intercepted and possibly neutralized. Following the same procedure of detecting and tracking the target made it possible to apply the prediction algorithm described previously to estimate the future positions and velocities of this target object. At this point, the predicted positions of the object were used to generate a goal point for the motion planning algorithm to seek out.

The original motion planning algorithm calculated its goal point based on a list of waypoints that followed the robot's desired lane of travel, and that it was required to visit. This goal point was typically located at the edge of the grid unless the vehicle was approaching an intersection or a curve in the road, or the end of its mission. For a target interception application, the goal point was calculated as the final predicted position of the object of interest at the search algorithm's time horizon. Using the final predicted position allowed the planning algorithm to direct the vehicle toward the location where it was believed the target would be in the future at the search time horizon, rather than where the target was currently located.

Depending on the speed of the moving target, this last predicted position might have been outside of the motion planner's search time horizon, in which case the target-defined goal point was snapped to the edge of the valid search space. This provided an attainable goal that was along the same heading as the final predicted position of the target. In this application, the motion planning algorithm directly required the prediction information for the target object to select this final predicted position and calculate the goal point.

In addition to generating the goal point based on the predicted position of the target object, the temporal grid was also able to aid the motion planning algorithm in generating a trajectory to intercept the target. In treating an object of interest as a target as opposed to an obstacle to be avoided, the object was represented differently in the temporal grid map. To reiterate, grid cells containing objects the robot needed to avoid were assigned low traversability values, which translated into a very high cost for a node that falls within that cell. For a target interception application, the cells that contained the final predicted position of the target did not need to be assigned untraversable values, but rather high traversability values to attract the planning algorithm to generate a path to this point.

This chapter describes the specific implementation of the new PTMP method for an unmanned ground vehicle tasked with navigating dynamic urban environments. The new TGC software component provided much of the functionality of the new method, along with the modified temporal motion planning algorithm. The next chapter describes the testing procedure and analyzes and discusses the results of the various tests conducted.

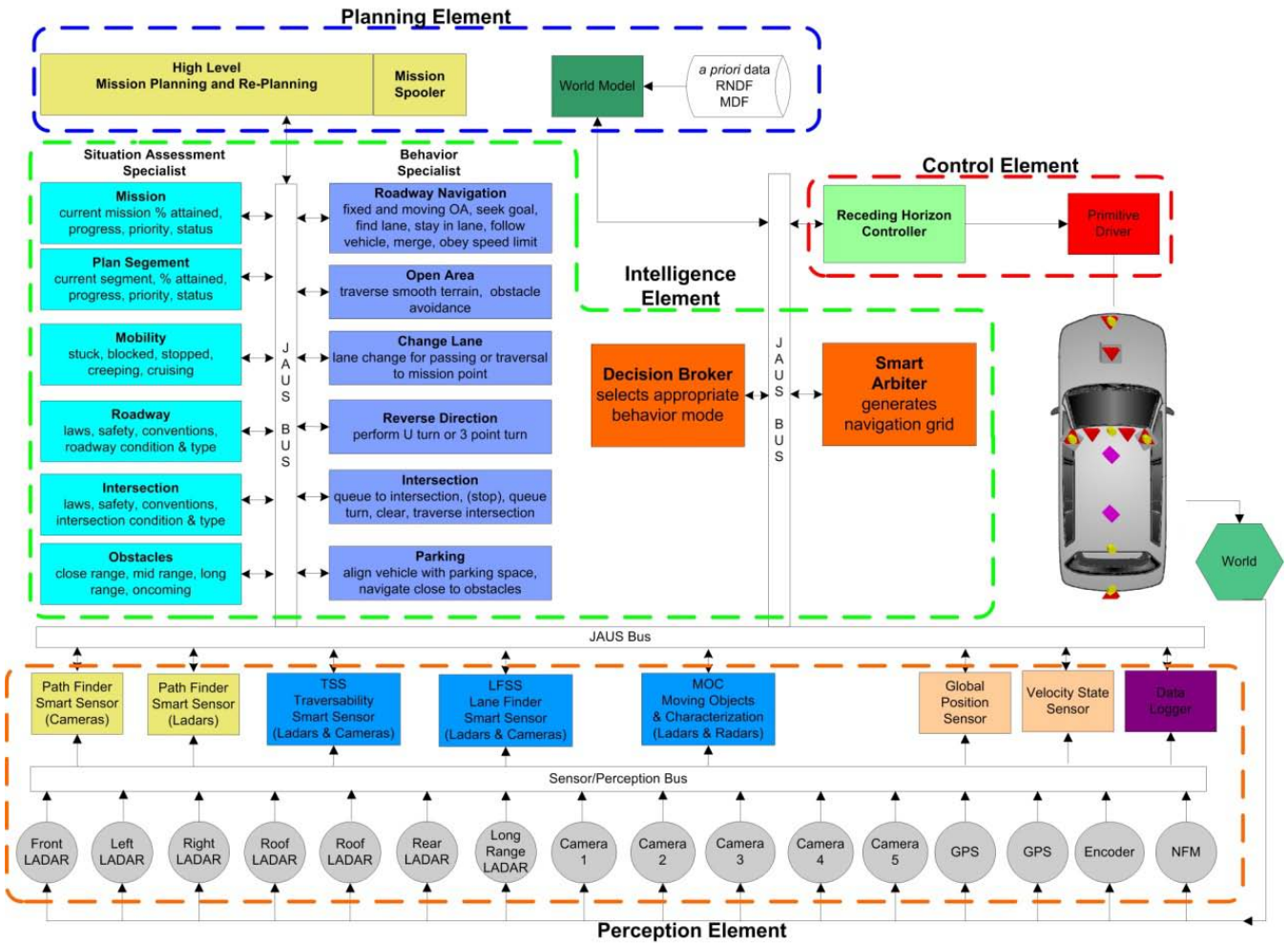

Figure 4-1. Urban NaviGator system architecture diagram.

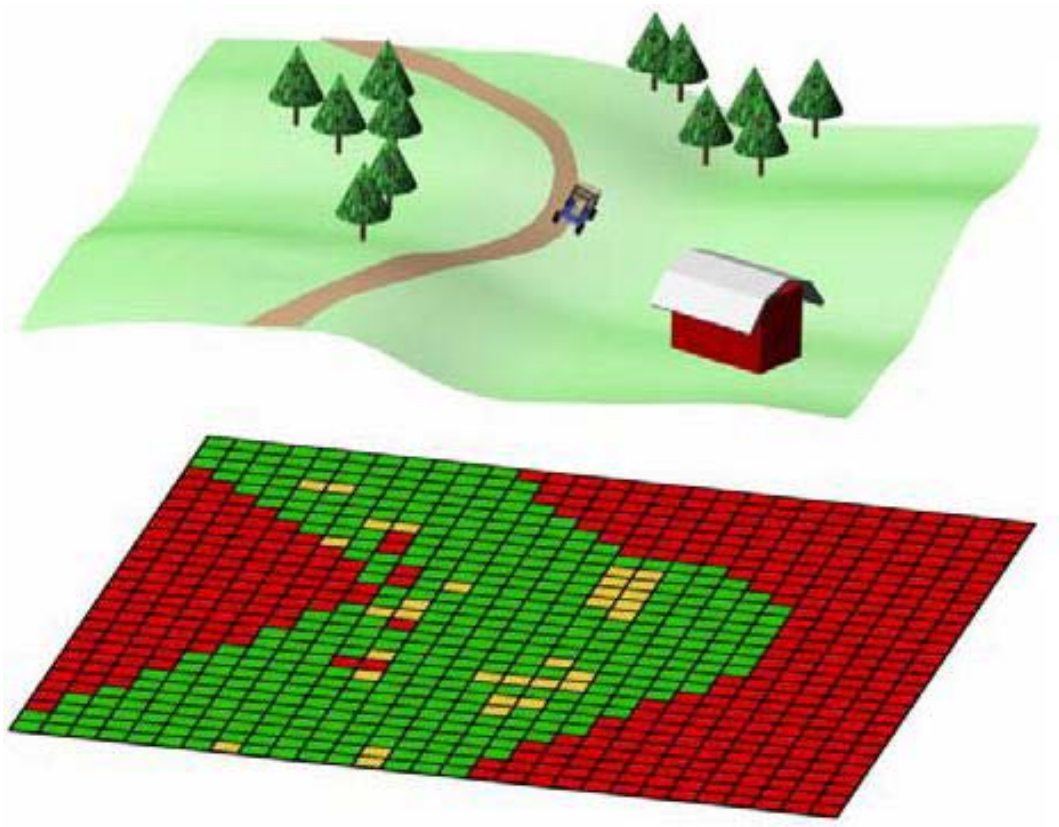

Figure 4-2. Traversability grid representation of robot's environment.

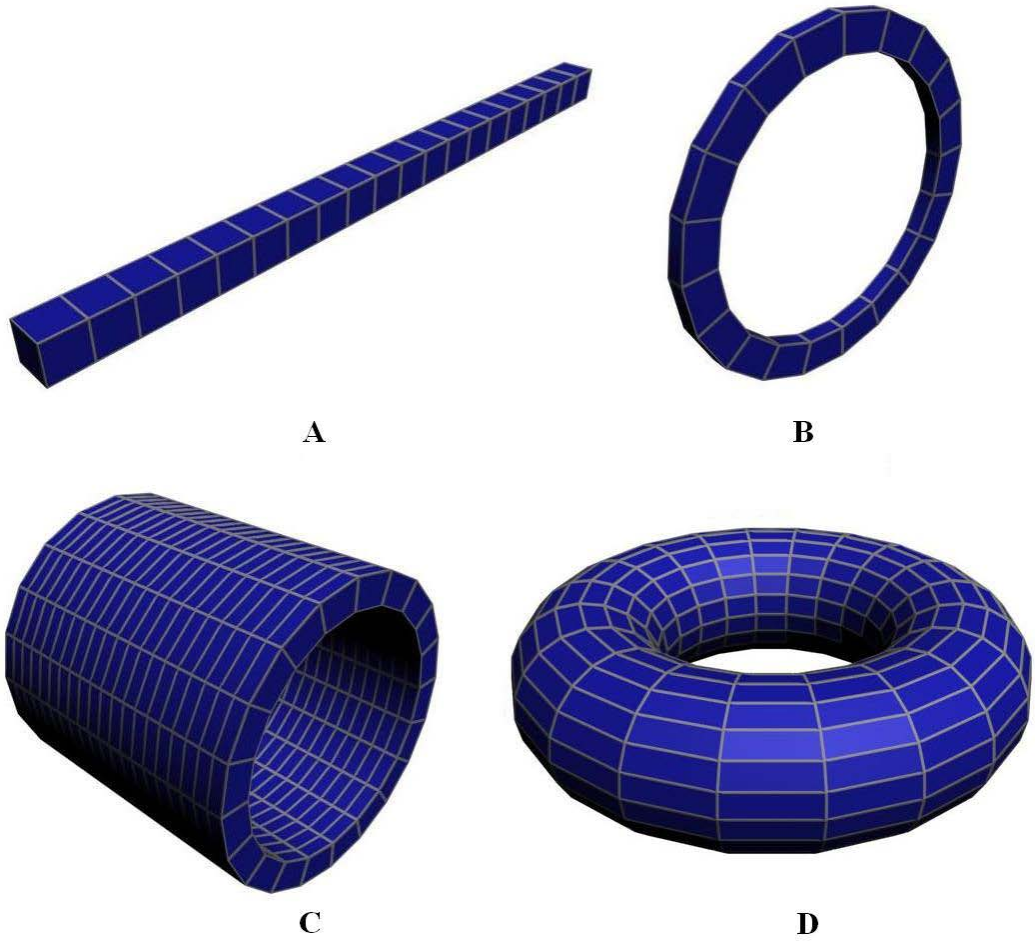

Figure 4-3. Formation of torus buffer. A) Single columns of grid cells, B) ring buffer, C) ring buffer array, D) torus buffer.

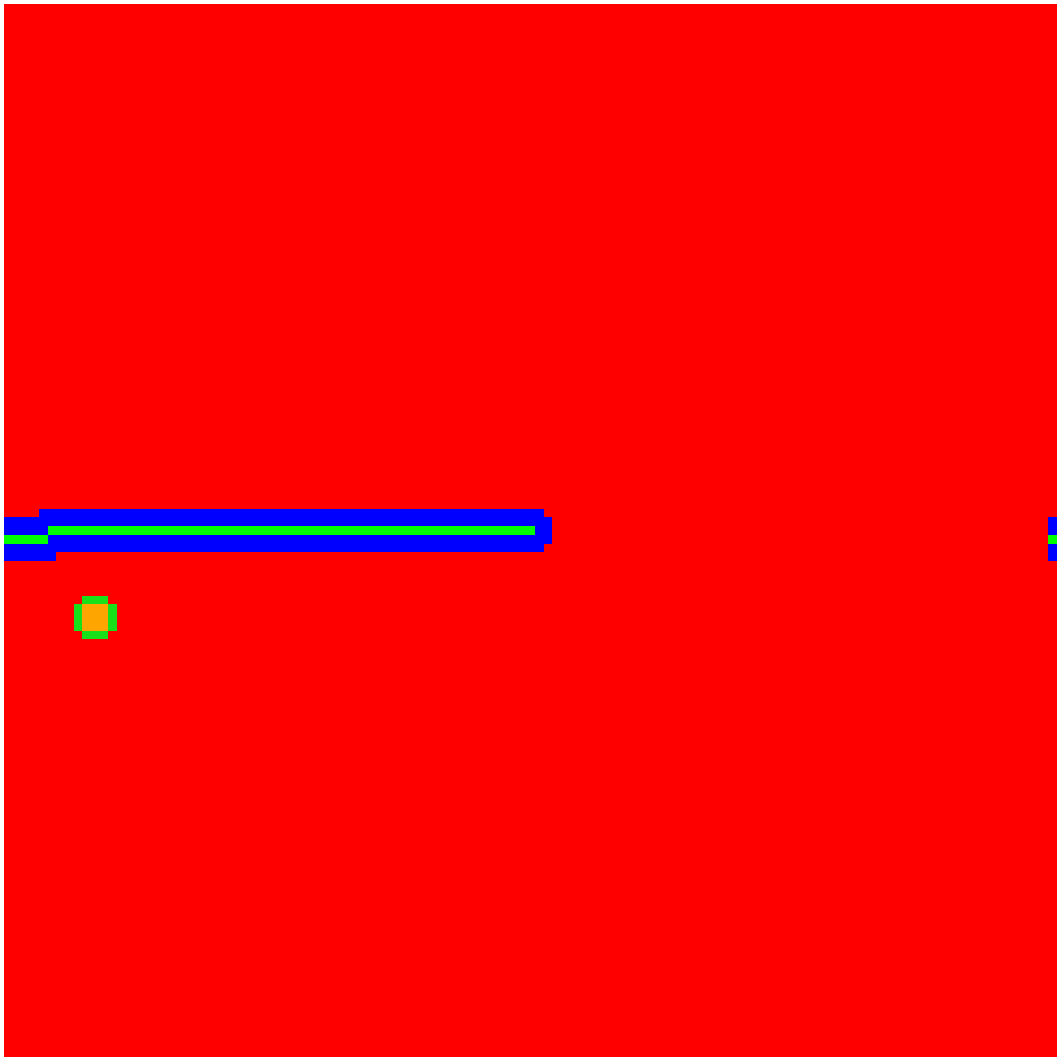

Figure 4-4. Motion planning algorithm input traversability grid.

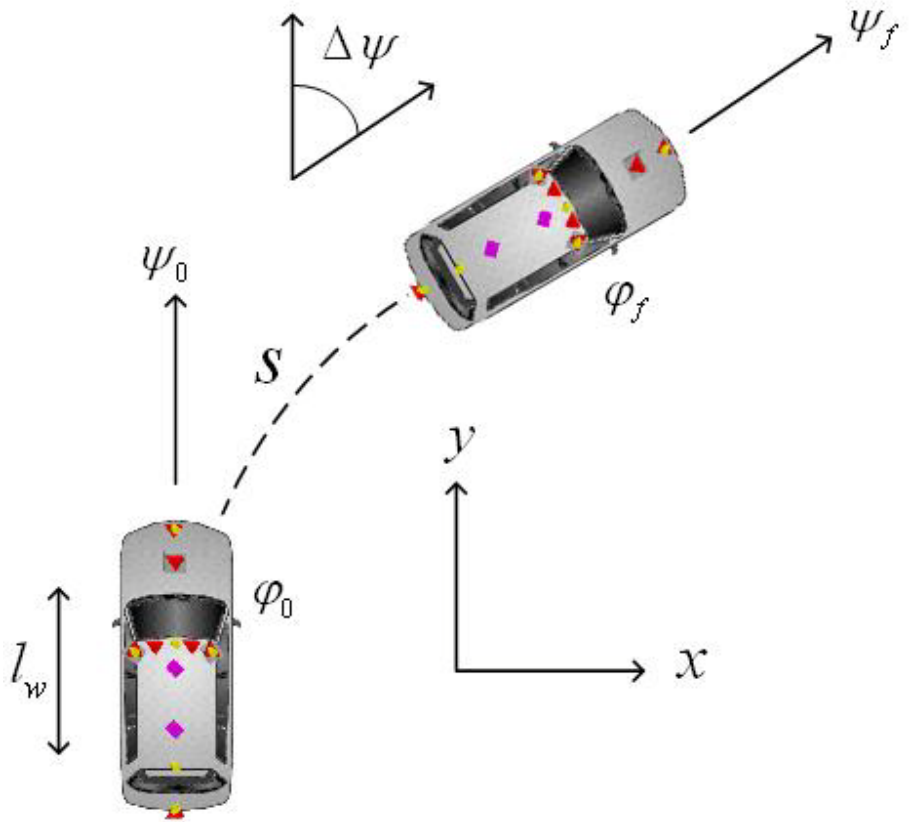

Figure 4-5. Vehicle kinematic model geometry.

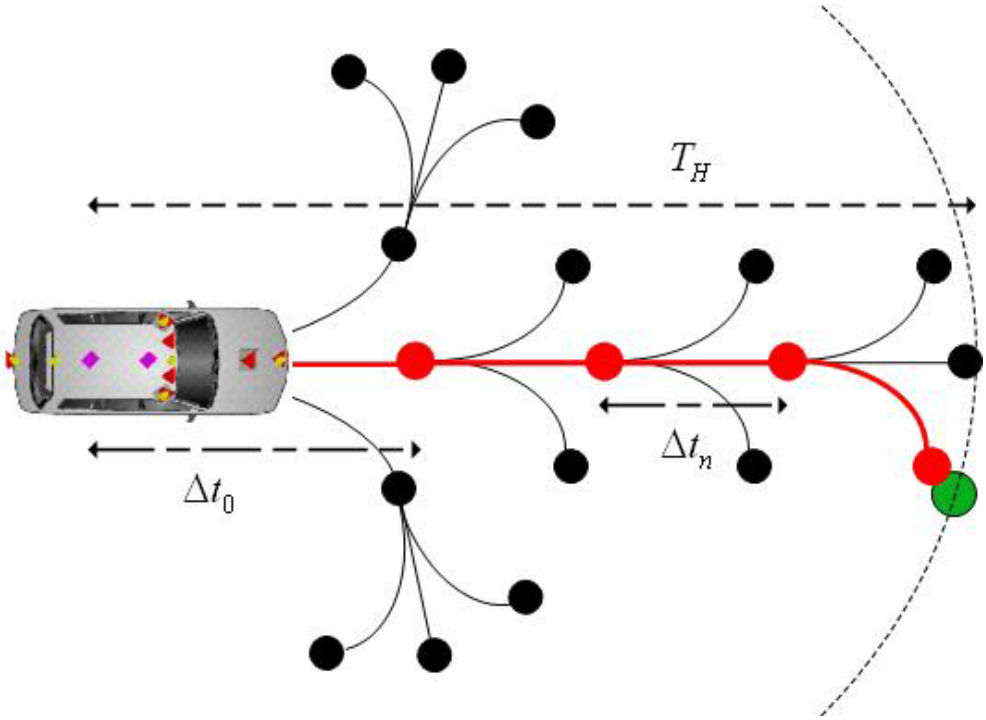

Figure 4-6. Illustration of search algorithm node expansion showing key search parameters.

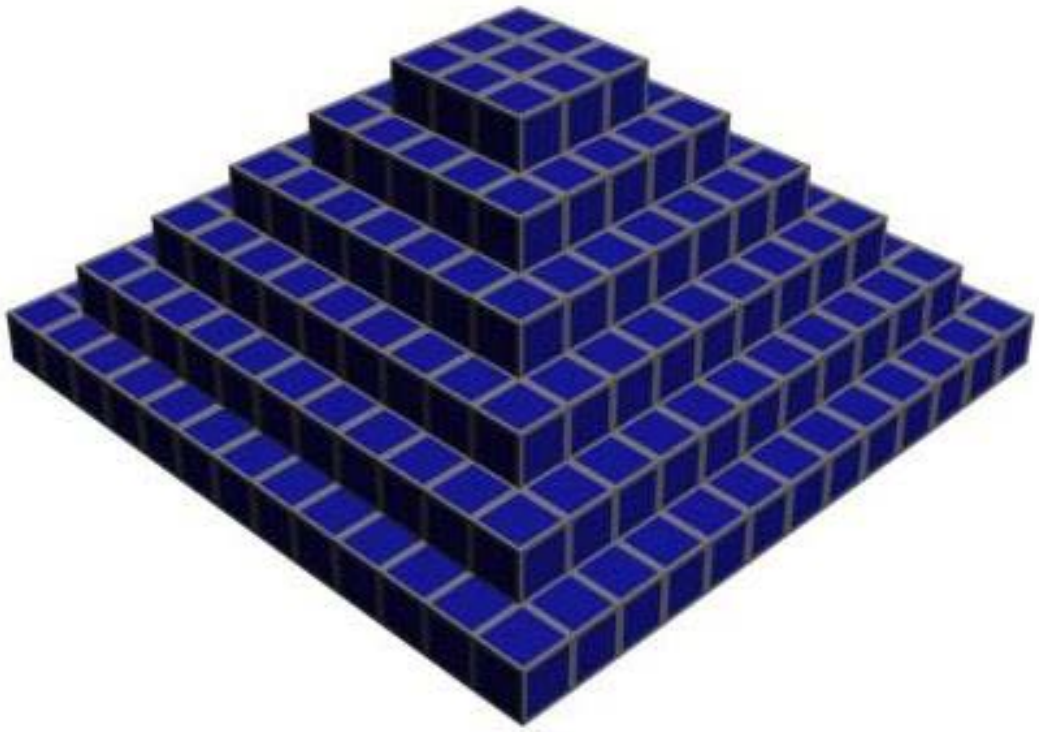

Figure 4-7. Pyramidal-shaped optimized temporal grid structure.

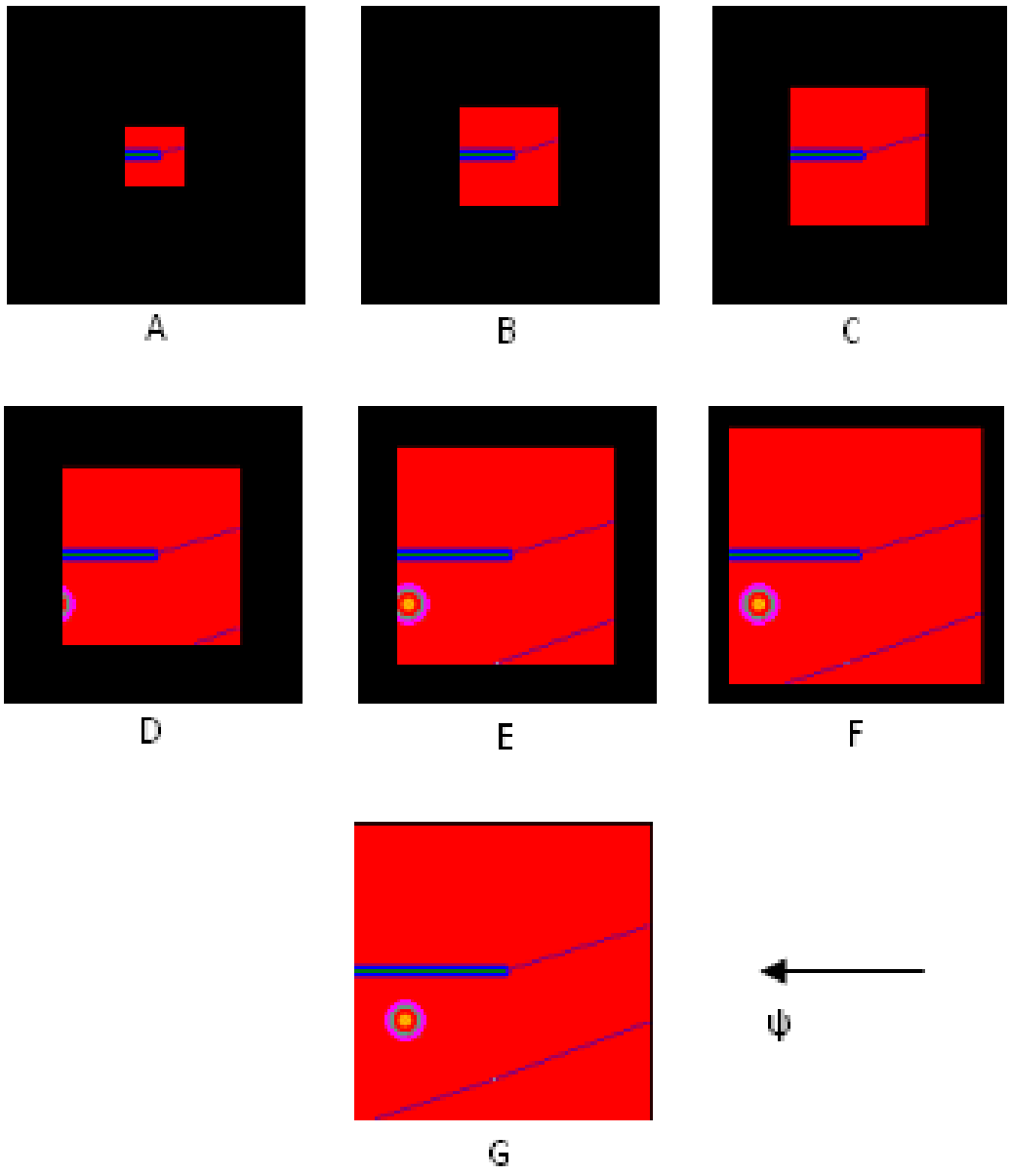

Figure 4-8. Optimized temporal grid layers showing static obstacle appearing.

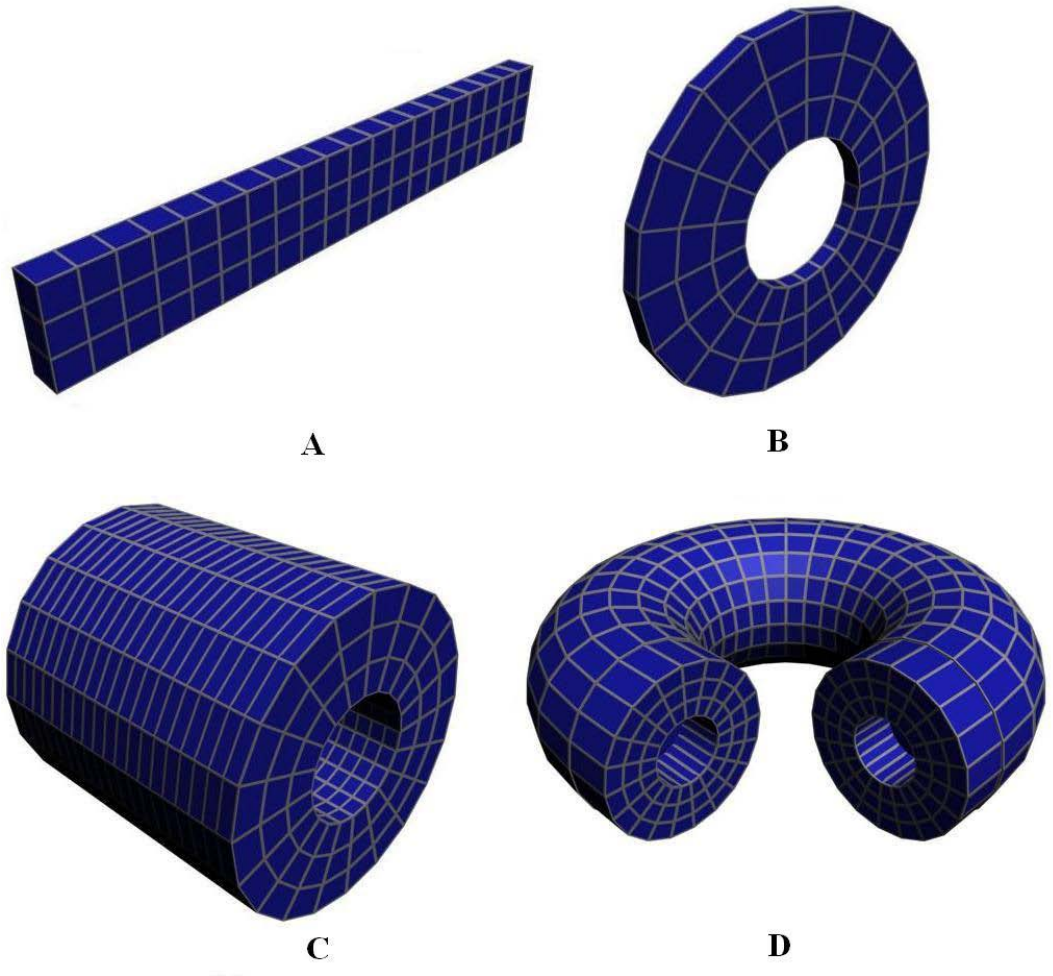

Figure 4-9. Formation of temporal torus buffer. A) Single temporal column of grid cells, B) temporal ring buffer, C) temporal ring buffer array, D) temporal torus buffer.

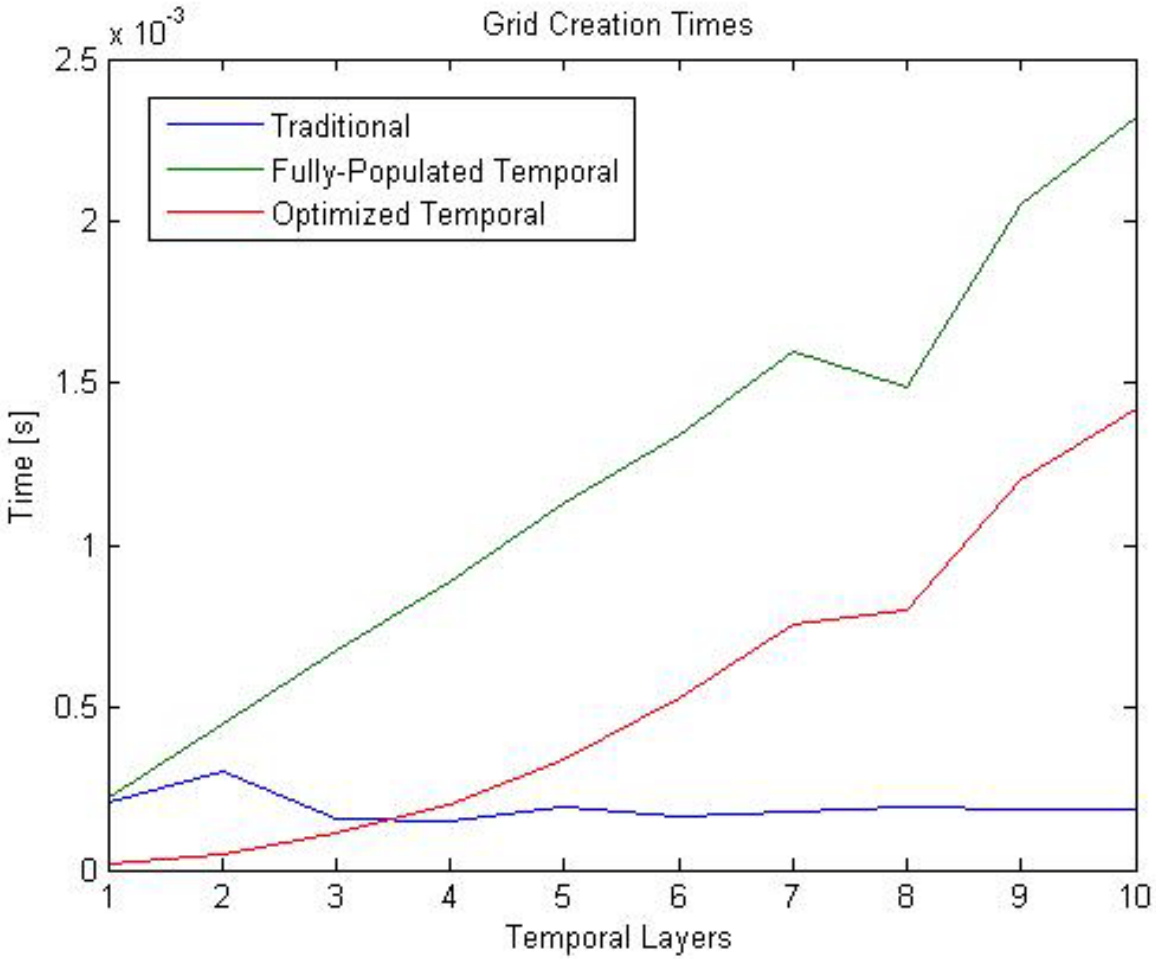

Figure 4-10. Creation times for TG, full temporal TG and optimized temporal TG.

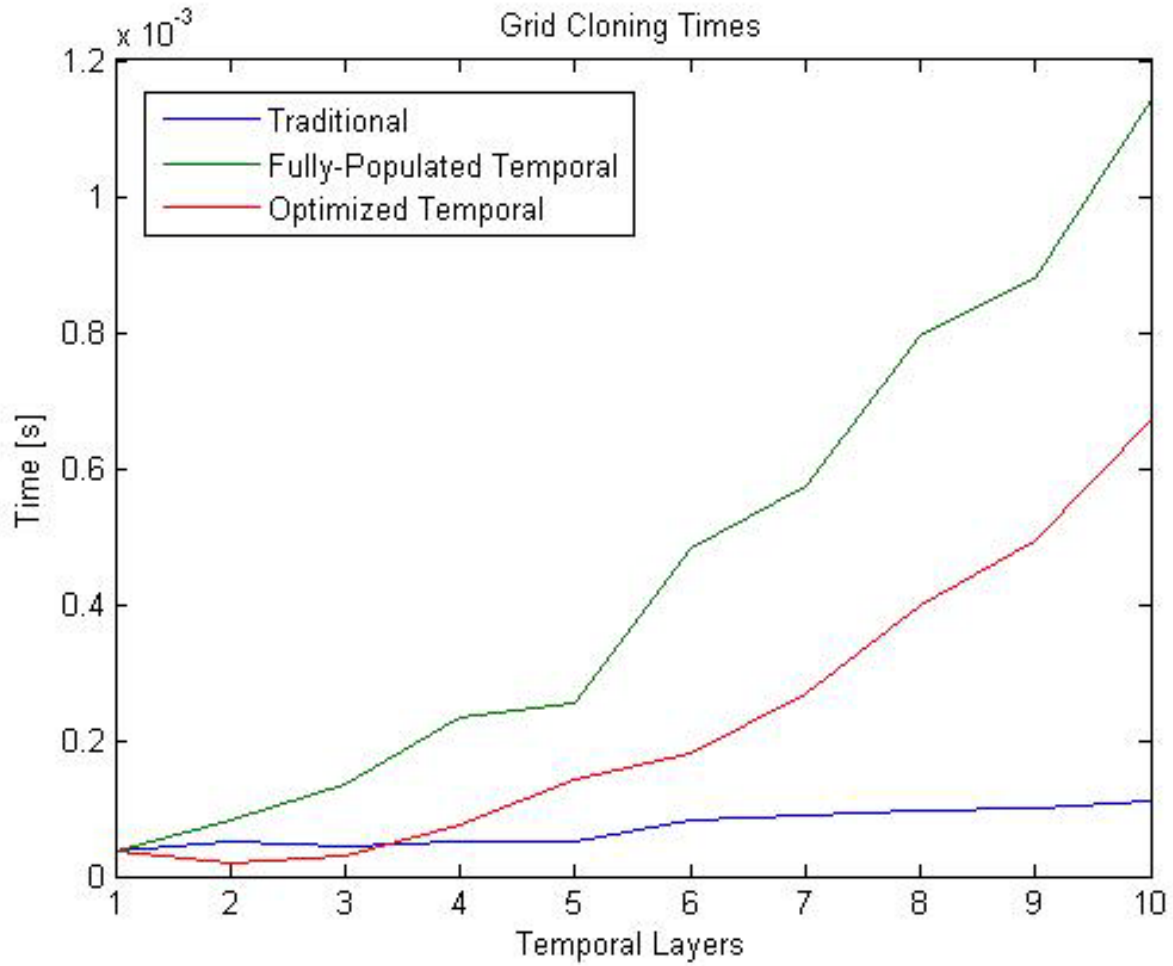

Figure 4-11. Cloning times for TG, full temporal TG and optimized temporal TG.

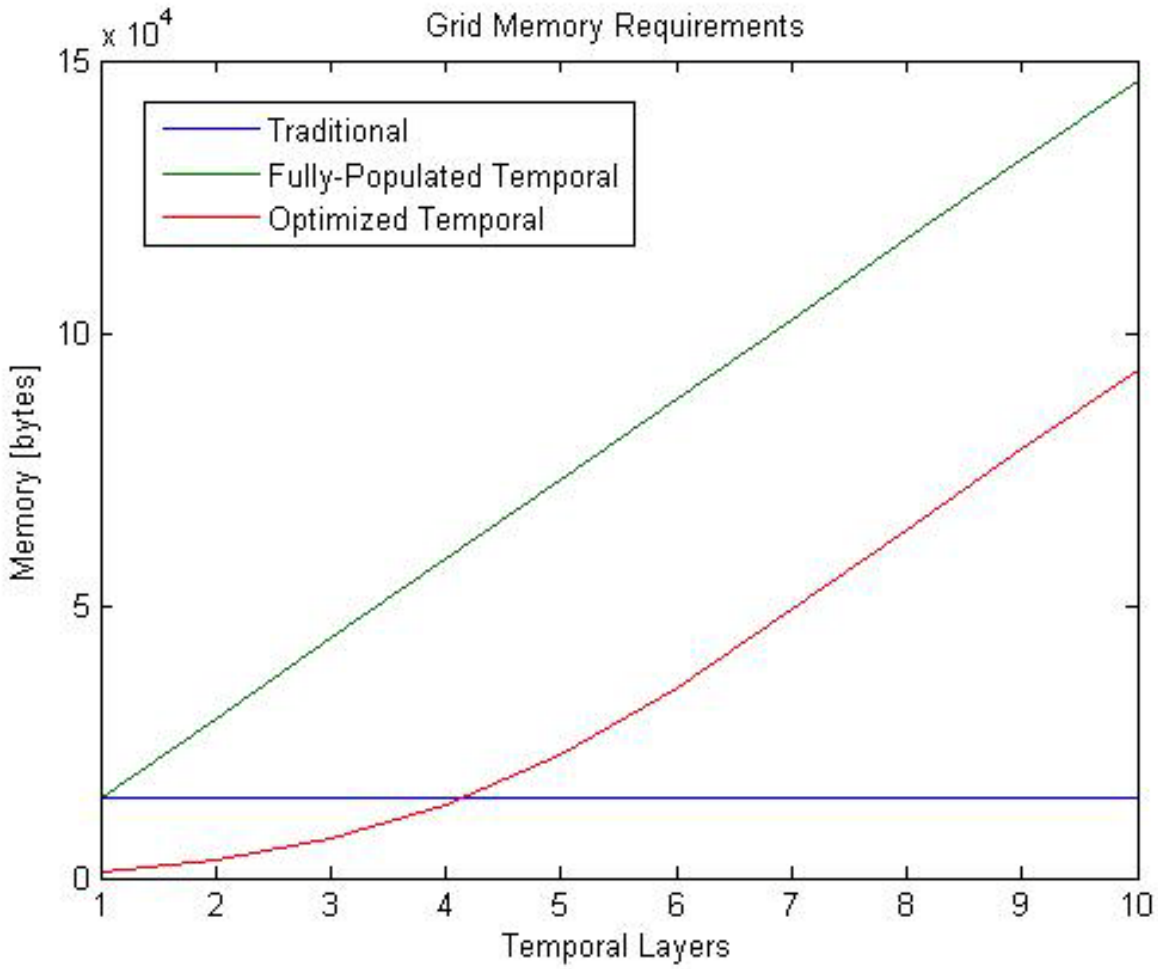

Figure 4-12. Memory requirements for TG, full temporal TG and optimized temporal TG.

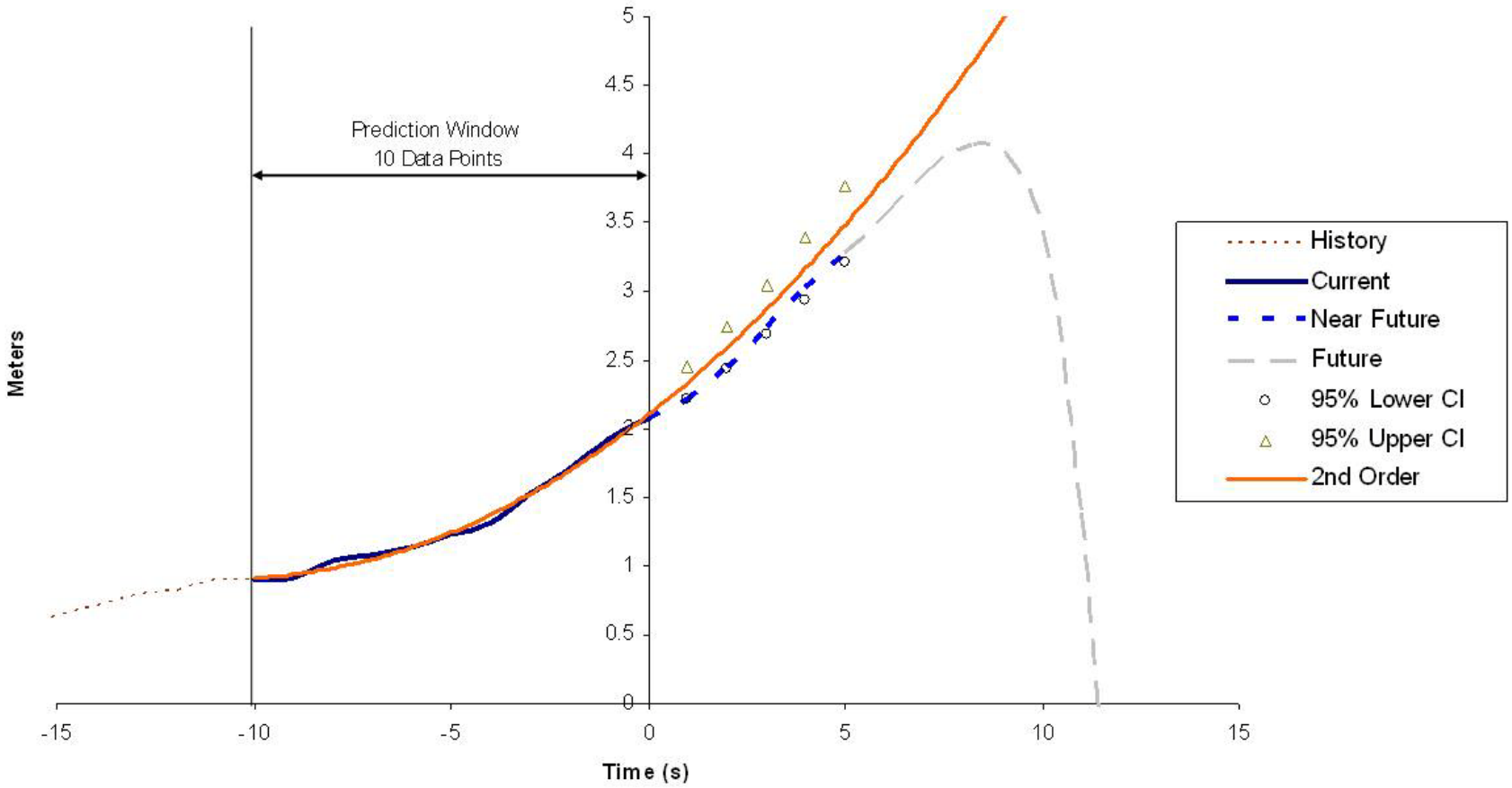

Figure 4-13. Prediction visualization showing polynomial curve fit and extrapolated data.

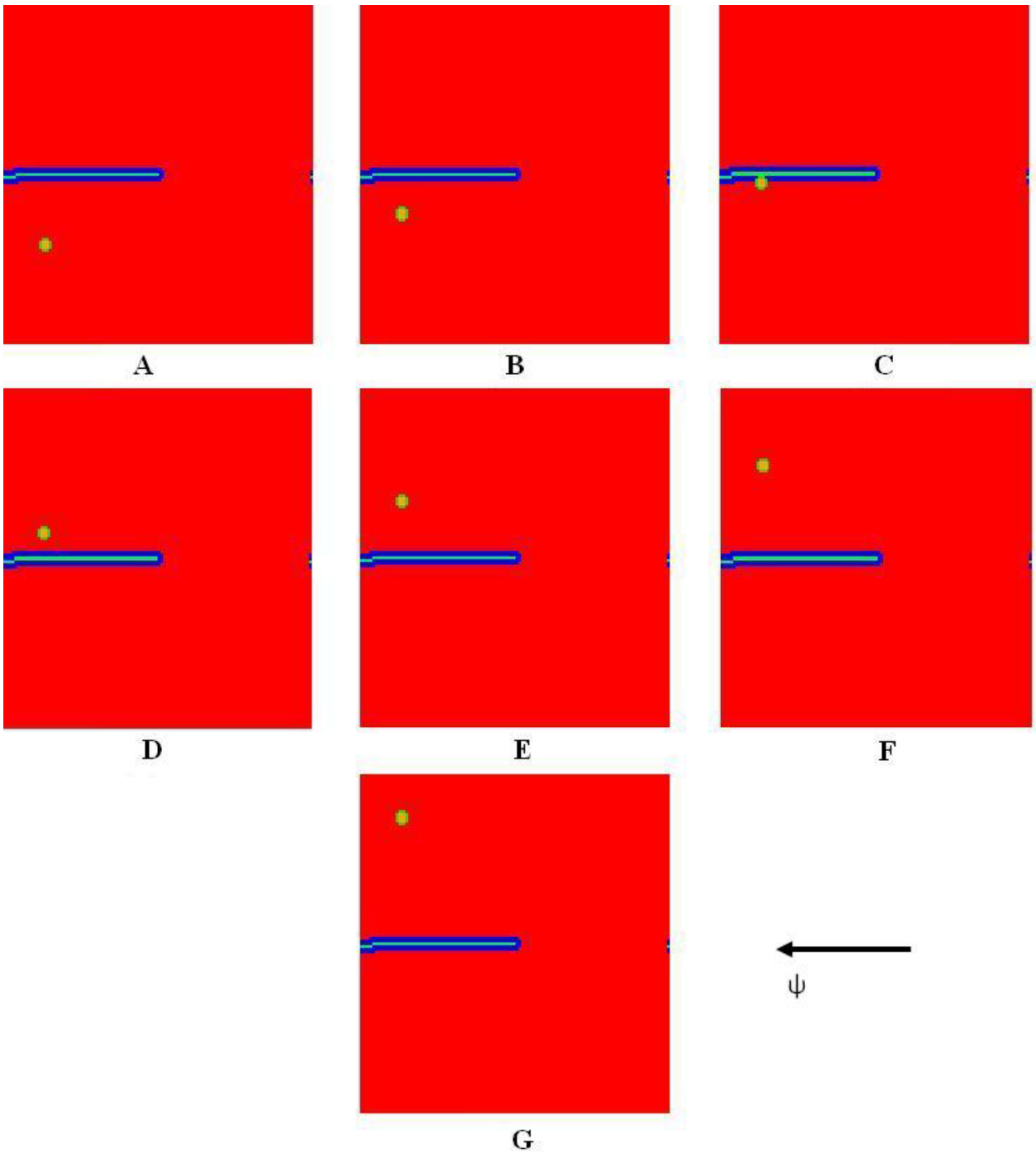

Figure 4-14. Full temporal grid layers showing predicted positions of obstacle.

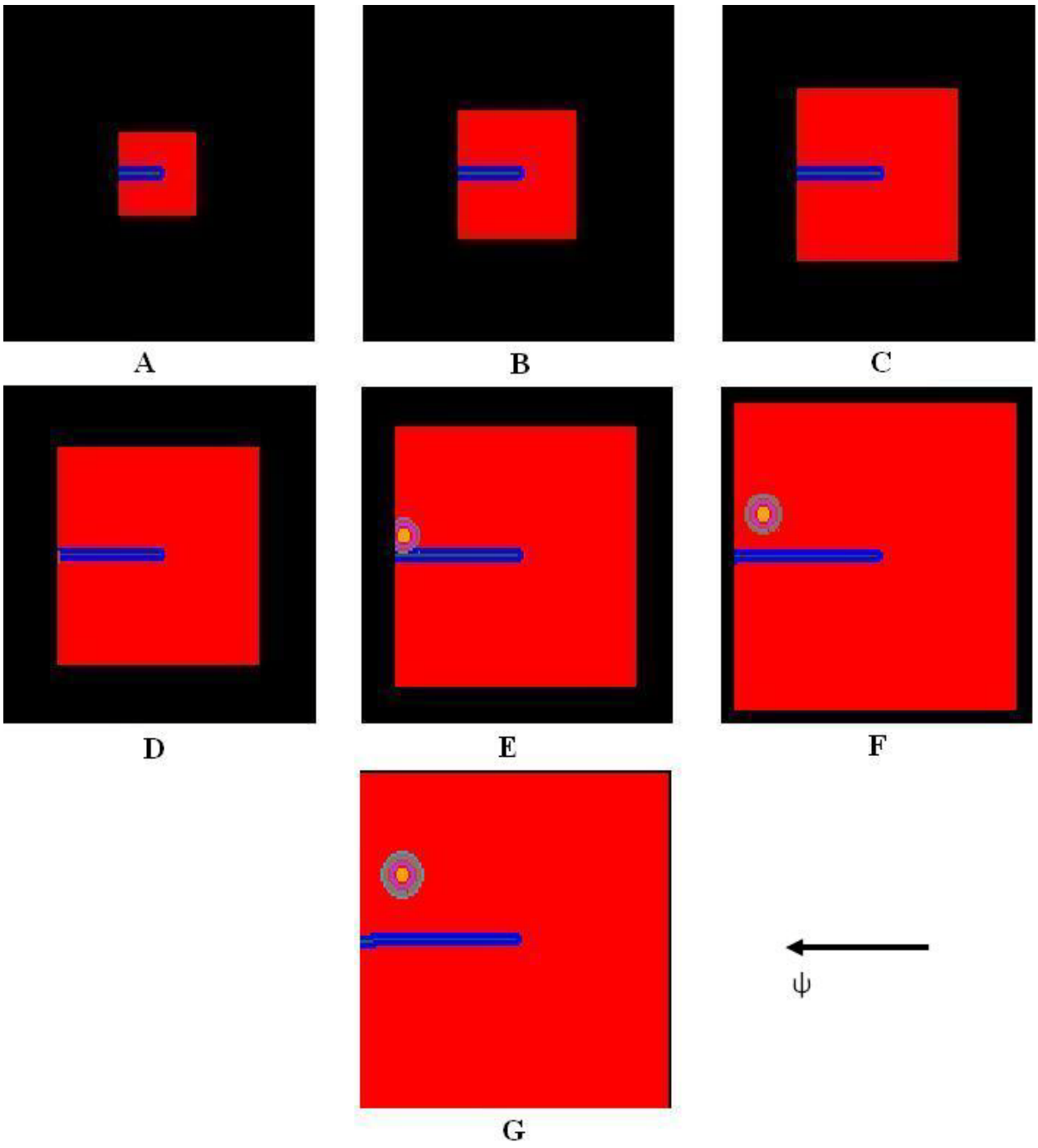

Figure 4-15. Optimized temporal grid layers showing predicted positions of obstacle.

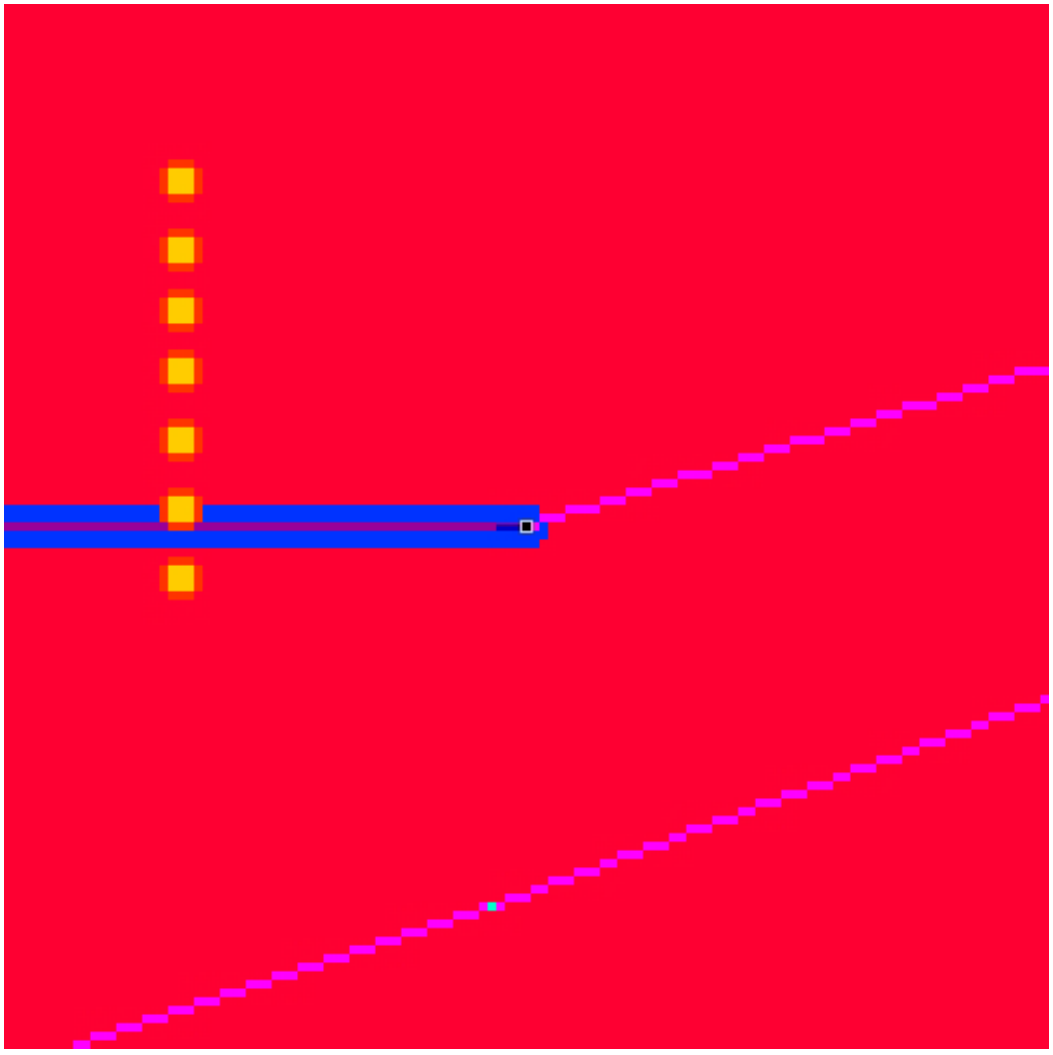

Figure 4-16. Future predicted positions of obstacle in single traversability grid.

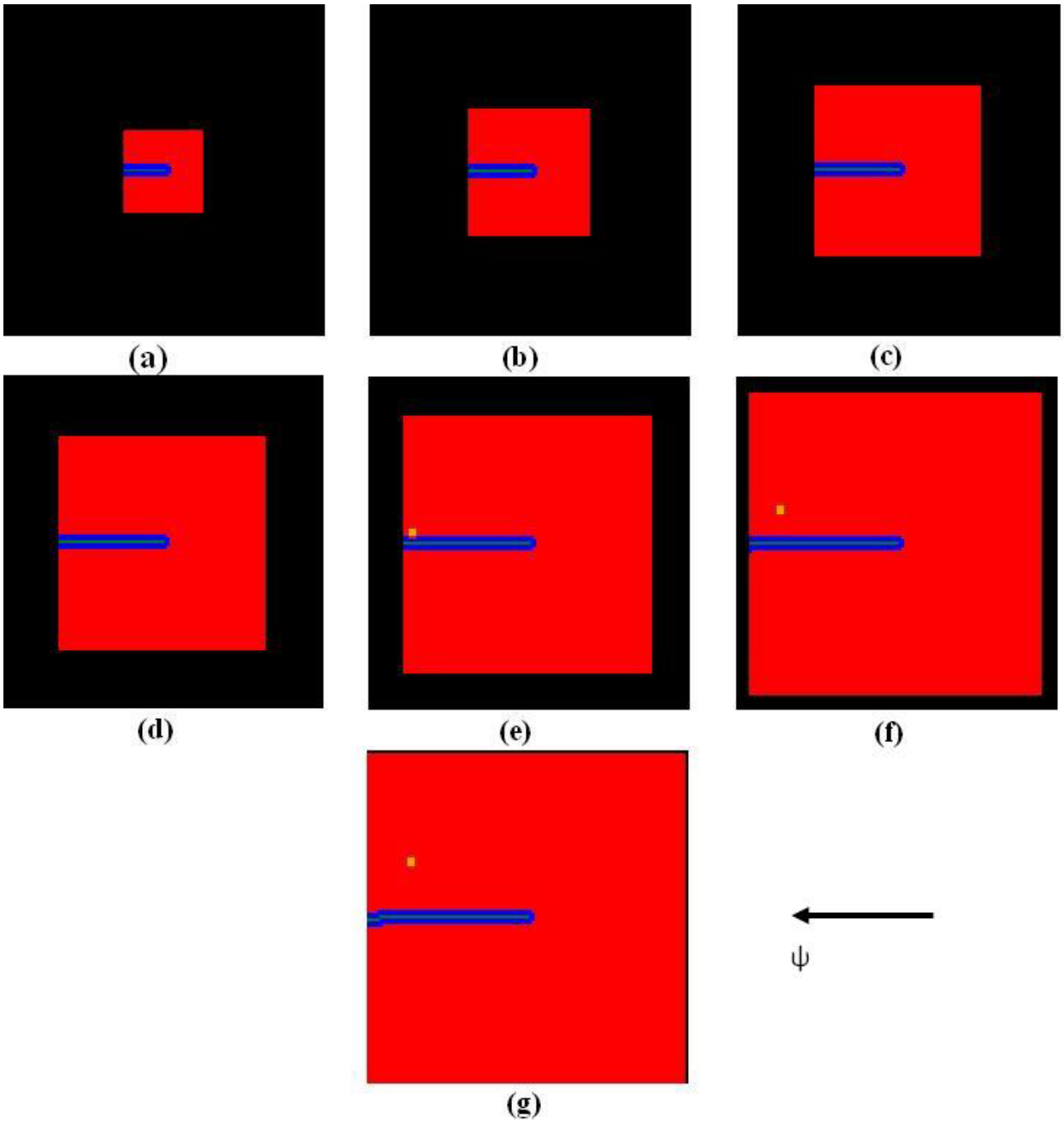

Figure 4-17. Optimized temporal grid showing simple dilation of predicted obstacle positions.

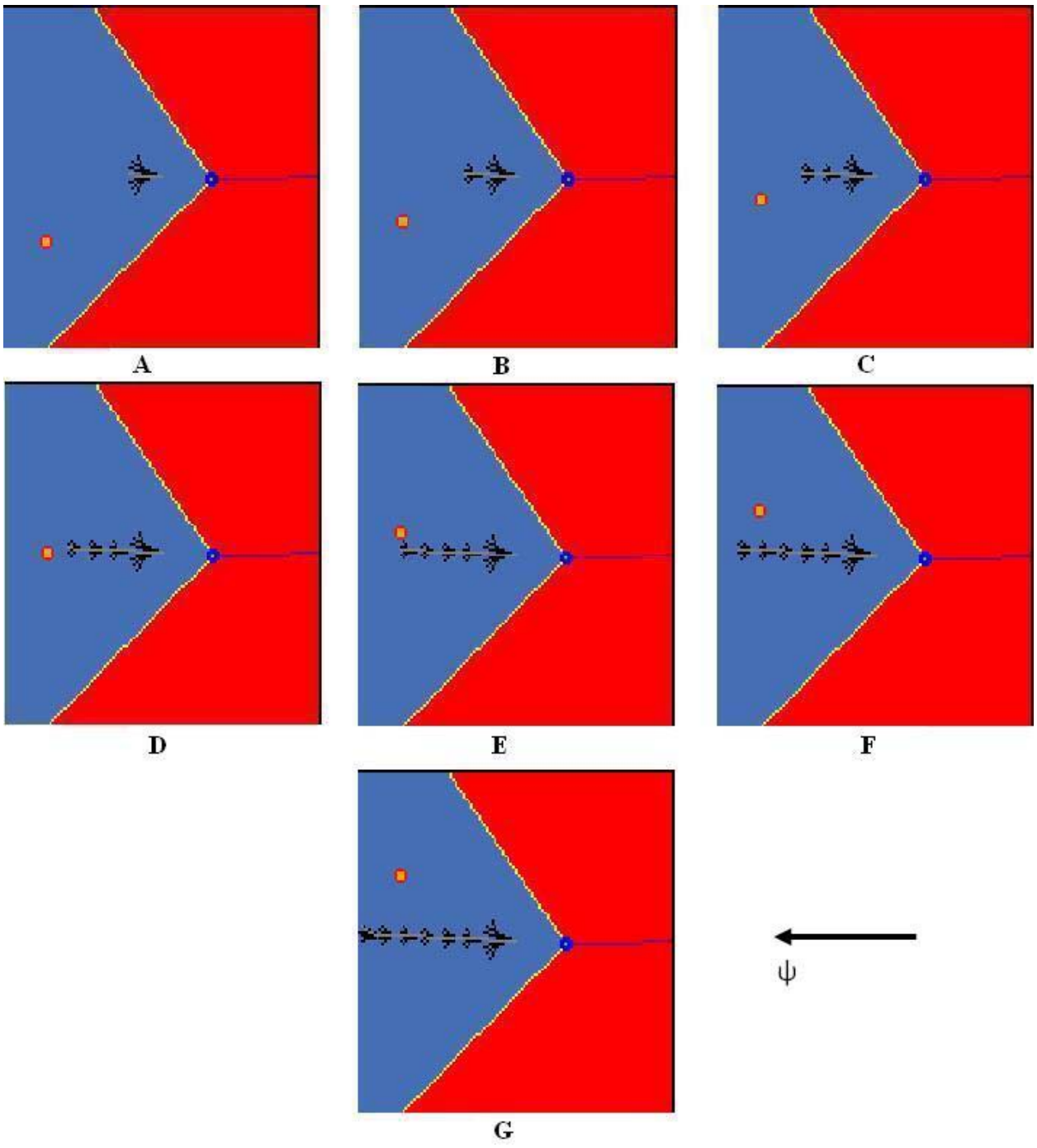

Figure 4-18. Full temporal grid layers showing expansion of search tree with obstacle.

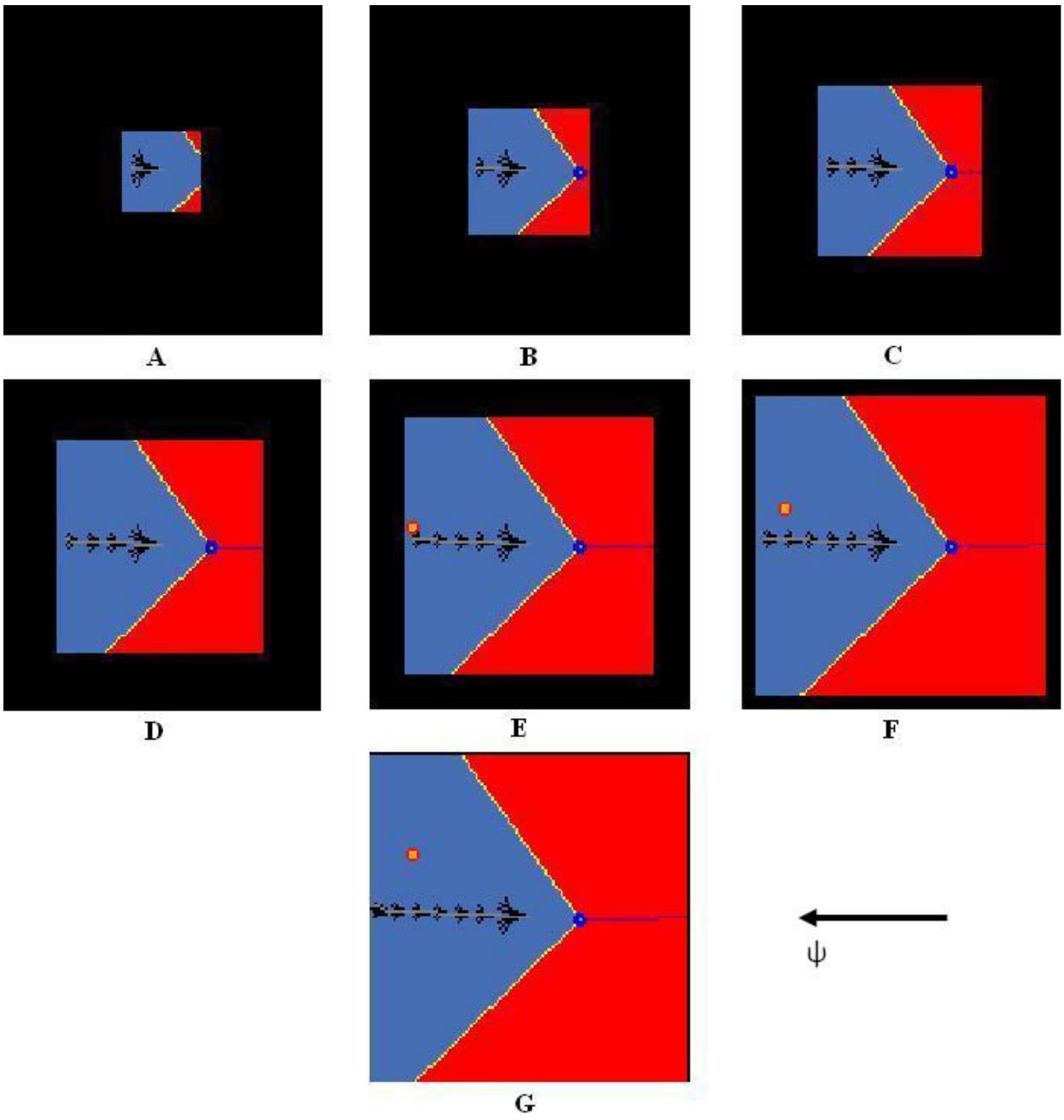

Figure 4-19. Optimized temporal grid layers showing expansion of search tree with obstacle.

Table 4-1. Temporal layer size parameters for $N_l = 1 \ldots 10$.

Temporal layer	Rows	Start row	End row	Columns	Start column	End column	Total cells
1	31	45	75	31	45	75	961
2	47	37	83	47	37	83	2209
3	63	29	91	63	29	91	3969
4	79	21	99	79	21	99	6241
5	95	13	107	95	13	107	9025
6	111	5	115	111	5	115	12321
7	121	0	120	121	0	120	14641
8	121	0	120	121	0	120	14641
9	121	0	120	121	0	120	14641
10	121	0	120	121	0	120	14641

Table 4-2. Distance horizon values for each temporal layer for initial temporal grid testing.

Temporal layer	1	2	3	4	5	6	7	8	9	10
Distance horizon [m]	6.0	10.0	14.0	18.0	22.0	26.0	30.0	34.0	38.0	42.0

Table 4-3. Grid creation times (in seconds) for $N_l = 1 \ldots 10$.

Temporal layer	Traversability grid creation times (s)	Fully-populated temporal traversability grid creation times (s)	Optimized temporal traversability grid creation times (s)
1	0.0002056	0.0002223	0.00001490
2	0.0002600	0.0004475	0.00004411
3	0.0001571	0.0006768	0.0001125
4	0.0001473	0.0008907	0.0002034
5	0.0001907	0.001131	0.0003427
6	0.0001635	0.001344	0.0005334
7	0.0001761	0.001598	0.0007579
8	0.0001933	0.001486	0.0008027
9	0.0001859	0.002052	0.001200
10	0.0001870	0.002322	0.001423

Table 4-4. Grid cloning times (in seconds) for $N_l = 1 \ldots 10$.

Temporal layer	Traversability grid cloning times (s)	Fully-populated temporal traversability grid cloning times (s)	Optimized Temporal traversability grid cloning times (s)
1	0.00003712	0.00003662	0.00003740
2	0.00004942	0.00008391	0.00001843
3	0.00004466	0.0001360	0.00002826
4	0.00004946	0.0002337	0.00007593
5	0.00005231	0.0002548	0.0001440
6	0.00008152	0.0004810	0.0001815
7	0.0009052	0.0005744	0.0002690
8	0.0009737	0.0007937	0.0003977
9	0.0001017	0.0008799	0.0004913
10	0.0001119	0.001141	0.0006715

Table 4-5. Memory requirements (in bytes) for $N_l = 1 \ldots 10$.

Temporal layer	Traversability grid memory usage (bytes)	Fully-populated temporal traversability grid memory usage (bytes)	Optimized temporal traversability grid memory usage (bytes)
1	14641	14641	961
2	14641	29282	3170
3	14641	43923	7139
4	14641	58564	13380
5	14641	73205	22405
6	14641	87846	34726
7	14641	102487	49367
8	14641	117128	64008
9	14641	131769	78649
10	14641	146410	93290

CHAPTER 5
TESTING AND RESULTS

The temporal motion planning method outlined in this document was implemented and tested extensively. Because of the complexity of maintaining a functional autonomous vehicle and the considerable resources in terms of manpower and facilities required to execute autonomous testing, comprehensive testing was first conducted in simulation to verify the performance of the various developed algorithms. While the results of these simulations were promising, live robotic testing was also necessary to practically confirm the validity of using a predictive temporal motion planning method for autonomous ground vehicle navigation.

This chapter outlines the various testing methods and metrics used to measure the performance of this method. The first section outlines the test plan for data collection and analysis of results to ensure effective investigation of the presented research. Next, the simulation procedure and results are described, followed by the same for live robotic testing. Finally, a summary of the test results is provided, which draws a few preliminary conclusions.

Test Plan

It was determined that the new PTMP method could be applied to several of the operating behaviors encountered during the DUC and to the target interception problem presented. A few of these behaviors included an unmanned ground vehicle attempting to follow a moving object in its desired lane of travel, navigating an intersection with oncoming vehicles present, passing a slow-moving vehicle in its desired lane and navigating an unstructured obstacle field. A subset of these behaviors was selected for thorough testing to validate the newly-developed planning method.

The first step was to conduct a series of baseline and control tests to collect data by which to compare the results of the PTMP tests against. The baseline tests involve running a motion

planning algorithm similar to that described in (Galluzzo, 2006) in situations matching those to be described shortly. However, no moving obstacles were present so that the pure, undisturbed motion planning outputs was measured. This provided the ideal test results in terms of the desired testing metrics which are outlined in a later section. The control tests introduced moving obstacles in the same scenarios to observe and quantify how the control outputs of the motion planning algorithm were affected by these objects. Finally, the same tests were conducted using the new PTMP algorithm to investigate its performance with moving obstacles present.

Figure 5-1 shows an aerial view of the test site at the Gainesville Raceway, which has served as the test facility for much of CIMAR's autonomous ground vehicle projects. The site provides paved road segments used in testing many of the required behaviors for the DUC, including straight and curved road segments with multiple lanes, intersections, dead-end streets and a parking area. Sections of this area were selected for simulating the required test environments. Figure 5-2 shows a close-up visualization of the waypoints defining the road segment used for the following behavior tests and Figure 5-3 shows the same for the waypoints and perimeter points defining the open, unstructured area used for the obstacle field behavior and target interception tests.

Following Behavior

The first behavior selected involved the robot maintaining its desired lane of travel while following another moving object in front of it. The following operating behavior is a crucial ability exhibited by an unmanned vehicle that may interact with other moving vehicles in an urban environment. Figure 5-4 illustrates the effect an obstacle situated in the vehicle's desired lane had on the motion planning algorithm's attempt to plan a path down the center of the lane as recorded during a control test of the following behavior. The presence of the obstacle, painted in yellow, forced the planner to attempt to find a path that took the vehicle out of the desired lane.

The wide region of expanded search nodes illustrates the difficulty of the attempt, and the light yellow line signifies the failed solution path.

Another problem which arose from this situation is illustrated as the solution path oscillates from the left side of the obstacle in Figures 5-4(a) and (b) to the right side of the obstacle in Figures 5-4(c) and (d). This oscillation could lead the vehicle to reach an unstable state that could ultimately lead to a collision with a vehicle in the adjacent lane or an object to the side of the desired lane of travel. This unacceptable behavior resulted from the moving object being treated as static while the planning algorithm expanded its search tree.

In the following behavior test, a similar situation was constructed to test the effectiveness of the developed predictive temporal motion planner when faced with the same scenario. The vehicle was placed in its desired lane behind another vehicle travelling in the same direction in a straight line. The lead obstacle vehicle was instructed to progress with a velocity that matched that of the robot such that the separation distance remained approximately constant. The use of obstacle motion prediction by the new temporal planning method should have allowed the motion planning algorithm to generate its control sequence smoothly as the object was projected further along the desired lane of travel as the planning time progressed. Table 5-1 outlines the following behavior test plan, including the purpose and description, expected results, and lists the types of data that were recorded for analysis.

Obstacle Field Behavior

The previously described following behavior test-case simulation took place on a structured road with a desired lane of travel and with the followed obstacle maintaining the same lane in a straight line. A less defined situation was chosen for the next round of testing. The images in Figure 5-5 show the robot attempting to plan a path in an open, unstructured environment around an obstacle that did not necessarily follow a prescribed path. The robot

itself was free to traverse any part of the open area to reach its goal. It is evident in the various traversability grids shown that the motion planner was having difficulty finding a consistent solution as the selected path oscillates in front of the obstacle in Figures 5-5(a) and (b) and behind the passing obstacle in Figures 5-5(c) and (d).

The test outlined for this situation sought to analyze the pure obstacle avoidance capabilities of the presented motion planning method. The robot was placed in one of these open, unstructured areas and commanded to traverse straight across to the opposite side. An obstacle vehicle was commanded to follow a path that crossed the straight-line path connecting the robot with its goal in a perpendicular manner at an intermediate distance. The velocity of the obstacle vehicle was set such that it passed the robot and was out of its way by the time the robot reached that position. Table 5-2 outlines the obstacle field behavior test plan, and contains the same sections as the following behavior test plan.

Target Interception

The final series of tests considers the target interception application of the PTMP method. These tests were similar to those of the obstacle field behavior situation with an object crossing perpendicularly to the current vehicle heading in the same open, unstructured environment. However, in this instance, the object was used to calculate the goal point that the robot was seeking to achieve rather than being treated as an obstacle that the robot needed to avoid. In this scenario, the motion planner directly received the prediction model of the object and used the final predicted position to calculate its goal point, toward which it attempted to plan a path. The motivation for this application came from the research being conducted by AFRL for perimeter surveillance and response. The ability of unmanned ground robots that are patrolling a secure perimeter to autonomously detect and intercept an intruder could provide useful deterrence and

allow for more time for a more forceful and appropriate response to be coordinated. Table 5-3 outlines this final test plan.

Test Metrics

A number of different performance measures were selected to be recorded to evaluate the new PTMP method when compared to existing motion planning techniques. These metrics served to investigate the effectiveness of the new method in allowing a robotic system to navigate in dynamic environments. The first set of metrics was associated with how efficient and effective the planning method was at finding a solution path. The first metric selected involved the success versus failure rate of the motion planning algorithm in generating trajectories that allowed the system to achieve its goal. It was found that obstacles crossing in the path of the vehicle could have caused a failure of the search algorithm using the traditional motion planning method, whereas the temporal motion planning method could have addressed this problem by projecting the positions of the obstacle forward with time. The number of search nodes expanded was also an important measure of efficiency of the motion planning algorithm. It was shown that moving obstacles could affect the search algorithm negatively and result in more of the search space being explored to find ways around the obstacle, while it was believed that the predictive temporal method would be able to use the minimum number of nodes since it was obvious that the obstacle would be out of the vehicle's path.

The cost of the entire solution path was a good comparative parameter which captured the optimality of the various methods. This cost again was based on the traversability values of the cells through which the solution trajectory pass and the distance from a particular node to the goal state, and was calculated according to Equations 4-14, 4-15, and 4-16 with simple cost units. For the following behavior test, because the center of the lane was painted with the lowest cost traversability, this meant that the lower the total path cost, the closer the solution trajectory

tracked the center of the lane. It was believed that this parameter would be less useful during the obstacle field behavior testing because all cells within the open, unstructured environment were assigned the same traversability values; however, the cost associated with the distance to the goal would still be applicable and could still have been used as a measure of performance.

The second set of metrics dealt with the solution path itself and the affect of the solution path on the motion of the vehicle. The deviation of the solution path was a measure of how the path diverged from a nominal straight-line path to the goal point. The sum of the heading angle changes instituted by each individual state change in the solution path was calculated. This provided evidence of the motion planner's attempts to divert the path around the obstacle versus simply planning toward the goal as if the obstacle was not present. Likewise, how well the solution path tracks the goal point was a good measure of the how the motion planning algorithm performed. Therefore, the heading angles from the vehicle to the goal state and from the vehicle to the final node on the solution path were calculated and compared.

The final metric involved the commanded steering effort, which was determined by the solution path. Ideally, the PTMP method would generate a solution path that minimized the motion of the vehicle when driving in a straight-line behind an obstacle and planning toward a goal point directly in front of the vehicle as an obstacle that it would not collide with passed in front of its straight-line path. However, any interference resulting from the obstacle would present itself as the commanded steering effort increased or oscillated.

All of these values were logged in a separate computational thread by the motion planning algorithm so that it did not interfere with the performance of the rest of the algorithm. Data was recorded, plotted and analyzed to determine the results of each of the tests. A last, visual metric was used to gauge the performance of the new method. Traditional and temporal traversability

grid images were recorded for all of the tests conducted with the output of the temporal motion planner displayed to provide visual evidence of how well the algorithm dealt with the various situations with the moving obstacles.

Simulation

To evaluate the new PTMP method before actually using it to autonomously drive the Urban NaviGator, extensive simulation was carried out. These simulations were executed for each test scenario described above using all of the key software components required for autonomous navigation. Most of these components were run exactly as they would be during autonomous navigation; however, a few simulators were run in place of other key components. A Global Position and Orientation Sensor (GPOS) simulator was run to reproduce position and orientation information and to facilitate vehicle motion during the simulations. A Primitive Driver (PD) simulator was run to allow the simulated vehicle to be placed in and taken out of an autonomous mode and to allow for simulated gear shifting. Finally, a simulated MO sensor was implemented to provide fabricated obstacle position and motion data. The various motion planning algorithms used for comparison and the new PTMP method were run in their natural state. Now each of the simulation scenarios is described in detail, along with the results.

Following Behavior

The simulation environment for the testing of the following operating behavior was created by generating a sequence of waypoints in a straight line for approximately 150 m to act as the desired lane of travel for the robot. The simulated vehicle was placed at the beginning of this straight road segment with an obstacle vehicle in front of it in the same desired lane of travel. This obstacle was defined by a single position in the lane and dilated by approximately the width of the vehicle in every direction to account for the simulated vehicle being treated as a single

point. This position and the trajectory and velocity of the obstacle were simulated in MO and reported to the TGC component.

The obstacle was initially located approximately 18 m in front of the vehicle. At this point the vehicle was put into a simulated autonomous state and the motion planner began generating its trajectory. The test began as the position simulator began moving the vehicle in a straight line between each of the waypoints that defined the testing road segment. As the vehicle began exhibiting motion, the simulated MO sensor began artificially moving the followed obstacle in a similar straight line between each of the waypoints. The vehicle velocity was set at a constant 4.47 mps (10 mph) and the velocity of the obstacle was set to match. This resulted in the followed obstacle remaining a constant distance in front of the vehicle as they progressed down the road segment.

Figure 5-6 shows a single fully-populated temporal grid produced by the TGC that shows the various temporal layers, and the associated predicted positions of the followed obstacle. Figure 5-7 then shows the temporal layers of the equivalent optimized temporal grid that was reported to the motion planning component. These images show that, for each successive temporal layer, the predicted position of the followed obstacle was outside of the motion planning algorithm's visible area. As this test was being conducted, both the TGC component and the motion planning algorithm logged data pertaining to the relevant testing metrics described in the earlier section. Table 5-4 shows a sample of predicted obstacle positions in both the local (vehicle) and global (UTM) coordinate systems and velocities for a following behavior test. A discrepancy is noticed in the desired starting X position of the obstacle (-18 m) and the initial predicted position of the obstacle (-12.96 m). This was likely caused by the asynchronous

nature of the desired rate of collection of data points with allowable provision of the data by the MO sensor which was then corrected.

The three separate tests outlined previously were conducted for this scenario to allow for comparison and contrast. The obvious measure of the success of the new PTMP algorithm was its ability to successfully generate a solution trajectory. The original algorithm used during the following behavior control test had a failure rate of approximately 14% as the followed obstacle presented problems while attempting to generate a trajectory down the center of the desired lane of travel. Even considering the successful attempt during this test, the solution trajectories would have effectively driven the robot out of the desired lane and possibly into an oncoming vehicle or other object. Conversely, the PTMP algorithm experienced a 100% success rate in finding a solution trajectory. Figure 5-8 shows the temporal layers of an output grid of the new motion planning algorithm showing each step of the search node expansion out to the time horizon in Figure 5-8(g).

The next metric of note was the number of search nodes expanded as the vehicle moved down the desired lane of travel. Figure 5-9 shows a plot of the number of search nodes used during the three tests over a period of time. The original motion planning algorithm expanded an average of 46 nodes while planning a trajectory down the center of the desired lane of travel, with a maximum of 334 nodes expanded during the baseline test with no followed object. The control test resulted in an average of 445 nodes being expanded with a maximum of 3,401 nodes. As would be expected, the original motion planning algorithm required many more nodes to be expanded while attempting to generate its trajectory around the obstacle, again referring to Figure 5-4. Finally, the new PTMP algorithm required an average of 203 search nodes to be expanded, with a maximum of 564 nodes expanded. While higher than that of the baseline test,

the number of nodes expanded using the PTMP was about half that required by the original algorithm used during the control test.

Another of the basic metrics used to evaluate the new motion planning algorithm was the solution path cost. Figure 5-10 displays a plot of these costs for the three separate tests conducted. The average cost for the original motion planner in the baseline test was approximately 1.778e15 cost units while that of the same planner during the control test was approximately 4.113e17 cost units, or over two orders of magnitude more costly. This result was to be expected as the solution trajectories generated by the motion planner during the control test were attempting to navigate around the followed obstacle and, therefore, resulted in the expanded search nodes falling in areas of the TG considered out of bounds with low traversability values and, thus, very high associated costs. Meanwhile the average cost of the solution trajectory generated by the new PTMP algorithm was approximately 1.892e15 cost units. This average cost using the temporal method was only approximately 6.4% larger than that of the original method from the baseline test, exemplifying that the temporal method was able to find trajectories that closely followed the low cost cells representing the center of the desired lane of travel.

The last metric to be analyzed was the final steering command coming from the motion planning algorithm and implemented as a control input to the robot. This is the most telling metric as it described how the vehicle will actually drove when confronted with an obstacle in front of it. An almost negligible average steering command magnitude of approximately 0.5% was achieved for the baseline test and depicts the original motion planning algorithm as nearly always finding a solution trajectory directly down the center of the lane with no obstacle present. On the other hand, the introduction of a followed obstacle in the desired lane of travel resulted in

an average magnitude of approximately 11.4% for the control test. This agrees with the other metrics and the images of the motion planner being forced to generate a solution trajectory that diverted out of the desired lane and into regions of the grid considered out of bounds in an attempt to navigate around the followed obstacle. However, the new temporal motion planner was able to generate solution trajectories with an average steering command magnitude of only 2.8%, achieving an approximately 75.4% reduction in average steering effort when compared to the control test. Figure 5-11 shows the recorded steering commands for all three following behavior tests.

Obstacle Field Behavior

The purpose of the obstacle field behavior simulation was to evaluate the pure obstacle avoidance capabilities of the new PTMP method in an open, unstructured environment. The environment was set up as a single inlet road segment, a large open area defined by a series of perimeter points, and a single outlet road segment. The open area zone was approximately 60 m long by 40 m wide and contained no defined lanes, intersections or constructs which could have limited the motion planning algorithm in its attempt to build its trajectory to the goal point. The simulated vehicle was initially situated just inside the entrance to the open area in an orientation that allowed it to move directly down the center of the zone in a straight line. The lone obstacle was again represented by the simulated MO sensor as a single position, but was then dilated in all directions by a length equivalent to the width of the robot plus a small buffer.

The obstacle was initially located approximately 15 m in front of the vehicle and 15 m to its left side. As with the following behavior tests, the vehicle was placed in a simulated autonomous state at which point the motion planning algorithm began generating its solution trajectory. The position simulator was then activated to begin moving the vehicle in a straight line down the center of the open area at approximately 2.235 mps (5 mph), while attempting to

plan toward a goal point approximately 30 m directly in front of it. At this same instant, the simulated MO sensor began updating the obstacles position in a way that led it to cross perpendicularly to the path followed by the robot at approximately 4.470 mps (10 mph).

The initial positions of the robot and obstacle and the assigned velocities were selected to create a scenario where the obstacle passed across the vehicle's desired path, but was well out of the path by the time the vehicle reached that location on its path. Figures 5-12 and 5-13 show the fully populated and optimized layers of a single temporal grid, respectively, and show the predicted positions of the obstacle crossing the desired path of the robot. These images serve to show that the obstacle was out of the way by the time the vehicle reached the crossing position, thus providing the expectation that the motion planning algorithm did not need to alter the straight-line trajectory of the vehicle. The same temporal motion planning data was recorded to verify this claim. Table 5-5 provides a sampling of the predicted positions and velocities of the moving obstacle for an instance of the obstacle field test.

The open area obstacle field behavior simulation was a bit more difficult to analyze in terms of the performance metrics used for the following behavior due to a number of factors. The solution path cost became less relevant because the traversability values all grid cells within the open area zone were equal to emphasize that the vehicle could drive anywhere within the perimeter of the zone. This resulted in the distance aspect of the search node cost playing a more important role in the cost analysis. Likewise, because of this uniform traversability, the number of search nodes varied little from one test to the next. Also, the passing of the obstacle in front of the vehicle was not a big enough disturbance to cause the motion planning algorithm to fail in finding a solution trajectory. Therefore, unlike during the following behavior simulations, the search algorithm failure rate was meaningless for the obstacle field behavior tests. This left the

solution path curvature and steering command as the main parameters used to evaluate the performance of the new temporal motion planning method in this open area obstacle avoidance situation.

As with the following behavior simulations, the analysis of steering control inputs commanded by the motion planning algorithm was the most basic and complete measure of the response of the vehicle to any disturbances in terms of solution trajectories. An obstacle crossing between the robot and its goal point at the edge of the grid had significant affects on the steering commanded by the planner as it attempted to navigate around the object to get to the goal point. This is evidenced in Figure 5-14 by the large magnitude of the steering commanded as the obstacle passes during the control test.

As was expected with no obstacle present, the original motion planning algorithm commanded 0% steering effort for the entire baseline test. However, as the obstacle passed across the vehicle's desired path during the control test, the steering command reached a magnitude of approximately 22.8%. Similarly concerning was the fact that actual command oscillated between positive and negative values of this magnitude in a short amount of time, signifying possible steering instability as the motion planner attempted to find a path in front of the passing obstacle initially only to switch to a path that fell behind the obstacle as it moved further. The PTMP method was able to match the baseline results, maintaining 0.0% steering command through the entire test, even as the obstacle crossed in front of the simulated vehicle. This is illustrated by Figure 5-15, which shows the individual layers of a temporal grid with the expanded search nodes and solution path of the temporal motion planning algorithm drawn as the predicted positions of the obstacle passed across the robot's desired path. The temporal layer shown in Figure 5-15(d) shows the predicted position of the obstacle at the fourth future time

step as having just passed the fourth generation of search nodes expanded by the motion planner such that it did not interfere with the solution path.

Analysis of the solution path deviation further substantiated the claim that the new temporal motion planning method would successfully generate trajectories that ignored the passing obstacle. The average path deviation for the baseline test was approximately 14.5° with a maximum value of 23.8°. The results from the control test showed a significant deviation of the path as the obstacle crossed in front of the vehicle. An average deviation of approximately 31.7° represented a 118% increase for the control test when compared to the baseline test, and the maximum deviation of 301.1° represented a 1,162% increase when compared to that of the baseline test. The new PTMP method was able to achieve results almost identical to those of the baseline test with an average deviation of approximately 14.4° and a maximum value of 24.1°. Figure 5-16 plots the results of the three tests conducted and shows the large jump in solution path deviation experienced during the control test when using the original motion planning algorithm, while the deviations from the baseline and temporal tests remained small and consistent.

Target Interception

The final simulation conducted to test the usefulness of the PTMP method was an assessment of the method's ability to facilitate a target interception capability. This test was set up in a similar manner to the obstacle field behavior simulation. However, rather than treating the predicted positions of the obstacle in each temporal layer as untraversable areas, the final predicted position in the last temporal layer was used to set the goal point for the motion planning algorithm to achieve. The same open, unstructured zone from the obstacle field behavior test was used, and the simulated obstacle began in a slightly different initial position of 20 m in front of the vehicle and 20 m to its left. The same straight-line obstacle trajectory was

followed, maintaining a velocity of 2.24 mps (5 mph) as it crossed the open area. Figure 5-17 shows the fully-populated layers of a temporal grid displaying the predicted positions of the target and Figure 5-18 shows the optimized temporal grid with the same predicted positions.

The simulation was conducted in a static manner such that the vehicle was stationary while the target crossed the open area in front of it. This approach was chosen because the position simulator did not actually consider the control commands of the vehicle when changing the vehicle position. Rather the vehicle would have travelled in a straight line among the waypoints defining the simulated path of traversal. Keeping the vehicle stationary allowed the natural output trajectory of the motion planning algorithm to be recorded and analyzed. The target was placed in motion by the simulated MO sensor after the TGC and PTMP began recording data.

Because this was a new application which could not be accomplished by the original motion planning algorithm as it was dependent on the predictive nature of the new motion planning method, it was unnecessary to compare the results of the simulation with any baseline or control test results. Therefore, the simulation was run only using the new PTMP algorithm, with the same performance metrics recorded. Evaluating the ability of the PTMP to successfully generate a trajectory to the calculated goal point coming from the predicted position of the target was the ultimate goal of this simulation.

The new method resulted in a 100% success rate in generating this trajectory, thus proving that this application had merit. Figure 5-19 depicts the PTMP populating its tree of nodes in the subsequent temporal layers of the grid, along with the final solution trajectory. This image also serves to show that the new motion planner was attempting to generate the trajectory to the goal point as defined by the final predicted position of the target, which was achieved and shown in Figure 5-19(g). Figures 5-19(e) and 5-19(f) show earlier predicted positions of the target, also

painted in green, but the search tree was still expanding in the direction of the final predicted position.

Figure 5-20 illustrates how well the new motion planning algorithm tracked the goal point as calculated by the final predicted position of the target object. The heading angle from the vehicle and the final predicted position of the target object and the heading angle from the vehicle to the final node on the solution trajectory are plotted to show how well the trajectory led the robot to intercept the target. An average discrepancy between these two heading angles of approximately 0.97° was achieved during the simulation, with a maximum value of 3.7°. The small magnitude of these values coupled with the perfect success rate of the motion planning algorithm confirmed that this target interception application was a feasible task for the new PTMP method. However, there was still room for improvement as this test only proved that the new method could be used to track the future position of the target and generate control inputs to drive the robot to that position. Velocity planning needs to also be considered in the event the target is moving faster or slower than the robot, and care must be taken so that the rendezvous is safe and collision free.

Testing Summary

The test scenarios described in the previous sections sought to quantify the performance of the new PTMP method and to compare the new method to an existing motion planning algorithm implementation when attempting to generate optimal trajectories in environments with moving obstacles. The results discussed for the following behavior and obstacle field behavior tests have shown that the new PTMP method exhibited improved performance in this type of situation. For each test scenario, a single obstacle was placed in motion and its positions were predicted and incorporated into the temporal grid. Table 5-5 displays the prediction times associated with each layer of the temporal grid for the various tests. These times corresponded with the steps used by

the motion planning algorithm in exploring potential vehicle states to determine an optimal trajectory.

The PTMP method was able to closely match the results of the baseline tests when comparing important parameters, as opposed to the original motion planning algorithm used during the control tests, which struggled for the chosen behaviors. It achieved a perfect success rate in attempting to generate trajectories while following a moving obstacle, as well as when attempting to navigate an obstacle field with a moving object crossing its desired path, and was able to minimize the total path deviation and associated steering commands. On the other hand, the control tests showed that the original motion planning algorithm struggled when encountering the moving objects, resulting in an unacceptable failure rate. It also generated paths that exhibited large deviations from a desired nominal, straight path and resulted in significant steering commands that would have driven the robot off the road or could have led to unstable oscillatory motions.

The new PTMP method also facilitated a target interception application that was not possible when using the compared original motion planning algorithm. The newly developed algorithm allowed for autonomously navigating to intercept a moving target object. It exhibited a perfect success rate in generating trajectories towards the final predicted position of the target object with minimal path deviation. This new application, coupled with the significant improvement in performance, in terms of the key performance metrics discussed, achieved during the behavioral test cases has proven that this new temporal motion planning method may be used to better manage the motion planning problem in dynamic environments.

This chapter outlines the test plan used for the validation of the new temporal motion planning method and discusses the results of those tests in detail. The tests concentrated on the

following and obstacle field operating behaviors and a new target interception application to show the advantages of the new method and its improved performance over the original static motion planning algorithm. The following chapter draws conclusions from the results of these tests and introduces several areas of recommended additional work for this new motion planning method.

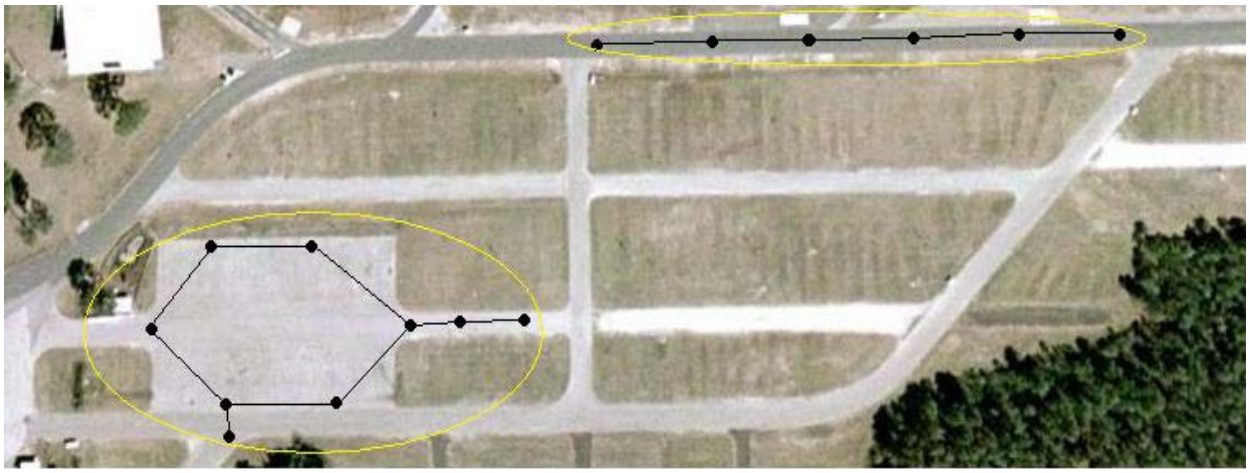

Figure 5-1. Gainesville Raceway pit area with points defining test areas.

Figure 5-2. Straight road segment selected for following behavior test with defining waypoints.

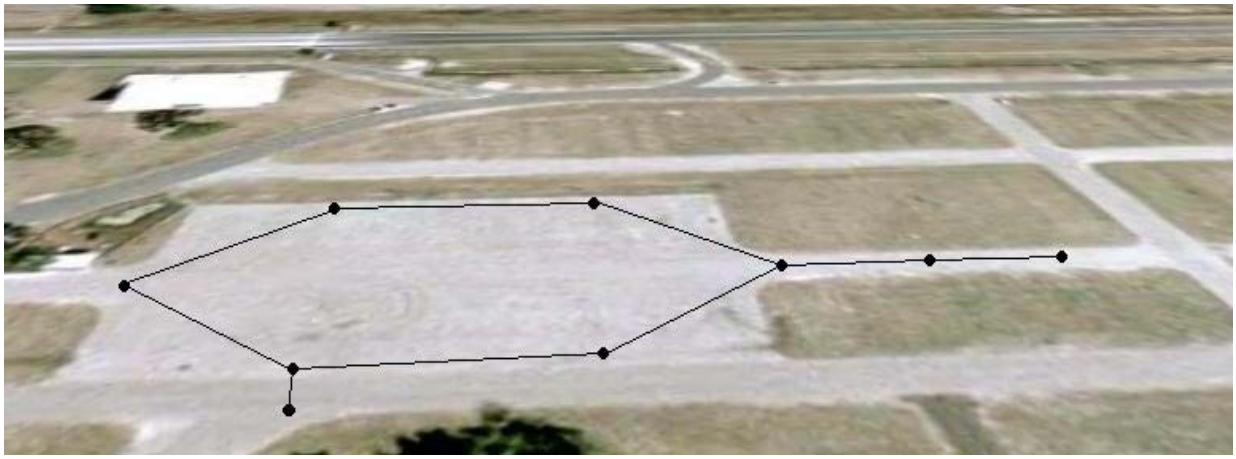

Figure 5-3. Open unstructured area with defining waypoints and perimeter points drawn.

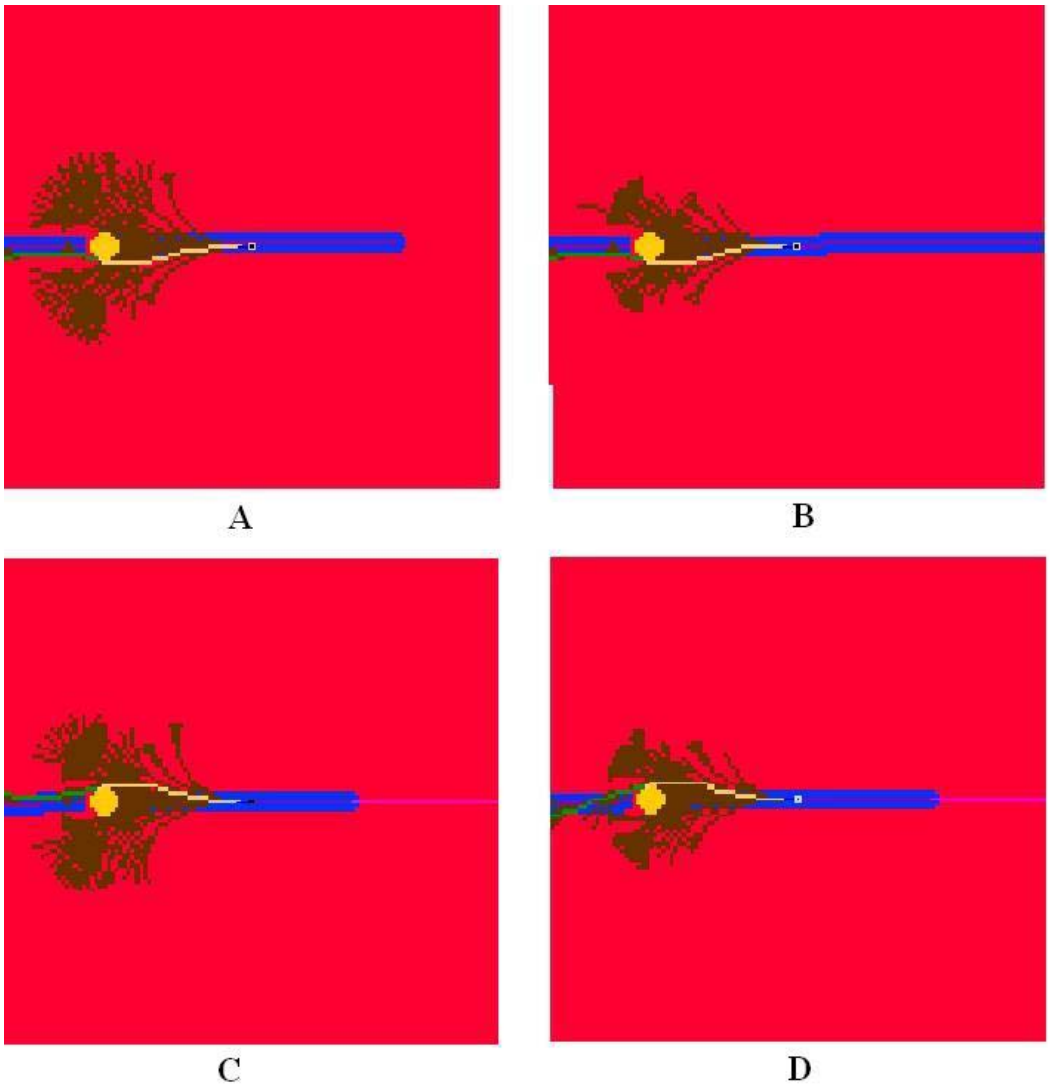

Figure 5-4. Sample following behavior control test results.

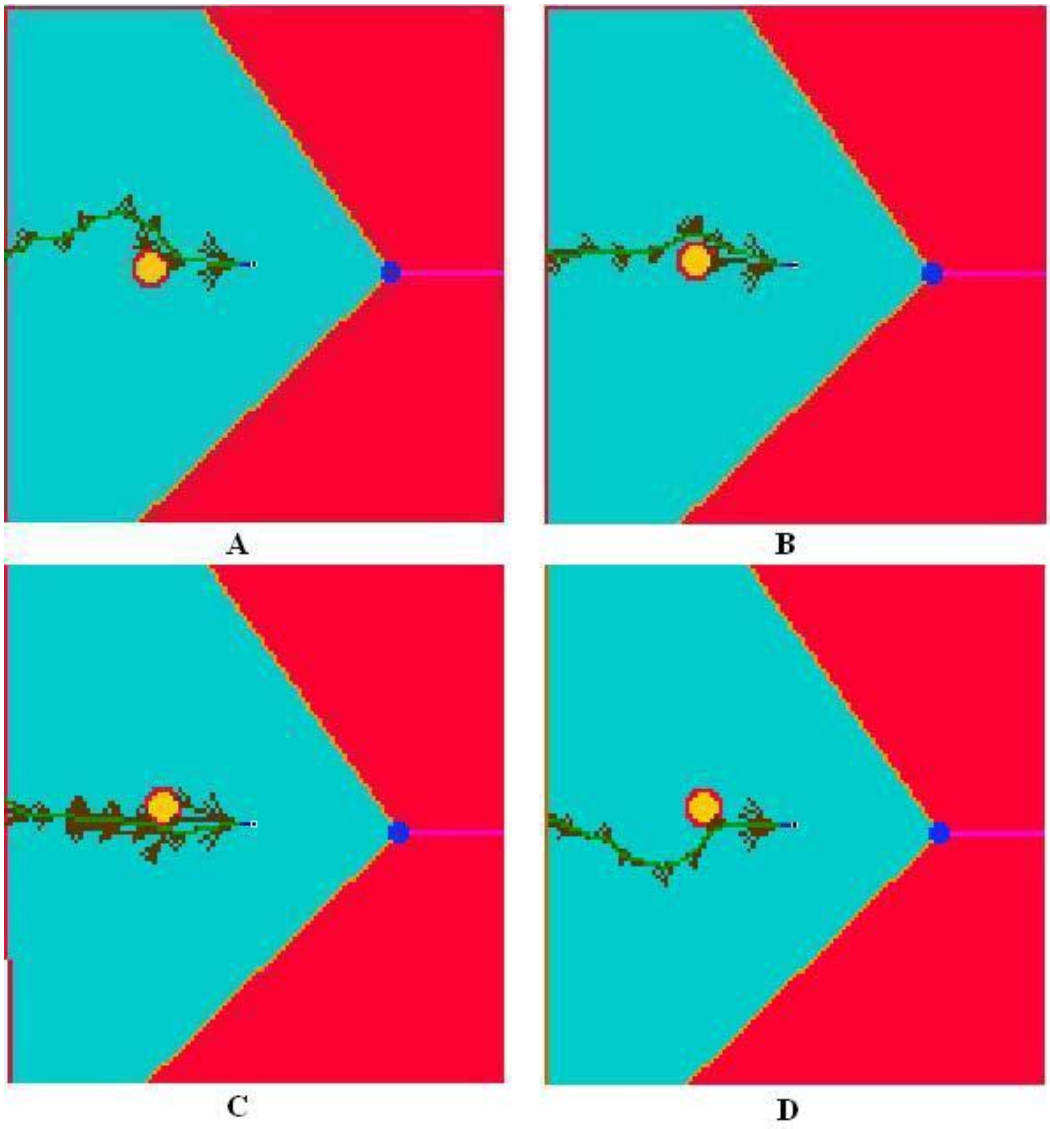

Figure 5-5. Sample obstacle field behavior control test results.

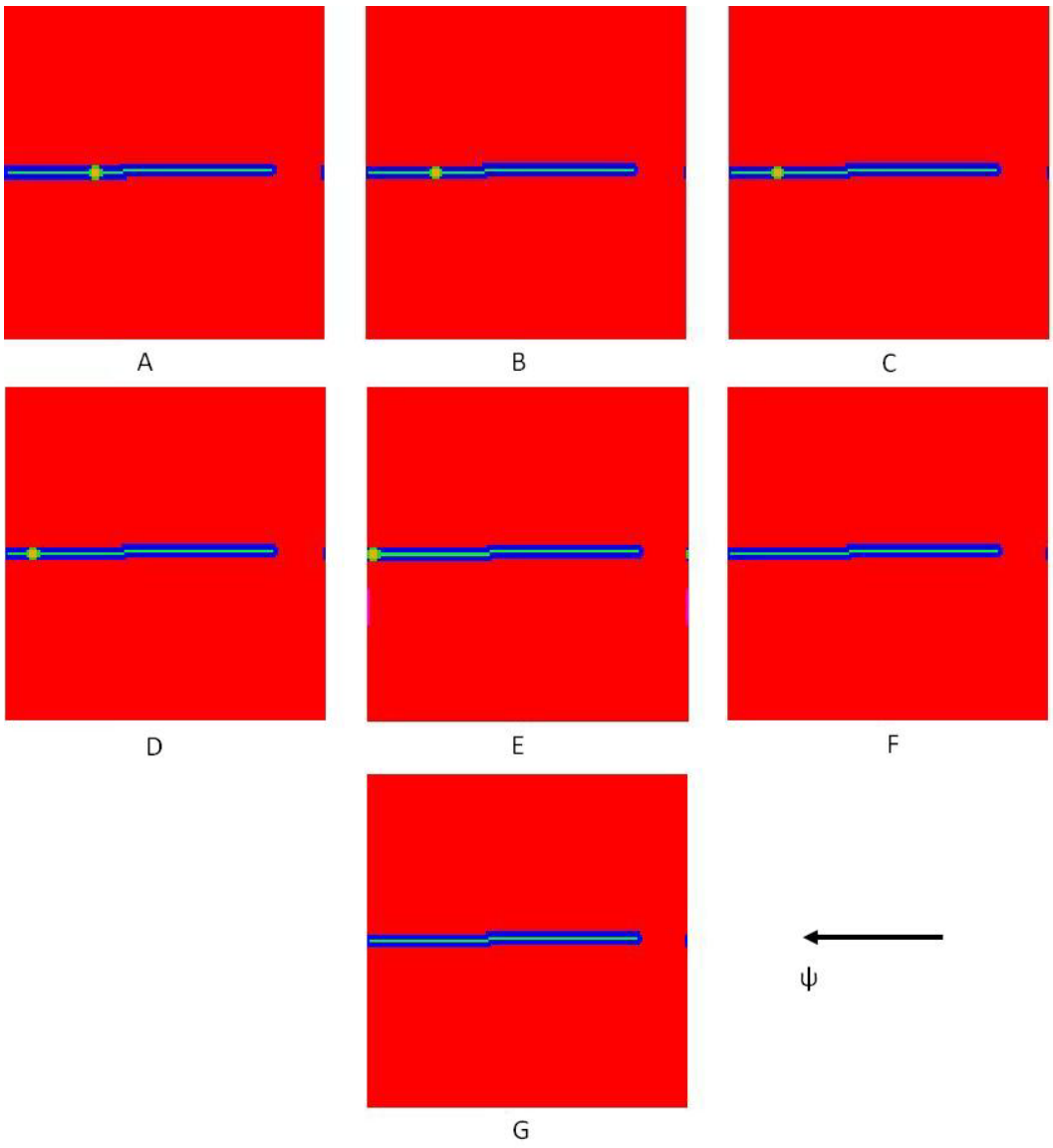

Figure 5-6. Sample full temporal grid layers with predicted position of followed object.

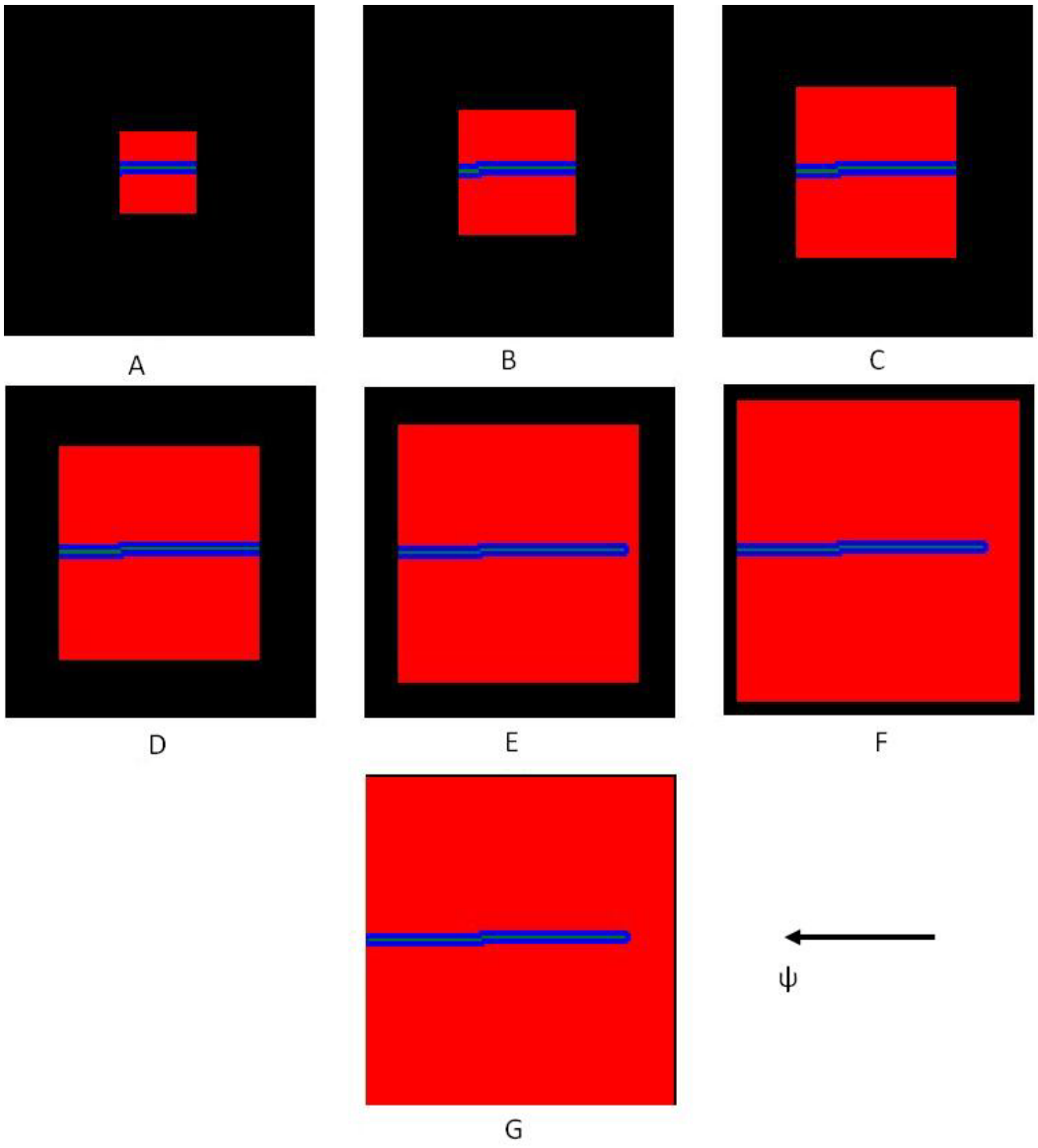

Figure 5-7. Sample optimized temporal layers of following behavior test temporal grid.

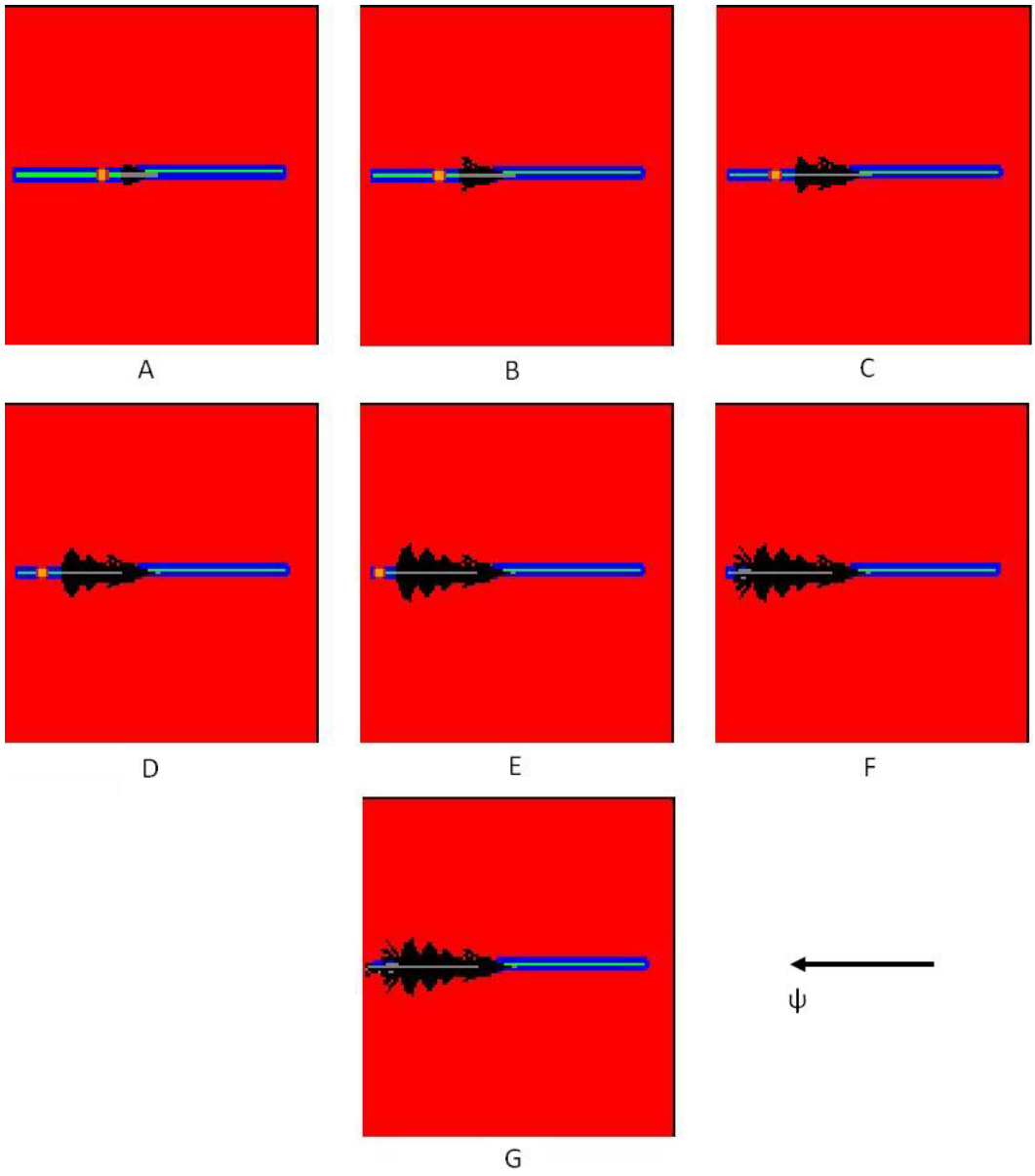

Figure 5-8. Sample output temporal grid from following behavior.

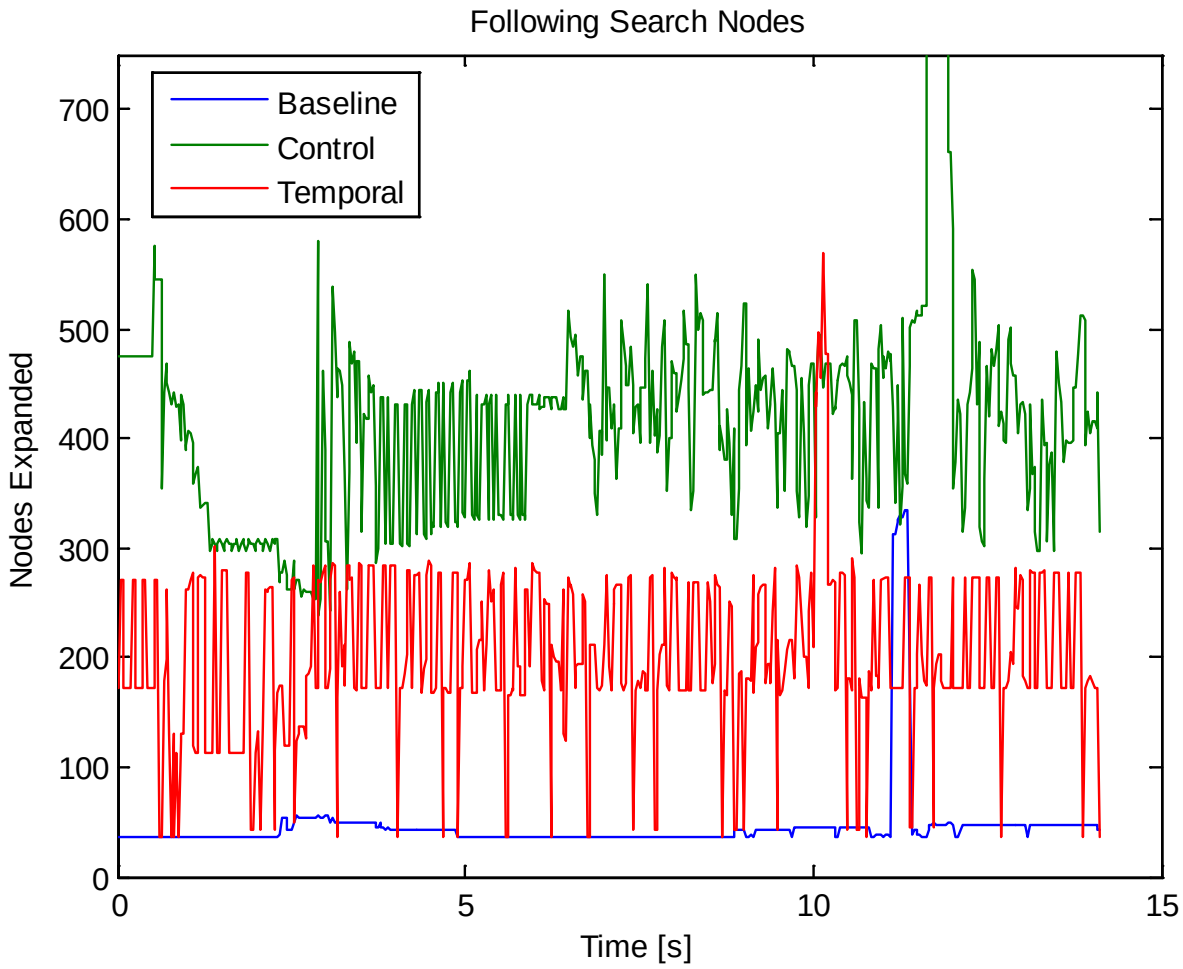

Figure 5-9. Number of expanded search nodes for following operating behavior tests.

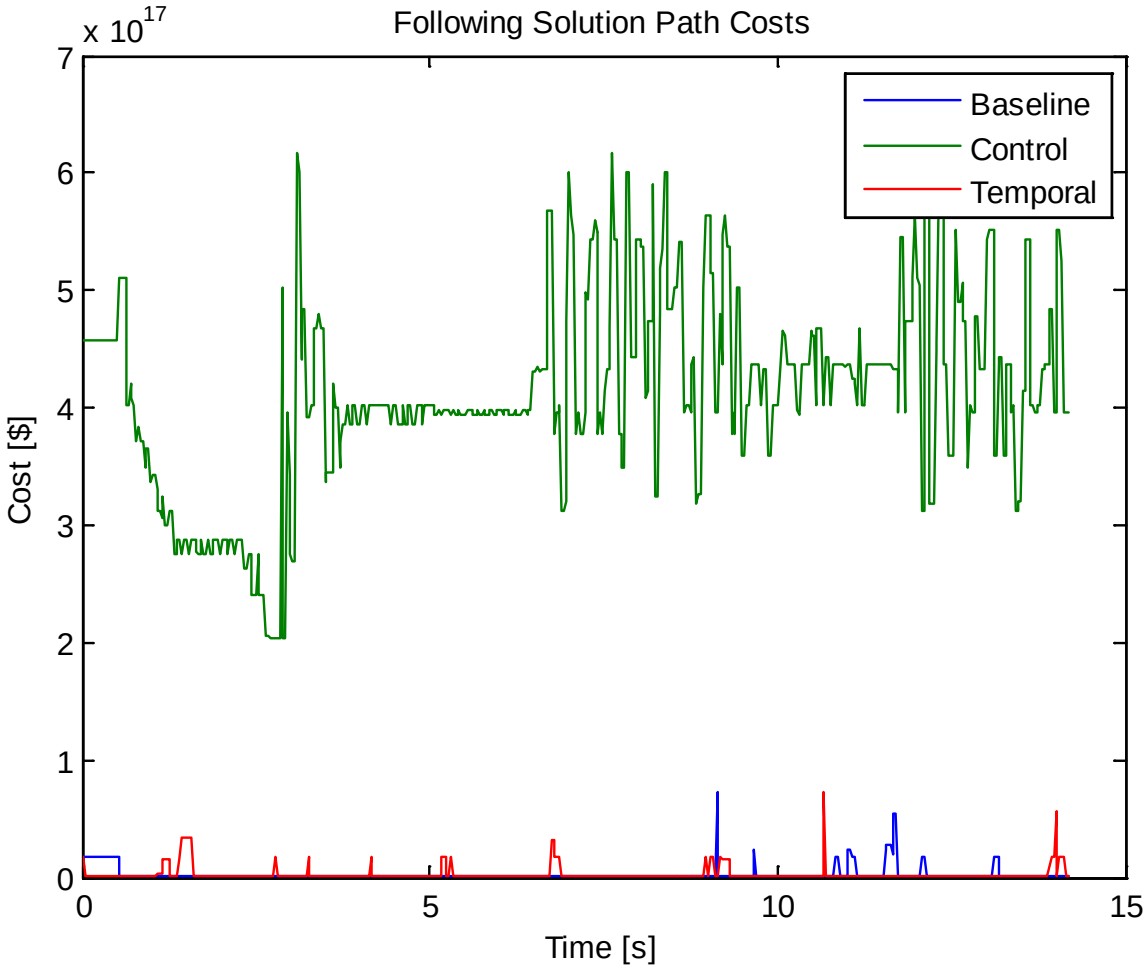

Figure 5-10. Solution trajectory costs for the following behavior tests.

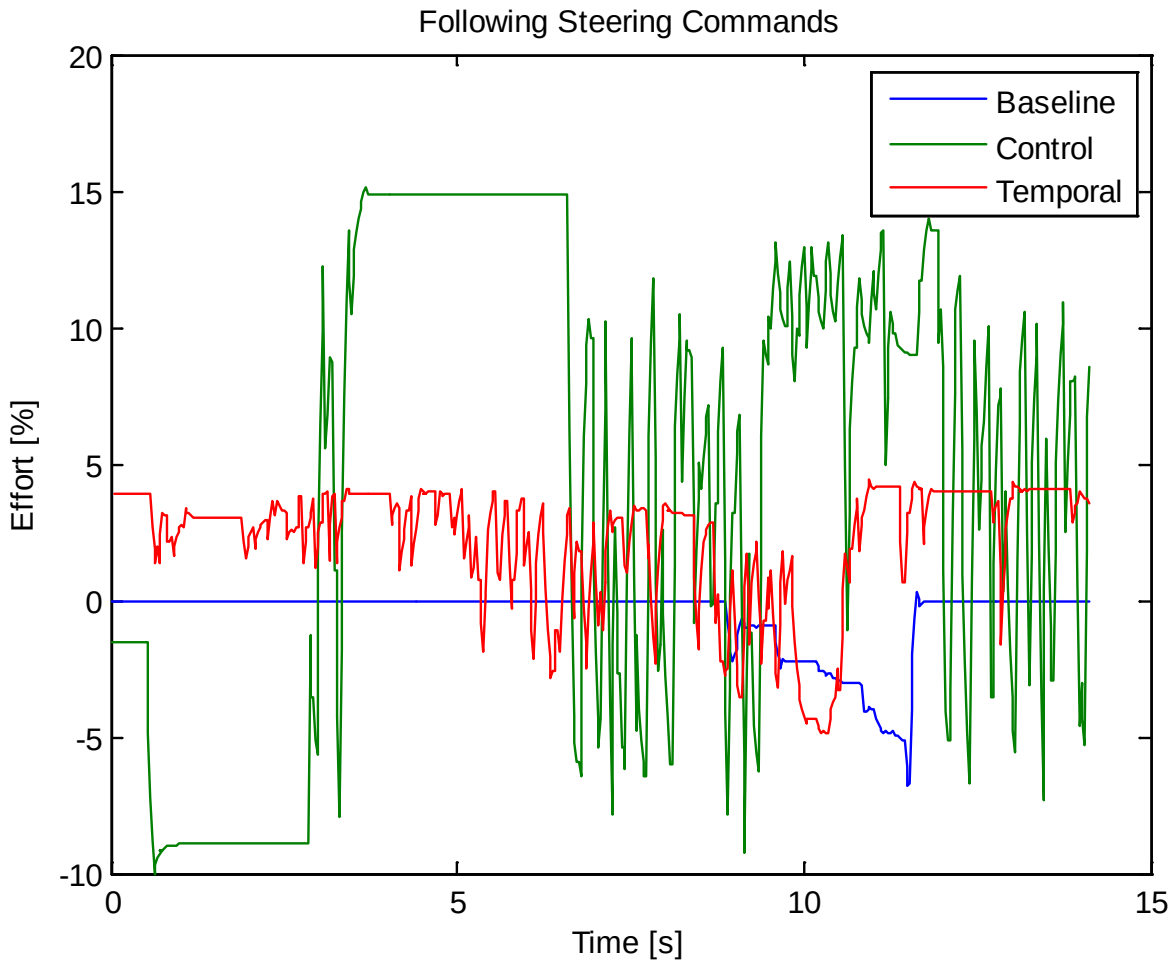

Figure 5-11. Steering commands for following behavior tests.

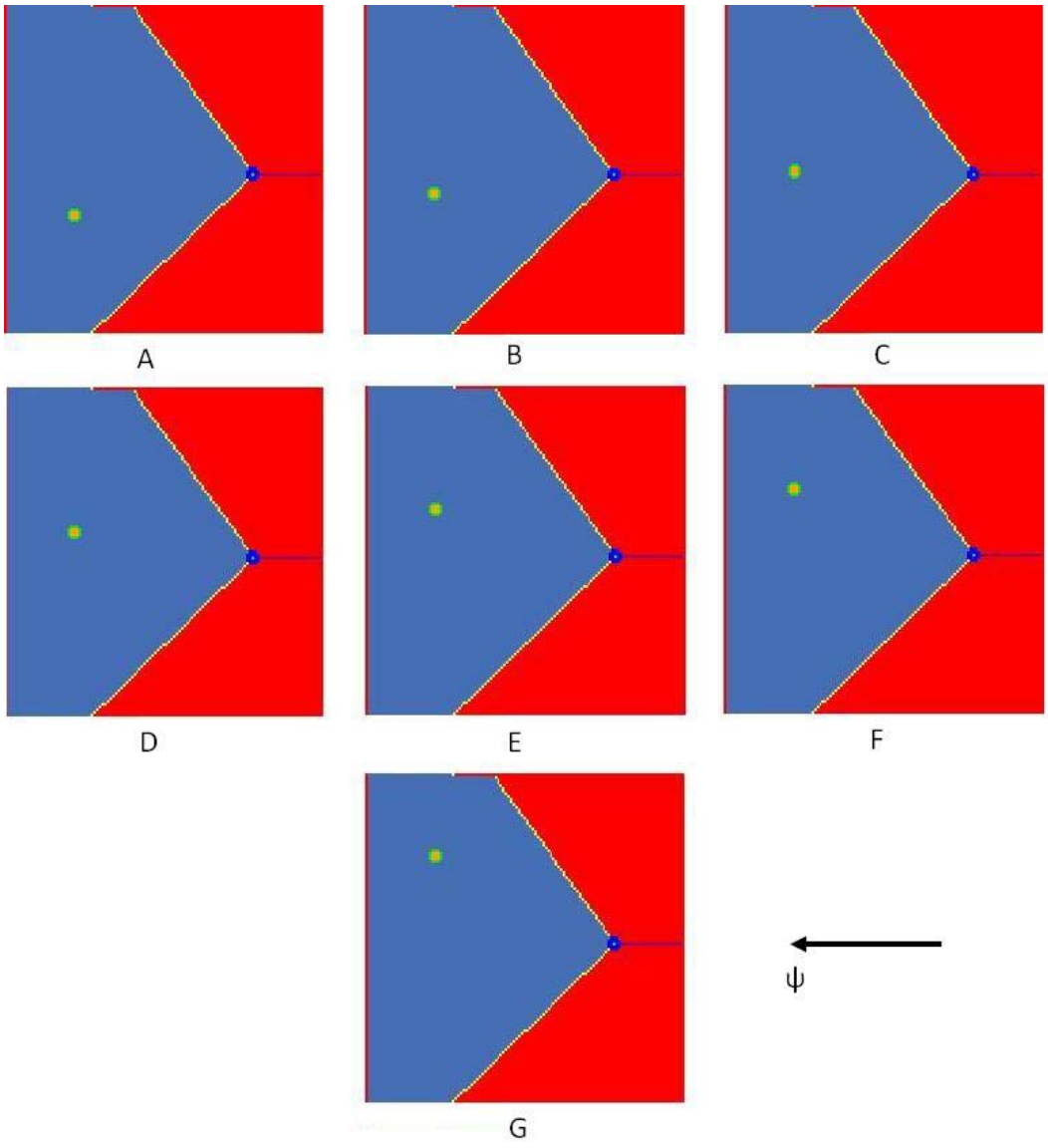

Figure 5-12. Sample full temporal layers showing predicted positions of obstacle.

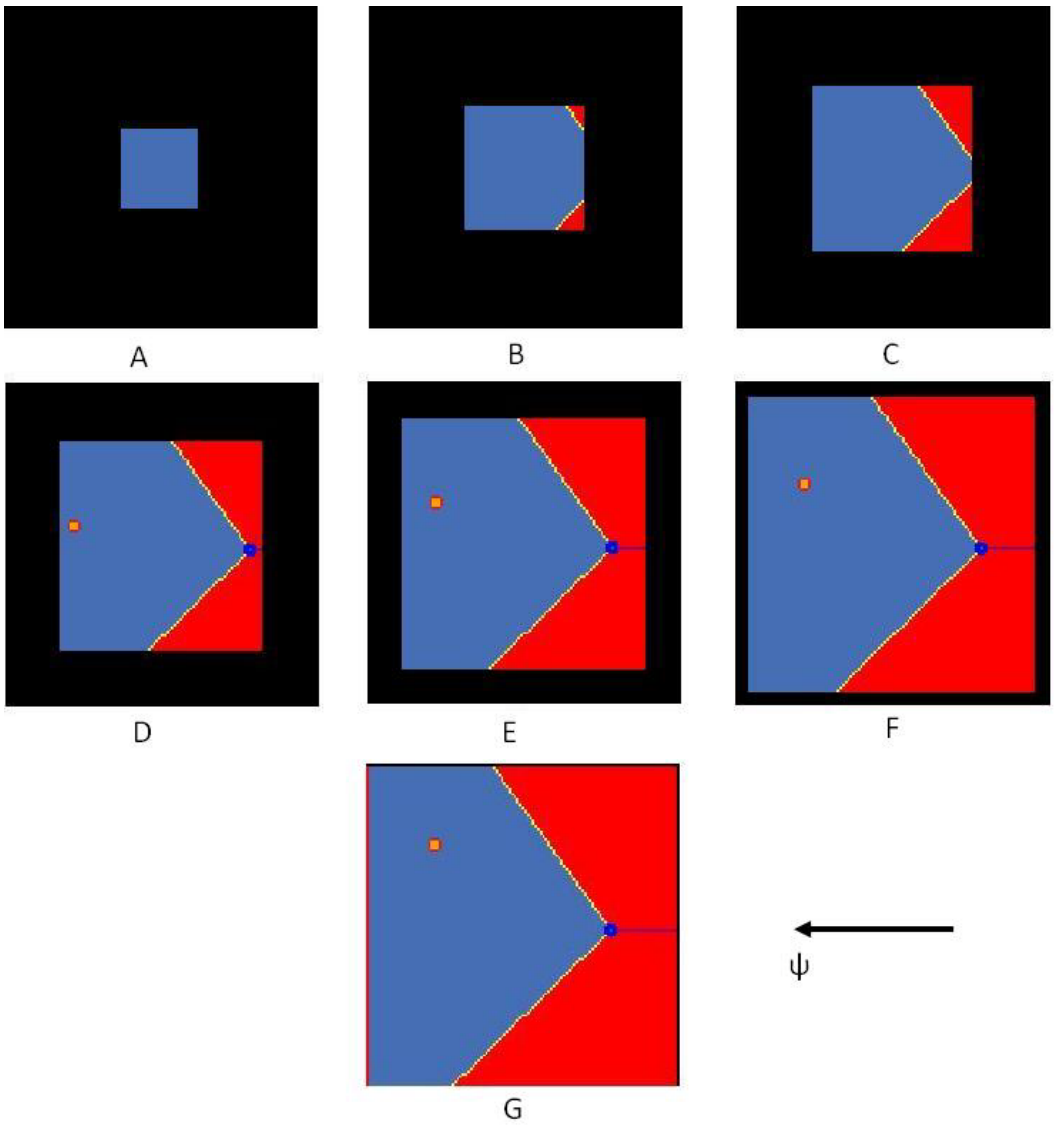

Figure 5-13. Sample optimized temporal grid layers from obstacle field behavior test..

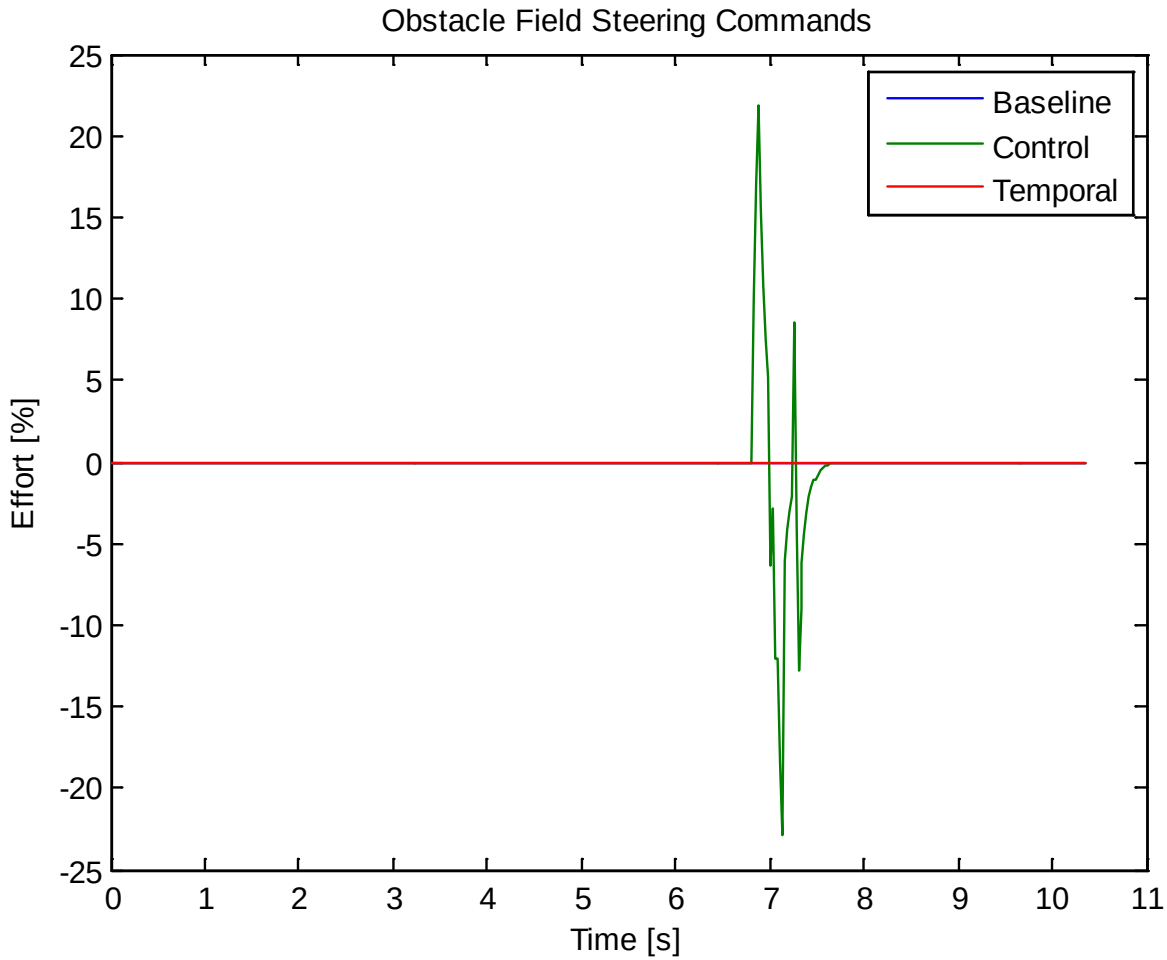

Figure 5-14. Steering commands for obstacle field behavior tests.

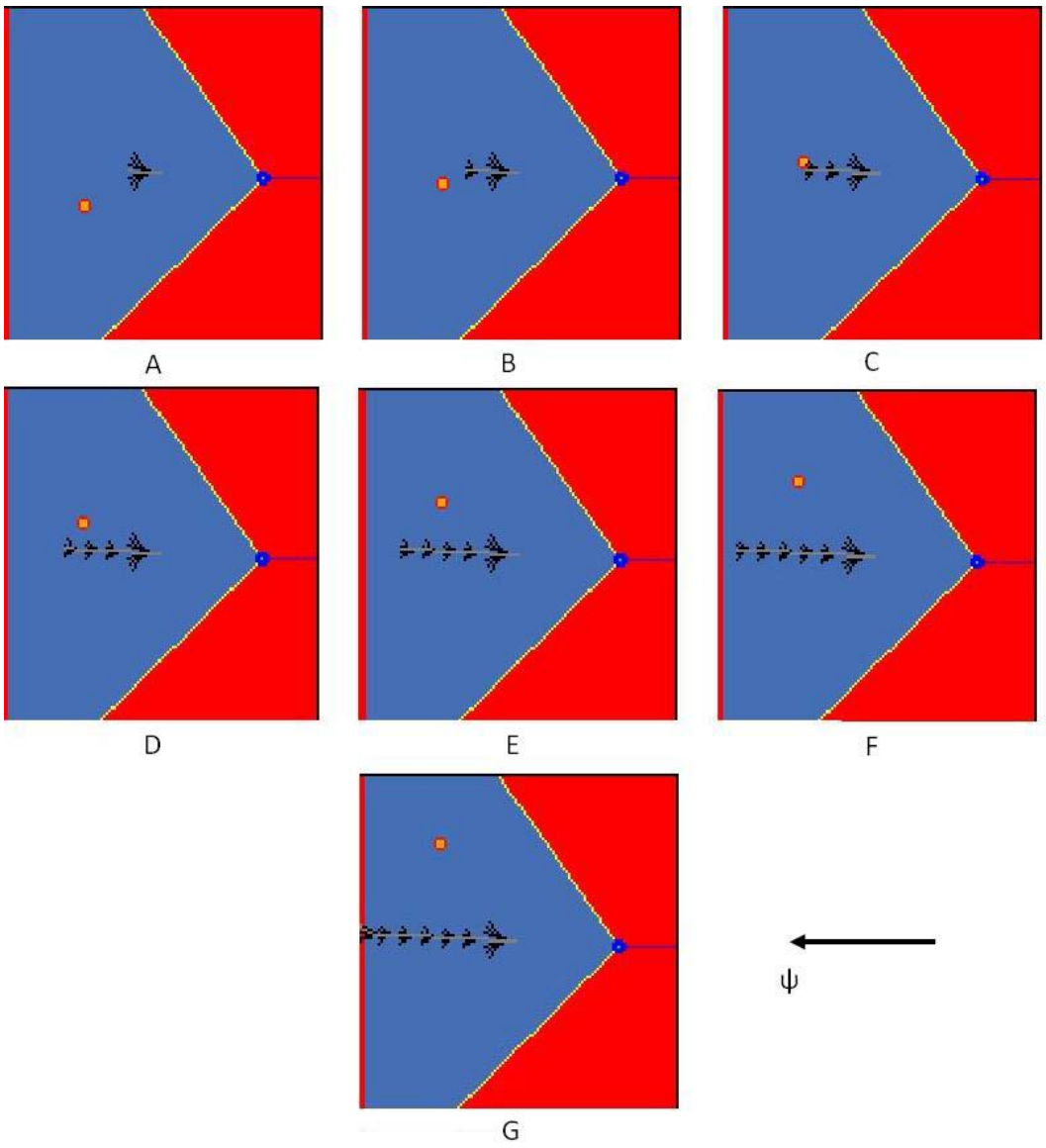

Figure 5-15. Sample output temporal grid of obstacle field behavior simulation.

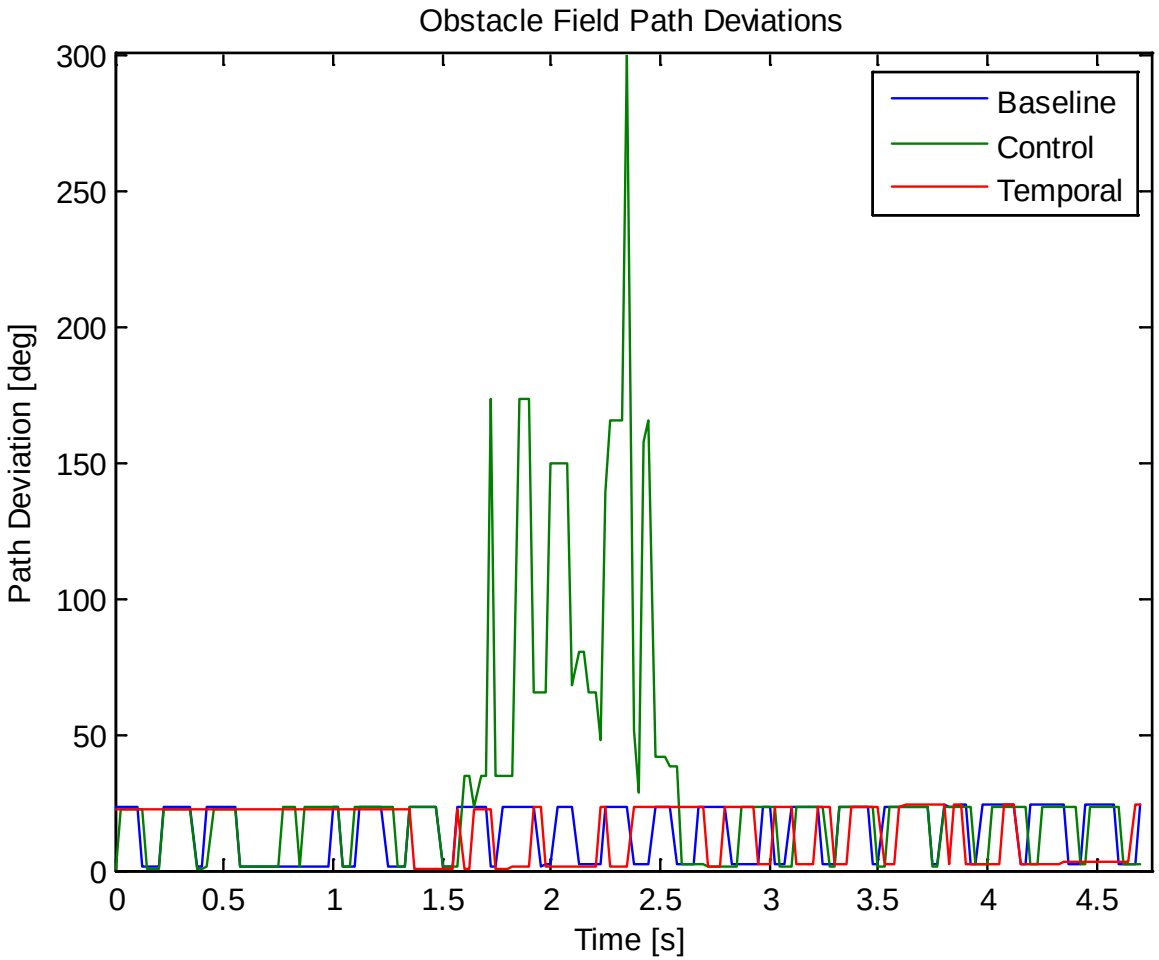

Figure 5-16. Path deviation results for obstacle field behavior tests.

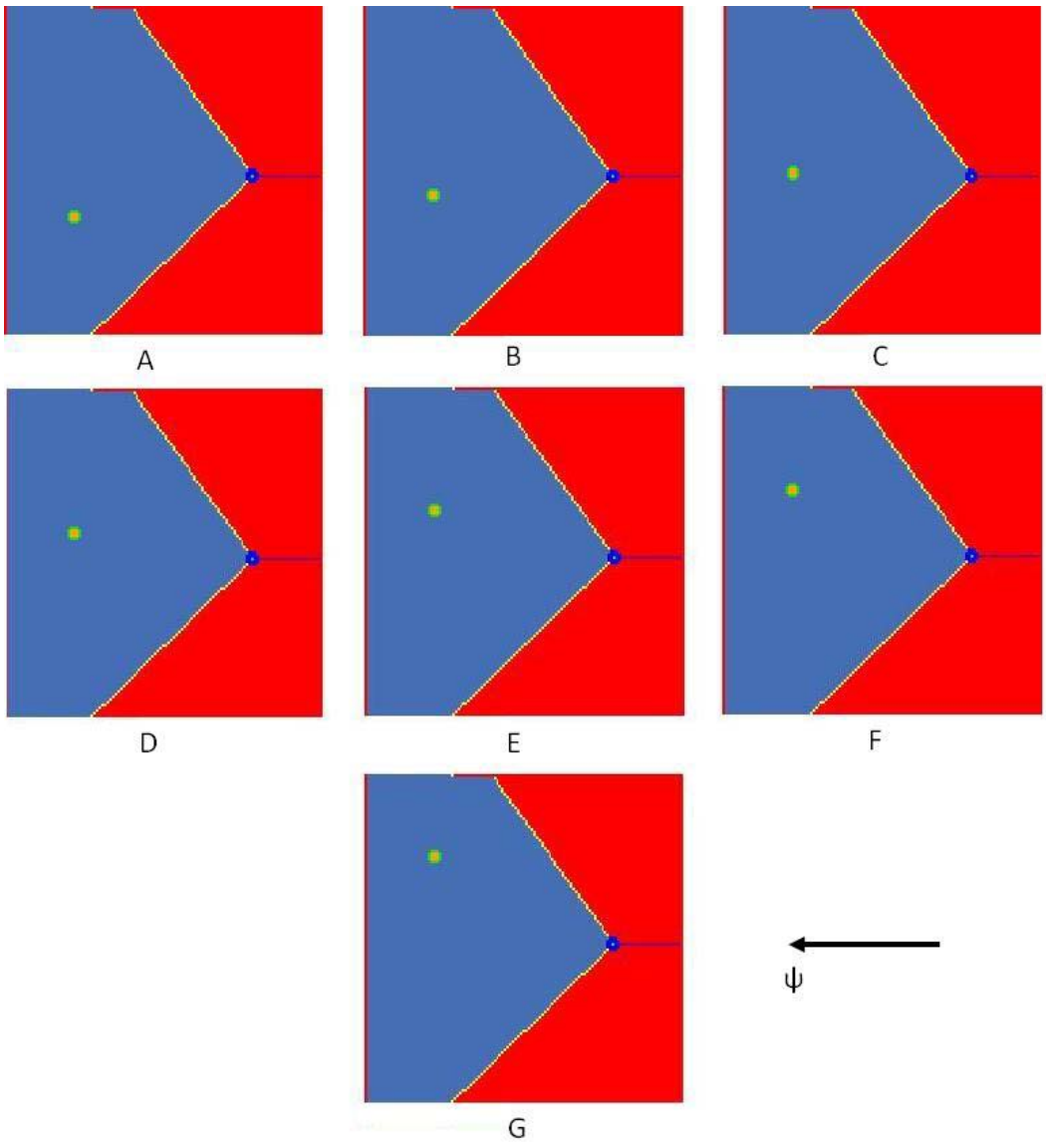

Figure 5-17. Sample full temporal grid layers showing progression of target object.

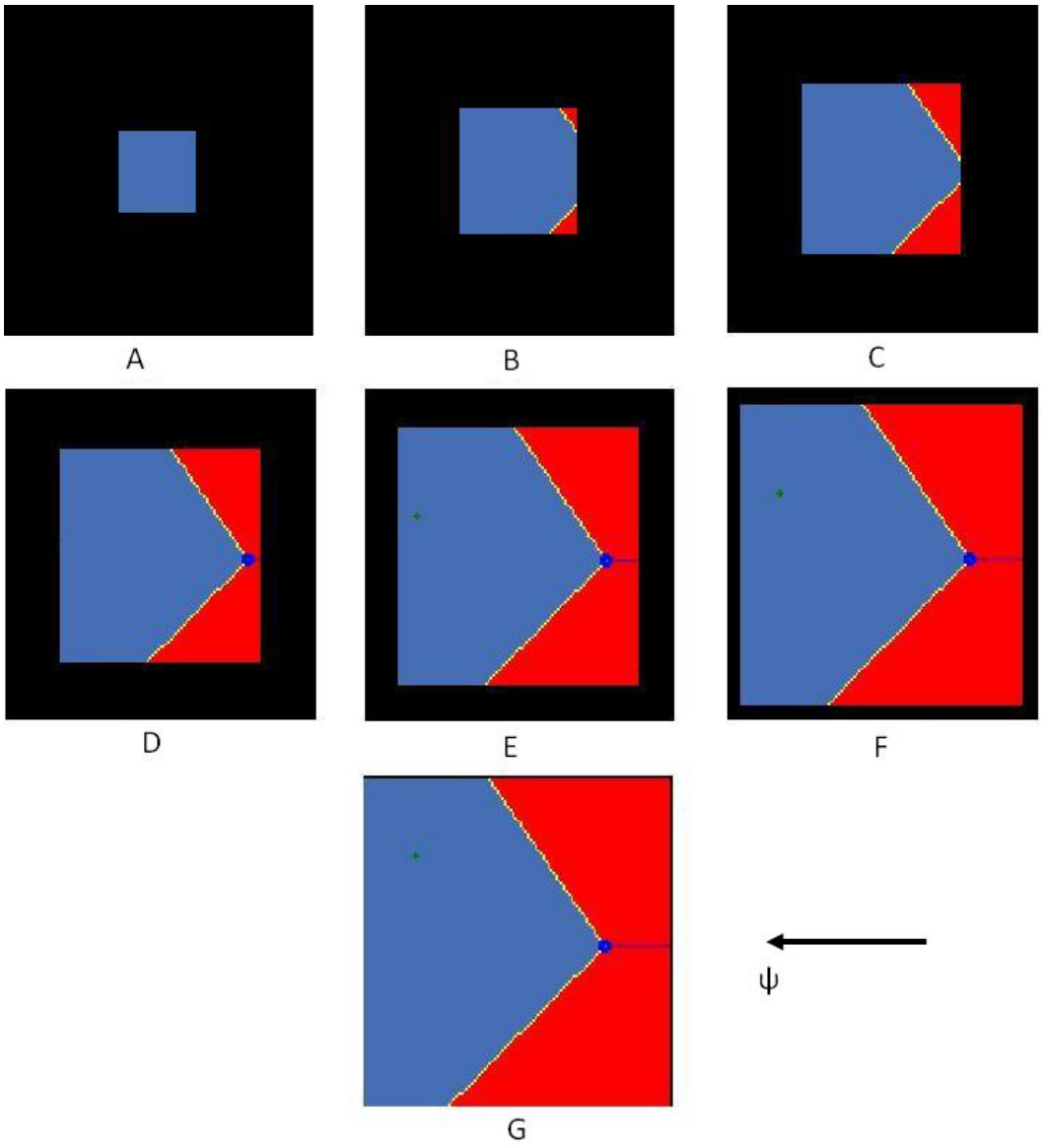

Figure 5-18. Sample optimized temporal grid layers showing progression of target object.

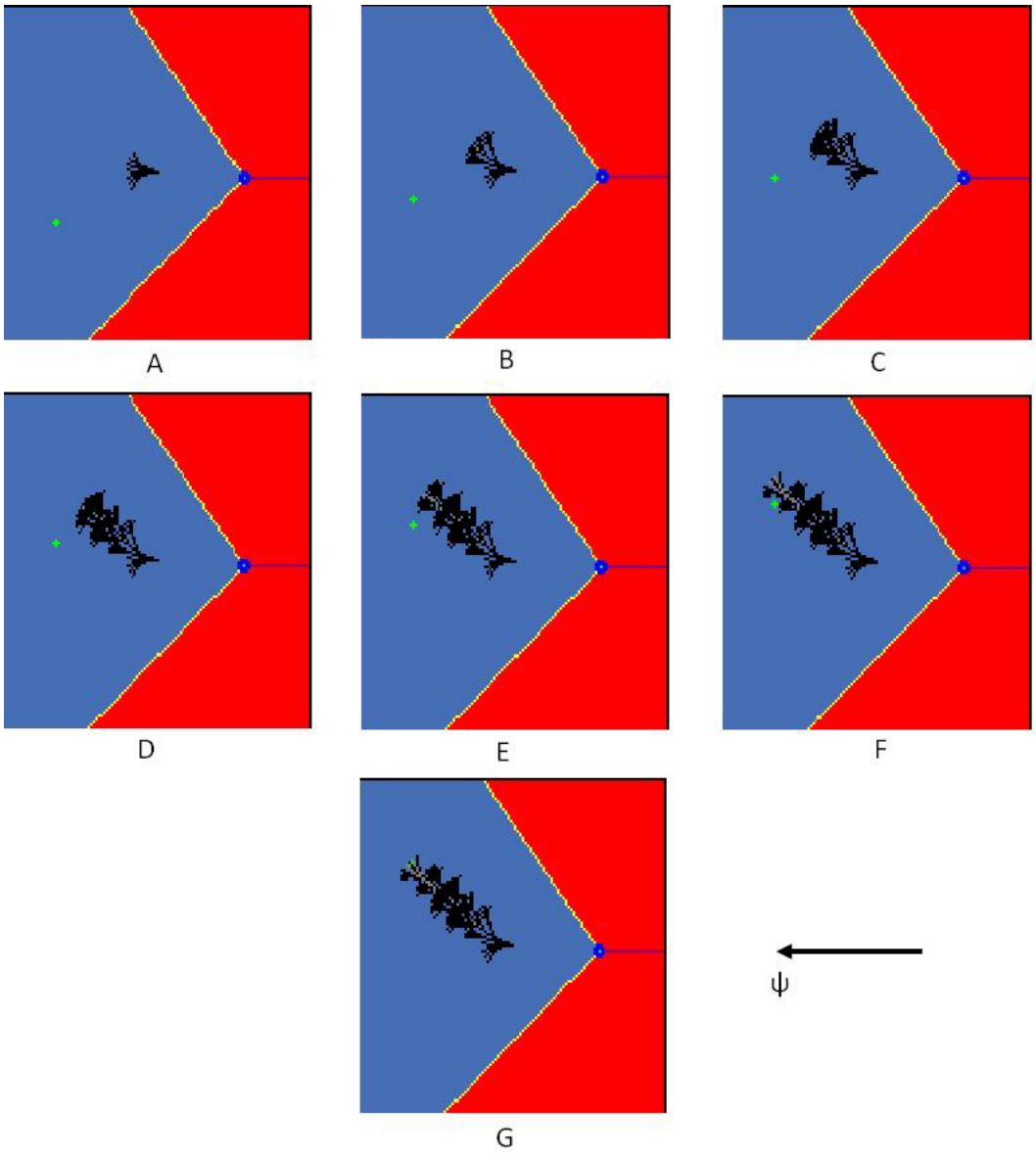

Figure 5-19. Sample full temporal grid output of motion planner during target interception test.

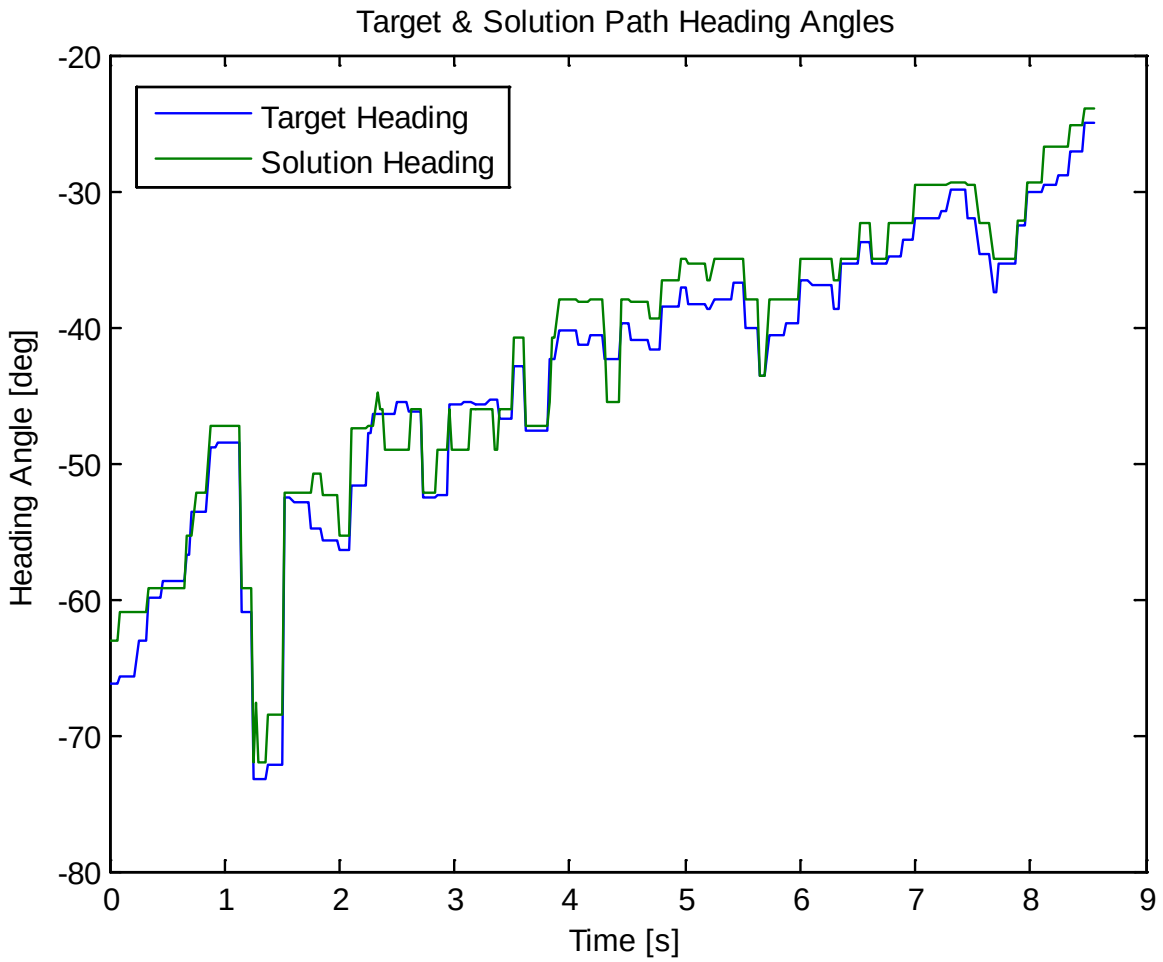

Figure 5-20. Target and solution path heading angles for target interception test.

Table 5-1. Following behavior test plan.

Purpose: Establish the PTMP method's ability to navigate a mobile ground robot down a desired lane of travel while following another moving vehicle, and to compare this performance with that of a non-temporal motion planner. This test will examine one of the simplest operating behaviors involving interaction with moving obstacles executed by the vehicle and could lead into more complicated behaviors.
Design: The vehicle will attempt to navigate down a straight road segment while following behind a moving vehicle in the same lane travelling in the same direction. The path will be built as a series of waypoints from an existing RNDF of the test facility and a test-specific MDF. The obstacle vehicle will maintain a speed approximately equal to of the robot such that no risk of the robot colliding is present. Robot velocity: (-4.4704 m/s, 0.0) Obstacle initial position: (-18.0 m, 0.0 m) Obstacle velocity: (-4.4704 m, 0.0 m)
Expected results: The PTMP method will show successful trajectory generation that allows the vehicle to navigate down the center of its desired lane of travel behind another moving vehicle in its lane.
Logged measurements: Input parameters: GPOS, VSS Output parameters: Steering effort, Solution path Other parameters: Component update rates, Goal point, Search success rate, Search node usage, Solution path deviation, Obstacle positions, Obstacle predicted positions

Table 5-2. Obstacle field behavior test plan.

Purpose: Establish the PTMP method's ability to navigate an open, unstructured area with moving obstacles crossing its path, and to compare this performance with that of a non-temporal motion planner. This test will examine and record the method's obstacle avoidance capabilities.
Design: The vehicle will attempt to navigate through an open, unstructured area in which it has nothing defining where it must travel, while a moving obstacle crosses perpendicularly to its desired path in such a way that it will be out of the way by the time the robot reaches that crossing position. The open area will be defined in an RNDF only as a set of perimeter points. Robot velocity: (-2.235 m/s, 0.0 m/s) Obstacle initial position: (-15.0 m, -15.0 m) Obstacle velocity: (0.0 m/s, 4.4704 m/s)
Expected results: The PTMP method will successfully generate a trajectory that allows the vehicle to achieve its goal while not being diverted by the obstacle crossing its path.
Logged measurements: Input parameters: GPOS, VSS Output parameters: Steering effort, Solution path Other parameters: Component update rates, Goal point, Search success rate, Search node usage, Solution path deviation, Obstacle positions, Obstacle predicted positions

Table 5-3. Target Interception test plan.

Purpose: Establish the use of the PTMP method for a target interception application.
Design: The vehicle will attempt to track the final predicted position of an obstacle traversing an open, unstructured area and generate a trajectory to intercept that position by treating it as the motion planning algorithm's goal. The target will move in a straight line in front of the robot. The robot will remain stationary throughout the test. Robot velocity: (0.0 m/s, 0.0 m/s) Obstacle initial position: (-20.0 m, -20.0 m) Obstacle velocity: (0.0 m/s, 2.235 m/s)
Expected results: The PTMP method will successfully treat the final predicted position of the target object as its goal point and generate a trajectory that will intercept this position. The test will take place in the same open area environment used for the obstacle field tests.
Logged measurements: Input parameters: GPOS, VSS Output parameters: Steering effort, Solution path Other parameters: Component update rates, Target goal point, Search success rate, Search node usage, Solution path deviation, Obstacle positions, Obstacle predicted positions

Table 5-4. Sample of predicted positions and velocities for following behavior testing.

Temporal layer	X position (m)		Y position (m)		X velocity (m/s)	Y velocity (m/s)
	Local	Global	Local	Global		
1	-12.96	378087.7	0.14	3292661.0	-4.47	0.00
2	-17.00	378083.6	0.14	3292661.0	-4.47	0.00
3	-21.11	378079.5	0.14	3292661.0	-4.47	0.00
4	-25.29	378075.3	0.14	3292661.0	-4.47	0.00
5	-29.52	378071.1	0.14	3292661.0	-4.47	0.00
6	-33.83	378066.8	0.14	3292661.0	-4.47	0.00
7	-38.20	378062.4	0.14	3292661.0	-4.47	0.00

Table 5-5. Sample of predicted positions and velocities for obstacle field behavior testing.

Temporal layer	X position (m)		Y position (m)		X velocity (m/s)	Y velocity (m/s)
	Local	Global	Local	Global		
1	-15.98	377378.8	-7.47	3292548.1	0.0	4.47
2	-15.98	377378.8	-3.42	3292552.2	0.0	4.47
3	-15.98	377378.8	0.63	3292556.2	0.0	4.47
4	-15.98	377378.8	4.67	3292560.3	0.0	4.47
5	-15.98	377378.8	8.70	3292564.3	0.0	4.47
6	-15.98	377378.8	12.73	3292568.3	0.0	4.47
7	-15.98	377378.8	16.75	3292572.3	0.0	4.47

Table 5-6. Prediction times for three test scenarios at various temporal.

Temporal layer	Distance step (m)	Following behavior prediction times (s)	Obstacle field prediction times (s)	Target interception prediction times (s)
1	6.0	1.3422	1.3422	2.6846
2	10.0	2.2370	2.2370	4.4743
3	14.0	3.1317	3.1317	6.2640
4	18.0	4.0265	4.0265	8.0537
5	22.0	4.9213	4.9213	9.8434
6	26.0	5.8160	5.8160	11.6331
7	30.0	6.7180	6.7180	13.4228

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

This dissertation describes a new and novel motion planning method developed by the author that considered the predicted motions of any obstacle present in the immediate environment. This newly termed predictive temporal motion planning method allowed a robotic system to generate a trajectory through a dynamic environment to successfully achieve its goal state. This final chapter seeks to close the discussion of this new method by first presenting several areas of potential future work that arose during the development and testing of the new planning method. It then draws conclusions from the theory and implementation discussed in Chapters 3 and 4 and from the test results of the validation process described in Chapter 5. It also outlines the main contributions of the PTMP method to the field of robotics.

Future Work

The research presented in this document laid the case for the feasibility and advantages of the PTMP method, but there are still several advancements that can improve the new method further and several other applications for which the method could be effectively used that are now discussed. The first of these improvements dealt with the continued optimization of the temporal grid structure. The implemented optimization technique involved simply building each temporal layer of the grid up to a size that was reachable by the associated generation of search node expansion, thus resulting in a pyramid-type structure with the initial temporal layers being represented by fairly small grids and the latter temporal layers being nearly fully-populated grids. While this was much more efficient than simply building a fully-populated grid for each temporal layer, it still allowed for the duplication of cells in different temporal layers that did not change traversability value from one layer to the next. Ideally, a tree structure approach could be used to add new branches to each cell only when the traversability value of that cell changed.

The associated time-step would need to be recorded for each change so that the motion planning algorithm could intelligently search through this tree structure to find the correct traversability value. This approach would result in the minimum size of the temporal grid structure being achieved, as new cells were added only when absolutely necessary.

Another area of future work considered the resizing of the temporal grid structure if the key search parameters changed during runtime. Testing for the presented research maintained constant values of initial time-step, normal time-step, and search horizon time; however, it was possible and likely that these parameters would change value as the robot executed different behaviors and experienced changes in speed. As these values changed, the number of temporal layers and the dimensions of each temporal layer that make up the temporal grid structure may have become invalid and would be required to be reconfigured to account for the new time-step or time horizon values. This would have required that the existing temporal grid structure was destroyed in memory, values for the number of temporal layers and dimensions of each layer were recalculated, and the new temporal grid structure was generated and populated with their corresponding traversability values. This ability would have allowed for more robustness as the robot executed long missions where the operating behaviors and conditions may have changed many times.

The implemented version of the PTMP method generated the minimum number of temporal layers as determined by calculating the minimum number of steps from the vehicle's location to the time horizon, considering the given time-step values. This approach worked well when generating a straight trajectory to the goal, in which case the minimum number of generations of search nodes was expanded; however, if the motion planning algorithm were required to build a trajectory around a static obstacle or around a curve in the road that forced it

to expand more than the minimum number of generations of search nodes, the latter generations would have had no associated temporal layer from which they could have been evaluated. The solution utilized during the presented research simply evaluated additional search node generations by using the last available temporal layer. A possible approach could have involved including a few additional temporal layers that included additional predictions to account for possible extraneous generations; however, this would still have resulted in the construction of the temporal potentially being sub-optimal. A more enlightened solution could have involved incrementally adding new layers as they became necessary.

The functionality of the TGC component essentially took the place of the SARB component that fused the multiple traversability grids coming from the various sensor components on the vehicle. The TGC component received the output traversability grid coming from the SARB as one of its inputs. It then copied the traversability data contained within this structure into the newly created temporal grid structure and amended the subsequent temporal layers as the predicted positions of any moving objects were calculated. To streamline the grid creation, the functionality of the SARB could have been consolidated into the TGC so that the temporal grid structure could have been created without the necessity of the traditional traversability grid. This would have involved the TGC obtaining the various sensor traversability grids and running the fusion algorithm, modifying cells as necessary resulting from the motion prediction of the obstacles, and finally, including this fused and modified data into the temporal grid structure.

Another area of future work considered testing the PTMP method on additional and more complex operating behaviors exhibited by the Urban NaviGator. This new method also had applications for scenarios such as intersection navigation with moving traffic. In this situation,

the robot may have been attempting to traverse through an intersection to merge onto a lane where the resident traffic vehicles did not have to stop, or vice versa, if the robot was attempting to turn off of such a lane onto a side street. The ability to represent the predicted positions of the other traffic vehicles could have allowed the PTMP to determine when it was safe to begin accelerating to merge into the desired lane.

Another behavior involved passing a vehicle that was in the desired lane of travel and that was moving slower than the desired velocity of the robot. The Urban NaviGator was able to pass static vehicles that were located in its lane; however, it was not capable of accomplishing this task of passing a slowly moving vehicle. The new PTMP method could have aided in the behavior by allowing the vehicle to generate a trajectory that would have considered the predicted future positions of the obstacle vehicle as it was traversing the desired lane. Upon determination that the vehicle needed to be passed, the robot would have shifted to an adjacent lane to overtake the other vehicle. As it was passing the vehicle, the predicted positions of the vehicle would have been displayed in the temporal grid so that the new motion planning algorithm could have determined a trajectory that would have safely navigated the robot back into the original lane in front of the obstacle vehicle. This would have required additional intelligence on the part of the PTMP as it would have needed to not only plan motion but velocity as well to overtake the vehicle and would have needed to monitor safe passing distances between the robot and the other vehicle.

A final area of potential future work involves adding intelligence to the new temporal motion planner. The incorporation of moving obstacle prediction allowed the algorithm to view if an obstacle was likely to pass in front of its desired path. In the presented cases, the planner was able to generate its solution trajectories without being affected by the predicted positions of

the obstacles. However, if it was determined that the obstacle would adversely affect the desired trajectory, it would be advantageous for the planner to intelligently choose whether to alter its desired path to traverse in front of the obstacle or to travel behind the object. Likewise, the motion planner could be extended to plan a velocity trajectory to either speed the vehicle up to pass before the object or slow down to let the object pass first and then follow.

Conclusions

The robot motion planning task is not trivial by any means, even after decades of research and development. The ability of a robot to generate a trajectory to move it from its current state to its goal state without colliding with any objects is of paramount importance to accomplishing its mission. The introduction of moving obstacles into the robot's environment only further complicates this problem. Situations such as those that arose during the 2007 DARPA Urban Challenge, in which unmanned ground vehicles were required to navigate urban environments while interacting with other moving vehicles, including other unmanned vehicles, are prime examples of the complex nature of the art of robotic motion planning. The research presented in this dissertation sought to provide a new and novel approach to robotic motion planning in dynamic environments. The first chapter provides an introduction to the motion planning problem in general, and discusses the inclusion of dynamic objects in the environment, while Chapter 2 recounts previous work that sought to address the issue of planning trajectories in dynamic environments. This is followed by a discussion of the theory behind the newly presented temporal motion planning method in Chapter 3 and an outline of the current implementation of the method on the Urban NaviGator in Chapter 4. A description of the validation procedure and summary of testing results is then provided in Chapter 5.

The new PTMP method sought to incorporate motion prediction for all moving obstacles in the robot's immediate environment in a way that aided the algorithm in generating a control

input sequence that allowed the robot to safely navigate to its goal. This was facilitated by the development and implementation of a new grid structure that included temporal layers that represented how the environment changed at distinct future time-steps. A prediction algorithm generated a model to estimate the future positions of any obstacles present, and these estimated future positions were represented in their respective layers of the temporal grid structure to provide this sense of how the environment changes. The temporal grid was then used by a motion planning algorithm to intelligently plan a trajectory through the robot's dynamic environment by considering the estimates of how the objects in the environment moved over time.

The concept of a predictive temporal grid was a particularly novel outcome of the presented research, and the test results described in Chapter 5 provided ample evidence that this new predictive temporal method was capable of improving motion planning not only for unmanned ground vehicles, but for other types of unmanned vehicles and other types of robotic systems in general, including industrial robotic manipulators. The new method, combined with some of the areas of future research detailed in the previous section also opened up new potential applications and could provide solutions to existing problems for robotic systems in dynamic environments.

LIST OF REFERENCES

Arbuckle, D., Howard, A., & Mataric, M. (2002, September). Temporal occupancy grids: A method for classifying the spatio-temporal properties of the environment. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Lausanne, Switzerland (pp. 409-414).

Armstrong, M. P. (1988, November). Temporality in spatial databases. In Proceedings of the GIS/LIS'88, San Antonio, Texas, USA (pp. 880-889).

Arras, K. O., Persson, J., Tomatis, N., & Siegwart, R. (2002, May). Real-time obstacle avoidance for polygonal robots with a reduced dynamic window. In Proceedings of the IEEE International Conference on Robotics and Automation, Washington, D.C. (pp. 3050-3055).

Belghith, K., Kabanza, F., Hartman, L., & Nkambou, R. (2006, May). Anytime dynamic path-planning with flexible probabilistic roadmaps. In Proceedings of the IEEE International Conference on Robotics and Automation, Orlando, FL (pp. 2372-2377).

Biber, P., & Duckett, T. (2005, June). Dynamic maps for long-term operation of mobile service robots. In Proceedings of the Robotics: Science and Systems Conference, Cambridge, MA (pp. 17-24).

Cao, Q., Huang, Y., & Zhou, J. (2006, October). An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Beijing, China (pp. 3331-3336).

Chang, C. C., & Song, K.-T. (1996, May). Dynamic motion planning based on real-time obstacle prediction. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN (pp. 2402-2407).

Chang, C. C., & Song, K.-T. (1997). Environment prediction for a mobile robot in a dynamic environment. IEEE Transactions on Robotics and Automation, 13(6), 862-872.

Coue, C., Fraichard, T., Bessiere, P., & Mazer, E. (2003, September). Using Bayesian programming for multi-sensor multi-target tracking in automotive applications. In Proceedings of the IEEE International Conference on Robotics and Automation, Tapei, Taiwan (pp. 2104-2109).

Crane, C. D., Armstrong, D. G., Ahmed, M., Solanki, S., MacArthur, D., Zawodny, E., Gray, S., Petroff, T., Griffis, M., & Evans, C. (2005, March). Development of an integrated sensor system for obstacle detection and terrain evaluation for application to unmanned ground vehicles. In Proceedings of the SPIE - The International Society for Optical Engineering - Unmanned Ground Vehicle Technology VII, Orlando, FL (pp. 156-165).

Croft, E. A., Fenton, R. G., & Benhabib, B. (1998). On-line robot planning strategy for target interception. Journal of Robotic Systems, 15(2), 97-114.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. Computer, 22(6), 46-57.

Elnagar, A., & Gupta, K. (1998). Motion prediction of moving objects based on autoregressive model. IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans., 28(6), 803-810.

Elnagar, A., & Hussein, A. M. (2003, September). An adaptive motion prediction model for trajectory planner systems. In Proceedings of the IEEE International Conference on Robotics and Automation, Tapei, Taiwan (pp. 2442-2447).

Ferguson, D., & Stentz, A. (2005, April). The delayed D* algorithm for efficient path replanning. In Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain (pp. 2045-2050).

Ferguson, D., & Stentz, A. (2007). Field D*: An interpolation-based path planner and replanner. Robotics Research, 28(239-253.

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. International Journal of Robotics Research, 17(7), 760-772.

Foka, A. F., & Trahanias, P. E. (2002, September). Predictive autonomous robot navigation. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, (pp. 490-495).

Fox, D., Burgard, W., & Thrun, S. (1996, November). Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Osaka, Japan (pp. 1280-1287).

Fox, D., Burgard, W., Thrun, S., & Cremers, A. B. (1998, May). Hybrid collision avoidance method for mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Lueven, Belgium (pp. 1238-1243).

Fraichard, T., & Laugier, C. (1992, May). Kinodynamic planning in a structured and time-varying 2D workspace. In Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France (pp. 1500-1505).

Fraichard, T., & Laugier, C. (1993, May). Path-velocity decomposition revisited and applied to dynamic trajectory planning. In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA (pp. 40-45).

Fujimura, K., & Samet, H. (1989). A hierarchical strategy for path planning among moving obstacles [mobile robot]. IEEE Transactions on Robotics and Automation, 5(1), 61-69.

Galluzzo, T. (2006). Simultaneous planning and control for autonomous ground vehicles. Ph.D. thesis, University of Florida, Gainesville, FL.

Ge, S. S., & Cui, Y. J. (2002). Dynamic motion planning for mobile robots using potential field method. Autonomous Robots, 13(3), 207-222.

GNU, "GNU Scientific Library ", 2008.

Grinstead, C., & Snell, J. (1997). Introduction to probability. Providence, R.I.: American Mathematical Society.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107.

Hatayama, M., & Matsuno, F. (2008, August). Temporal GIS for information collection system using robot technology in a damaged building. In Proceedings of the SICE Annual Conference, Tokyo, Japan (pp. 1653-1656).

Houshangi, N. (1990, May). Control of a robotic manipulator to grasp a moving target using vision. In Proceedings of the IEEE Conference on Robotics and Automation, Cincinnati, OH (pp. 604-609).

Hsu, D., Kindel, R., Latombe, J.-C., & Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. International Journal of Robotics Research, 21(3), 233-255.

Hujic, D., Zak, G., Croft, E., Fenton, R. G., Mills, J. K., & Benhabib, B. (1995, August). Active prediction, planning and execution system for interception of moving objects. In Proceedings of the IEEE International Symposium on Assembly and Task Planning, Pittsburgh, PA (pp. 347-352).

Hwang, K.-S., & Ju, M.-Y. (1999, October). Automatic generation of a collision free speed profile for the maneuvering motion. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Tokyo, Japan (pp. 708-713).

Hwang, K.-S., & Ju, M.-Y. (2002). Speed planning for a maneuvering motion. Journal of Intelligent and Robotic Systems: Theory and Applications, 33(1), 25-44.

Ishikawa, K., Meguro, J.-I., Amano, Y., Hashizume, T., Takiguchi, J.-I., Kurosaki, R., & Hatayama, M. (2005, June). Parking-vehicles recognition using spatial temporal data (a study of mobile robot surveillance system using spatial temporal GIS part 2). In Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics, Kobe, Japan (pp. 151-157).

Jaillet, L., & Simeon, T. (2004, September). A PRM-based motion planner for dynamically changing environments. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Sendai, Japan (pp. 1606-1611).

Kant, K., & Zucker, S. (1988, April). Planning collision-free trajectories in time-varying environments: A two-level hierarchy. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA (pp. 304-313).

Kant, K., & Zucker, S. W. (1986). Toward efficient trajectory planning: The path-velocity decomposition. International Journal of Robotics Research, 5(3), 72-89.

Kehtarnavaz, N., & Li, S. (1988, June). A collision-free navigation scheme in the presence of moving obstacles. In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI (pp. 808-813).

Kent, D. (2007). Storing and predicting dynamic attributes in a world model knowledge store. Ph.D. thesis, University of Florida, Gainesville, FL.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research, 5(1), 90-98.

Kindel, R., Hsu, D., Latombe, J.-C., & Rock, S. (2000, April). Kinodynamic motion planning amidst moving obstacles. In Proceedings of the IEEE International Conference on Robotics and Automation, San Fransisco, CA (pp. 537-543).

Kluge, B. (2003, October). Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV (pp. 376-381).

Ko, N. Y., & Lee, B. H. (1996, November). Avoidability measure in moving obstacle avoidance problem and its use for robot motion planning. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Osaka, Japan (pp. 1296-1303).

Koenig, S., & Likhachev, M. (2002, July). D* lite. In Proceedings of the National Conference on Artificial Intelligence, Edmonton, Alberta, Canada (pp. 476-483).

Kunwar, F., & Benhabib, B. (2006). Rendezvous-guidance trajectory planning for robotic dynamic obstacle avoidance and interception. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 36(6), 1432-1441.

Langran, G. (1990). Temporal GIS design tradeoffs. Journal of the Urban and Regional Information Systems Association, 2(1), 16-25.

Langran, G., & Chrisman, N. R. (1988). A framework for temporal geographic information. Cartographica: The International Journal for Geographic Information and Geovisualization, 25(3), 1-14.

Large, F., Laugier, C., & Shiller, Z. (2005). Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles. Autonomous Robots, 19(2), 159-171.

Laugier, C., Petti, S., Vasquez, D., Yguel, M., Fraichard, T., & Aycard, O. (2005). Steps toward safe navigation in open and dynamic environments. In Springer Tracts in Advanced Robotics: Autonomous Navigation in Dynamic Environments. (pp. 45-82). Berlin / Heidelberg, Germany: Springer.

Leonard, J. J., & Durrant-Whyte, H. F. (1990). Simultaneous map building and localization for an autonomous mobile robot (Technical Memorandum No. 103274). Washington, DC: NASA.

Li, Y., Li, C., & Song, R. (2008, December). A new hybrid algorithm of dynamic obstacle avoidance based on dynamic rolling planning and RBFNN. In Proceedings of the IEEE International Conference on Robotics and Biomimetics, Bangkok, Thailand (pp. 2064-2068).

Longley, P., Goodchild, M. F., Maguire, D. J., & Rhind, D. W. (2001). Geographical information systems and science. New York, NY: John Wiley & Sons, Ltd.

Mitsou, N. C., & Tzafestas, C. S. (2007, June). Temporal occupancy grid for mobile robot dynamic environment mapping. In Proceedings of the Mediterranean Conference on Control and Automation, Athens, Greece (pp. 4433892).

Nilsson, N. (1998). Artificial Intelligence: A New Synthesis. San Francisco: Morgan Kaufmann Publishers, Inc.

Ogren, P., & Leonard, N. E. (2002, October). A tractable convergent dynamic window approach to obstacle avoidance. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Lausanne, Switzerland (pp. 595-600).

Pereira, G. A. S., Campos, M. F. M., & Aguirre, L. A. (2000, October). Improved control of visually observed robotic agents based on autoregressive model prediction. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Takamatsu, Japan (pp. 608-613).

Poty, A., Melchior, P., & Oustaloup, A. (2004, August). Dynamic path planning by fractional potential. In Proceedings of the IEEE International Conference on Computational Cybernetics, Vienna, Austria (pp. 365-371).

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. IEEE ASSP Magazine, 3(1), 4-16.

Rennekamp, T., Homeier, K., & Kroeger, T. (2006, October). Distributed sensing and prediction of obstacle motions for mobile robot motion planning. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Beijing, China (pp. 4833-4838).

Robert, J., & Sharf, I. (2007, September). Autonomous capture of free-floating objects using predictive approach. In Proceedings of the International Astronautical Federation - 58th International Astronautical Congress, Hyderabad, India (pp. 4160-4174).

Rohrmuller, F., Althoff, M., Wollherr, D., & Buss, M. (2008, September). Probabilistic mapping of dynamic obstacles using Markov chains for replanning in dynamic environments. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Nice, France (pp. 2504-2510).

Rude, M. (1997). Collision avoidance by using space-time representations of motion processes. Autonomous Robots, 4(1), 101-119.

Seder, M., Macek, K., & Petrovic, I. (2005, November). An integrated approach to real-time mobile robot control in partially known indoor environments. In Proceedings of the Annual Conference of the IEEE Industrial Electronics Society, Raleigh, NC (pp. 1785-1790).

Seder, M., & Petrovic, I. (2007, April). Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy (pp. 1986-1991).

Shiller, Z., Large, F., & Sekhavat, S. (2001, May). Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Korea (pp. 3716-3721).

Song, K.-T., & Chang, C. C. (1999). Reactive navigation in dynamic environment using a multisensor predictor. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 29(6), 870-880.

Stentz, A. (1995, August). The focussed D* algorithm for real-time replanning. In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, Quebec, Canada (pp. 1652-1659).

Stentz, A. (1994, May). Optimal and efficient path planning for partially-known environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA (pp. 3310-3317).

Svestka, P., & Overmars, M. (1994). Motion planning for car-like robots using a probabilistic learning approach (Technical Report UU-CS-1994-33). Utrecht, the Netherlands: Utrecht University.

Svestka, P., & Overmars, M. H. (1995, May). Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In Proceedings of the IEEE International Conference on Robotics and Automation, Nagoya, Aichi, Japan (pp. 1631-1636).

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. Cambridge, MA: MIT Press.

van den Berg, J. P., & Overmars, M. H. (2005). Roadmap-based motion planning in dynamic environments. IEEE Transactions on Robotics, 21(5), 885-897.

Worboys, M. F. (1994). Unified model for spatial and temporal information. The Computer Journal, 37(1), 26-34.

Yu, H., & Su, T. (2003). Destination driven motion planning via obstacle motion prediction and multi-state path repair. Journal of Intelligent and Robotic Systems: Theory and Applications, 36(2), 149-173.

Yung, N. H. C., & Ye, C. (1998, October). Avoidance of moving obstacles through behavior fusion and motion prediction. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA (pp. 3424-3429).

Zhu, Q. (1991). Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. IEEE Transactions on Robotics and Automation, 7(3), 390-397.

Zhu, Q. (1990, August). A stochastic algorithm for obstacle motion prediction in visual guidance of robot motion. In Proceedings of the IEEE International Conference on Systems Engineering, Pittsburgh, PA (pp. 216-219).

Zhuang, H.-Z., Du, S.-X., & Wu, T.-J. (2006). On-line real-time path planning of mobile robots in dynamic uncertain environment. Journal of Zhejiang University: Science, 7(4), 516-524.

BIOGRAPHICAL SKETCH

Eric Thorn was born in 1981 and raised in Pensacola, FL. He received B.S. degrees in both aerospace engineering and mechanical engineering from the University of Florida in Gainesville, Florida in December, 2004. He also received an M.S. in mechanical engineering from the University of Florida in August, 2007. He is currently working as a Graduate Research Assistant and completing a Doctoral degree in mechanical engineering at the Center for Intelligent Machines and Robotics (CIMAR) at the University of Florida. His research focuses on unmanned ground vehicle motion planning and control. He plans to continue his career as a research engineer in the Department of Intelligent Systems at the Southwest Research Institute in San Antonio, Texas.