



Calhoun: The NPS Institutional Archive
DSpace Repository

NPS Scholarship

Theses

2006-12

Rapid motion planning and autonomous obstacle avoidance for unmanned vehicles

Lewis, Laird-Philip Ryan

Monterey, CA; Naval Postgraduate School

<https://hdl.handle.net/10945/2460>

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**RAPID MOTION PLANNING AND AUTONOMOUS
OBSTACLE AVOIDANCE FOR UNMANNED VEHICLES**

by

Laird-Philip Ryan Lewis

December 2006

Thesis Advisor:
Second Reader:

I. Michael Ross
Wei Kang

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 2006	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE Rapid Motion Planning and Autonomous Obstacle Avoidance for Unmanned Vehicles		5. FUNDING NUMBERS	
6. AUTHOR(S) Laird-Philip Ryan Lewis		8. PERFORMING ORGANIZATION REPORT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		12b. DISTRIBUTION CODE	
13. ABSTRACT This work introduces the use of optimal control methods for path planning and control of autonomous vehicles in an obstacle-rich environment. Traditional techniques harbor non-optimal, closed architectures primarily derived at a time when computational complexity could significantly hinder overall system performance. Advancements in computing power, miniaturization, and numerical methods permit the utilization of online, optimal path planning and control, thereby improving system flexibility and autonomy. The backbone of this concept is state-of-the-art optimal control techniques involving pseudospectral methods and sequential quadratic programming. Although this research focuses on a robotic car or Unmanned Ground Vehicle (UGV), several systems, including an Unmanned Aerial Vehicle (UAV) and a pendulum on a rotational base, are detailed to illustrating the technique's modularity. With respect to the UGV, optimal control methods permit the optimization of maneuver parameters while accounting for complex vehicle kinematics and workspace obstacles, represented as dynamic and path constraints respectively. The path constraints are modeled such that an obstacle of any shape or size can be included. Maneuvering trajectories are first generated in an open-loop architecture, followed by an application of these same techniques in feedback form. Lastly, model fidelity is increased to improve control over vehicle behavior and closed-loop performance and a local knowledge scenario is evaluated.			
14. SUBJECT TERMS Optimal Control, Trajectory Optimization, Path Planning, Autonomous Ground Vehicles, Real-Time Optimal Control, DIDO, Pseudospectral Method Application			15. NUMBER OF PAGES 161
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**RAPID MOTION PLANNING AND AUTONOMOUS OBSTACLE AVOIDANCE
FOR UNMANNED VEHICLES**

Laird-Philip R. Lewis
Lieutenant, United States Navy
B.S. in Aerospace Engineering, Virginia Tech, 2001

Submitted in partial fulfillment of the
requirements for the degrees of

**ASTRONAUTICAL ENGINEER
AND
MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL
December 2006**

Author: Laird-Philip Ryan Lewis

Approved by: I. Michael Ross
Thesis Advisor

Wei Kang
Second Reader

Anthony J. Healey
Chairman, Department of Mechanical and Astronautical
Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This work introduces the use of optimal control methods for path planning and control of autonomous vehicles in an obstacle-rich environment. Traditional techniques harbor non-optimal, closed architectures primarily derived at a time when computational complexity could significantly hinder overall system performance. Advancements in computing power, miniaturization, and numerical methods permit the utilization of online, optimal path planning and control, thereby improving system flexibility and autonomy. The backbone of this concept is state-of-the-art optimal control techniques involving pseudospectral methods and sequential quadratic programming. Although this research focuses on a robotic car or Unmanned Ground Vehicle (UGV), several systems, including an Unmanned Aerial Vehicle (UAV) and a pendulum on a rotational base, are detailed for the purpose of illustrating the technique's modularity. With respect to the UGV, optimal control methods permit the optimization of maneuver parameters while accounting for complex vehicle kinematics and workspace obstacles, represented as dynamic and path constraints respectively. The path constraints are modeled such that an obstacle of any shape or size can be included. Maneuvering trajectories are first generated in an open-loop architecture, followed by an application of these same techniques in feedback form. Lastly, model fidelity is increased to improve control over vehicle behavior and closed-loop performance and a local knowledge scenario is evaluated.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	PROBLEM DEFINITION.....	3
	A. MOTIVATION.....	3
	B. CONSTRAINTS AND KINEMATICS.....	5
	C. PREVIOUS APPROACHES.....	8
	1. Overview.....	8
	2. Background.....	9
	3. Survey of Previous Methods.....	10
	D. DIDO.....	17
	E. THE OPTIMAL CONTROL PROBLEM.....	17
	1. Path Constraints.....	18
	2. Scaling and Balancing.....	22
	3. Verification and Validation.....	23
III.	TRICYCLE PROBLEM.....	29
	A. MOTIVATION.....	29
	B. PROBLEM DEFINITION.....	29
	C. SOLUTION VALIDATION.....	30
	D. RESULTS.....	36
IV.	FOUR-WHEELED CAR.....	43
	A. OPEN LOOP.....	43
	1. Validation.....	43
	2. Results.....	51
	<i>a. Obstacle Rich Environments.....</i>	<i>51</i>
	<i>b. Parallel Parking.....</i>	<i>52</i>
	<i>c. Moving Obstacles.....</i>	<i>56</i>
	<i>d. Moving Target.....</i>	<i>57</i>
	<i>e. Actuator Failures.....</i>	<i>61</i>
	B. CLOSED LOOP.....	62
	1. Motivation.....	62
	2. Problems with Closing the Loop.....	62
	3. A New Problem Definition.....	63
	4. Solution Approaches.....	64
	5. Results.....	73
	<i>a. Penalty Function.....</i>	<i>73</i>
	<i>b. Resolving Uncertainty Approach.....</i>	<i>87</i>
	6. Improving Model Fidelity.....	89
	7. Robustness through Local Knowledge.....	95
	8. Computation Time.....	104
	C. CONCLUSIONS.....	107
V.	UNMANNED AERIAL VEHICLE (UAV).....	111
	A. MOTIVATION.....	111

B.	PROBLEM DEFINITION	111
C.	SOLUTION VALIDATION AND RESULTS.....	115
D.	CONCLUSIONS	120
VI.	INVERTED PENDULUM	121
A.	MOTIVATION	121
B.	PROBLEM DEFINITION	122
C.	NECESSARY CONDITIONS	125
D.	RESULTS	126
E.	CONCLUSIONS	133
VII.	CONCLUSIONS	135
A.	RESEARCH CONCLUSIONS.....	135
B.	FUTURE WORK.....	136
	LIST OF REFERENCES	137
	INITIAL DISTRIBUTION LIST	143

LIST OF FIGURES

Figure 1	Mars Exploration Rovers (From Ref. 3).....	4
Figure 2	A Car Model with Front-Wheel Steering.....	6
Figure 3	Two Link System in Work Space (left) and Configuration Space (right).....	9
Figure 4	Unit P-Norms for $P = 1, 2,$ and Infinity Respectively.	19
Figure 5	Simple Geometric Shape Constructed by Three Lines.	20
Figure 6	Constraint Representation of a Simple Polygonal Shape.	21
Figure 7	Smoothed Polygonal Constraint.	22
Figure 8	Optimal Tricycle Kinematic Positional Trajectory (left) and Angular Trajectory (right).....	31
Figure 9	Scaled Control Trajectories for Validation Problem.	32
Figure 10	Costate Trajectories for Validation Problem.	32
Figure 11	Covectors for Control (left) and Path (right) for Validation Problem.	33
Figure 12	Plot of the Hamiltonian Minimization Condition Equations for the Validation Problem.	33
Figure 13	Plot of the Value of the Hamiltonian for the Validation Problem.	34
Figure 14	Propagated versus DIDO Generated Positional (left) and Angular (right) Trajectories.	34
Figure 15	Bellman’s Principle Test for Validation Problem.....	35
Figure 16	Trajectory Cost for Continuous and Segmented Paths.	36
Figure 17	Tricycle Optimal Path in Square Obstacle Environment.....	37
Figure 18	Adjusted Tricycle Optimal Path in Square Obstacle Environment (0.1m Obstacle Extension).	38
Figure 19	Further Adjusted Tricycle Optimal Path in Square Obstacle Environment (0.25m Obstacle Extension).....	38
Figure 20	Optimal Tricycle Trajectory around Polygonal Shape.	40
Figure 21	Low-node Pre-computational Step for Maze Problem.	41
Figure 22	Optimal Tricycle Trajectory through a Maze Problem.....	42
Figure 23	Car Optimal Control Validation Problem.....	43
Figure 24	Angular Trajectory for Car Validation Problem.....	44
Figure 25	Control Trajectory for the Car Validation Problem.....	45
Figure 26	Control Hamiltonian Value for the Car Validation Problem.....	46
Figure 27	Hamiltonian Minimization Condition Verification for the Car Validation Problem.	46
Figure 28	Costate Values for the Car Validation Problem.....	47
Figure 29	Path Constraint Covectors for the Car Validation Problem.....	47
Figure 30	Control Covectors for the Car Validation Problem.	48
Figure 31	DIDO Generated and Propagated Positional Trajectories.	49
Figure 32	DIDO Generated and Propagated Angular Trajectories.	49
Figure 33	Verification of Bellman’s Principle with Three Optimal Sub-trajectories.	50
Figure 34	Verification of Bellman’s Principle by Cost Comparison.	50
Figure 35	Time-Optimal Trajectory through 32 Obstacle Field.	51
Figure 36	Optimal Trajectory through Complex Ellipse Environment.....	52
Figure 37	Time-Optimal Parallel Parking Maneuver; Location is Behind the Car.....	53

Figure 38	Two-Dimensional View of Backward Parallel Parking Maneuver.	54
Figure 39	Time-Optimal Parallel Parking Maneuver; Location is in Front of the Car.	55
Figure 40	Two-Dimensional View of Forward Parallel Parking Maneuver.	55
Figure 41	Time-Optimal Trajectory with Moving Obstacles.	56
Figure 42	Time-Optimal Trajectory with Moving Obstacles, Strobe Picture.	57
Figure 43	Time-Optimal Trajectory for Moving Target.	58
Figure 44	Time-Optimal Trajectory for Moving Target, Strobe Picture.	59
Figure 45	Second, Time-Optimal Trajectory with Moving Target.	60
Figure 46	Second, Time-Optimal Trajectory with Moving Target, Strobe Picture.	60
Figure 47	Optimal Trajectory during Control Failure, Must Turn Left.	61
Figure 48	Block Diagram of the Basic Closed-loop Optimal Control Architecture.	64
Figure 49	Contour Plot of Penalty Function for a Simple Three-Obstacle Map.	67
Figure 50	Surface Plot of Penalty Function for a Simple Three-Obstacle Map.	67
Figure 51	The Maze Problem First Introduced in the Tricycle Section.	68
Figure 52	Maze Problem Visualized Through the Penalty Function Function.	69
Figure 53	The Maze Problem Represented with the Polygonal Shape Penalty Function.	70
Figure 54	Example of Range-Dependent Buffer around Obstacles.	72
Figure 55	Illustration of Relationship between Buffer Size and Distance to Obstacle.	73
Figure 56	Potential Function Trajectory when Maximum Time is Four Percent Larger than Optimal Time.	74
Figure 57	Potential Function Trajectory when Maximum Time is Five Percent Larger than Optimal Time.	75
Figure 58	A Maze Optimal Trajectory Generated from No Initial Guess.	76
Figure 59	A Maze Optimal Trajectory Generated from a Four Point Initial Guess.	77
Figure 60	Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Spline Interpolation.	78
Figure 61	Closed-Loop and Open-Loop Control Trajectory Comparison.	78
Figure 62	Description of How Calculation Time Affects Trajectory Traveled.	80
Figure 63	Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Shape- Preserving Piecewise Cubic Interpolation.	82
Figure 64	Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Shape-Preserving Piecewise Cubic Interpolation.	82
Figure 65	Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Pchip Interpolation and a Relaxed Maneuver Time.	83
Figure 66	Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Pchip Interpolation and a Relaxed Maneuver Time.	84
Figure 67	Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and 30-Node Solutions.	85
Figure 68	Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and 30-Node Solutions.	85
Figure 69	Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and Increased Update Frequency.	86
Figure 70	Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and Increased Update Frequency.	87

Figure 71	Resolving Uncertainty Closed-Loop Results, Compared to Initial Optimal Trajectory.....	88
Figure 72	Resolving Uncertainty Closed-Loop Control Trajectory, Compared to Initial Optimal Trajectory.	88
Figure 73	Closed-Loop Positional Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.0 Second Update Rate.	90
Figure 74	Closed-Loop Positional Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.5 Second Update Rate.	91
Figure 75	Closed-Loop Angular Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.5 Second Update Rate.	92
Figure 76	Closed-Loop Control Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.5 Second Update Rate.	92
Figure 77	Closed-Loop Positional Trajectory Using Penalty Function Approach, Improved Model, and 1.0 Second Update Rate.	93
Figure 78	Closed-Loop Angular Trajectory Using Penalty Function Approach, Improved Model, and 1.0 Second Update Rate.	94
Figure 79	Closed-Loop Control Trajectory Using Penalty Function Approach, Improved Model, and 1.0 Second Update Rate.	94
Figure 80	Resolving Uncertainty Method Results for the Local Knowledge Problem....	96
Figure 81	Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a 0.5 Second Update Rate.....	97
Figure 82	Orientation Trajectory for a Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a 0.5 Second Update Rate.	97
Figure 83	Control Trajectory for a Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a 0.5 Second Update Rate.	98
Figure 84	Local Knowledge Problem Solved by the Penalty Function Method.....	99
Figure 85	Orientation Trajectory for a Local Knowledge Problem Solved by the Penalty Function Method.....	99
Figure 86	Control Trajectory for a Local Knowledge Problem Solved by the Penalty Function Method.....	100
Figure 87	Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a Five-DOF State Vector.	101
Figure 88	Orientation Trajectory of a Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a Five-DOF State Vector.....	101
Figure 89	Control Trajectory of Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a Five-DOF State Vector.....	102
Figure 90	Local Knowledge Problem Solved by the Penalty Function Method and Using a Five-DOF State Vector.....	103
Figure 91	Orientation Trajectory of a Local Knowledge Problem Solved by the Penalty Function Method and Using a Five-DOF State Vector.	103
Figure 92	Control Trajectory of a Local Knowledge Problem Solved by the Penalty Function Method and Using a Five-DOF State Vector.	104
Figure 93	Histogram of Computation Times for Global Knowledge Problem Solved with Penalty Function Method and Three-DOF State Vector.	106
Figure 94	Histogram of Computation Times for Global Knowledge Problem Solved with Resolving Uncertainty Method and Three-DOF State Vector.....	107

Figure 95	Illustration of the Dual Micro-Processor Concept.....	109
Figure 96	UAV Kinematic Diagram (modified from Ref. 50).....	112
Figure 97	UAV Kinematic Trajectory through City Environment.	115
Figure 98	Propagated UAV Trajectory and DIDO-Generated UAV Trajectory.	116
Figure 99	Control Trajectory for Optimal UAV Trajectory.....	117
Figure 100	Hamiltonian Value for Optimal UAV Trajectory.....	117
Figure 101	Hamiltonian Minimization Condition Verification for Optimal UAV Trajectory.....	118
Figure 102	Path Covectors for Optimal UAV Trajectory.	118
Figure 103	Control Covectors for Optimal UAV Trajectory.....	119
Figure 104	Costates for Optimal UAV Trajectory.....	120
Figure 105	ECP Model 205a Torsional Control System with A-51 Inverted Pendulum Assembly.....	121
Figure 106	Inverted Pendulum on a Rotating Base Model.	123
Figure 107	Pendulum and Disk Angular Trajectories for Optimal Inversion Maneuver....	127
Figure 108	Pendulum and Disk Angular Velocities for Optimal Inversion Maneuver....	127
Figure 109	Torque Trajectory for Optimal Inversion Maneuver.	128
Figure 110	Control Convector with Control Trajectory in the Background to Display Switching Nature.	129
Figure 111	Hamiltonian for Pendulum Inversion Problem.....	129
Figure 112	Pendulum Inversion Maneuver Costates.	130
Figure 113	Pendulum Inversion Maneuver Path Covector.	131
Figure 114	Pendulum Position Feasibility Verification.....	132
Figure 115	Disk Position Feasibility Verification.....	132
Figure 116	Images of Optimal Inversion Maneuver.	133

LIST OF TABLES

Table 1	Effect of Problem Formulation on Computation Time.....	39
Table 2	Mapping of Vehicle Position to Shape-Based Penalty Function Value.	66
Table 3	Mapping of Vehicle Position to Polygonal Shape Penalty Function Value. ...	70
Table 4	Computation Times (All Simulations Use Global Information).....	105
Table 5	Inverted Pendulum Parameters and Variables.	123

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

As I look back upon my accomplishments, my first thoughts are always of my wife. Gina was with me throughout, offering endless encouragement and support. I strongly doubt that I could have achieved success without her, and I am certain that success would not feel as sweet without her.

My son, Caidon, is growing up too fast. I am glad that I have had the opportunity, in the last two years, to know him a little better and be the father that I have wanted to be. I hope that I never make him read this document.

Dr. Ross, my thesis advisor, has given me a great opportunity to learn and advance my own research skills. He forced me to ask questions and find solutions on my own and look a little deeper at the world around. I appreciate him always giving me the right amount of guidance.

Lastly, the gentlemen of the Guidance, Navigation, and Control Lab bear a considerable amount of credit for the completion of this thesis. Pooya Sekhavat, Qi Gong, Kevin Bollino, and Joe Dituri have all been quite amenable to my loud music and constant badgering about DIDO and theoretical questions.

Thanks to everyone who participated in my educational process.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Autonomy has long been a goal for numerous scientists, engineers, military planners, and science fiction enthusiasts, and in spite of their best efforts, many advances remain out of reach. The word often found side-by-side with autonomy is robotics. Robots have mastered specialized, repetitive tasking, and in this role they have assisted humans by providing information, accomplishing tedious work, and performing dangerous jobs. Yet, despite the advances in robotics, the transfer of these technologies into the commercial domain remains minimal. There are three reasons for this long gestational period. First, these technologies are prohibitively expensive. State-of-art robots are found in universities and government laboratories primarily because they are the only ones who can afford them. Second, they are complex. Of course, this fact feeds the first issue since difficulty is often related to price, but complexity also refers to the fact that it is difficult to program a robot to accomplish even ten repeatable tasks safely. Thus leading to the third and final reason why robot technology has not proliferated the commercial market; they are not overly useful. Unlike cartoon depictions of robotic maids that can cook you eggs while they wax your car and shine your shoes, the \$300 Roomba® can only vacuum a room. The key is that true functionality is not possible without autonomy.

Most scientists and market experts agree that it is only a matter of time before robot and autonomy technologies advance to the point where their products can be truly useful at a reasonable price. This will be accomplished by software packages and systems that can have a multi-functional portability and a dynamic adaptability. While there are several areas within the robotics discipline that require attention, the focus of this work is the area of path planning and motion control. The intent of this research is to show how optimal control methods can accomplish the path planning and motion control problems in unmanned or robotic systems and simultaneously improve system autonomy. It will be shown that these techniques hold the potential for mathematically optimal solutions while possessing the ability to account for dynamic and kinematic constraints, a characteristic that other techniques cannot demonstrate. Previous stereotypes of optimization techniques are that they are too computationally intensive to be applied in

an online manner, but this work evidences the contrary. In addition, optimal control methods enjoy an inherent portability; they are rapidly adaptable to new obstacle environments, robot conditions, and even entirely different dynamical systems. This means that one system can be used to drive a car down the highway, maneuver through a parking lot, and eventually parallel park, and, incidentally, it can also maneuver the car safely in the event of a flat tire. Optimal control methods provide functionality through autonomy.

This thesis is organized into six subsequent sections. The Section II details the problem definition including, what problem is solved, motivation, problem characteristics, problem formulation, and how it was solved in the past. Section III begins the analysis and results portion of the thesis by testing the optimal control techniques on a tricycle model. Section IV is the heart of this work, and it pertains to a four-wheeled car model, i.e. unmanned ground vehicle. Planning and control methods are simulated for this model in terms of open-loop and closed-loop architectures, demonstrating techniques used to handle uncertainty. Section V applies the same techniques to a three-dimensional Unmanned Aerial Vehicle (UAV) model and Section VI to an inverted pendulum. These examples illustrate the portability and capability of these techniques to handle challenging dynamics. Section VII provides future work and conclusions.

II. PROBLEM DEFINITION

A. MOTIVATION

The advances in modern computing, specifically relating to speed and cost, are leading to an infiltration of processing power into everyday life. Revolutionizing information flow, automating previously human tasks, and improving process performance are only a brief list of significant impacts that computers have made to date. A revolution in controls engineering and application is now on the horizon. Many of the control techniques studied and implemented today are the remnants of methods developed when electronic computers were cumbersome, expensive, and impractical for onboard application. Changing the method of control application will prove evolutionary for both military and civil applications.

The area of robotics relies heavily on a confluence of computing and controls among other disciplines. In the 1960's computer coordinated cameras and mechanical arms were incapable of reliably accomplishing tasks that were simple for a toddler. Forty years later Sony was able to successfully mass-produce and sell AIBO robot pets despite a price greater than \$1000 [1]. These small, dog-shaped robots incorporated an onboard computer and sensors thereby enabling the owner to operate their pet with a high-level language remotely or autonomously. Outside of private industry, many government organizations have staked a significant investment in robotics. Unmanned and autonomous vehicles are crucial to the National Aeronautics and Space Administration's (NASA) plans for the "Moon, Mars, and Beyond." The use of robotic missions on the Moon and Mars will help to provide enabling information while minimizing the risk and cost of eventual human exploration missions [2]. In 1997 the Mars rover Sojourner was constrained to remain near its lander; motion plans were generated offline on Earth and uploaded. The Mars Exploration Rovers (MERs), pictured in Figure 1 [3], were developed with greater autonomous capability, thereby enabling more valuable scientific and exploratory missions to be accomplished. Spirit and Opportunity continue to function far beyond their intended mission, at significant distances from their landing sites, and they provide data not obtainable by human endeavors now.



Figure 1 Mars Exploration Rovers (From Ref. 3)

The Department of Defense (DoD) also places an emphasis on the development of unmanned and autonomous vehicle technologies. Air, ground, surface, and underwater vehicles that are unmanned or autonomous present operational advantages to battlefield and unit commanders. These technologies remove soldiers from harmful missions while providing equal intelligence, reconnaissance, defense, and potentially offensive capabilities. The National Defense Authorization Act of 2001 states that the Armed Forces shall strive to field technologies permitting the employment of unmanned vehicles in greater numbers. The Act goes further to mention that by 2010 one-third of the operational deep strike combat aircraft be unmanned and by 2015 one-third of the ground combat vehicles be unmanned [4]. The purpose of the Defense Advanced Research Projects Agency's (DARPA) Grand Challenge, an autonomous ground vehicle competition, is to precipitate the systems and expertise necessary to field the unmanned vehicles needed in the military.

Autonomous vehicles and autonomous behaviors are enabled by numerous technology areas including but not limited to: mission planning and reasoning, sensing, mapping, recognition, path planning, trajectory following, and motion control. While all of these areas benefit from advances in computing technology, the focus of this research is in exploring the advancements in the fields of path planning and trajectory following. The methods demonstrated within have the capability to be implemented in the autonomous or unmanned vehicles of the future to provide greater autonomy with

optimum performance. Even when this technology is incorporated into an unmanned, remotely controlled vehicle such as a common UAV, greater vehicle autonomy facilitates less human oversight and involvement thereby permitting a reduction in operational costs.

Discussing these new methods of controlling vehicles, describing how these new control methods apply to autonomous vehicles, and demonstrating their ability to improve vehicle performance is the purpose of the remainder of this paper. Numerous vehicles and models can be extracted from previous research for this analysis, and true to this statement, several different models were evaluated and will be discussed. However, the main thrust of this thesis focuses on a car simulation. The rationale for utilizing the car lies in its direct relevance to civil and military applications and its relatively challenging dynamics.

B. CONSTRAINTS AND KINEMATICS

The four-wheeled car is one of the most well studied models with respect to kinematics and path planning; a logical research topic given its influence and profusion into human life. However, the car presents a challenge for engineers due to the nature of the constraints inherent to the system. Anyone who has attempted to parallel park is well aware of the fact that, for most cars, only the front wheels are capable of turning and the back wheels must roll without slipping. While this fact constrains the motion of a car, it does not necessarily inhibit the feasible states - position and orientation - that a car can achieve, and this is representative of a nonholonomic constraint. Holonomic constraints can be expressed solely as a function of the state variables and possibly time, and each linearly independent holonomic constraint reduces the number of degrees of freedom or number of variables needed to wholly describe a system. Nonholonomic constraints, on the other hand, cannot be expressed entirely as a function of the state variables and times; the constraint equation will include derivatives of the state variables. As it is the case with the car model, nonholonomic constraints do not lower the degrees of freedom of the system.

The kinematic model of the car used throughout this research is shown in Figure 2. The state vector is composed of two position variables and an orientation variable,

Equation (1). The x-y location of the car represents the current position of the center point of the rear axle. The car's orientation is measured with respect to the horizontal axis and is presented as the state variable theta.

$$\underline{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad (1)$$

The car's control variables, Equation (2), include velocity and phi, the angle of the steering wheels with respect to the car's heading.

$$\underline{u} = \begin{bmatrix} v \\ \phi \end{bmatrix} \quad \text{where} \quad \underline{u} \in \mathbf{U} = \left\{ \begin{array}{l} v: v_{\min} \leq v(t) \leq v_{\max} \\ \phi: \phi_{\min} \leq \phi(t) \leq \phi_{\max} \end{array} \right\} \quad (2)$$

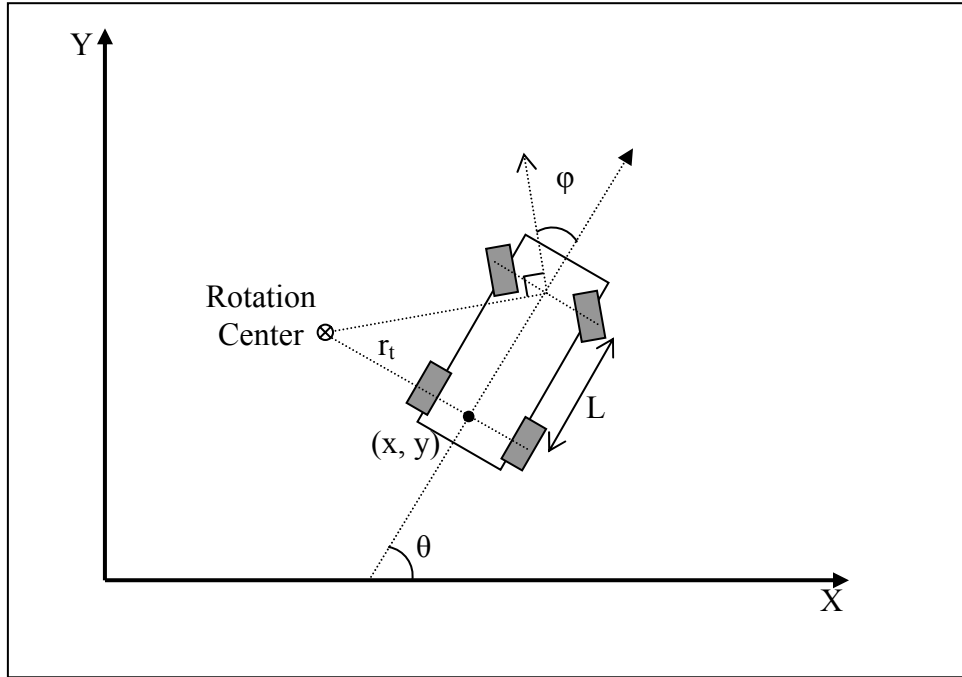


Figure 2 A Car Model with Front-Wheel Steering.

The constant 'L' defines the distance between the forward and rear axles, and the variable r_t reflects the instantaneous turning radius.

Significant items easily overlooked in Equation (2) are the limits placed upon the control variables. The angle of the forward wheels with respect to the car's heading, hence forth referred to as the steering angle, can not be greater than ±90° for both

computational and practical reasons. The maximal limit of the steering angle places a restriction on the minimum turning radius, which can be determined by Equation (3).

$$r_{t,\min} = \frac{L}{\tan \phi_{\max}} \quad (3)$$

Several variations of the steering angle and velocity limits were considered previously [5]. There are two well-known and explored variations. The Reeds-Shepp car demonstrates only three distinct speed capabilities – forward maximum, stop, reverse maximum – as displayed in Equation (4).

$$\underline{u} \in \mathbf{U} = \left\{ \begin{array}{l} v: \quad v = \{-1, 0, 1\} \\ \phi: \quad \phi_{\min} \leq \phi(t) \leq \phi_{\max} \end{array} \right\} \quad (4)$$

The Dubin's car is similar but lacks the capability to move in reverse, Equation (5).

$$\underline{u} \in \mathbf{U} = \left\{ \begin{array}{l} v: \quad v = \{0, 1\} \\ \phi: \quad \phi_{\min} \leq \phi(t) \leq \phi_{\max} \end{array} \right\} \quad (5)$$

These approaches are neglected in this analysis for the sake of reality, as most cars are capable of throttling velocity. In addition, the car is permitted to move in a backwards direction.

The nonholonomic constraint of rolling without slipping described previously is derived in [5] and presented in Equation (6).

$$-\dot{x} \sin \theta + \dot{y} \cos \theta = 0 \quad (6)$$

This constraint leads to the kinematic equations of motion appearing in Equation (7).

$$\dot{\underline{x}} = \begin{bmatrix} v \cos \theta \\ v \sin \theta \\ \frac{v}{L} \tan \phi \end{bmatrix} \quad (7)$$

This problem is simple in appearance, yet its non-linearity creates difficulties for many path planning methods. It is noteworthy that in the current configuration the controls can change instantaneously. This characteristic may produce results that are undesirable for a specific robot, and in that case the velocity could be included as a state variable and the acceleration becomes the control variable.

C. PREVIOUS APPROACHES

1. Overview

Path planning problems have been explored with great intensity for several decades. In general, path planning problems are under-determined and often non-linear in their truest form. For these reasons, planning methods fall into four generic categories including: completeness, optimality, computational complexity, and scalability [5-7]. Completeness refers to a method's ability to guarantee a successful solution if one does exist, and likewise report if no solution is possible. Optimality specifies the ability of a method to select a path that best fits a predetermined criterion. Computational complexity provides a relative value for the amount of time a method requires to determine a solution, and this tends to delineate the method's ability to be used in an online or offline manner. Scalability is a method's portability to different or more dimensionally complex systems. Permissible problem complexity is a characteristic implicit to completeness and optimality. In a simple example where a method is incapable of handling a non-linear system of equations, completeness and optimality designations for that method carry less weight due to the system's inaccuracy in problem representation.

In the past, real-life applications of path planning methods led to one last defining characteristic, the ability to function with local information or the restriction to global information. These terms have often experienced usage in parallel to the offline versus online classification, due to a relatively longer computation time. The distinctiveness of global and local methods has blurred with the progression of modern computing. If an offline, global information planner can run at a sufficient speed, it can be implemented in a feedback loop to provide an updated solution as new sensor information becomes available. This concept is a centerpiece of this research and the ability to utilize this technique is reliant equally upon the idiosyncrasies of the system controlled, the speed of the planner, and the speed of the computer.

Having defined the four general evaluation criteria for each of the planning methods, the purpose of the following sections is to provide an informational background to the language used in various planning methods and then provide a quick survey of the

current state of planning methods. The primary focus of this assessment is the application planning methods to the general car or Unmanned Ground Vehicle (UGV) problem defined before.

2. Background

Before any of the planning methods can be described in detail, it is necessary to specify some of the concepts which are fundamental to their use.

Many robot and manipulator problems are seen visually in a workspace but are more easily handled mathematically in a configuration space [6]. The difference between the two is most simply described by an example. For a two-link system, the workspace is the left-half of Figure 3 which shows the system as any person would see it in real life. In this case the objective is to place the end of the second link into the goal position without hitting the obstacle with any portion of the linkage system. The right-half of Figure 3 displays the configuration space representation of the same problem; it is the space of all possible states of the system. The two-link system is composed of many particles, which occupy a significant volume in the workspace; in contrast, the system's representation in configuration space requires one point. Any state or configuration that would cause the two-link system to contact the obstacle in the workspace is eliminated from the free configuration space.

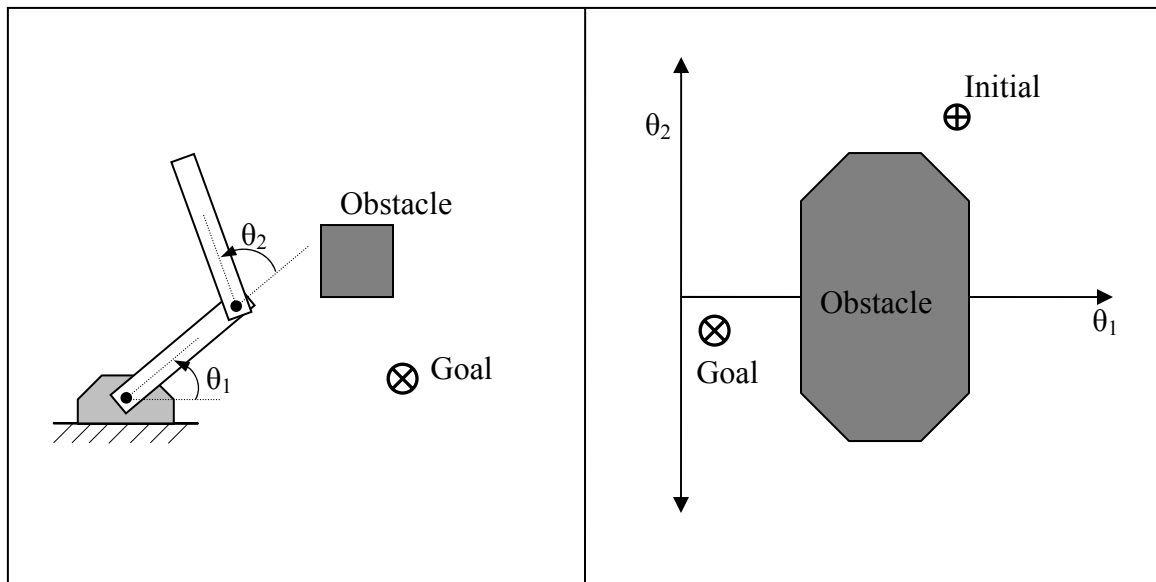


Figure 3 Two Link System in Work Space (left) and Configuration Space (right).

This is a straightforward example; however, the dimension of the configuration space increases with the number of degrees of freedom of the system studied. Also mapping obstacles from the workspace to the configuration space can become an intensive undertaking.

3. Survey of Previous Methods

Numerous planning methods have been developed and continue to evolve for the purposes of vehicle motion. Detailed development and analysis of the various methods are included in the listed references [5, 6, 7]. These methods include artificial potential fields and distance transformations, roadmaps, cell decompositions, sample-based algorithms, analytical approaches, and optimal control methods.

Artificial potential fields have stimulated interest for use in conjunction with the online, obstacle avoidance problem for over thirty years [8]. The fundamentals behind potential functions are simple; they only need to be real-valued and differentiable over a configuration space. Similar to any real-world potential field, the gradient of the potential can be thought of as a force. A more intuitive visualization tool is to picture the robot as a positively charged particle in configuration space. The potential functions provide forces such that obstacles emit positive charges and the goal emits a negative charge. Naturally, the particle is repelled from the obstacles and drawn to the goal, and mathematically it follows the path of the steepest gradient. With this simplistic approach, it is possible to plan the robot's path in an online, robust manner with local information.

This approach suffers from several drawbacks however. First, the particle can get stuck at a local minimum in the configuration space that does not correspond to the goal – a completeness problem. This concern requires addressing in each application of potential functions. Second, optimality criteria are difficult to satisfy, and third nonholonomic constraints and vehicle kinematics are difficult to accommodate. The first concern has been addressed in several ways. Reference [9] combines the potential field with a distance transform which is another potential field method based on mapping the space to a grid and determining the distance from every position to the goal. This method can provide near optimal results but tends to scale and handle nonholonomic constraints poorly. References [10, 11, 12] solve the problem by imposing random motion and an escape force during local minimum conditions. More generally [6, 7] describe the

possibility to develop a navigation functions that contain only one minimum, at the goal. However, these functions can only be created with global information, thereby sacrificing one of the method's benefits. Solution optimality, with respect to distance traveled, was improved in [12, 13] by placing tunable gains on the repulsive forces representing the obstacles, but this displayed only moderate improvement and portends a lack of portability particularly to other forms of optimality. Lastly, potential functions have been applied to nonholonomic systems and problems with motion constraints. Reference [14] attempts to apply the potential function method to multiple, nonholonomic vehicles but limits the research to a simplistic tricycle model. Reference [11] applies potential functions to the high-speed navigation of UGVs such that dynamic constraints – vehicle side slip and rollover – are satisfied.

In general, potential functions provide a valuable, robust, online path planning method, but it lacks the capability to produce an optimal result and vehicle constraints remain difficult to manage.

Roadmaps are most easily described by their name. When a driver is planning a long trip via a map, he or she will typically plan most of the trip over the highways. Then the only thing left to do is to find fast access to the highway from the starting location and from the highway to the finish. These three actions summarize the three actions involved in Roadmap planning. Routes or “highways” traverse the configuration or work space. The first task is to plan a route from a node near the start to a node near the finish. Once this is accomplished, the only remaining task is to plan a route from the start to the “highway” and from the “highway” to the finish. In their general form, Roadmaps require global information and in that case, they can generate a path that is optimal with respect to distance traveled [6]. Roadmaps are efficient once the roads are created, and if the environment is static, the same roadmap could be used throughout a path. In the case of local information, the obstacles are not known a priori, and Roadmap implementation is more challenging. Several approaches enable the formation of roads in the case of local or time variant information including: sensor-based construction, discretization, graphical methods, or sampling methods [15, 16, 17]. Sampling methods appear to be more widely used now, and they create roads by randomly generating and connecting nodes in the configuration space. During the road construction phase, obstacles are

accounted for and infeasible nodes or paths are disallowed. One disadvantage of random and probabilistic methods is the difficulty in ensuring solution completeness, so even a solution is infeasible the robot may be unaware [18]. [17] uses a predefined set of probabilistically generated roadmaps in the work space to manage a dynamic environment by simply activating and deactivating particular nodes of the roadmap. Without accounting for vehicle or system dynamics, these methods cannot guarantee optimality by means other than distance, and velocity, actuator, and nonholonomic constraints are hard to take into account. This means that the resulting path may create a non-trivial control problem. Computational complexity is problem and implementation method dependent.

Cell Decompositions represent free space by an association of various shaped regions known as cells. The boundaries of the cell could be based upon graphical or boundary information or assigned arbitrarily, and often a common cell shape is used throughout the space to avoid greater memory requirements. Cells that share a common boundary are referred to as adjacent and a map or graph of adjacency information permits the planning of a path from the starting location to the finish. Often an optimal search method such as the A* algorithm is implemented such that the final solution has a minimal distance. By discretizing the workspace or configuration space, the optimality of the solution is only truly valid as the resolution of the grid cells becomes high. This results in an adverse computational burden unless a quadtree or octree data structure is implemented [19]. As with Roadmap methods, recent advances in this area have proved the value of incorporating random sampling methods [20] and probabilistic methods [18, 21]. These methods can be far more efficient, especially for high degrees of freedom and challenging obstacle arrangements, than normal deterministic methods. On the other hand they can suffer from a lack of completeness. In general, not all Cell Decomposition methods take into account the vehicle nonholonomic and actuator constraints. For this reason, optimality criteria are often limited and the control problem can become challenging.

As is evidenced by portions of the preceding paragraphs on Roadmaps and Cell Decompositions, sampling methods for path planning are becoming very popular. Their attractiveness is due to their portability to various and intricate problems including

numerous degrees of freedom, complex vehicle constraints, and various workspace constraints. While not complete, these methods are frequently either probabilistically or resolution complete, meaning that completeness is guaranteed as computation time approaches infinity or as grid size becomes infinitely small. In general, there are three types of path planners involving sampling methods. The first type is a multiple-query method, evidenced by the Probabilistic Roadmaps, PRM [22], which attempts to map a connectivity graph of the entire free space. This has the advantage of being able to handle changing initial and final positions once the maps are created. The second type is the single-query method; Rapidly-exploring Random Trees (RRTs) [23, 24, 25] and Expansive-Space Trees (ESTs) [26] fall into this category. These methods branch out from either the initial position or final position, or both, and explore paths until the goal is reached, at which time the method ceases. These methods are generally faster than the multiple-query methods but require repeated computation for each new initial or final position. The last type of sampling method is simply a combination of the previous two; thereby attempting to harness the advantages of both methods. An example of this technique is the Sample-based Roadmap of Trees (SRT) [27].

Though the methods described above are significantly different, each of them contains four key actions including: node generation, node connection determination, inter-node path planning, and post path processing. The term ‘node’ is used throughout technical literature since each configuration of the vehicle is represented by one point in configuration space. Initially only the starting and final configurations, workspace obstacles, and vehicle constraints are known. New nodes generate in a random, quasi-random, or deterministic manner or they can be biased depending on manipulability, obstacles, or other means. The location and specifics of the node generation depend heavily upon the type of sampling method, as described above. The new configuration is typically tested immediately for compatibility with the workspace constraints and invalid nodes are removed. Multiple-query methods desire to search the entire free space so new nodes are often drawn uniformly about the configuration space or biased toward areas with obstacles. Single query methods only want to solve the problem of getting from point “A” to point “B” so a new node will typically be chosen conditionally with the help of a weight or bias and be an extension from old nodes.

During the next action, the planner decides to which of the previous nodes the newly generated configuration will be connected. This most often translates to a strategy of connection to the “n” closest nodes that are within a set distance. The distance calculation is important for overall algorithm speed and is non-trivial for problems that encompass translation and rotation or high degrees of freedom. After the connection nodes are determined, the path planner must attempt to connect the nodes selected. Variations of this path planner can account for significant vehicle kinematic and dynamic constraints, but fidelity and speed of the path planner must be traded with the number of samples needed to accurately realize a feasible solution. The path planner must also be concerned with the viability of the path with respect to the obstacles. Typically this is accomplished via interpolation of the path. Resolution of this check affects computational speed and for that reason some methods, referred to as “lazy” methods, ignore this check all together until a complete path – from start to finish – is found. This is assumed to minimize the number of obstacle checks required.

Once a feasible solution is found that connects the starting and finishing positions, post path processing can be accomplished. Usual processing methods focus on path shortness or smoothness. As with previous actions, the complexity and speed of this process must be traded with the strategy of utilizing a denser node base to determine what compromise provides the best results. Once the solution trajectory is generated the control problem has been reduced to a problem of trajectory following; in fact this is similarly true of all methods mentioned up until this point except for the potential functions. However, applications have endeavored to extend the scope of trajectory planning toward control planning or dynamic planning [28]. This is possible with sampling methods since the only significant change in implementation is an increase in dimension size, easily handled by this method vice the ones mentioned previously. This implementation can simplify the overall control problem – problem of path generation and path following – and therefore represents a significant advantage if it can be accomplished in a reasonable computational time.

Sampling methods have been expanded in great depth in literature. Numerous papers focus on their application to problems with complex vehicle and nonholonomic constraints [25, 29], dynamic environments [29, 30], multiple robots [31], desired

optimality [32], and high dimensionality [17, 28]. However, in general, sampling methods do not provide an optimal trajectory across the entire state space or configuration space, even with the assistance of post processing [32]. These methods are best used to find feasible solutions regardless of system complexity and work space constraints.

The next path planning method that will be discussed is the analytical approach. This name is used not because each implementation produces one specific equation that governs all motion; even with a relatively simple problem, this task would be intractable. Nor does this name imply the solution is found completely by hand. In general this designation refers to the process of analyzing one problem and simplifying it to the point where it can be handled more easily. Often these methods require numerical solutions and tunable parameters but can provide solutions online with relatively little overhead. [33, 34] create feasible trajectories for car kinematics and constraints utilizing piecewise continuous polynomials. [35] also parameterizes the solution but incorporates its use in a dynamic environment. Subsequently that method was tested in simulation on a robot with a limited sensor range. [36] solves the car problem in a time optimal manner and [37] applies a similar method to environments with moving obstacles. Despite the many benefits of analytical methods, most carry a stiff penalty of required global information and perfect knowledge of environments. Even without this drawback, these methods are ill-advised for larger problems; so their scalability and portability is at best questionable and at worst impossible.

The final path planning method discussed is the optimal control approach. An overview of optimal control techniques is provided in more detail in [7, 38, 39]. In general, these methods solve for the control trajectory that answers the planning problem while minimizing a cost function and satisfying constraints. These problems are too complex to be solved analytically and therefore are solved numerically. With a good solver, an optimal control problem primarily becomes one of problem formulation. In the past, these methods suffered from the need of a good initial guess, long computational times, and poor convergence. The latter two concerns are addressed by correct problem formulation and generally all three have benefited by advancements in computing power and algorithms. Due to the ease in accounting for constraints and the lure of an optimal

solution, these methods have long been used in an offline manner to provide control trajectories for complex problems such as reentry and spacecraft mission planning [40, 41, 42].

Historically, optimal control techniques were employed extensively in path planning problems, and the reason for this was related to the aforementioned reasons of impracticality and uncertainty. The computational complexity, convergence issues, and burden of an adequate guess made the application of these techniques intractable despite the desirability of the solutions. Further, the model, sensor, and navigation uncertainty marginalized the benefits in favor of an architecture less complex. The work presented here does not develop a new control technique; rather, it simply makes use of advances in computing and optimal control algorithms that make this application possible. While this work is intended to demonstrate applicability, future work will analyze the complications of uncertainty and its effect on optimality.

Generally, optimal control methods are complete and high problem dimensionality creates problems for only specific optimal control techniques. Grid-based search methods contain grids that grow exponentially with the number of degrees of freedom; therefore, more successful and generic numerical algorithms discretize the problem with respect to time.

Recent research [43, 44] has demonstrated optimal control's application to online control functions for complex systems including spacecraft attitude control and atmospheric reentry of a reusable launch vehicle. These papers have shown the capability to utilize optimal control in a feedback form thereby accounting for disturbances in the vehicle's environment and removing the need for a trajectory tracker in the overall control scheme. The obvious conclusion that can be drawn from these most recent studies in the application of optimal control is that the method limitations lie not in optimality, scalability, or completeness, rather it lies in computational complexity and even this argument is proving to be weak.

Robotics engineers are notorious for deftly analyzing planning techniques until a problem is found that proves a method's weaknesses or inabilities. One such problem for optimal control might involve an overly complex maze. In this case it may be advisable

to utilize a technique efficient are constructing feasible trajectories such as the sampling methods to generate the initial guess for the optimal controller. The key point here being that benefits of an optimal solution dictate a powerful argument for its exploitation in every instance.

Optimal control methods are the focus of this research not only for the advantages mentioned above but also the lack of previous exploration in the application of these methods to the car problem and unmanned vehicle control in general. The focus of this study is to advance the understanding of the capability of optimal control methods and by utilizing a powerful optimization tool, demonstrate the means to employ these methods in real systems online.

D. DIDO

This investigation into vehicle control was accomplished at the Guidance, Navigation, and Control Laboratory at the Naval Postgraduate School where an optimization tool, DIDO[®], is available for usage with university studies. The specific methods and algorithms incorporated into this tool are not discussed in this paper and are not within the scope of this research. In General, DIDO is a numerical optimization tool that incorporates pseudospectral methods with a sequential quadratic programmer, and thereby is capable of rapidly generating extremals for properly formulated optimal control problems. Proper formulation implies that the variables are well scaled and balanced and that all dynamics and constraints are differentiable. Information and illustrations portraying DIDO's development, viability, and applicability for solving optimal control problems is found in references [45, 46].

From a user's standpoint, this tool affords a relatively quick learning process, and provides invaluable information beyond the vehicle's state and control trajectory. The program will relate to the user if the solution is not feasible as it is currently coded, thus providing an avenue for completeness. Also the user receives solution information including the value of the Hamiltonian, costates, and covectors. This allows the user to evaluate the solution with respect to the necessary conditions for optimality.

E. THE OPTIMAL CONTROL PROBLEM

Integrating the information from the Kinematics and Constraints subsection it is possible to formulate the optimal control problem as Equation Set (8).

$$\begin{aligned}
\underline{x} &= \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \in X = R^3, \quad \underline{u} \in U := \left\{ \begin{array}{l} v : v_{\min} \leq v \leq v_{\max} \\ \phi : \phi_{\min} \leq \phi \leq \phi_{\max} \end{array} \right\} \\
\text{Minimize} \quad & J[\underline{x}(\cdot), \underline{u}(\cdot), t_f] = t_f \\
\text{Subject to} \quad & \dot{\underline{x}}(t) = \begin{bmatrix} v \cdot \cos(\theta) \\ v \cdot \sin(\theta) \\ \frac{v}{L} \cdot \tan(\phi) \end{bmatrix} \\
& \underline{x}(t_0) = \underline{0} \\
& t_0 = 0 \\
& e(\underline{x}_f, t_f) = \begin{bmatrix} x(t_f) - x_f \\ y(t_f) - y_f \end{bmatrix} = \underline{0} \\
& h_i(x(t), y(t)) > 0
\end{aligned} \tag{8}$$

In this format the function $h_i(x(t), y(t))$ represents the path constraints and the number of path constraints dictates the number of functions in the problem. The other vital concept introduced by this problem is the cost function. Any number and type of cost functions could have been used including: distance, time, and energy for example. While much of the previous research focuses on distance optimal criteria, the reasoning for this approach is not evident and is often related to the limitations of the solving technique. Time optimal solutions are chosen partially to display the capability of this method, but they are also selected to set a new, more realistic standard for optimality. However, it should be kept in mind that the specific cost function used is insignificant with this approach, as any can be incorporated.

Two concepts must be explored in more detail before this optimal control problem can be solved. The first is the path constraints, and the second is the scaling and balancing.

1. Path Constraints

One advantage of optimal control methods is the ease of representing constraints. In general a constraint can be a function of the states, controls, or time so long as it can be written algebraically. For the car problem, there exist vehicle constraints – holonomic

kinematics and control constraints – and path constraints. The path constraints are the obstacles around which the car must maneuver. For simplicity and computational efficiency it was desirable to represent the path constraints as continuous algebraic functions. With this framework in mind, the p-norm was used to create most generic shapes including: diamonds, circles and ellipses, and squares and rectangles. Figure 4 demonstrates how the p-norm can be manipulated to create these obstacle shapes; the ellipse and rectangle are simply extensions of the circle and square respectively where the distance along the x and y axes are dissimilar. Equation (9) presents the generic equation used to build these shapes. Specifically the square and rectangle are created by an infinity norm, which is the equivalent to selecting the maximum magnitude. Approximation of this function can be accomplished using Equation (9) and a large value for the exponent. Higher values for the exponent yielded sharper corners, but they often created numerical concerns. The square in Figure 4 was created by setting the exponent to 100.

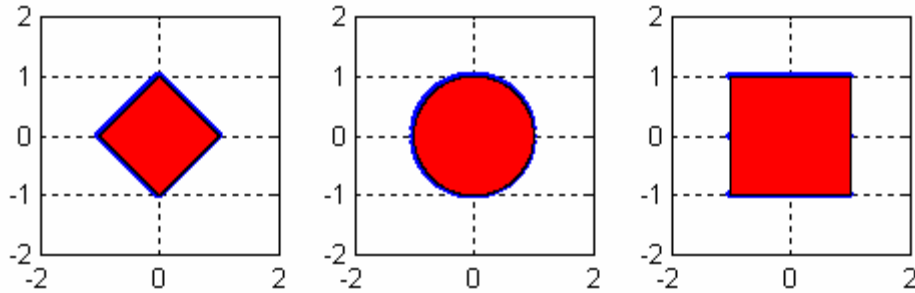


Figure 4 Unit P-Norms for P = 1, 2, and Infinity Respectively.

$$h_i(x(t), y(t)) = \left(\frac{x(t) - x_c}{a} \right)^p + \left(\frac{y(t) - y_c}{b} \right)^p - c^p \quad (9)$$

Modeling shapes outside of the generic geometries presented above would be a powerful option and, in most cases, more realistic of the random nature of the world. However, even simple polygonal shapes are non-trivial since they can not be wholly described by one algebraic equation. Figure 5 is a straightforward example of a simple shape, described by the confluence of three lines, which can not be characterized by the p-norm.

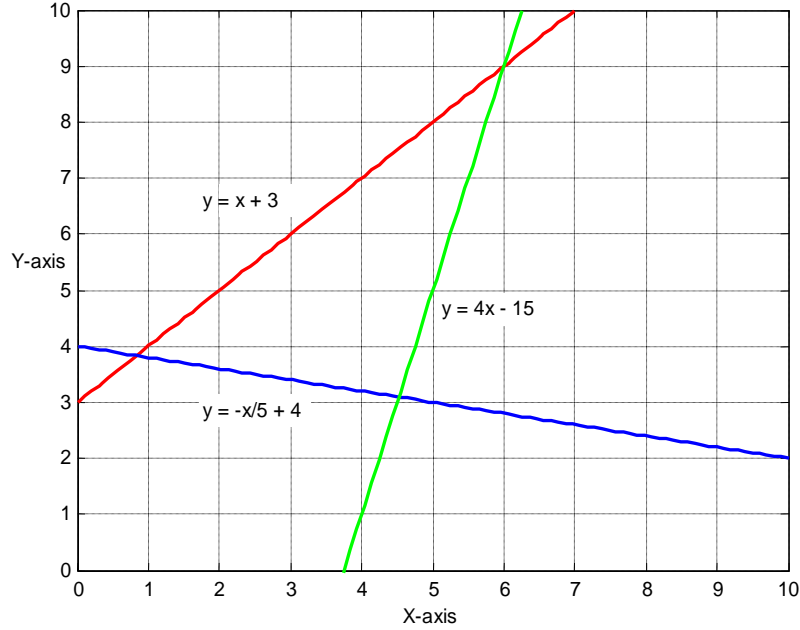


Figure 5 Simple Geometric Shape Constructed by Three Lines.

Using the equation for each line it is possible to create a series of inequality constraints that defines each point within the obstacle. This concept is evidenced by Equation Set (10).

$$\underline{g}(\underline{x}) = \begin{bmatrix} g_1(\underline{x}) \\ g_2(\underline{x}) \\ g_3(\underline{x}) \end{bmatrix} = \begin{bmatrix} -y + x + 3 \\ y + \frac{x}{5} - 4 \\ y - 4x + 15 \end{bmatrix} > \underline{0} \quad (10)$$

While every point within the obstacle satisfies every one of the three constraints, every point outside the obstacle must not satisfy at least one. Essentially this is an ‘or’ statement. Given that continuous and smooth functions are desirable for DIDO implementation, a simple logical operation could not be implemented. Instead the resulting \underline{g} vector is searched for the minimum entry, and if this entry is less than zero it can be stated that the point is outside the obstacle. Mathematically every feasible point on the map satisfies Equation (11).

$$h_i(x(t), y(t)) = \min \{g_1(\underline{x}), g_2(\underline{x}), g_3(\underline{x})\} < 0 \quad (11)$$

Figure 6 clarifies this concept. The left panel shows the mathematical constraints of the triangular obstacle and right panel shows the resulting values of the new path constraint function over the positional domain.

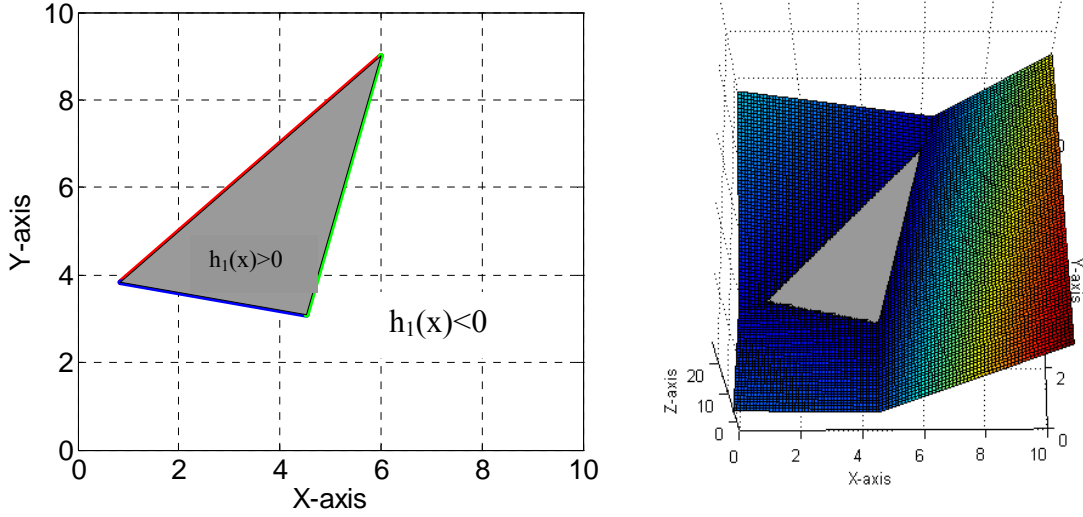


Figure 6 Constraint Representation of a Simple Polygonal Shape.

The sharp corners are captured exactly by this method. The result is continuous and grows linearly with distance from the obstacle; however, the result is not smooth, i.e. non-differentiable. The non-smoothness arrives from the fact that the path constraint is essentially the union of three planes.

It is possible to correct the non-smoothness problem, but this comes at a loss of accuracy in rendering the sharp corners. Equation (12) displays the mathematical expression for this approach, and it occurs in three distinct steps.

$$h_{p,i}(x(t), y(t)) = \ln \left[\left(e^{-g_1(x) \cdot p} + e^{-g_2(x) \cdot p} + e^{-g_3(x) \cdot p} \right)^{1/p} \right] > 0 \quad (12)$$

In the first step the elements of \underline{g} are multiplied by negative one and made into an exponent. This maps all previously positive values of \underline{g} to numbers between zero and one and all previously negative values to numbers greater than one. The next step is to take a p-norm; this step extracts the largest, single value of \underline{g} . The last step uses the natural logarithm to scale the values back to the original range. In this case a positive value for the path constraint indicates a position outside the obstacle. Both the power of the p-norm and the base of the exponent/logarithm can be altered to provide better smoothness. Figure 7 provides a picture of a result; the loss of sharpness around the corners is visible. This provides a good example of where non-smooth optimization

would be a valuable tool. Without a precise representation of the corner of an obstacle it is necessary during implementation to add a small buffer around obstacles to ensure that the path produced is actually obstruction-free.

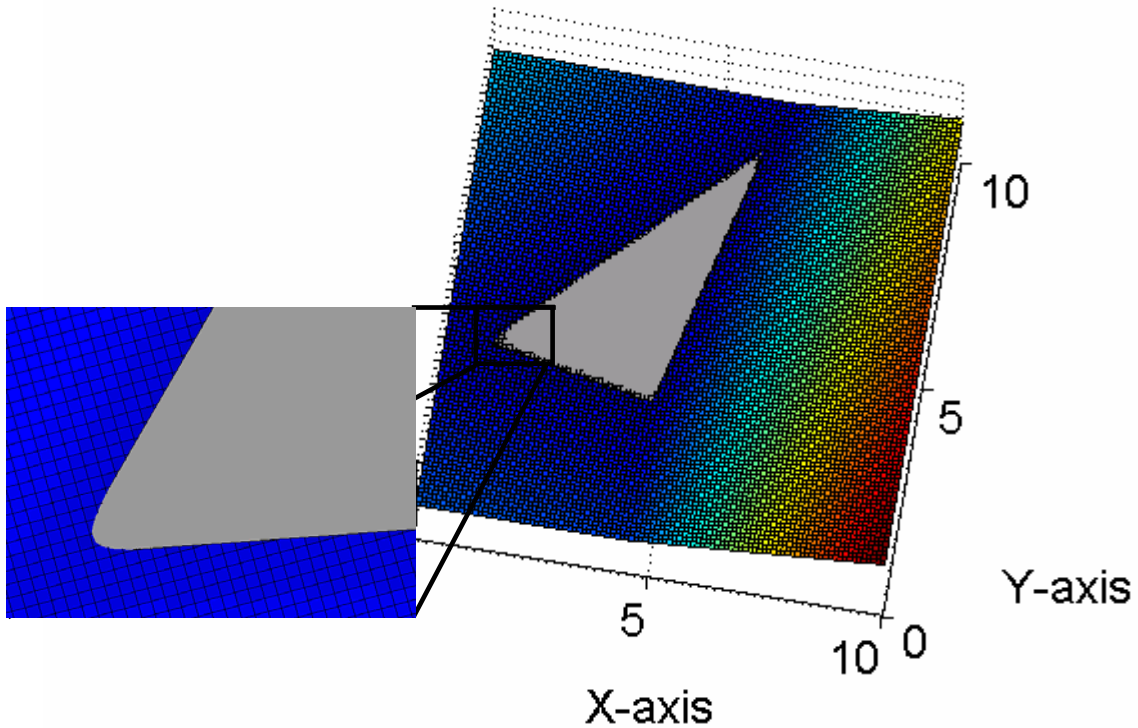


Figure 7 Smoothed Polygonal Constraint.

2. Scaling and Balancing

The quality of numerical solutions to optimal control problems relates directly to the quality of the problem formulation. This fact is true not only in terms of the accuracy of the answer, but also with respect to the computational time. One key issue involved with problem formation is scaling and balancing. Addressing this issue requires that the problem be reformulated such that the optimization algorithm only sees a problem that is scaled and balanced appropriately. To this end a scaling by variable approach was taken and is evidenced in Equation Set (13). The value of the scaling time unit is not specifically provided because in typical application it varied with respect to the overall traveling distance and the maximum speed of the vehicle.

$$\begin{aligned}
\bar{\underline{x}} &= \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{bmatrix} = \begin{bmatrix} x/X \\ y/Y \\ \theta/\Theta \end{bmatrix} & \text{where } & \begin{aligned} X &= x(t_f) \\ Y &= y(t_f) \\ \Theta &= \pi \end{aligned} \\
\bar{\underline{u}} &= \begin{bmatrix} \bar{v} \\ \bar{\phi} \end{bmatrix} = \begin{bmatrix} v/V \\ \phi/\Phi \end{bmatrix} & \text{where } & \begin{aligned} V &= v_{\max} \\ \Phi &= \phi_{\max} \end{aligned} \\
\bar{t} &= t/T & \text{where } & T : \text{problem specific}
\end{aligned} \tag{13}$$

Utilizing the new scaled variables the problem statement can be rewritten as is presented in Equation Set (14). While this process seems trivial, it is crucial in regards to the derivation of the necessary conditions for optimality as is witnessed in the next subsection.

$$\begin{aligned}
\bar{\underline{x}} &= \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{bmatrix} \in \bar{X} = R^3, \quad \bar{\underline{u}} \in \bar{U} := \begin{cases} \bar{v} : -1 \leq \bar{v} \leq 1 \\ \bar{\phi} : -1 \leq \bar{\phi} \leq 1 \end{cases} \\
\text{Minimize } & J[\bar{\underline{x}}(\cdot), \bar{\underline{u}}(\cdot), \bar{t}_f] = \bar{t}_f \\
\text{Subject to } & \dot{\bar{\underline{x}}}(\bar{t}) = \begin{bmatrix} \bar{v} \cdot \cos(\Theta \bar{\theta}) \frac{V \cdot T}{X} \\ \bar{v} \cdot \sin(\Theta \bar{\theta}) \frac{V \cdot T}{Y} \\ \frac{\bar{v}}{L} \cdot \tan(\Phi \bar{\phi}) \frac{V \cdot T}{\Theta} \end{bmatrix} \\
& \bar{\underline{x}}(t_0) = [\underline{0}] \\
& \bar{t}_0 = 0 \\
& e(\bar{\underline{x}}_f, \bar{t}_f) = \begin{bmatrix} \bar{x}(\bar{t}_f) - \bar{x}_f \\ \bar{y}(\bar{t}_f) - \bar{y}_f \end{bmatrix} = [\underline{0}] \\
& h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) > 0
\end{aligned} \tag{14}$$

3. Verification and Validation

The dynamic optimization problem stated above has the goal of finding a function, i.e. the entire control history, and is therefore infinite dimensional. The Minimum Principle converts the infinite dimensional problem to an instantaneous finite dimensional

mathematical programming problem that can be solved numerically by DIDO to find an extremal. Proving the global optimality of a solution is intractable, but it is always sensible to verify the necessary conditions, Bellman's principle, and path feasibility for each solution obtained. The latter is checked by propagating the initial conditions and system dynamics with the available control trajectory using MATLAB's Runge-Kutta algorithm. Bellman's Principle can be verified by extracting points from along the optimal solution. Finally, the necessary conditions are verified algebraically and visually using the derivation that follows.

For the scaled and balanced problem provided in the previous subsection, the control Hamiltonian can be written as the following:

$$H(\bar{\lambda}, \bar{x}, \bar{u}, \bar{t}) = \bar{\lambda}^T \cdot \begin{bmatrix} \bar{v} \cdot \cos(\Theta \bar{\theta}) \frac{V \cdot T}{X} \\ \bar{v} \cdot \sin(\Theta \bar{\theta}) \frac{V \cdot T}{Y} \\ \frac{\bar{v}}{L} \cdot \tan(\Phi \bar{\phi}) \frac{V \cdot T}{\Theta} \end{bmatrix} \quad (15)$$

where $\bar{\lambda}^T = [\bar{\lambda}_x \quad \bar{\lambda}_y \quad \bar{\lambda}_\theta]$

Now the optimal control problem can be reformulated and presented as Equation (16).

$$\begin{aligned} & \text{Minimize}_{\bar{u}} \quad H(\bar{\lambda}, \bar{x}, \bar{u}, \bar{t}) \\ & \text{Subject to} \quad \bar{u} \in U \\ & \quad \quad \quad h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) > 0 \end{aligned} \quad (16)$$

In Equation (17) the Lagrangian of the Hamiltonian is found so that the control and path constraints are included.

$$\bar{H}(\bar{\mu}, \bar{\lambda}, \bar{x}, \bar{u}, \bar{t}) = \bar{\lambda}^T \cdot \begin{bmatrix} \bar{v} \cdot \cos(\Theta \bar{\theta}) \frac{V \cdot T}{X} \\ \bar{v} \cdot \sin(\Theta \bar{\theta}) \frac{V \cdot T}{Y} \\ \frac{\bar{v}}{L} \cdot \tan(\Phi \bar{\phi}) \frac{V \cdot T}{\Theta} \end{bmatrix} + \bar{\mu}^T \begin{bmatrix} \bar{v} \\ \bar{\phi} \\ \bar{h}(x(\bar{t}), y(\bar{t})) \end{bmatrix} \quad (17)$$

where $\bar{\mu}^T = [\bar{\mu}_v \quad \bar{\mu}_\phi \quad \bar{\mu}_h]$

After presenting these equations it is possible to formulate the necessary conditions for optimality.

The first provision is the Hamiltonian Minimization Condition (HMC) and it requires that the partial derivative of the Lagrangian of the Hamiltonian with respect to the control variables be equal to zero, Equation (18).

$$\frac{\partial \bar{H}}{\partial \bar{u}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{\lambda}_x \cdot \cos(\Theta \bar{\theta}) \cdot \frac{T \cdot V}{X} + \bar{\lambda}_y \cdot \sin(\Theta \bar{\theta}) \cdot \frac{T \cdot V}{Y} + \bar{\lambda}_\theta \cdot \tan(\Phi \bar{\phi}) \cdot \frac{T \cdot V}{L \cdot \Theta} + \bar{\mu}_v \\ \bar{\lambda}_\theta \cdot \bar{v} \cdot \sec^2(\Phi \bar{\phi}) \cdot \frac{T \cdot \Phi}{L \cdot \Theta} + \bar{\mu}_\phi \end{bmatrix} \quad (18)$$

Complementing this is the Karush-Kuhn-Tucker (KKT) conditions for the covectors.

$$\begin{aligned} \bar{\mu}_v & \begin{cases} \leq 0 & \bar{v}(\bar{t}) = -1 \\ = 0 & \text{for } -1 < \bar{v}(\bar{t}) < 1 \\ \geq 0 & \bar{v}(\bar{t}) = 1 \end{cases} \\ \bar{\mu}_\phi & \begin{cases} \leq 0 & \bar{\phi}(\bar{t}) = -1 \\ = 0 & \text{for } -1 < \bar{\phi}(\bar{t}) < 1 \\ \geq 0 & \bar{\phi}(\bar{t}) = 1 \end{cases} \\ \bar{\mu}_h & \begin{cases} = 0 & \text{for } h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) > 0 \\ \leq 0 & \text{for } h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) = 0 \end{cases} \end{aligned} \quad (19)$$

For a time optimal problem, it can be expected that the vehicle velocity will be at the maximum value and in most cases the vehicle steering angle will be somewhere between the minimum and maximum. The first expectation translates to $\bar{\mu}_v \geq 0$, but in general this tells little about the other terms in the first equation of the HMC. The second expectation leads to the conclusion that typically $\bar{\mu}_\phi = 0$ and the first term in the second equation of the HMC will also need to be zero to satisfy optimality conditions. This can only be true if $\bar{\lambda}_\theta = 0$ when the car is in motion.

The next necessary condition is the Adjoint Equation and it describes the change of the costates over time. Equation (20) provides the derivation of this equation.

$$\dot{\bar{\lambda}} = \begin{bmatrix} \dot{\lambda}_x \\ \dot{\lambda}_y \\ \dot{\lambda}_\theta \end{bmatrix} = - \frac{\partial \bar{H}}{\partial \bar{x}} = \begin{bmatrix} \bar{\mu}_{h,i} \cdot \frac{\partial h_i}{\partial \bar{x}} + \bar{\mu}_{h,i+1} \cdot \frac{\partial h_{i+1}}{\partial \bar{x}} + \dots \\ \bar{\mu}_{h,i} \cdot \frac{\partial h_i}{\partial \bar{y}} + \bar{\mu}_{h,i+1} \cdot \frac{\partial h_{i+1}}{\partial \bar{y}} + \dots \\ -\lambda_x \cdot v \cdot \sin(\Theta \bar{\theta}) \cdot \frac{T \cdot V \cdot \Theta}{X} + \lambda_y \cdot v \cdot \cos(\Theta \bar{\theta}) \cdot \frac{T \cdot V \cdot \Theta}{Y} \end{bmatrix} \quad (20)$$

Several important concepts can be extracted from this information. First, λ_x and λ_y remain constant whenever the vehicle is not touching an obstacle; this is derived with the assistance of the KKT conditions which show that $\bar{\mu}_n = 0$ when $h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) > 0$. The second key concept is more involved. As discussed previously, $\bar{\lambda}_\theta$ is expected to be zero when the car is in motion; however, the equation for $\dot{\bar{\lambda}}_\theta$ shows that $\bar{\lambda}_\theta$ must also be constant when the vehicle is not in motion. Therefore, Equation (21) must be true at all times that the maximum steering angle is not being used, and this, in turn, forces $\bar{\lambda}_x$ and $\bar{\lambda}_y$, by virtue of the equation for $\dot{\bar{\lambda}}_\theta$, to compensate for changes in $\bar{\theta}$, the car's orientation angle, that would otherwise cause Equation (21) not to be true.

$$\bar{\lambda}_\theta = 0 \quad (21)$$

This fact may initially appear trivial, but since θ can also be thought of as the direction of instantaneous motion for the car, it can lead to a second deduction. It can be deduced that since λ_x and λ_y only change when touching an obstacle, $\bar{\theta}$ can only change when the car is touching an obstacle, and when the car is not touching an obstacle it must be traveling along a straight path. This proves mathematically a seemingly logical conclusion and the basis of the research presented in [36]. As a reminder though, this analysis is not valid when the maximum steering angle is being used; at that time $\bar{\mu}_\phi$ would have a value other than zero although it is not expected to occur often.

The next stipulation is the Terminal Transversality Condition (TTC), which helps to provide boundary conditions for the costates. The TTC can be stated as Equation (22).

$$\bar{\lambda}(\bar{t}_f) = \frac{\partial \bar{E}}{\partial \bar{x}_f} \quad \text{where} \quad \bar{E}(\bar{v}, \bar{x}_f, \bar{t}_f) = E(\bar{x}_f, \bar{t}_f) + \bar{v}^T e(\bar{x}_f, \bar{t}_f) \quad (22)$$

$$\text{and} \quad \bar{v}^T = [\bar{v}_x \quad \bar{v}_y \quad \bar{v}_\theta]$$

In this equation $\bar{E}(\bar{v}, \bar{x}_f, \bar{t}_f)$ is the Endpoint Lagrangian and $E(\bar{x}_f, \bar{t}_f)$ is the Endpoint Cost. Applying this condition to the car problem yields Equation (23).

$$\begin{aligned} \bar{E}(\bar{\underline{v}}, \bar{\underline{x}}_f, \bar{t}_f) &= \bar{t}_f + \bar{\underline{v}}^T \begin{bmatrix} \bar{x}(\bar{t}_f) - \bar{x}_f \\ \bar{y}(\bar{t}_f) - \bar{y}_f \end{bmatrix} \\ \bar{\underline{\lambda}}(\bar{t}_f) &= \begin{bmatrix} \bar{\lambda}_x(\bar{t}_f) \\ \bar{\lambda}_y(\bar{t}_f) \\ \bar{\lambda}_\theta(\bar{t}_f) \end{bmatrix} = \begin{bmatrix} \bar{v}_x \\ \bar{v}_y \\ 0 \end{bmatrix} \end{aligned} \quad (23)$$

Unfortunately, this necessary condition only specifies that the final value of $\bar{\lambda}_\theta$ be equal to zero; a fact that was deduced previously.

The final two necessary conditions are considered simultaneously since the both pertain to the value of the Hamiltonian. The first is the Hamiltonian Value Condition (HVC), Equation (24), which is applied in situations such as this where the final time is not fixed.

$$\bar{H}(\bar{t}_f) + \frac{\partial \bar{E}}{\partial \bar{t}_f} = 0 \Rightarrow \bar{H}(\bar{t}_f) = -1 \quad (24)$$

This result is complemented by the Hamiltonian Evolution Equation (HEE), Equation (25), which describes the evolution of the Hamiltonian.

$$\dot{\bar{H}} = \frac{\partial \bar{H}}{\partial t} \Rightarrow \dot{\bar{H}} = 0 \quad (25)$$

Together these two outcomes show that the value of the Hamiltonian must be a constant negative one throughout the trajectory.

In summary, these aforementioned conditions, along side the feasibility verification, cannot prove global optimality, but in the very least it can demonstrate that the solution obtained through DIDO is viable and locally optimal. Therefore, even in the worst case, this method provides more capability to the user than the approaches used previously.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TRICYCLE PROBLEM

A. MOTIVATION

In the process of validating results for the four-wheeled car problem, it was first desirable to ensure that the solution technique could be validated on a less complex problem. The tricycle problem was a perfect candidate to satisfy this request, and given its extensive use by previous path planning techniques, it provides a good foundation for introducing the capability of optimal control methods. The dynamics of the tricycle are similar but less complex than the four-wheeled car, thereby allowing the tricycle to rotate in place. Other than this nuance, the problem definitions contain many parallels, making the tricycle problem an ideal step in an iterative process of proving the capability of a new control technique.

B. PROBLEM DEFINITION

Similar to the four-wheeled car, the tricycle has two rear wheels, however, this is where the similarities end. There is only one forward steering wheel and it typically provides the driving power. The kinematics of a tricycle system can be represented as Equation (26).

$$\underline{\dot{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cdot \cos(\theta) \\ v \cdot \sin(\theta) \\ \omega \end{bmatrix} \quad (26)$$

In this representation the positional coordinates reflect the position of the forward wheel where it touches the ground, but the equation for $\dot{\theta}$ is simplified. The scaled optimal control problem can be devised as Equation (27). The parallels between this problem formulation and the original drive the Adjoint Equation, TTC, HEE, and HVC to be unchanged from the previous derivation. The HMC and KKT Conditions do change and this variation is evidenced in Equations (28) and (29). The conclusions drawn from the car problem necessary conditions are reflected here. When the tricycle is not changing its

heading, $\bar{\lambda}_\theta$ must be zero, and $\bar{\lambda}_x$ and $\bar{\lambda}_y$ must be constant when the tricycle is not touching an obstacle. Also the Hamiltonian must be a constant value of negative one throughout the trajectory, this consistent with all minimal time problems.

$$\bar{\underline{x}} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{bmatrix} \in \bar{X} = R^3, \quad \bar{\underline{u}} \in \bar{U} := \left\{ \begin{array}{l} \bar{v} : -1 \leq \bar{v} \leq 1 \\ \bar{\omega} : -1 \leq \bar{\omega} \leq 1 \end{array} \right\}$$

$$\text{Minimize } J[\bar{\underline{x}}(\cdot), \bar{\underline{u}}(\cdot), \bar{t}_f] = \bar{t}_f$$

$$\text{Subject to } \dot{\bar{\underline{x}}}(\bar{t}) = \begin{bmatrix} \bar{v} \cdot \cos(\Theta \bar{\theta}) \frac{V \cdot T}{X} \\ \bar{v} \cdot \sin(\Theta \bar{\theta}) \frac{V \cdot T}{Y} \\ \bar{\omega} \cdot \frac{T}{\Theta} \end{bmatrix}$$

$$\bar{\underline{x}}(t_0) = [\underline{0}] \quad (27)$$

$$\bar{t}_0 = 0$$

$$e(\bar{\underline{x}}_f, \bar{t}_f) = \begin{bmatrix} \bar{x}(\bar{t}_f) - \bar{x}_f \\ \bar{y}(\bar{t}_f) - \bar{y}_f \end{bmatrix} = [\underline{0}]$$

$$h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) > 0$$

$$(HMC) \quad \frac{\partial \bar{H}}{\partial \bar{\underline{u}}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{\lambda}_x \cdot \cos(\Theta \bar{\theta}) \cdot \frac{T \cdot V}{X} + \bar{\lambda}_y \cdot \sin(\Theta \bar{\theta}) \cdot \frac{T \cdot V}{Y} + \bar{\mu}_v \\ \bar{\lambda}_\theta + \bar{\mu}_\omega \end{bmatrix} \quad (28)$$

$$\bar{\mu}_v \begin{cases} \leq 0 & \bar{v}(\bar{t}) = -1 \\ = 0 & \text{for } -1 < \bar{v}(\bar{t}) < 1 \\ \geq 0 & \bar{v}(\bar{t}) = 1 \end{cases}$$

$$(KKT) \quad \bar{\mu}_\omega \begin{cases} \leq 0 & \bar{\omega}(\bar{t}) = -1 \\ = 0 & \text{for } -1 < \bar{\omega}(\bar{t}) < 1 \\ \geq 0 & \bar{\omega}(\bar{t}) = 1 \end{cases} \quad (29)$$

$$\bar{\mu}_h \begin{cases} = 0 & \text{for } h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) > 0 \\ \leq 0 & \text{for } h_i(\bar{x}(\bar{t}), \bar{y}(\bar{t})) = 0 \end{cases}$$

C. SOLUTION VALIDATION

Validation of optimal solutions occurs in three steps including: necessary conditions, feasibility, and Bellman's Principle. Throughout this paper the

aforementioned verification techniques will be applied and displayed for one instance of a particular problem formulation. Validation will not be repeated for each application of a problem formulation so long as the structure remains unaltered and the repeated display offers no further information.

The problem selected for method validation involves three circular shapes; the optimal trajectory and obstacles are presented in Figure 8.

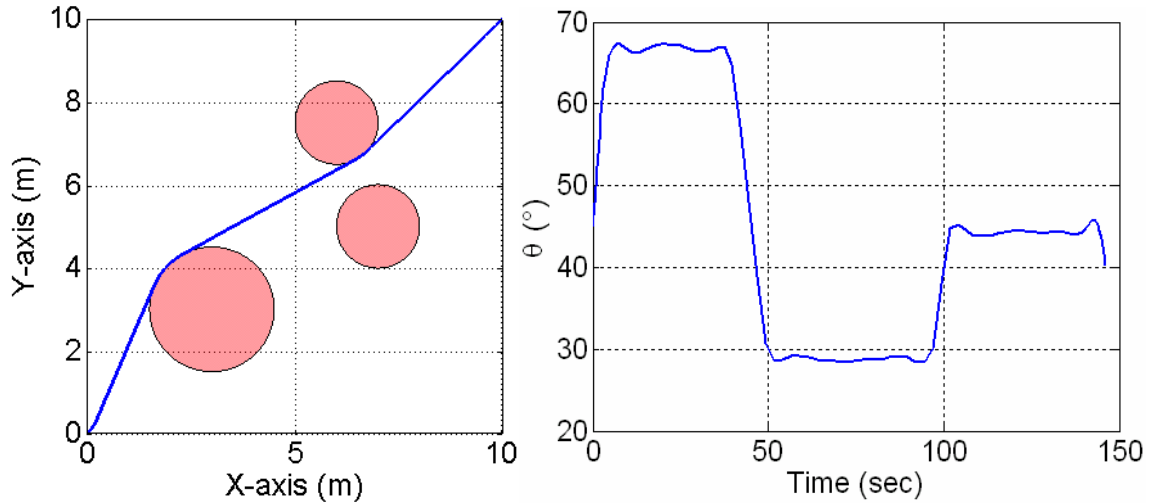


Figure 8 Optimal Tricycle Kinematic Positional Trajectory (left) and Angular Trajectory (right).

From the trajectories presented here, it is apparent that the vehicle is traveling in a straight line when it is not touching an obstacle. The positional and angular trajectories are presented in an unscaled form; the remaining figures give information in scaled form since it is more important to see the scale of the variables inside DIDO. Figure 9 gives the scaled control trajectory. The maximum velocity is used at all times. The maximum steering angle is applied only at the beginning of the trajectory, and this means that, according to the KKT conditions and HMC, $\bar{\mu}_\omega$ and $\bar{\lambda}_\theta$ should be zero at all times except during the beginning. Figure 10 and Figure 11 display the costates and covectors, which instantiate this situation. Further, Figure 10 and Figure 11 can be studied to reveal that $\bar{\lambda}_x$ and $\bar{\lambda}_y$ only change in value when the tricycle touches an obstacle, i.e. when the path constraint covectors become active.

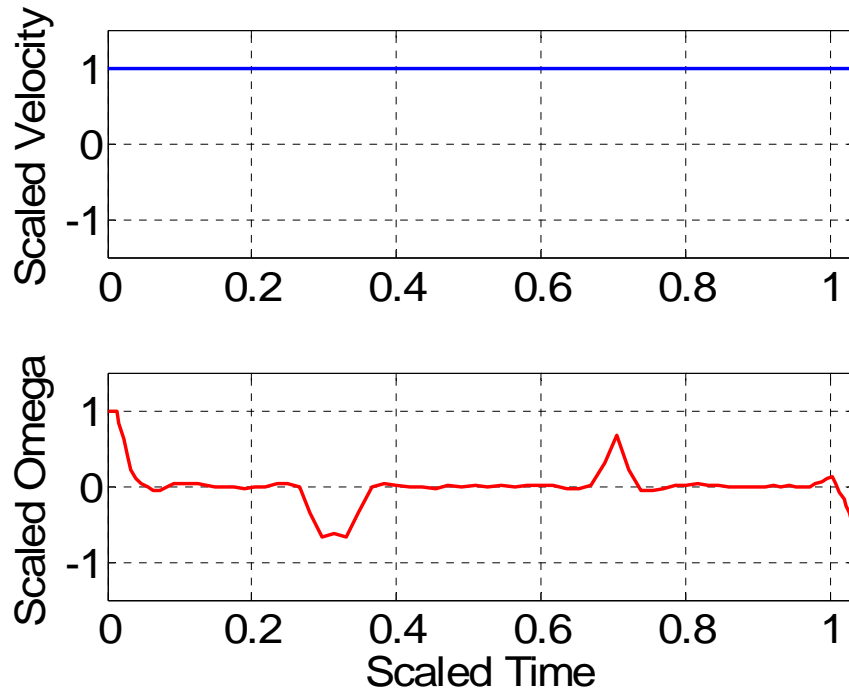


Figure 9 Scaled Control Trajectories for Validation Problem.

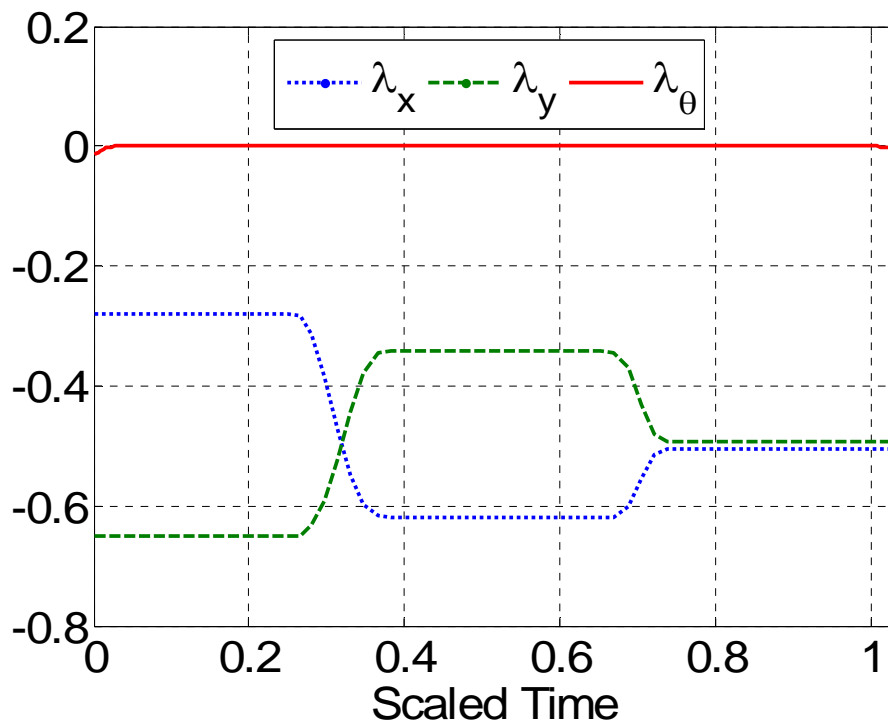


Figure 10 Costate Trajectories for Validation Problem.

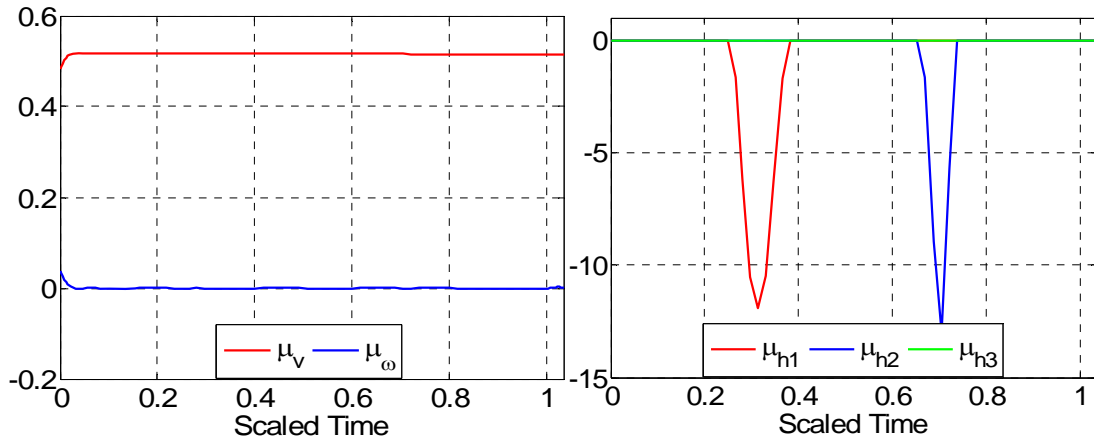


Figure 11 Covectors for Control (left) and Path (right) for Validation Problem.

The HMC equations, from Equation (28), are plotted in Figure 12 showing that this condition is confirmed within numerical precision.

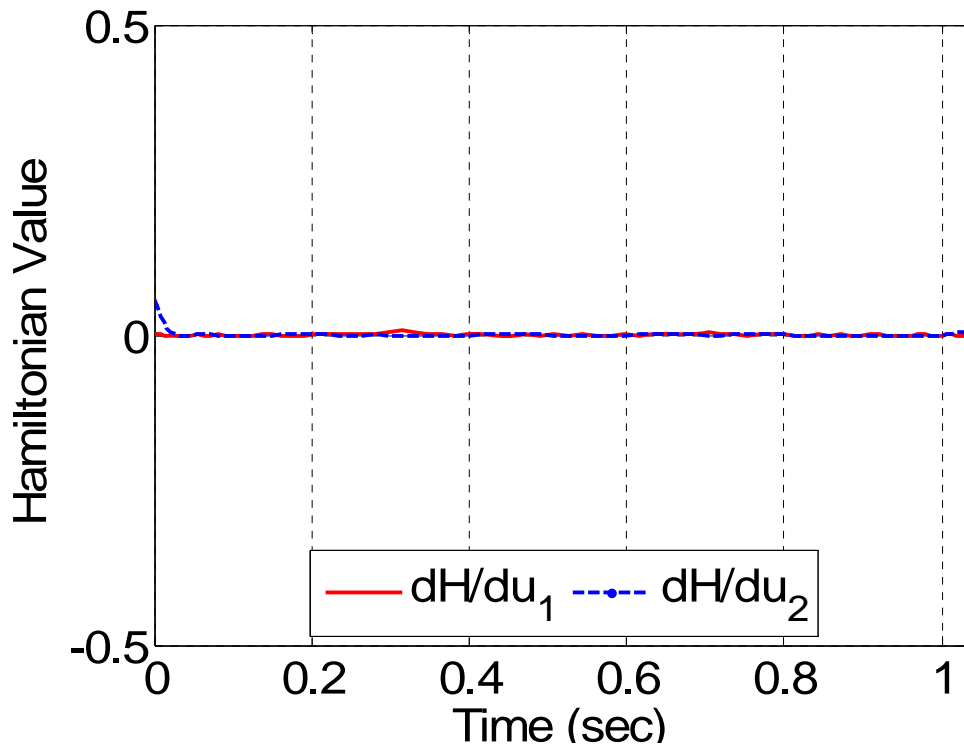


Figure 12 Plot of the Hamiltonian Minimization Condition Equations for the Validation Problem.

The value of the Hamiltonian is plotted Figure 13; this figure validates the HEE and HVC.

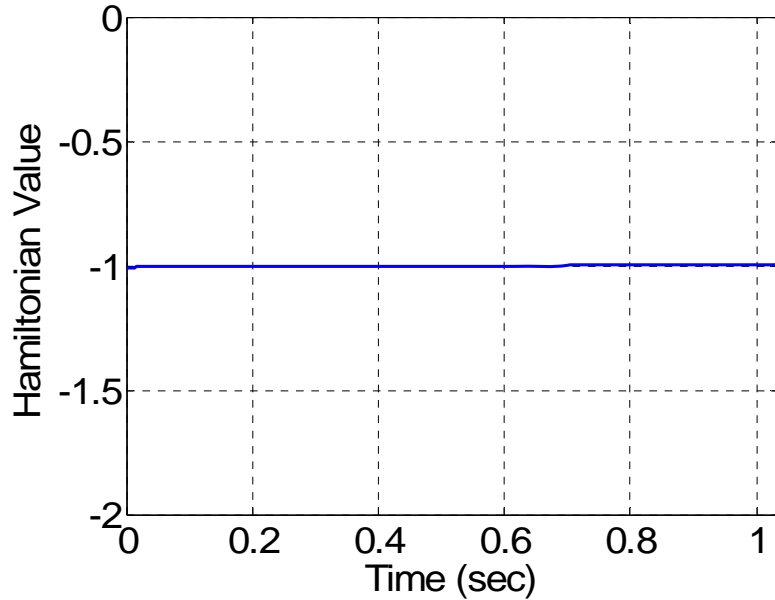


Figure 13 Plot of the Value of the Hamiltonian for the Validation Problem.

The next step of validation involves state propagation of the tricycle model using the DIDO generated control trajectory. This step employs MATLAB's 'ode45' command which is based on an explicit Runge-Kutta formula. Figure 14 gives the propagated positional and angular trajectories side-by-side with the DIDO generated trajectories for comparison. Due to the similarity between the two trajectories, they are nearly plotted in the same graph

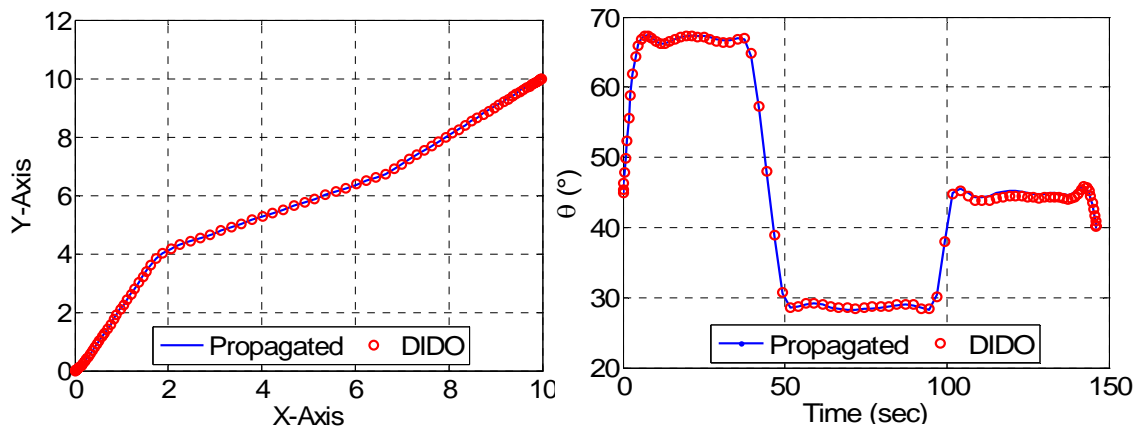


Figure 14 Propagated versus DIDO Generated Positional (left) and Angular (right) Trajectories.

The last step in validation is verification of Bellman's Principle. This rule can best be explained by a thought experiment. Imagine that A and B are two points that

have been connected by an optimal path, and point C lies along this optimal path. If the previous statements are true, then the optimal path that connects point A to point C and likewise connects point B to point C must coincide with the original path between A and B. This principle is straightforward, but it provides a benchmark to which all optimization techniques can be tested. For the tricycle problem, two intermediate points along the generated optimal solution were selected for testing of Bellman's Principle. With the two endpoints and two intermediate positions, three optimal sub-trajectories were calculated and compared to the original shown above. Figure 15 displays the results of this comparison. The original trajectory is shown in black and the sub-trajectories are presented in each a different color. The paths are identical.

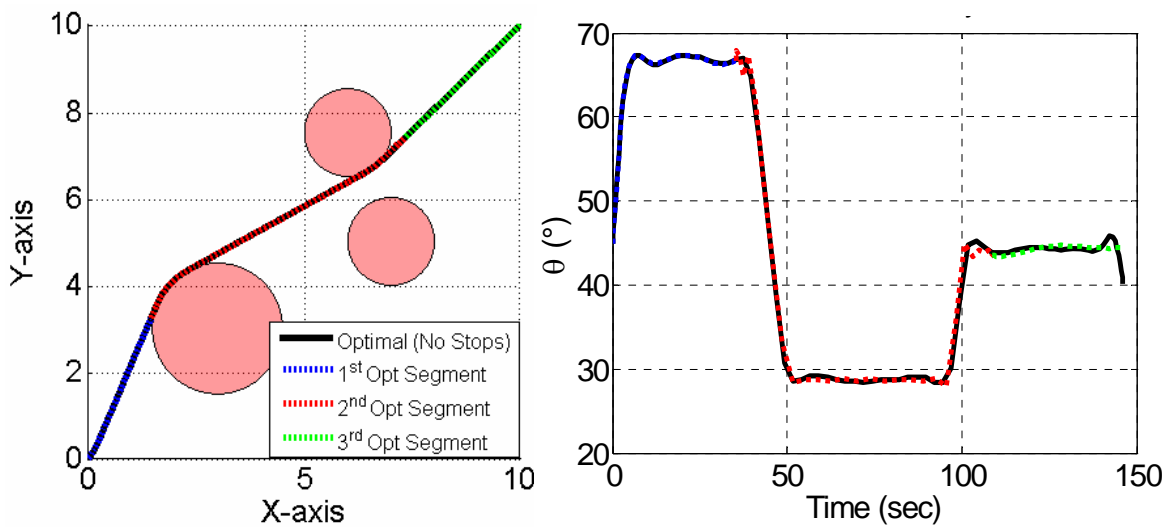


Figure 15 Bellman's Principle Test for Validation Problem.

A similar point can be made by Figure 16 which shows that the total cost of the original trajectory is equivalent to the sum of the sub-trajectory costs.

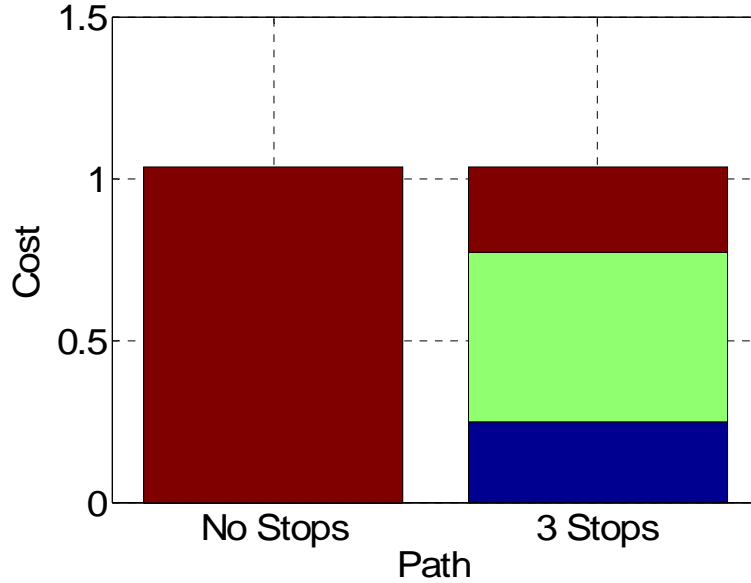


Figure 16 Trajectory Cost for Continuous and Segmented Paths.

D. RESULTS

Other aspects of the simulation technique could be tested with the tricycle model. The validation problem from the previous section used three circular obstacles. Other flat-edged p-norm shapes and the polygonal shapes needed to be evaluated.

Figure 17 shows the DIDO generated optimal path for a problem similar to the validation problem with the exception of using squares generated by the p-norm – with an exponent value of 100. In addition to the path, the trajectory points created by DIDO are also displayed – each point appears as an ‘x.’ DIDO turns the optimal control problem into a finite-dimensional parameter optimization problem, and in doing such, it discretizes the trajectory with respect to time. The degree of discretization used by DIDO is a user-defined quantity, specified by the number of nodes. At each node, optimality and satisfaction of the constraints are ensured. In other words, the discretized solution is feasible but the continuous solution is not. From Figure 17 the limitations of DIDO generated optimal solutions are apparent. The nodes of the optimal solution satisfy the constraints but the path that is propagated in between the nodes does not. This condition is caused by the discretization of the optimal control problem and by the inaccurate representation of the square shape by the 100-norm – vice the infinity norm.

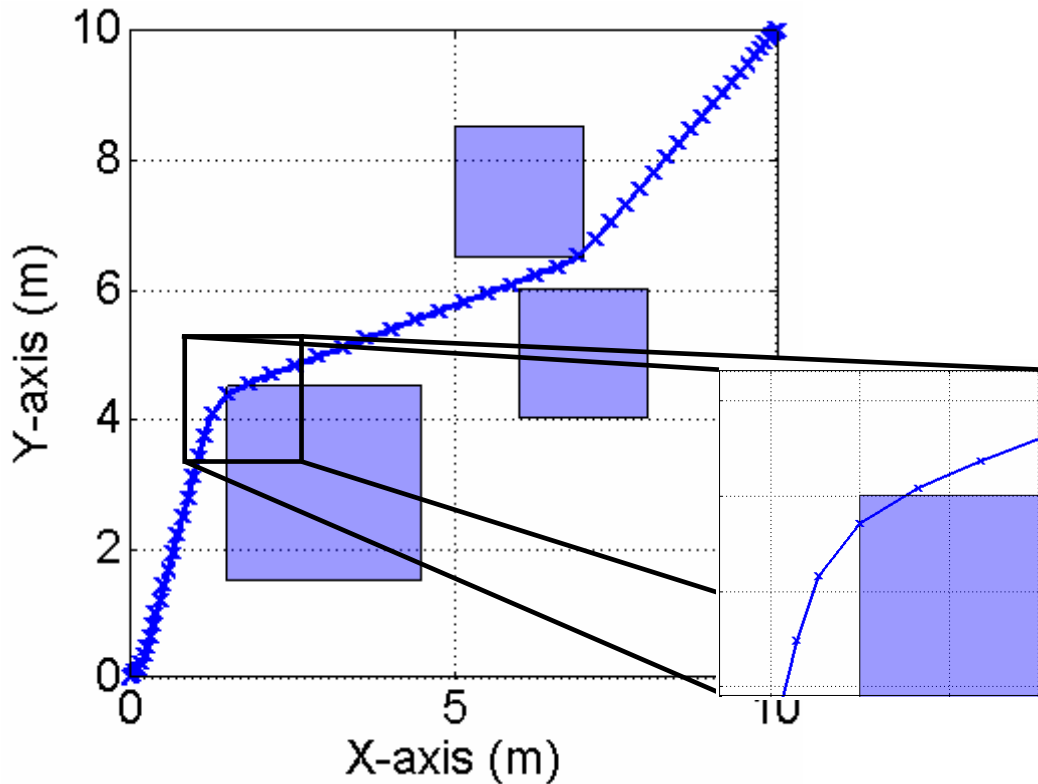


Figure 17 Tricycle Optimal Path in Square Obstacle Environment.

Two solutions to this problem can be implemented. The first is to use a greater degree of discretization. By using more nodes, the optimal path will be less likely to cut-the-corner, but the drawback of this approach is increased computational complexity. The second approach is to fictitiously extend the size of the obstacle when describing the path constraint. This can ensure that the path lies outside the actual obstacle, and is necessary regardless to ensure that a vehicle with size beyond a single point can traverse the path. Figure 18 shows a new optimal path generated by increasing the number of nodes and extending the obstacle size by 0.1m. The result is satisfactory as the path now lies outside the obstacle. Prudent judgment must be exercised when implementing either of these corrections. Increasing the number of nodes can significantly add to calculation time, Table 1, with minor improvement in solution accuracy. In general, increasing the number of nodes alone will not remove the problem of driving through a sharp corner, and it has a limiting effect on closed-loop problems. For this reason, it is not the preferred method of rectifying the problem.

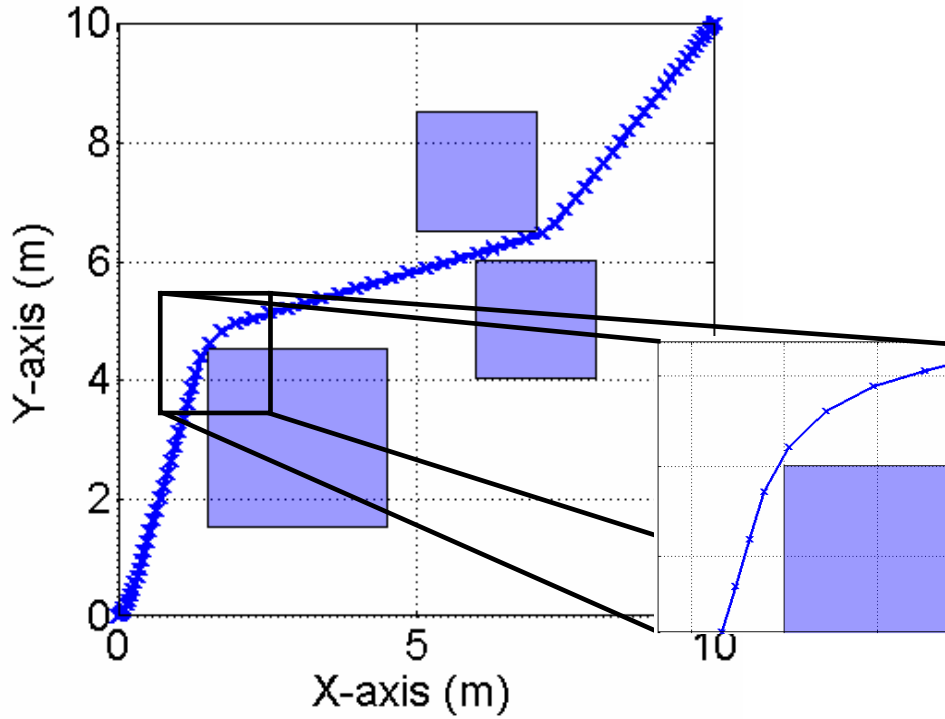


Figure 18 Adjusted Tricycle Optimal Path in Square Obstacle Environment (0.1m Obstacle Extension).

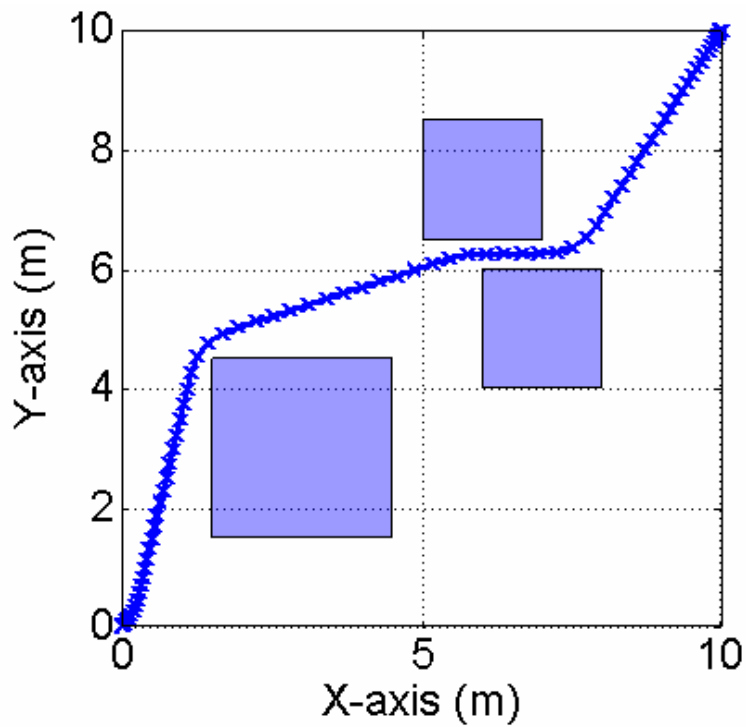


Figure 19 Further Adjusted Tricycle Optimal Path in Square Obstacle Environment (0.25m Obstacle Extension)

Table 1 Effect of Problem Formulation on Computation Time

	No Extension	0.1m Extension	0.20m Extension
20 Node Pre-Computation	24 seconds	16 seconds	18 seconds
60 Node Solution	36 seconds	40 seconds	44 seconds
80 Node Solution	51 seconds	61 seconds	101 seconds

Correcting the corner problem by extending the obstacle can also have a negative effect on the result. As can be seen in Figure 18 and Figure 19, increasing the extension length forces the resultant path to be less optimal with respect to the actual obstacle size. Logic dictates that the obstacles can be extended to a point where the optimal path travels outside the obstacle formation. The big picture perspective however, is that in robotics, optimality is not the only criteria for judging the value of a path. Due to sensor error, model error, and other uncertainties, an optimal path that exactly touches an obstacle would likely result in failure. In other words, an optimal solution is worthless if it results in real-world failure. The solution that will be addressed in paper is to balance the desirability of optimal solutions with the necessity of obstacle avoidance.

Given that the polygonal shapes also have sharp corners, it is no surprise that optimal paths experience the same problems around these shapes. This problem can be resolved in the manner discussed previously. Figure 20 shows an optimal trajectory around a five-sided polygonal shape; the constraint shape was extended to permit proper path planning.

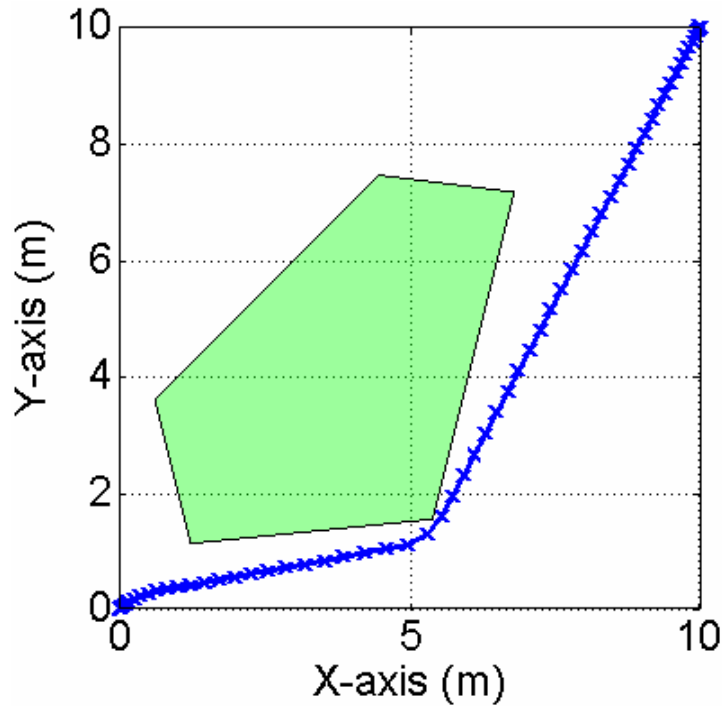


Figure 20 Optimal Tricycle Trajectory around Polygonal Shape.

Some problems, such as the maze problem shown in Figure 21 and Figure 22 create a more substantial challenge for DIDO. Until this point all results have been generated by a low-node pre-computation step followed by a more accurate, high-node solution. This method, referred to as bootstrapping, not only improves accuracy but also aids computation time since the pre-computation step provides a good guess for the final solution. The initial guess given in the pre-computation step provides only the initial and final locations, i.e. no guess.

The maze problem has narrow, long obstacles which are not captured well by the low-node pre-computation trajectory, Figure 21; therefore, a poor guess is given to the high-node solution step. This issue creates an intractable problem for DIDO, i.e. the high-node DIDO solution creates infeasibilities, but this is a problem caused by global information and a bad low-node guess. Upon receiving the infeasibilities the DIDO program recommends that the user implement a different guess. In general, the complexity of the path constraints creates a similar result regardless of the number of nodes utilized when the initial guess is a “no guess.”

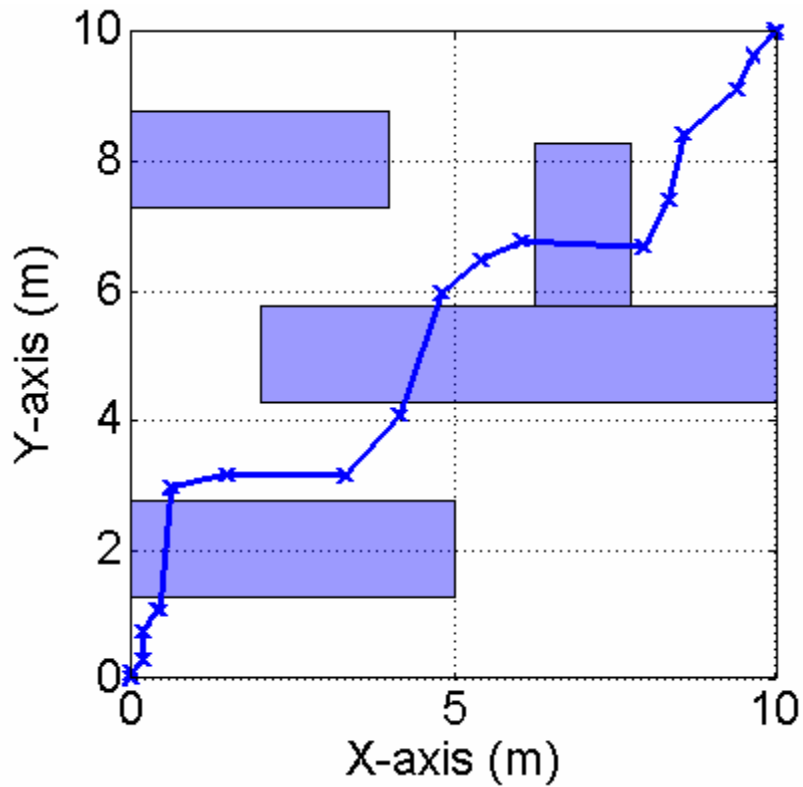


Figure 21 Low-node Pre-computational Step for Maze Problem.

The solution in Figure 22 is generated by providing DIDO with a simple four-point initial guess, two of which are the initial and final positions. This illustrates DIDO's strengths and weaknesses. Generally DIDO can be used with poor or minimal guess information; however, complex path constraints can create problems if a bad guess is supplied. Human intervention can remain null even in complex problems so long as a technique is implemented to provide DIDO with a tangible guess. This can be accomplished with one of the sampling methods mentioned earlier. Later versions of DIDO will address this concern internally and remove the necessary human intervention.

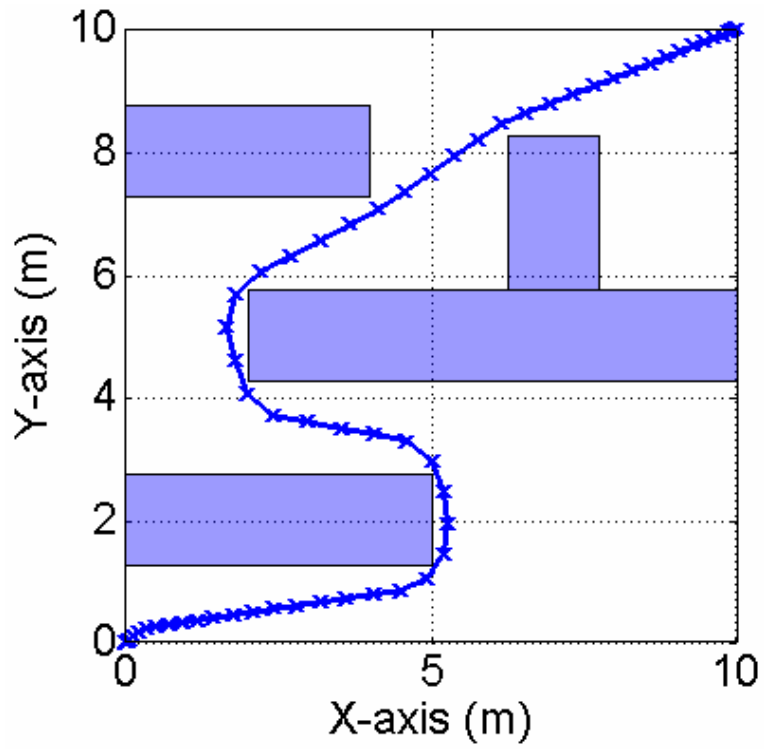


Figure 22 Optimal Tricycle Trajectory through a Maze Problem.

IV. FOUR-WHEELED CAR

A. OPEN LOOP

The previous section validated the open loop optimal control techniques with the simplified tricycle dynamics, and this section applies the same concepts to the four-wheeled car problem. The lessons learned with regard to path constraint representation directly transfer from the tricycle problem to the car problem since they relate only to workspace constraints, not to vehicle characteristics.

1. Validation

The first objective was validation of the car problem results, thereby permitting greater exploration of applications. The validation problem used for the car, shown in Figure 23, consists of several obstacle shapes including squares, circles, ellipses, rectangles, and diamonds. As was explained in the previous section, the obstacle size was fictitiously enlarged during constraint representation to allocate adequate clearance for the vehicle and permit the car to successfully maneuver sharp corners.

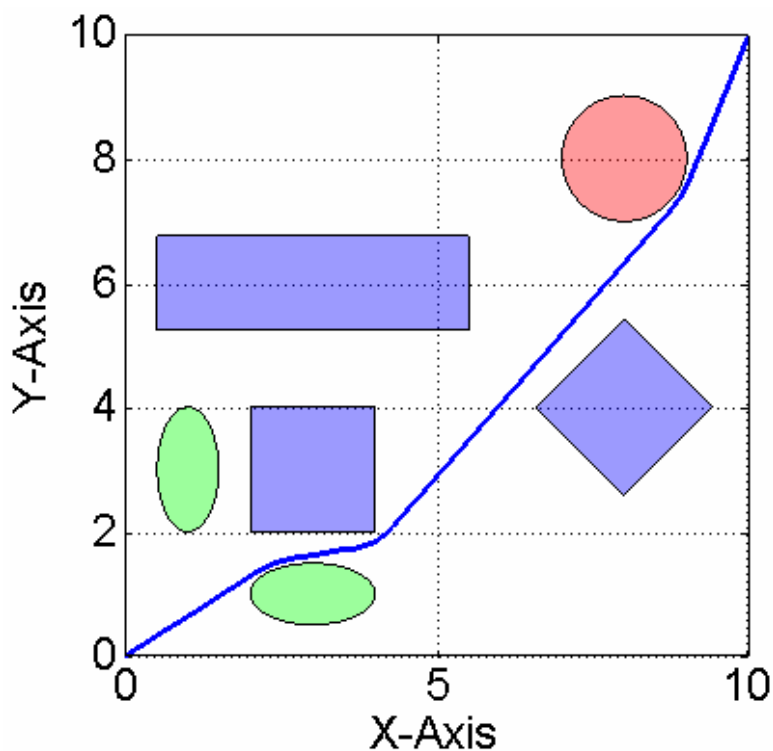


Figure 23 Car Optimal Control Validation Problem.

The initial and final conditions for this problem are described by Equation (30).

$$\underline{x}(t) = \underline{x}^i = \begin{bmatrix} 0 & 0 & \pi/4 \end{bmatrix}^T \quad e(\underline{x}, t_f) = \begin{bmatrix} x(t_f) - 10 & y(t_f) - 10 \end{bmatrix}^T = \underline{0} \quad (30)$$

Figure 23 provides the optimal positional trajectory. The car encounters only three obstacles along its path, and no path constraint violations occur during the trajectory. Figure 24 displays the angular trajectory of the solution; the results are presented in degrees to ease visualization. Despite the noise in the solution, the car travels in a straight line when it is not contacting an obstacle.

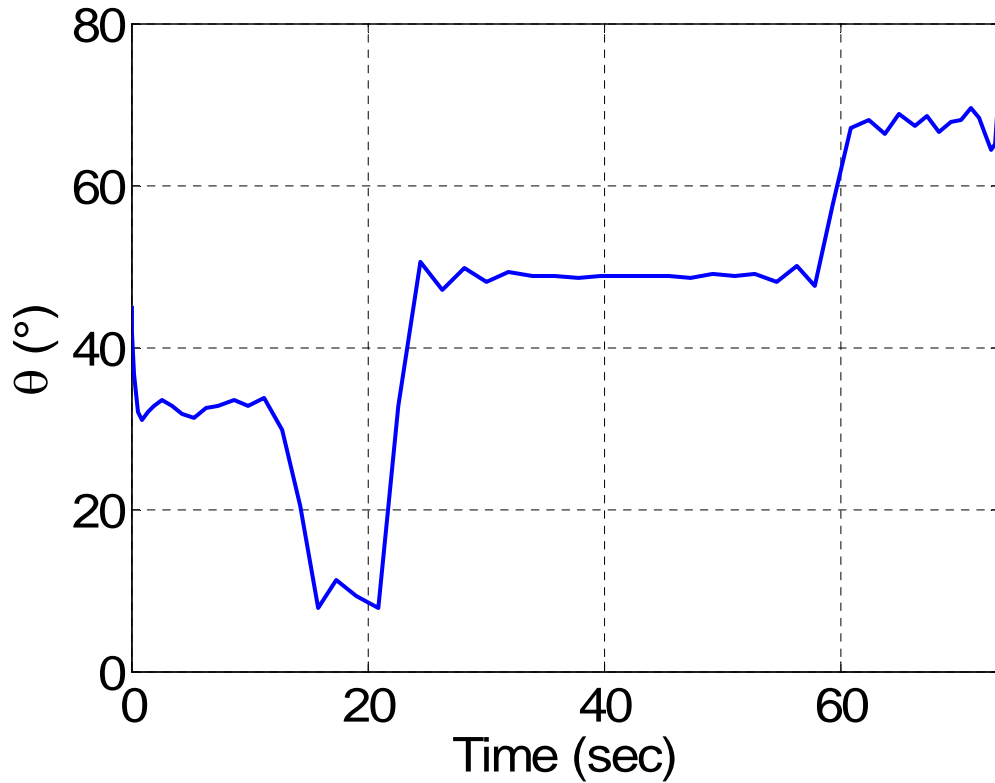


Figure 24 Angular Trajectory for Car Validation Problem.

The control trajectory is presented in Figure 25, and once again, the angle is given in degrees for the purposes of visualization. As expected, the velocity is at its maximum value throughout the maneuver and the steering angle value is primarily between its extremes of $\pm 35^\circ$. The first situation causes the velocity covector to be active throughout the maneuver and the latter has the opposite effect on the steering angle covector.

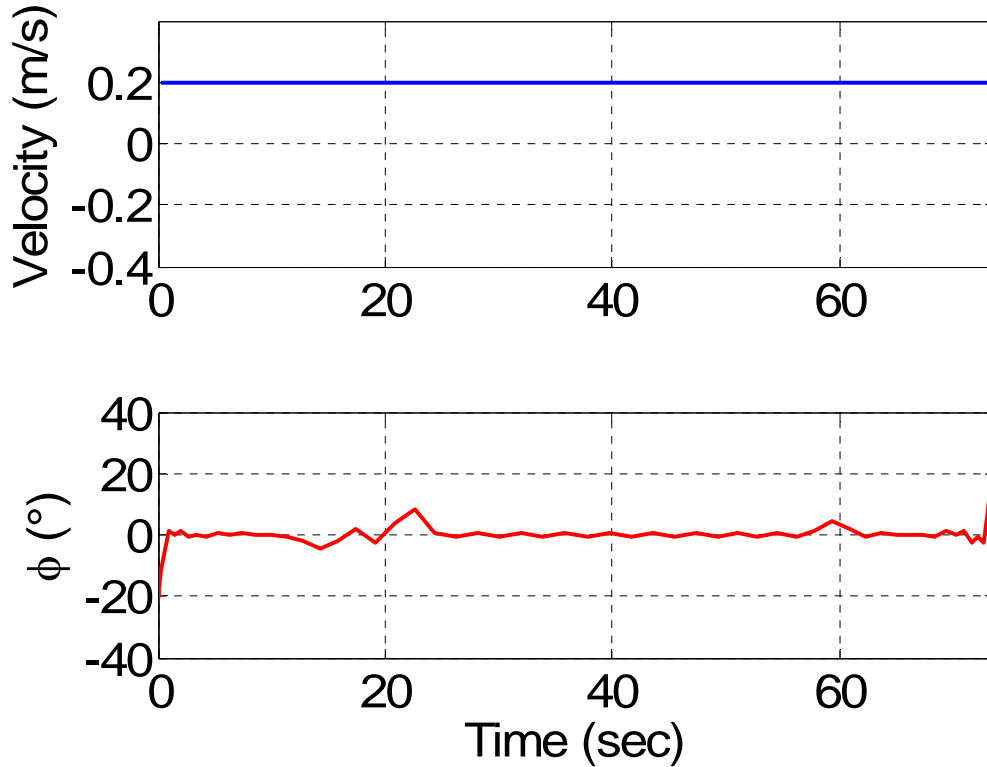


Figure 25 Control Trajectory for the Car Validation Problem.

Figure 26 through Figure 30 display the necessary condition graphs for the car maneuvering optimization problem including, in the order of presentation: the control Hamiltonian value, the Hamiltonian Minimization Condition (HMC) verification, costates, path covectors, and control covectors. Numerical error is visible in the figures of the Hamiltonian and HMC verification; however, the value Control Hamiltonian is averaged to negative one over the length of the maneuver – condition ensured by DIDO. As was predicted by the necessary conditions, $\bar{\lambda}_x$ and $\bar{\lambda}_y$ remain constant unless the path constraint covectors become active, when the vehicle is near an obstacle. In addition, $\bar{\lambda}_\theta$ remains a constant zero throughout the trajectory.

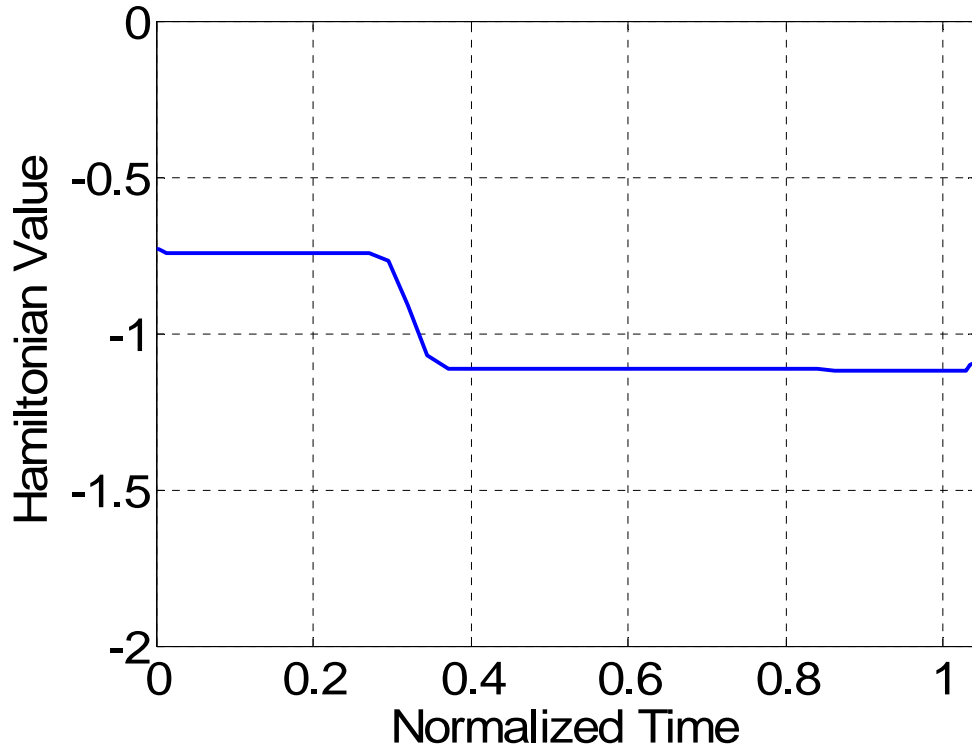


Figure 26 Control Hamiltonian Value for the Car Validation Problem.

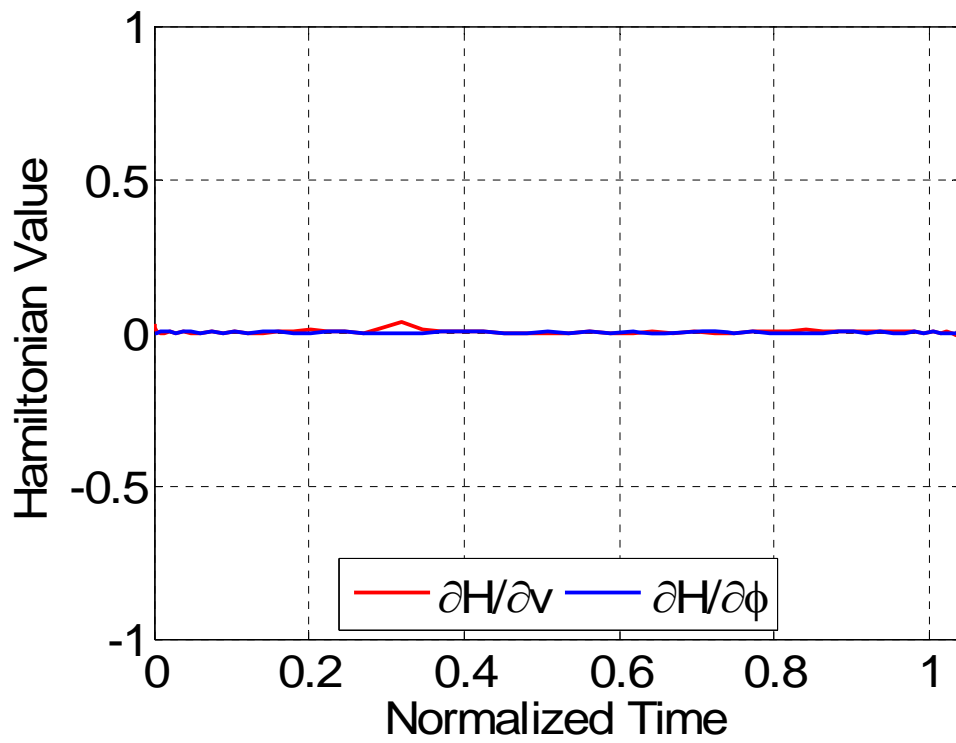


Figure 27 Hamiltonian Minimization Condition Verification for the Car Validation Problem.

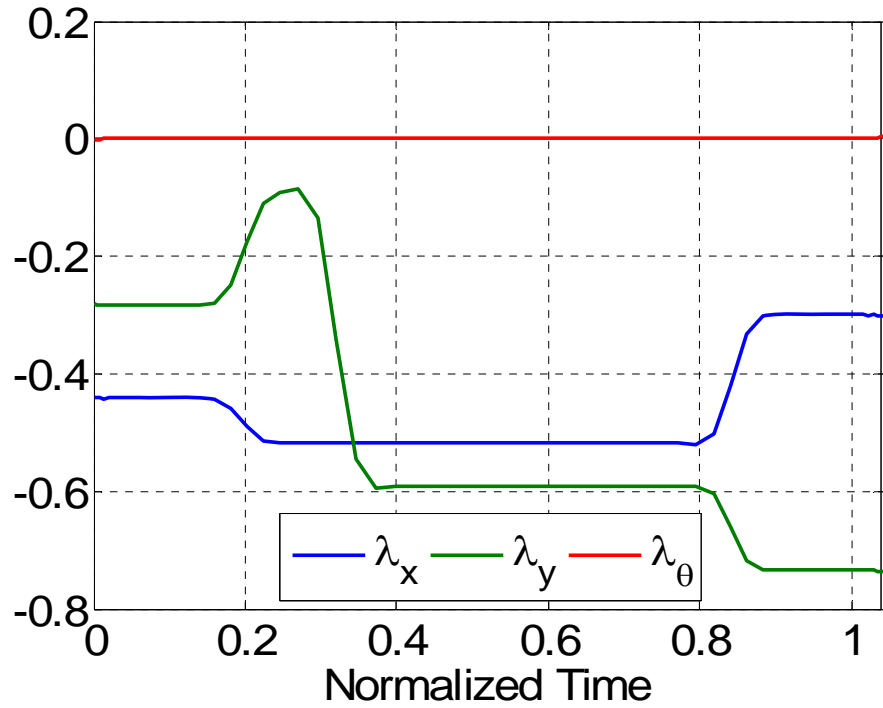


Figure 28 Costate Values for the Car Validation Problem.

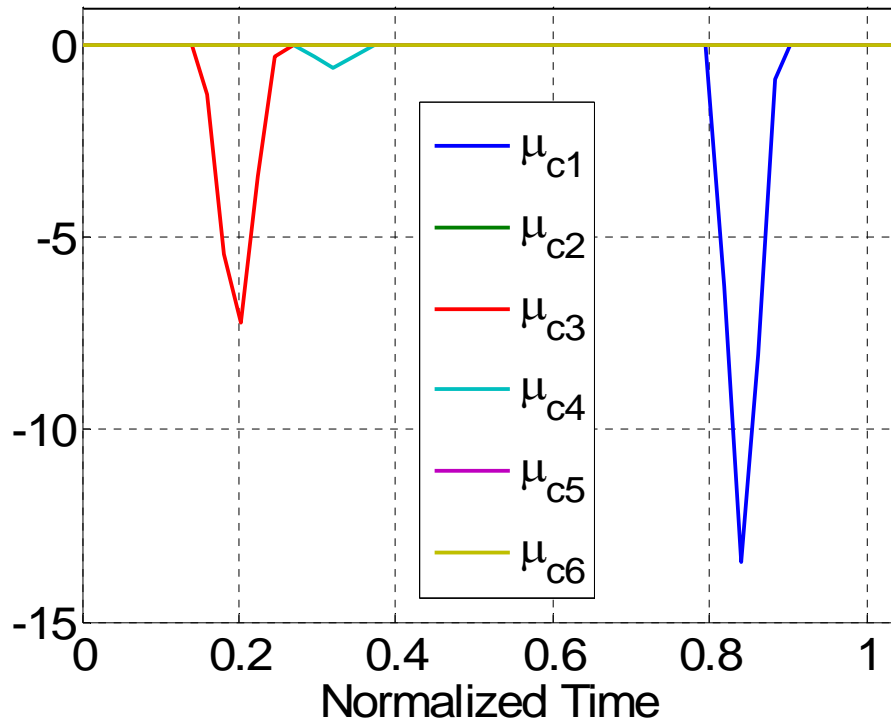


Figure 29 Path Constraint Covectors for the Car Validation Problem.

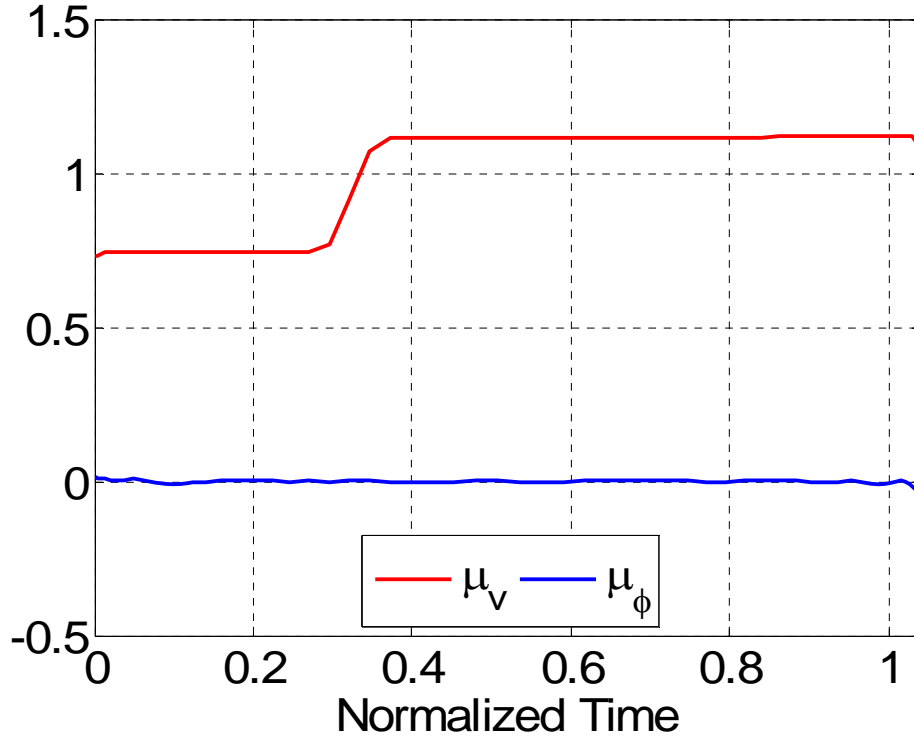


Figure 30 Control Covectors for the Car Validation Problem.

Figure 31 and Figure 32 display the state variables, propagated with a Runge-Kutta, algorithm, and compare them to the DIDO generated state variables. In each case, the propagated states are identical to the DIDO generated states within interpolation error; therefore, the kinematic trajectories generated by DIDO are deemed feasible.

Lastly, Bellman's principle was validated, accomplished by generating three sub-trajectories that initiate at points along the original optimal path and terminate at the endpoint manifold. For the sake of simplicity, the starting points for the sub-trajectories were chosen to be approximately the one-third, one-half, and two-thirds distance locations along the original path. Figure 33 shows that the original optimal kinematic trajectory and the subsequent optimal sub-trajectories all lay along the same path. Figure 34 further evidences the fact that each trajectory lies along the same path; in each case, the final cost was identical.

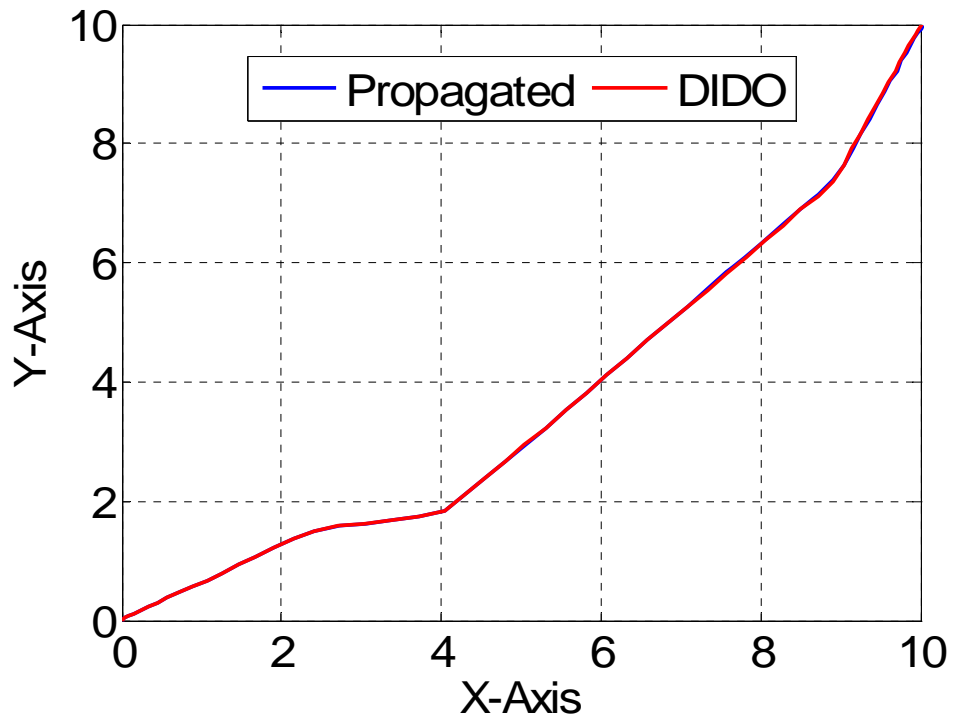


Figure 31 DIDO Generated and Propagated Positional Trajectories.

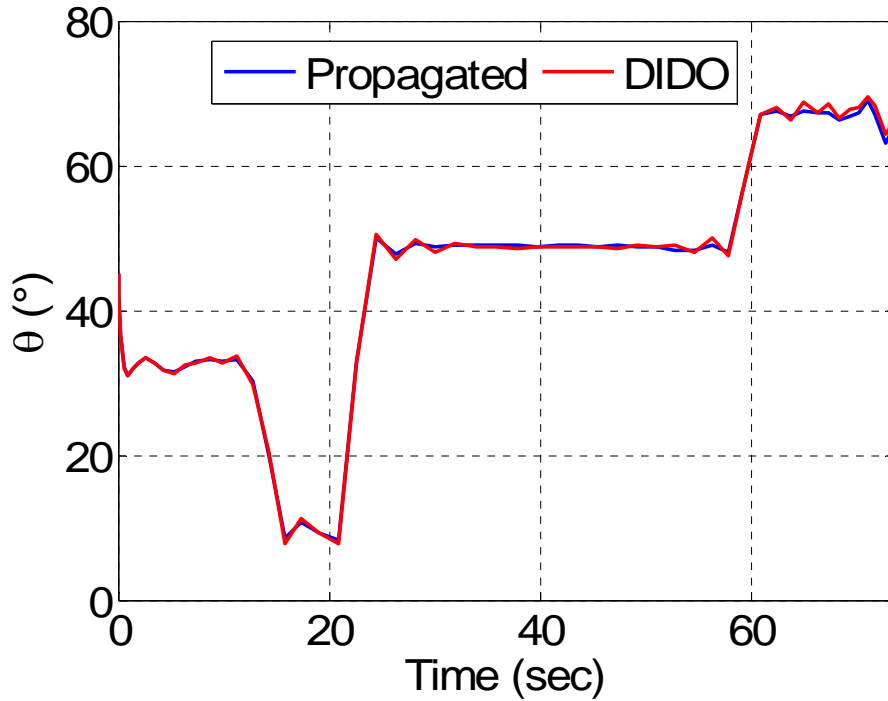


Figure 32 DIDO Generated and Propagated Angular Trajectories.

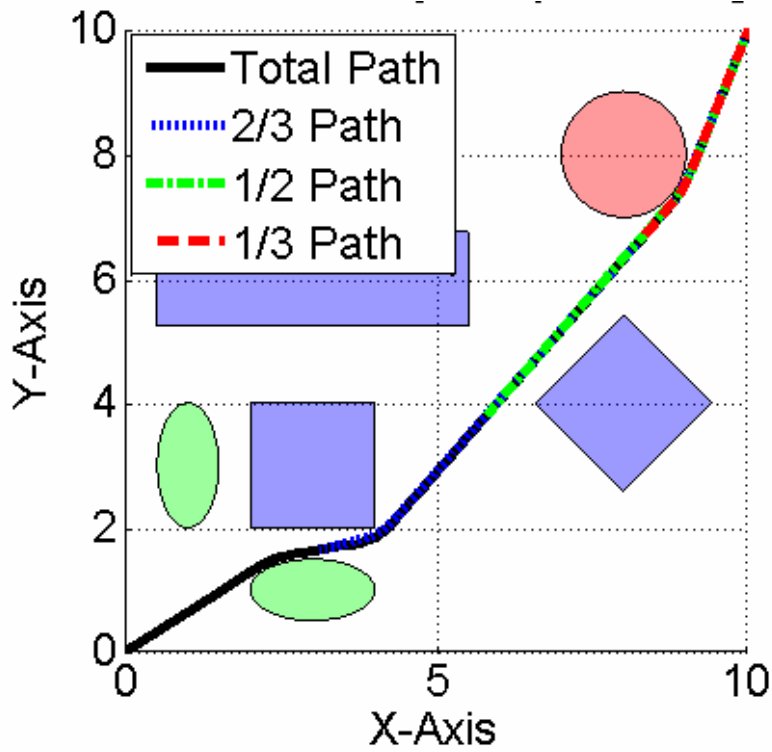


Figure 33 Verification of Bellman's Principle with Three Optimal Sub-trajectories.

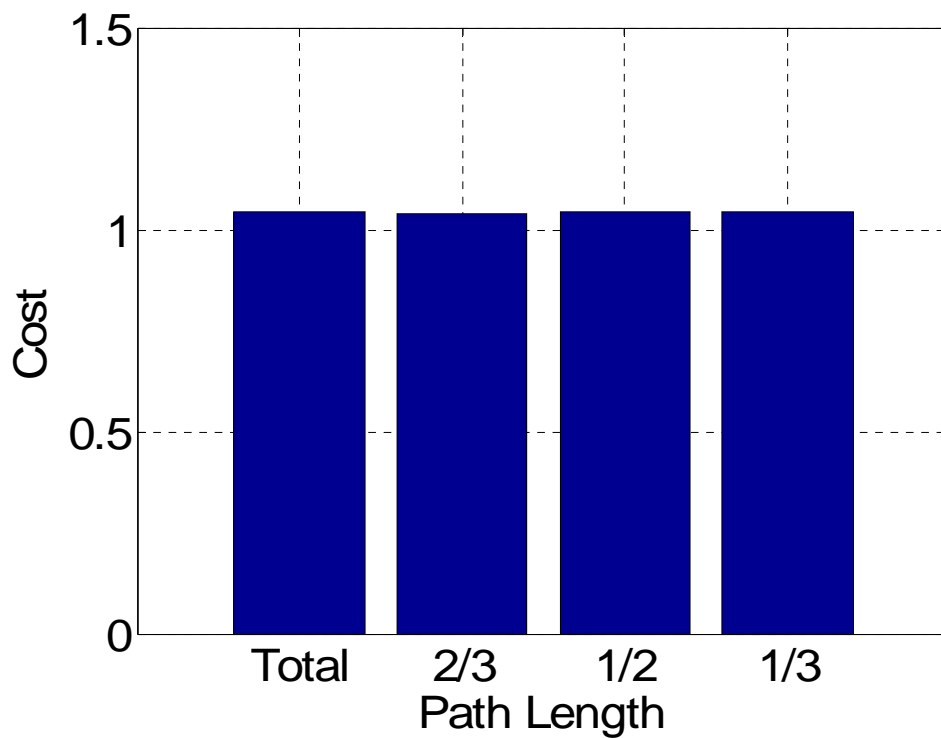


Figure 34 Verification of Bellman's Principle by Cost Comparison.

2. Results

Having proven the validity of the DIDO created optimal solutions, this subsection exploits the portability and capability of this method while solving the car problem under various, complex scenarios.

a. *Obstacle Rich Environments*

Obstacle rich environment problems only occur under the condition of global information. By having instantaneous knowledge of the entire workspace, the location of every obstacle is known prior to any exploration, and while this is thought to be an easier problem than that typically faced in robotics, it can create computational challenges for the planning algorithms. Many specific algorithms were created for the sole purpose of helping to maneuver through particular workspace obstacle configurations including cluttered obstacles, tight openings, spiral mazes, and potential field traps. In general, DIDO can handle these problems without the need of a guess, and DIDO solutions provide the benefit of kinematic satisfaction and optimality.

Figure 35 gives a time-optimal trajectory through 32 circular obstacles.

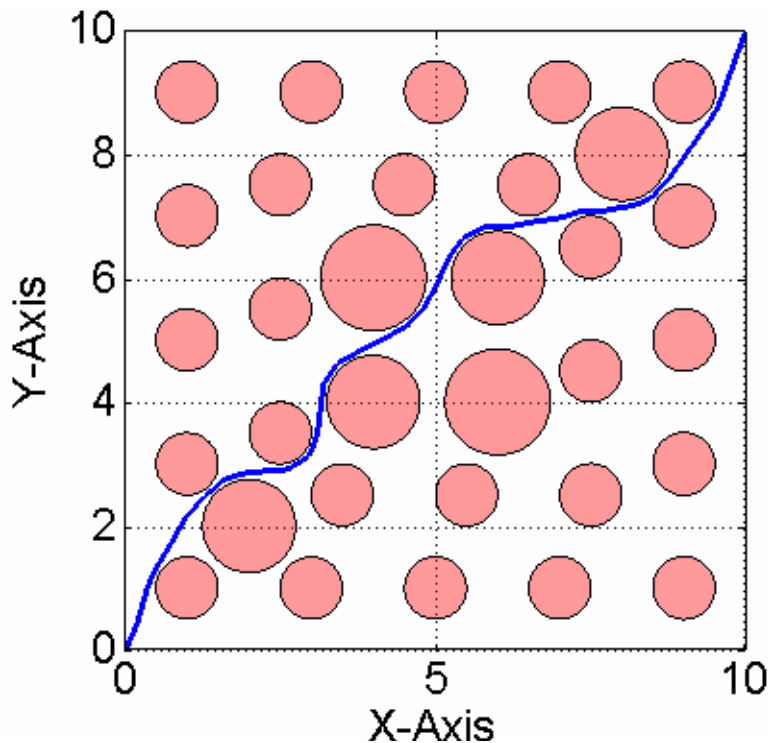


Figure 35 Time-Optimal Trajectory through 32 Obstacle Field.

Figure 36 includes a trajectory through a complex ellipse environment.

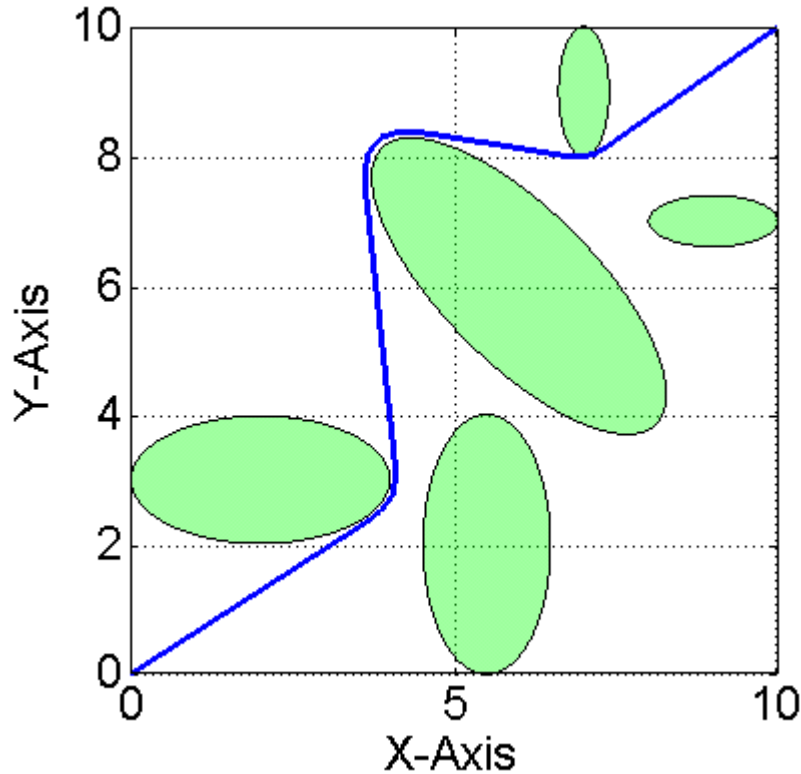


Figure 36 Optimal Trajectory through Complex Ellipse Environment.

In each of these cases, the first computation was a low-node solution, which was later used as a guess to a high-node solution. The initial guess given to the low-node solution involved only the initial and final positions.

b. Parallel Parking

Parallel parking creates an interesting challenge because the car will be maneuvering in a small space to position and orient itself correctly. The planning algorithm must be aware of the entire volume of the car to plan accordingly. In previous problems, the car has driven forward causing the front and rear axles to traverse nearly the same trajectory on the ground. For this reason, the x-y state representation of the mid-point of the rear axle and the fictitious expansion of the obstacles was sufficient to ensure that no collisions occurred. For the parallel parking problem, this was no longer sufficient, and one simple solution was the addition of a second point on the car from which to derive path constraints – the mid-point of the front axle. This approach did not

affect the system dynamics or boundary conditions, but it doubled the number of path constraints. Calculation of the mid-point of the front axle could be handled simply with the rear axle position, the length between the axles, and the orientation of the car.

Figure 37 shows picture frames from the traditional parallel parking problem where the car backs down upon the parking location.

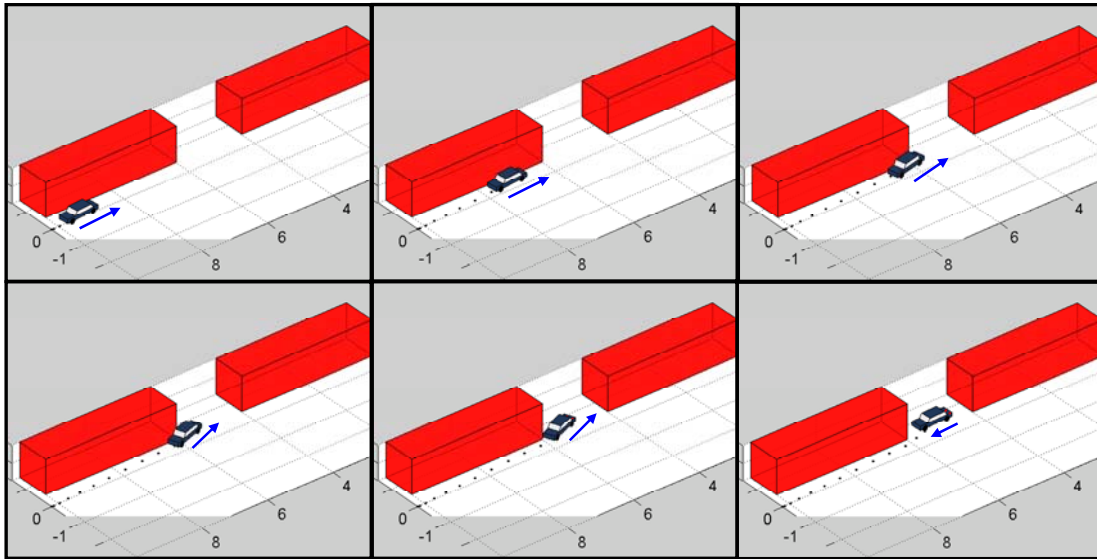


Figure 37 Time-Optimal Parallel Parking Maneuver; Location is Behind the Car.

Figure 38 pictures the same scenario; however, the illustration is now two-dimensional. From this illustration, it is apparent that the final vehicle position is not centered between the two obstacles; the reason for this is the fact that the state variables provide the position of the center-point of the rear axle.

The maneuver requires only two changes in steering wheel direction. The wheels are first oriented to drive the back of the car into the space, and then the steering angle is reversed to get the front wheels into the space. This maneuver requires no reorientation since it takes advantage of the fact that the steering wheels are in the front of the car. Also the vehicle does not proceed to the space by moving parallel to the obstacles, which is common in normal driving; however, people

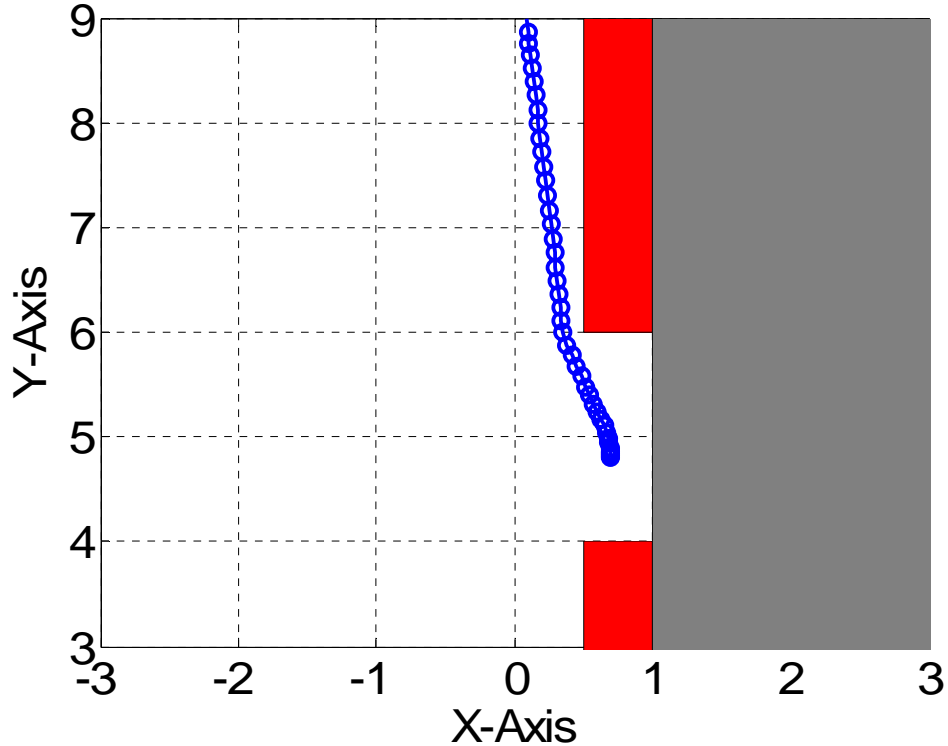


Figure 38 Two-Dimensional View of Backward Parallel Parking Maneuver.

Figure 39 and Figure 40 shows the time-optimal parallel parking maneuver when the problem is initiated from below an open parking spot. Typical real-world driver actions include pulling in front of the space so that the problem is presented as it was previously. This simulation shows that the redefinition of the problem is not necessary and reduces time optimality; however, the trajectory is more complex. As seen below, the parallel parking maneuver could not be accomplished without reorientation of the car to attain the proper lateral position within the parking space, i.e. four steering directions were utilized. Most drivers prefer to park with as little complexity as possible; therefore, it is not surprising that the majority of them approach the problem in the first manner. In robotics, it is not necessary to be concerned with trajectory complexity or standard problem representation; rather, path optimality should be the goal. This problem evidences a situation where non-conventional problem representation can actually simplify solution attainment by not requiring an algorithm to position the correctly to back down upon the parking spot.

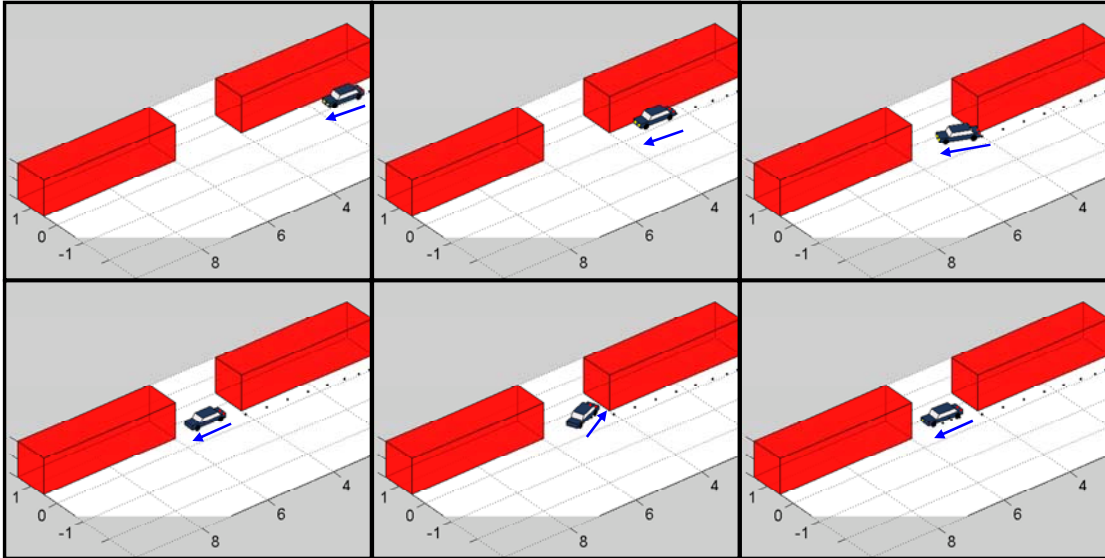


Figure 39 Time-Optimal Parallel Parking Maneuver; Location is in Front of the Car.

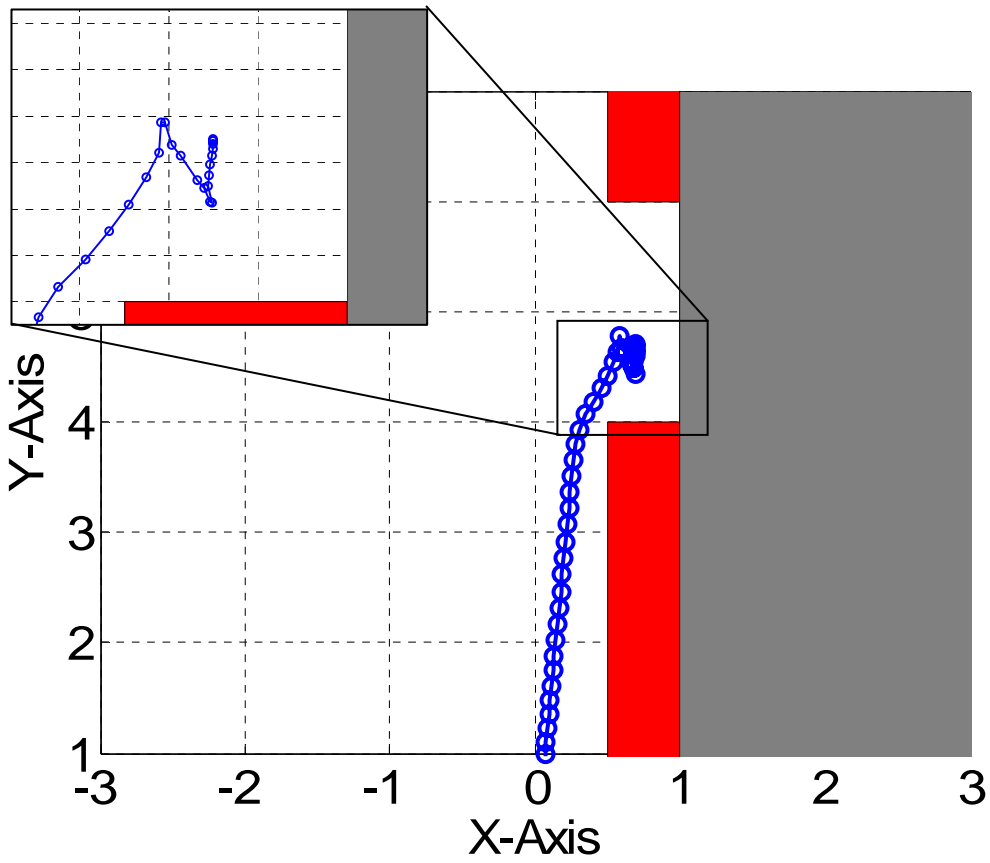


Figure 40 Two-Dimensional View of Forward Parallel Parking Maneuver.

Figure 42 shows the same trajectory, but it does so from the perspective of a camera and strobe light. This view allows the observer to better visualize the trajectory taken by the car as it attempts to avoid the moving obstacles.

This scenario evidences the fact that optimal control solutions are ideal to solving problems with dynamic environments; the problem statement changes little with the addition of moving obstacles. During the real-world implementation of this technique with moving obstacles, the challenge primarily shifts to sensor identification and tracking of obstacles so that adequate information can be passed to the path planner.

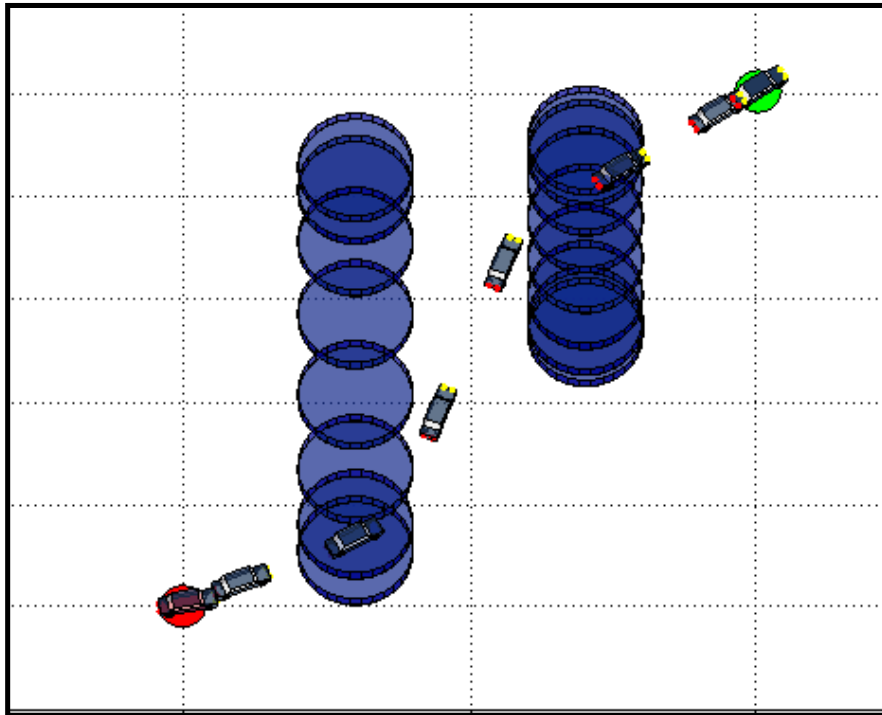


Figure 42 Time-Optimal Trajectory with Moving Obstacles, Strobe Picture.

d. Moving Target

A similar experiment was conducted using a dynamic endpoint manifold. For this problem, the obstacles remained stationary but the final position was moving. This condition also requires minimal alteration within the optimal control problem framework for resolution. In the case of the time optimal problem, the Endpoint Lagrangian adds a new term explicit with the final time – problems that do not optimize

final time would not previously have any terms. This addition does not change the fact that the Lagrangian of the Hamiltonian should be constant, but it does mean that it will have a different final value, from the Hamiltonian Value Condition.

Figure 43 and Figure 44 display the results in a manner similar to the dynamic obstacle problem figures. While this problem solution satisfies the necessary conditions for being an extremal solution, Figure 44 presents visual evidence that portends a solution that would be globally optimal. Interestingly, the extremal trajectory travels between the two obstacles to reach the final position; intuition would lead the observer to believe that the globally optimal solution would travel below both obstacles and reach the final position in a more expeditious manner. To explain why this occurred, it is necessary to understand that the initial guess given to DIDO involved only two points. The first point is the starting position and the final point was where the endpoint manifold would be at a time long after calculated maneuver time.

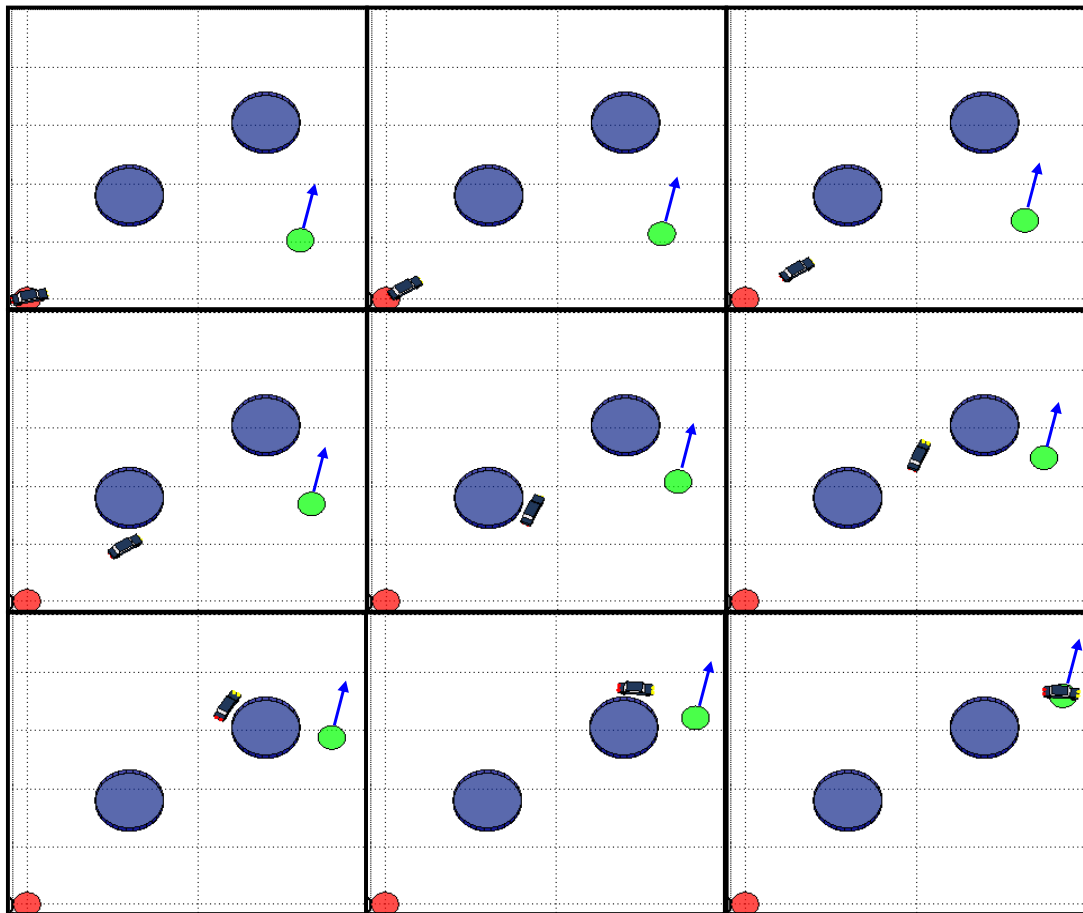


Figure 43 Time-Optimal Trajectory for Moving Target.

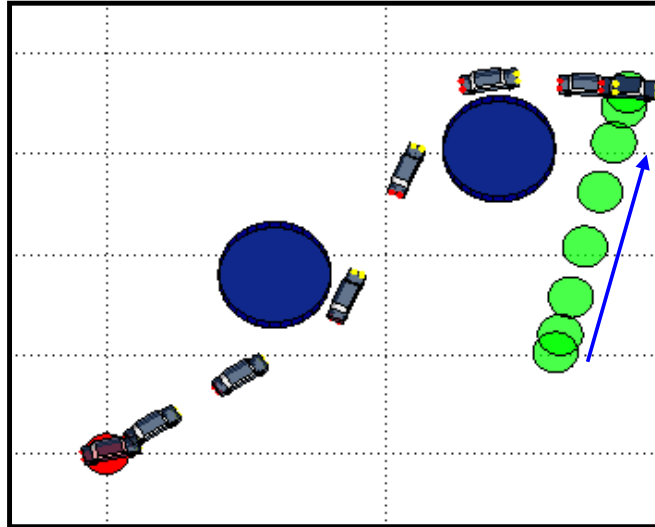


Figure 44 Time-Optimal Trajectory for Moving Target, Strobe Picture.

To evaluate how the guess affects the optimal solution, the problem was run again with a different initial guess. Figure 45 and Figure 46 show the optimal trajectory when the initial guess uses a different second point. For this simulation, the second point of the initial guess was the position of the endpoint manifold at the point in time of the previous maneuver's completion. The result is more intuitive and less costly maneuver than the previous solution, and the trajectory satisfies the necessary conditions for being an extremal path. The difference in trajectory cost between the first and second solutions was approximately 12%; therefore, the second path was also far more numerically optimal.

This example evidences two important facts concerning optimal control methods. First, the solutions are not necessarily globally optimal; even verification of the necessary conditions ensures only that the path is extremal. Second, the initial guess plays a significant role in the result achieved. Overall, the most important aspect of applying optimal control methods to robotics problems is the generation of a feasible solution; in the very least, the resulting trajectories are extremal and feasible. All other path planning methods create only feasible trajectories; therefore the results shown here provide an improvement to the other techniques.

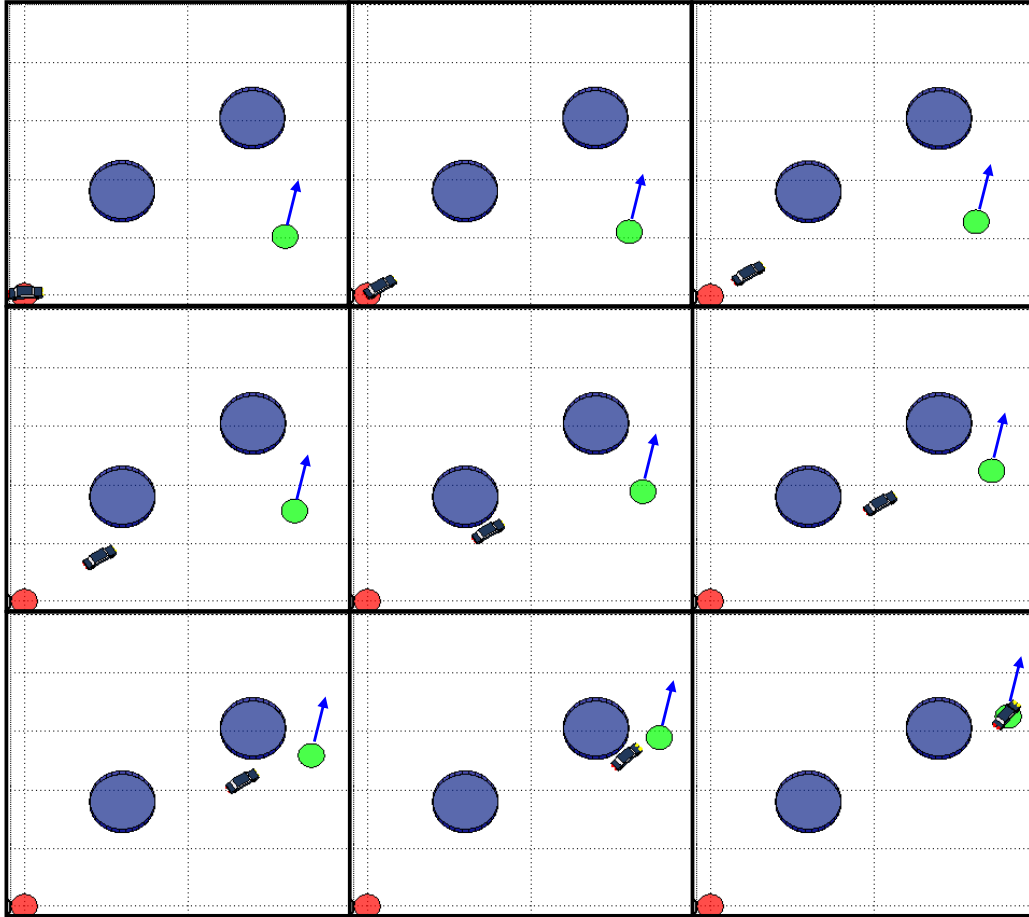


Figure 45 Second, Time-Optimal Trajectory with Moving Target.

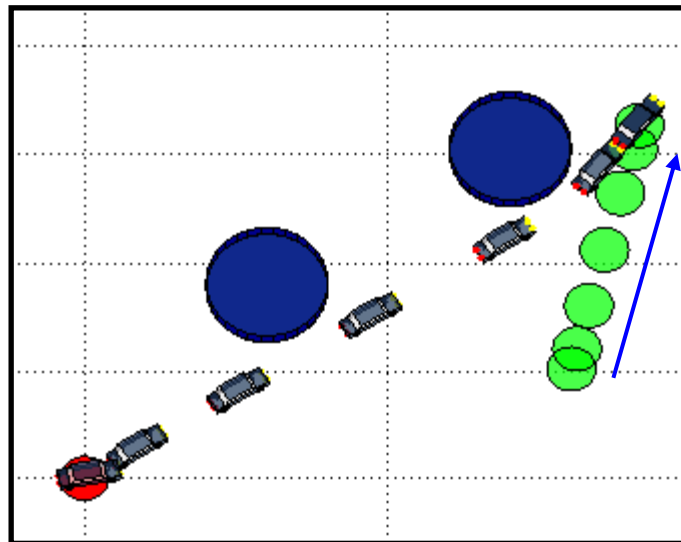


Figure 46 Second, Time-Optimal Trajectory with Moving Target, Strobe Picture.

e. Actuator Failures

Optimal control methods offer advantages for dynamic environments but they also prove useful when actuator failures limit the control capability. If ever the steering wheel gets stuck or the car can only move backwards, a new optimal trajectory can be calculated that takes this problem into account. This ability enables greater autonomy by allowing vehicles to continue activity despite a failure.

Figure 47 pictures an optimal trajectory when the car can only turn left, i.e. the steering angle is limited between two positive values. Two types of motion occur during this maneuver. The first type involves the car driving forward and using the minimum steering angle, producing the maximum turning radius. The second type of motion occurs at the end of each forward segment. At that time, the car stops its forward motion and then backs up as it is using a maximum steering angle. This motion allows the car to reorient in the minimum amount of distance, the minimum turning radius. Therefore, the optimal trajectory is composed of forward motion maximal turning radii and backward motion minimal turning radii.

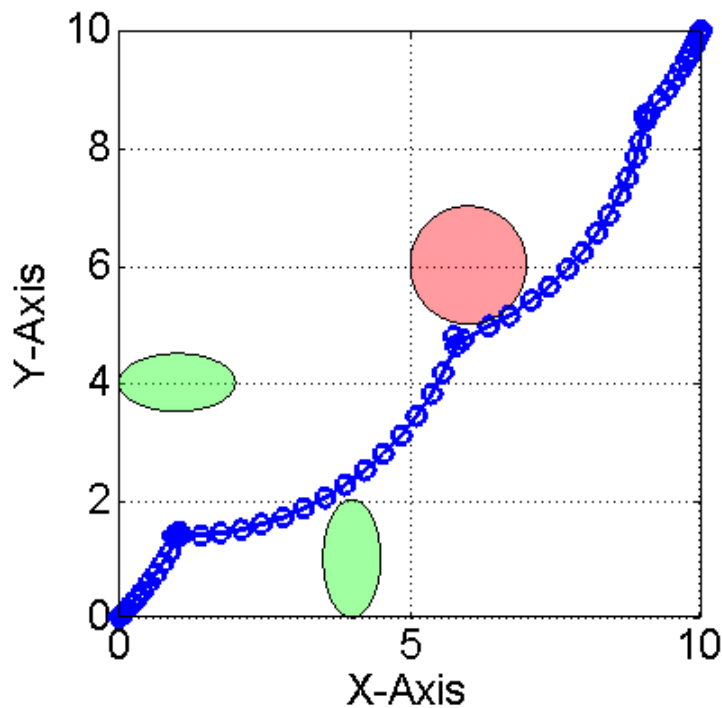


Figure 47 Optimal Trajectory during Control Failure, Must Turn Left.

B. CLOSED LOOP

1. Motivation

The previous section demonstrated the capability of optimal control methods in developing open-loop solutions that are both feasible and optimal. Application of these methods directly to a car robot would require one of two modifications to synthesize the control signals. Vehicular control signals could be generated either by the use of a trajectory following subsystem to track the trajectory created by DIDO or directly from the output of DIDO by changing the problem definition to include the control signals as the controls. While both of these methods have merits, they also have experience limitations in regard to real-world implementation. These methods do not operate effectively if only local information is available, if the vehicle model is not accurate, if sensor data is not perfect, or if external, unforeseen forces act on the vehicle. Each of these is an example of uncertainty and the only means of managing uncertainty is feedback.

The focus of the following research is to provide precursory analysis of optimal control application to robotics platforms, and the eventual goal is the successful integration of these techniques into real-world platforms to prove their significance. The only means by which this can be accomplished is through closed-loop application of optimal control methods, and that problem is the focus of the remainder of this section.

2. Problems with Closing the Loop

Closed-loop implementation of optimal control methods is similar in many aspects to testing Bellman's principle; in fact, if there was no uncertainty, then closed-loop optimal control would only prove Bellman's principle. With the existence of real-world uncertainty, the vehicle does not exactly follow the predicted path; therefore, additional optimal solutions do not necessarily parallel the previous.

The first requirement needed to conduct simulations is a means by which to model problem uncertainty. Numerical errors partially mimic the uncertainty seen in real life. By interpolating the control solutions generated by DIDO and propagating the results with a Runge-Kutta algorithm, the vehicle's future position can be predicted at any time, but this will never agree precisely with the DIDO position. The difference between the DIDO generated optimal solution and the propagated result was chosen to

represent problem uncertainty. The interpolation technique, i.e. cubic, spline, etc., is a critical decision in the modeling process and it can significantly alter the result. While it is typically desirable to employ the technique that most closely captures the control trajectory, occasionally it was useful to use a different method to analyze the ability of the controller to overcome considerable uncertainty. Generally, numerical error is not likely equivalent to model, sensor, and navigation error; therefore, the assumption made here to simulate all uncertainty as numerical error will require more future analysis.

Incorporating problem uncertainty leads to a significant dilemma in utilizing optimal control methods for closed-loop control. In general, optimal solutions touch path constraints or obstacles since optimal solutions necessitate extremes. This solution, however, is no longer desirable when problem uncertainty is considered. During simulation, uncertainty can cause the vehicle to move into a path constraint that it was previously touching; this forces an infeasible problem since the vehicle's position is not in a valid location. The real-world equivalent of this concern is that the vehicle will run into a wall.

The final concern associated with a closed-loop architecture is the ability to update the solution with a high enough frequency to complete the problem successfully. Naturally, precise system models and high node solutions would rarely require updates, but they require lengthy computational periods. The lengthy computational period is acceptable only if the system models are understood to a high precision, i.e. little uncertainty. On the other hand, low-node solutions support expeditious computations and due to the frequency of update, they have the ability to support imprecise models and high uncertainty. Logically, the concern with using low-node solutions is the inability to capture fine details; however, DIDO node placement and frequent recalculation generates a final solution with much greater resolution than that of an open loop trajectory.

3. A New Problem Definition

The inherent contradiction between time or distance optimal trajectories and uncertainty tends toward a new definition of optimality in the face of real-world application. Uncertainty dictates the use of solution feasibility and safety as key measurements of solution optimality for this problem. Accomplishing this task without sacrificing all the advantages afforded by the time or distance optimal solution is the key

challenge of the remainder of this section. After all, requiring only that a solution provide a feasible and safe solution is within the realm of several other planning techniques.

4. Solution Approaches

Several concepts were developed to balance time optimality and uncertainty management; they will be presented below. The following subsection presents the results of simulations involving the methods shown here. Control trajectory interpolation and state propagation were used to represent actual system motion. The basic block diagram of the control simulator is shown in Figure 48. The DIDO generated pre-run trajectory serves the purpose of guess generation for the subsequent online calculations, this permits low calculation times and the use of low-node solutions. Each online DIDO calculation provides an optimal control trajectory and supplies guess information for the ensuing calculation.

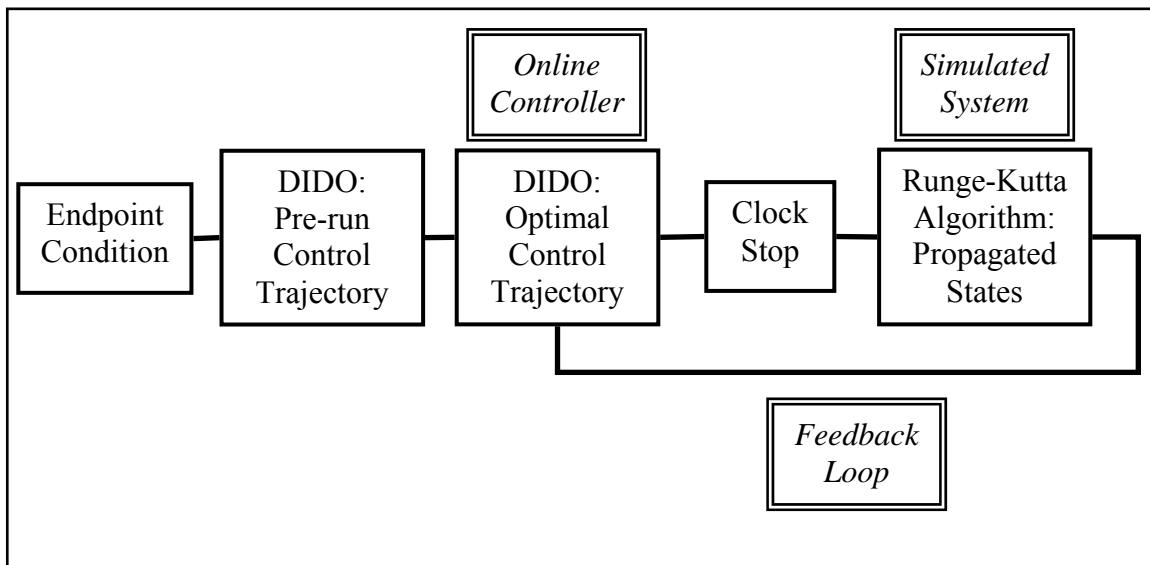


Figure 48 Block Diagram of the Basic Closed-loop Optimal Control Architecture.

In order to evaluate each method based only on the generated solution, the computation time was fixed to one second; therefore, actual system motion was permitted for exactly the one second between control solution updates regardless of the computation time – evidenced by the clock stop above. After problem completion, the

actual computational times were evaluated, but initially the primary focus was on the ability to generate successful closed-loop trajectories in the face of uncertainty. A more thorough evaluation of computation time concerns and solutions is handled in a later subsection. Additionally, the one-second computational time was chosen with arbitrariness, and it was not known a priori if that update frequency would be sufficient to complete the problem.

The first closed-loop approach removed the path constraints and added obstacle avoidance as a portion of the cost function. This approach shares some similarities with the potential functions and for that reason was referred to as the Penalty Function approach. Optimality was maintained by limiting the permissible maneuver time. For example, a pre-run, open-loop computation with path constraints yields an optimal maneuver time, t_f^* . Following the pre-run calculation, the path constraints are removed, the cost function is changed to promote obstacle avoidance, and the maximum maneuver time is set to $t_f^* + \Delta t$, where Δt represents the small percentage of time beyond the optimal that is permitted for the new maneuver. In theory, this approach preserves time optimality while enabling greater separation from the obstacles.

Implementation of this technique first requires the development of a desirable cost function. The goal of the cost function was to yield an infinite cost inside the obstacle and zero cost at a minimal distance from the obstacle. One general limitation of current optimal control techniques is the need to provide inputs that are continuous and smooth. With that limitation in mind, Equation (31) was developed as a double exponential equation using the previously derived p-norm shapes to generate the desired obstacle cost.

$$h_i(x, y) = \left(\frac{(x(t) - x_c)}{a} \right)^p + \left(\frac{(y(t) - y_c)}{b} \right)^p - c^p \quad (31)$$

$$F_{pfc} = \left(e^{-h_i(x, y)} - 1 \right)$$

Table 2 demonstrates how the double exponential equation generates the desired costs based on the vehicle position and $h_i(x, y)$, the p-norm obstacle shape formula displayed in Equation (31) and exemplified in Figure 4.

Table 2 Mapping of Vehicle Position to Shape-Based Penalty Function Value.

	Inside Obstacle	On Obstacle	Outside Obstacle
$h_i(x, y)$	< 0	$= 0$	> 0
$-h_i(x, y)$	> 0	$= 0$	< 0
$e^{-h_i(x, y)}$	> 1	$= 1$	< 1
$e^{e^{-h_i(x, y)}}$	$\rightarrow \infty$	$= e$	$\rightarrow 1$
$F_{PFC} = e^{e^{-h_i(x, y)}} - 1$	$\rightarrow \infty$	$= e - 1$	$\rightarrow 0$

When the vehicle is traversing a map with multiple obstacles, only the closest obstacle should affect the cost function and the resulting vehicle motion. Mathematically, the maximum value of a set of individual obstacle costs cannot be selected; this mathematical operation creates non-smoothness. Instead, the p-norm is used once again; this time to isolate the largest cost of all the workspace obstacles. This concept is evidenced in Equation (32). Small values of the exponent can be used in this case without a significant loss in accuracy.

$$F_{PFC} = \left(\left(e^{e^{-h_i(x, y)}} - 1 \right)^p + \left(e^{e^{-h_{i+1}(x, y)}} - 1 \right)^p + \left(e^{e^{-h_{i+2}(x, y)}} - 1 \right)^p + \dots \right)^{1/p} \quad (32)$$

Figure 49 and Figure 50 show the effect of applying this cost function to a simple map with three circular obstacles, and they evidence the numerical maps described by Table 2. The contour plot, Figure 49, also shows the exact size of the obstacles, displayed in green. The resulting obstacle costs are not non-smooth but are close.

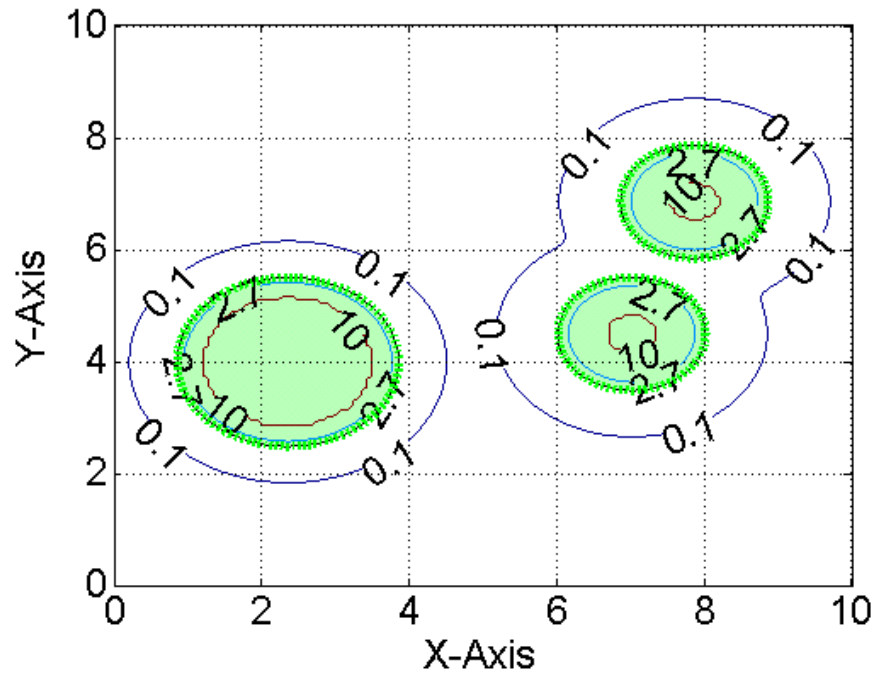


Figure 49 Contour Plot of Penalty Function for a Simple Three-Obstacle Map.

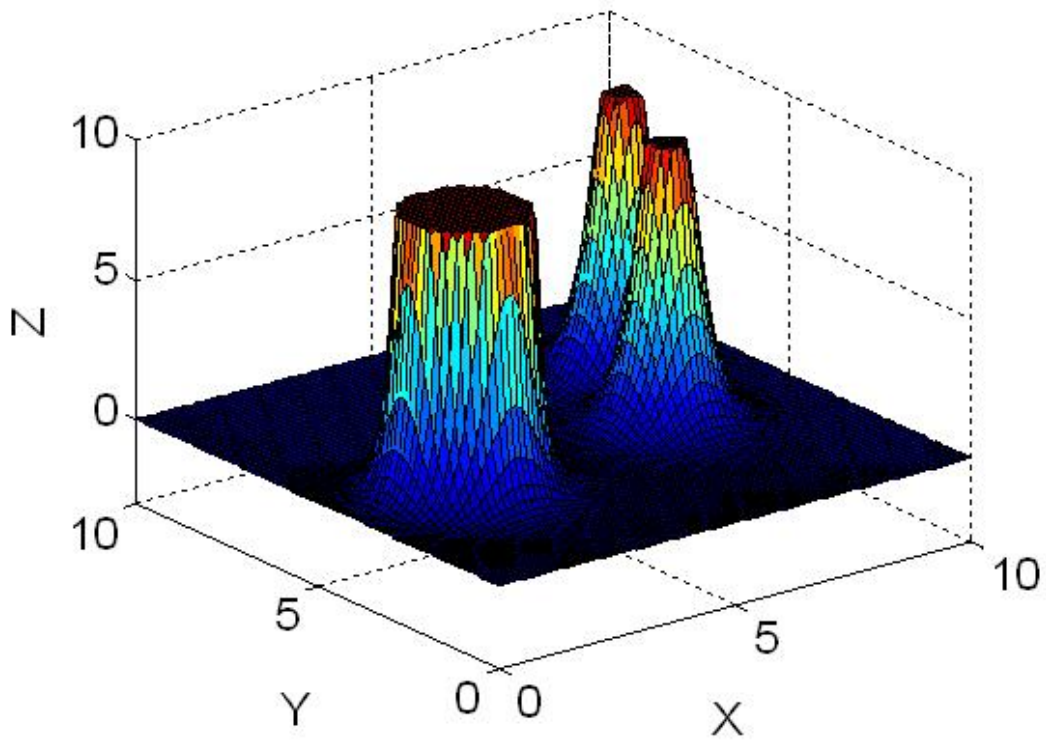


Figure 50 Surface Plot of Penalty Function for a Simple Three-Obstacle Map.

The concern with using the Penalty Function approach is the inability to ensure the generation of a feasible trajectory mathematically. Without the use of path constraints, it is technically feasible to generate a trajectory that travels through an obstacle. For problems similar to the three-circular obstacle setup presented above the probability of such an occurrence is not high because the workspace is uncluttered. The map problem introduced in the Tricycle section, Figure 51, is a good example of a problem that can create complications for the Penalty Function approach.

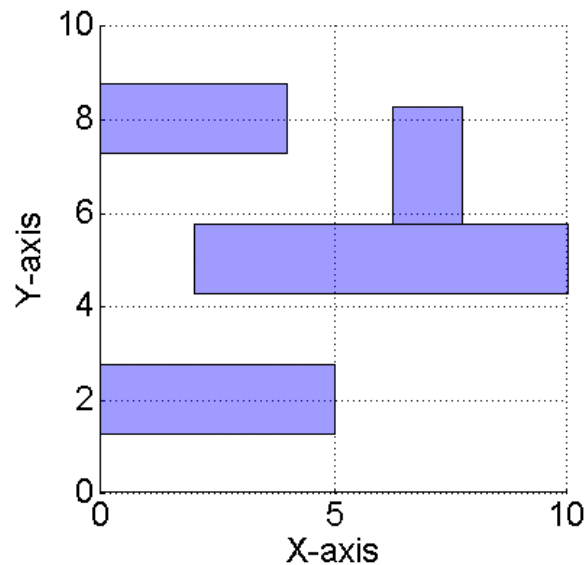


Figure 51 The Maze Problem First Introduced in the Tricycle Section.

Figure 52 shows the maze from the perspective of the Penalty Function function presented in Equation (32). The disadvantages of this approach are evident. The potential field is a shape-based representation of the obstacle, and the cost is based upon the vehicle position with respect to the geometric center of the obstacle. While this is effective for shapes with similar dimensions in both the x and y directions such as circles and square, it is ineffectual for shapes with one dimension longer than the other, i.e. rectangles and ellipses. This situation provides evidence to the importance of correct problem formulation and representation. Exact maze representation is only accomplished with a discontinuous cost function or path constraint; however, optimal control problems require continuous and differentiable inputs.

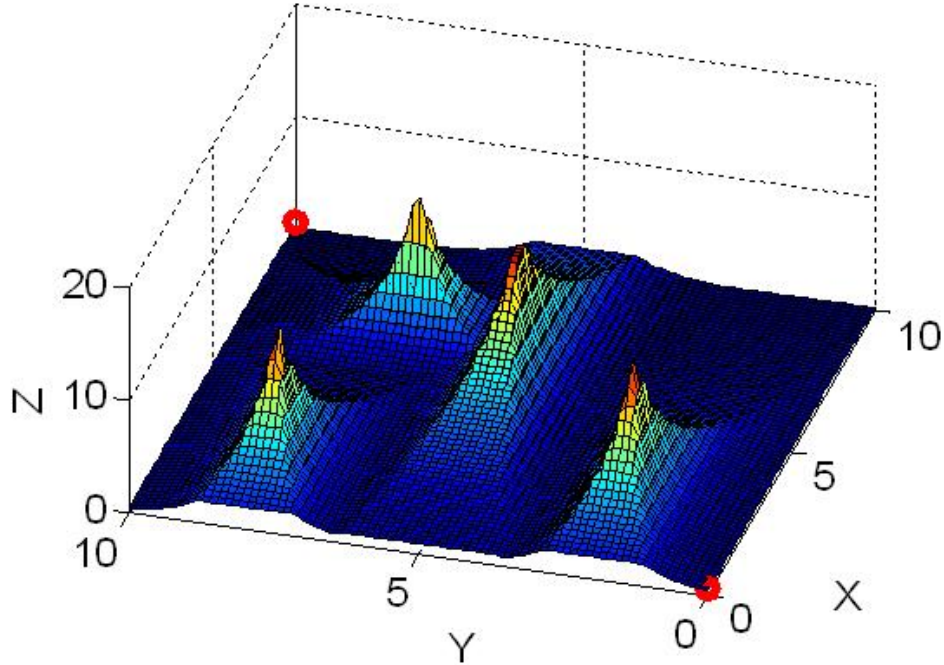


Figure 52 Maze Problem Visualized Through the Penalty Function Function.

The result of the shape-based representation is a high cost only at the geometric center of a symmetric obstacle; this leaves many avenues for the vehicle to plan a real-world infeasible trajectory. A solution to this situation is obstacle representation in a manner similar to the polygonal shape function shown previously in Equations (10) - (12). That technique created an obstacle shape out of the intersection of multiple line segments, each represented by a function $g_i(x,y)$. The polygonal shape method determined obstacle interference by a path constraint function, $h_{p,i}(x,y)$, value less than zero. To make a satisfactory cost equation, the original constraint function was multiplied by negative one and then used as an exponential, Equation (33). The p-norm is used in the cost function so that the vehicle cost at a particular location is the result of the largest single obstacle cost and not the aggregate of all obstacle costs.

$$h_{p,i}(x(t), y(t)) = \ln \left[\left(e^{-g_1(x)} \cdot p + e^{-g_2(x)} \cdot p + e^{-g_3(x)} \cdot p \right)^{1/p} + \dots \right] \quad (33)$$

$$F_{PFC,P} = \left[e^{-h_i(x(t),x(t)) \cdot p} + e^{-h_{i+1}(x(t),x(t)) \cdot p} + \dots \right]^{1/p}$$

Table 3 shows how the vehicle positions map to cost values in the new cost function.

Table 3 Mapping of Vehicle Position to Polygonal Shape Penalty Function Value.

	Inside Obstacle	On Obstacle	Outside Obstacle
$h_i(x, y)$	< 0	$= 0$	> 0
$-h_i(x, y)$	> 0	$= 0$	< 0
$F_{PFC,P} = e^{-h_i(x,y)}$	> 1	$= 1$	< 1

The polygonal shape mapping is similar to the p-norm shape mapping, but the difference is that resulting polygonal shape cost is not referenced to the geometric center of an obstacle. Rather, it is referenced to the line segments that constructed the shape; therefore, it is more portable to any workspace obstacle. The polygonal method can also account for the walls of the workspace, which would normally be a part of any maze. Figure 53 presents the maze problem represented with the new polygonal shape cost function; the walls were also included. The representation of the obstacles is improved with this technique over the previous. The initial and final positions were moved to [1, 1] and [9, 9] so to not interfere with the walls.

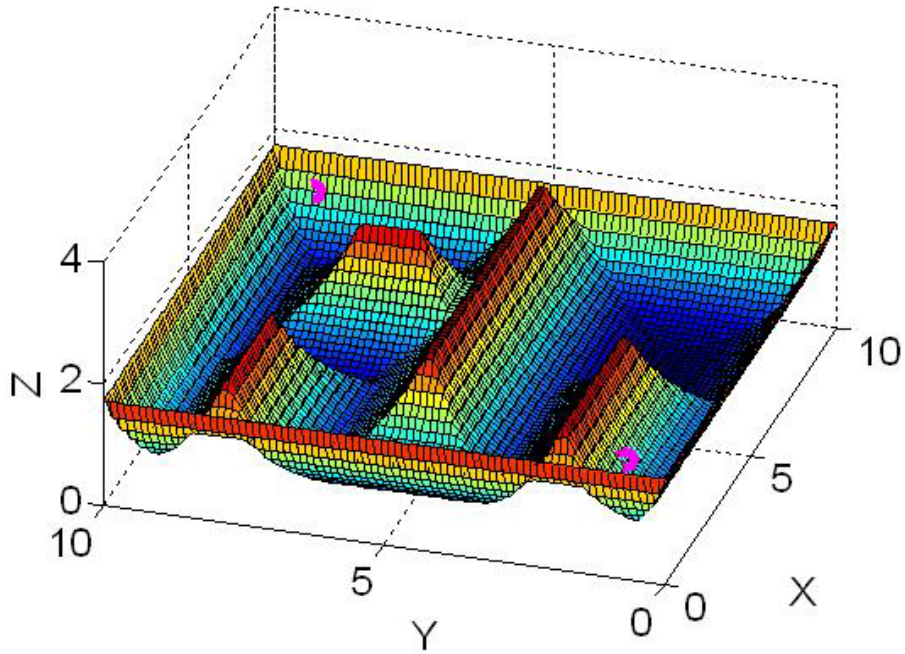


Figure 53 The Maze Problem Represented with the Polygonal Shape Penalty Function.

Using the Penalty Function concept the obstacles can be represented using either the p-norm shapes or with line segments intersecting to create a polygonal shape. Both methods can actually be used interchangeably within one cost function depending upon the representation needs. Experimentation with this technique is analyzed in the following subsection.

The second approach to solving the closed-loop robotics problem is anecdotal in nature and its creation was motivated out of a desire to keep the problem formulation unchanged. Naturally, sensor information pertaining to an obstacle includes uncertainty and the degree of uncertainty increases with the range to the obstacle. Making use of this concept, the fictitious obstacle extension or obstacle buffer, which was previously used to allow room for the volume of the vehicle to pass, can be altered from a constant to a range-dependant value. When the vehicle is far from the obstacle, the buffer is larger than it is when the vehicle is closer, just as the size and location uncertainty of an obstacle decreases as distance to the obstacle decreases, Figure 54. The parallelism between the buffer size and sensor error resulted in the name, “Resolving Uncertainty Approach.” This decreasing buffer around the obstacle permits error during vehicle motion since a vehicle position, which could violate a path constraint in one control iteration, would place the vehicle outside the obstacle in the next iteration. Of course, the buffer cannot decrease infinitely and the control update frequency must be adequate to correct the errors before impact with the actual obstacle occurs.

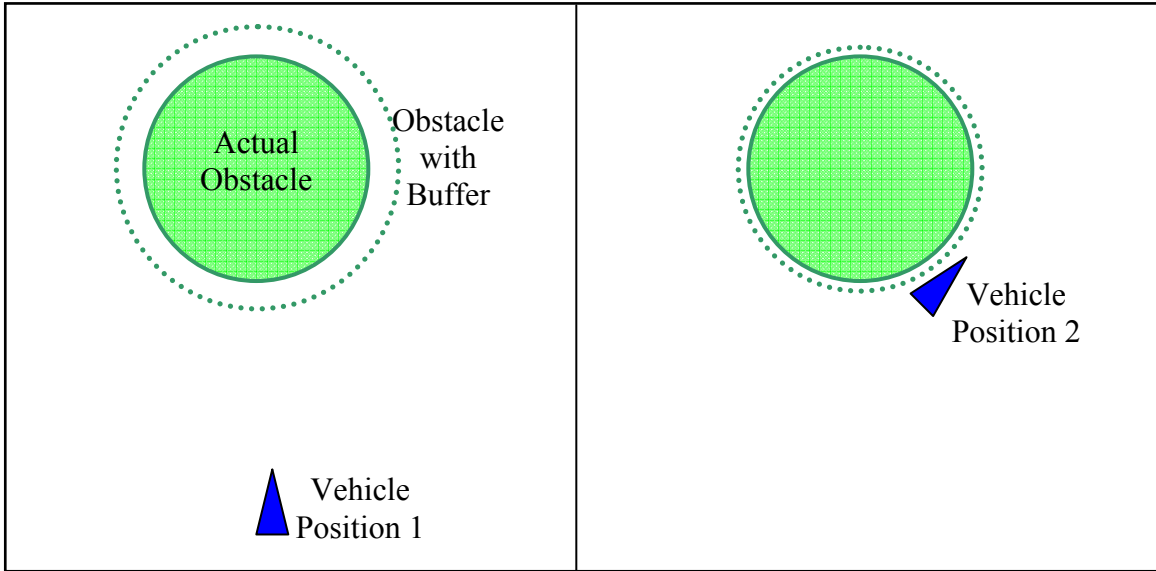


Figure 54 Example of Range-Dependent Buffer around Obstacles.

The risk of uncertainty causing the vehicle to hit the obstacle or generating an infeasible position is only substantial near an obstacle. For this reason, it is not necessary and it is undesirable for the buffer size to change drastically when the vehicle is far away. For example, increasing the buffer size linearly with distance would create an optimal trajectory that is far from optimal with respect to the actual obstacle size. The purpose of the buffer is not to change the trajectory significantly but to ensure safety. For this reason, it was desirable to have a buffer size whose growth slowed with increasing distance, and this led to the use of a fractional exponent. The basic equation adopted for use with the range-dependent buffer is shown in Equation (34) and Figure 55 illustrates how the buffer size changes with respect to distance.

$$d_{buffer} = a_1 \cdot (d_i + a_2)^{1/a_3} + a_4 \quad (34)$$

In this equation, d_{buffer} is the distance of the buffer from the edge of the actual obstacle, d_i is the distance from the vehicle to the obstacle, and a_1 through a_4 are all constants affecting the buffer characteristics. Figure 55, a plot of the buffer values as they vary with vehicle distance, displays that the majority of buffer change occurs near the obstacle, and a minimum buffer value exists so as to minimize the risk of physical collision.

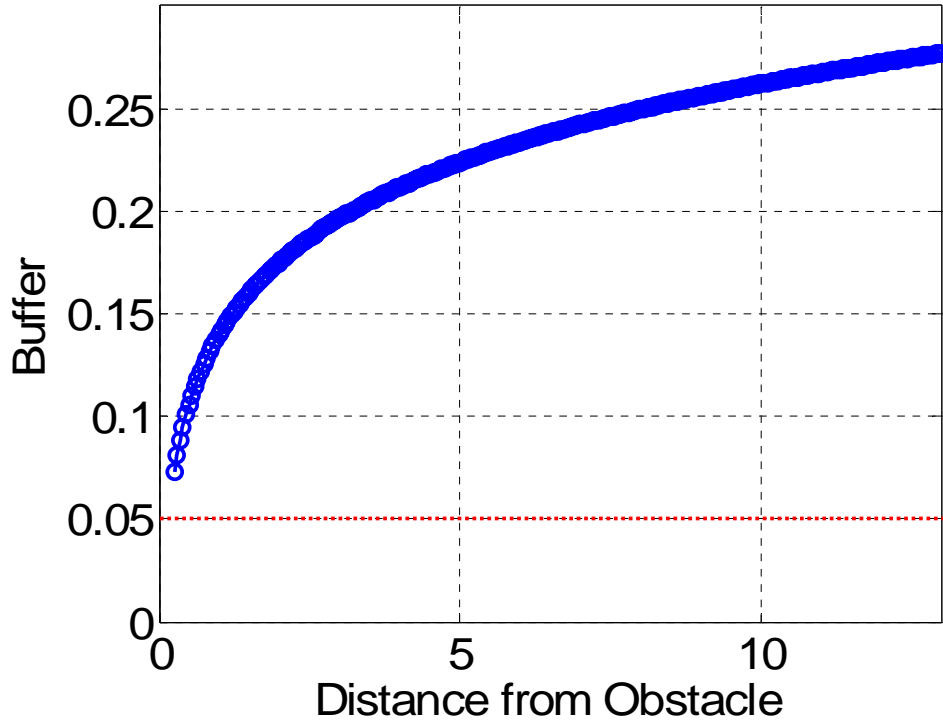


Figure 55 Illustration of Relationship between Buffer Size and Distance to Obstacle.

5. Results

a. Penalty Function

The Penalty Function approach was first tested in an open-loop. The first step in this method is computation of the time optimal trajectory so that the maximum allowable Penalty Function maneuver time can be calculated from the optimum trajectory time. The percentage of the optimum trajectory time given to the new maneuver significantly impacts the outcome. Figure 56 shows the Penalty Function trajectory in comparison to the time optimal trajectory when the maximum maneuver time for the Penalty Function trajectory is four percent greater than the computed optimum time.

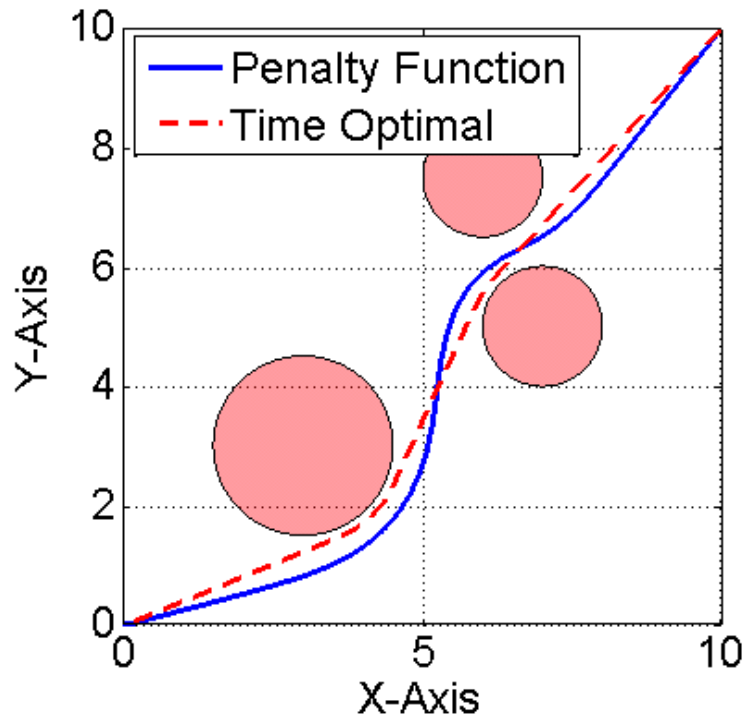


Figure 56 Potential Function Trajectory when Maximum Time is Four Percent Larger than Optimal Time.

Figure 57 depicts a similar scene except that the Penalty Function trajectory's maximum maneuver time is five percent greater than the optimum time. Although both maneuvers are feasible, the four percent longer trajectory resembles the optimal maneuver but the five percent longer trajectory does not. In general, both trajectories provide greater trajectory robustness from the standpoint of avoiding obstacle impact; however, the closed-loop architecture presented below used of the four percent longer trajectory since it was more complex and time optimal.

Before proceeding with the closed-loop results, the maze problem was revisited. As was shown in the Tricycle Section, the standard time-optimal problem formulation could not solve the maze problem without the benefit of a four-point guess to aid with guidance. The purpose of experimentation was the determination of whether or not the Penalty Function approach would change that result. The cost used in this trial

was identical to cost shown in Figure 53. Utilizing a no-guess initial input, the resulting trajectory is shown in Figure 58. The solution was a local extremal trajectory that bisected two walls.

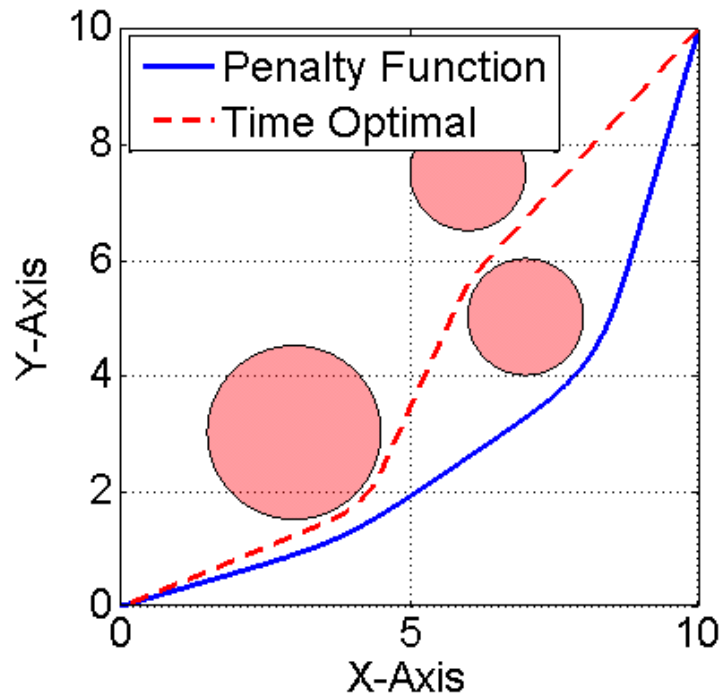


Figure 57 Potential Function Trajectory when Maximum Time is Five Percent Larger than Optimal Time.

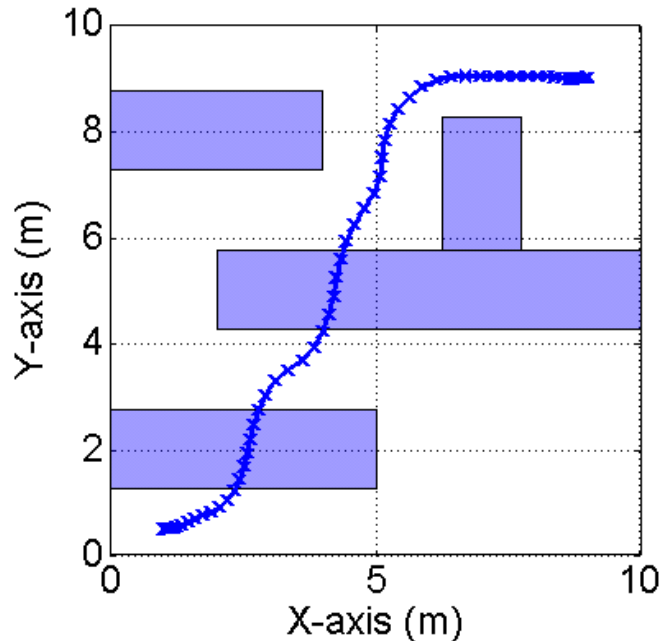


Figure 58 A Maze Optimal Trajectory Generated from No Initial Guess.

While the trajectory was infeasible with respect to real-world application, it is apparent that the planner was attempting to minimize cost. Each wall is intersected nearly perpendicularly as the planner attempted to shorten the length of stay. Figure 59 shows the optimal result following supply of a four-point initial guess. Similarly to the previous maze results, DIDO easily found local extremal trajectories unless a basic concept of the actual solution was provided as the guess. Once again, this issue may only be a reflection of the possession of global knowledge. Real-world local knowledge problems would not initially realize the overall problem complexity, thereby simplifying the solution process.

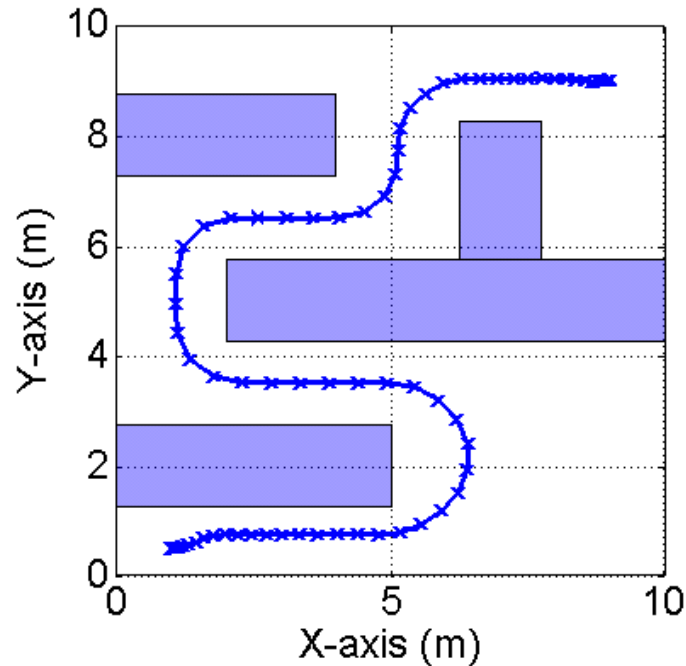


Figure 59 A Maze Optimal Trajectory Generated from a Four Point Initial Guess.

Returning to the closed-loop problem, Figure 60 and Figure 61, the positional and control trajectories respectively, depict the initial attempt to solve the closed-loop optimal control problem. The simulated computation time was one second, 20-node solutions were used, and the interpolation technique was spline interpolation. The closed-loop and open-loop trajectories are presented side-by-side for comparison, and the optimization problem output-nodes are represented as 'x' marks. The nodes are graphed so that the increased resolution of the closed-loop solution is visible. Despite the increased resolution, the closed-loop maneuver does not mimic the open-loop trajectory. There are five reasons for the disparity between the two trajectories.

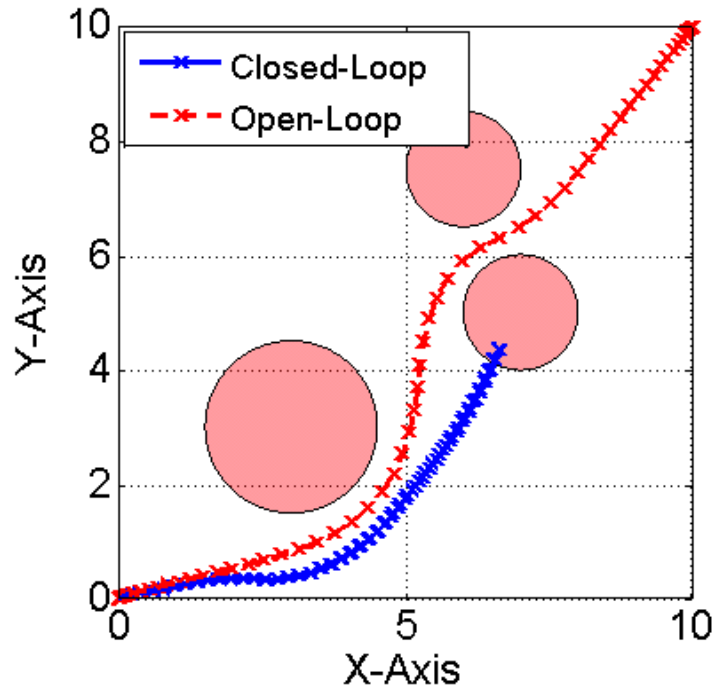


Figure 60 Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Spline Interpolation.

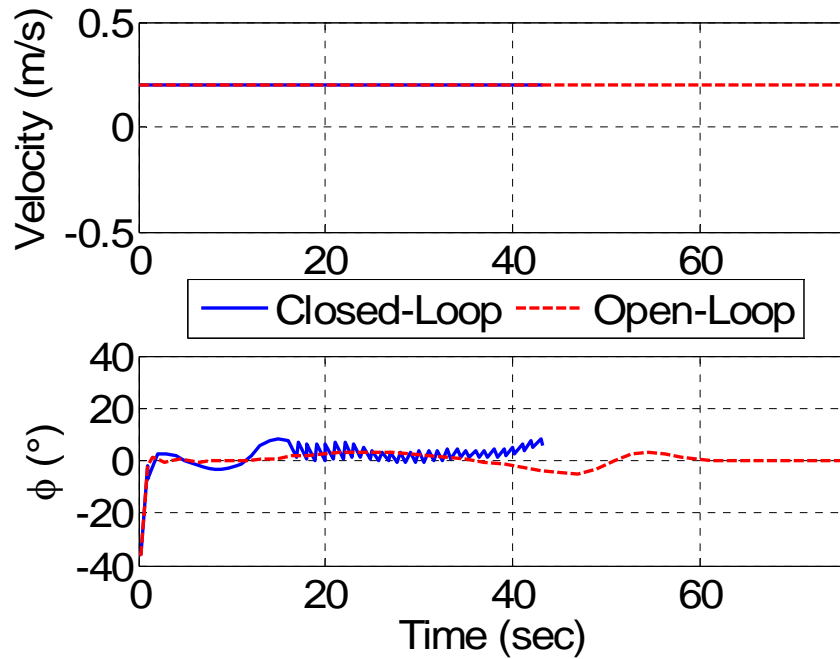


Figure 61 Closed-Loop and Open-Loop Control Trajectory Comparison.

The first reason for the closed-loop failure is the use of spline interpolation on the control trajectory. Spline interpolation was used with great success on open-loop situations, but the closed-loop approach creates high steering rates. This creates a problem because the spline technique permits overshoot. The interpolation problem is resolved by using MATLAB's 'pchip' interpolation, a shape-preserving, piecewise cubic interpolation that permits no overshoot.

The second reason for the closed-loop maneuver failure is the one-second computation time. How this condition affects the controller is most easily illustrated by an example; Figure 62 shows a five-scene instance where computation time causes a control failure. Both the workspace and control trajectories are displayed. The vehicle is traveling along a trajectory when a new obstacle appears, which forces the vehicle to begin computing a new trajectory. In the third scene, the new trajectory is available for use; however, the vehicle has traveled for one second since the computation began. For this reason, the control trajectory must be truncated by one second before it is applied. In the fourth scene, the truncated trajectory is performed, but truncation has caused the new trajectory to be unsatisfactory in avoiding the threat since part of the new control trajectory is lost. The fifth scene provides the actual trajectory taken from beginning to impact, and in the control trajectory a jump in the control input is visible. This jump resembles the many discontinuities or jumps that are shown in Figure 61; this indicates that the controller is attempting to perform a maneuver that is not being captured.

Of course, the simulated scenario uses global information; therefore, no new obstacles can pop up to create the problem just explained. Instead, the third and fourth reasons for the closed-loop failure create a similar situation, despite global information. First, the system model is of a low fidelity, i.e. only a kinematic model is used; therefore, the controls rates are not limited. This permits control jumps from one trajectory solution to the next. While this may create implementation issues, it also creates the condition where an important portion of the control trajectory occurs within the first truncated second. Second, a low node solution is used, i.e. only 20 nodes, and this introduces significant error and disparity into each successive solution.

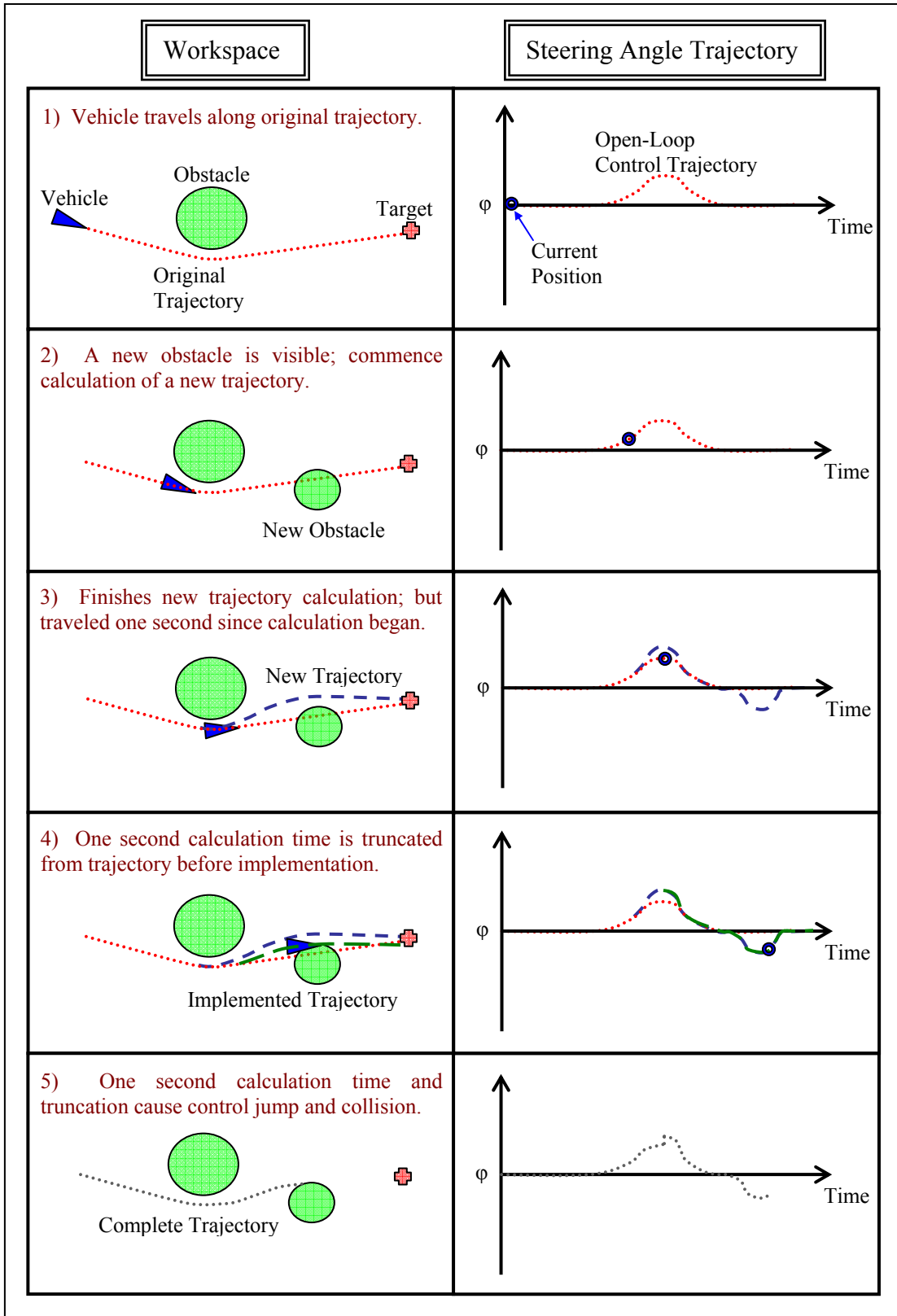


Figure 62 Description of How Calculation Time Affects Trajectory Traveled.

Higher-node solutions are possible but are typically undesirable for the increased computation time. This simulation was run on a Dell Dimension® 4400 computer with a Windows® 2000 Operating System (OS), an Intel Pentium IV 1.60GHz processor, and 500MB of Random Access Memory (RAM). The mean computation time for the 20-node solution was approximately 15 seconds, greater than an order of magnitude more than the simulated clock time. The computation time concerns will be discussed in detail in a later section. In general, the model fidelity and low-node solution problems can be fixed, but they both come at a price of computation time.

The fifth reason for closed-loop failure is the fixed maneuver time. The closed-loop trajectory deviates from the open-loop trajectory from the initial position, and with a fixed maneuver time, there must be a zero sum gain. The vehicle is traveling at maximum speed throughout the trajectory, so if one turn is made wider then the next turn must be made shorter. This condition essentially forces the vehicle to drive over the small obstacle. While the maneuver minimizes cost, it must satisfy all of the endpoint conditions. This problem could be resolved by relaxing the maneuver time.

Each of the five concerns mentioned above will be addressed in the process of resolving the closed-loop method, but solutions will be implemented incrementally so that each improvement can be evaluated individually to witness its effect. The only concern that is not addressed in this subsection is the low model fidelity, which will be discussed in a separate subsection.

Figure 63 and Figure 64 present the closed-loop results when the interpolation technique is pchip. It is visible from both trajectories that less deviation occurs with respect to the open-loop path; however, enough deviation occurs that the fixed maneuver time eventually forces the vehicle to ignore the small obstacle that it hits. From this result, it is apparent that the fixed maneuver time must be addressed. It is important to note that this is a case of a static environment and with a dynamic environment, the maneuver time would need to be variable anyways.

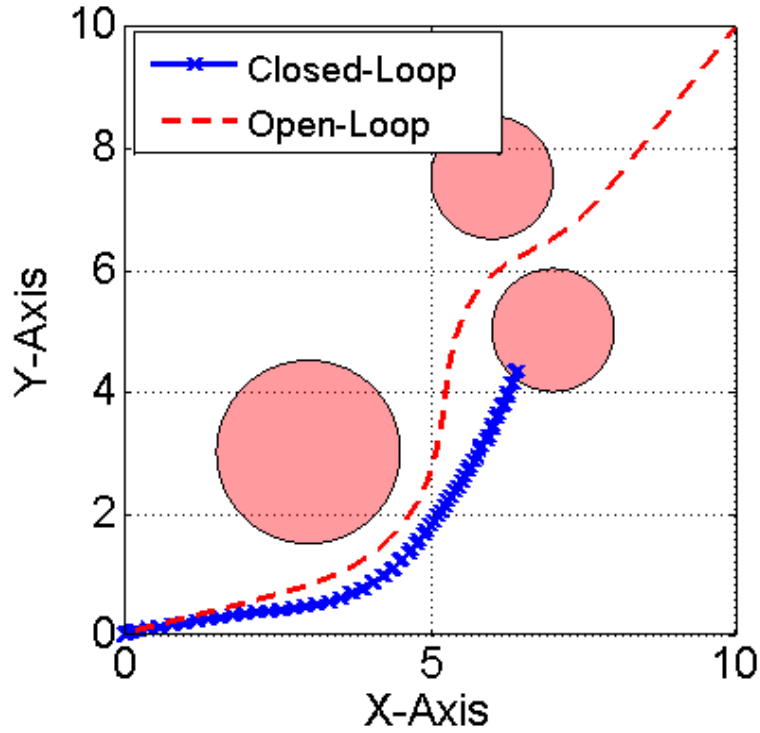


Figure 63 Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Shape-Preserving Piecewise Cubic Interpolation.

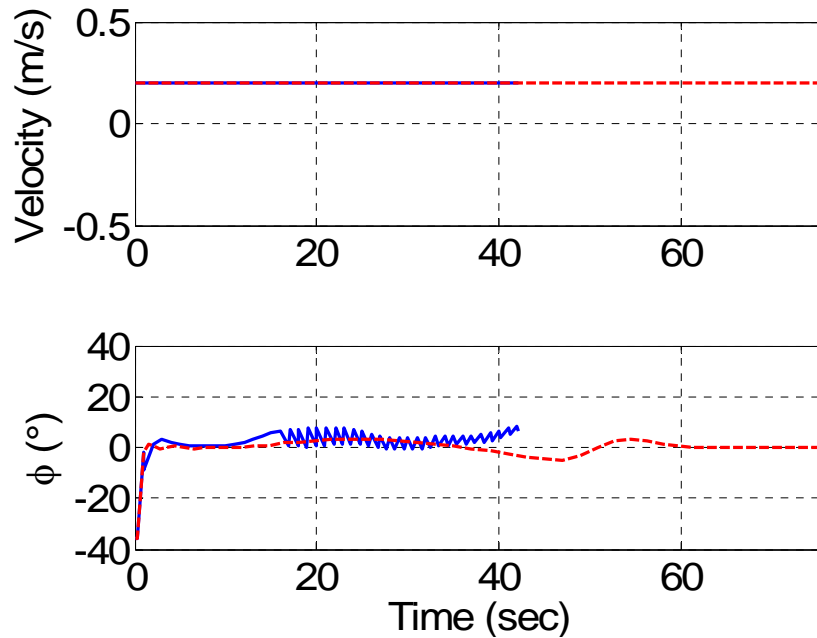


Figure 64 Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Shape-Preserving Piecewise Cubic Interpolation.

Relaxation of the maximum maneuver time must be accomplished judiciously so that optimality and robustness characteristics are balanced. To do this, increases in maneuver time were related to increases in maneuver cost. As it was seen from the previous examples, a limited maneuver time causes the vehicle to accept high obstacle costs in an attempt to satisfy the endpoint condition; therefore, a 25% increase in maneuver cost created a two percent increase in maximum maneuver time. Figure 65 and Figure 66 show the closed-loop results when the maximum maneuver time is relaxed. From the positional trajectory, it is visible that the vehicle no longer attempts to drive directly through the small obstacle in an attempt to reach the finish. The resulting control trajectory more closely follows the open-loop control trajectory, but the non-smoothness of the response is indicative of the fact that either the update frequency must be increased, the model fidelity must increase, or higher-node solutions must be used.

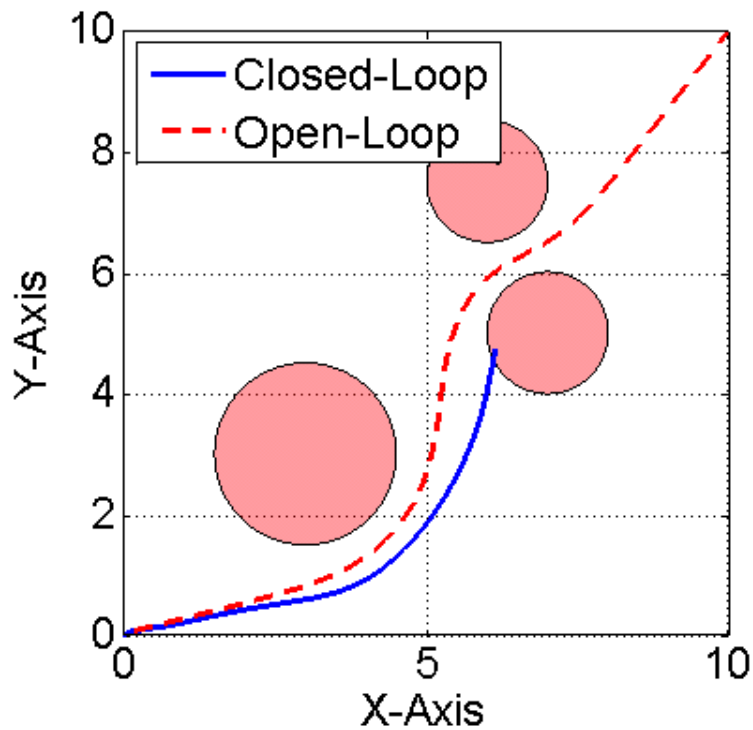


Figure 65 Closed-Loop and Open-Loop Trajectory Comparison;
Utilizing Pchip Interpolation and a Relaxed Maneuver Time.

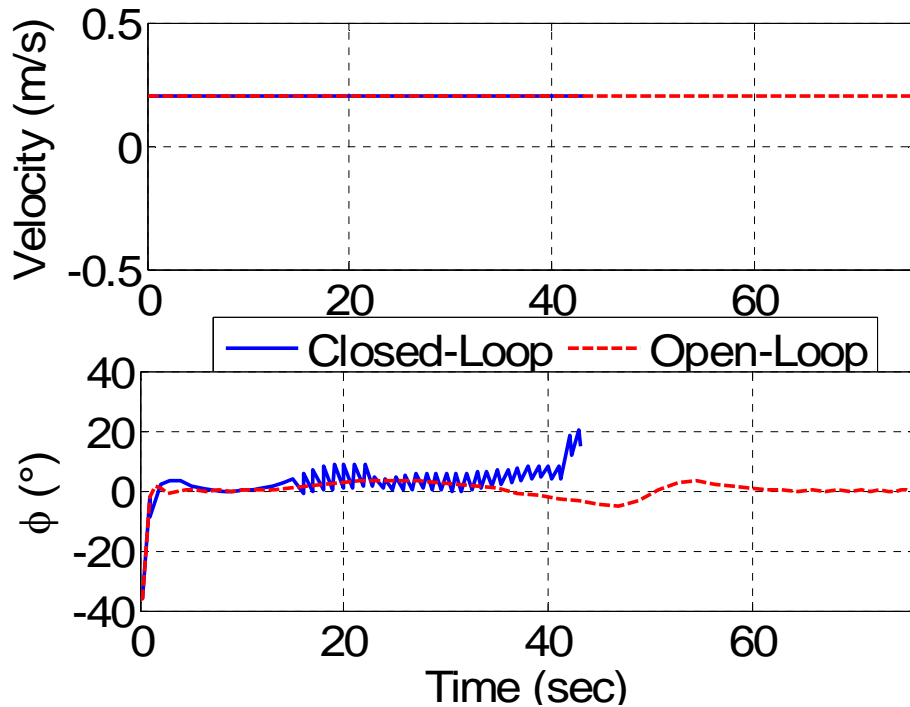


Figure 66 Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Pchip Interpolation and a Relaxed Maneuver Time.

In an attempt to observe the affect of using a high-node solution during closed-loop implementation, the discretization was changed from 20 nodes to 30 nodes. Figure 67 and Figure 68 show the results from the 30-node solution implementations; the average computation time increased to over 25 seconds – an increase of 66%. The increased accuracy of the high-node trajectories improves the ability of the closed-loop maneuver to parallel the open-loop trajectory, but at the narrow passage between two obstacles, the technique fails. The reason for failure is clearly the update frequency and model fidelity; the control trajectory displays significant non-smoothness as the controller attempts to provide erratic evasive maneuvers that are repeatedly truncated. Logically the only two ideas that can resolve the failure of the closed-loop Penalty Function approach would be a higher update frequency or greater model fidelity. While the attempt at using higher-node solutions created limited improvement at an increase in computational complexity, the work here is not all encompassing. Future detailed analysis could find improved performance at higher node-levels; however, that analysis is not endeavored in this research.

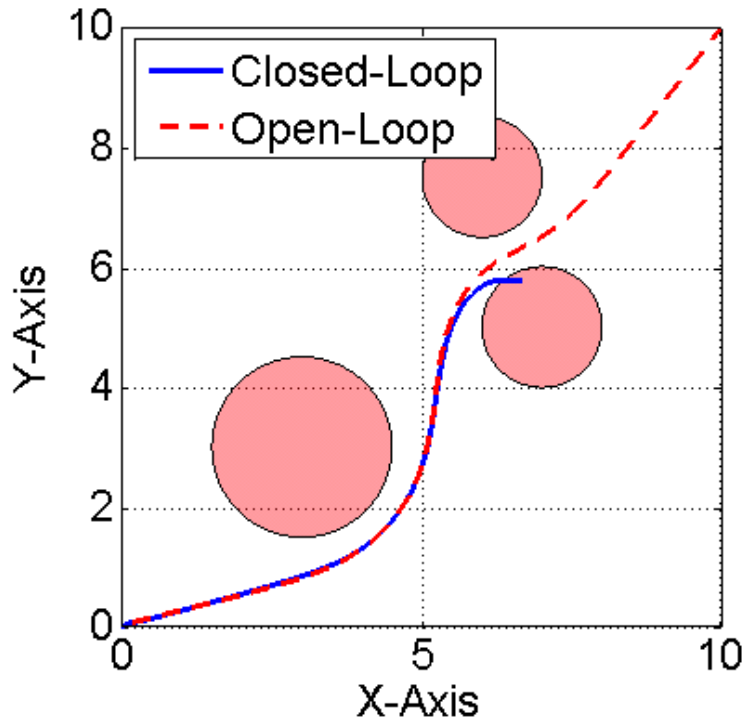


Figure 67 Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and 30-Node Solutions.

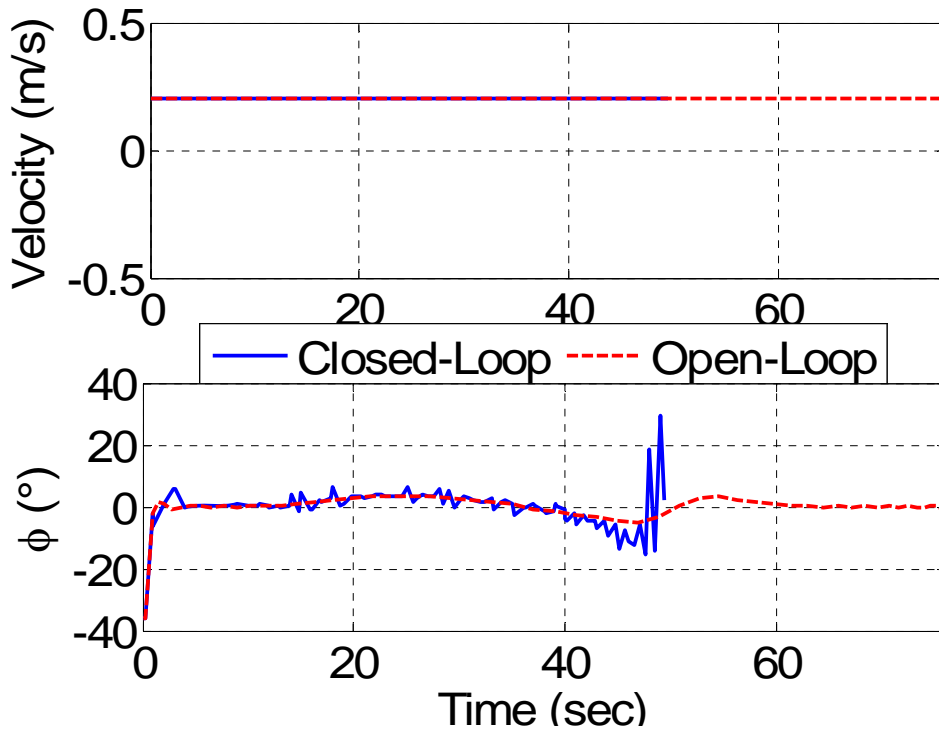


Figure 68 Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and 30-Node Solutions.

The next approach to closed-loop resolution was an increase the update frequency; the number of nodes used in each closed-loop solution was reduced, again, to 20. Incorporating a control frequency of one update every 0.5 seconds resulted in maneuvering success around the obstacles, but the optimal control computations slowed down near the end of the trajectory, i.e. within 10 percent of the end point, the calculation times ballooned to several minutes. The cause of the computational hardship was the fixed maneuver time. Two simple solutions could resolve this issue. As the endpoint condition is approached, the maximum maneuver time can be increased or the control architecture can switch to a time-optimal method. Both approaches allowed the maneuver to be successful. Figure 69 and Figure 70 show the closed-loop maneuver when the control architecture is changed to a time optimal technique near the endpoint condition. In general, this closed-loop approach followed the open-loop maneuver with much greater precision and the average computation time was approximately six seconds. This should be surprising since the previous run involving 20-node solutions averaged more than 15 seconds per solution. The difference is that the end of the trajectory is not complex; therefore, computation times decrease to a value between one and two seconds near the endpoint manifold. This condition impacts the average computation time in an illusive way.

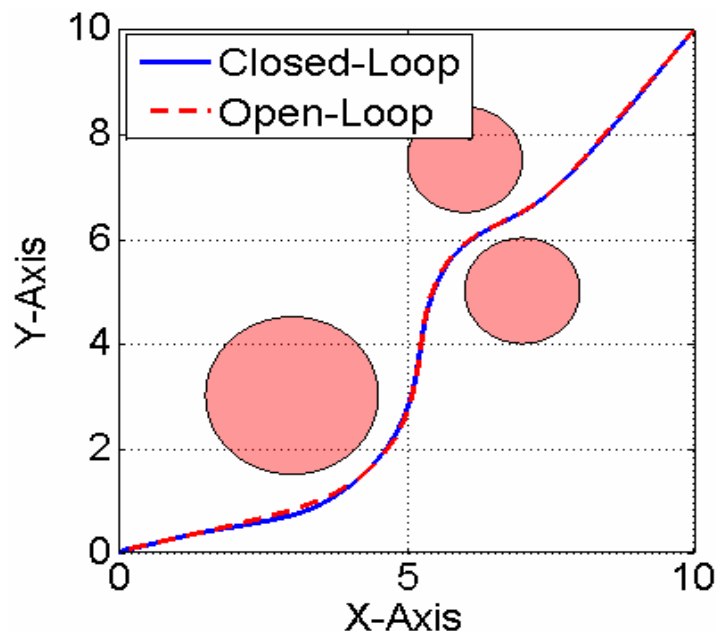


Figure 69 Closed-Loop and Open-Loop Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and Increased Update Frequency.

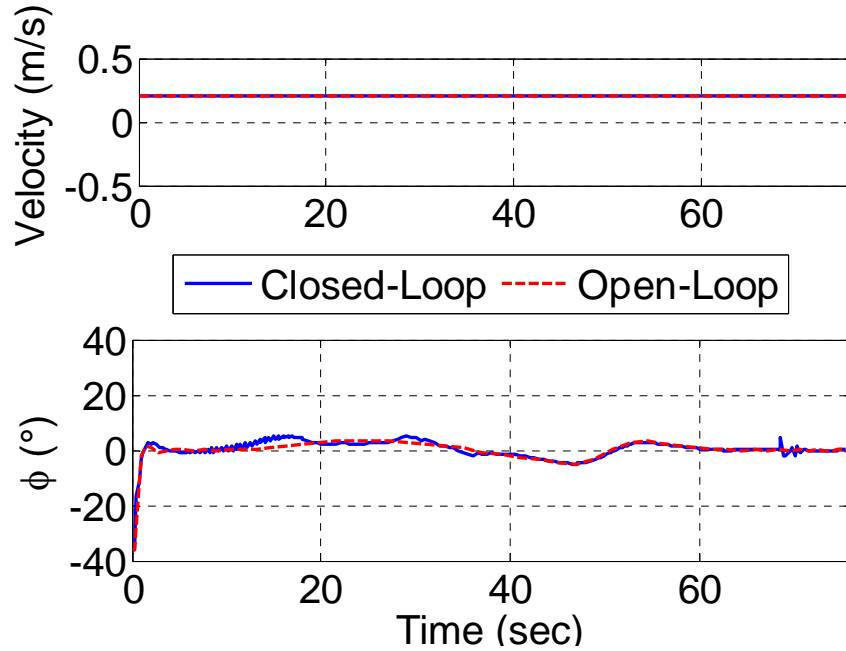


Figure 70 Closed-Loop and Open-Loop Control Trajectory Comparison; Utilizing Pchip Interpolation, Relaxed Maneuver Time, and Increased Update Frequency.

b. Resolving Uncertainty Approach

Implementation of the Resolving Uncertainty Approach was less challenging. The first attempt was successful and the architecture parameters included one second simulated computation times, 20 node solutions, and pchip interpolation for the control trajectory. Figure 71 and Figure 72 display the results from the Resolving Uncertainty Approach of closed-loop control; the results are compared to the initial optimal trajectory computed from the global information of the obstacles before vehicle motion. When at the starting position, the obstacle buffers are at their largest size and this causes the optimal trajectory to give a significant distance of safety from the actual obstacle. Each obstacle in Figure 71 is represented by two circles. The inner circle is blue and it represents the actual obstacle. The outer obstacle is magenta and it represents the simulated obstacle. In most cases, the vehicle traveled close enough to the obstacles to make the circles appear as one; the exception is the highest obstacle on the map, which maintained the two-circle distinction.

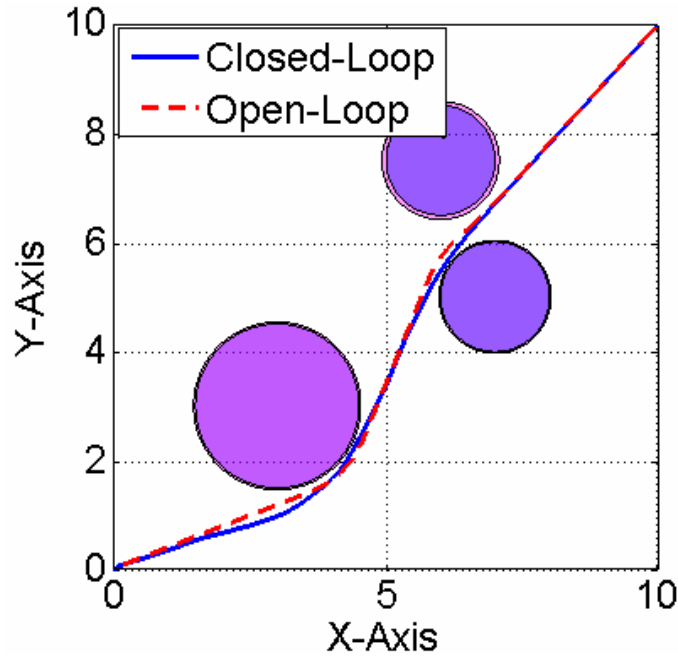


Figure 71 Resolving Uncertainty Closed-Loop Results, Compared to Initial Optimal Trajectory.

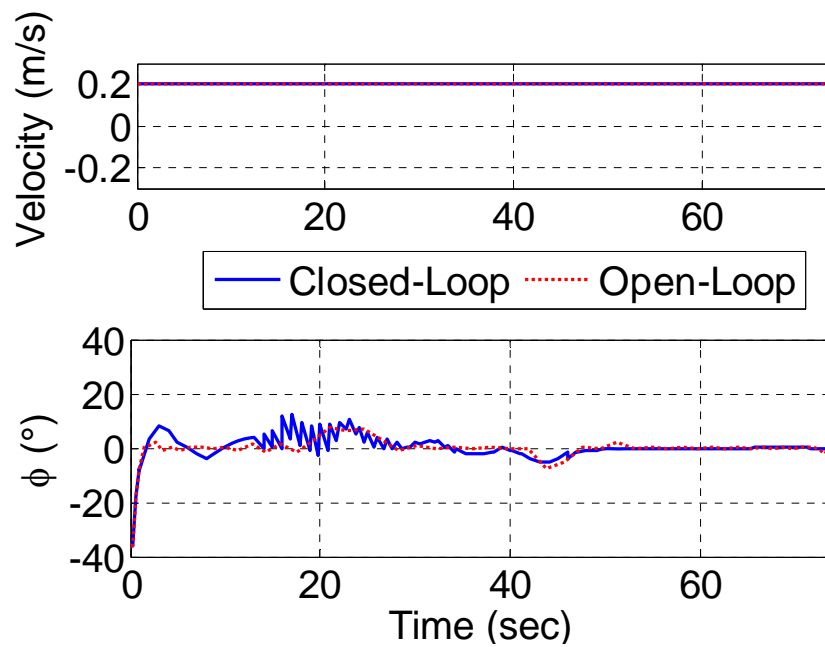


Figure 72 Resolving Uncertainty Closed-Loop Control Trajectory, Compared to Initial Optimal Trajectory.

Despite the inability of the closed-loop response to mimic the initial optimal trajectory, success was more easily achieved with this technique than the potential field approach. The one-second update frequency, insufficient for the Penalty Function approach, created no problems in this simulation, and the average computation time was only 3.7 seconds nearly half of the previous. Increased model fidelity, increased update frequency, and increased solution discretization could all improve solution optimality.

6. Improving Model Fidelity

The previous closed-loop results displayed significant discontinuities over the control trajectory; these discontinuities helped to create path planning difficulties in the face of uncertainty and infrequent control updates. The simplified, kinematic, car model was thought to be one of the reasons for the non-smooth results. Improving model fidelity to include limitations on the rate of control change is one possible solution for removing “chatter” in the controls. In changing the model, system response speed and overall trajectory optimality are sacrificed for the sake of greater control realism and insensitivity to uncertainty.

Placing limits on the rate of control variable change in an optimal control problem requires manipulation of the problem formulation. The actual system controls, velocity and steering angle, become state variables and the rate of change of velocity and steering angle become the controls, Equation (35).

$$\underline{x} = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \phi \end{bmatrix} \quad \underline{u} = \begin{bmatrix} \dot{v} \\ \dot{\phi} \end{bmatrix} \quad (35)$$

Essentially, the original control variables are now required to be smooth while the new control variables, i.e. the time derivative of old control variables, are allowed to be non-smooth. Propagation of the vehicle dynamics remains unchanged despite the change in problem formulation. While \dot{v} and $\dot{\phi}$ can solve the chatter problem, they provide unnecessary information with respect to the control of the robot; therefore, propagation of vehicle position and orientation is still accomplished with the velocity and steering angle.

In general, the increase in dimension of the state vector increases the complexity of the optimal control problem, and the computation time was expected to worsen.

Initial tests of the new problem formulation occurred on the three-obstacle problem exhibited above for the closed-loop architecture demonstration. The control method utilized for the test was the resolving uncertainty approach. Figure 73 displays the control trajectory solution for the three-obstacle problem using the new model; this can be compared directly to Figure 72, the old control trajectory for the same problem.

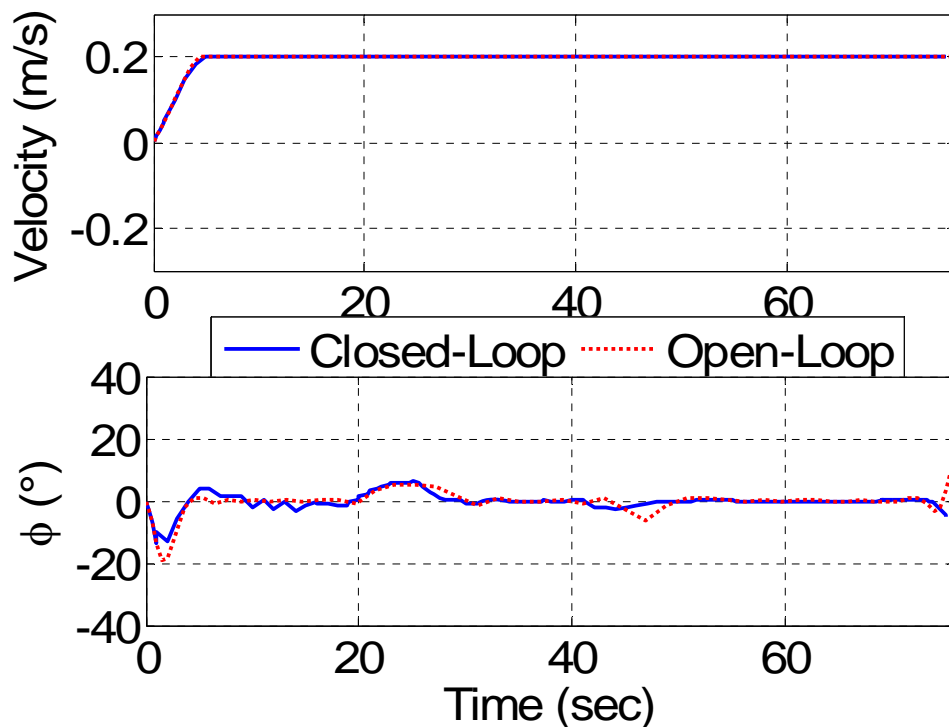


Figure 73 Closed-Loop Positional Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.0 Second Update Rate.

In both results, pchip interpolation, 20-node solutions, and one-second update times were used. The new model produced a control trajectory that was more smooth and displayed greater resemblance to the open-loop trajectory than the old model. The drawback was a loss of computational speed; the average computation time was 5.5 seconds.

The results shown above demonstrate the capability of the new model to solve the problem that was solved without the increase in model complexity; therefore, the

important assessment tests the new model under a lower update frequency. Figure 74 through Figure 76 show the closed-loop results of the improved car model when the control updates every 1.5 seconds, an increase of 50%. The controller was successful under this model; an achievement not possible before. In fact, the closed-loop trajectories display good resemblance to the open-loop trajectories despite the drop in update frequency. These results display the possible advantages of increasing model fidelity with the resolving uncertainty method, and it was important to test whether the same was true of the Penalty Function method.

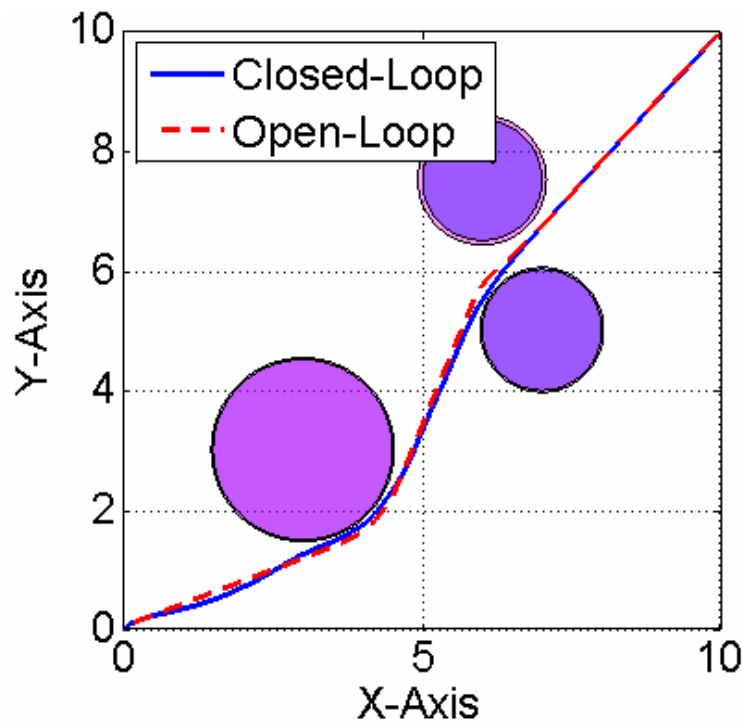


Figure 74 Closed-Loop Positional Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.5 Second Update Rate.

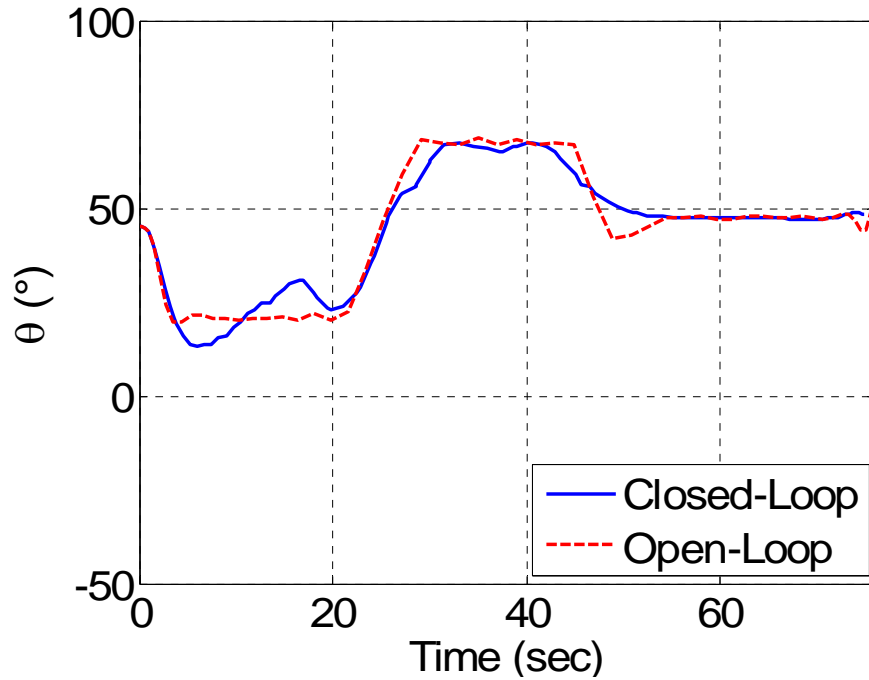


Figure 75 Closed-Loop Angular Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.5 Second Update Rate.

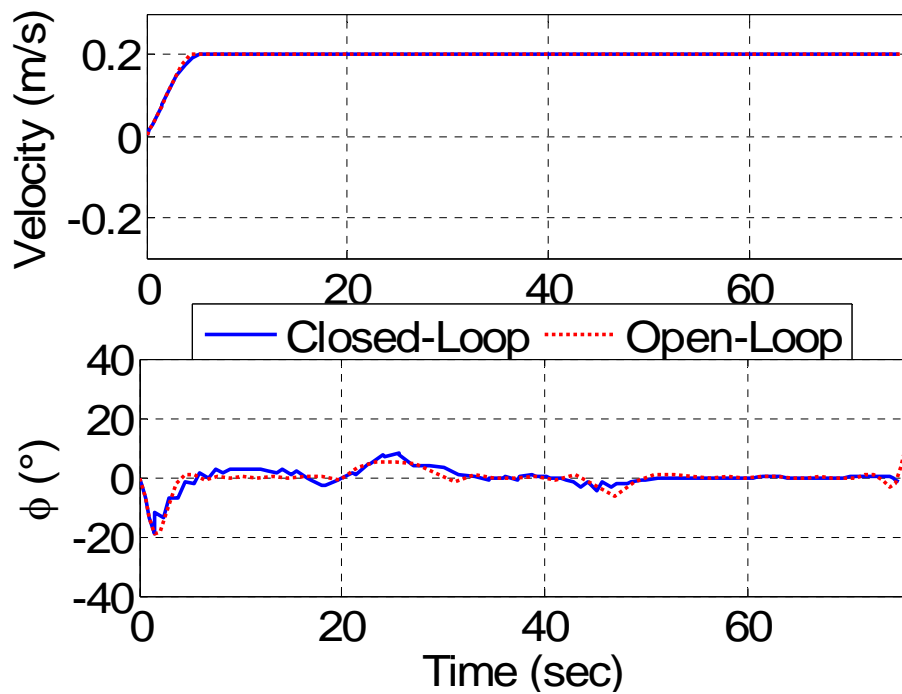


Figure 76 Closed-Loop Control Trajectory Using Resolving Uncertainty Approach, Improved Model, and 1.5 Second Update Rate.

Figure 77 through Figure 79 display the closed-loop results of the Penalty Function approach when incorporating a one-second update time and the new car model. The successful results here can be compared directly to the failure presented in Figure 65; the only different between the two problems is the model fidelity. While the average computation time increased to 13.5 seconds, the closed-loop response transformed from complete failure to near equality to with the open-loop response. From the results presented here, it is apparent that limiting the rate of change of the controls can significantly benefit the closed-loop response, so long as adequate computational time is available to accommodate the increased complexity.

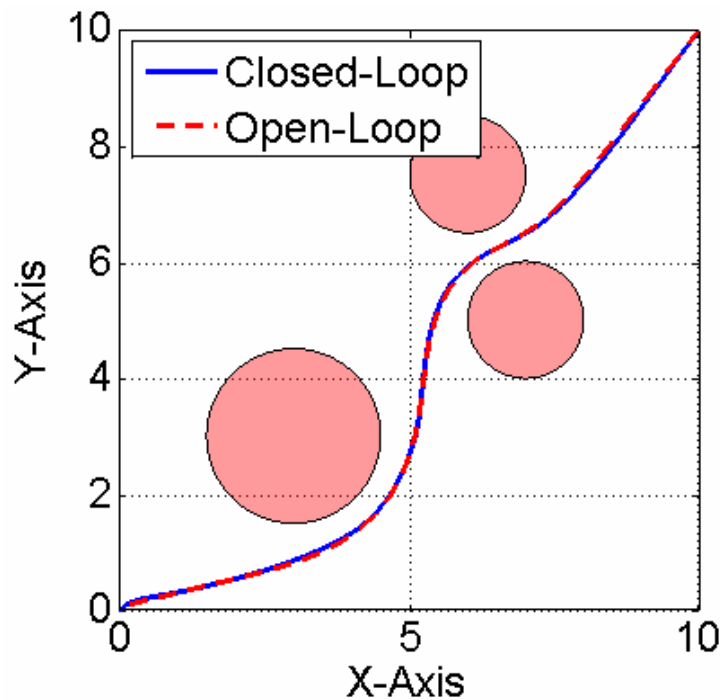


Figure 77 Closed-Loop Positional Trajectory Using Penalty Function Approach, Improved Model, and 1.0 Second Update Rate.

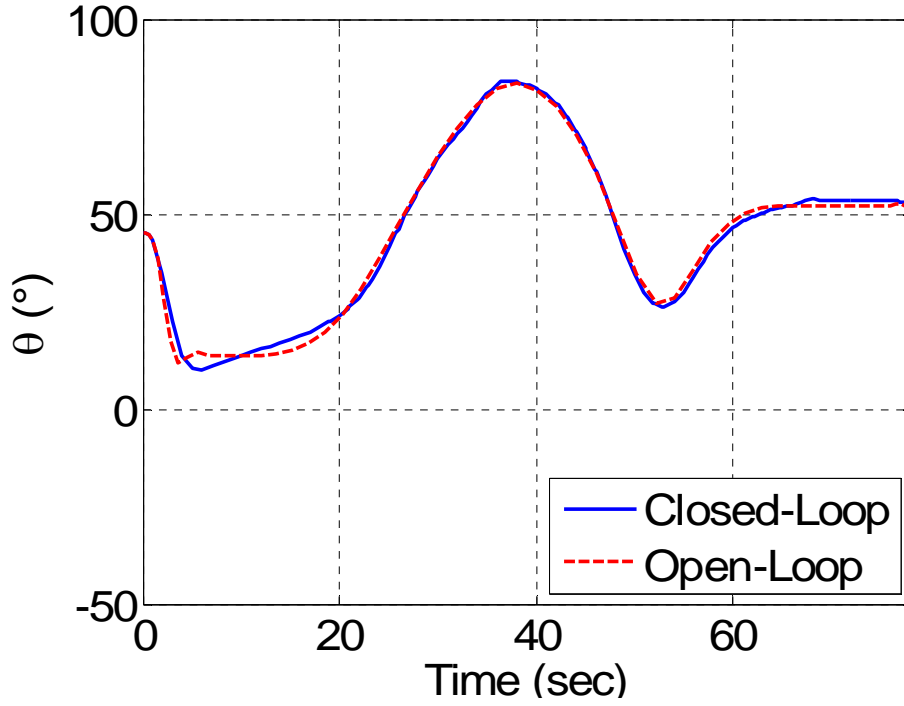


Figure 78 Closed-Loop Angular Trajectory Using Penalty Function Approach, Improved Model, and 1.0 Second Update Rate.

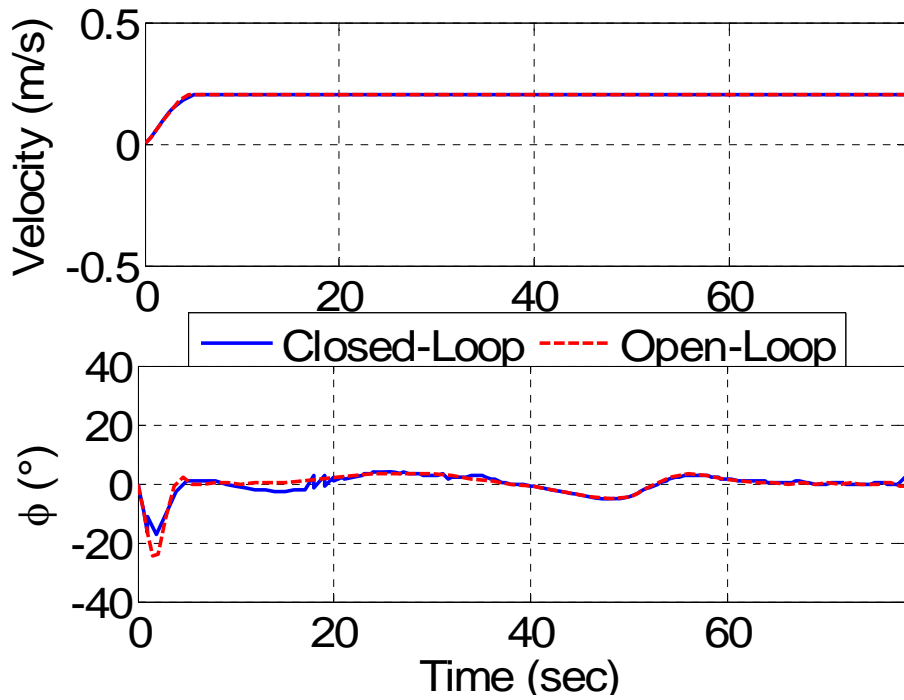


Figure 79 Closed-Loop Control Trajectory Using Penalty Function Approach, Improved Model, and 1.0 Second Update Rate.

7. Robustness through Local Knowledge

Up until this point, global knowledge was assumed available and the closed-loop architecture was tested against a static environment. Numerical error in interpolation and propagation created the uncertainty in the close-loop system. Real systems have limits to the range of sensor information; therefore, a local knowledge situation is more likely in real applications. Simulating a local knowledge condition benefits closed-loop analysis in two ways. First, it incorporates a greater sense of reality, and second, it offers a test of robustness. Not knowing the location of all obstacles a priori, will require the system to make more drastic correction maneuvers during operation and the success or failure of such maneuvers provides an indication of the system's capability.

A simple local knowledge problem was devised using the same three circular obstacle problem from before. The car begins the trajectory at the point (0,0); therefore, the larger obstacle's position is known at the commencement of the maneuver. The smaller obstacles' positions are known until after the vehicle moves from behind the first. Some realism is lost on the fact that entire obstacle position and dimension are known instantaneously; however, localization and mapping problems were not within the scope of this work.

Figure 80 shows the closed-loop results of the Resolving Uncertainty method under the condition of local knowledge. The characteristics of this simulation, including the number of nodes used, the update frequency of 1.0 second, and the three-variable state vector, are equivalent to the results shown in Figure 71; the key difference being that the local knowledge case was unsuccessful. The reason for failure is the lack of ability to limit the rate of control change and the slow update rate, both of which have been identified previously.

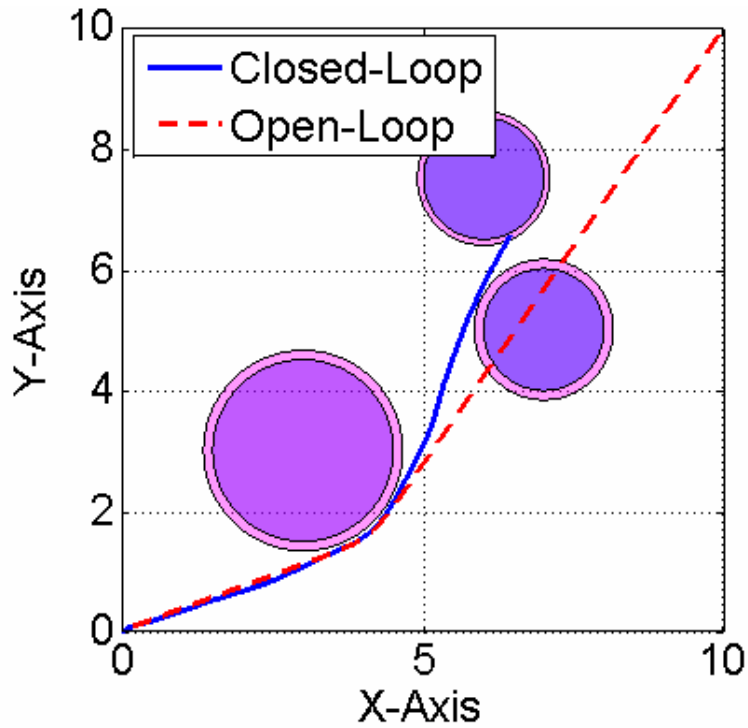


Figure 80 Resolving Uncertainty Method Results for the Local Knowledge Problem.

Figure 81 through Figure 83 display closed-loop results for a similar problem formulation involving the Resolving Uncertainty method; however, the update frequency is increased such that an update is received every 0.5 seconds. The result was a successful trajectory. The availability of local information is visible from the open-loop trajectory that was calculated prior to the motion of the vehicle; that trajectory travels through a smaller obstacle that becomes visible after the maneuver begins.

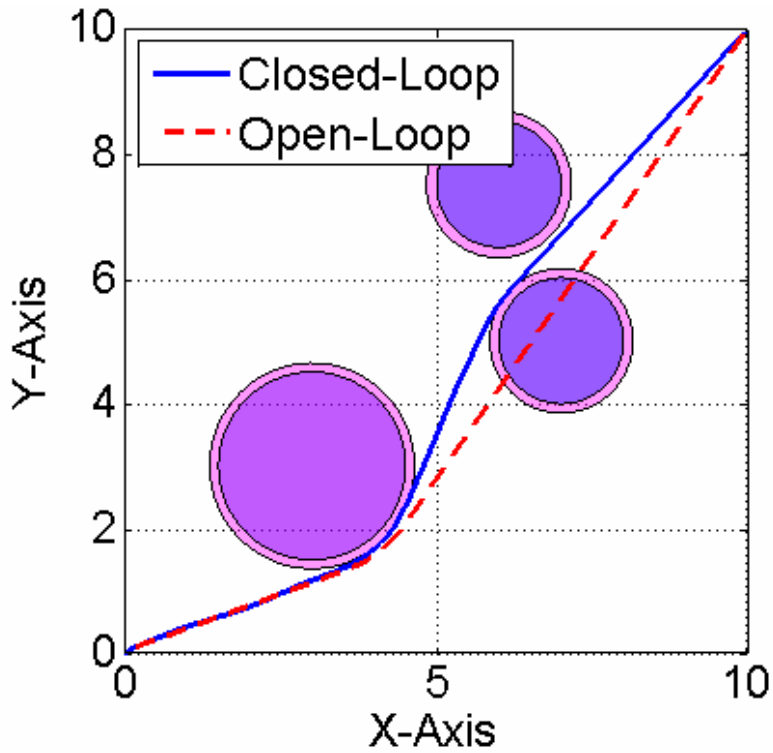


Figure 81 Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a 0.5 Second Update Rate.

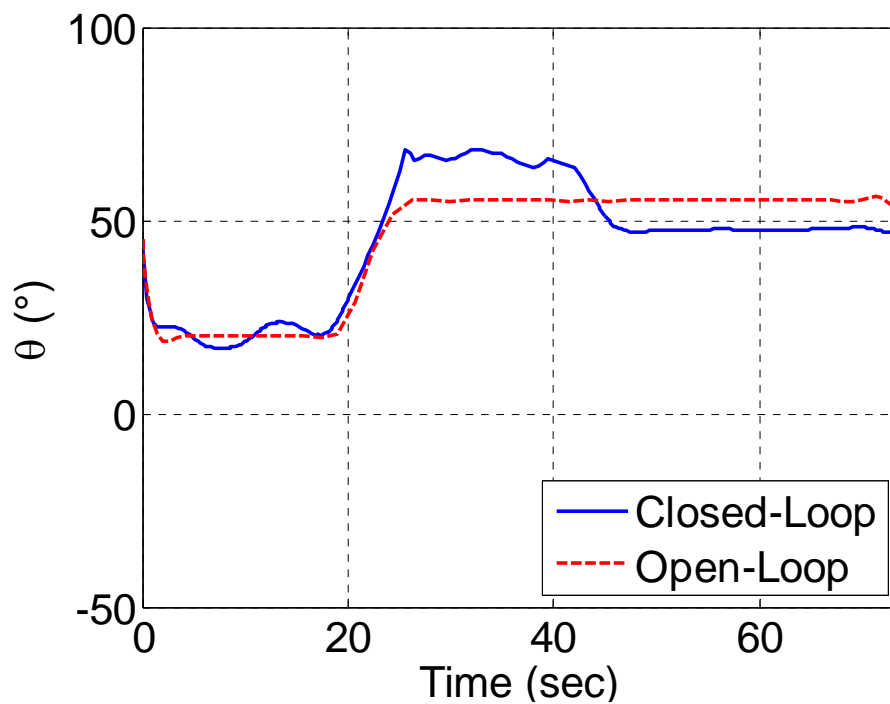


Figure 82 Orientation Trajectory for a Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a 0.5 Second Update Rate.

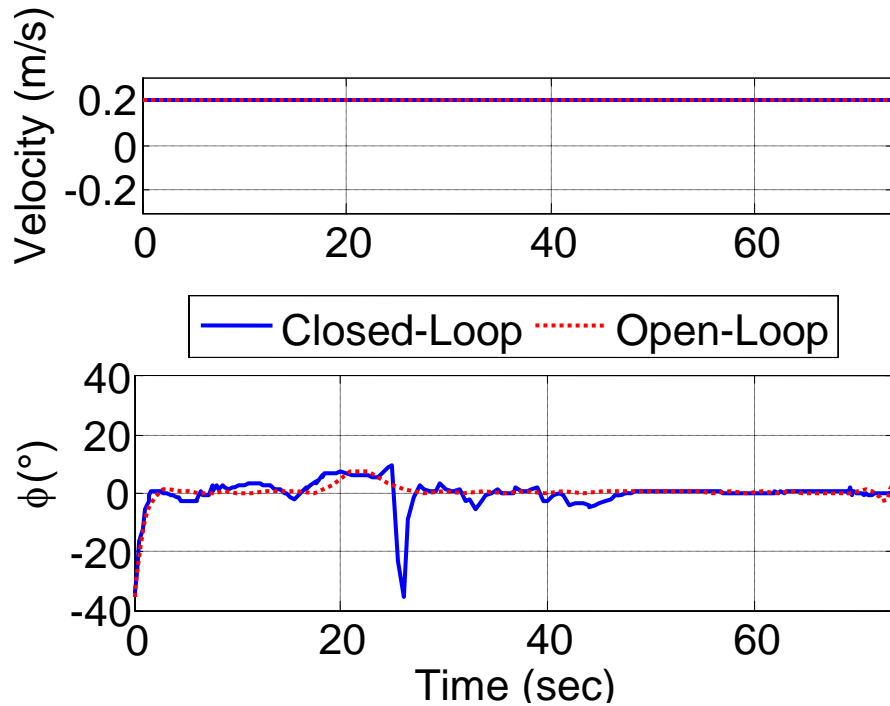


Figure 83 Control Trajectory for a Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a 0.5 Second Update Rate.

A similar test was run using the Penalty Function method. Figure 84 through Figure 86 show the results of the Penalty Function application to the local knowledge problem. The application parameters used here were equivalent to the results shown in Figure 69; this includes a 0.5-second update rate, 20-node solutions, and a three-variable state vector. When the previously unseen obstacle became visible, the total maneuver cost increased because the trajectory traversed an obstacle. The increase in maneuver cost permitted a relaxation of maximum maneuver time, as discussed previously, and this allowed the controller to plan a trajectory away from the new obstacles. The trajectory generated here is less time optimal than the Resolving Uncertainty trajectory; however, the Penalty Function trajectory enjoys greater robustness, i.e. the trajectory is less susceptible to failure due to uncertainty. The average computation time experienced in this simulation, less than four seconds, was improved over the same problem with global information; the two second average improvement can be attributed to the fact that the new trajectory requires less complicated maneuvers.

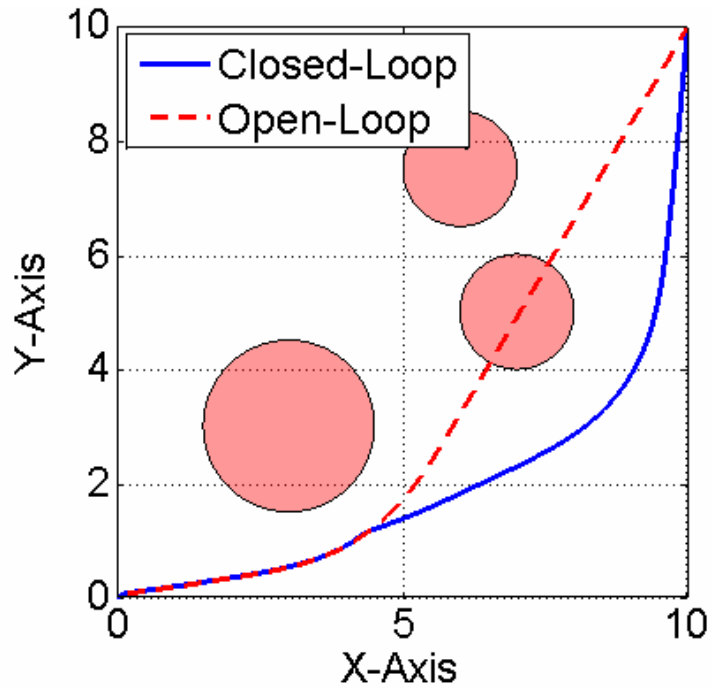


Figure 84 Local Knowledge Problem Solved by the Penalty Function Method.

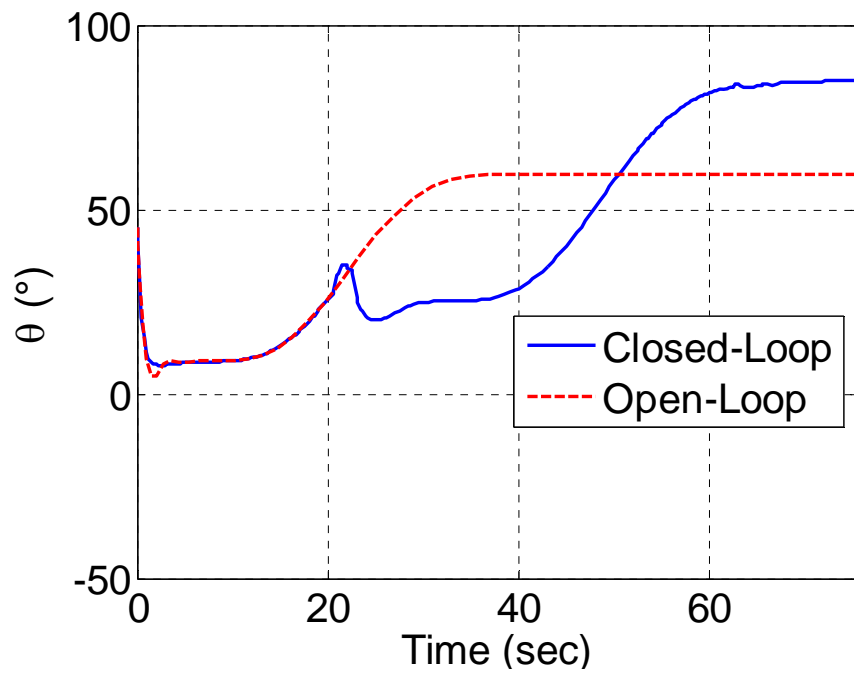


Figure 85 Orientation Trajectory for a Local Knowledge Problem Solved by the Penalty Function Method.

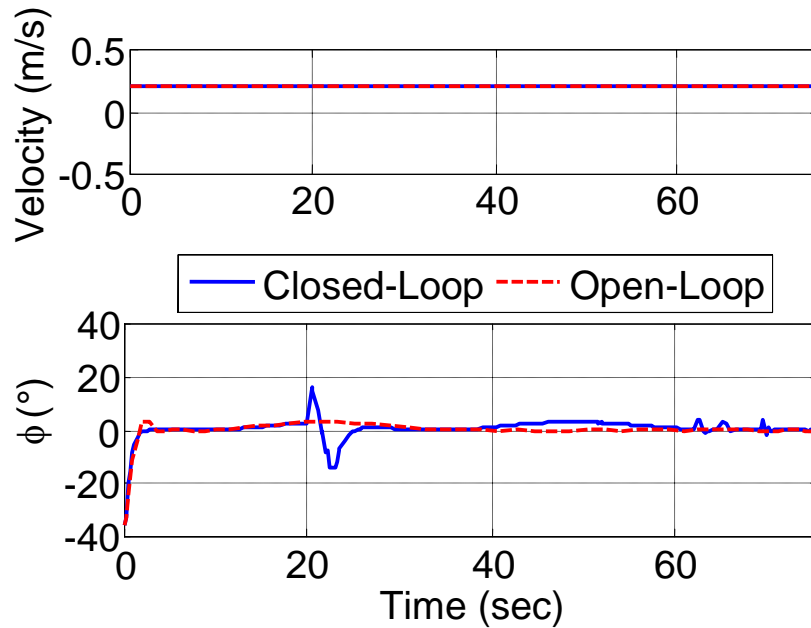


Figure 86 Control Trajectory for a Local Knowledge Problem Solved by the Penalty Function Method.

The next simulations evaluated the effect of incorporating the five-DOF state vector for limiting the rate of control variable change. The method evaluated was the Resolving Uncertainty technique and the results are presented in Figure 87 through Figure 89. The maneuver was successful, but most importantly the use of the five-DOF state vector permitted a decrease in update frequency to one update every 1.5 seconds. This is a significant improvement to the required 0.5-second update rate of the three-DOF local knowledge problem. In addition, the average calculation time decreased from 5.5 seconds for the global knowledge case to 5.1 seconds. The improvement in computation time reflects the fact that maneuver complexity decreases. In the case of local knowledge, the vehicle only has one obstacle to avoid as the maneuver commences, and the other two obstacles only appear after the first obstacle is nearly passed.

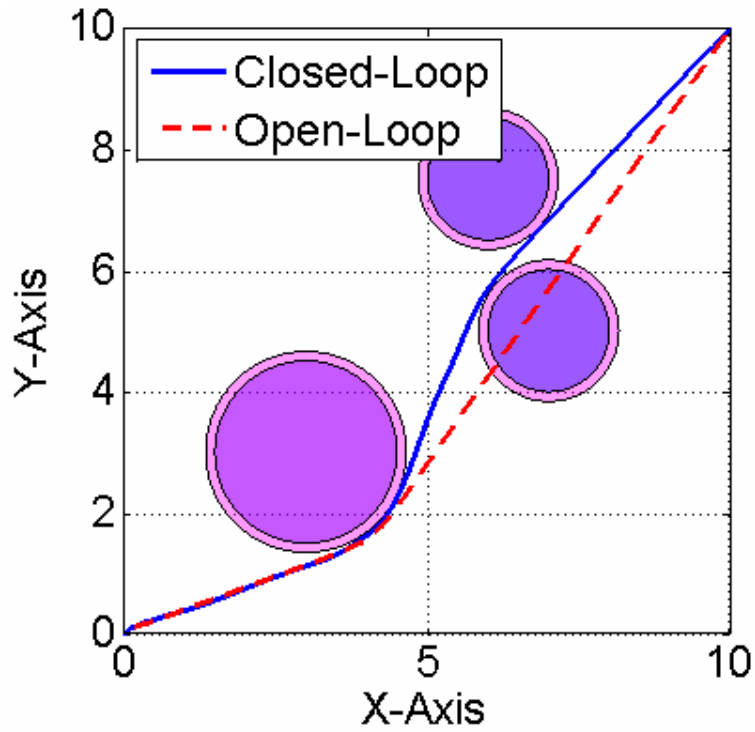


Figure 87 Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a Five-DOF State Vector.

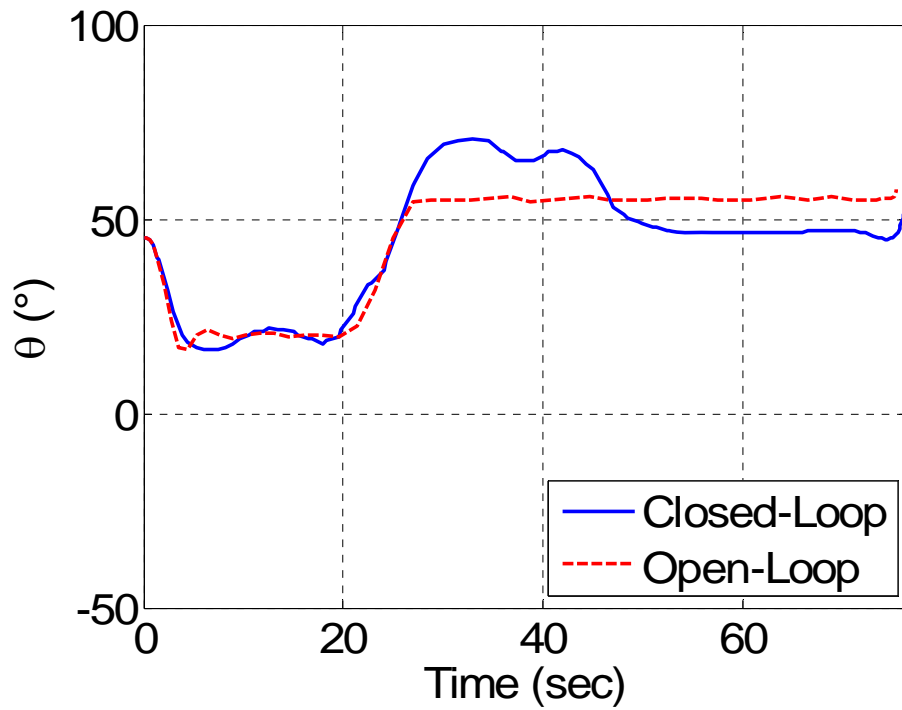


Figure 88 Orientation Trajectory of a Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a Five-DOF State Vector.

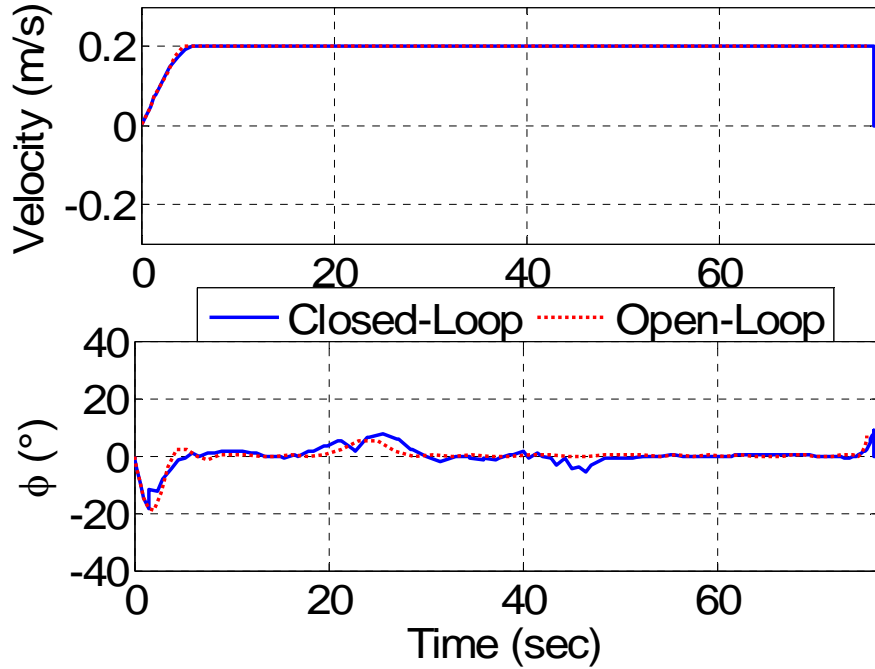


Figure 89 Control Trajectory of Local Knowledge Problem Solved by the Resolving Uncertainty Method and Using a Five-DOF State Vector.

Figure 90 through Figure 92 show the local knowledge problem solved with the Penalty Function method and the five-DOF state vector. The update frequency can be decreased in this simulation, but the result provides little meaning since the maneuver requires no complex movements near the obstacles. The average computation time during this simulation was 3.7 seconds, a ten-second reduction from the global knowledge case. The cause of this significant improvement is the same as the previous cases; however, the original computation time was particularly high, 13.5 seconds. It can be concluded that the local knowledge problem actually simplifies the optimal control calculation; therefore, computation times can actually improve under the more realistic simulations. Greater analysis is necessary to evaluate planning behavior under other obstacle configurations and situations.

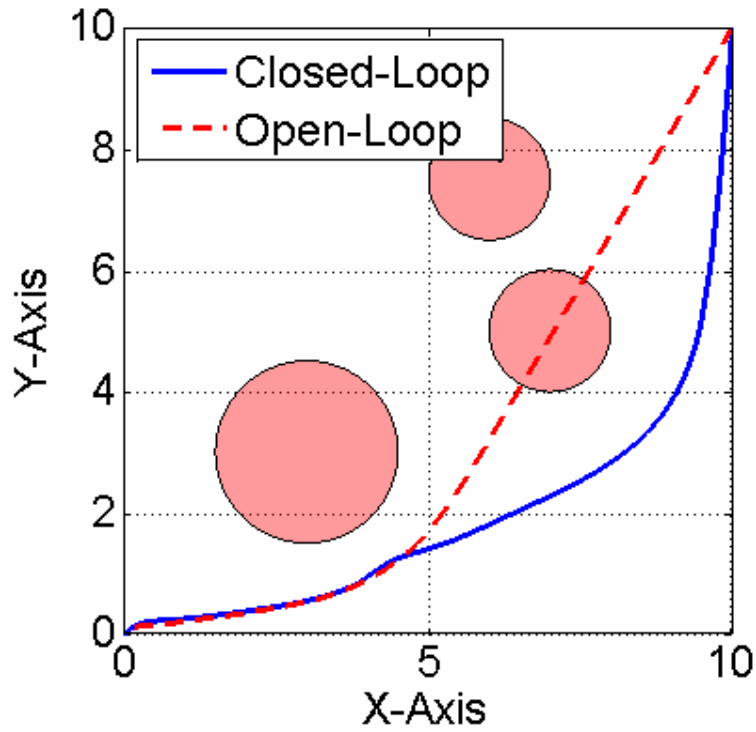


Figure 90 Local Knowledge Problem Solved by the Penalty Function Method and Using a Five-DOF State Vector.

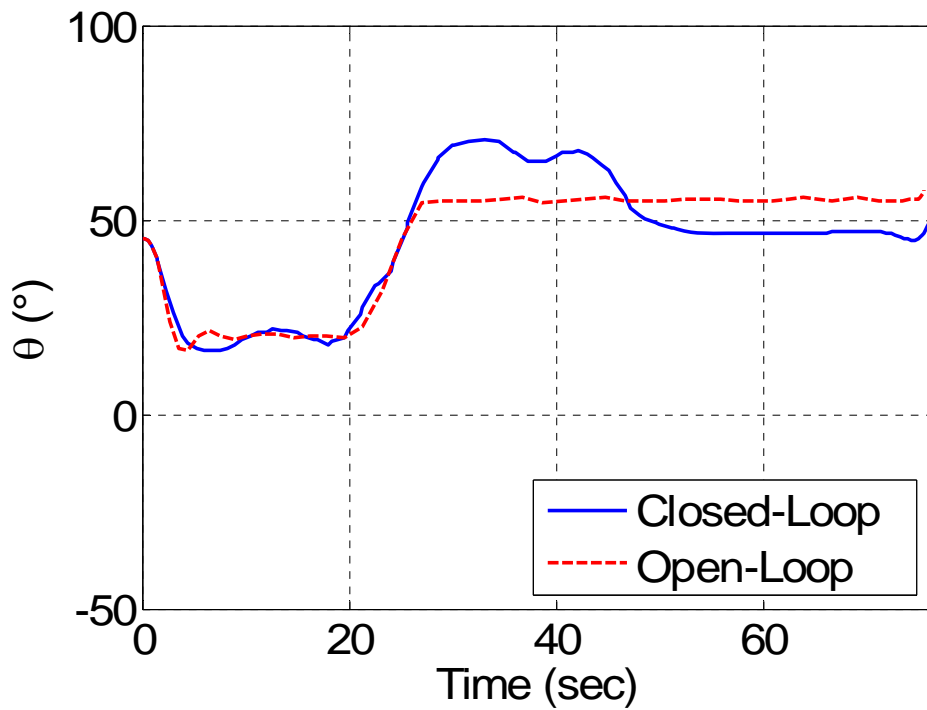


Figure 91 Orientation Trajectory of a Local Knowledge Problem Solved by the Penalty Function Method and Using a Five-DOF State Vector.

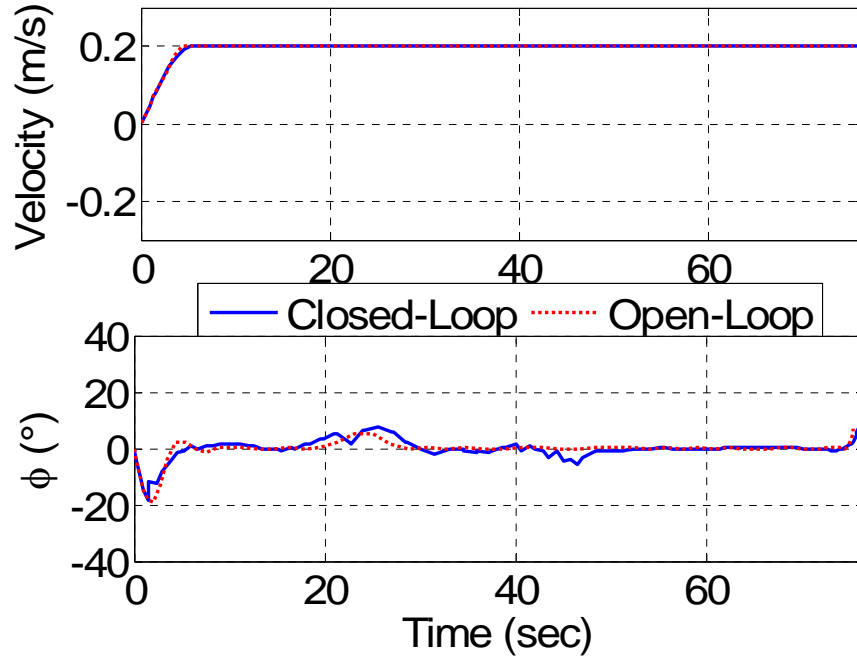


Figure 92 Control Trajectory of a Local Knowledge Problem Solved by the Penalty Function Method and Using a Five-DOF State Vector.

8. Computation Time

Throughout the closed-loop research, the control computation times were documented, but they had no effect on the operation of the path planner. In a real-world application, the computation time would be vital to successfully controlling the vehicle; a fact demonstrated several times during this research by use of an inadequate update frequency. The reason for not including the computation times was twofold. First, this work intended to illustrate only the applicability of these techniques; successful implementation was a future goal. Second, several techniques are currently available to speed the process of optimal control computation, and the necessary time was not available to incorporate all of these improvements.

A networked, Dell Dimension® 4400 computer with a Windows® 2000 OS, an Intel Pentium IV 1.60GHz processor, and 500MB of RAM was used during this research. Table 4 displays mean calculation times for simulations involving global information. Generally, the Penalty Function approach requires more time than Resolving Uncertainty; the same is true of the five-DOF state vector and higher-node solutions.

Table 4 Computation Times (All Simulations Use Global Information)

Update Rate (sec)	Problem	Closed-Loop Technique	State Vector DOF	Number of Nodes	Mean Calc. Time (sec)
1.0	Three-Circular Obstacles	Resolving Uncertainty	3	20	3.7
0.5	Three-Circular Obstacles	Penalty Function	3	20	6.2
1.0	Three-Circular Obstacles	Resolving Uncertainty	3	40	6.2
0.5	Three-Circular Obstacles	Penalty Function	3	40	5.6
1.0	Three-Circular Obstacles	Resolving Uncertainty	3	60	8.5
0.5	Three-Circular Obstacles	Penalty Function	3	60	10.2
1.0	Three-Circular Obstacles	Resolving Uncertainty	5	20	5.5
1.0	Three-Circular Obstacles	Penalty Function	5	20	13.5

Given that these times are the mean computation times and that the required update rates were between 0.5 and 1.5 seconds, it is safe to say that an improvement in computation time of at least one order of magnitude is necessary to consider real world application.

An order of magnitude improvement may seem unrealistic, but several methods exist to cover this in the near future. First, the computer utilized for these test results was more than two years old. At the time of this research, commercially available computers reach speeds in excess of 4 GHz, so a simple remedy that would more than halve the listed computation time is an upgrade in hardware. Second, a significant software burden limits the productivity of the current hardware setup. The aforementioned computer operates with a Windows® 2000 operating system and the DIDO program functions with MATLAB code. In the thesis by Moon [47], it was concluded that converting DIDO to a C program and running it off of a Linux-based computer would improve the operating times by a factor of 10 to 100. A similar solution is proposed in [45]. Third, the coded optimization problem can be optimized for computation speed through problem

reformulation or detailed scaling and balancing [48, 39]. Fourth, the DIDO optimization tool, which is used in this work only as a means of evidencing path planning ideas, has several upcoming enhancements planned that can and will improve solution speed [49]. In the very least, newer versions will permit the user to input the problem Jacobian. This will save computation time because current versions find it numerically.

To instantiate these claims with a simple example, the global information problems were rerun on a newer computer. The computer was a Dell GX280 with 1GB RAM, a 3.8 GHz processor, and the operating system was Windows XP. Under identical parameters to the results shown in Table 4, a problem was run that utilized the Penalty Function method with 20-node solutions, a 0.5-second update rate, and a three-DOF state vector. A histogram of the computation times is shown in Figure 93.

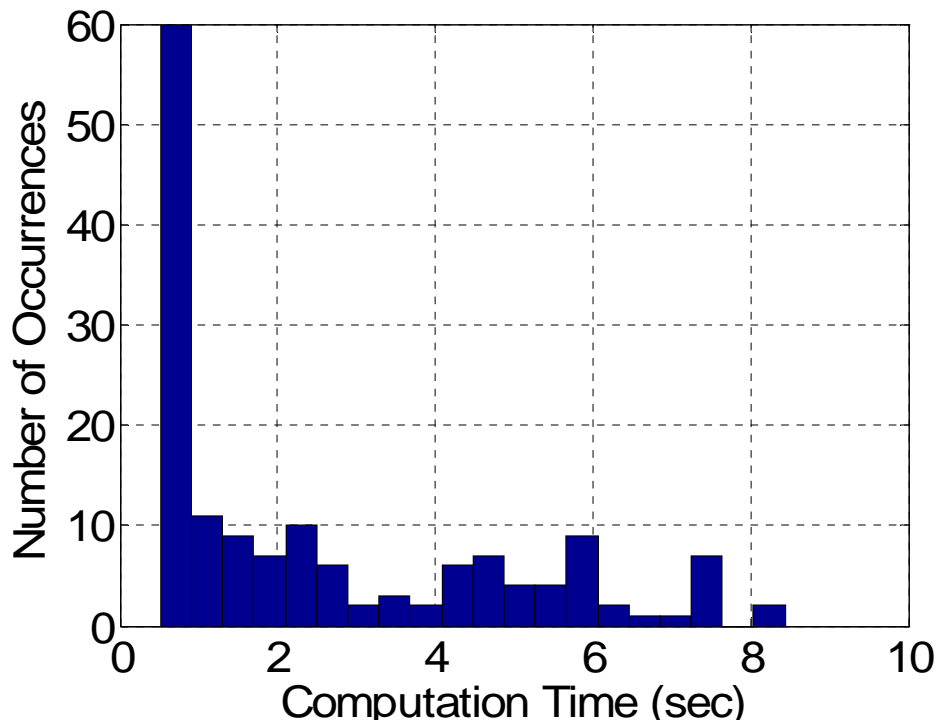


Figure 93 Histogram of Computation Times for Global Knowledge Problem Solved with Penalty Function Method and Three-DOF State Vector.

The resulting average computation time was 2.6 seconds, roughly 40% of the previous computation time. A similar simulation was run using the Resolving Uncertainty method,

with the parameters equivalent to those shown in Table 4. The average computation in this simulation was 1.4 seconds. Once again, this is roughly 40% of the computation time recorded from the original simulation. This is clearly close to the one-second required update rate; the maximum recorded computation time was approximately 3.5 seconds. While this simple hardware change does not achieve the order of magnitude improvement needed, to provides a 60% improvement using hardware technology over one year old. By utilizing current hardware and incorporating the other improvements mentioned above, the operation of real-time, online optimal path planning is within the realm of the possible.

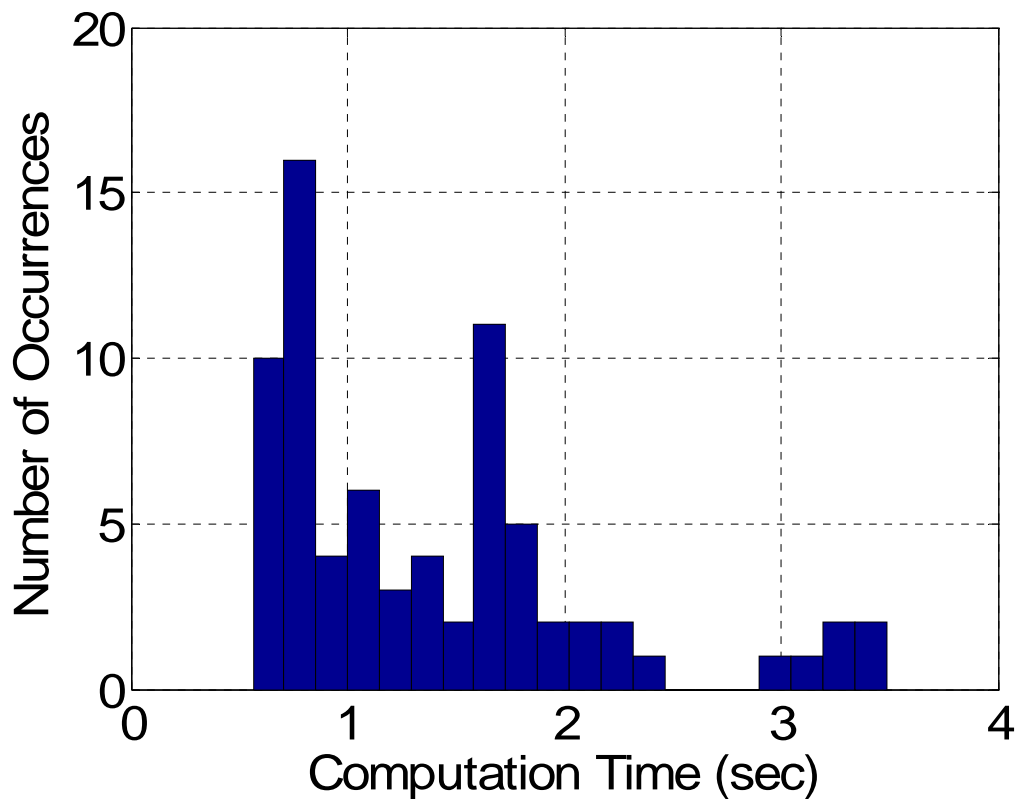


Figure 94 Histogram of Computation Times for Global Knowledge Problem Solved with Resolving Uncertainty Method and Three-DOF State Vector.

C. CONCLUSIONS

The main thrust of this work was geared toward the application of optimal control techniques to the path planning problem of unmanned vehicles, and this section evaluated that problem with significant depth from the standpoint of an unmanned ground vehicle. The open-loop analysis built upon the similar work presented for the tricycle. Constraint

representation and problem formulation differed little; however, a variety of problems were evaluated including moving targets, moving obstacles, complex, close-quarters maneuvering, and planning under the condition of a control failure. Other path planning techniques would not be able to solve this assortment of problems so quickly and easily; this provides a window to the versatility of optimal control methods.

Control of an unmanned vehicle by the generated optimal control trajectory would require a means to handle uncertainty or new sensor information. Two basic architectures exist to achieve this goal. First, a path tracking program could follow a continually updated optimal trajectory. Second, the optimal control method could directly provide the control input. In either case, the presence of uncertainty changes the problem formulation since the standard optimal control problem with path constraints can suffer from solution infeasibilities in the face of uncertainty. This concern led to a redefinition of path optimality to include a balance between time optimality and path feasibility.

Two simple techniques were introduced to circumvent the issues that the standard optimal control problem has with uncertainty. One method, the Penalty Function approach, removed the path constraints and added an obstacle proximity penalty to the cost function. The second method, the Resolving Uncertainty approach, treated the obstacle area with a level of uncertainty that was related to the distance between the obstacle and the vehicle. Adding a zone of exclusion or buffer to each obstacle provided the room necessary to account to for the system uncertainty. Neither approach is intended to be the only solution to this problem; however, they do to evidence the fact that the uncertainty problem can be surmounted. Each method was simulated using a Runge-Kutta algorithm to propagate the state variables and the results were evaluated to observe several characteristics including feasibility, computation time, robustness, and required update rate. This work also included analysis on the effects of increased model fidelity; the state vector was increased to five variables so that the rate of control variable change could be limited. In general, the increase in state vector size permitted a less frequent update in control solution; however, the computation time also increased.

All path planning simulations involved a clock stop so that the simulated vehicles did not incur a loss of adequate control from the extended computation times. Analysis of the actual computation times revealed solution updates that arrived an order of magnitude slower than the experimentally derived minimum update rate. Several solutions were proposed that could resolve this discrepancy in future work that involve currently available technologies and methods. As a simple example, two problems were executed on a newer computer system and a 60% reduction in computation time was recorded.

Significant analysis is required in the future, before these path planning techniques can be implemented on a real system. Ignoring the other robotics related problems, i.e. sensing, mapping, and localization, several concerns still exist. Complex obstacle configurations and poor initial guesses have been shown to create infeasible and incomplete solutions with DIDO; further research must explore how to mitigate this concern. One concept that may alleviate this concern during actual implementation is dual micro-processor utilization, Figure 95.

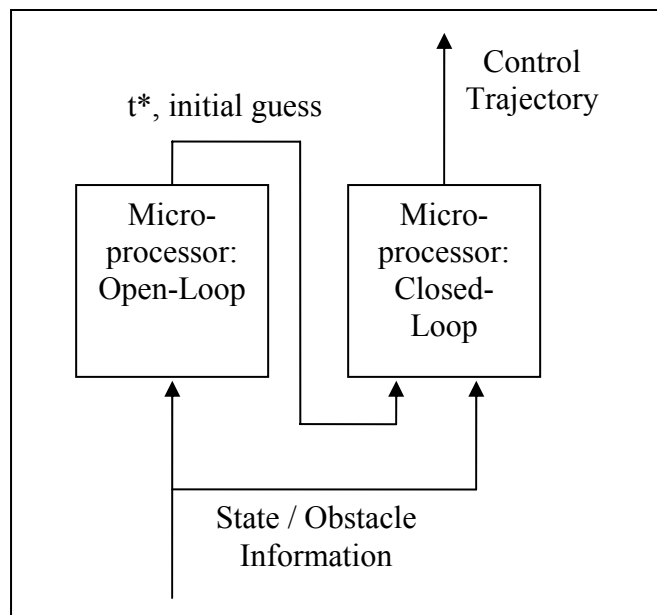


Figure 95 Illustration of the Dual Micro-Processor Concept.

In this concept, one processor is used for in an open-loop manner, and it provides the initial guess to the closed-loop processor. It also provides the optimal maneuver time, t^* , for the maximum allowable maneuver time in the case of the Penalty Function approach. The open-loop processor inputs are the state and obstacle information. With no initial guess, the solution would take longer, but it would not be biased or suffer from a poor previous solution. This concept may not solve all of the infeasibility problems, but it can offer several advantages. More simulation is necessary to understand where the weaknesses exist.

In conclusion, this section illustrates two key points. First, the technology and techniques are available to incorporate optimal control techniques into the path planning problem of robotics. Second, optimal control techniques have the capability to be used successfully to solve the path planning problem, and they provide many advantages over past methods.

V. UNMANNED AERIAL VEHICLE (UAV)

A. MOTIVATION

Unmanned Aerial Vehicles (UAVs) are already integral assets to battle field commanders in the military. Their popularity has inspired significant research and development, i.e. the Air Force's Predator and Global Hawk and the Army's Future Combat System (FCS) UAVs. Usually these vehicles are piloted remotely and even though autonomous offensive actions are unlikely to be incorporated into their design in the near future, greater autonomy can allow UAVs to require less human oversight and achieve better performance. Autonomous capabilities would enable UAVs to operate, maneuver, or land successfully without an established communications link, providing greater reliability. Further, reduced manpower needs, in terms of watchstanding and training, equates to significant monetary savings. Instead of a watchstanding team responsible for one UAV, one team could be responsible for a group or swarm of UAVs.

With respect to the DIDO implementation of the UAV problem, the addition of a third spatial dimension creates little change with respect to formulation. Transforming the code from a car problem to the UAV problem occurs in a matter of minutes to hours, instantiating the portability of the DIDO algorithm. The same constraint representation principles derived for the tricycle and car are easily adapted for the three dimensional space of the UAV, and this fact is leveraged to analyze a downtown scenario for a UAV. Most UAVs operate at an altitude too high for concern of obstacles such as buildings, but this is driven by their large size and remote nature of control. Smaller, personnel-carried and operated UAVs with autonomous guidance, navigation, and control (GNC) systems onboard could allow greater combat flexibility and are more inline with FCS concepts. These future systems demand the ability of tasking at the strategic level, and an autonomous GNC system would be a key portion of meeting this requirement.

B. PROBLEM DEFINITION

This research employs simple UAV kinematics, which, in part, reflects the fact that future personnel-carried UAVs will not necessarily resemble contemporary airplane dynamics. By developing optimal kinematic trajectories, an online path tracking system could follow the generated trajectory.

Figure 96, which is modified from [50], diagrams the UAV kinematics utilized. There are three state variables and three control variables. The state variables represent the three spatial degrees of freedom. The three control variables are the vehicle velocity, v , the flight path angle, γ , and the heading angle, ξ . The flight path angle is measured from the local horizontal to the velocity vector, and the heading angle is measured from a reference heading – e.g. due North – to the velocity vector.

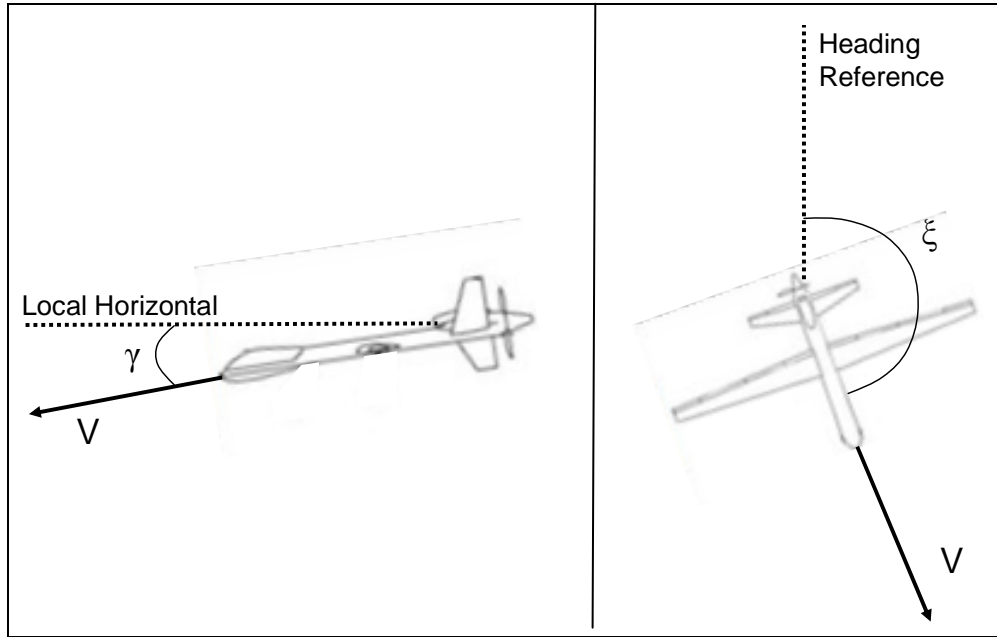


Figure 96 UAV Kinematic Diagram (After: Ref. 50)

The scaled problem formulation is presented in Equation Set (36) and (37). The scaling constants are apparent as the capital version of the lower case variables.

$$\begin{aligned} \bar{\underline{x}} &= \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} \in R^3 \\ \bar{\underline{u}} &= \begin{bmatrix} \bar{v} \\ \bar{\gamma} \\ \bar{\xi} \end{bmatrix} \in U = \left\{ \begin{array}{l} \bar{v} : \bar{v}_{\min} \leq \bar{v}(\bar{t}) \leq \bar{v}_{\max} \\ \bar{\gamma} : \bar{\gamma}_{\min} \leq \bar{\gamma}(\bar{t}) \leq \bar{\gamma}_{\max} \\ \bar{\xi} \in R \end{array} \right\} \end{aligned} \quad (36)$$

$$\begin{aligned}
& \underline{\bar{x}} \in R^3 \quad \underline{\bar{u}} \in U \\
\text{Minimize} \quad & \bar{J}[\underline{\bar{x}}, \underline{\bar{u}}, \bar{t}_f] = \bar{t}_f = t_f / T \\
\text{Subject to} \quad & \\
& \dot{\underline{\bar{x}}} = \begin{bmatrix} \bar{v} \cdot \cos(\Gamma \bar{\gamma}) \cdot \cos(\Xi \bar{\xi}) \cdot \frac{T \cdot V}{X} \\ \bar{v} \cdot \cos(\Gamma \bar{\gamma}) \cdot \sin(\Xi \bar{\xi}) \cdot \frac{T \cdot V}{Y} \\ \bar{v} \cdot \sin(\Gamma \bar{\gamma}) \cdot \frac{T \cdot V}{Z} \end{bmatrix} \quad (37) \\
& \underline{\bar{x}}^T(\bar{t}_o) = [x^i, y^i, z^i] \\
& \bar{t}_o = 0 \\
& \underline{\bar{x}}^T(\bar{t}_f) = [x^f, y^f, z^f] \\
& h_i(\bar{x}, \bar{y}, \bar{z}) \geq 0
\end{aligned}$$

The path constraints, $h_i(\bar{x}, \bar{y}, \bar{z})$, are represented in a manner identical to the car and tricycle problems – p-norm technique – with the addition of the third spatial dimension. This permits the construction of a simulation city environment.

By creating the control Hamiltonian, the problem is rewritten as Equation Set (38).

$$\begin{aligned}
& \underline{\lambda}^T = [\bar{\lambda}_x, \bar{\lambda}_y, \bar{\lambda}_z] \\
\text{Minimize} \quad & H(\underline{\lambda}, \underline{\bar{x}}, \underline{\bar{u}}, \bar{t}) = \bar{\lambda}_x \cdot \bar{v} \cdot \cos(\Gamma \bar{\gamma}) \cdot \cos(\Xi \bar{\xi}) \cdot \frac{T \cdot V}{X} + \\
& \bar{\lambda}_y \cdot \bar{v} \cdot \cos(\Gamma \bar{\gamma}) \cdot \sin(\Xi \bar{\xi}) \cdot \frac{T \cdot V}{Y} + \bar{\lambda}_z \cdot \bar{v} \cdot \sin(\Gamma \bar{\gamma}) \cdot \frac{T \cdot V}{Z} \quad (38) \\
\text{Subject to} \quad & \underline{\bar{u}} \in \bar{U} = \left\{ \begin{array}{l} \bar{v} : \bar{v}_{\min} \leq \bar{v}(\bar{t}) \leq \bar{v}_{\max} \\ \bar{\gamma} : \bar{\gamma}_{\min} \leq \bar{\gamma}(\bar{t}) \leq \bar{\gamma}_{\max} \\ \bar{\xi} \in R \end{array} \right\} \\
& h_i(\bar{x}, \bar{y}, \bar{z}) \geq 0
\end{aligned}$$

The Lagrangian of the Hamiltonian incorporates the control and path constraints and it is presented in Equation (39).

$$\begin{aligned}
\bar{H}(\underline{\lambda}, \underline{\bar{x}}, \underline{\bar{u}}, \bar{t}) = & \bar{\lambda}_x \cdot \bar{v} \cdot \cos(\Gamma \bar{\gamma}) \cdot \cos(\Xi \bar{\xi}) \cdot \frac{T \cdot V}{X} + \bar{\lambda}_y \cdot \bar{v} \cdot \cos(\Gamma \bar{\gamma}) \cdot \sin(\Xi \bar{\xi}) \cdot \frac{T \cdot V}{Y} + \\
& \bar{\lambda}_z \cdot \bar{v} \cdot \sin(\Gamma \bar{\gamma}) \cdot \frac{T \cdot V}{Z} + \bar{\mu}_v \cdot \bar{v} + \bar{\mu}_\gamma \cdot \bar{\gamma} + \bar{\mu}_\xi \cdot \bar{\xi} + \bar{\mu}_{h,i} h_i + \dots \quad (39)
\end{aligned}$$

The Karush-Kuhn-Tucker (KKT) Conditions for the Hamiltonian Minimization Condition (HMC) are given in Equation Set (40).

$$\begin{aligned}
\frac{\partial \bar{H}}{\partial \bar{u}} = \underline{0} = & \\
\left[\begin{array}{l} \bar{\lambda}_x \cos(\Gamma \bar{\gamma}) \cos(\Xi \bar{\xi}) \frac{T \cdot V}{X} + \bar{\lambda}_y \cos(\Gamma \bar{\gamma}) \sin(\Xi \bar{\xi}) \frac{T \cdot V}{Y} + \bar{\lambda}_z \sin(\Gamma \bar{\gamma}) \frac{T \cdot V}{Z} + \bar{\mu}_v \\ -\bar{\lambda}_x \bar{v} \sin(\Gamma \bar{\gamma}) \cos(\Xi \bar{\xi}) \frac{T \cdot V \cdot \Gamma}{X} - \bar{\lambda}_y \bar{v} \sin(\Gamma \bar{\gamma}) \sin(\Xi \bar{\xi}) \frac{T \cdot V \cdot \Gamma}{Y} + \bar{\lambda}_z \bar{v} \cos(\Gamma \bar{\gamma}) \frac{T \cdot V \cdot \Gamma}{Z} + \bar{\mu}_\gamma \\ -\bar{\lambda}_x \bar{v} \cos(\Gamma \bar{\gamma}) \sin(\Xi \bar{\xi}) \frac{T \cdot V \cdot \Xi}{X} + \bar{\lambda}_y \bar{v} \cos(\Gamma \bar{\gamma}) \cos(\Xi \bar{\xi}) \frac{T \cdot V \cdot \Xi}{Y} \end{array} \right] & \\
\bar{\mu}_v \begin{cases} \leq 0 & \text{for } \bar{v}(\bar{t}) = \bar{v}_{\min} \\ = 0 & \text{for } \bar{v}_{\min} < \bar{v}(\bar{t}) < \bar{v}_{\max} \\ \geq 0 & \text{for } \bar{v}(\bar{t}) = \bar{v}_{\max} \end{cases} & \\
\bar{\mu}_\gamma \begin{cases} \leq 0 & \text{for } \bar{\gamma}(\bar{t}) = \bar{\gamma}_{\min} \\ = 0 & \text{for } \bar{\gamma}_{\min} < \bar{\gamma}(\bar{t}) < \bar{\gamma}_{\max} \\ \geq 0 & \text{for } \bar{\gamma}(\bar{t}) = \bar{\gamma}_{\max} \end{cases} & \\
\bar{\mu}_{h,i} \begin{cases} \leq 0 & \text{for } h(\bar{x}, \bar{y}, \bar{z}) = 0 \\ = 0 & \text{for } h(\bar{x}, \bar{y}, \bar{z}) > 0 \end{cases} & \\
\end{aligned} \tag{40}$$

The UAV will most often fly at a non-maximal flight path angle; therefore, $\bar{\mu}_\gamma$ is expected to equal zero throughout most trajectories. On the contrary, given the time optimality criterion, the maximum velocity should be used throughout the trajectory, and this should force $\bar{\mu}_v$ to be greater than zero. Equation (41) shows the derivation of the Adjoint Equation, and this in conjunction with the KKT conditions implies that the costates remain constant unless the trajectory touches an obstacle.

$$\dot{\bar{\lambda}} = -\frac{\partial \bar{H}}{\partial \bar{x}} = \begin{bmatrix} \dot{\bar{\lambda}}_x \\ \dot{\bar{\lambda}}_y \\ \dot{\bar{\lambda}}_z \end{bmatrix} = \begin{bmatrix} \bar{\mu}_{h,i} \cdot \frac{\partial h_i}{\partial \bar{x}} + \dots \\ \bar{\mu}_{h,i} \cdot \frac{\partial h_i}{\partial \bar{y}} + \dots \\ \bar{\mu}_{h,i} \cdot \frac{\partial h_i}{\partial \bar{z}} + \dots \end{bmatrix} \tag{41}$$

The Terminal Transversality Condition adds no new boundary condition information, but the Hamiltonian Value Condition and Hamiltonian Evolution Equation combine to generate the Hamiltonian constraint presented in Equation (42).

$$\bar{H}(\bar{\mu}, \bar{\lambda}, \bar{x}, \bar{u}, \bar{t}) = -1 \quad (42)$$

C. SOLUTION VALIDATION AND RESULTS

The results presented below utilize the starting and ending points presented in Equation (43).

$$\underline{x}^i = [0 \ 0 \ 2.5]^T \quad \underline{x}^f = [10 \ 10 \ 1]^T \quad (43)$$

Figure 97 presents the DIDO-generated, time-optimal UAV trajectory through City Environment.

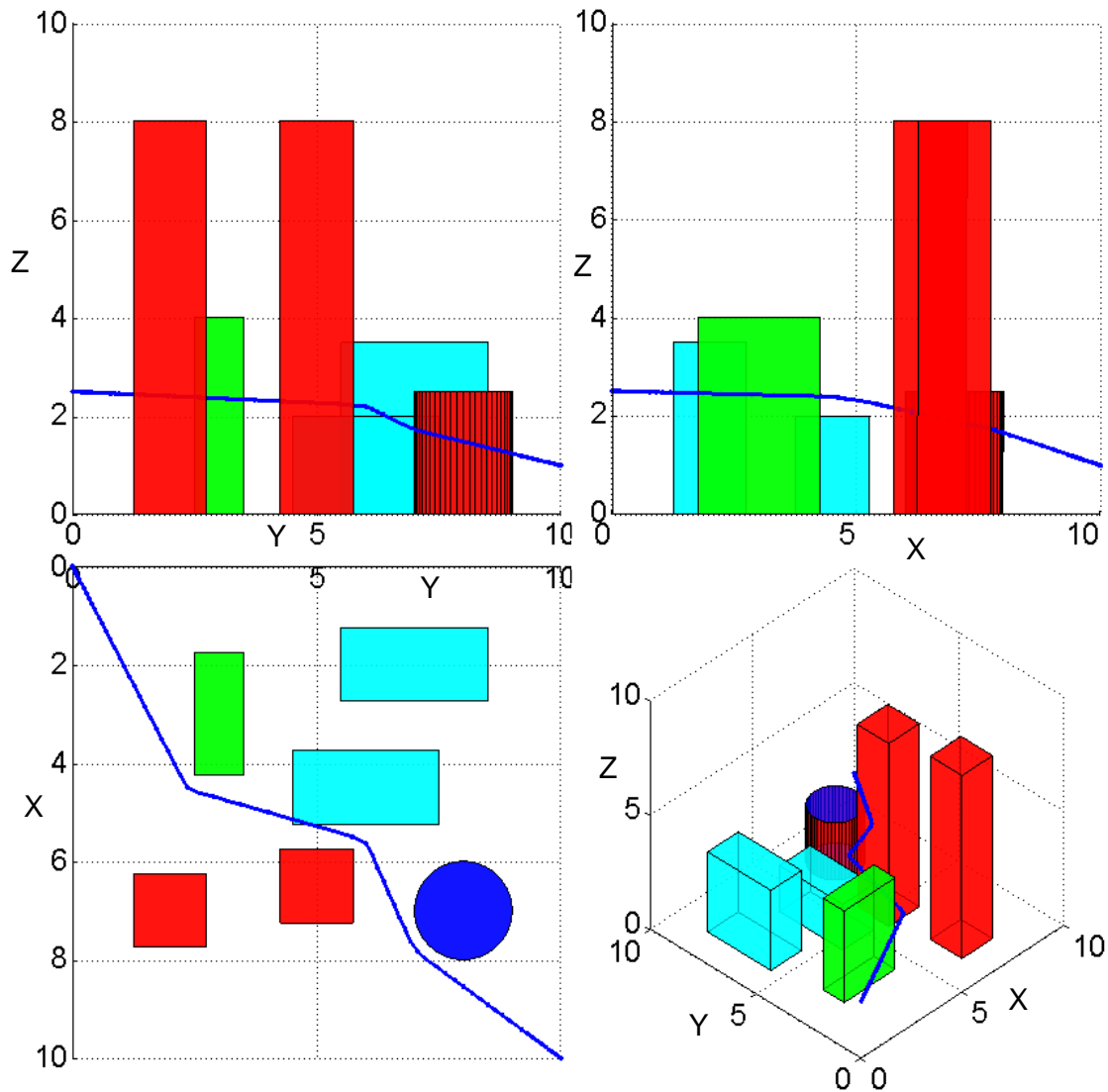


Figure 97 UAV Kinematic Trajectory through City Environment.

The buffer extended around the obstacles during constraint definition is obvious from the final trajectory. In real-world implementation, this concept could be considered a factor of safety. Figure 98 graphs the DIDO-generated solution side-by-side with the propagated solution to demonstrate feasibility.

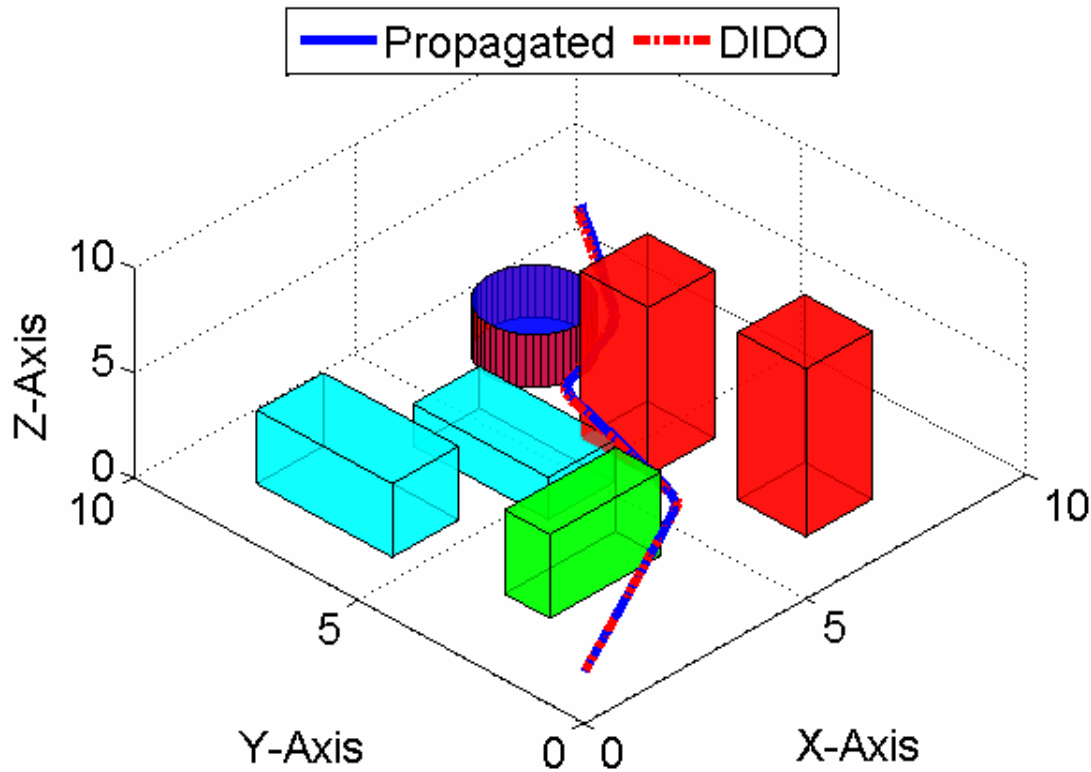


Figure 98 Propagated UAV Trajectory and DIDO-Generated UAV Trajectory.

The following four pages present the remainder of the necessary conditions and trajectory information in Figure 99 through Figure 104 including, in order of appearance: the control trajectory, the Hamiltonian value, HMC verification, the path covectors, the control covectors, and the costates.

The velocity is at a maximum value throughout the trajectory, and this forces $\bar{\mu}_v$ to be a positive value. The remaining control values are presented in degrees to aid in understanding, and the numerical variation of the solution is visible. This variation could be smoothed before implementation.

The Hamiltonian value averages to negative one but varies about this position.

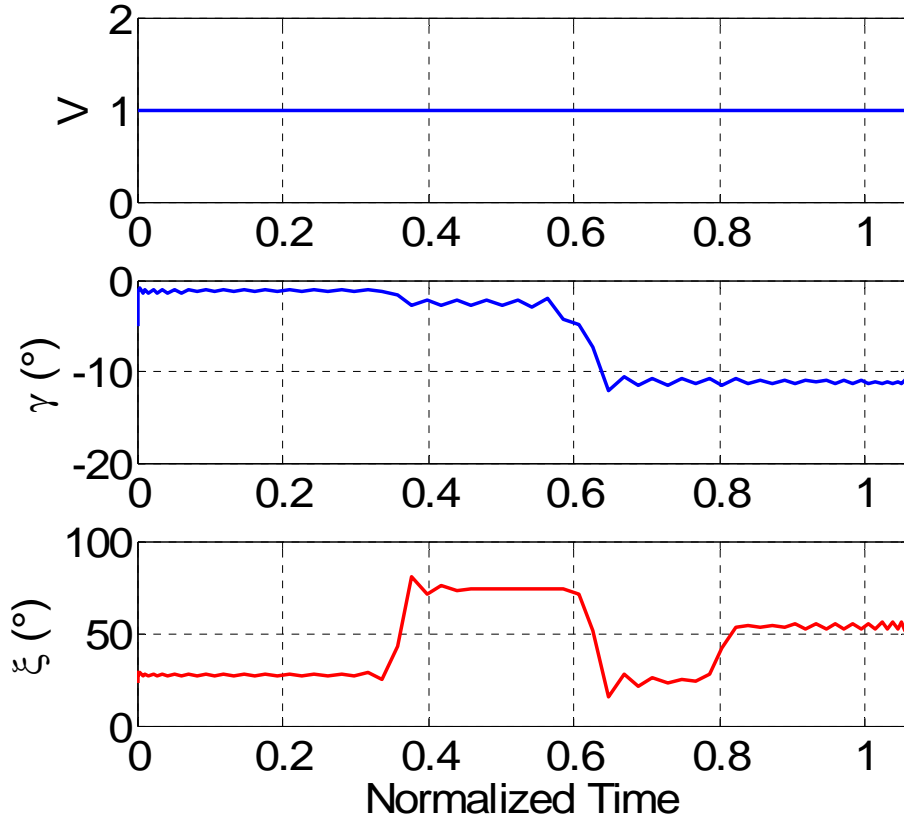


Figure 99 Control Trajectory for Optimal UAV Trajectory.

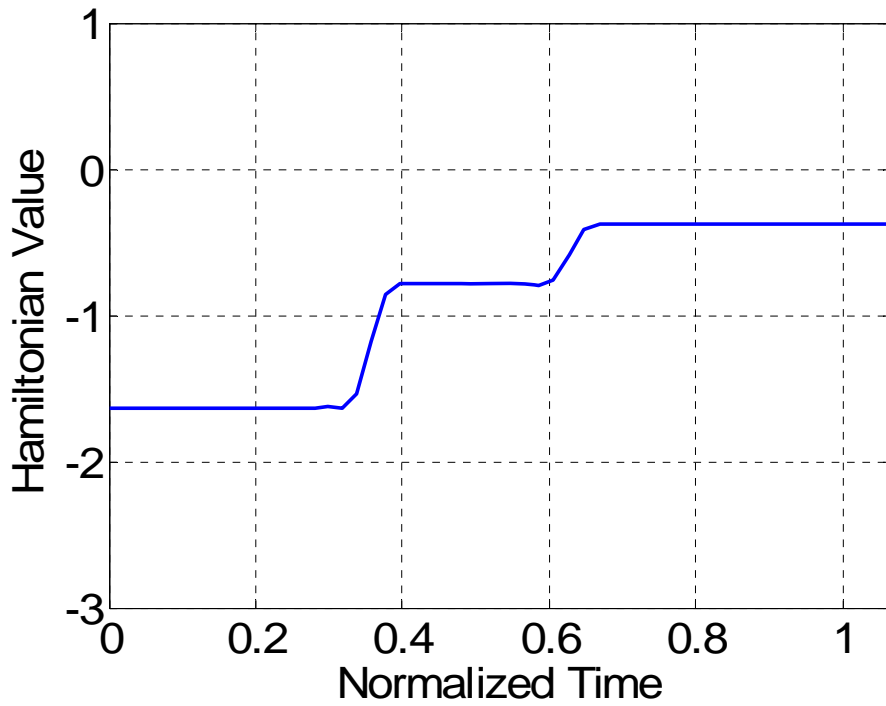


Figure 100 Hamiltonian Value for Optimal UAV Trajectory.

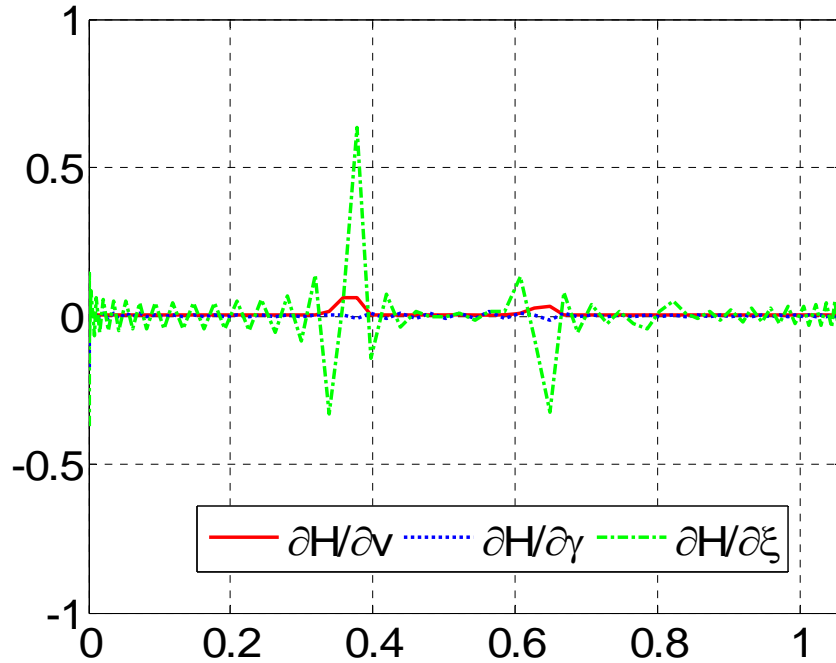


Figure 101 Hamiltonian Minimization Condition Verification for Optimal UAV Trajectory.

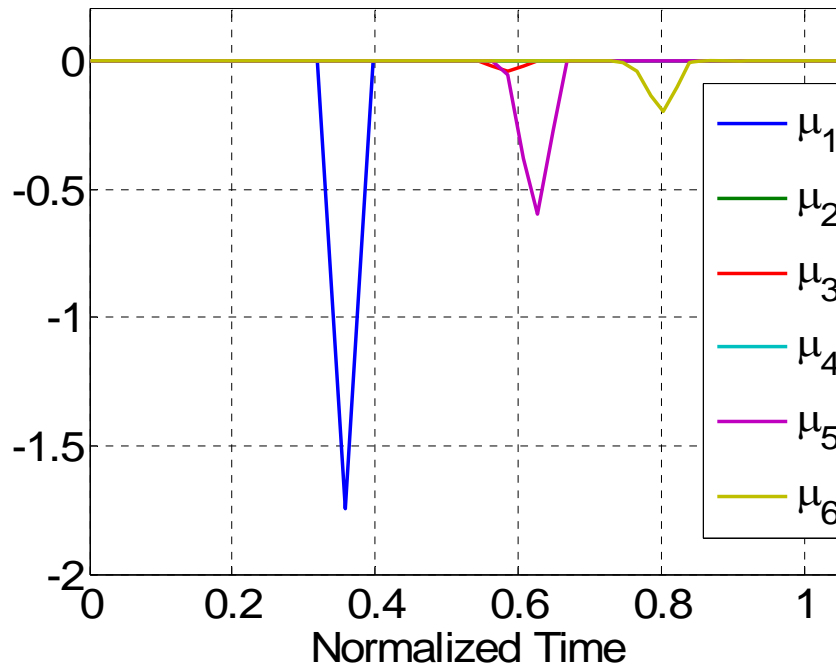


Figure 102 Path Covectors for Optimal UAV Trajectory.

The HMC verification further evidences the numerical error present in the system since all values should be zero. This figure alongside the figure for the path covectors

shows more information though. Obstacles one through five represent typical buildings and have sharp corners. Obstacle six has no corners, a cylinder. Obstacle six is encountered at an approximate scaled time of 0.8 while the other obstacles are encountered near times of 0.35 and 0.6. Optimization theory indicates that the associated path constraint covectors will be active and the related costates will change at these times, Equation (41), Figure 102, and Figure 104. These times also visually coincide with points of numerical imprecision in the results. This conclusion is reflected in Figure 100 and Figure 101 for the Hamiltonian and HMC verification respectively where the numerical variations occur near the scaled times of 0.35 and 0.6. In particular, the imprecision is most directly coincident with the vehicle's interaction with the buildings possessing sharp corners. To explore this concern, the effect of obstacle representation in the UAV model will be studied more in the future.

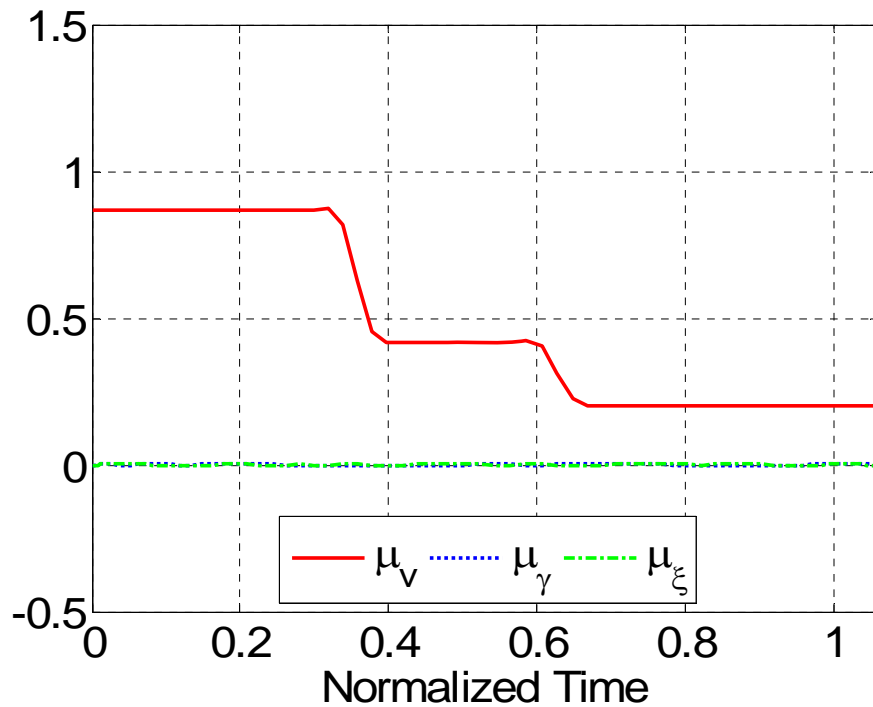


Figure 103 Control Covectors for Optimal UAV Trajectory.

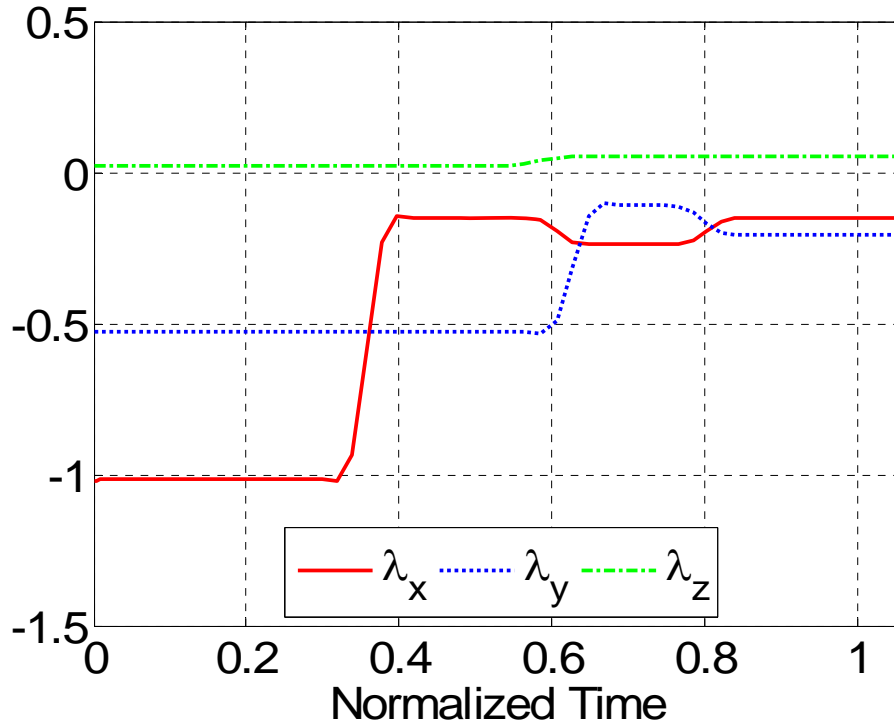


Figure 104 Costates for Optimal UAV Trajectory.

D. CONCLUSIONS

The UAV trajectory results presented here for a busy city environment are feasible, but indicate some error with respect to optimization. The source of this error appears to be associated with obstacle representation, which needs to be explored in future work. In general, the generation of a feasible trajectory is the most important key to real-world implementation and even a near-optimal result would be readily accepted. Further iterations of this analysis should evaluate the usefulness of modeling specific aircraft dynamics and control surfaces during problem definition.

VI. INVERTED PENDULUM

A. MOTIVATION

The inverted pendulum is a classic, nonlinear control problem. In its most generic form, it offers two unknowns and one control variable thus comprising an under-controlled system. While this problem is general, it shares many characteristics with other real-world control problems and is therefore an often-studied experimental problem. The particular control problem at the crux of this section is the time optimal inversion of a pendulum. The real-world system that served as a basis for simulation modeling is the Educational Control Products (ECP) Model 205a Torsional Control System with A-51 Inverted Pendulum Accessory, Figure 105, an inverted pendulum on a rotating base.



Figure 105 ECP Model 205a Torsional Control System with A-51 Inverted Pendulum Assembly.

The motivation for this particular portion of the research is primarily advancement in the knowledge and capability of utilizing optimal control methods for solving complex control problems. This problem is solved without unnecessary simplification or assumptions. The eventual goal of this work would be the application of these techniques to open-loop and closed-loop control of a pendulum. The primary challenges associated with this problem are the intricate dynamics, inherent instability, and a small time-constant.

The Torsional Control System included a MATLAB-based controller that served as an interface to the electric motor and permitted evaluation of user-defined control architectures. Specifically the Inverted Pendulum Assembly included a controller that could invert the pendulum successfully using two distinct modes of operation. The first mode successively pumped energy into the pendulum to iteratively raise its height of swing. The second mode, a simple PD technique, became active once the pendulum was within a certain angular threshold of complete inversion. This controller allowed for solution comparison once optimal results became available. Initial runs of the controller inverted the pendulum in approximately 20 seconds, but this was later refined, under separate research, to approximately 1.5 seconds.

B. PROBLEM DEFINITION

The inverted pendulum system is pictured in Figure 106. The only control variable is the torque input into the base disk, actuated by an electric motor. Supports for the system physically limit the rotation of the base disk to $\pm 65^\circ$ and input voltage to the electric motor is limited to a maximum magnitude of 5 Volts which translates to approximately 1.5N-m. All parameters and variables used to describe the problem are presented in Table 5. For the sake of brevity, the derivation of the moments of inertia for the disk and pendulum is not included here but it can be found in [51].

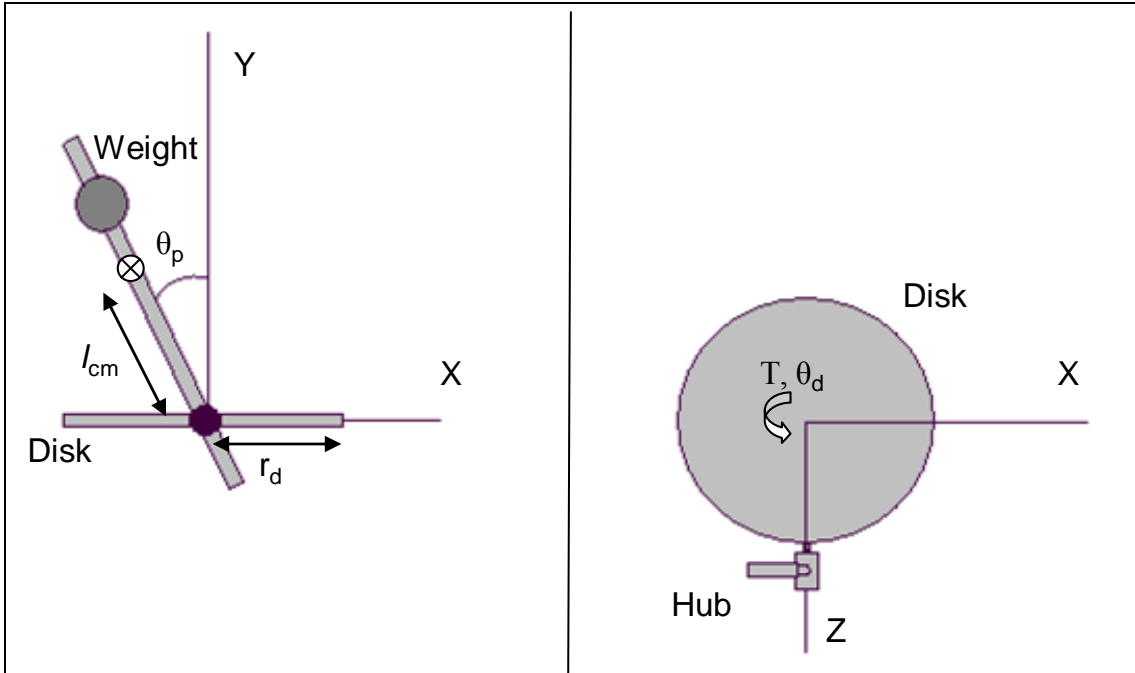


Figure 106 Inverted Pendulum on a Rotating Base Model.

Table 5 Inverted Pendulum Parameters and Variables.

Symbol	Description
M_p	Mass of pendulum
l_{cm}	Distance from rotation point to c.m.
r_d	Radius of disk
g	Acceleration due to gravity
$J_{y,d}$	Inertia of disk about y-axis
$J_{x,p}$	Inertia of pendulum about x-axis
$J_{y,p}$	Inertia of pendulum about y-axis
$J_{z,p}$	Inertia of pendulum about z-axis
T	Input torque to base disk
θ_d	Angular position of the disk
θ_p	Angular position of the pendulum

The system's equations of motion, found by Lagrange's Equation and the kinetic and potential energy, are shown in Equations (44)-(51).

$$A = M_p l_{cm} r_d \quad (44)$$

$$B = J_{x,p} - J_{y,p} + M_p l_{cm}^2 \quad (45)$$

$$C = M_p g l_{cm} \quad (46)$$

$$D = J_{z,p} + M_p l_{cm}^2 \quad (47)$$

$$E = J_{y,d} + M_p r_d^2 \quad (48)$$

$$F = J_{x,p} + M_p l_{cm}^2 \quad (49)$$

$$\begin{aligned} \ddot{\theta}_p = p(\dot{\theta}_p, \dot{\theta}_d, \theta_p, \theta_d, T) = & \\ & \frac{A \cdot \cos(\theta_p) [B \cdot \dot{\theta}_d \dot{\theta}_p \sin(\theta_p) - A \cdot \dot{\theta}_p^2 \sin(\theta_p)] + \frac{1}{2} B \cdot \dot{\theta}_d^2 \sin(2\theta_p) + C \cdot \sin(\theta_p)}{D \cdot (E + F \cdot \sin^2(\theta_p) + J_{y,p} \cos^2(\theta_p)) - (A \cdot \cos(\theta_p))^2} - & (50) \\ & \frac{A \cdot \cos(\theta_p) \cdot T}{D \cdot (E + F \cdot \sin^2(\theta_p) + J_{y,p} \cos^2(\theta_p)) - (A \cdot \cos(\theta_p))^2} \end{aligned}$$

$$\begin{aligned} \ddot{\theta}_d = d(\dot{\theta}_d, \dot{\theta}_p, \theta_p, T) = & \\ & \frac{\frac{-A \cdot \cos(\theta_p)}{D} [\frac{1}{2} B \cdot \dot{\theta}_d^2 \sin(2\theta_p) + C \cdot \sin(\theta_p)] - B \cdot \dot{\theta}_p \dot{\theta}_d \sin(2\theta_p) + A \cdot \dot{\theta}_p^2 \sin(\theta_p)}{E + F \cdot \sin^2(\theta_p) + J_{y,p} \cos^2(\theta_p) - \frac{(A \cdot \cos(\theta_p))^2}{D}} + & (51) \\ & \frac{T}{E + F \cdot \sin^2(\theta_p) + J_{y,p} \cos^2(\theta_p) - \frac{(A \cdot \cos(\theta_p))^2}{D}} \end{aligned}$$

The variables are then scaled and the problem statement is presented as Equation (52). All scaling factors are represented by a ‘U’ and the subscript denotes the particular variable to which it applies. The variables for the disk and pendulum angular positions are not scaled because these quantities are measured in radians and scaling is unnecessary. The torque and rotation limitation discussed before are introduced into the problem as a control and path constraints.

$$\begin{aligned} \bar{\underline{x}}^T &= [\theta_p, \theta_d, \dot{\theta}_p, \dot{\theta}_d] = \left[\theta_p, \theta_d, \frac{\dot{\theta}_p}{U_p}, \frac{\dot{\theta}_d}{U_d} \right] \quad \bar{u} = [\bar{T}] = \left[\frac{T}{U_T} \right] \\ \text{Minimize} \quad \bar{J}[\bar{\underline{x}}, \bar{u}, \bar{t}_f] &= \bar{t}_f = \frac{t_f}{U_t} \\ \text{Subject to} \quad \dot{\bar{\underline{x}}} &= \begin{bmatrix} \dot{\theta}_p \\ \dot{\theta}_d \\ p(\theta_p, \theta_d, \dot{\theta}_p, \dot{\theta}_d, T) \cdot \frac{U_t}{U_p} \\ d(\theta_p, \theta_d, \dot{\theta}_p, \dot{\theta}_d, T) \cdot \frac{U_t}{U_d} \end{bmatrix} \\ \bar{\underline{x}}^T(\bar{t}_o) &= [\pi, 0, 0, 0] \\ \bar{t}_o &= 0 \\ \bar{\underline{x}}^T(\bar{t}_f) &= [0, 0, 0, 0] \\ \bar{u} \in \bar{U} &= \{ \bar{T} : -1 \leq \bar{T} \leq 1 \} \\ \theta_d \in X &= \left\{ \theta_d : -\frac{65}{180}\pi \leq \theta_d \leq \frac{65}{180}\pi \right\} \end{aligned} \tag{52}$$

C. NECESSARY CONDITIONS

Equation (53) presents the inverted pendulum Hamiltonian and resulting minimization problem.

$$\begin{aligned} \underline{\lambda}^T &= [\bar{\lambda}_1, \bar{\lambda}_2, \bar{\lambda}_3, \bar{\lambda}_4] \\ \text{Minimize} \quad H(\underline{\lambda}, \bar{\underline{x}}, \bar{u}, \bar{t}) &= \bar{\lambda}_1 \dot{\theta}_p + \bar{\lambda}_2 \dot{\theta}_d + \bar{\lambda}_3 f(\bar{\underline{x}}, u, t) \cdot \frac{U_t}{U_p} + \bar{\lambda}_4 g(\bar{\underline{x}}, u, t) \cdot \frac{U_t}{U_d} \\ \text{Subject to} \quad \bar{u} \in \bar{U} &= \{ \bar{T} : -1 \leq \bar{T} \leq 1 \} \\ \theta_d \in X &= \left\{ \theta_d : -\frac{65}{180}\pi \leq \theta_d \leq \frac{65}{180}\pi \right\} \end{aligned} \tag{53}$$

The constraints are accounted for mathematically in the Lagrangian of the Hamiltonian, Equation (54).

$$\bar{H}(\bar{\underline{\mu}}, \bar{\underline{\lambda}}, \bar{\underline{x}}, \bar{u}, \bar{t}) = \bar{\lambda}_1 \dot{\theta}_p + \bar{\lambda}_2 \dot{\theta}_d + \bar{\lambda}_3 f(\bar{\underline{x}}, u, t) \cdot \frac{U_t}{U_p} + \bar{\lambda}_4 g(\bar{\underline{x}}, u, t) \cdot \frac{U_t}{U_d} + \bar{\mu}_1 \bar{u} + \bar{\mu}_2 \theta_d \tag{54}$$

The Hamiltonian Minimization Condition (HMC) is found by taking the partial derivative of Equation (54) with respect to the control variable. The HMC and the KKT conditions are presented in Equation (55).

$$\begin{aligned}
\frac{\partial \bar{H}}{\partial u} = 0 = & \bar{\lambda}_3 \cdot \frac{U_T \cdot U_t}{U_p} \left[\frac{-A \cdot \cos(\theta_p)}{D \cdot E + D^2 \cdot \sin^2(\theta_p) + (J_{y,p} \cdot D - A^2) \cdot \cos^2(\theta_p)} \right] + \\
& \bar{\lambda}_4 \cdot \frac{U_T \cdot U_t}{U_d} \left[\frac{1}{E + F \cdot \sin^2(\theta_p) + \left(J_{y,p} - \frac{A^2}{D} \right) \cdot \cos^2(\theta_p)} \right] + \bar{\mu}_1 \\
\mu_1 & \begin{cases} \leq 0 & \text{for } T = T_{\max} \\ = 0 & \text{for } T_{\max} < T < T_{\max} \\ \geq 0 & \text{for } T = T_{\max} \end{cases} \\
\mu_2 & \begin{cases} \leq 0 & \text{for } \theta_d = \theta_{d,\max} \\ = 0 & \text{for } \theta_{d,\max} < \theta_d < \theta_{d,\max} \\ \geq 0 & \text{for } \theta_d = \theta_{d,\max} \end{cases}
\end{aligned} \tag{55}$$

In this case the Adjoint Equation leads to complex formulas for $\dot{\bar{\lambda}}$, providing little information with the exception of Equation (56).

$$\dot{\bar{\lambda}}_2 = -\bar{\mu}_2 \tag{56}$$

With an endpoint specification for each of the state variables, the Terminal Transversality Condition (TTC) also provides no new information. Lastly the Hamiltonian Evolution Equation (HEE) and Hamiltonian Value Condition (HVC) lead to the common result for time optimal problems presented in Equation (57).

$$\bar{H}(\bar{\mu}, \bar{\lambda}, \bar{x}, \bar{u}, \bar{t}) = -1 \tag{57}$$

D. RESULTS

The optimal inversion trajectory received from DIDO is presented in Figure 107 and Figure 108. The optimal maneuver time is less than 0.7 seconds.

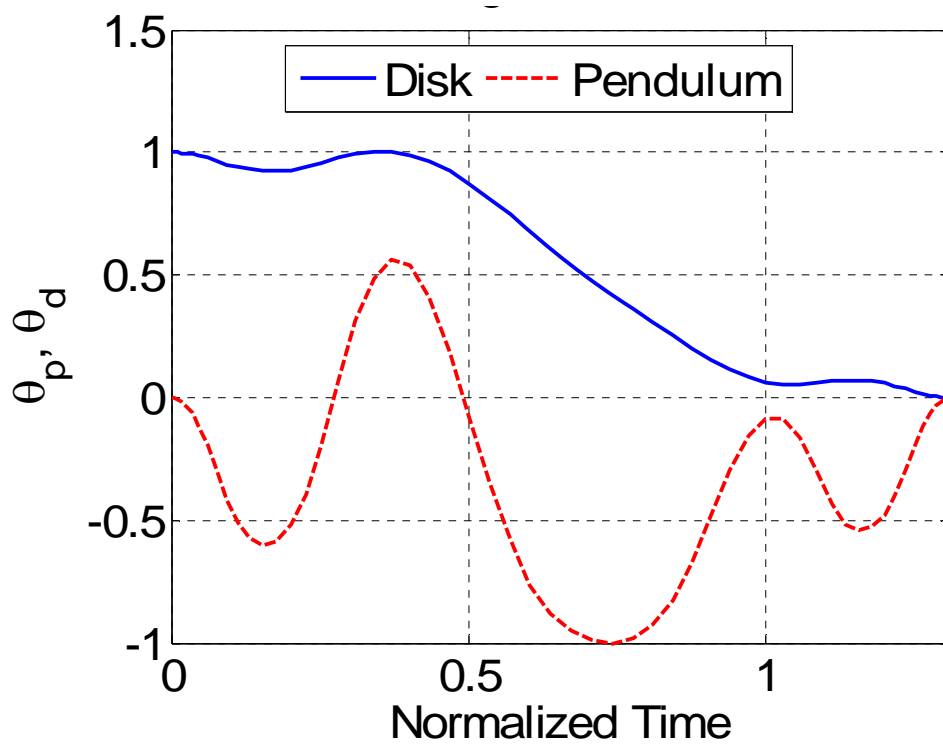


Figure 107 Pendulum and Disk Angular Trajectories for Optimal Inversion Maneuver.

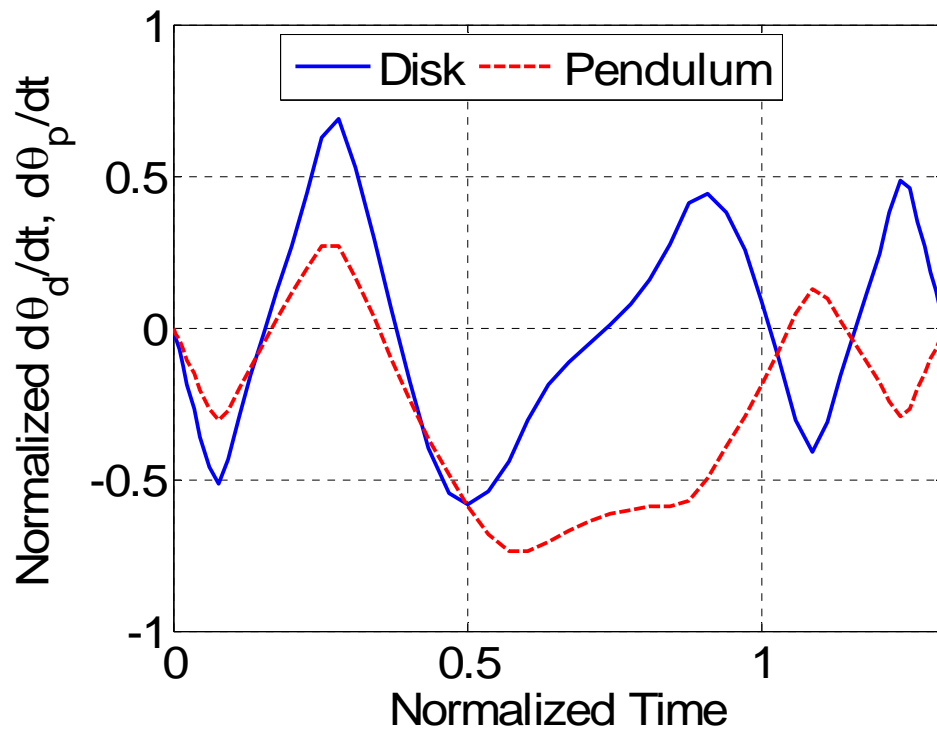


Figure 108 Pendulum and Disk Angular Velocities for Optimal Inversion Maneuver.

The optimal torque trajectory is displayed in Figure 109. The bang-bang nature of the controls is intuitive given the optimality criteria, time, and the linear relationship between the controls and the dynamics.

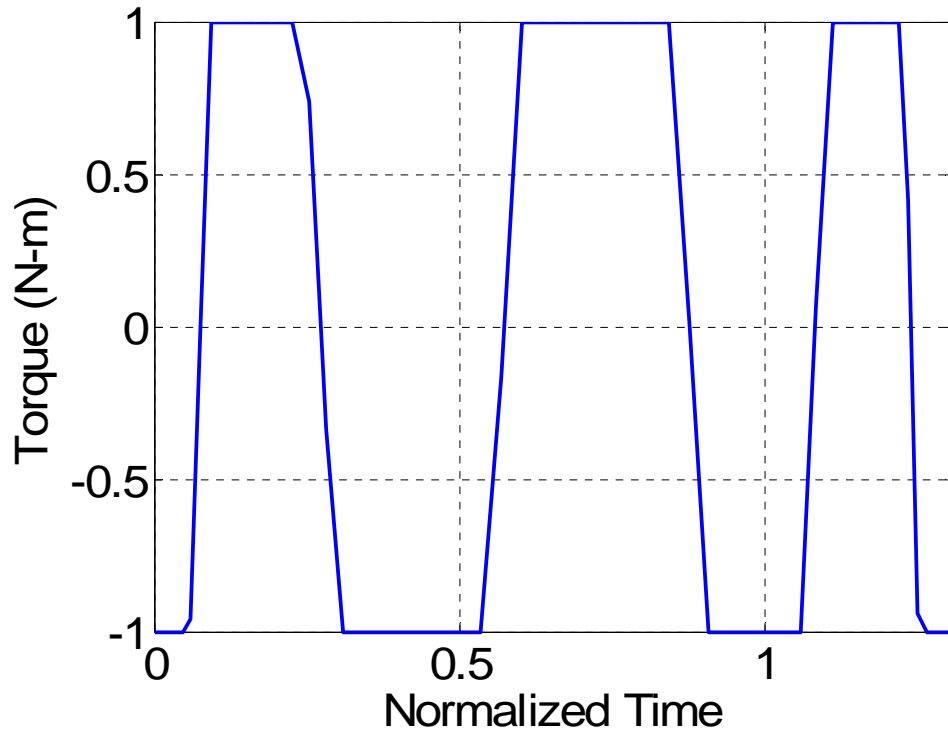


Figure 109 Torque Trajectory for Optimal Inversion Maneuver.

Figure 110 shows the control covector with the control trajectory superimposed in the background. This shows how the control covector acts as a switching function for the control trajectory. A positive value for $\bar{\mu}_1$ is associated with a positive value for the control variable and vice versa.

Figure 111 displays the Hamiltonian value throughout the trajectory. The Hamiltonian oscillates about a value of negative one which is its time average, but generally validates the necessary condition in Equation (57).

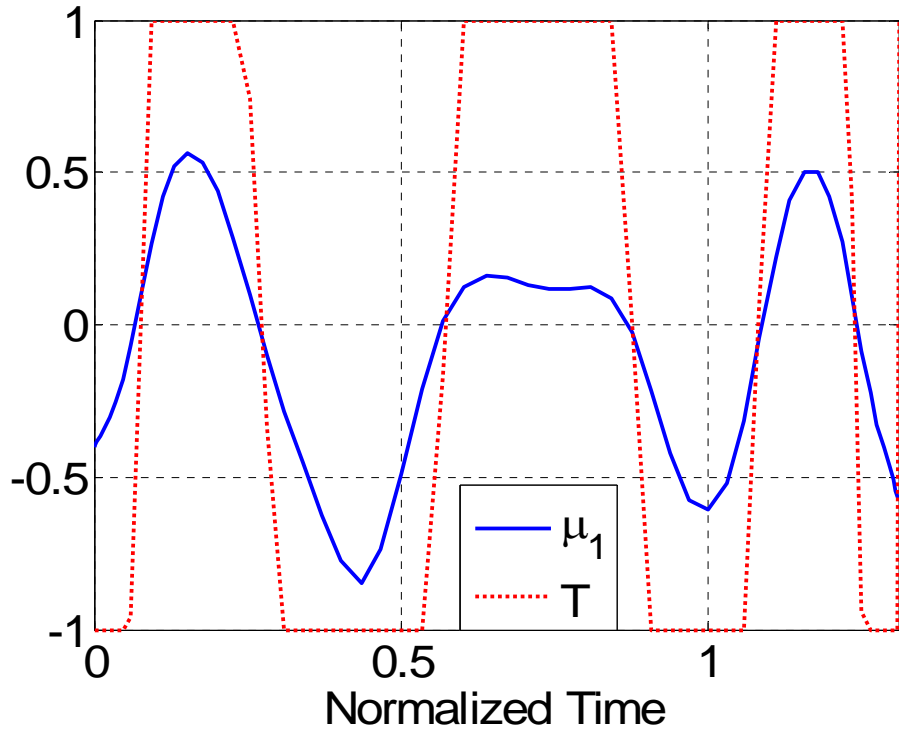


Figure 110 Control Convector with Control Trajectory in the Background to Display Switching Nature.

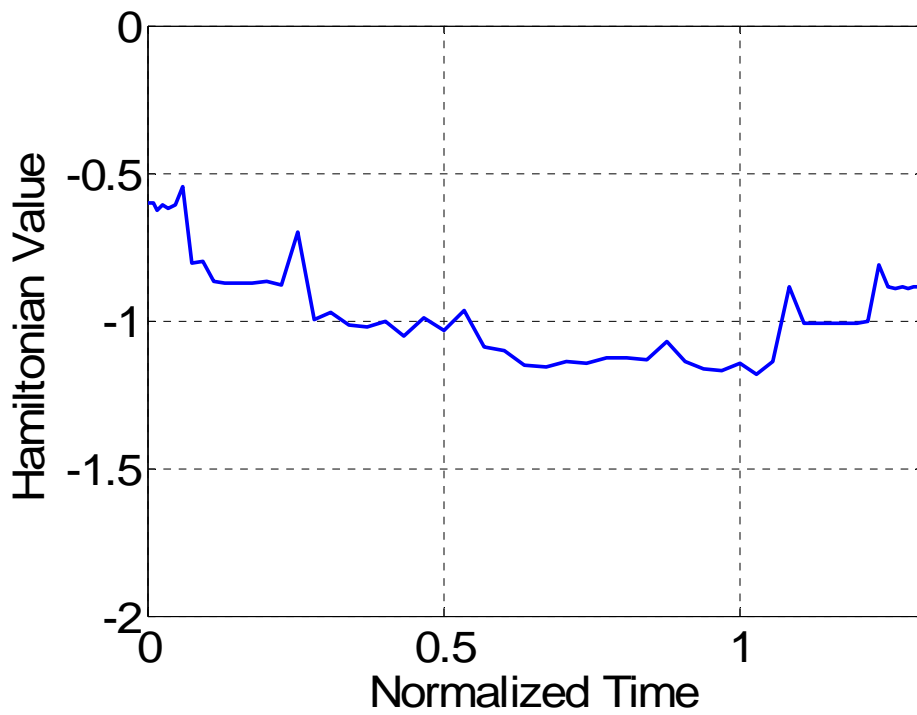


Figure 111 Hamiltonian for Pendulum Inversion Problem.

Figure 112, plot of the costates, and Figure 113, plot of the path covector, instantiate Equation (56) which equates the change in the second costate, $\bar{\lambda}_2$, to the negative of the second covector, $\bar{\mu}_2$. From the KKT conditions, $\bar{\mu}_2$ only becomes active when the system is at an extremal disk angular position, and for this reason $\bar{\lambda}_2$ must be constant whenever θ_d is not equal to the minimum or maximum. The extremal disk angular position is identified in Figure 107 to occur at a normalized time of approximately 0.75. The value is similarly verified by the activation of the path constraint covector, $\bar{\mu}_2$, and a change in the costate for the disk position, $\bar{\lambda}_2$.

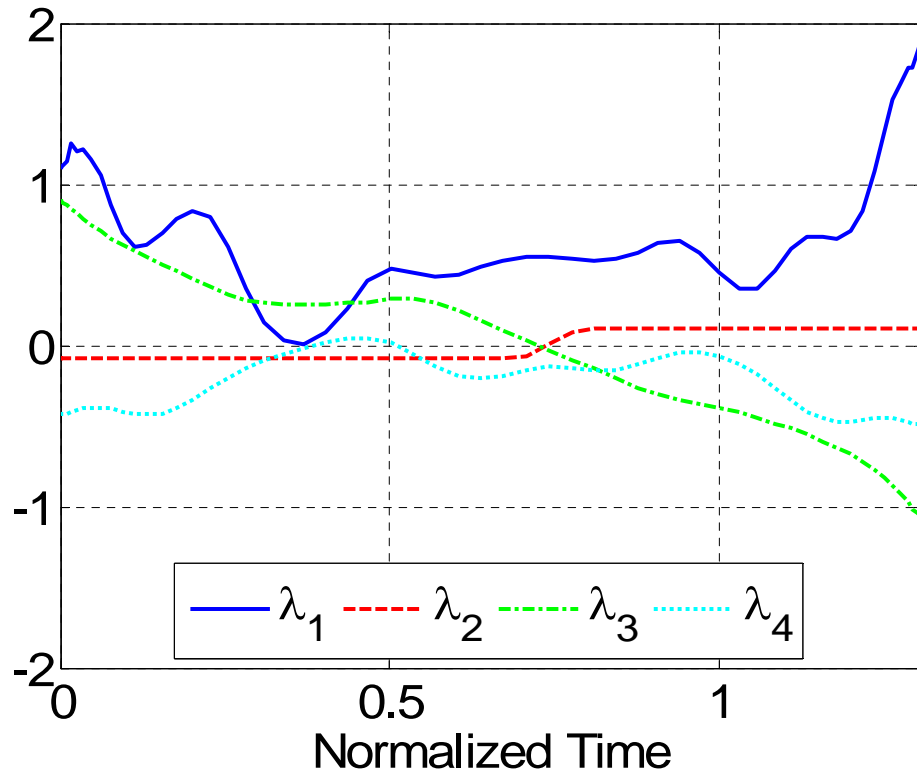


Figure 112 Pendulum Inversion Maneuver Costates.

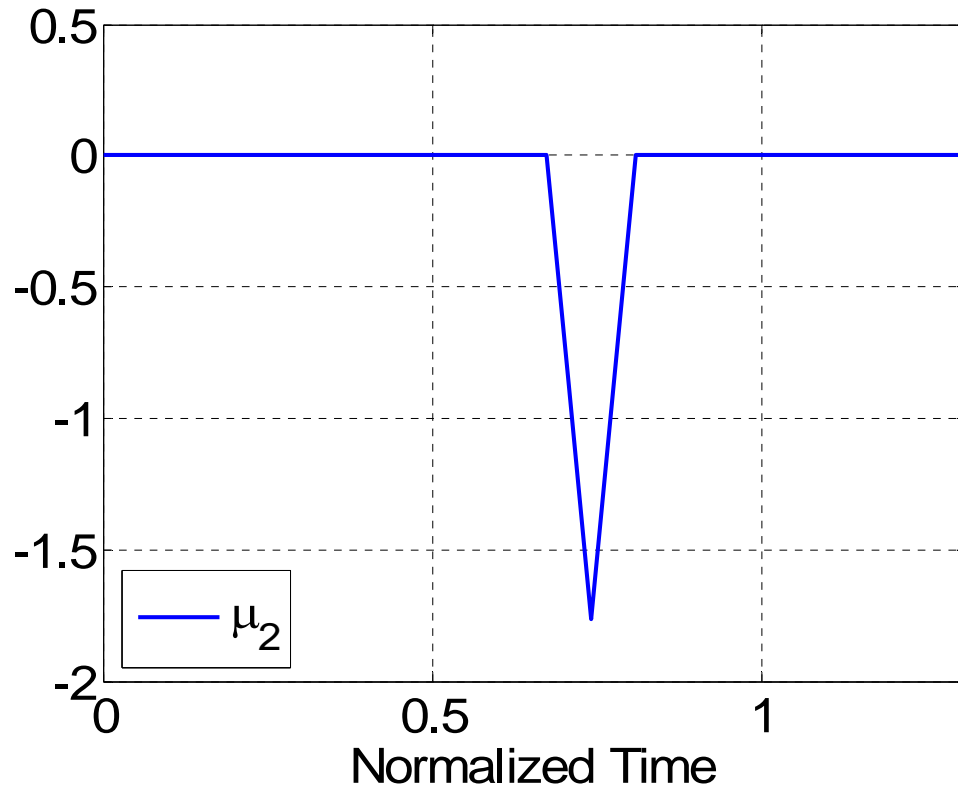


Figure 113 Pendulum Inversion Maneuver Path Covector.

Solution feasibility is verified with a Runge-Kutta algorithm and the resultant propagated and DIDO generated trajectories are presented together in Figure 114 and Figure 115. The difference between the two graphs is minimal and it can be attributed to interpolation error.

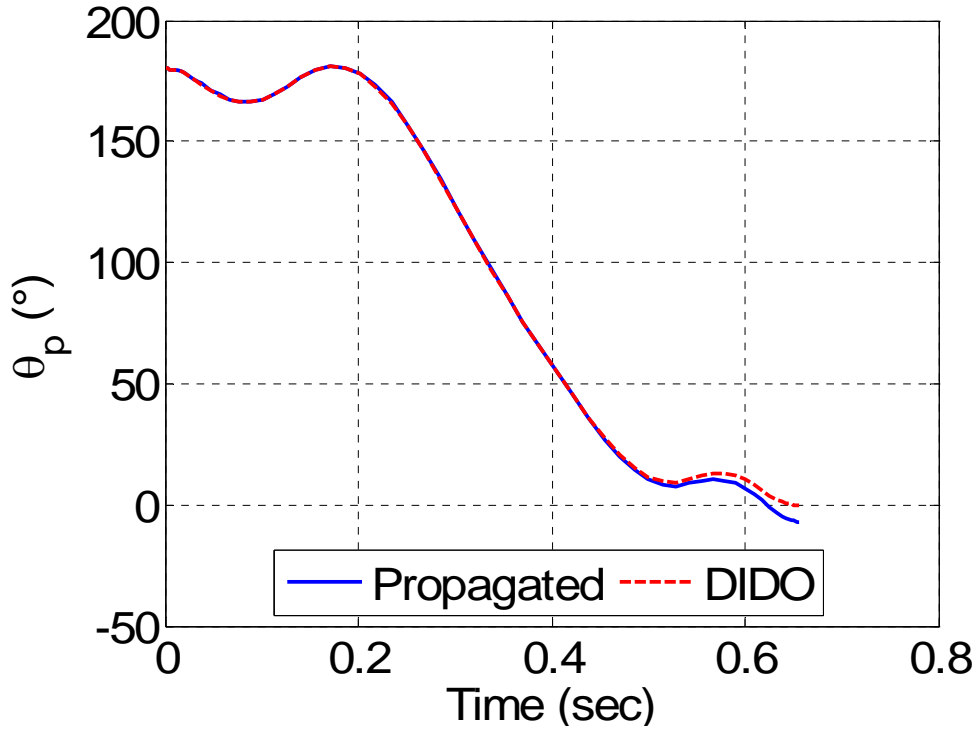


Figure 114 Pendulum Position Feasibility Verification.

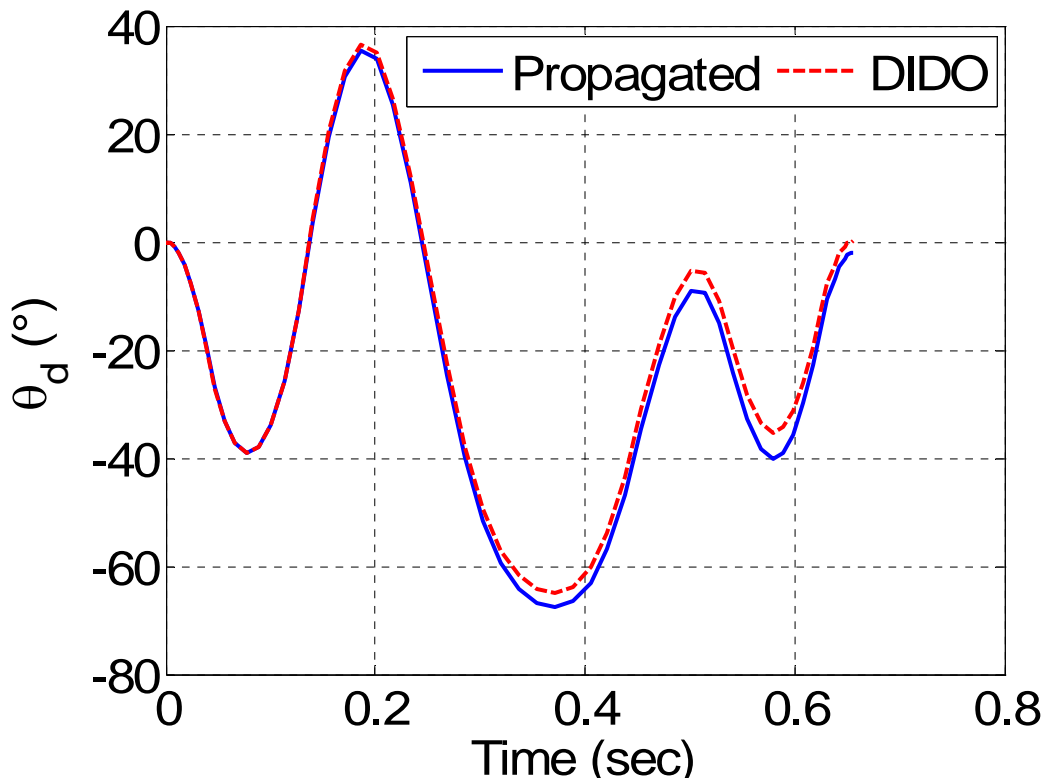


Figure 115 Disk Position Feasibility Verification.

Figure 116 provides multiple snapshots of the inversion maneuver so that it can be assessed from a visual perspective. The disk performs four successive turning maneuvers to invert the pendulum, the first eight frames below. The remainder of the frames exhibits the slowing of the pendulum's momentum.

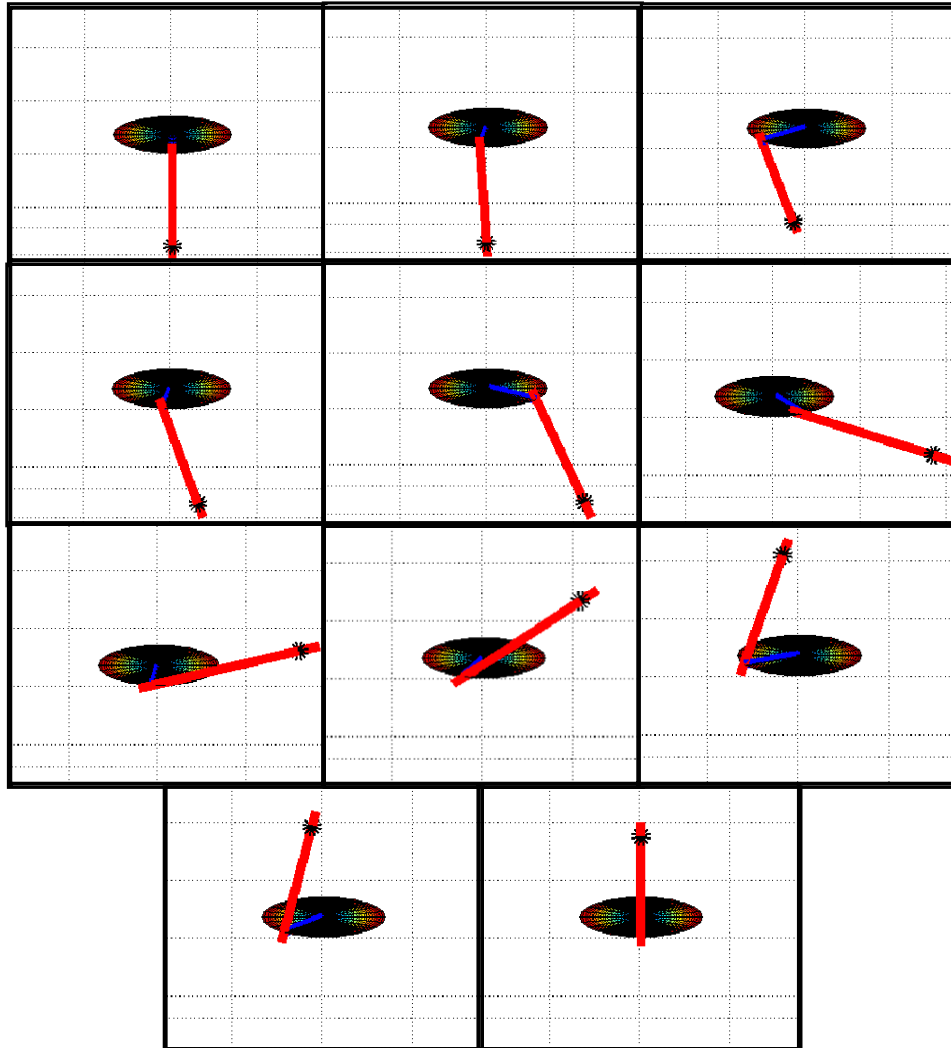


Figure 116 Images of Optimal Inversion Maneuver.

E. CONCLUSIONS

The open-loop solution generated above could be utilized immediately with the inverted pendulum assembly, but its uses would be restricted to open-loop implementation. Ensuring continual inversion would dictate the use a technique similar to the PD feedback method mentioned previously once the optimal trajectory is

completed. The optimal control trajectory would improve the inversion time and the alternate feedback method would maintain the pendulum at its final position and correct for minor disturbances. Eventual goals include the incorporation of an online optimal controller, and with the rate of computer advancements and the upcoming advancements in the DIDO algorithm it is not illogical to think that this can occur within the span of a few years.

VII. CONCLUSIONS

A. RESEARCH CONCLUSIONS

This work presented numerous examples of systems that can benefit from the application of optimal control methods to the path planning and motion control problems. In the past, path planning techniques were capable of producing feasible trajectories in an efficient manner; however, optimality and the satisfaction of complex constraints were sacrificed. Meanwhile, optimization methods have long been considered to provide attractive results, i.e. feasible, optimal, and capable of satisfying any constraint, but they were understood to be limited by solution convergence problems and inhibitive computational complexity. Advances in computer hardware and optimization algorithms have made many of these concerns a thing of the past, and this thesis provides evidence to this claim.

While several systems, e.g. tricycle, four-wheeled car, aerial vehicle, and pendulum, were analyzed in this work, the main focus was on a unmanned ground vehicle concept. Open-loop, time-optimal simulations were run to demonstrate the capability of the system to handle various scenarios involving dynamic environments and complex constraints, thereby providing evidence to the fact that this approach is rapidly reconfigurable to the current situation. Real-world applications of unmanned and autonomous systems require the ability to handle uncertainty during normal operation. Two concepts were detailed in an effort to tackle the reality of uncertainty. First, a balance between optimality and trajectory robustness was made such that solution extremals and uncertainty could not drive the vehicle into a dangerous or infeasible condition. Second, feedback was incorporated as a means to update repeatedly the control trajectory of the vehicle. This approach allowed for the successful completion of a mission where only local knowledge was available. The computation time was disregarded during simulations, but the manner in which to resolve this temporal problem was discussed in detail.

In general, optimal control techniques were used to solve a myriad of problems with intricate dynamics, complex and dynamic constraints, and uncertainty. They have

proved their value in solving the basic robotics' path planning problem in a manner that is both feasible and optimal. Previous discussions would have questioned the use of optimal control techniques with robotics as a matter of how; now the question is a matter when.

B. FUTURE WORK

The overarching goal of this research is the implementation of optimal control techniques in real-world, robotic path planning. An effort is currently underway at the Guidance, Navigation, and Control Laboratory at the Naval Postgraduate School to acquire several variants of unmanned, remotely operated vehicles – both ground and aerial – for real-world testing of path planning tools and techniques. Before this can be accomplished however, it will be necessary to advance the current research in the areas of complete control architecture, computation time, robustness, and effectiveness under complex obstacle configurations. Of course, addressing these issues resolves only a portion of the entire robotics problem, which includes sensing, localization, and mapping.

Other areas of work include analysis of these techniques for use with cooperative motion between multiple vehicles. In this arena, optimal control techniques present the ability to resolve tactical planning scenarios and strategic allocation and scheduling dilemmas. In other words, optimal control methods could decide which vehicle best replaces a lost asset and then determine the best trajectory for the new vehicle to use when replacing that asset.

LIST OF REFERENCES

1. Moravec, H., "Robots, After All," *Communications of the ACM*, Vol. 46, No. 10, pp. 90-97, October 2003.
2. National Aeronautics and Space Administration, *The Vision for Space Exploration*, Washington, D.C., February 2004.
3. National Aeronautics and Space Administration, "Mars Exploration Rovers," [http://www.nasa.gov/mission_pages/mer/index.html], October 2006.
4. 106th Congress, "National Defense Authorization Act, Fiscal Year 2001," Public Law 106-398, 30 October 2000.
5. LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006.
6. Choset, H. et al., *Principles of Robot Motion; Theory, Algorithms, and Implementation*, The MIT Press, 2005.
7. Zefran, M., *Continuous Methods for Motion Planning*, Doctoral Dissertation, pp. 1-14, University of Pennsylvania, December 1996.
8. Khatib, O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal for Robotics Research*, v.5, no.1, pp. 90-99, 1986.
9. Wang, L.C., Yong, L.S., and Ang Jr., M.H., "Hybrid of Global Path Planning and Local Navigation Implemented on a Mobile Robot in Indoor Environment," paper presented at the International Symposium on Intelligent Control, Vancouver, Canada, 27-30 October 2002.
10. Barraquand, J., Langlois, B., and Latombe, J., "Numerical Potential Field Techniques for Robot Path Planning," *IEEE Transactions on Systems, Man, and Cybernetics*, v. 22, no. 2, March/April 1992.
11. Shimoda, S., Kuroda, Y., and Iagnemma, K., "Potential Field Navigation of High Speed Unmanned Ground Vehicles on Uneven Terrain," paper presented at the International Conference on Robotics and Automation, Barcelona, Spain, April 2005.
12. Vadakkepat, P., Tan, K.C., and Ming-Liang, W., "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning," *IEEE*, 2000.
13. Poty, A., Melchior, P., and Oustaloup, A., "Dynamic Path Planning for Mobile Robots Using Fractional Potential Field," *IEEE*, 2004.

14. Loizou, S.G. and Kyriakopoulos, K.J., "Closed Loop Navigation for Multiple Non-Holonomic Vehicles," *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, 14-19 September 2003.
15. Ito, P., "Construction Probabilistic Roadmaps with Powerful Local Planning and Path Optimization," *Proceedings of the 2002 IEEE International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.
16. Kazemi, M. and Mehrandezh, M., "Robotic Navigation Using Harmonic Function-based Probabilistic Roadmaps," *Proceedings of the 2004 IEEE International Conference on Intelligent Robots & Automation*, New Orleans, LA, April 2004.
17. Kallmann, M. and Mataric, M., "Motion Planning Using Dynamic Roadmaps," *Proceedings of the 2004 IEEE International Conference on Intelligent Robots & Automation*, New Orleans, LA, April 2004.
18. Lingelbach, F., "Path Planning for Mobile Manipulation using Probabilistic Cell Decomposition," *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2 October 2004.
19. Xu, B., Chen, D.Z., and Szczerba, R.J., "A New Algorithm and Simulation for Computing Optimal Paths in a Dynamic and Weighted 2-D Environment," *IEEE*, 2000.
20. Rosell, J. and Iniguez, P., "Path Planning using Harmonic Functions and Probabilistic Cell Decomposition," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.
21. Lingelbach, F., "Path Planning Using Probabilistic Cell Decomposition," *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, LA, 2004.
22. Kavraki, L.E., Latombe, J.C., Svestka, P., and Overmars, M.H., "Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, v. 12, no. 4, June 1996.
23. LaValle, S.M. and Kuffner, J.J., "Rapidly-exploring Random Trees: Progress and Prospects," *Algorithmic and Computational Robotics: New Directions*, pp. 293-308. A K Peters, Wellesley, MA, 2001.
24. Kim, J. and Esposito, J.M., "Adaptive Sample Bias for Rapidly-exploring Random Trees with Applications to Test Generation," Paper presented at the 2005 American Control Conference, Portland, OR, 8-10 June 2005.

25. LaValle, S.M. and Kuffner Jr., J.J., "Randomized Kinodynamic Planning," *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, Detroit, MI, May 1999.
26. Hsu, D., Latombe, J.C., and Motwani, R., "Path Planning in Expansive Configuration Spaces," *IEEE International Conference on Robotics and Automation*, 1997.
27. Plaku, E., Bekris, K.E., Chen, B.Y., Andrew, M.L., and Kavraki, L.E., "Sampling-based Roadmap of Trees for Parallel Motion Planning," *IEEE Transactions on Robotics*, v. 21, no. 4, August 2005.
28. LaValle, S.M. and Kuffner, J.J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, v. 20, no. 5, May 2001.
29. Hsu, D., Kindel, R., Latombe, J.C., and Rock, S., "Randomized Kinodynamic Motion Planning with Moving Obstacles," *International Journal of Robotics Research*, v. 21, no. 3, March 2002.
30. Frazzoli, E., Dahleh, M.A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, v. 25, no. 1, January-February 2002.
31. LaValle, S.M. and Hutchinson, S.A., "Optimal Motion Planning for Multiple Robots having Independent Goals," *IEEE Transactions on Robotics and Automation*, v. 14, no. 6, December 1998.
32. LaValle, S.M., "From Dynamic Programming to RRTs: Algorithmic Design of Feasible Trajectories," *Control Problems in Robotics*, pp. 19-37, Springer-Verlag, Berlin, 2002.
33. Wang, J, Qu, Z., Guo, Y., and Yang, J., "A Reduced-Order Analytical Solution to Mobile Robot Trajectory Generation in the Presence of Moving Obstacles," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, 2004.
34. Yang, J. and Qu, Z., "Trajectory Planning for UGVs in an Environment with 'Hard' and 'Soft' Obstacles," Presented at AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO, 21-24 August 2006.
35. Yang, J., Daoui, A., Gu, Z., Wang, J., and Hull, R.A., "An Optimal and Real-Time Solution to Parameterized Mobile Robot Trajectories in the Presence of Moving Obstacles," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

36. Sundar, S. and Shiller Z., "Optimal Obstacle Avoidance Based on the Hamilton-Jacobi-Bellman Equation," *IEEE Transactions on Robotics and Automation*, v. 13, no. 2, April 1997.
37. Fiorini, P. and Shiller, Z., "Time Optimal Trajectory Planning in Dynamic Environments," *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, April 1996.
38. Betts, J.T., "Survey of Numerical Methods for Trajectory Optimization," *AIAA Journal of Guidance, Control, and Dynamics*, v. 21, no. 2, pp. 193-207, March-April 1998.
39. Ross, I.M., *Lecture Notes in Control and Optimization*, Naval Postgraduate School, Monterey, CA, 1998.
40. Josselyn, S. and Ross, I.M., "Rapid Verification Method for the Trajectory Optimization of Reentry Vehicles," *J. Guidance*, v. 26, no. 3, 2002.
41. Shaffer, P.J., Ross, I.M., Oppenheimer, M.W., and Doman, D.B., "Optimal Trajectory Reconfiguration and Retargeting for a Reuseable Launch Vehicle," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, 15-18 August 2005.
42. Bollino, K.P., Oppenheimer, M.W., and Doman, D.D., "Optimal Guidance Command Generation and Tracking for Reuseable Launch Vehicle Reentry," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, 21-24 August 2006.
43. Sekhavat, P., Fleming, A., and Ross, I.M., "Time-Optimal Nonlinear Feedback Control for the NPSAT1 Spacecraft," *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Monterey, CA, 24-28 July 2005.
44. Bollino, K., Ross, I.M., and Doman, D., "Optimal Nonlinear Feedback Guidance for Reentry Vehicles," *AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, 2006.
45. Ross, I.M. and Fahroo, F., "Issues in the Real-Time Computation of Optimal Control," *Mathematical and Computer Modeling*, Pergamon Publication, 2005.
46. Ross, I.M., D'Souza, C., Fahroo, F., and Ross, J.B., "A Fast Approach to Multi-Stage Launch Vehicle Trajectory Optimization," *AIAA*.
47. Moon, R.L., "Migration of a Real-Time Optimal Control Algorithm: From MATLAB to Field Programmable Gate Array," Master's Thesis, Naval Postgraduate School, Monterey, CA, December 2005.

48. Ross, I.M., Gong, Q., and Sekhavat, P., “Low-Thrust, High-Accuracy Trajectory Optimization,” *Journal of Guidance, Control, and Dynamics*, 2006.
49. Strizzi, J., Ross, I.M., and Fahroo, F., “Towards Real-Time Computation of Optimal Controls for Nonlinear Systems,” *Proceedings of the 2002 AIAA Guidance, Navigation, and Control Conference*, AIAA Paper No. 2002-4945, August 2002.
50. General Atomics Aeronautical Systems, “Aircraft Platforms: Altair,” [<http://www.ga-asi.com/products/index.php>], 26 October 2006.
51. Educational Control Products, *Manual for A-51 Inverted Pendulum Accessory*, v. 1.4, Bell Canyon, CA, 2003.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. I. Michael Ross
Naval Postgraduate School
Monterey, California
4. Wei Kang
Naval Postgraduate School
Monterey, California
5. Kevin P. Bollino
Naval Postgraduate School
Monterey, California
6. Guidance, Navigation, and Control Laboratory
Naval Postgraduate School
Monterey, California