

Steering Control of an Autonomous Ground Vehicle with Application to the
DARPA Urban Challenge

by

Stefan F. Campbell

B.S. Mechanical Engineering (2005)

University of Notre Dame

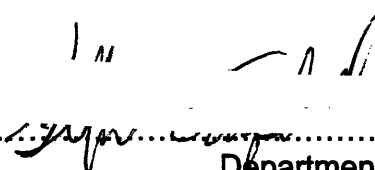
Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree
of Master of Science in Mechanical Engineering

at the

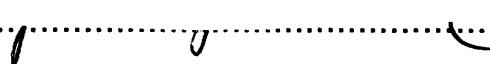
Massachusetts Institute of Technology

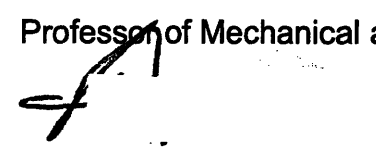
September 2007

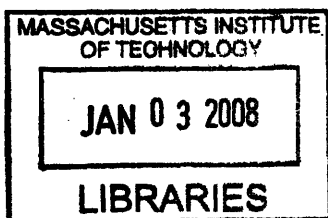
© 2007 Massachusetts Institute of Technology
All rights reserved

Signature of Author.....
Department of Mechanical Engineering
July 11, 2007

Certified by.....
Karl Iagnemma
Principal Research Scientist
Thesis Co-supervisor

Certified by.....
John Leonard
Professor of Mechanical and Ocean Engineering
Thesis Co-supervisor

Accepted by.....
Lallit Anand
Chairman, Committee on Graduate Students
Department of Mechanical Engineering



ARCHIVES

Steering Control of an Autonomous Ground Vehicle with Application to the DARPA Urban Challenge

By

Stefan F. Campbell

Submitted to the Department of Mechanical Engineering
on July 11, 2007 in Partial Fulfillment of the Requirements of
Master of Science in Mechanical Engineering

Abstract

Fundamental to the design of an Ackerman steered autonomous ground vehicle is the development of a low-level controller that effectively performs trajectory or path tracking. Though ample literature is available on various methods for controlling ground vehicles, little information is presented on the implementation and tuning of such controllers. Moreover, few sources extend ground vehicle control to driving in reverse.

This work presents a novel approach to the implementation of the traditional “pure pursuit” style controller in which a dynamic vehicle model is used to map from the path curvature specified by the pure pursuit algorithm to the vehicle’s actual steering angle. Additionally, an analytical methodology using a linear model of straight-line path following is used to tune the pure pursuit look-ahead distance. This pure pursuit controller is then contrasted with a simulation-based controller that uses a kinematic model to predict the vehicle’s response to a series of different steering inputs; a performance metric is used to select the best command given these predictions. Successful trajectory control results are presented at speeds up to 22 mph.

The second focus of this work is the control of a front-wheel steered vehicle driving in reverse. Novel to this work is the presentation of pure pursuit as a stable solution to this problem. Pure pursuit is then contrasted with the mechanism-based controller that was developed by Patwardhan et al. at the University of California Berkeley. In presenting this controller, a new method employing a linear kinematic vehicle model is used to tune the controller parameters. It is then shown that, under specific conditions, the mechanism-based controller and the pure pursuit controller are identical. Both controllers are then compared with the simulation-based controller adapted for driving in reverse. Results are presented at speeds up to 6.7 mph.

Results for the implementation of these controllers were collected using a 2006 Land Rover LR3 developed for MIT’s entry into the 2007 DARPA Urban Challenge. Results ultimately illustrate the respective strengths and weaknesses of the pure pursuit class of controllers.

Thesis Co-supervisor: Karl Iagnemma
Title: Principal Research Scientist

Thesis Co-supervisor: John Leonard
Title: Professor of Mechanical and Ocean Engineering

Acknowledgements

First and foremost I have to acknowledge the support of my advisor Dr. John Leonard. As a member of the Mechanical Engineering Department at MIT, my first year of study was colored by a frantic search for both an advisor and a project that appropriately suited my interests. Just as time was running out, Dr. Leonard offered me funding on an ideal project that turned into the basis for this work. Had Dr. Leonard not been so kind as to fund me at that time, I would not have remained at MIT nor would I have been able to complete my degree program. It is for this reason that I am eternally grateful to Dr. Leonard.

Thanks must also be given to Dr. Karl Iagnemma. Dr. Iagnemma, acting as my co-advisor with Dr. Leonard, offered me superb guidance and a great insight into the field of ground robotics. Though it was not easy for Dr. Iagnemma to make time for me in his schedule, he consistently made talking with me a priority. Moreover, Dr. Iagnemma was always encouraging and his positive attitude helped mitigate the stresses of a very time constrained project.

I would also like to thank Dr. Rohan Abeyaratne who, as the head of the Mechanical Engineering Department at MIT, showed great patience in helping fund me while I searched for an acceptable project.

I also owe a great debt of gratitude to Dr. Yoshi Kuwata, Gaston Fiore, and Justin Teo for helping me with much of the technical content that is presented in this thesis.

I must also thank my family, but especially my sister Megan Campbell. Without the persistent encouragement of my sister, I would not have continued to search frantically for funding and would have given up long before the submission of this work. This is not to mention the fact that Megan set the standard for academic excellence that I merely followed as a child. Had I not had such a great role model to emulate, I do not honestly believe I would have ended up at an institute as prestigious as MIT. I must also thank Jahan Minoo for his persistent and often colorful encouragement. Jahan's cunning wit never failed to make me laugh and definitely helped add levity to an unpleasant situation. Thanks must of course go to my parents Florcia Hamilton, Michael Hamilton, and William Campbell. As my parents, I will always be appreciative of the sacrifices they made on my behalf.

Finally, thanks must be given to my friends Kevin Boyle, Arman Haidari, Kiel Martin, Brian Masse, John Miller, Katie Miller, Jaclyn Rhodes, Brian Skinn, and Wyatt Tenhaeff for making weekends at MIT enjoyable.

Contents

| | |
|--|-----------|
| ABSTRACT | 3 |
| ACKNOWLEDGEMENTS | 4 |
| CONTENTS | 5 |
| LIST OF FIGURES | 7 |
| LIST OF TABLES | 11 |
| NOMENCLATURE | 12 |
| NOMENCLATURE | 12 |
| 1 INTRODUCTION | 16 |
| 1.1 BACKGROUND OF EXISTING WORK AND ASSOCIATED LIMITATIONS | 16 |
| 1.2 MIT AND THE DARPA URBAN CHALLENGE | 25 |
| 2 FINDING CONTROL FEEDBACK PARAMETERS | 31 |
| 2.1 REPRESENTING THE PATH | 31 |
| 2.2 SEARCH FOR POSITION | 33 |
| 2.3 PRE-SEARCH | 46 |
| 2.4 SEARCH FOR GOAL POINT | 50 |
| 2.5 ALTERNATIVE CONTROLLERS | 58 |
| 2.6 CHAPTER CONCLUSIONS | 59 |
| 3 SYSTEM MODELING | 60 |
| 3.1 ACKERMAN STEERING | 60 |
| 3.2 SINGLE-TRACK KINEMATIC MODEL | 61 |
| 3.3 STEERING MAP | 64 |
| 3.4 DYNAMIC MODELS | 65 |
| 3.4.1 LINEAR REVERSE DRIVING MODEL | 66 |
| 3.4.2 LINEAR FORWARD DRIVING MODEL | 70 |
| 3.5 IDENTIFICATION OF UNDERSTEER COEFFICIENT | 73 |
| 3.6 ACKERMAN STEERING VS. DYNAMIC MODEL | 79 |
| 3.7 STEERING SYSTEM MODEL | 81 |
| 3.8 CHAPTER CONCLUSIONS | 84 |
| 4 THE PURE PURSUIT CONTROLLER | 85 |
| 4.1 BASIC PURE PURSUIT | 85 |
| 4.2 STEERING ANGLE | 87 |
| 4.3 LINEARIZED STRAIGHT LINE FOLLOWING | 88 |
| 4.4 NONLINEAR MODIFICATIONS TO PURE PURSUIT ALGORITHM | 100 |

| | | |
|----------|---|------------|
| 4.5 | SIMULATION-BASED APPROACH TO STEERING ANGLE SELECTION | 102 |
| 4.6 | CHAPTER CONCLUSIONS | 110 |
| 5 | VEHICLE CONTROL IN REVERSE..... | 112 |
| 5.1 | SIMPLIFYING ASSUMPTION FOR DRIVING IN REVERSE | 112 |
| 5.2 | PURE PURSUIT IN REVERSE | 113 |
| 5.3 | SIMULATION-BASED APPROACH TO STEERING ANGLE SELECTION | 117 |
| 5.4 | MECHANISM-BASED APPROACH..... | 118 |
| 5.5 | CHAPTER CONCLUSIONS | 126 |
| 6 | RESULTS | 128 |
| 6.1 | PURE PURSUIT FOR FORWARD DRIVING | 130 |
| 6.2 | SIMULATION-BASED CONTROL FOR FORWARD DRIVING..... | 138 |
| 6.3 | PURE PURSUIT FOR DRIVING IN REVERSE | 146 |
| 6.4 | MECHANISM-BASED CONTROL FOR DRIVING IN REVERSE | 148 |
| 6.5 | SIMULATION-BASED CONTROL FOR DRIVING IN REVERSE | 150 |
| 6.6 | CONCLUSIONS | 154 |
| | REFERENCES | 156 |
| | APPENDIX A | 158 |
| A.1 | VEHICLE ROLLOVER | 158 |
| A.2 | LATERAL SLIDING | 160 |
| A.3 | SPEED REGULATION..... | 163 |
| | APPENDIX B..... | 166 |
| B.1 | MECHANISM-BASED CONTROLLER FOR FORWARD DRIVING..... | 167 |
| B.2 | PARAMETER TUNING..... | 169 |
| | APPENDIX C | 173 |
| C.1 | VEHICLES..... | 173 |
| C.2 | EMC CONTROL SYSTEM FOR STEERING..... | 178 |
| | APPENDIX D | 181 |
| D.1 | EMC ACTUATOR..... | 182 |
| D.2 | CONTROL LOOP AND SPEED SELECTION..... | 183 |
| D.3 | VOLTAGE MAP | 186 |
| D.4 | SPEED CONTROL RESULTS | 191 |

List of Figures

| | |
|---|----|
| FIGURE 1.1: REPRESENTATIVE OPERATING DOMAIN FOR A GROUND VEHICLE..... | 17 |
| FIGURE 1.2: EXAMPLE OF GAMMA BOUNDARY..... | 18 |
| FIGURE 1.3: STABILIZING GAINS AND GAINS WHICH STABILIZE EXTREMAL POINTS ONLY | 19 |
| FIGURE 1.4: SLIDING MANIFOLD..... | 20 |
| FIGURE 1.5: STANFORD CONTROLLER PARAMETERS..... | 22 |
| FIGURE 1.6: LOOK-AHEAD DISTANCE | 24 |
| FIGURE 1.7: HIGH-LEVEL VIEW OF SYSTEM CONTROL | 28 |
| FIGURE 2.1: PIECEWISE LINEAR PATH REPRESENTATION..... | 32 |
| FIGURE 2.2: CROSS-TRACK ERROR..... | 34 |
| FIGURE 2.3: VEHICLE RELATIVE POSITION PARAMETERS | 36 |
| FIGURE 2.4: RELEVANCY REGIONS FOR A SIMPLE LINEAR PATH | 37 |
| FIGURE 2.5: RELEVANCY ZONES FOR NON-COLLINEAR SEGMENTS | 38 |
| FIGURE 2.6: VEHICLE TRAVELING ON THE INSIDE OF A TURN | 39 |
| FIGURE 2.7: DEAD ZONE CASE..... | 41 |
| FIGURE 2.8: FIRST NODE IN FRONT OF VEHICLE | 42 |
| FIGURE 2.9: FLOW DIAGRAM FOR FINDING RELEVANCE..... | 44 |
| FIGURE 2.10: UPDATE TO A PATH WITH PARALLEL LINE SEGMENTS | 47 |
| FIGURE 2.11: NODE POSITION RELATIVE TO VEHICLE | 48 |
| FIGURE 2.12: GOAL POINT FOR PURE PURSUIT..... | 50 |
| FIGURE 2.13: GOAL POINT DETERMINATION FOR LARGE PATH SEGMENTS | 53 |
| FIGURE 2.14: GOAL POINT DEAD ZONE | 55 |
| FIGURE 2.15: FLOW DIAGRAM FOR GOAL POINT SEARCH..... | 56 |
| FIGURE 3.1: ACKERMAN STEERING GEOMETRY | 61 |
| FIGURE 3.2: KINEMATIC SINGLE-TRACK MODEL | 62 |
| FIGURE 3.3: REVERSE FREE BODY DIAGRAM..... | 67 |
| FIGURE 3.4: FREE-BODY DIAGRAM FOR FORWARD DRIVING..... | 71 |
| FIGURE 3.5: MOMENT OF INERTIA..... | 74 |
| FIGURE 3.6: DETERMINATION OF K_{us} FOR TALOS-I | 75 |
| FIGURE 3.7: TALOS- I VELOCITY PROFILE FOR UNDERSTEER EXPERIMENT | 76 |
| FIGURE 3.8: TALOS-I POSITION FOR UNDERSTEER EXPERIMENT | 76 |
| FIGURE 3.9: DETERMINATION OF K_{us} FOR TALOS-II..... | 77 |
| FIGURE 3.10: TALOS-II VELOCITY PROFILE FOR UNDERSTEER EXPERIMENT | 77 |
| FIGURE 3.11: TALOS-II POSITION FOR UNDERSTEER EXPERIMENT..... | 78 |
| FIGURE 3.12: COMPARISON BETWEEN ACKERMAN AND DYNAMICS STEERING MODEL FOR TALOS-I..... | 79 |

| | |
|--|-----|
| FIGURE 3.13: COMPARISON BETWEEN ACKERMAN AND DYNAMIC STEERING MODEL FOR TALOS-II | 80 |
| FIGURE 3.14: SMALL STEP RESPONSE OF STEERING WHEEL ACTUATOR | 83 |
| FIGURE 3.15: LARGE STEP RESPONSE OF STEERING WHEEL ACTUATOR..... | 83 |
| FIGURE 4.1: PURE PURSUIT GOAL POINT | 87 |
| FIGURE 4.2: PURE PURSUIT STRAIGHT LINE FOLLOWING..... | 89 |
| FIGURE 4.3: PURE PURSUIT ANGLE DECOMPOSITION..... | 90 |
| FIGURE 4.4: ACKERMAN PURE PURSUIT AND DYNAMIC PURE PURSUIT..... | 92 |
| FIGURE 4.5: SYSTEM BLOCK DIAGRAM..... | 93 |
| FIGURE 4.6: SIMPLIFIED SYSTEM BLOCK DIAGRAM..... | 95 |
| FIGURE 4.7: LOOK-AHEAD DISTANCE AS A FUNCTION OF SPEED | 96 |
| FIGURE 4.8: LINEARIZED TALOS-I MODEL WITH DYNAMIC PURE PURSUIT GAINS | 98 |
| FIGURE 4.9: LINEARIZED TALOS-II MODEL WITH DYNAMIC PURE PURSUIT GAINS | 98 |
| FIGURE 4.10: LINEARIZED TALOS-I MODEL WITH ACKERMAN BASED PURE PURSUIT GAINS..... | 99 |
| FIGURE 4.11: LINEARIZED TALOS-II MODEL WITH ACKERMAN BASED PURE PURSUIT GAINS | 100 |
| FIGURE 4.12: USING VEHICLE'S POSITION ON PATH TO DEFINE PURE PURSUIT GOAL POINT..... | 101 |
| FIGURE 4.13: CLOTHOIDAL CURVE | 103 |
| FIGURE 4.14: PURE PURSUIT FAILURE MODE..... | 104 |
| FIGURE 4.15: SIMULATION-BASED APPROACH TO STEERING ANGLE SELECTION..... | 107 |
| FIGURE 4.16: SIMULATION-BASED APPROACH TO STEERING ANGLE SELECTION..... | 108 |
| FIGURE 4.17: EVALUATING SEARCHABLE SPACE FOR PRESENCE OF GOAL POINT | 110 |
| FIGURE 5.1: CROSS-TRACK ERROR FOR VEHICLE MOVING IN REVERSE | 114 |
| FIGURE 5.2: TALOS-I POLE-ZERO PLOT FOR PURE PURSUIT IN REVERSE | 116 |
| FIGURE 5.3: TALOS-II POLE-ZERO PLOT FOR PURE PURSUIT IN REVERSE..... | 117 |
| FIGURE 5.4: MECHANISM-BASED CONTROLLER | 119 |
| FIGURE 5.5: TALOS-I POLE-ZERO PLOT FOR MECHANISM-BASED CONTROLLER..... | 124 |
| FIGURE 5.6: TALOS-II POLE-ZERO PLOT FOR MECHANISM-BASED CONTROLLER | 124 |
| FIGURE 6.1: MIT TEST-SITE COURSE..... | 129 |
| FIGURE 6.2: FIGURE-EIGHT CONTROLLER EVALUATION COURSE | 129 |
| FIGURE 6.3: TALOS-II TRACKING A COARSE RECTANGULAR PATH | 130 |
| FIGURE 6.4: DESIRED STEERING ANGLE FOR TRAVERSING RECTANGULAR PATH..... | 131 |
| FIGURE 6.5: LOOK-AHEAD DISTANCE FOR TRAVERSING RECTANGULAR PATH | 132 |
| FIGURE 6.6: CROSS-TRACK ERROR FOR TRAVERSING A RECTANGULAR PATH | 133 |
| FIGURE 6.7: TALOS-II TRAVERSING FIGURE EIGHT COURSE WITH PURE PURSUIT CONTROLLER..... | 134 |
| FIGURE 6.8: CROSS-TRACK ERROR FOR TRAVERSING FIGURE EIGHT COURSE..... | 134 |
| FIGURE 6.9: TALOS-II TRAVERSING FIGURE EIGHT COURSE AT 6 M/S..... | 136 |
| FIGURE 6.10: CROSS-TRACK ERROR FOR TRAVERSING FIGURE EIGHT COURSE AT 6 M/S | 136 |
| FIGURE 6.11: 10 M STEP IN DESIRED PATH..... | 137 |

| | |
|--|-----|
| FIGURE 6.12: CROSS-TRACK ERROR FOR 10 M STEP INPUT | 138 |
| FIGURE 6.13: TALOS-II TRAVERSING FIGURE EIGHT COURSE WITH SBC | 139 |
| FIGURE 6.14: CROSS-TRACK ERROR FOR FIGURE EIGHT COURSE WITH SBC..... | 140 |
| FIGURE 6.15: DESIRED STEER ANGLE FOR FIGURE EIGHT COURSE WITH MPC..... | 140 |
| FIGURE 6.16: TALOS-II TRAVERSING FIGURE EIGHT COURSE AT 6 M/S WITH SBC | 141 |
| FIGURE 6.17: CROSS-TRACK ERROR FOR FIGURE EIGHT COURSE AT 6 M/S WITH SBC | 142 |
| FIGURE 6.18: TALOS-II TRAVERSING RECTANGULAR PATH WITH SBC | 143 |
| FIGURE 6.19: CROSS-TRACK ERROR FOR SBC TRACKING RECTANGULAR PATH..... | 143 |
| FIGURE 6.20: SBC VS. PURE PURSUIT CONTROLLER..... | 145 |
| FIGURE 6.21: CROSS-TRACK ERROR FOR COMPARING SBC AND PURE PURSUIT CONTROLLER..... | 145 |
| FIGURE 6.22: DESIRED STEER ANGLE FOR SBC VS. PURE PURSUIT CONTROLLER..... | 146 |
| FIGURE 6.23: PURE PURSUIT CONTROLLER FOR REVERSE AT 3 M/S | 147 |
| FIGURE 6.24: CROSS-TRACK ERROR FOR PURE PURSUIT CONTROLLER IN REVERSE..... | 147 |
| FIGURE 6.25: DESIRED STEER ANGLE FOR PURE PURSUIT CONTROLLER IN REVERSE..... | 148 |
| FIGURE 6.26: MECHANISM-BASED CONTROLLER FOR REVERSE AT 3 M/S | 149 |
| FIGURE 6.27: CROSS-TRACK ERROR FOR MECHANISM-BASED CONTROLLER | 149 |
| FIGURE 6.28: DESIRED STEER ANGLE FOR MECHANISM-BASED CONTROLLER | 150 |
| FIGURE 6.29: SBC FOR REVERSE AT 3 M/S | 151 |
| FIGURE 6.30: CROSS-TRACK ERROR FOR SBC IN REVERSE | 151 |
| FIGURE 6.31: DESIRED STEER ANGLE FOR SBC IN REVERSE | 152 |
| FIGURE 6.32: MPC VS. PURE PURSUIT STYLE CONTROLLER..... | 153 |
| FIGURE 6.33: CROSS-TRACK ERROR FOR MPC VS. PURE PURSUIT STYLE CONTROLLER | 153 |
| FIGURE 6.34: DESIRED STEER ANGLE FOR MPC VS. PURE PURSUIT STYLE CONTROLLER | 154 |
| FIGURE A.1: VEHICLE FREE-BODY DIAGRAM (FROM REAR)..... | 159 |
| FIGURE A.2: TYPICAL STEERING ANGLE CONSTRAINTS..... | 162 |
| FIGURE A.3: LOOK-AHEAD DISTANCE TUNING USING VEHICLE PERFORMANCE CONSTRAINTS | 163 |
| FIGURE A.4: EMERGENCY SPEED REGULATOR..... | 165 |
| FIGURE B.1: MECHANISM CONTROLLER FOR FORWARD DRIVING | 167 |
| FIGURE B.2: VELOCITY DEPENDENCE OF PARAMETERS A AND B | 171 |
| FIGURE B.3: POLE-ZERO MAP FOR MECHANISM-BASED FORWARD CONTROLLER..... | 172 |
| FIGURE C.1: TALOS-I (2006 FORD ESCAPE) AT SITE VISIT COURSE..... | 173 |
| FIGURE C.2: TALOS-II | 175 |
| FIGURE C.3: EMC INTERFACE | 176 |
| FIGURE D.1: EMC GAS/BRAKE ACTUATOR | 182 |
| FIGURE D.2: ANTI-WINDUP WITH PID CONTROL LAW | 186 |
| FIGURE D.3: GENERAL U_{ACTION} TO VOLTAGE MAP | 188 |
| FIGURE D.4: MODIFIED VOLTAGE MAP | 190 |

FIGURE D.5: STEP INPUT OF 6 *M/S* 192
FIGURE D.6: STEP INPUT OF 13 *M/S* 192

List of Tables

| | |
|---|-----|
| TABLE 2.1: CASE AND CONDITIONS FOR RELEVANCY | 43 |
| TABLE 2.2: CASES AND CONDITIONS FOR GOAL POINT SEARCH | 57 |
| TABLE 3.1: VEHICLE MODEL SUMMARY | 73 |
| TABLE C.1: RELEVANT VEHICLE PARAMETERS | 177 |
| TABLE C.2: EMC DIRECT CURRENT VOLTAGE SIGNALS | 179 |
| TABLE D.1: VOLATAGE MAP PARAMETERS FOR TEAM MIT | 191 |

Nomenclature

| | |
|-------------|---|
| a |mechanism-based controller tunable parameter |
| a_{cmd} | commanded acceleration |
| a_x | .. acceleration of the vehicle's center of mass in the x -direction of a body fixed frame |
| a_y | .. acceleration of the vehicle's center of mass in the y -direction of a body fixed frame |
| b |mechanism-based controller tunable parameter |
| C | cornering stiffness of tires |
| D_{ap} | distance between (i_x, i_y) and (L_x, L_y) |
| D_c |comfortable stopping distance |
| d_{extra} | tuning parameter for designing comfortable stopping distance |
| F_{xf} |tire force in the x direction of the front tire in a dynamic, single-track model |
| F_{xr} |tire force in the x direction of the rear tire in a dynamic, single-track model |
| F_{yf} |tire force in the y direction of the front tire in a dynamic, single-track model |
| F_{yr} |tire force in the y direction of the rear tire in a dynamic, single-track model |
| G_x | x -component of pure pursuit goal point |
| G_y | y -component of pure pursuit goal point |
| i | node or path number |
| i_r |index for the relevant path segment |
| i_x | x -component of node i location |
| i_y | y -component of node i location |
| I_z | vehicle's moment of inertia about the z -axis of a body fixed frame |
| K_{csd} |comfortable stopping distance gain |
| K_{ct} | L_a cross-track gain |
| K_d | derivative term gain |
| K_i |integral term gain |
| K_{La} | L_a speed gain |
| K_p |proportional term gain |
| K_{steer} | ratio from angular displacement from the steering wheel to δ_i |
| K_{us} | understeer coefficient |

| | | |
|--------------|-------|---|
| K_{wup} | | anti-windup term gain |
| $L_{a\ min}$ | | minimum L_a distance |
| L_a | | look-ahead distance |
| L_{ax} | | x -component of look-ahead location |
| L_{ay} | | y -component of look-ahead location |
| L_b | | vehicle's length from the center of the front wheels to the center of the rear wheels |
| l_f | | distance from front wheels to vehicle's center of mass |
| L_p | | length of path segment |
| l_r | | distance from rear wheels to vehicle's center of mass |
| L_{tw} | | vehicle's track-width from the right wheels to the left wheels |
| L_x | | x -component of vehicle's projected position on path |
| L_y | | y -component of vehicle's projected position on path |
| m_b | | slope of voltage map when actuating brake pedal |
| m_g | | slope of voltage map when actuating gas pedal |
| Q_1 | | vector from vehicle position to node i of path segment i |
| Q_2 | | vector from vehicle position to node $i+1$ of path segment i |
| q_i | | extremal plants for robust control |
| R | | radius of curvature |
| s | | speed |
| S_{center} | | steering wheel actuator voltage that causes the vehicle to drive straight |
| S_{change} | | changeover speed for speed scheduling L_a |
| S_{max} | | steering wheel actuator voltage that turns the steering wheel full to the right |
| S_{min} | | steering wheel actuator voltage that turns the steering wheel full to the left |
| T_{tl} | | 360° turns of the steering wheel from full left to full right |
| T_{min} | | minimum steady-state cruising time |
| u | | velocity of front axle center for Stanford Controller |
| U_{action} | | pid controller output for gas/brake actuation |
| U_{coast} | | voltage map parameter used to design coasting regime |
| u_{max} | | vector from goal point to end of vehicle's maximum trajectory |
| U_{max} | | maximum value of U_{action} |
| u_{min} | | vector from goal point to end of vehicle's minimum trajectory |

| | | |
|---------------|-------|--|
| U_{min} | | minimum value of U_{action} |
| U_o | | cross-over of U_{action} from actuating throttle to actuating brake pedal |
| V | | vehicle velocity from center of rear axle |
| V_{brake} | | brake pedal actuator voltage that just begins to decelerate the vehicle |
| v_{cmd} | | commanded velocity |
| V_{decel} | | brake actuator voltage used to design m_b |
| V_{gas} | | gas pedal actuator voltage that just begins to accelerate the vehicle |
| V_{max} | | maximum allowable input voltage to the gas pedal actuator |
| V_{rest} | | brake actuator voltage that keeps the vehicle at rest from a stop |
| v_x | | velocity of the vehicle's center of mass in the x direction of a body fixed frame |
| v_y | | velocity of the vehicle's center of mass in the y direction of a body fixed frame |
| x_b | | x axis of vehicle's body fixed frame |
| x_g | | x axis of goal point fixed coordinate system |
| x_p | | x -component of vehicle's location |
| y_b | | y axis of vehicle's body fixed frame |
| y_d | | desired y position of the vehicle's center of mass |
| y_g | | y axis of goal point fixed coordinate system |
| y_p | | y -component of vehicle's location |
| α | | time lag associated with actuating brake pedal |
| α_1 | | angle between Q_1 and path segment i |
| α_2 | | angle between Q_2 and path segment i |
| α_f | | slip angle for front tire in a single-track model |
| α_r | | slip angle for rear tire in a single-track model |
| β | | bisecting angle between segments i and $i+1$; side-slip angle in vehicle dynamics |
| γ | | control angle for mechanism-based controller |
| δ | | an average of the angular position of the vehicle's front two wheels |
| δ_d | | desired steering angle outputted from pure pursuit controller |
| δ_i | | angular position of the vehicle's inside wheel when entering a turn |
| δ_o | | angular position of the vehicle's outside wheel when entering a turn |
| δ_w | | angular position of vehicle's steering wheel |
| ε | | cross-track error |

ζ a system's damping
 ζ_d the desired damping for a system
 η pure pursuit angle
 θ angle between segment i and segment $i+1$
 κ curvature
 Ψ vehicle's absolute yaw
 ω_d the desired natural frequency for a system
 ω_{max} maximum slew rate of steering wheel actuator
 ω_n a system's natural frequency

1 Introduction

The field of automatic vehicle control is well established and the available literature on the topic is vast. To begin this thesis, several of the most common approaches to automatic vehicle control will be presented. Here the intention is to show the limitations and challenges associated with these controllers in a context that motivates the selection of an alternative approach. This discussion will be followed by a description of the Defense Advanced Research Project Agency's (DARPA) Urban Challenge, which funded this work, and the associated engineering challenges. Ultimately, the goal of this research was not to revolutionize the vehicle control field, but to quickly establish a functional control system for MIT's first serious entry into the DARPA competition.

1.1 Background of Existing Work and Associated Limitations

The field of automatic steering control is fairly well established. Design solutions extend from simple lead-lag controllers [31] to advanced sliding mode control concepts [11] to the use of screw theory [32]. An exceedingly prevalent approach to automatic steering control is the use of robust control theory, as in [1] and [2]. In this method, a linear vehicle model is identified as having unknown parameters with a range of obtainable values. For steering control, these parameters are usually selected as vehicle mass and velocity. This defines an operating domain Q with extremal values $q_1 - q_4$, as illustrated in Figure 1.1.

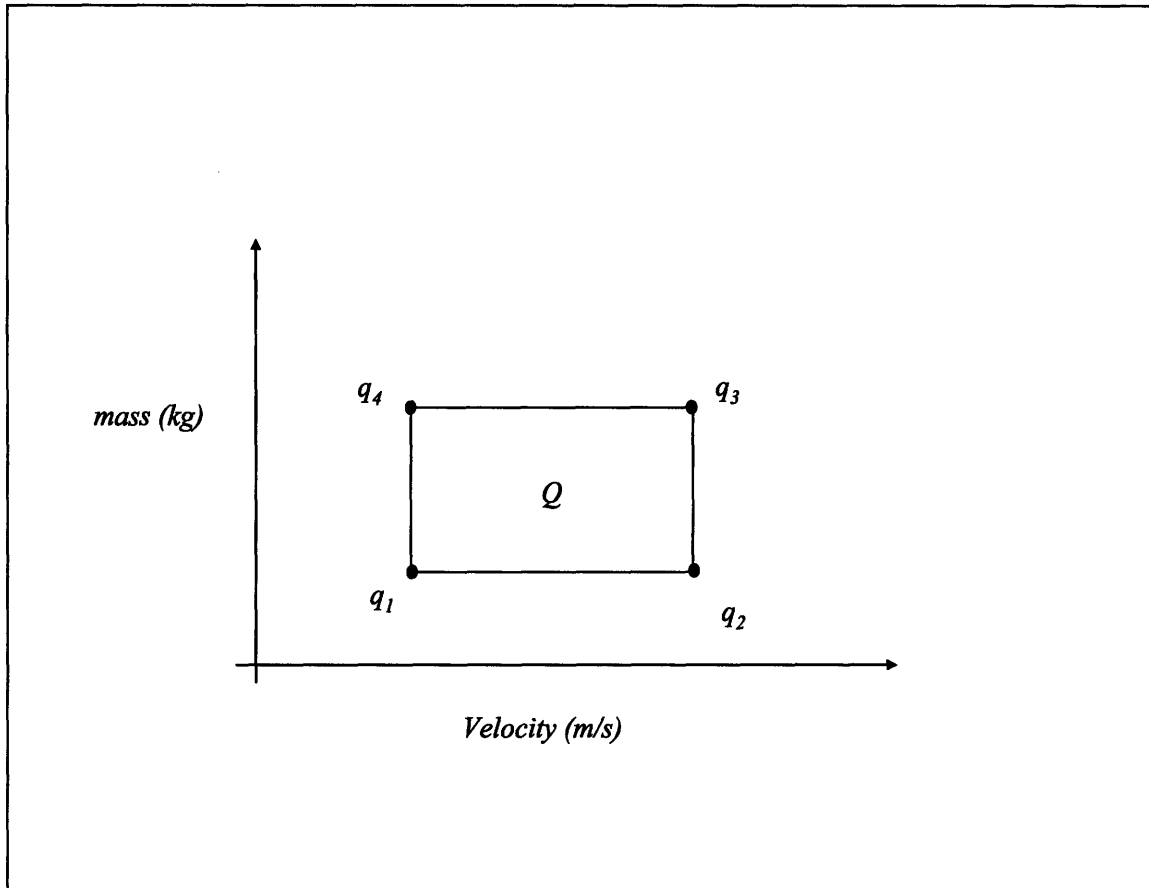


Figure 1.1: Representative Operating Domain for a Ground Vehicle

One design approach is to then define a linear controller and an acceptable region for the closed loop pole locations in the s plane, often denoted as the $\partial\Gamma$ boundary (illustrated in Figure 1.2). Selecting one of the extremal plants q_i , the values of the control gains k which place the poles of the system on the boundary $\partial\Gamma$ are mapped. This generates a curve which divides the space of control gains (k space) into regions which may or may not Γ -stabilize the system. Here a set of gains is Γ stabilizing for the system q_i if they effectively place the system poles inside the $\partial\Gamma$ boundary. By testing a single point from each region of the k space, the stabilizing region is found. This process is repeated for all extremal plants to produce overlaying Γ -stabilizing regions in the k space. From this, a set of gains which stabilize the system for all the extremal plants is selected, provided such a set of gains exist (this is not guaranteed).

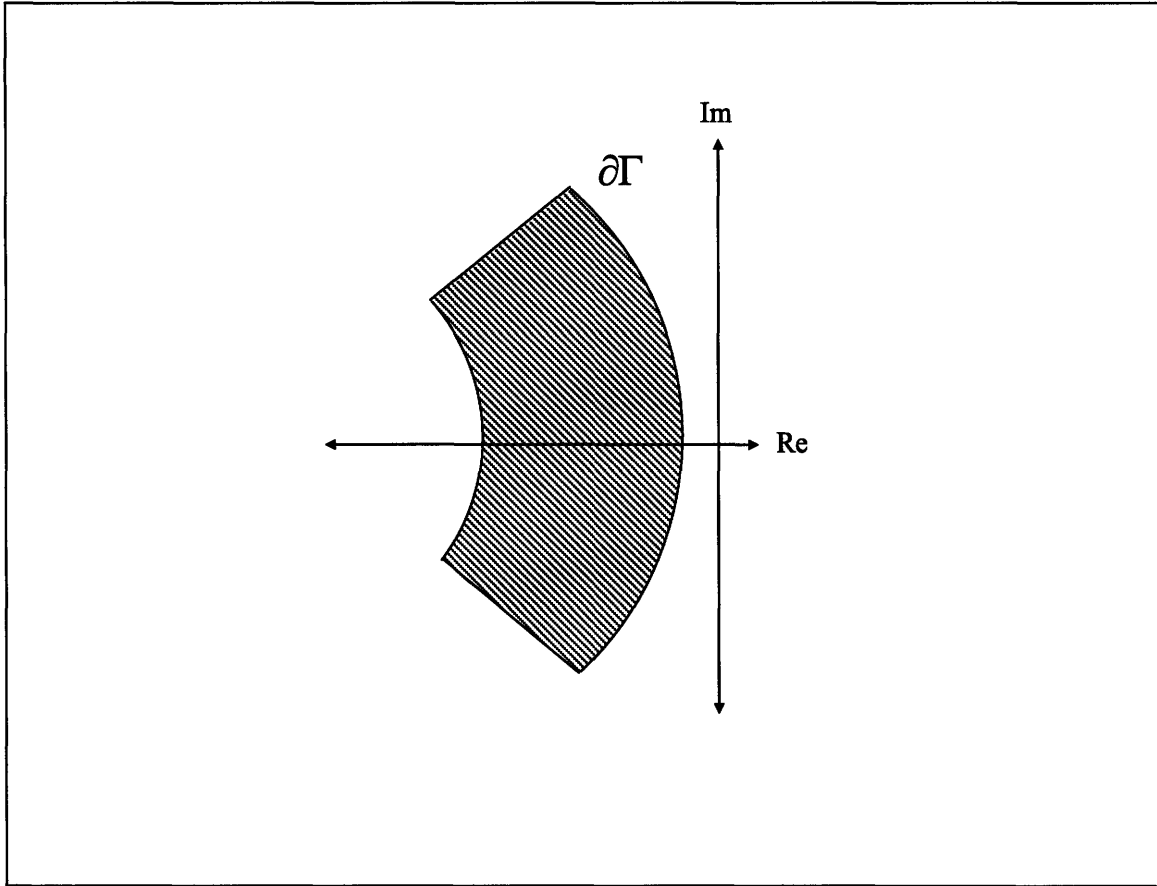


Figure 1.2: Example of Gamma Boundary

The final step is to verify that these gains stabilize the entire operating range, not just the extremal plants. To verify this, the values of the unknown parameters that place the system poles on the $\partial\Gamma$ boundary (using the selected gains in the linear controller) are mapped. If the set of gains Γ - stabilizes the entire operating range, this map will yield a curve that completely encircles the operating domain \mathcal{Q} . Such a case is illustrated by the solid line in Figure 1.3. Conversely, the dashed line in Figure 1.3 illustrates a case in which the set of gains selected only stabilize the extremal plants and not the entire operating range.

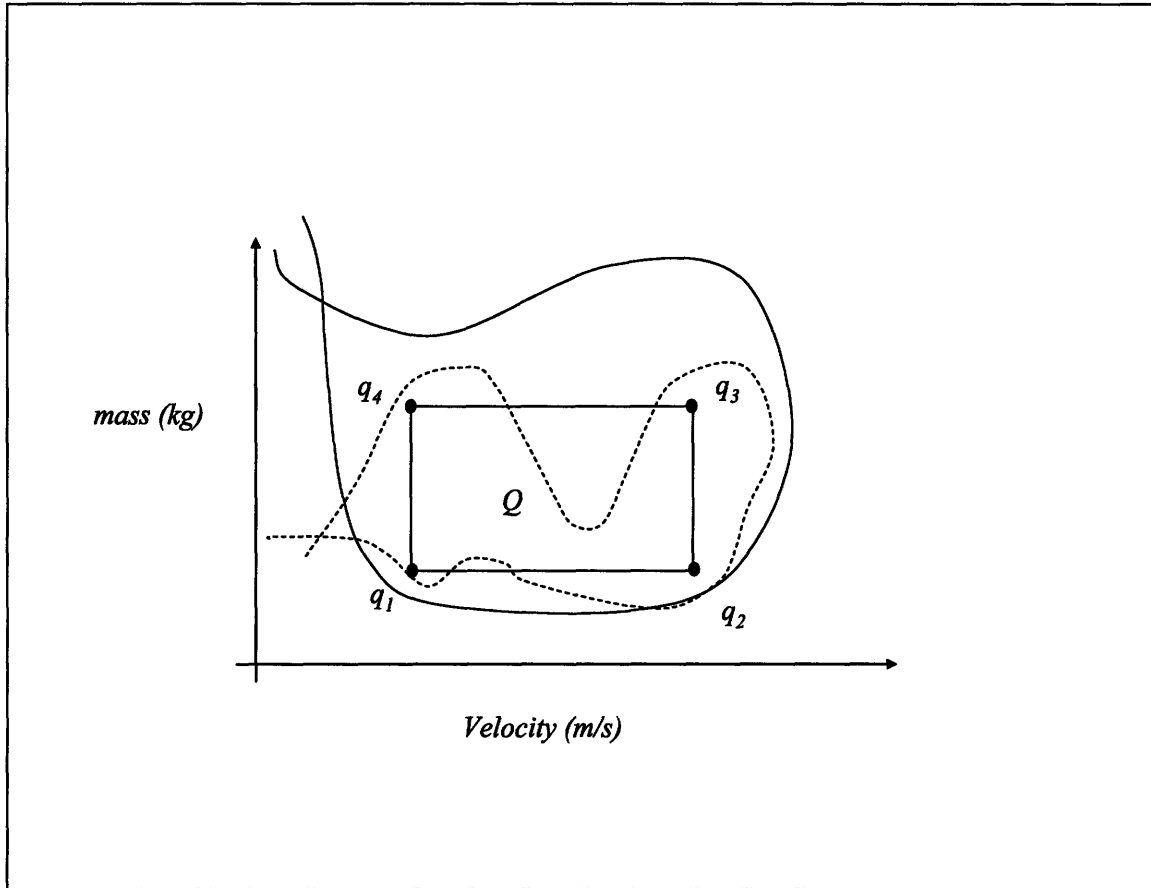


Figure 1.3: Stabilizing Gains and Gains which Stabilize Extremal Points Only

The thorough discussion provided up to this point has been intended to illustrate the complexity associated with robust control systems. Indeed, even greater complications arise once more than two poles of a system are being Γ -stabilized and when more than two gains are involved. In general, when this is the case, logical assumptions have to be made in order to reduce the number of unknown gains as well as the number of poles (or in some cases zeros) that need to be moved to two. Moreover, the above description greatly simplifies robust control theory as the real complication lies in finding the k space map that places the system poles on $\partial\Gamma$. Ultimately the rigor involved in this approach and other robust control approaches, such as the H_∞ controller [9], is a key motivator for exploring the pure pursuit algorithm.

The literature also often contains references to sliding mode control [25]. In this approach, a sliding manifold $s(x;t) = 0$ is defined over the phase plane and a discontinuous control law is selected to drive the system states (x) onto this surface. Referencing Figure 1.4, if the system is above the manifold, the control law $u+$ is used to drive the system towards $s(x;t)$. Conversely, if the system state is below the manifold, the control law $u-$ is used to drive the system towards the surface $s(x;t)$. A challenge of sliding mode control is that it can exhibit a chattering about the manifold with the control switching quickly between $u-$ and $u+$. In application, using this approach requires explicitly handling this chattering affect, a non-trivial design challenge.

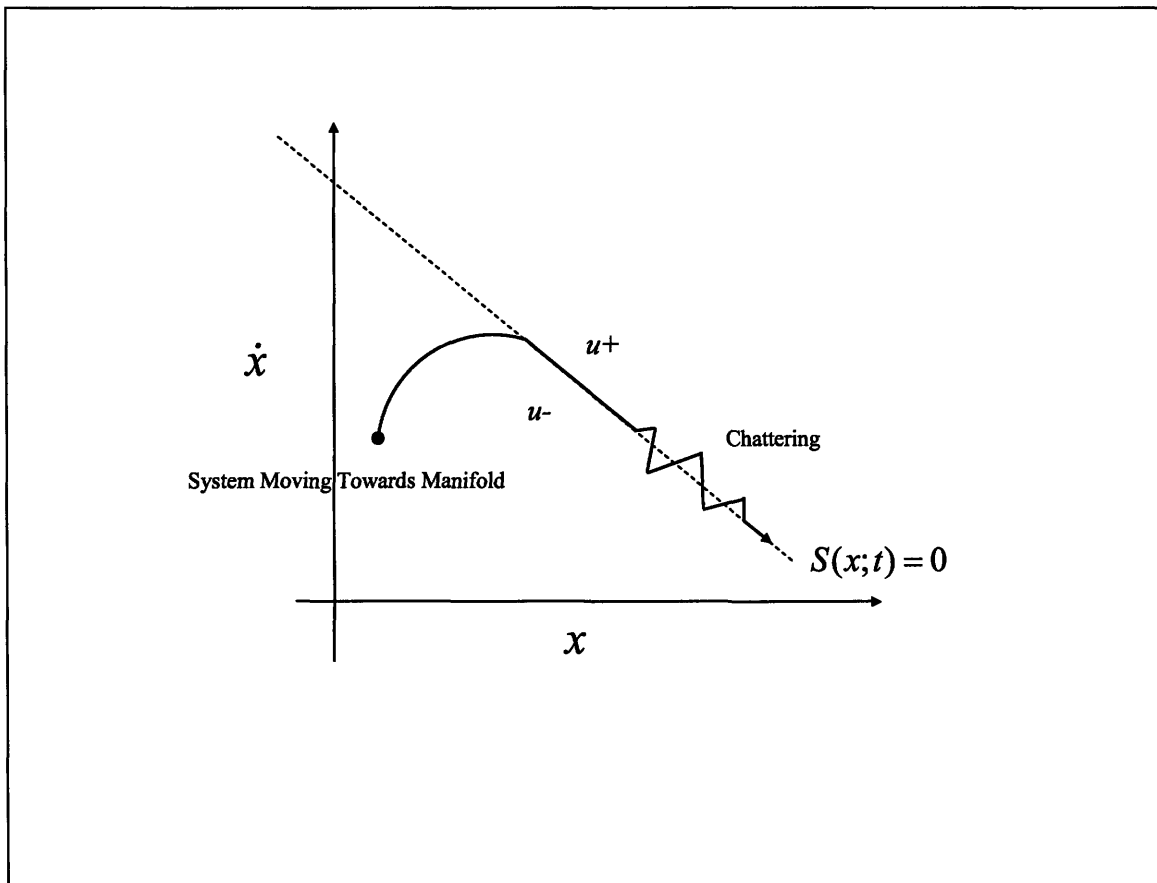


Figure 1.4: Sliding Manifold

A more classical approach was adopted by the SciAutonics-Auburn Engineering team in the previous DARPA Grand Challenge [30]. Here, the transfer function from

vehicle yaw rate ($\dot{\psi}$) to steering input (δ) was assumed second order with unknown coefficients, as in equation 1.1.

$$\frac{\dot{\psi}(s)}{\delta(s)} = \frac{k(s+n)}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad (1.1)$$

The unknown coefficients were identified by experimentally sending a chirp signal to the steering input and curve fitting the system response. This was done at a wide range of speeds as the unknown coefficients were found to be speed dependent. The selected control system was then a proportional-derivative controller (PD) on heading error with the controller gains speed scheduled to maintain fixed closed loop poles. This approach ultimately requires a large amount of system identification, which can present a challenge when limited space and time are available for such work.

The Stanford team that won the last DARPA Grand Challenge used the control law presented below in equation 1.2 [29]. The relevant parameters are illustrated in Figure 1.5.

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{k\varepsilon(t)}{u(t)}\right) \quad (1.2)$$

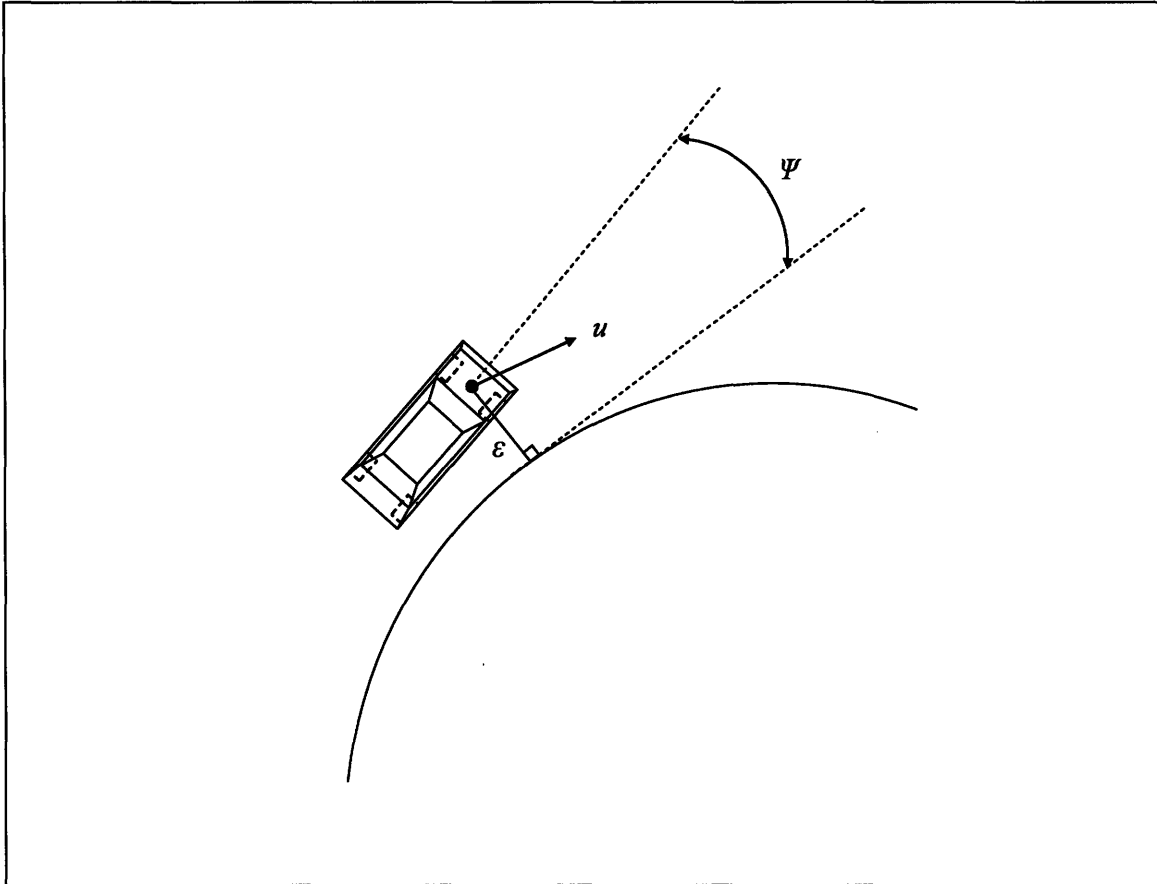


Figure 1.5: Stanford Controller Parameters

Here the underlying assumption is that the steering input should be parallel to the tangent of the path when there is zero cross-track error (ϵ). When cross-track error is non-zero, the desired steer angle is supplemented by a term that is the inverse tangent of the ratio of cross-track error to the velocity of the vehicle's front axle (u). Intuitively, for a given speed, the larger the cross-track error the larger this term of the control law becomes. Moreover, for a given cross-track error, this term will shrink with increasing speed. This results in a rather intuitive behavior. Moreover, it can be shown that for a small cross-track error the error will converge to zero exponentially.

Many automatic steering controllers, especially those involving linear control, measure cross-track error from a point some distance ahead of the vehicle, as illustrated in Figure 1.6. This distance is commonly termed the look-ahead (L_a) distance. In general,

the use of a look-ahead distance is thought to yield results that more closely match human driving [21] by replicating a driver's ability to preview the road ahead. Often, in order to ensure stability, the magnitude of this look-ahead distance is dynamically adjusted with speed, as in [7] and [14]. It should be understood that a real sensor is not projected in front of the vehicle to make this measurement, but rather an estimate of the vehicle's yaw and knowledge of the desired path are used to geometrically calculate the cross-track error at the look-ahead point. The converse approach is to use a *look-down reference system* in which measurements are taken from positions on the vehicle. Experiments in which a single front bumper (or axle) sensor is used are generally only conducted at low speeds. However, this point should not be overstated as systems utilizing measurements of error at multiple locations, such as at the front and rear bumper [12], have proven stable at highway speeds. As will be presented in Chapter 2 and Chapter 4, pure pursuit control schemes use a slightly different definition of a look-ahead distance. Specifically, the look-ahead distance is used to define the radius of a circle that encloses the vehicle. The intersection of this circle with the path then defines a goal point that the vehicle is directed towards.

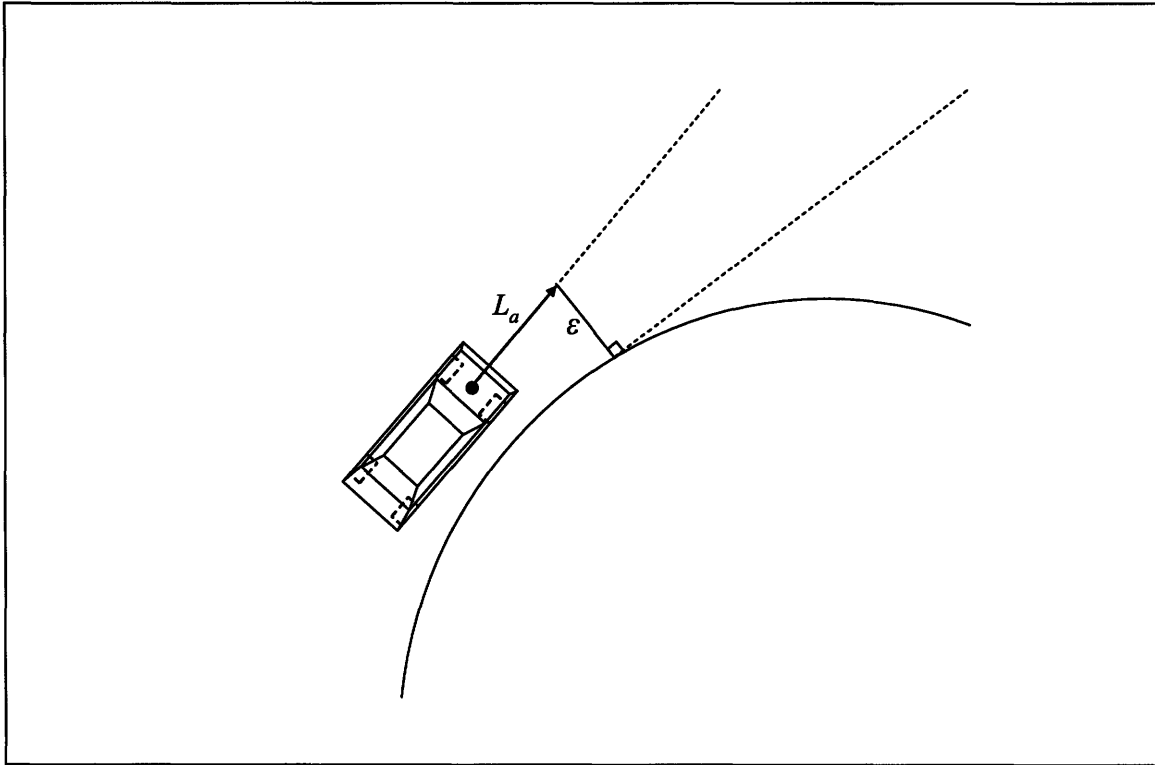


Figure 1.6: Look-ahead Distance

As mentioned at the beginning of this section, a myriad of control approaches exist for automatic steering control and only a small cross-section of the available methods has been presented here. It should be noted that many works revolve around adapting different control strategies to a variety of different sensor types. For instance, in the California PATH program [26], control algorithms were designed to work in conjunction with magnetic markers placed in the road. Sensors mounted at key locations on the vehicle detected cross-track error relative to these markers and the orientation of the vehicle with respect to the road was inferred. With regard to the DARPA Grand Challenge and the DARPA Urban Challenge, much of the work revolves around systems which make use of an IMU and a GPS receiver, as in [5], [13], [19], and [24]. An IMU is an *inertial measurement unit* that uses gyroscopes to provide accurate roll, pitch, and yaw rate information on a vehicle. Additionally, many IMUs include accelerometers which

provide linear acceleration information. Most teams participating in the DARPA challenges then used *global positioning satellite* (GPS) information to correct the drift associated with IMU sensors. A highly accurate 6-DOF estimate of a vehicle's position can then be obtained using Kalman Filters or Extended Kalman Filters. It is noted here that there is ample literature on this subject and that the issue of vehicle state estimation is not addressed in this thesis; it is assumed that this information is readily available.

1.2 MIT and the DARPA Urban Challenge

This work was conducted as part of MIT's first serious entry into the DARPA challenges. The DARPA Urban Challenge, or DUC for short, is intended to motivate the research necessary for developing autonomous vehicles for military application. The goal is to replace soldier driven vehicles with autonomous vehicles that can robustly execute the tasks assigned to them. The Urban Challenge specifically aims to have autonomous vehicles conduct simulated supplying missions inside of complex urban environments. The DUC and the previous DARPA Grand Challenges were initiated as a result of a 2001 congressional mandate (i.e. National Defense Authorization Act for Fiscal Year 2001, Public Law 106-398, Section 220) to have one-third of all ground, combat vehicles unmanned by 2015. Given the obvious military applications, however, the field of autonomous ground vehicles has the potential to revolutionize both warfare and the lives of everyday civilians. The California PATH program, for instance, was motivated by an effort to create autonomous highway systems that would reduce passenger vehicle accidents while increasing the volume of traffic flow on the United States' existing road infrastructure. Considering any industry, few would not benefit from the addition of

autonomous ground vehicles. As such, the field of unmanned ground vehicles is nearly ubiquitous in its range of applications.

The DUC differs from the previous challenges in that it will be conducted in an urban environment, as aforementioned, with moving traffic. This necessitates being able to identify traffic signals, road markings, moving vehicles, static obstacles, and pedestrians. Additionally, the vehicle has to interact with traffic in a natural fashion through intersections, passing, changing lanes, and merging. A resulting key challenge is then the need to execute maneuvers in reverse. A traditional human driver does not drive for extended periods of time in reverse, but often uses this gear setting to execute three-point turns, k -turns, and parking maneuvers. Assuming that any effective autonomous ground vehicle will exceed the capabilities of the average human driver, the decision was made by Team MIT to implement a robust reverse-driving controller that matches the performance of the forward-driving controller at low speeds (3 - 4 m/s). In terms of military applications, a vehicle could be faced with the option of executing a slow n -point turn or executing a rapid reversing procedure over a much longer, more complicated path. If the movement of supplies in an expeditious manner is critical, the vehicle should be able to effectively execute the latter of these options with no increase in the likelihood of failure. The work presented here is thus focused on selecting and developing a robust controller for traditional forward driving as well as driving in reverse.

In entering the DUC, MIT was at a distinct disadvantage because it had not participated in any of the previous challenges. Many competing teams already had several years of experience and could focus on the new challenges associated with the DUC by relying upon the low-level control systems they had already developed and

robustly tested. Team MIT thus endeavored to select an approach that could be rapidly implemented so that the strict milestones set forth in the DUC schedule would be met. Indeed the entire project of developing a fully autonomous ground vehicle was to be completed in under a year and a half. This placed three key restrictions on the low-level controller design. First, the approach had to have a history of robust, stable performance. Second, the low-level control system was to require a minimal amount of system identification so that different vehicles could be rapidly brought on board. Finally, the controller was to be implemented in such a way that future work involving different control laws could be achieved with minimal adjustment to the system's infrastructure.

To begin, Team MIT has been built around two vehicles. The test bed is a 2006 Ford Escape named *Talos-I* while the race vehicle is a Land Rover LR3 referenced as *Talos-II* (details on both of these vehicle's are provided in Appendix C). Drive-by-wire control systems purchased from Electronic Mobility Controls (EMC) have been installed on both vehicles. This system is generally used by disabled drivers and entails placing an actuator on the steering wheel as well as a single motor on both the gas and brake. These servo mechanisms then receive commands from the vehicle controller, which is the focus of this work. In general the distinction between the vehicle controller and the actuator controller (PID based) will not be made, but it should be implicitly understood. This is illustrated schematically in Figure 1.7. In general, from the perspective of planning, this is a severe simplification of the system. However, for the purposes of this thesis, this simplification will suffice.

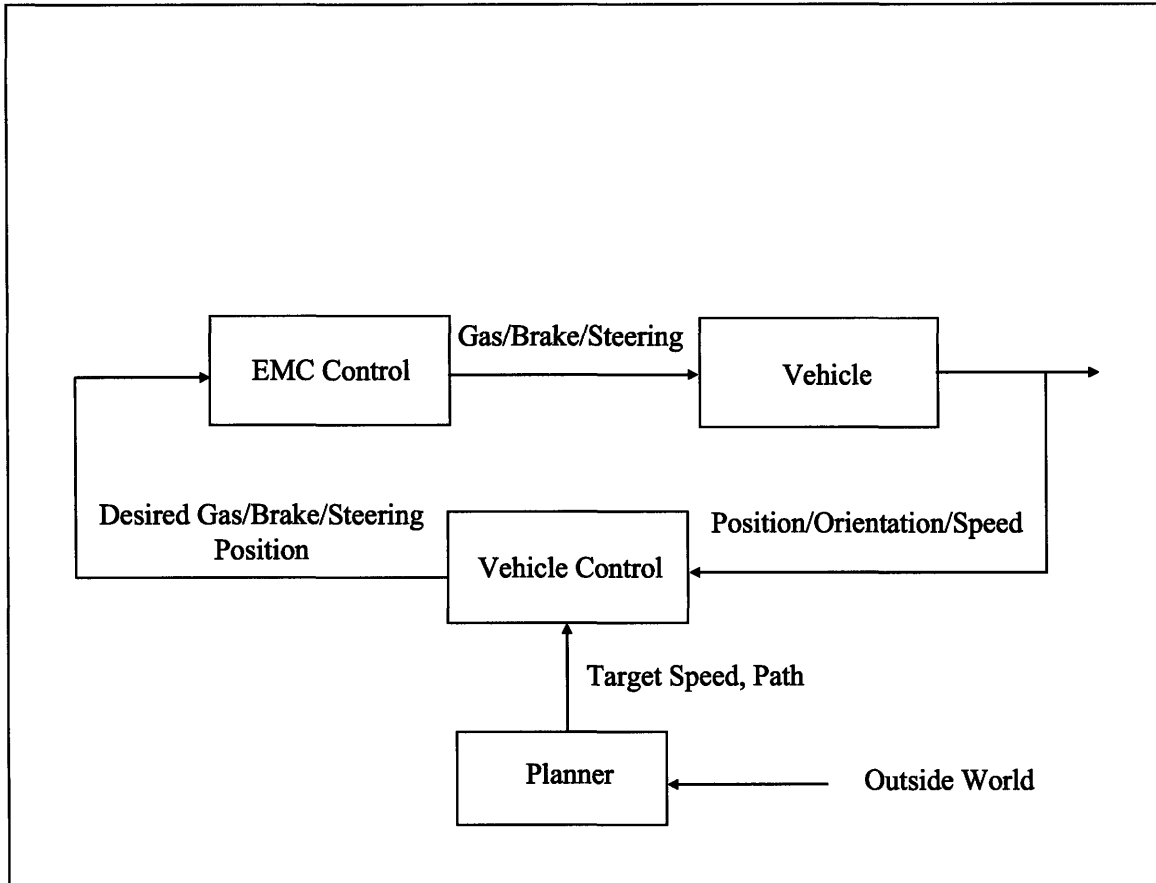


Figure 1.7: High-Level View of System Control

The vehicle control scheme adopted for this work was the pure pursuit class of controllers. This type of controller offers several key advantages over the controllers discussed in section 1.1. First, these controllers are simple to implement and have a history of robust operation. Thus, this controls approach satisfied one of the key design requirements originally outlined by Team MIT. The traditional implementation of this controller often relies upon field testing to refine performance. Alternatively, this work will present a novel method for tuning the pure pursuit controller that uses linearized models and a simple pole-zero analysis. This provided a guarantee on stability prior to any field work and helped reduce development time. Moreover, Chapter 4 will present a variation on the pure pursuit controller that used a dynamic vehicle model as opposed to the traditional Ackerman based model. Analysis showed that this approach yields an

increase in performance at higher speed. Moreover, this alteration to the traditional pure pursuit controller required only simple system identification and was thus consistent with MIT's design philosophy. Additionally, a variation on pure pursuit in which a forward simulation of the vehicle's performance is used to select the best steering input was investigated. This variation ultimately improved stability in situations in which the vehicle's initial steer angle was poorly aligned with the desired heading direction. In less extreme situations, this approach performed nearly identical with the traditional pure pursuit controller.

A second advantage of the pure pursuit class of controllers is that these controllers are flexible with respect to path representation. Unlike many linear controllers, this approach places no restriction on the smoothness of the path specified by the planner. As a result, the designers working on the planner are afforded much greater freedom to develop a variety of algorithms. Ultimately, Team MIT selected an algorithm in which paths were represented by piece-wise linear segments.

Finally, this work will show that the pure pursuit style of controller is stable for driving in reverse and is amenable to the aforementioned pole-zero analysis. Additionally, the reverse controller developed in [22] is investigated. Chapter 5 presents this controller along with a novel tuning procedure. Ultimately, it is shown that this controller is equivalent to a pure pursuit style of controller with the addition of a second tuning parameter. Though not pertinent to the fast implementation of a pure pursuit style controller, the controller of [22] is also shown to be adaptable to forward driving in Appendix B.

A safety concern for any autonomous ground vehicle project is that the controller will issue a command which may cause the vehicle to begin laterally sliding or potentially rollover. To combat these dangerous conditions, Appendix A presents modeling techniques that constrain the controller commands to a safe operating range given the vehicle's current traveling speed and orientation.

Ultimately the work presented here serves as the documentation of the control system used by Team MIT in the first DARPA Urban Challenge. As additional challenges are held, it is expected that the results of this work will be built upon and extended by future members of Team MIT. Given this expectation, a strong effort was made to develop a system architecture that could easily adapt to new control laws. That is the focus of Chapter 2 of this work. Though this chapter offers a preview of the pure pursuit style of controller, it aims to help explain some of the significant coding decisions that were made in implementing the current control system. In presenting this work as a whole, it is the hope of the author that future controls work will be completed simply and easily with little start up time.

2 Finding Control Feedback Parameters

This chapter will discuss the search algorithms required for the identification of parameters used by the pure pursuit controller. These searches include the search for the vehicle's position as projected onto the path and the associated cross-track error, the search for the pure pursuit goal point, as well as a pre-search to help initialize these two searches. Implementing these algorithms as searches ultimately helps create a system that is robust to large cross-track errors and poor path tracking. Interestingly, the topic of these searches or similarly motivated algorithms is largely ignored in autonomous, path-tracking literature. However, much of the time cost associated with implementing and developing the pure pursuit controller (or any controller for that matter) is accounted for here. Because this material is so fundamental and may be generalized to alternative control schemes, it is presented first.

2.1 Representing the Path

In order to determine the relevant parameters for implementing the pure pursuit control law, a convention for representing the desired vehicle path has to be fixed upon. From a higher-level planning perspective, it is preferred not to constrain the manner in which paths are generated. To avoid this, paths shall be represented in a piece-wise linear fashion. As illustrated in Figure 2.1, a smooth trajectory having continuous first and second order derivatives would be represented by a dense list of points in which linear

path segments join one point to the next. The denser the list of points, the more closely the piece-wise-linear path will approximate the true path. Thus this approach to path representation can be conceptualized as a discretization of a smooth path as well a means for generating non-smooth paths. Adopting this convention now allows the higher-level planner to generate paths under any convention (polynomials, clothoids, etc ...) so long as the path is discretized as a series of points before it is sent to the controller. It should be noted that the points defining the path and individual path segments will subsequently be referred to as *nodes*.

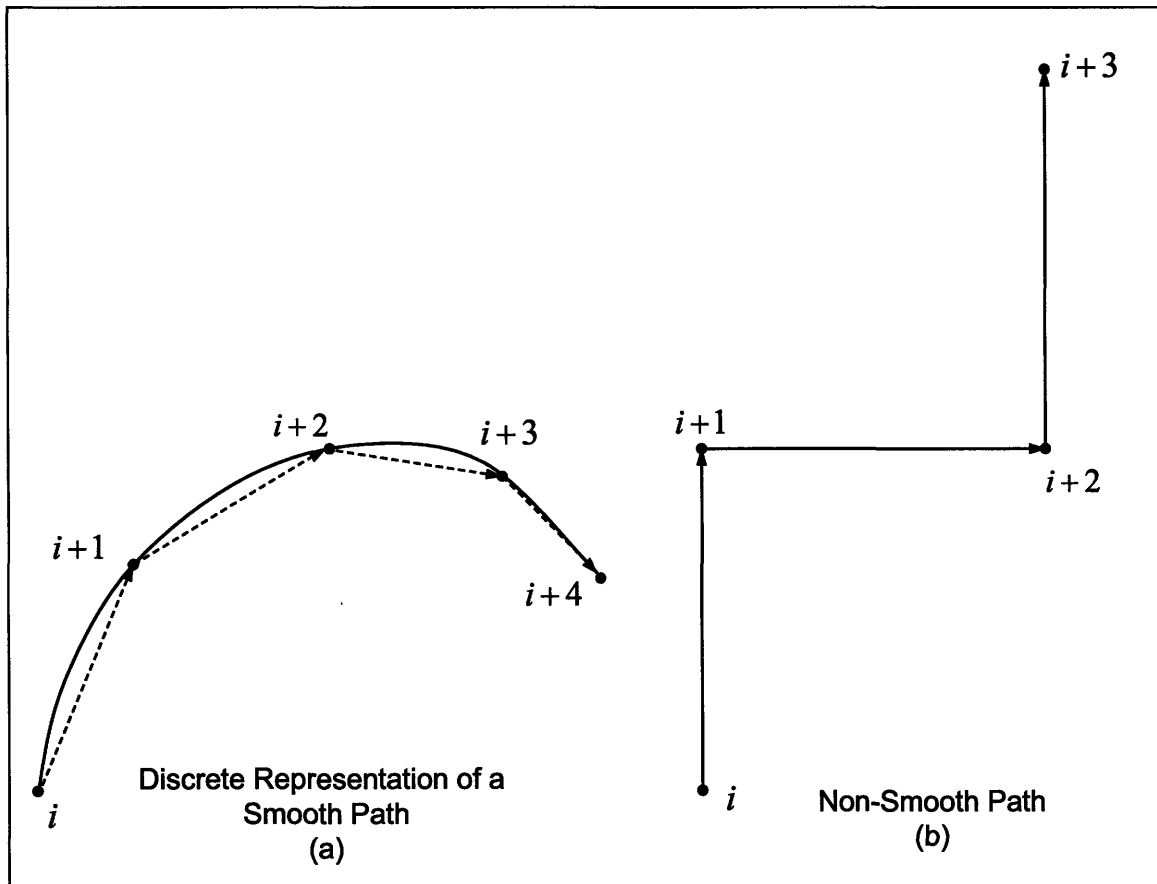


Figure 2.1: Piecewise Linear Path Representation

Several assumptions are made about how these piecewise linear paths are communicated to the controller. First, it is assumed that the discrete nodes defining the path are contained in an ordered list. The order of these nodes is the order in which each

node should be reached (the i -th node should be reached before node $i+1$). Second, it is assumed that all of the piece-wise linear segments are connected. As illustrated above in Figure 2.1, the controller will automatically connect points with a straight line. Thirdly, it is assumed that the direction of travel is always from node i to node $i+1$ (this assumption is implied by the previous assumptions, but is stated explicitly for clarity). As a consequence of these assumptions, a path segment is referenced by the i -th node which initializes it. More precisely, a segment joining node i to node $i+1$ is referenced as segment i . Finally, it is assumed that all paths are given in a two-dimensional plane in which every node has coordinates (i_x, i_y) .

2.2 Search for Position

We choose to define cross-track error (ε), referring to Figure 2.2, as the length of the line segment that passes through the vehicle's center of gravity (c.g.) and is perpendicular to the path tangent. The point at which this line intersects the path is defined as the vehicle's projected location on the path (L_x, L_y) . From this point forward, the path segment in which the vehicle's projected location lies will be referred to as the *relevant path segment*. Rigorously defining relevancy is critical as it is often the case that multiple line segments in a single path allow for a valid calculation of ε . As such, the remainder of this section will outline the definition for *relevant* as it is the foundation for determining both ε and (L_x, L_y) . Moreover, the search for the relevant path segment is critical in that it facilitates the search for the pure pursuit goal point (section 2.4).

As a result of the assumptions in section 2.1, the determination of whether a path segment is relevant or not becomes a geometric problem. The search strategy is to then proceed through the list of nodes and inspect to see if a path segment is relevant. If a

segment is not relevant, the search proceeds to the next segment defined in the list of nodes. Inherently there are several caveats that should be noted here. First, this approach has no means to distinguish between multiple segments that satisfy the conditions for relevancy. As such, a pre-search that will be explained in section 2.3 will be used to locate the vehicle in the path every time a new path plan is sent to the controller. The relevancy search algorithm is then able to safely select the first segment in the list of nodes that is relevant and terminate. Second, for long paths or highly discretized paths, this approach will be inefficient. To improve this, the search algorithm keeps track of the last path segment that was found to be relevant and initializes the next search from this location in the list of nodes. This counter must, of course, be reset every time a new path is received.

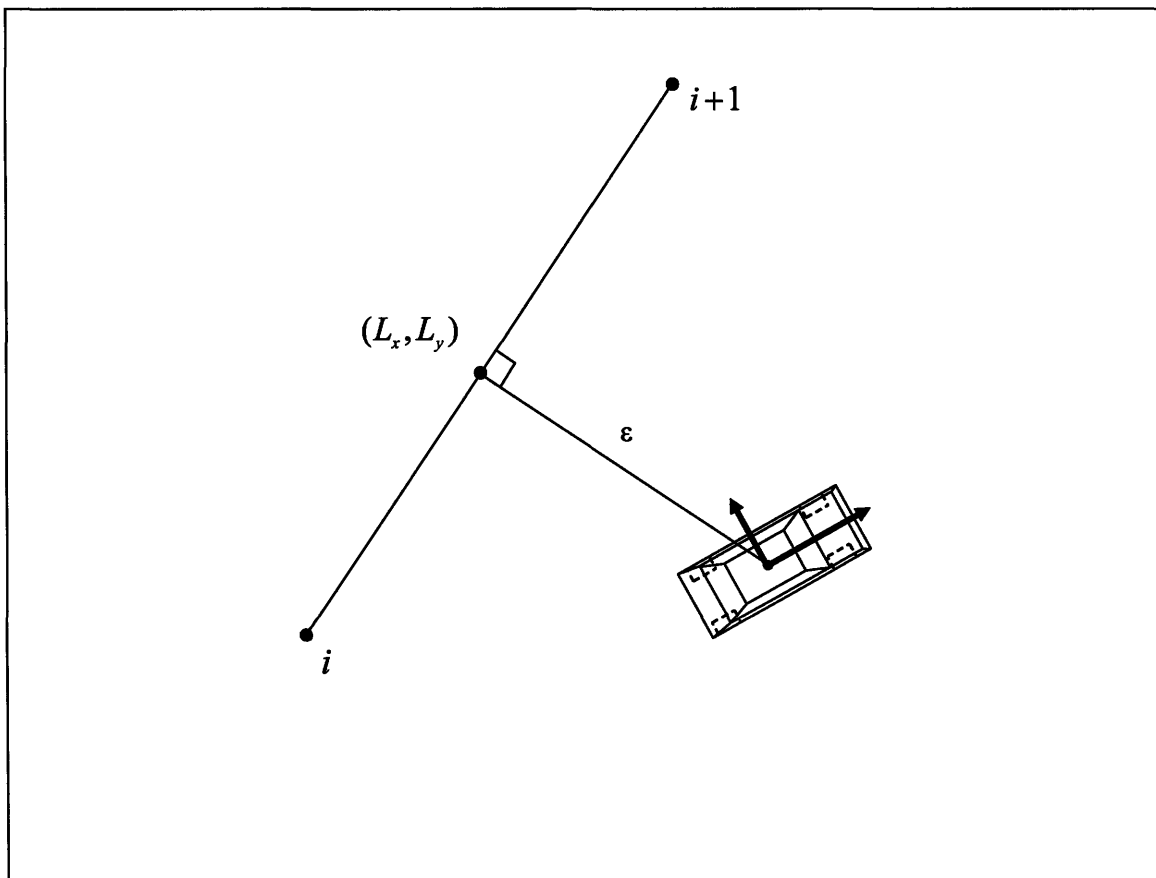


Figure 2.2: Cross-track Error

The determination of whether a path segment is relevant requires the calculation of several parameters. Referring to Figure 2.3, the vector Q_1 is the vector from the i -th node to the vehicle's c.g. location. The vector Q_2 is the vector from the vehicle's c.g. location to node $i+1$. The angles from these vectors to the path segment are α_1 and α_2 respectively. Finally, the angle from the i -th to the $i+1$ -th path segment is θ . Here the subscripts 1 and 2 denote the first (i -th) and second ($i+1$ -th) nodes of a path segment, not the first and second nodes in a node list. In implementing the search for relevancy, these parameters are all calculated upon the inspection of each path segment. Additionally, the signs of all angles are defined by a right-handed coordinate system in which the z axis is projected out of the page. Moreover, it is assumed that these parameters and the path are all defined in the same coordinate system in which the vehicle's position is known. It is also assumed that knowledge of the vehicle's position and orientation is available, as noted in Chapter 1 of this thesis.

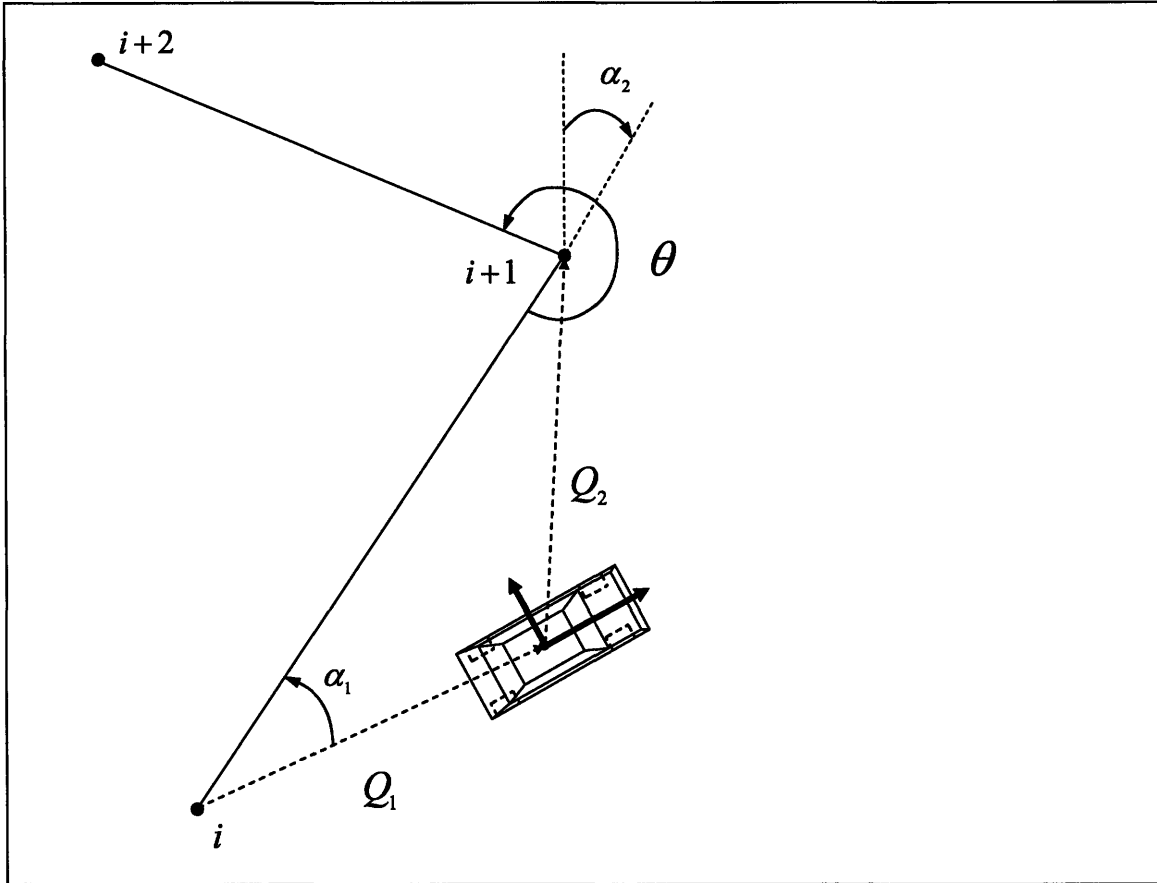


Figure 2.3: Vehicle Relative Position Parameters

In the most trivial case of a series of straight path segments, relevancy is defined by the dashed boundary lines illustrated in Figure 2.4. These lines project from the nodes defining a path segment and are perpendicular to the path itself. If the vehicle's position lays between the bounding lines of nodes i and $i+1$, then segment i is relevant and the vehicle's projected location lies on this segment.

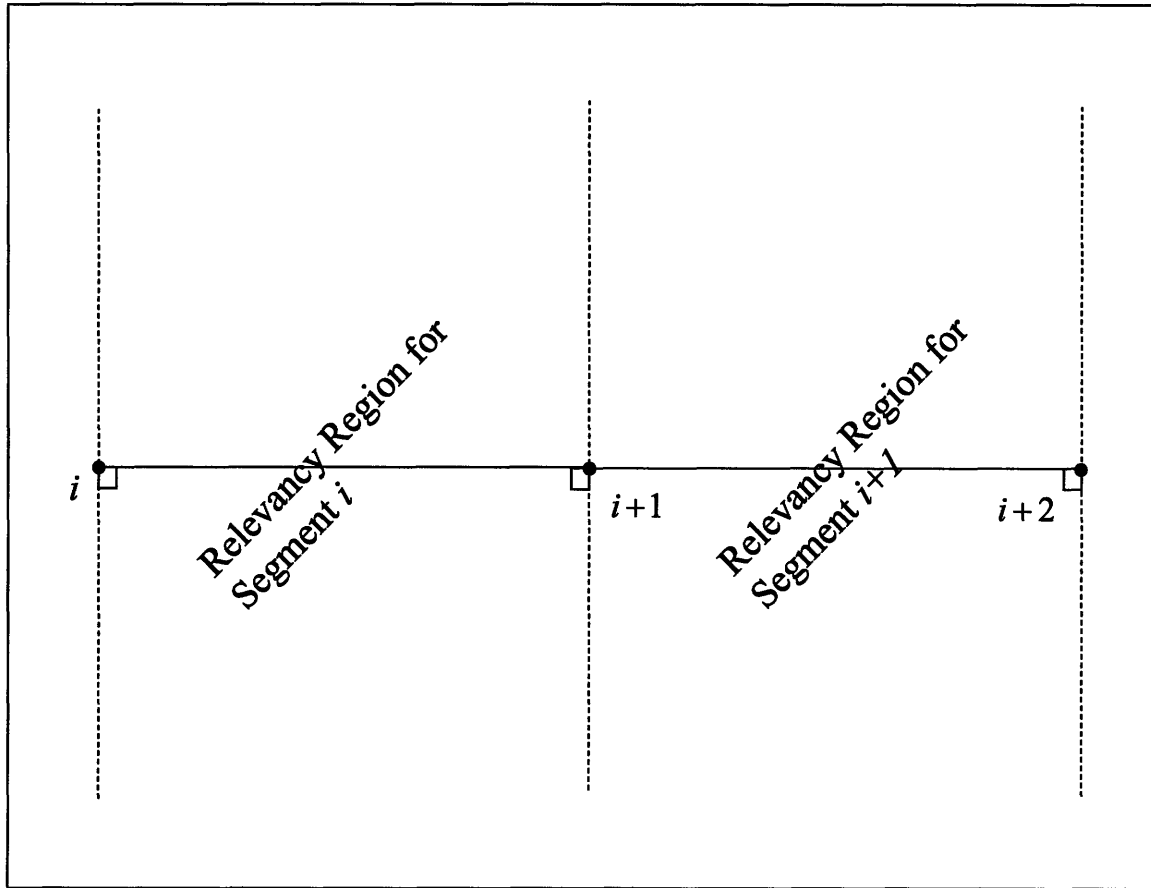


Figure 2.4: Relevancy Regions for a Simple Linear Path

The case illustrated in Figure 2.4 is unrealistic; it would be redundant for a higher level planner to develop a series of collinear path segments as opposed to a single path segment of identical length. However, this case is illustrative of the overall concept of relevancy and the methodology that will follow throughout this section. Analyzing Figure 2.4, one finds that both $|\alpha_1|$ and $|\alpha_2|$ dictate relevancy. If the magnitudes of both of these angles are less than or equal to $\pi/2$, then the segment is relevant. If the magnitude of either of these angles is greater than $\pi/2$, then the segment is not relevant. Ultimately, every relevant path segment satisfies these conditions. The determination of relevancy is, however, made significantly more complicated by path segments which are not collinear. As illustrated in Figure 2.5, the bounding lines for non-collinear path segments create an *overlapping zone* on the inside of a turn and a *dead zone* on the outside of a turn.

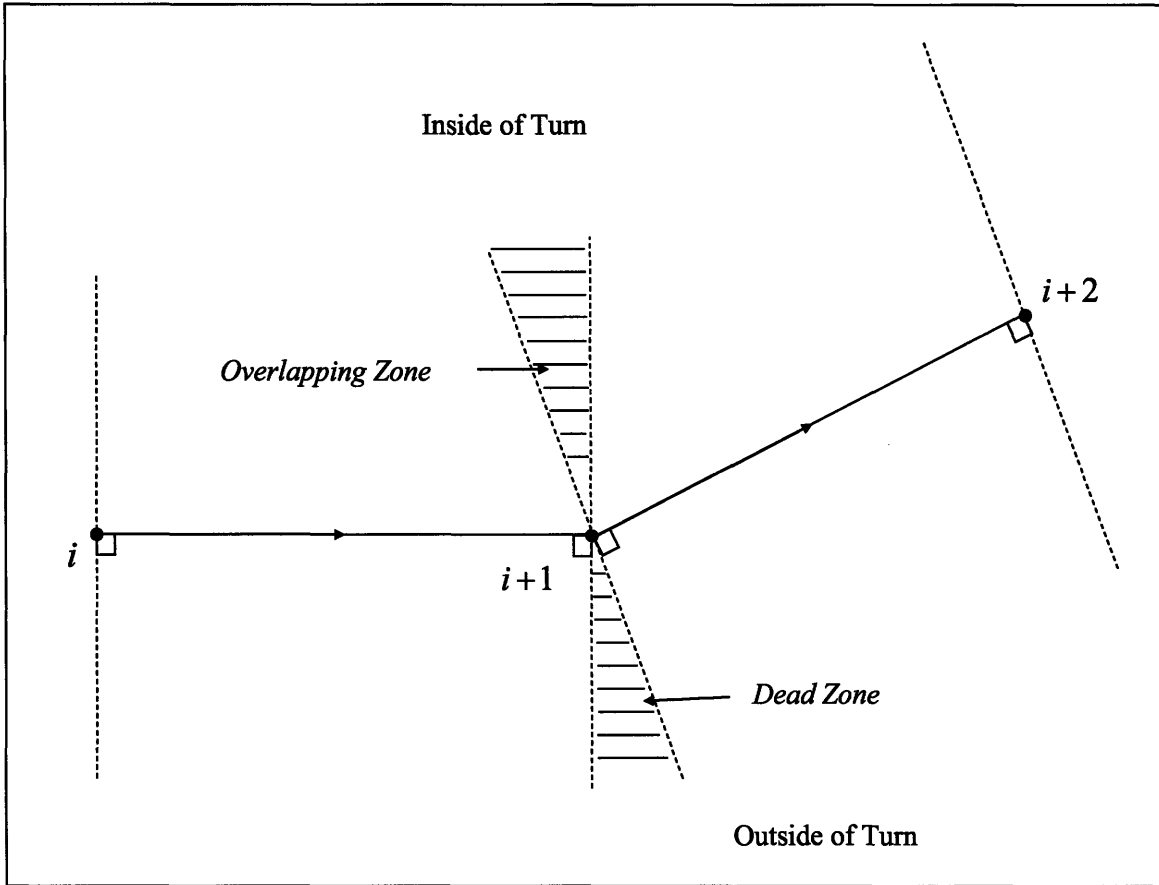


Figure 2.5: Relevancy Zones For Non-Collinear Segments

Referring to Figure 2.6, we define what will be termed the *bisector angle* β as half of the angle between two joined line segments ($\beta = \theta/2$). If the vehicle is on the inside of a turn defined by segments i and $i+1$, relevancy switches from segment i to segment $i+1$ when the vehicle crosses the line bisecting these two segments. This effectively eliminates the ambiguity associated with the overlapping zone and ensures that the vehicle's position on the path is continuous. Analyzing the geometry of Figure 2.6 reveals that one of the following two conditions must be verified for a vehicle to be traveling on the inside of a turn:

$$\alpha_2 \leq 0 \text{ and } \beta > 0 \quad (2.1)$$

or

$$\alpha_2 \geq 0 \text{ and } \beta < 0 \quad (2.2)$$

Conditions 2.1 and 2.2 facilitate the evaluation of four potential cases: the vehicle is on the inside of a turn, the vehicle is on the outside of a turn, the vehicle is in a dead zone, or the first node is in front of the vehicle.

i. Inside of Turn

In this case, either condition 2.1 or condition 2.2 is satisfied. If the segment is relevant, then the following condition will also be met:

$$|\alpha_2| \leq |\beta| \tag{2.3}$$

If condition 2.3 fails, then the vehicle is on the inside of a turn and the segment under inspection is not relevant because the vehicle has crossed the bisecting line. A case in which conditions 2.1 and 2.3 are satisfied is illustrated in Figure 2.6.

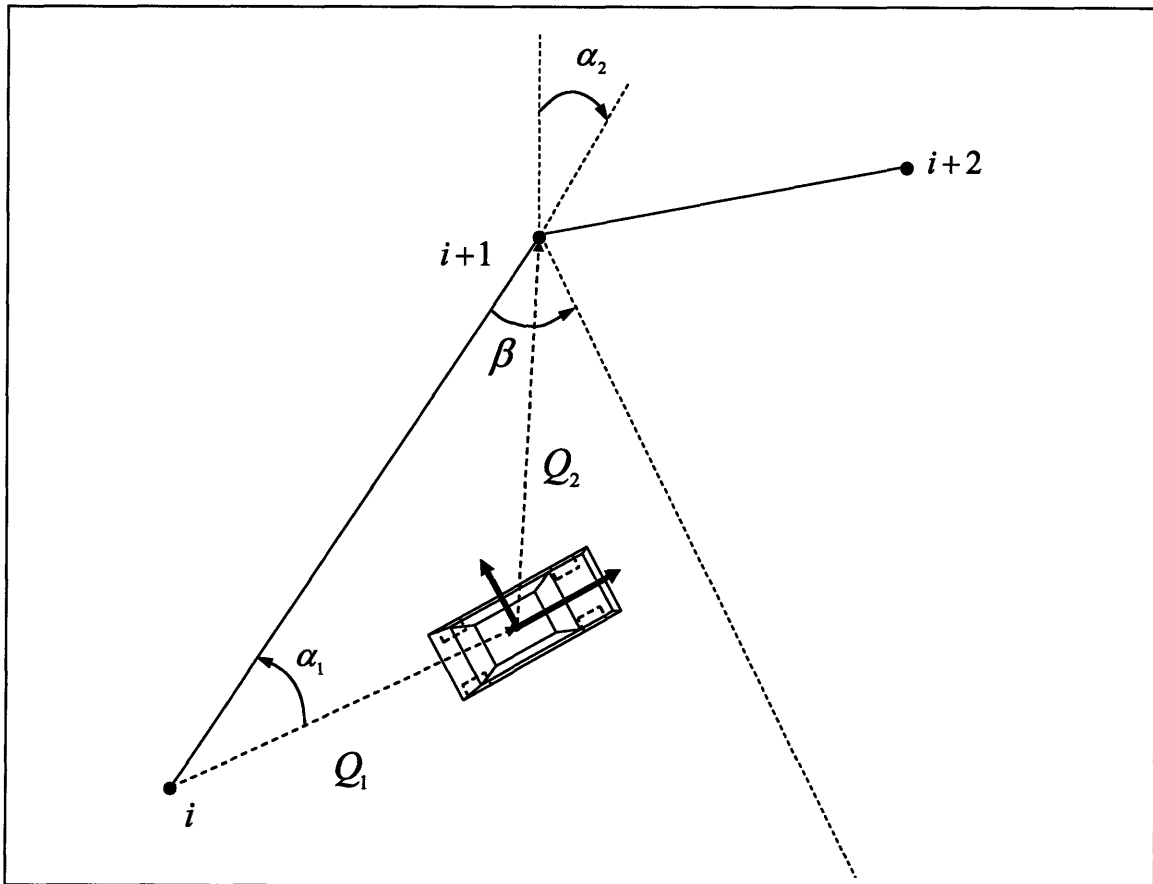


Figure 2.6: Vehicle Traveling on the Inside of a Turn

ii. Outside of Turn

If conditions 2.1 and 2.2 both fail, then the vehicle is traveling on the outside of a turn and the vehicle's position relative to a bisecting line does not need to be evaluated. If the magnitudes of both α_1 and α_2 are less than or equal to $\pi/2$, then the vehicle is on the outside of the turn and the segment is relevant. If the magnitude of either α_1 or α_2 is greater than $\pi/2$, then the vehicle is on the outside of a turn and the path segment is either not relevant or the vehicle is in a dead-zone.

iii. Dead-Zone

The dead zone case is now presented in Figure 2.7. In this instance, the vehicle is on the outside of a turn, but it is ambiguous as to whether segment i or $i+1$ is relevant. The ambiguity occurs because $|\alpha_2|$ for segment i is greater than $\pi/2$ and $|\alpha_1|$ for segment $i+1$ is also greater than $\pi/2$. To verify that the vehicle is in the dead zone, conditions 2.1 and 2.2 must fail and the following two conditions must be met:

$$|\alpha_2| > \frac{\pi}{2}$$

&

(2.4)

$$|\alpha_2| - |\theta| - \frac{\pi}{2} < 0$$

If these conditions are verified, the corner point $i+1$ is selected as the vehicle's position on the path. The cross-track error is then defined as the distance to this point. In order that some segment is always considered relevant, the i -th segment is chosen as the relevant path segment. This is done so that the search algorithm can initialize the next search from this location in the node list. If the $i+1$ -th segment were defined as the relevant segment, at the next iteration the search algorithm would not be able to identify

if the vehicle was still in the dead zone as the search algorithm always looks forward in the list of nodes. Once the vehicle leaves the dead zone, the previous cases are applied.

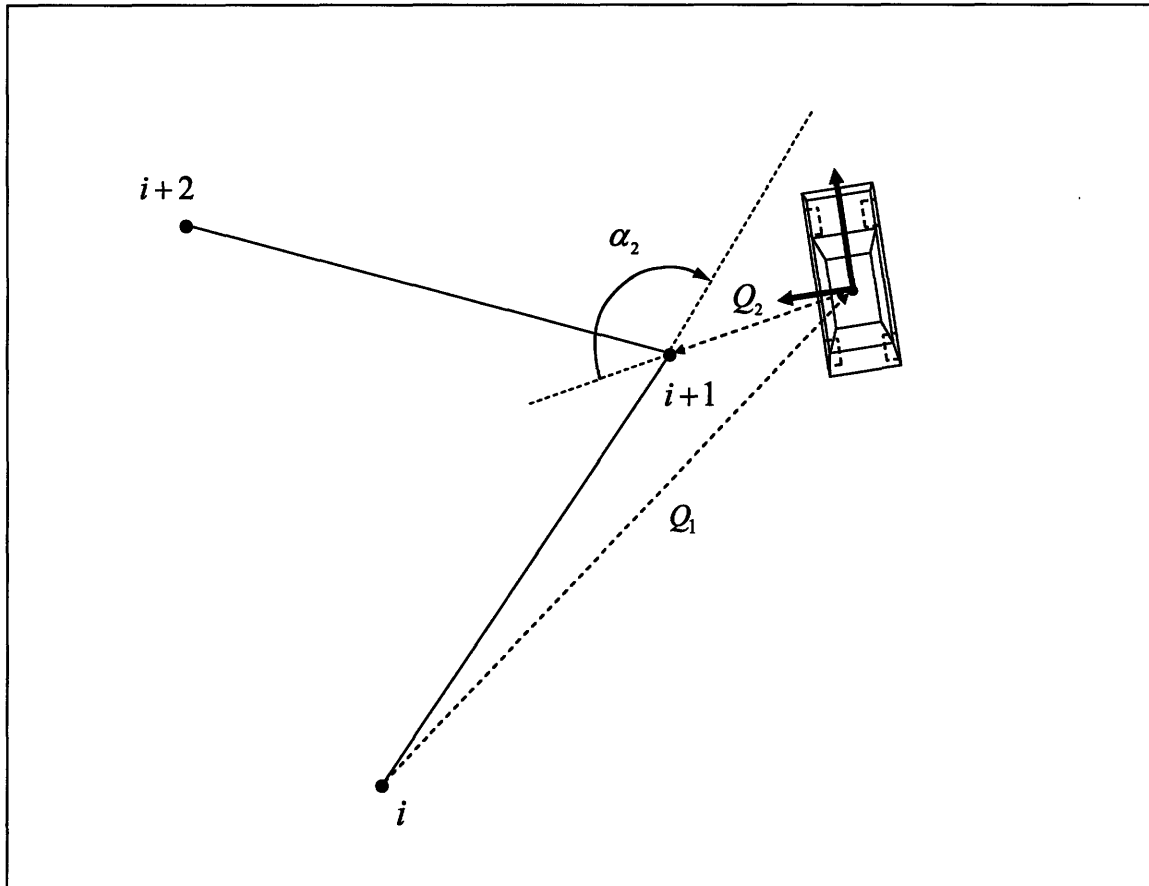


Figure 2.7: Dead Zone Case

iv. First Node in Front of Vehicle

As a final consideration, if the magnitude of α_1 from the first node in a new path list is verified as being greater than $\pi/2$, then the first path segment is assumed in front of the vehicle. This case might occur if the higher level planner plans from a point some distance in front of the vehicle as opposed to the vehicle's c.g. position at the time of planning. In order that the vehicle will proceed along the path, the first segment is selected as the relevant segment. An illustration of this case is presented in Figure 2.8.

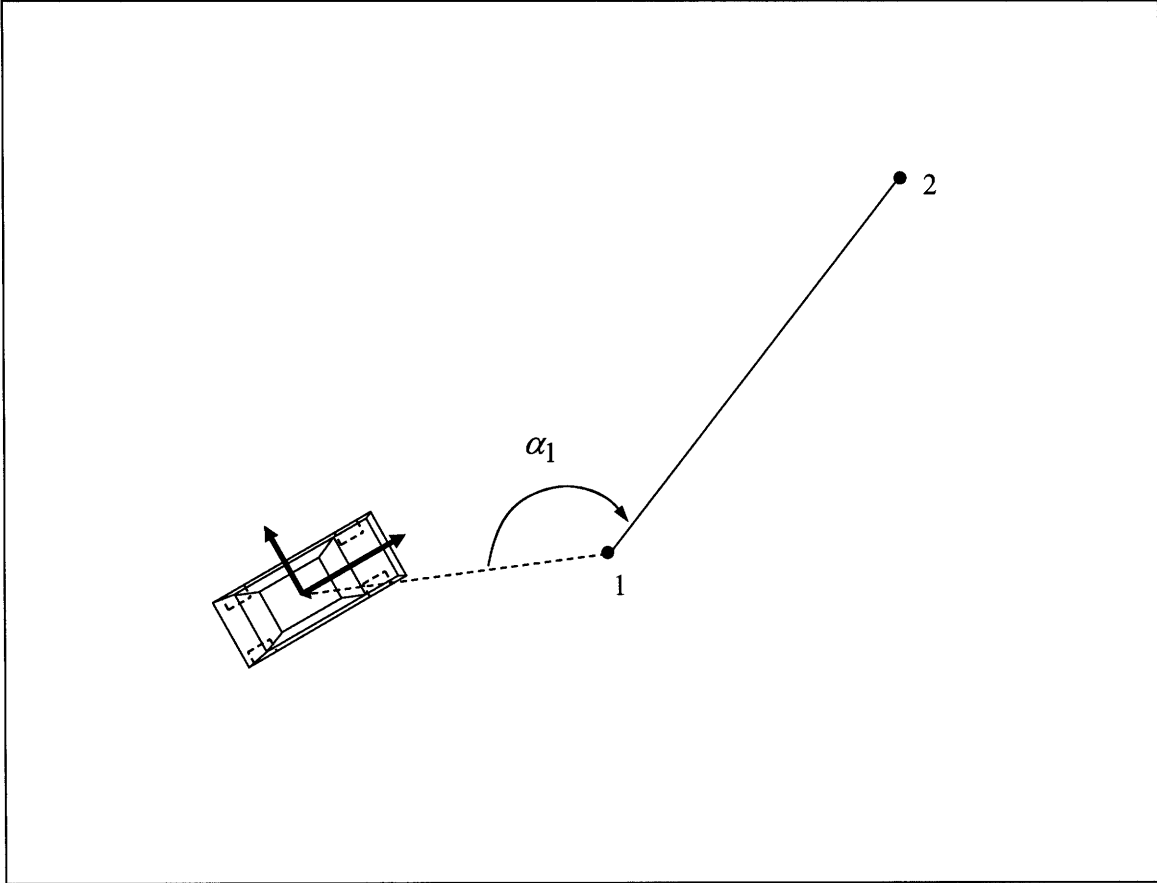


Figure 2.8: First Node in Front of Vehicle

To summarize the potential cases, Table 2.1 is presented below. Here only the conditions that are necessary and sufficient to verify the case are listed.

| Case | Verified Condition | Failing Conditions |
|--------------------------------|---|---|
| First Node in Front of Vehicle | $ \alpha_1 > \pi/2 \& i_r = 1$ | - |
| Inside of Turn – Relevant | $ \alpha_1 \leq \pi/2$ $\alpha_2 < 0 \& \beta > 0$ (2.1) or $\alpha_2 > 0 \& \beta < 0$ (2.2) $ \alpha_2 \leq \beta $ (2.3) | - |
| Inside of Turn – Not Relevant | $ \alpha_1 \leq \pi/2$ $\alpha_2 \leq 0 \& \beta > 0$ (2.1) Or $\alpha_2 \geq 0 \& \beta < 0$ (2.2) | $ \alpha_2 \leq \beta $ (2.3) |
| Outside of Turn – Relevant | $ \alpha_1 \leq \pi/2, \alpha_2 \leq \pi/2$ | $\alpha_2 \leq 0 \& \beta > 0$ (2.1) $\alpha_2 \geq 0 \& \beta < 0$ (2.2) |
| Outside of Turn – Not Relevant | $ \alpha_1 \leq \pi/2, \alpha_2 > \pi/2$ | $\alpha_2 \leq 0 \& \beta > 0$ (2.1) $\alpha_2 \geq 0 \& \beta < 0$ (2.2) $ \alpha_2 - \theta - \pi/2 < 0$ (2.4) |
| Dead Zone | $ \alpha_1 < \pi/2$ $ \alpha_2 - \theta - \pi/2 < 0, \alpha_2 > \pi/2$ (2.4) | $\alpha_2 \leq 0 \& \beta > 0$ (2.1) $\alpha_2 \geq 0 \& \beta < 0$ (2.2) |

Table 2.1: Case and Conditions for Relevancy

A flow diagram illustrating this relevancy search is provided in Figure 2.9.

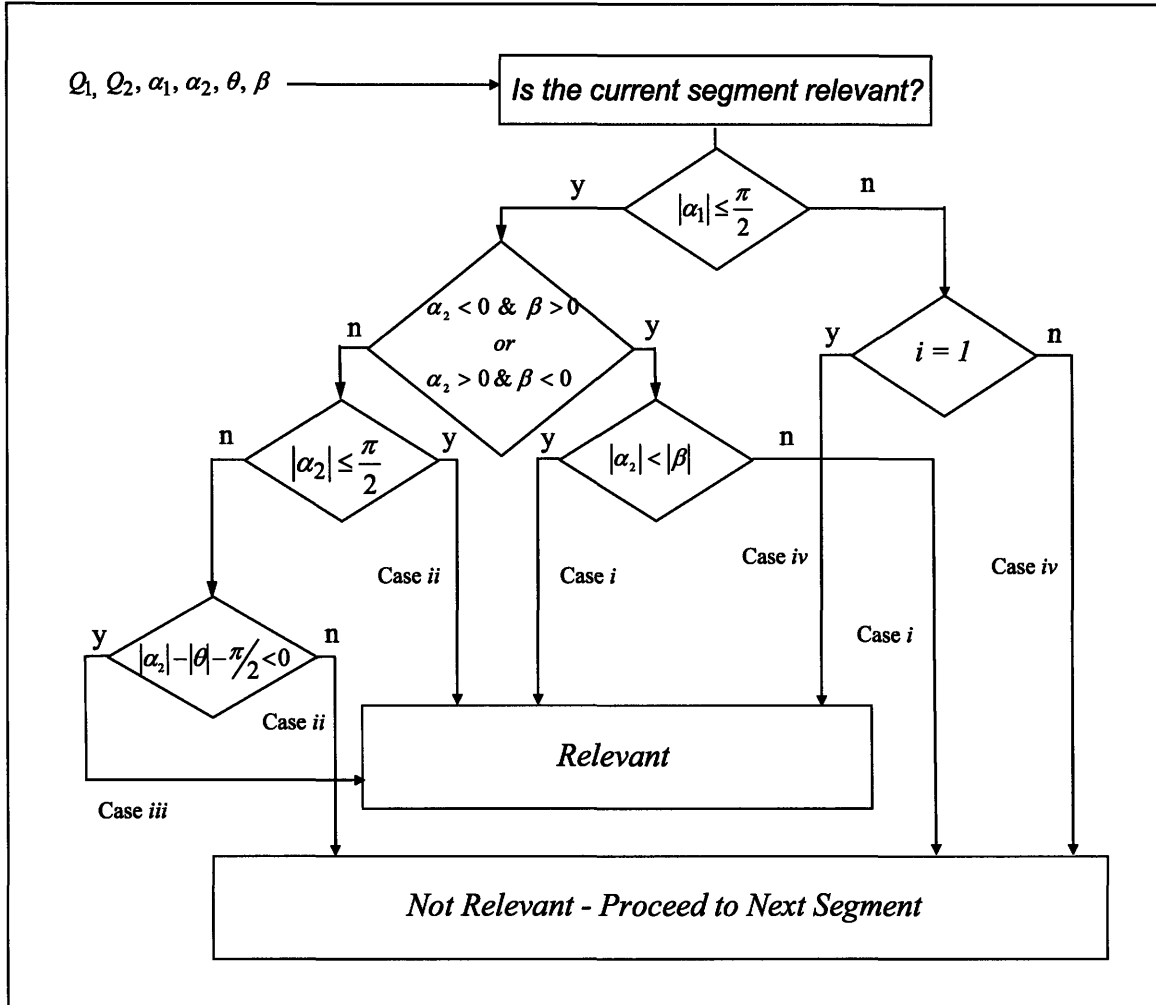


Figure 2.9: Flow Diagram for Finding Relevance

Once the relevant segment is selected, provided it is not a dead zone case, the cross-track error (ε) is defined simply below:

$$\varepsilon = -|Q_1| \sin \alpha_1 \quad (2.5)$$

The cross-track error is a vector quantity as it will have both a direction and a magnitude. One can consider a coordinate system with its origin located at the i -th node with the x axis directed along the relevant line segment and the z axis projected out of the page. A positive cross-track error is then in the positive y direction and a negative cross-track error is in the negative y direction, relative to this path-segment fixed coordinate system.

Additionally, the x and y coordinates of the vehicle's projected location (as represented in an identical coordinate system as the desired path and the vehicle's position) are defined as:

$$(L_x, L_y) = \left(i_x + \frac{|Q_1| i_x \cos \alpha_1}{\sqrt{i_x^2 + i_y^2}}, i_y + \frac{|Q_1| i_y \cos \alpha_1}{\sqrt{i_x^2 + i_y^2}} \right) \quad (2.6)$$

This equation is applied even in case *iv* (First Node in Front of Vehicle).

Finally, the vehicle's distance along the relevant path segment is defined as the distance between the point (L_x, L_y) and (i_x, i_y) . This value is presented below in equation 2.7. In the dead zone case, D_{ap} will be equivalent to the length of the i -th path segment. In the case in which the first path segment is in front of the vehicle, D_{ap} is assigned a negative value and (L_x, L_y) are still defined as in equation 2.6.

$$D_{ap} = \sqrt{(L_x - i_x)^2 + (L_y - i_y)^2} \quad (2.7)$$

This quantity proves useful for a number of different control's approaches. Specifically, [16] proposes an approach in which steering and speed control are coupled, thereby making it necessary to track the vehicle's progress along the path and compare that quantity with the expected position of the vehicle at that instant. Using equation 2.7 and summing D_{ap} with the lengths of all preceding path segments then provides an estimate of the total distance the vehicle has traveled along the path.

Consideration must now be given to how the vehicle's projected location on the path and the associated cross-track error are defined as the vehicle completes a path. The search algorithm can easily identify such a situation if the previously relevant path segment is the last segment defined by the node list and the magnitude of α_2 is greater than $\pi/2$. If this is the case, the search algorithm extends the last path segment to infinity

and proceeds to determine cross-track error and the vehicle's projected location from equations 2.5 and 2.6.

2.3 Pre-Search

The search for a vehicle's projected location on the path can be complicated by sending the vehicle the path at some time t and then resending the identical path (or a nearly identical path) at a later time $t+\Delta t$. In practice, this situation might arise for several reasons. As an example, the higher-level path planner sends a path and the low-level controller immediately attempts to track this path. Before the path planner supplants the current path with a new one, a sub-routine (using additional sensor data, such as GPS information), may update the location of the nodes defining the existing path. This update will result in the controller receiving a nearly identical path to the one it is already executing. In the case of the path illustrated in Figure 2.10, the vehicle would select the first path segment as the relevant path segment rather than correctly selecting the third path segment. This occurs because multiple path segments satisfy relevancy conditions and the algorithm of section 2.2 treats every path it receives as *new* and will automatically reset and select the first relevant path segment. As a result it is imperative that the vehicle competently initialize its position along any path it receives.

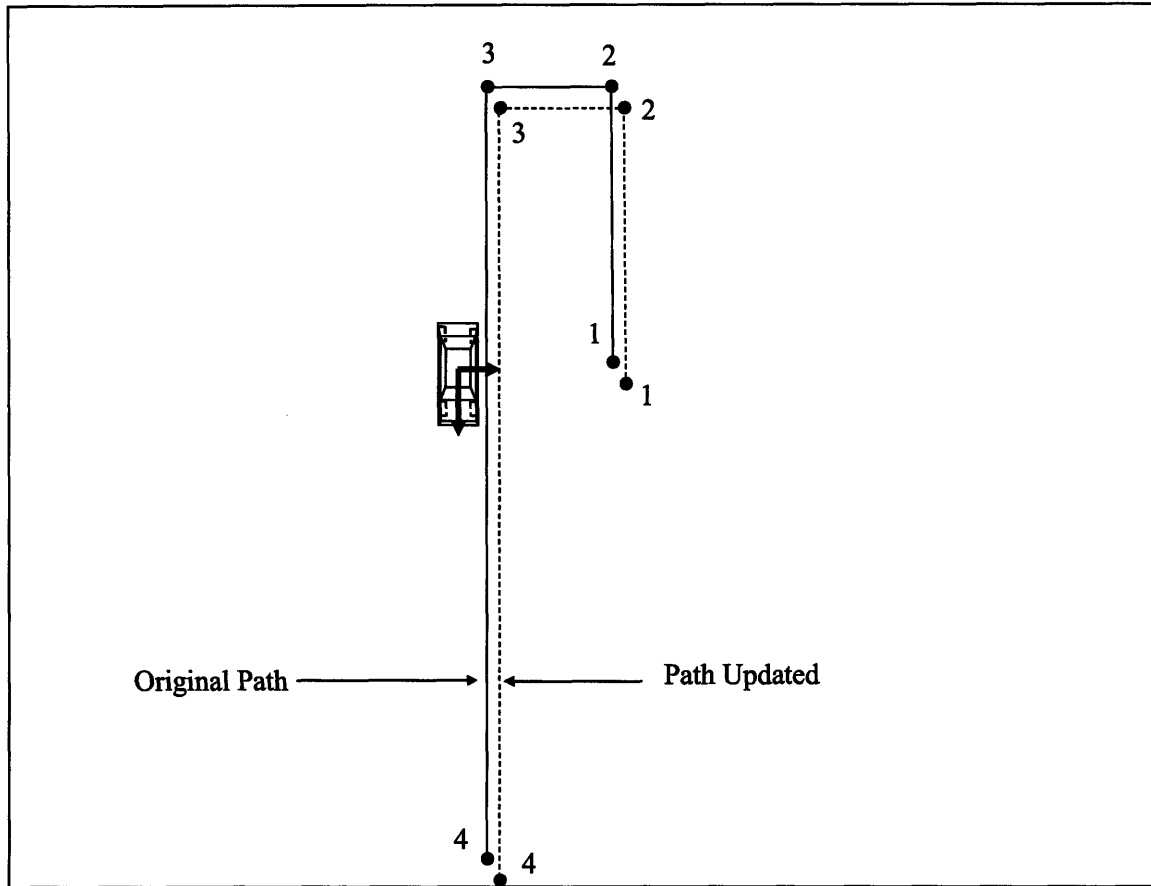


Figure 2.10: Update to a Path with Parallel Line Segments

The typical approach [8] to this problem is to begin searching for the relevant path segment by searching for the closest node. That is to say, the path segment with the closest point i will be selected to initialize the search algorithm discussed in sections 2.2. As a consideration in implementing this approach, because a path is defined by the i -th segment, it must be determined whether the closest point is generally in front of or behind the vehicle. If the point is in front of the vehicle, all proceeding searches should be initialized at the $i-1$ path segment. If the closest node is generally behind the vehicle, that node may be used to initialize the search. This is a result of the convention selected for referencing path segments. It is here noted that a node is considered in front of the vehicle if the magnitude of the angle between the x axis of the vehicle's body fixed frame

and a vector joining the origin of this frame and the node is less than $\pi/2$. Otherwise the node is considered behind the vehicle. This is illustrated graphically in Figure 2.11.

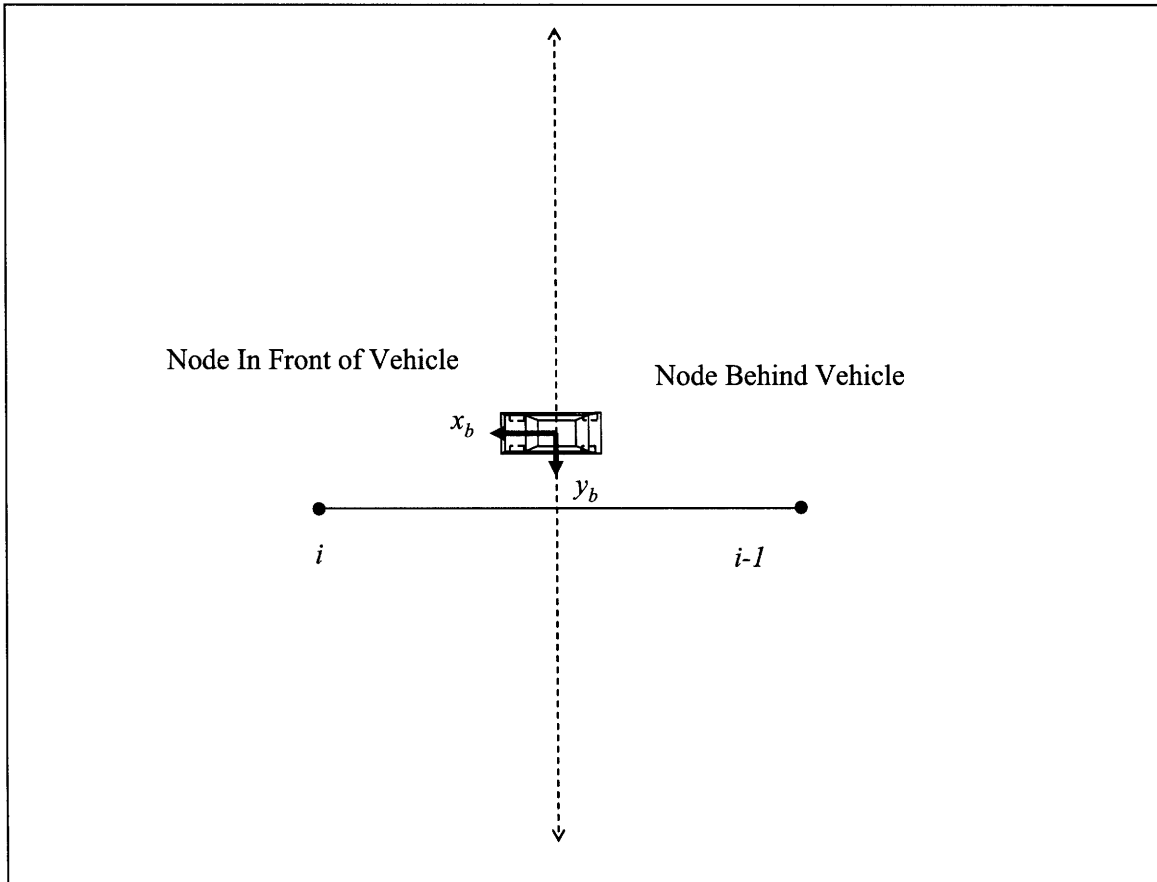


Figure 2.11: Node Position Relative to Vehicle

In practice, this approach works well. However, in the case illustrated in Figure 2.10, the nodes are orientated such that node 1 is the closest node. If the path was resented at this point, the vehicle would still begin tracking the first line segment which had already been traversed. An intuitive approach to solving this problem would be to discard path segments which would require a full 180° (or a near 180°) turn. However, this requires an arbitrary metric or weighting function for defining what is too great a change in orientation. This may work well with many control algorithms, but this approach will eliminate a key feature of pure pursuit. Specifically, to initiate a turn around maneuver in pure pursuit, the path can be specified precisely (an arc or a loop), or a path consisting of

a line directly behind the vehicle can be sent and the pure pursuit algorithm will initiate the turn automatically. This will be explained in greater detail in Chapters 4 and 5.

To preserve this capability as well as eliminate the complication of sending identical paths repeatedly, a search for the segment with the smallest cross-track error is done as a pre-search to initialize all other searches. The search process is to select each path segment and check for relevancy. If the segment is relevant, the cross-track error is calculated and stored in memory along with the segment number. If the segment is not relevant, no action is taken. The segment with the smallest cross-track error is selected as the segment at which all proceeding searches should be initialized.

In general, this approach works well but is not fail safe. A situation in which two segments have identical cross-track error could arise. In such a case, the segment which is more closely aligned with the vehicle's heading should be selected. This approach will safely select the correct segment when the vehicle is tracking the path within a reasonable tolerance and will make a reasonable decision when there are multiple path segments to consider.

An argument can be made that this search is unnecessary as long as any existing path is pruned of traversed path segments before it is resent. However, this can be a non-trivial problem if the path is shifting due to GPS fluctuations. A segment can be considered traversed or completed at one time, but incomplete once new GPS data is taken. Thus the addition of a pre-search adds a layer of redundancy even if conservative pruning is done to the path.

2.4 Search for Goal Point

The pure pursuit control algorithm selects a look-ahead distance (L_a) which defines a *virtual circle* surrounding the vehicle's position at any point in time. The point at which this circle intersects the path defines the *goal point* (G_x, G_y), as illustrated in Figure 2.12. The steering commands issued by the pure pursuit controller then use the goal point as an instantaneous target, as will be explained in greater detail in section 4.1. As with the search of section 2.2, it is necessary to search for this goal point in a robust fashion. The decision to represent the path with piece-wise linear segments now becomes particularly convenient. The intersection of a circle and a polynomial is a non-trivial problem to solve analytical. The intersection of a line and a circle is, however, very convenient to solve.

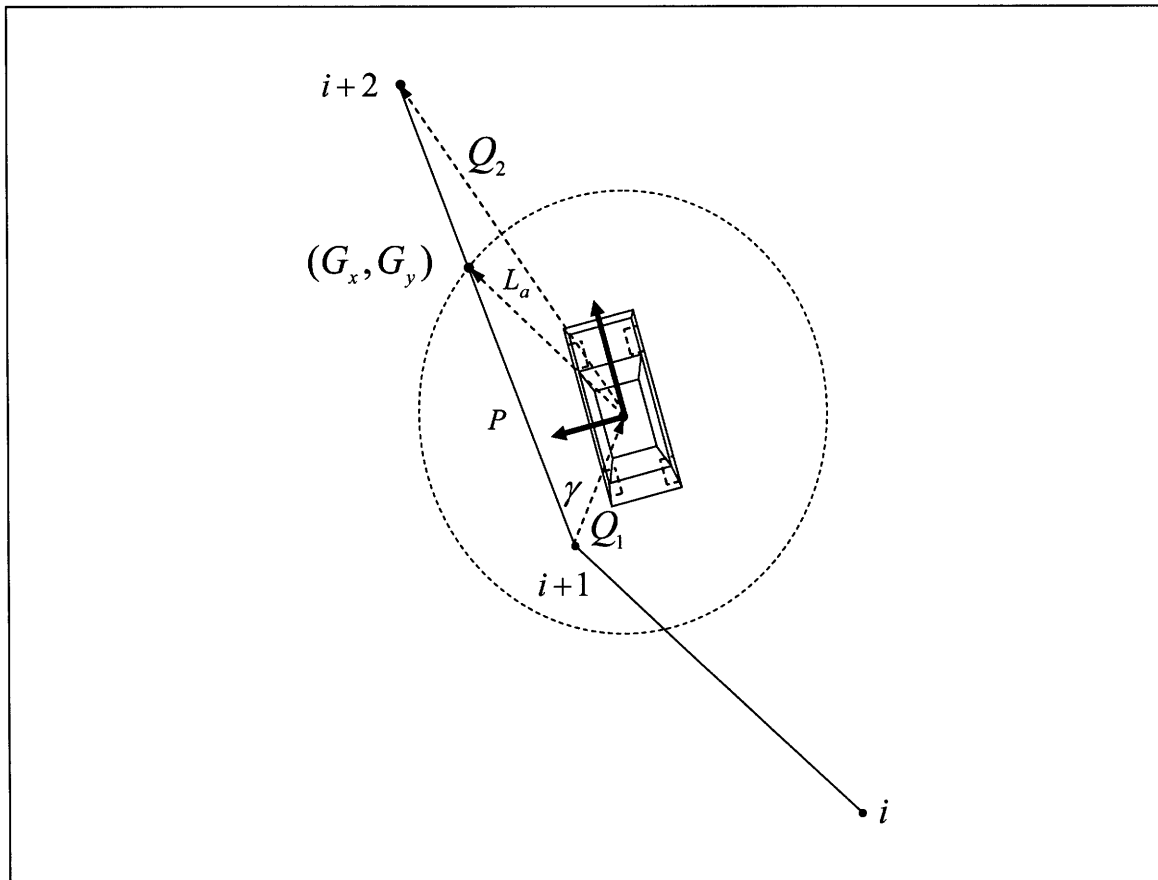


Figure 2.12: Goal Point for Pure Pursuit

The search for the goal point involves sequentially inspecting each path segment to determine if the goal point lies along that portion of the path. As in section 2.2, there are a finite number of cases that must be considered and the remainder of this section will present those cases. It is assumed before the search for the goal point is initiated, however, that the following pieces of information are already known: the relevant section, the vehicle's location on the path, and the cross-track error. All of this information is garnered by running the search of section 2.2 prior to this algorithm.

i. First Node Outside Virtual Circle

The search algorithm initially checks to see if the relevant line segment is the first path segment. If this is the case and the first node is also found to be outside of the vehicle's virtual circle, then the first node in the node list is chosen as the initial goal point. This condition is checked by verifying that the magnitudes of Q_1 and Q_2 (for the first path segment) are both greater than L_a and that the magnitude of α_1 is greater than $\pi/2$.

ii. Common Case

The search proceeds to check whether the magnitude of Q_1 is less than L_a and the magnitude of Q_2 is greater than L_a . If this is the case, it is obvious that the intersection of the vehicle's virtual circle and the path occurs along the i -th segment. Referring to Figure 2.12, the distance P can be found by applying the law of cosines twice, resulting in the following pair of equations:

$$L_a^2 = P^2 + |Q_1|^2 - 2P|Q_1|\cos\gamma \quad (2.8)$$

$$|Q_2| = |Q_1|^2 + L_p^2 - 2L_p|Q_1|\cos\gamma \quad (2.9)$$

Here L_p is the length of the path segment in which the goal point is to be found. The solution to the system is then given by the following:

$$\cos \gamma = \frac{|Q_2|^2 - |Q_1|^2 - L_p^2}{-2L_p|Q_1|} \quad (2.10)$$

$$P = |Q_1| \cos \gamma + \sqrt{|Q_1|^2 (\cos^2 \gamma - 1) + L_a^2} \quad (2.11)$$

The location of the goal point is then simply given below:

$$(G_x, G_y) = P \left(\frac{i_{x+1} - i_x}{L_p}, \frac{i_{y+1} - i_y}{L_p} \right) + (i_x, i_y) \quad (2.12)$$

With a dense set of nodes, the above case will be the most frequently applied, thereby justifying the name *common case*.

iii. Two Intersection Points

When large line segments define the path, several other situations may occur. In the case illustrated in Figure 2.13, the magnitude of Q_1 and Q_2 are greater than the look-ahead distance L_a , but the virtual circle still intersects the illustrated path segment. This situation is then identified by the fact that the cross-track error is less than the magnitude of the look-ahead distance. It is clear that:

$$P = \sqrt{L_a^2 - \epsilon^2} . \quad (2.13)$$

The coordinates of the goal point are then defined as the following:

$$(G_x, G_y) = P \left(\frac{i_{x+1} - i_x}{L_p}, \frac{i_{y+1} - i_y}{L_p} \right) + (L_x, L_y) . \quad (2.14)$$

It should be noted that this case only needs to be considered for the relevant path segment. More precisely, as the algorithm iterates through the path segments, it can stop considering this case once it has proceeded beyond the relevant path segment.

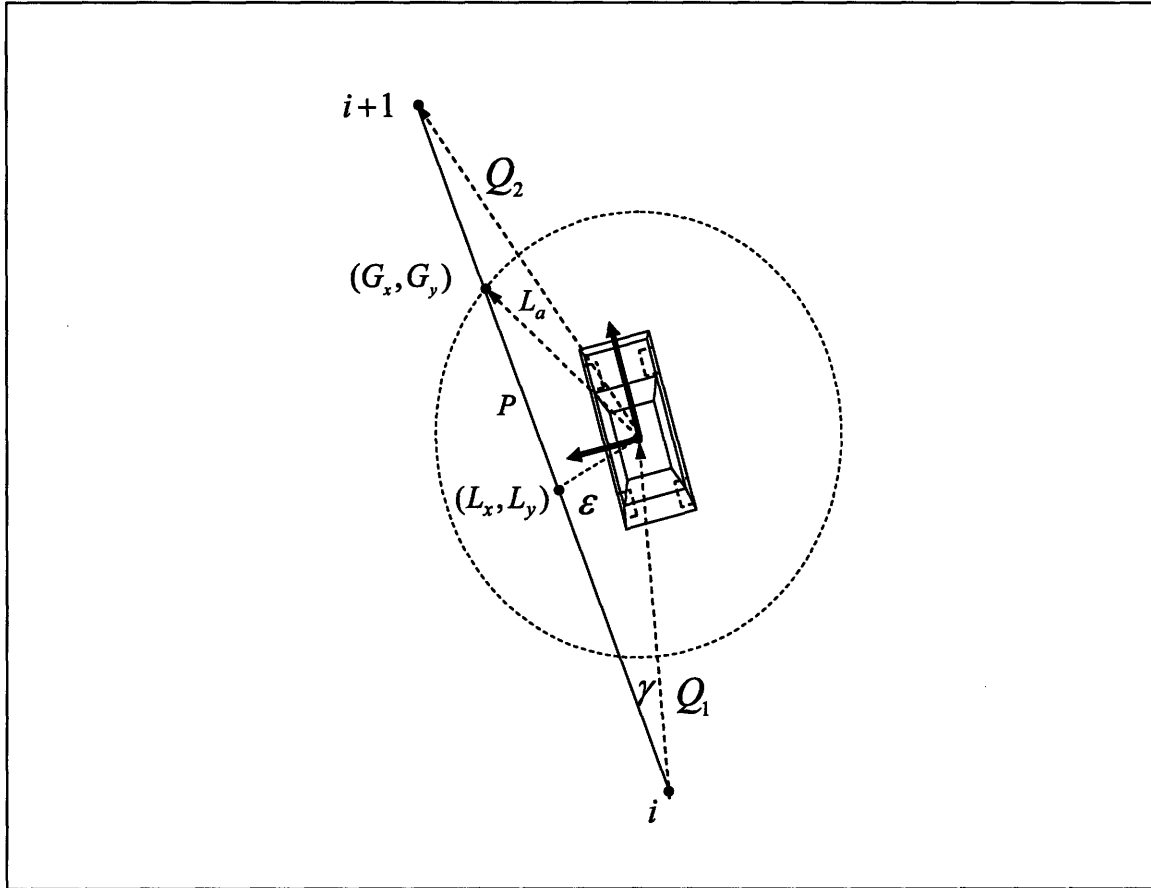


Figure 2.13: Goal Point Determination for Large Path Segments

iv. No Intersection Points

Instances in which the virtual circle no longer intersects the relevant path segment must now be considered. The case considered first is identical to the one illustrated in Figure 2.13, but now the cross-track error is greater than the look-ahead distance (Q_1 and Q_2 are both still assumed greater than L_a). In this instance, the goal point is not clearly defined and a selection must be made such that the pure pursuit controller corrects for this cross-track error in a stable manner. This is an issue that will be addressed in greater detail in section 4.4. For now, the closest point on the path is selected as the goal point. As such the goal point is given as the following:

$$(G_x, G_y) = D_{ap} \left(\frac{i_x}{L_p}, \frac{i_y}{L_p} \right) + (i_x, i_y) \quad (2.15)$$

Here D_{ap} is defined as in equation 2.7.

v. Goal Point Dead Zone

As with the relevancy search, there is an analog to the *dead zone* case in which the relevant line segment is ill-defined and the vehicle's virtual circle does not intersect any path segment. This situation can only arise when the relevancy search finds the vehicle located in a *dead zone*. However, the lack of a clearly defined relevant path segment with regards to the vehicle's position does not guarantee that a goal point is not well-defined. If the cross-track error (which in the dead zone case is the distance to the point $i+1$, where i is the enumerator for the relevant segment) is less than the look-ahead distance, then the common case will apply for a proceeding line segment. This condition ensures that the $i+1$ -th path segment intersects the virtual circle. If however, the cross-track error, Q_1 , and Q_2 (as measured from the relevant path segment) are all greater than L_a , then equation 2.15 is used to define the goal point. Thus the goal point will remain the point $i+1$ until the vehicle is no longer in the dead zone or the virtual circle intersects the path again. This case will be referred to as the *goal point dead zone* case so that the correlation with the dead zone relevancy check is clear. A graphical illustration of this case is presented in Figure 2.14.

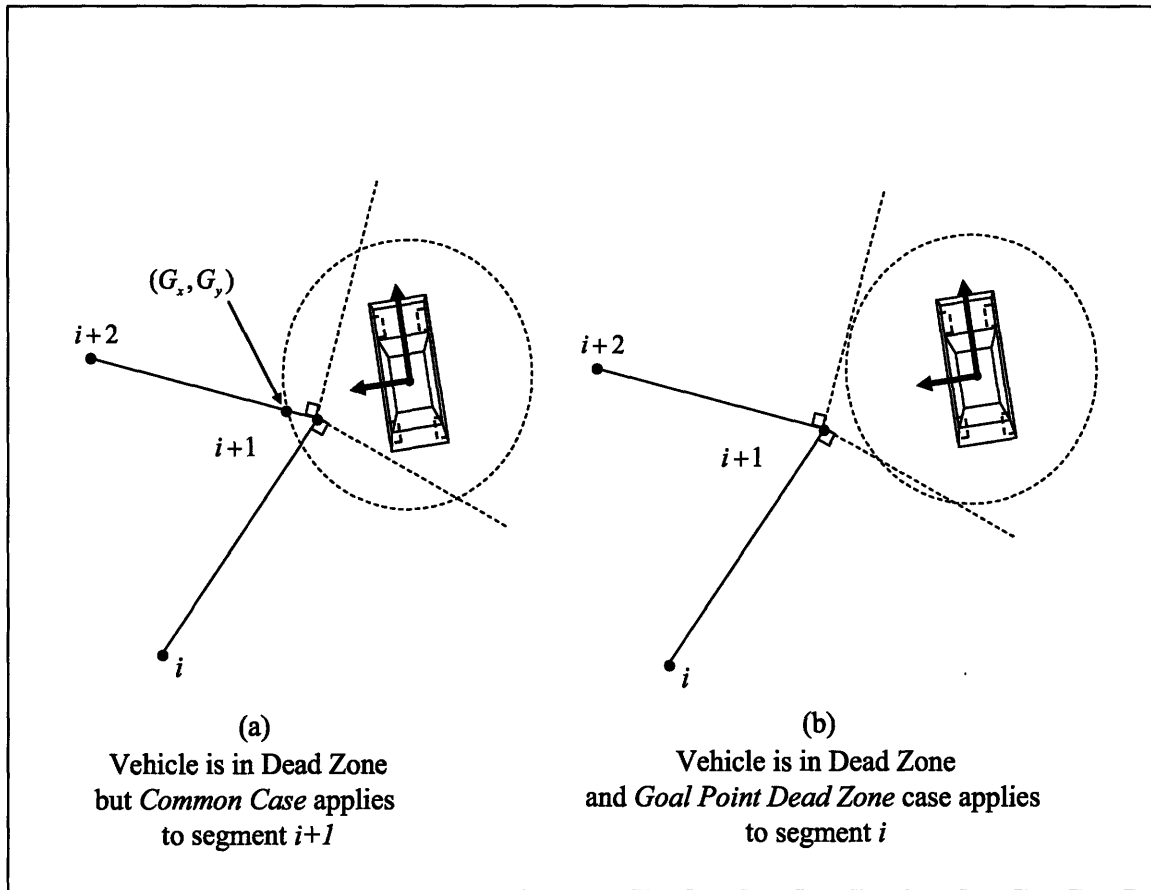


Figure 2.14: Goal Point Dead Zone

As illustrated in Figure 2.13, multiple intersections of the virtual circle can and will occur. The manner in which the goal point is defined in the above cases, however, ensures that the goal point is always the intersection point that is the furthest along the path segment in the direction of travel.

In summary, this search procedure first checks the relevant line segment for all cases. If none of these cases is verified on the relevant line segment, then the search algorithm proceeds to the next line segment. However, at this point, the search algorithm will only consider the common case (*ii*). This is for the reason that cross-track error is only defined for the relevant line segment and cases other than the common case cannot occur outside of the relevant line segment.

A flow diagram of this search is presented in Figure 2.15.

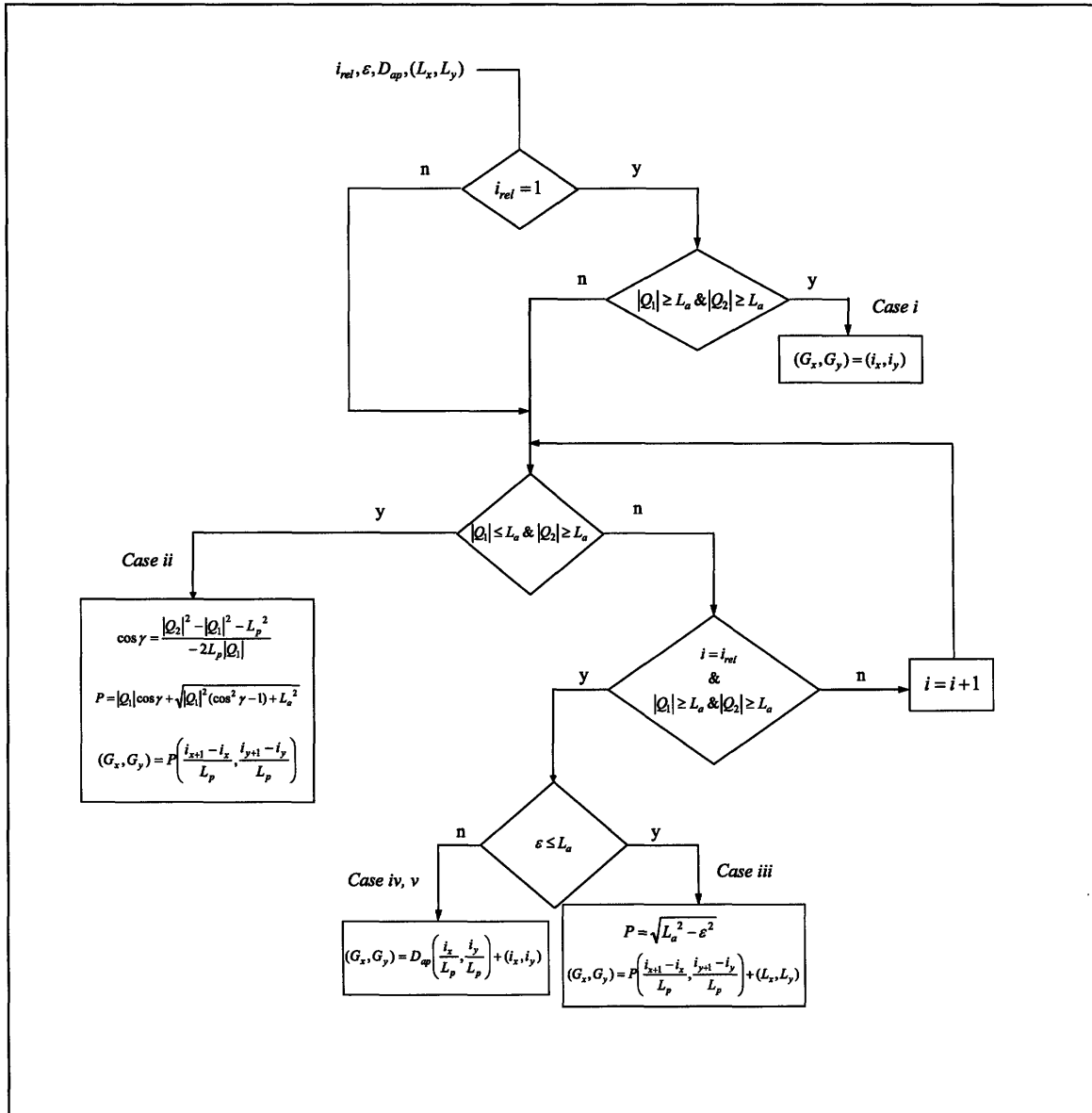


Figure 2.15: Flow Diagram for Goal Point Search

It is noted here that in implementing the search algorithm, cases *iv* and *v* (*no intersection point* and *goal point dead zone* cases respectively) are indistinguishable as a result of the way the relevant path segment is selected when the vehicle is located in a dead-zone. In principle, however, these cases are different and are thus differentiated in this presentation of the search. A table illustrating the cases is presented below in Table 2.2. Here i_r is the index of the relevant path segment.

| Case | Segment | Conditions |
|---|--------------|---|
| First Node Outside Virtual Circle | i_r | $ Q_1 > L_a, Q_2 > L_a, \alpha_1 > \pi/2$ |
| Two Intersection Points | i_r | $ Q_1 > L_a, Q_2 > L_a, \varepsilon \leq L_a, \alpha_1 \leq \pi/2$ |
| No Intersection Points | i_r | $ Q_1 > L_a, Q_2 > L_a, \varepsilon > L_a, \alpha_2 \leq \pi/2$ |
| Goal Point Dead Zone | i_r | $ Q_1 > L_a, Q_2 > L_a, \varepsilon > L_a, \alpha_2 > \pi/2$ |
| One Intersection Point (Common Case) | $i \geq i_r$ | $ Q_1 > L_a, Q_2 \leq L_a$ |

Table 2.2: Cases and Conditions for Goal Point Search

In searching for a goal point, consideration must be given to how to define a goal point as the path comes to an end. As the last node enters the virtual circle surrounding the car, the goal point search identifies this situation and extends the last path segment to infinity. Such a situation is identified by checking the magnitude of Q_2 for the last path segment, assuming it is the relevant path segment. This approach ensures that a goal point is always defined by equations 2.13 and 2.14. This is analogous to the approach taken in the relevancy search.

An alternative approach to the one presented here is to choose a step size ΔL and move sequentially along the path [15]. At every step the distance from the vehicle to the current point on the path is calculated. When this distance goes from being less than to greater than the value of L_a , the intersection point has been passed. The location at which this change occurs can then be used to approximate the location of the goal point. To initiate this search one could use the vehicle's projected location onto the path. This

approach suffers from two limitations: the precision at which the goal point is known is dictated by selecting an arbitrary step size and there is no clear way to handle the *no intersection point* case. Moreover, when a larger look-ahead distance is used or when a complicated path is involved, this process could be quite slow.

2.5 Alternative Controllers

The algorithms presented here are applicable to alternative vehicle controllers, not just the pure pursuit controller. As alluded to in Chapter 1, many control algorithms use a fixed look-ahead distance, as in Figure 1.6. To calculate the cross-track error for this look-ahead distance, the relevancy search can be implemented with the vehicle's position (x_p, y_p) adjusted to the look-ahead location (L_{ax}, L_{ay}) . This conversion is easily done using the look-ahead distance (L_a) and the vehicle's orientation (ψ) in a global coordinate frame, as in equation 2.16.

$$(L_{ax}, L_{ay}) = (x_p + L_a \cos(\psi), y_p + L_a \sin(\psi)) \quad (2.16)$$

The relevancy search is also applicable to any of the control schemes discussed in Chapter 1. These approaches all rely upon cross-track information which can be easily extracted from this search algorithm. Additionally, the Stanford controller also uses knowledge of the path tangent at the vehicle's projected location (L_x, L_y) . This is easily approximated once the relevant line segment is known.

As a result of adopting the approach presented here, a control scheme which is not tightly coupled to the planner's choice of path representation can be easily developed. As a result of this, control laws and planning algorithms may be changed with little to no alterations made to these core search algorithms.

2.6 Chapter Conclusions

This chapter explored three major concepts: how to represent a path, how to search for the vehicle's position along the path, and how to search for the goal point that will be used by the pure pursuit controller. In general, these search algorithms are entirely geometric; having a list of nodes defining a path and the vehicle's position facilitates all the requisite calculations. As mentioned briefly in section 2.4, alternative search algorithms are applicable and may prove slightly more computationally efficient with some associated cost in accuracy. However, the implementation of the algorithms presented here is by no means computationally intensive, especially considering commonly available computational resources. As such, the precision of these algorithms in determining the cross-track error, the vehicle's location on the path, and the location of the goal point is predicated only on the degree to which the path is discretized and how accurately the vehicle's true location is known. This is the primary motivation for implementing the above approach.

As briefly mentioned in section 2.1, the vehicle's external world is treated as a two dimensional plane. Additional complexity arises if the previous algorithms are to be extended to three dimensional paths (each node would have an elevation component). Though such a problem would be more complex, it is fundamentally possible to implement a similar geometric solution.

3 System Modeling

This chapter will present some of the fundamental techniques used to model the dynamics of a traditional automobile. Specifically, kinematic, dynamic, and steady-state models are presented for both forward and reverse driving. The utility of each of these models will be demonstrated in subsequent chapters. In presenting this material, an effort is made to note the key assumptions underlying each of these models. Indeed, more complex models which rely upon fewer assumptions do exist and can be found in [9] and [27]. In addition to presenting these models, the results of simple system identification experiments are presented for *Talos-I* and *Talos-II*.

3.1 Ackerman Steering

The fundamental basis for kinematic models is the *Ackerman steering geometry* assumption. Referring to Figure 3.1, this is the assumption that during a turn each of a vehicle's tires traverses an arc path with a common center. This implies that each tire is in pure rolling with zero lateral acceleration. For such a steering geometry, the relationship between the angle of the inside and outside tire is given by [33]:

$$\cot \delta_o - \cot \delta_i = \frac{L_{tw}}{L_b} \quad (3.1)$$

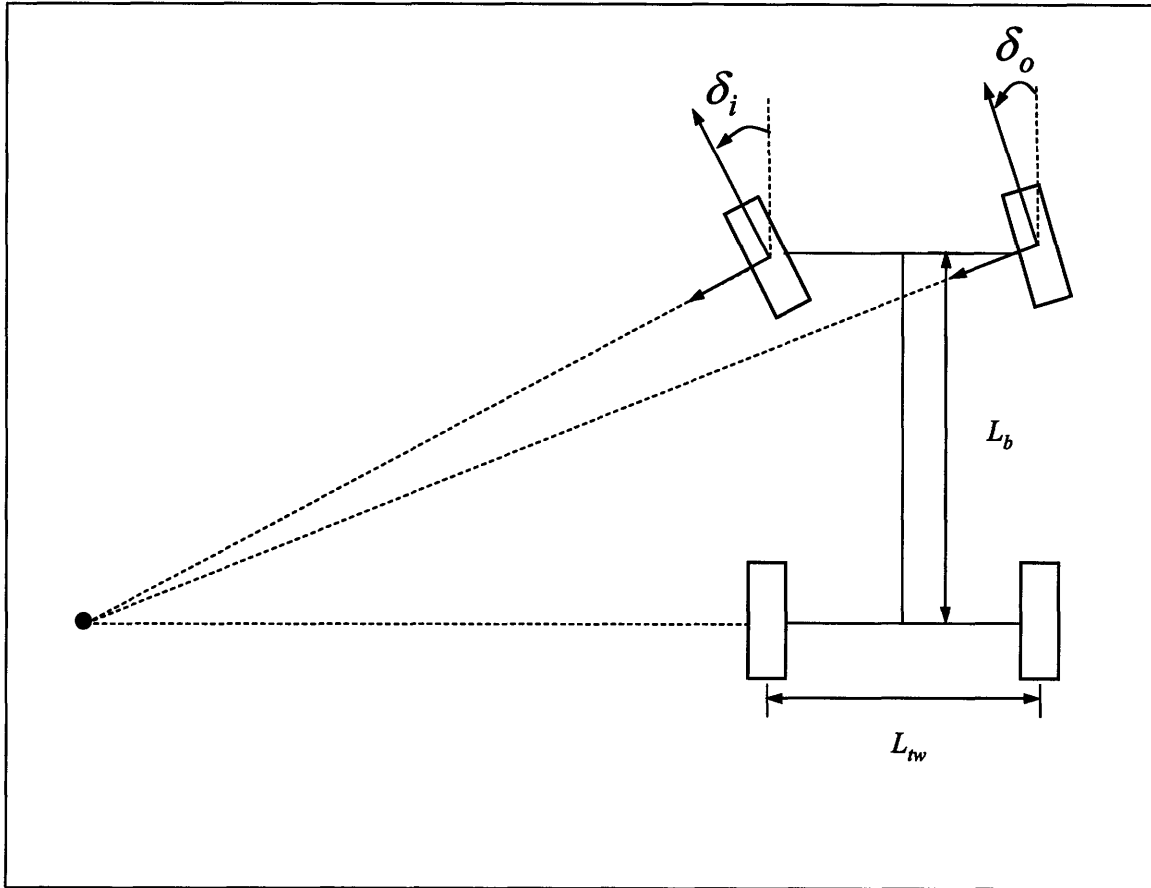


Figure 3.1: Ackerman Steering Geometry

On most cars, this idealization of steering geometry is a reasonable predictor of the relationship between the inside and outside wheel angles. If a vehicle deviates largely from Ackerman steering, then the tires will have excess scrub, or lateral sliding during the negotiation of a turn

3.2 Single-Track Kinematic Model

Many of the equations of motion developed from this point forward will be based upon a simplification in which both the front and rear wheels of a vehicle are lumped together to form a single front and a single rear tire. This simplification is often referred to as a *single-track model* or a *bicycle model*; the latter of these names belies the utility of this approach. As aforementioned, one can find more complicated models which include

roll and pitch dynamics. Even in these works, however, more advanced models are used only for simulation; the single-track model is still used for analysis and controller development.

Referring to Figure 3.2, a body fixed coordinate system is affixed to the vehicle's rear axle. This is done so that the origin of this frame has a velocity purely in the x direction, as a result of the Ackerman steering assumption. The vehicle's yaw and yaw rate are then defined by the angle between the positive x_b axis and the X axis of a fixed inertial system. The vehicle's yaw rate can be related to the front wheel angle (δ) and the velocity of the rear axle (V) by the following:

$$\dot{\psi} = \frac{V \tan \delta}{L_b} \quad (3.2)$$

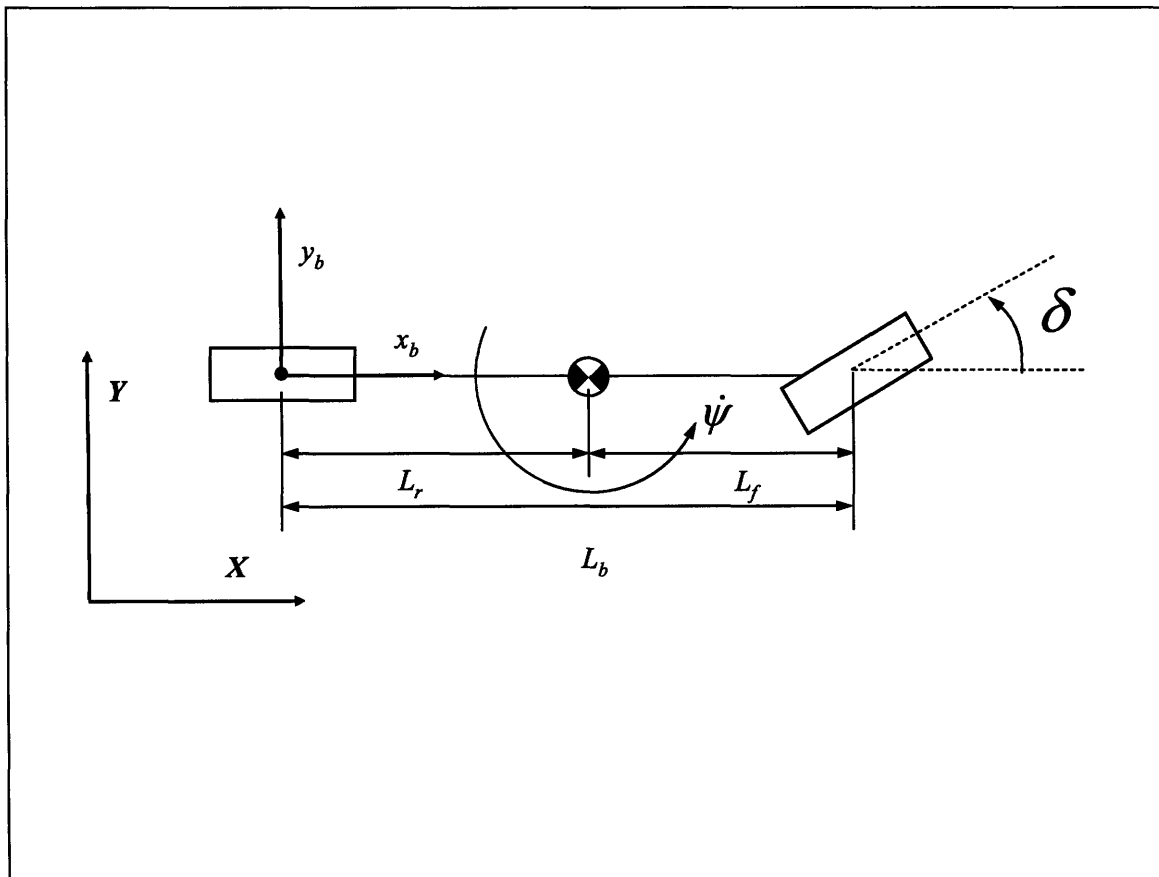


Figure 3.2: Kinematic Single-Track Model

The translation of the body fixed frame origin, as represented in the inertial frame XYZ, is then given by:

$$\begin{aligned}\dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi\end{aligned}\tag{3.3}$$

Equations 3.2 and 3.3 then comprise a basic kinematic model for the system. A linearized state space model of the system is the following:

$$\begin{bmatrix} \dot{\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ V & 0 \end{bmatrix} \begin{bmatrix} \psi \\ y \end{bmatrix} + \begin{bmatrix} V \\ \frac{V}{L_b} \\ 0 \end{bmatrix} \delta\tag{3.4}$$

For a kinematic model of reverse, the body fixed frame is rotated 180° such that the velocity of the rear axle is still positive along the x axis and the z axis remains out of the page. The resulting equations are then the following:

$$\begin{aligned}\dot{\psi} &= \frac{-V \tan \delta}{L} \\ \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi\end{aligned}\tag{3.5}$$

It is not necessary, however, to rely upon a single-track model when a kinematic or Ackerman based approach to modeling is taken. By using a *double track* model (as illustrated in Figure 3.1), equation 3.2 may be rewritten in terms of the inside steer angle (δ_i) as the following:

$$\dot{\psi} = \frac{2V \tan \delta_i}{2L_b + L_{tw} \tan |\delta_i|}\tag{3.6}$$

equations 3.3 would then remain unchanged. As with the single-track model this is trivially extended to driving in reverse. The purpose for introducing a model that uses the inside steering angle will be made clear in the next section.

3.3 Steering Map

Do to the simplification of the single-track model, it is necessary to relate δ of the model to the vehicle's actual steering linkage geometry. In general it is common to take δ as the average of the inside and outside wheel angles, δ_i and δ_o respectively, as in equation 3.7.

$$\delta = \frac{\delta_i + \delta_o}{2} \quad (3.7)$$

In practice, a rack and pinion steering linkage, which is common on many vehicles, has a fixed steering ratio K_{steer} . This value usually relates the angular displacement of the inside turning wheel to the angular displacement of the steering wheel (δ_w). This is simply:

$$\delta_w = K_{steer} \delta_i \quad (3.8)$$

A non-trivial problem lies in converting from the average steer angle to the inside steering angle and then to the steering wheel position. Combining 3.7 and 3.1 and rearranging terms yields equation 3.9. For simplicity, terms involving cotangent (as a result of equation 3.1) have been removed through the application of trigonometric identities.

$$\delta = \frac{\frac{\pi}{2} - \tan^{-1}\left(\frac{L_{tw}}{L_b} + \tan\left(\frac{\pi}{2} - \delta_i\right)\right) + \delta_i}{2} \quad (3.9)$$

To map from the average steer angle to the inside steer angle, a table relating the two angles is created using equation 3.9. Then, given an average steer angle, linear interpolation of this table is used to estimate the correct inside steer angle. This is a reasonable approach as the inside steer angle has a limited range of values, usually on the order of 0° to 30° . Generating such a table, linearly interpolating, and then assigning the

correct sign to the angle is computationally straightforward. In practice, a large table with many data pairs is desirable as it will provide greater accuracy. Searching through such a table sequentially, however, can be slow and time consuming. To combat this, a traditional binary search algorithm should be employed. Moreover, in using this map on a controller, it was observed that the controller output (the desired average steer angle) did not generally change significantly between controller updates (controller updated at 25 Hz for Team MIT). As a result, maintaining a counter for where in the table the last steering angle was found and then quickly checking this point in the table and the preceding and proceeding points often yielded the correct location, obviating the initialization of the binary search. Once the correct inside steering angle is found, it is then related to the actual steering wheel angle through equation 3.8.

3.4 Dynamic Models

Using the single-track simplification introduced in section 3.2, dynamic models for both forward and reverse driving can be formulated. Steady-state models for cornering performance then naturally follow from these dynamic models. Before these models are introduced, some basic concepts will be briefly explained. Here the interested reader is referred to [10] for greater information on these concepts.

Fundamental to developing a dynamic vehicle model is a model of how a tire develops the lateral forces required for turning. When a force is applied to the center of a wheel, the velocity of the tire center will be at an angle α with respect to the vertical plane of the tire. This is a result of the deflection of the tire's wheel center with respect to the actual contact patch. This angle is referred to as the *slip angle*; this name is given despite the fact that the wheel's contact patch is not necessarily slipping or sliding with

respect to the ground. For small slip angles, the lateral force generated by the tire is proportional to the slip angle. This is represented below with the proportional constant C , which is termed the tire's cornering stiffness.

$$F = C\alpha \quad (3.10)$$

A second fundamental concept is *vehicle side-slip*. Referring to Figure 3.1, if a vehicle enters a turn with all four tires in pure rolling, the velocity of the vehicle's c.g. will be at some angle β with respect to the vehicle's longitudinal axis. This angle is referred to as the side-slip angle, once again, despite the fact that all four tires are may be in pure rolling. This illustrates the concept of side-slip, but it should be noted that dynamic models will not assume Ackerman steering. As such, the velocity of the front and rear wheels will not be aligned with the vertical plane of the tire. As discussed in the previous paragraph, this is necessary for any vehicle to negotiate a turn. In essence, a car has to deviate from perfect Ackerman steering, at least in small part, in order to turn.

These fundamental concepts are used in the derivation of a linear, dynamic vehicle model for both forward and reverse driving. Because the basic approach for developing both models is identical, only the reverse model will be explicitly derived as it is often the case in literature that the reverse model is entirely ignored.

3.4.1 Linear Reverse Driving Model

Before deriving the model, several assumptions should be explicitly stated. First, the vehicle's roll and pitch dynamics will be neglected. Second, the vehicle has been assumed symmetric about its longitudinal axis. Finally, the cornering stiffnesses for the front and rear tires have been assumed identical.

With the above assumptions in hand, the free body diagram with the appropriate coordinate system is presented in Figure 3.3. For the sake of clarity, the location of the fixed wheel is still referenced as the rear of the vehicle and the location of the steerable wheel is still referenced as the front of the vehicle. Additionally, the distance between the c.g. and the rear wheel is l_r and the distance from the front of the vehicle to the c.g. is the distance l_f , as is illustrated in Figure 3.2. Universally, all r subscripts refer to a quantity associated with the rear of the vehicle and all subscripts f refer to a quantity associated with the front of the vehicle.

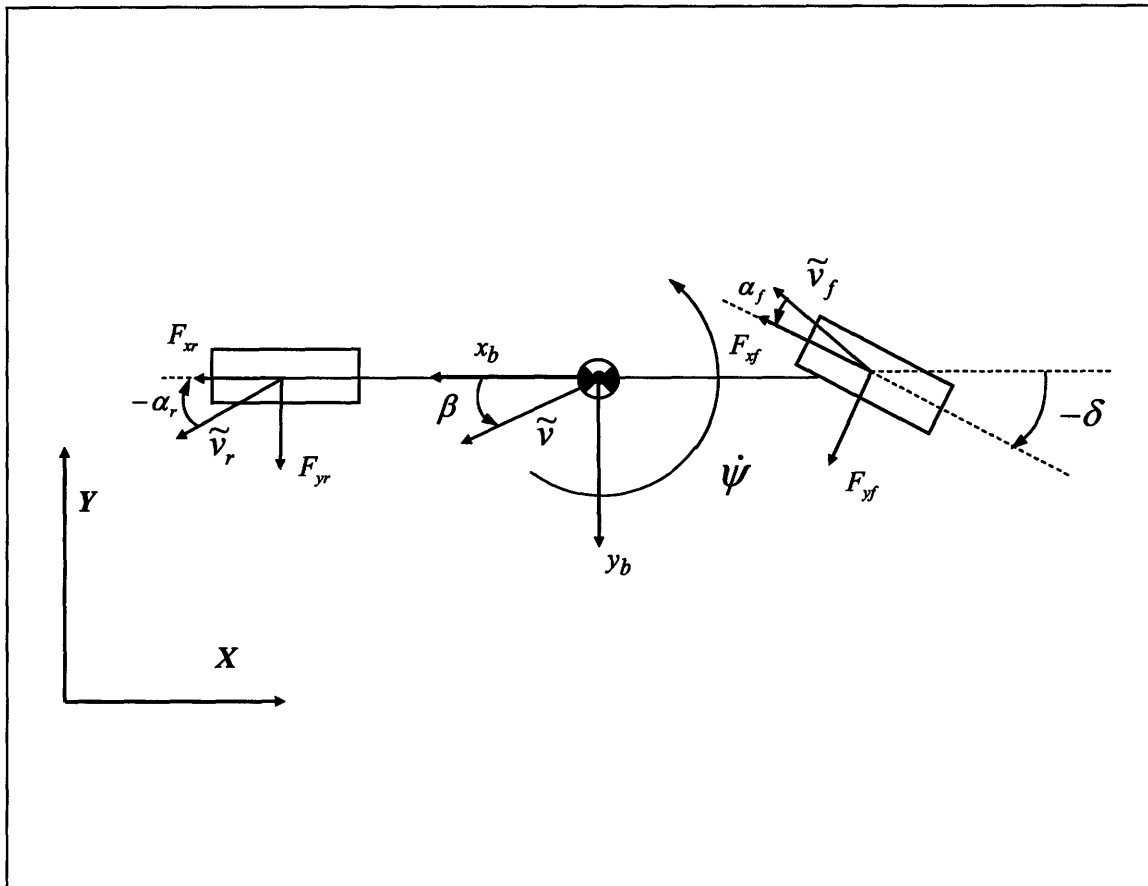


Figure 3.3: Reverse Free Body Diagram

The acceleration of the center of mass with respect to the inertial coordinate system XYZ but represented in the body fixed $x_b y_b z_b$ frame is given by equation 3.11 [17].

$$\frac{d\tilde{v}}{dt} \Big|_{XYZ} = \frac{d\tilde{v}}{dt} \Big|_{xyz} + \dot{\psi} \times \tilde{v} \quad (3.11)$$

Application of equation 3.11 yields the following acceleration for the vehicle's center of mass:

$$\begin{aligned} a_x &= (\dot{v}_x - v_y \dot{\psi}) \\ a_y &= (\dot{v}_y + v_x \dot{\psi}) \end{aligned} \quad (3.12)$$

For convenience, the acceleration (a) and velocity vectors of the c.g. have been separated into their x and y components respectively. Using 3.12, the sum of the forces on the vehicle is given by equation 3.13:

$$\begin{aligned} m(\dot{v}_x - v_y \dot{\psi}) &= F_{xr} - F_{yf} \sin \delta + F_{xf} \cos \delta_f \\ m(\dot{v}_y + v_x \dot{\psi}) &= F_{yr} + F_{yf} \cos \delta + F_{xf} \sin \delta \end{aligned} \quad (3.13)$$

Three simplifying assumptions are now made. First it is assumed that the traction forces (F_{xr} and F_{xf}) are such that the magnitude of the vehicle's velocity (speed) remains constant. This implies that these forces balance all external forces which have been neglected. As a result, these terms are removed from equations 3.13. Second, the steering angle is assumed small. Finally, the side-slip angle is also assumed to be small, which results in the following approximations.

$$\begin{aligned} v_x &= |\tilde{v}| \cos \beta \approx |\tilde{v}| \\ v_y &= |\tilde{v}| \sin \beta \approx |\tilde{v}| \beta \end{aligned} \quad (3.14)$$

From this point forward, the vehicle's constant speed will be denoted as v_x . Application of these simplifying assumptions yields equations 3.15.

$$\begin{aligned} m(-v_x \dot{\beta} \dot{\psi}) &= -F_{yf} \delta \\ m(v_x \dot{\beta} + v_x \dot{\psi}) &= F_{yr} + F_{yf} \end{aligned} \quad (3.15)$$

Moments are then summed about the c.g. and the above assumptions are immediately applied to yield:

$$I_z \dot{\psi} = l_r F_{yr} - l_f F_{yf}. \quad (3.16)$$

To complete the model, it is necessary to define the external forces relative to the side-slip angle and cornering stiffness. Fundamentally, dynamic modeling differs from kinematic modeling in that there is no assumption made regarding pure rolling. As a result, the slip angles are given as the following:

$$\begin{aligned} \tan(-\alpha_r) &= \frac{l_r \dot{\psi} + v_x \beta}{v_x} \\ \tan(\delta - \alpha_f) &= \frac{v_x \beta - l_f \dot{\psi}}{v_x} \end{aligned} \quad (3.17)$$

Combining 3.14, 3.15, 3.16 and 3.17 with a small angle assumption for the slip angles yields the linearized model of equation 3.18. Here the states have been chosen as the yaw rate and side-slip angle.

$$\begin{bmatrix} \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{-2C}{mv_x} & -1 + \frac{C(l_f - l_r)}{mv_x^2} \\ \frac{C(l_f - l_r)}{I_z} & \frac{-C(l_r^2 + l_f^2)}{I_z v_x} \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C}{mv_x} \\ -\frac{Cl_f}{I_z} \end{bmatrix} \delta \quad (3.18)$$

For the remainder of this text, the individual terms of equation 3.18 will be referenced as illustrated below:

$$\begin{bmatrix} \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} a_{R11} & a_{R12} \\ a_{R21} & a_{R22} \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} b_{R11} \\ b_{R21} \end{bmatrix} \delta \quad (3.19)$$

For steady-state vehicle performance, the derivatives of side-slip and yaw rate are set to zero and the yaw rate is assumed to be v_x/R , where R is the radius of curvature for a maneuver. The resulting equation is given below:

$$\delta = \frac{(l_r - l_f)mv_x^2}{CRL_b} - \frac{L_b}{R} \quad (3.20)$$

This equation is often rewritten as the following [33]:

$$\delta = K_{us} \frac{v_x^2}{gR} - \frac{L_b}{R} \quad (3.21)$$

Here K_{us} is termed the understeer coefficient and serves as a measure of the vehicle's cornering performance. This parameter will be explained in section 3.4.2 in terms of forward driving.

3.4.2 Linear Forward Driving Model

A linear model for forward driving is presented in equation 3.22. A free body diagram for this model is presented in Figure 3.4.

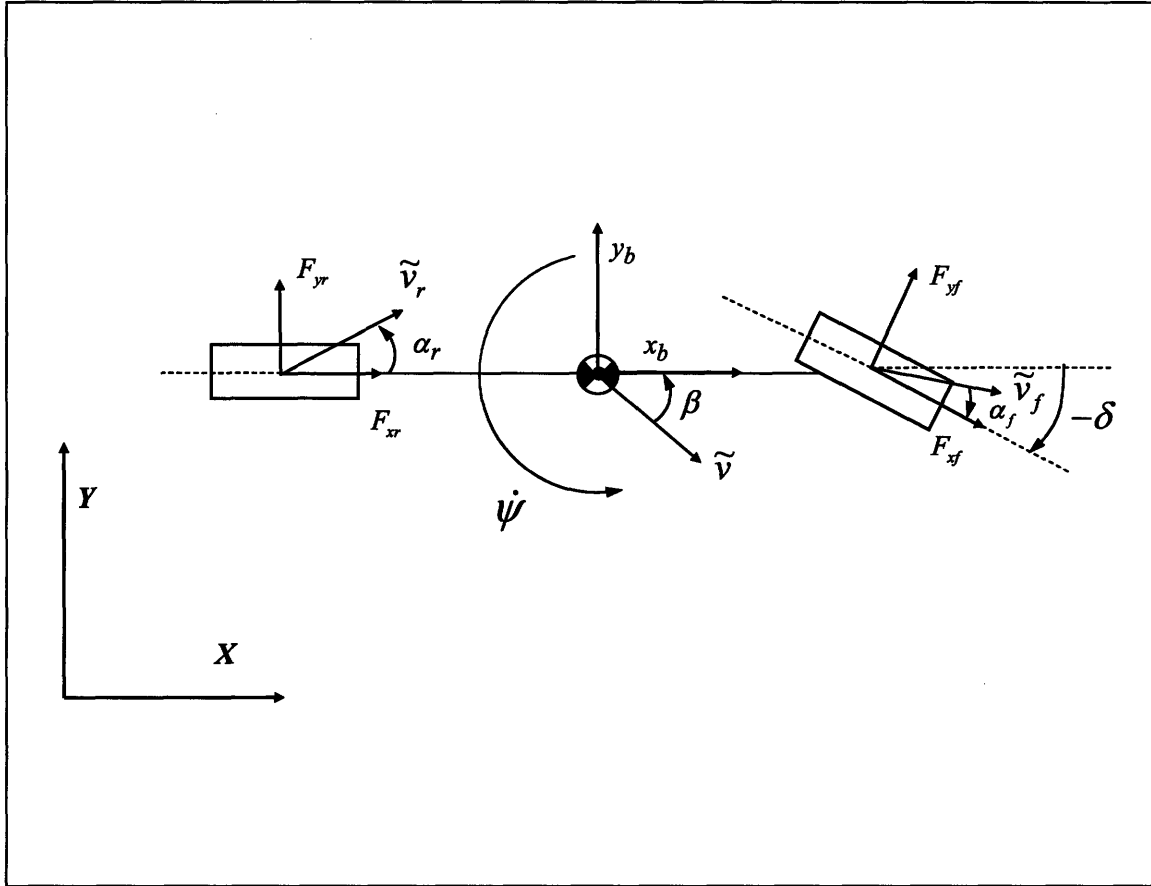


Figure 3.4: Free-Body Diagram for Forward Driving

$$\begin{bmatrix} \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{-2C}{mv_x} & -1 + \frac{C(l_r - l_f)}{mv_x^2} \\ \frac{C(l_r - l_f)}{I_z} & \frac{-C(l_r^2 + l_f^2)}{I_z v_x} \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C}{mv_x} \\ \frac{Cl_f}{I_z} \end{bmatrix} \delta \quad (3.22)$$

As before, this version of the model assumes that the front and rear wheels have an identical cornering stiffness, that roll and pitch dynamics are negligible, and that the vehicle is symmetric about its longitudinal axis. As with the reverse model, the terms defining the component matrices of equation 3.22 will be referenced as illustrated below:

$$\begin{bmatrix} \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} a_{F11} & a_{F12} \\ a_{F21} & a_{F22} \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} b_{F11} \\ b_{F21} \end{bmatrix} \delta \quad (3.23)$$

Equations describing steady-state cornering can then be simply derived, as with the reverse model, by setting $\dot{\beta}$ and $\ddot{\psi}$ equal to zero and defining the yaw rate as v_x / R . The resulting equation is the following:

$$\delta = \frac{L_b}{R} + \frac{(l_r - l_f)mv_x^2}{CRL_b} \quad (3.24)$$

The understeer coefficient of section 3.4.1 is immediately recognizable. The concept of understeer and oversteer is now introduced in terms of forward driving. When K_{us} is positive, the steering angle required to negotiate a turn of radius R increases with the square of the vehicle's velocity. This is termed understeer. Conversely, when K_{us} is negative, the steering angle required to negotiate the same turn decreases with the square of velocity. This is termed oversteer. If K_{us} is zero, then the steering angle required to negotiate a turn is not velocity dependent. This is termed *neutral steer* and is the case when a vehicle adheres very closely to the Ackerman steering assumption. In general this coefficient is an important component for determining the driving characteristics of a vehicle.

To help summarize the many vehicle models presented in this chapter, Table 3.1 is presented below. Here it is illustrated how the steady-state models may be used to help simulate vehicle performance by providing an estimate of vehicle yaw rate.

| Model Name | Model For Vehicle Yaw | Model For Vehicle Position |
|-------------------------------------|--|--|
| Single-Track Kinematic, Forward | $\dot{\psi} = \frac{V \tan \delta}{L_b}$ | $\begin{aligned} \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned}$ |
| Double-Track Kinematic, Forward | $\dot{\psi} = \frac{2V \tan \delta_i}{2L_b + L_{tw} \tan \delta_i }$ | $\begin{aligned} \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned}$ |
| Single-Track Kinematic, Reverse | $\dot{\psi} = \frac{-V \tan \delta}{L_b}$ | $\begin{aligned} \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned}$ |
| Double-Track Kinematic, Reverse | $\dot{\psi} = \frac{-2V \tan \delta_i}{2L_b + L_{tw} \tan \delta_i }$ | $\begin{aligned} \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned}$ |
| Single-Track Dynamic, Forward | $\begin{bmatrix} \dot{\beta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} a_{F11} & a_{F12} \\ a_{F21} & a_{F22} \end{bmatrix} \begin{bmatrix} \beta \\ \psi \end{bmatrix} + \begin{bmatrix} b_{F11} \\ b_{F21} \end{bmatrix} \delta$ | $\begin{aligned} \dot{X} &= V \cos(\psi + \beta) \\ \dot{Y} &= V \sin(\psi + \beta) \end{aligned}$ |
| Single-Track Dynamic, Reverse | $\begin{bmatrix} \dot{\beta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} a_{R11} & a_{R12} \\ a_{R21} & a_{R22} \end{bmatrix} \begin{bmatrix} \beta \\ \psi \end{bmatrix} + \begin{bmatrix} b_{R11} \\ b_{R21} \end{bmatrix} \delta$ | $\begin{aligned} \dot{X} &= V \cos(\psi + \beta) \\ \dot{Y} &= V \sin(\psi + \beta) \end{aligned}$ |
| Single-Track, Steady-State, Forward | $\dot{\psi} = \frac{\delta v_x g}{(L_b g + K_{us} v_x^2)}$ | $\begin{aligned} \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned}$ |
| Single-Track, Steady-State, Reverse | $\dot{\psi} = \frac{\delta v_x g}{(K_{us} v_x^2 - L_b g)}$ | $\begin{aligned} \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned}$ |

Table 3.1: Vehicle Model Summary

3.5 Identification of Understeer Coefficient

In general, the understeer coefficient is not a constant quantity; it varies with the operating conditions of the vehicle. Moreover, the nature with which this term varies is a function of the vehicle's design. However, the data collected for this research suggests that this value varies only slightly at low speeds, thereby making it reasonable to treat the understeer coefficient as a constant.

System identification was performed to quickly get an estimate of the vehicle's steady-state performance as well as to provide an estimate of the cornering stiffness of

the vehicle's tires. With the latter quantity in hand, the models of equations 3.18 and 3.22 may then be used for analysis purposes. Most of the vehicle's remaining parameters, such as l_r , l_f , m , L_b , and L_{tw} are approximated from documentation provided by the vehicle manufacturer (reference Appendix C for these values). The vehicle's moment of inertia is then approximated by treating the vehicle as two point masses joined by a mass-less rod. This is illustrated graphically in Figure 3.5.

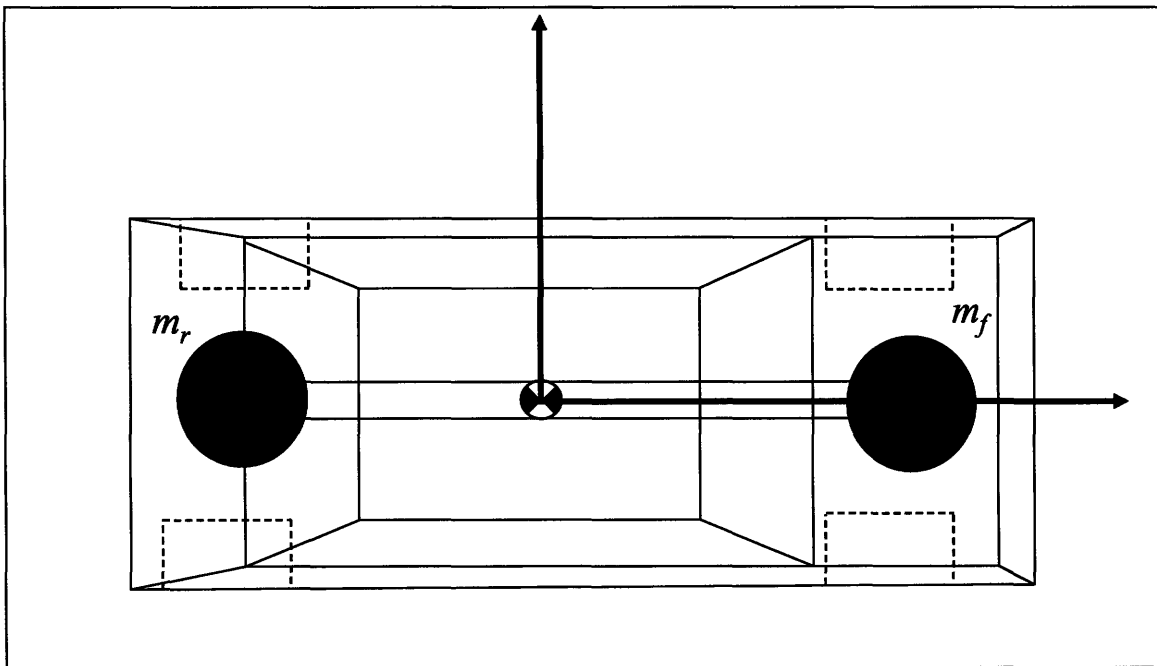


Figure 3.5: Moment of Inertia

The moment of inertia for the vehicle is then given as the following:

$$I_z = l_r l_f (m_r + m_f) \quad (3.25)$$

The understeer coefficient may be calculated in one of several ways. For this particular application, the simplest approach is to perform a constant steer test [33]. In such a test the steering wheel is held at a fixed position and the vehicle is driven in a circle at varying speeds. Taking the differential of equation 3.24 yields the following:

$$d\delta = \frac{K_{us}}{g} d(a_y) + L_b d\left(\frac{1}{R}\right) \quad (3.26)$$

Here this expression has been simplified by substituting in the definition

$$a_y = \frac{v_x^2}{R}. \quad (3.27)$$

Because the steer angle remains constant for this test, the following relationship holds:

$$\frac{d\left(\frac{1}{R}\right)}{d(a_y)} = -\frac{K_{us}}{gL_b} \quad (3.28)$$

Determination of the understeer coefficient is then achieved by measuring the lateral acceleration and path curvature as the vehicle's steering position is held fixed and speed is varied. In this particular application, an inertial measurement unit providing accurate yaw rate information was employed. Additionally, speed information was ascertained from a GPS receiver mounted in the vehicle. For Talos-I, a representative data set is presented below in Figure 3.6.

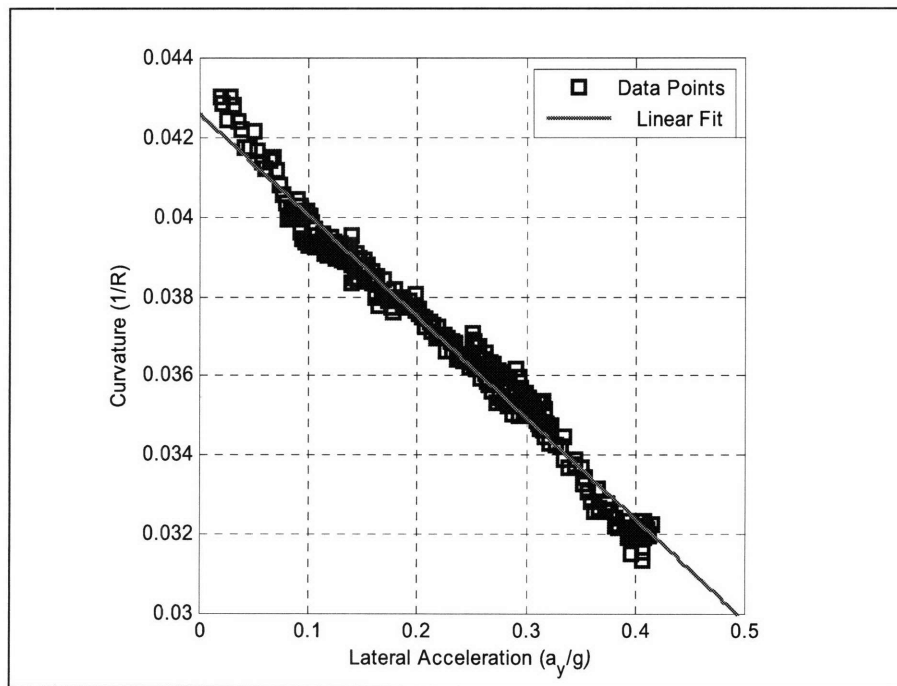


Figure 3.6: Determination of K_{us} for Talos-I

To help illustrate the nature of the experiment, the vehicle's speed and position are presented below in Figure 3.7 and Figure 3.8 respectively.

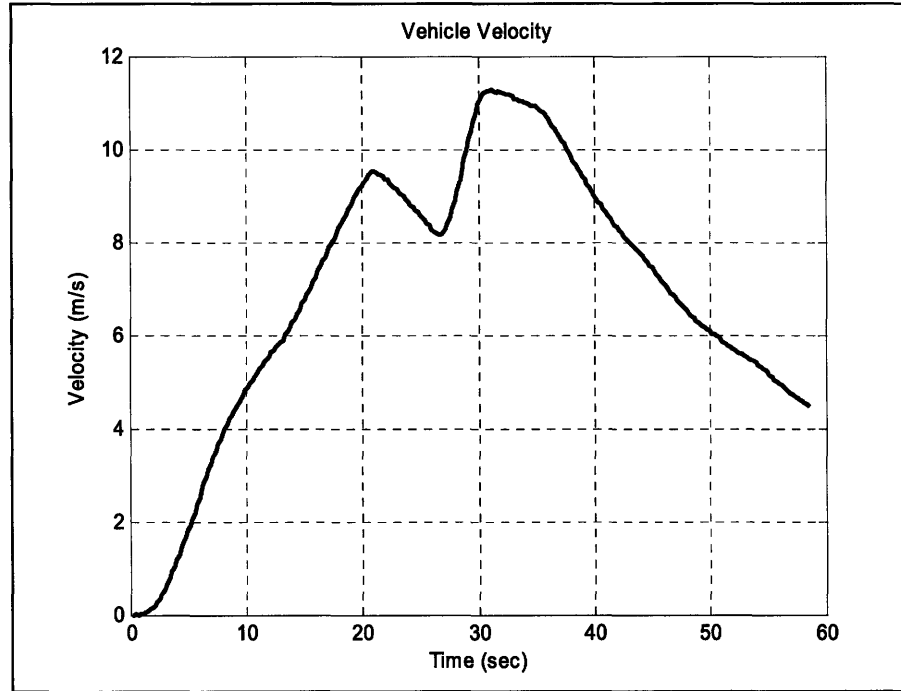


Figure 3.7: Talos- I Velocity Profile for Understeer Experiment

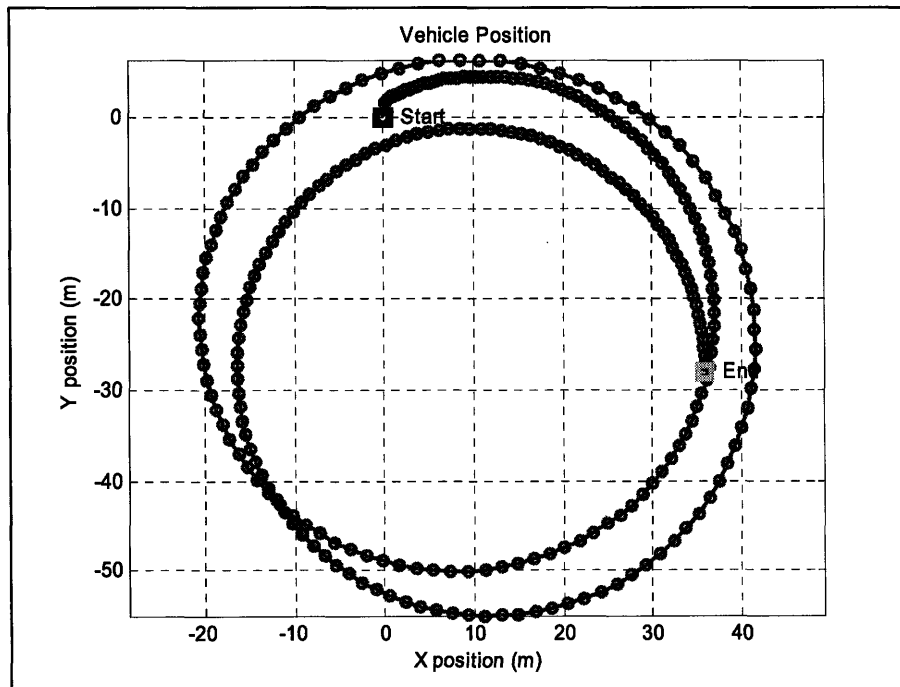


Figure 3.8: Talos-I Position for Understeer Experiment

An identical experiment was performed on Talos-II. The results of that experiment and the associated speed profile and position plots are presented in Figures 3.9, 3.10, and 3.11.

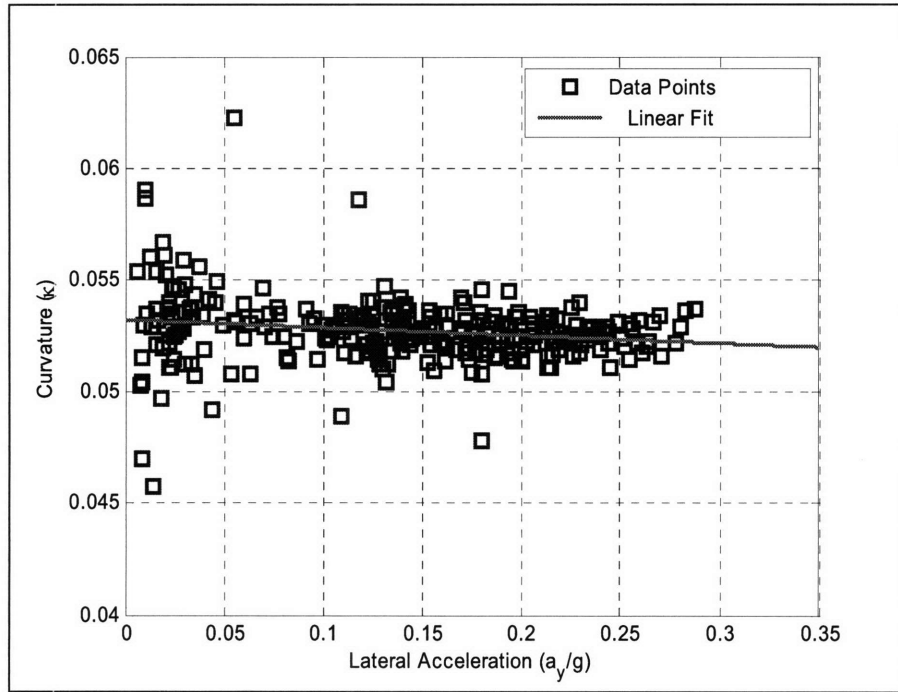


Figure 3.9: Determination of K_{us} for Talos-II

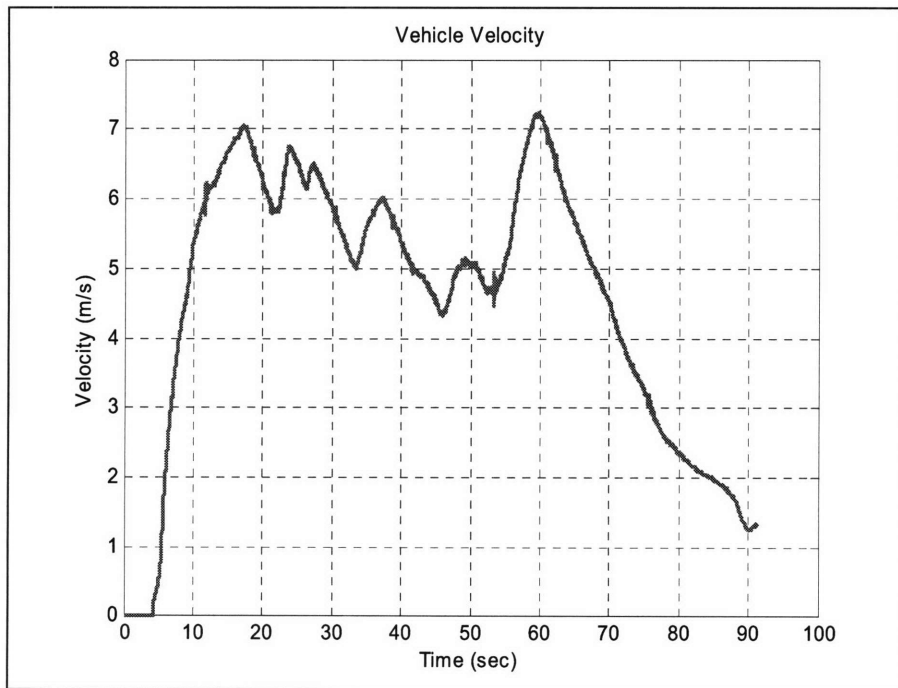


Figure 3.10: Talos-II Velocity Profile for Understeer Experiment

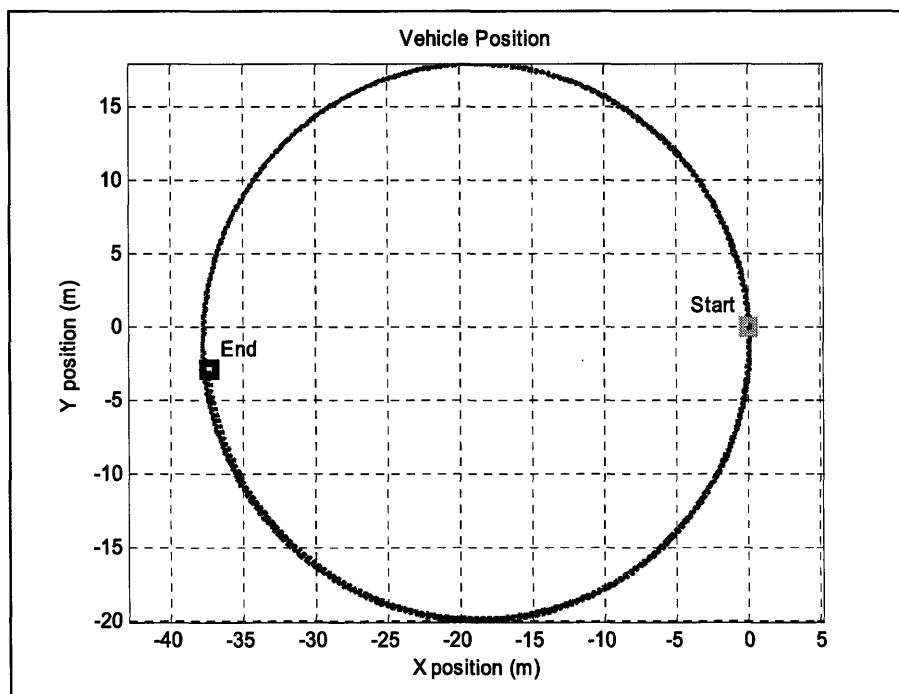


Figure 3.11: Talos-II Position for Understeer Experiment

Based on [33], the expectation is that the relationship between lateral acceleration and curvature would be non-linear (polynomial of degree two). However, as illustrated in Figure 3.6 and Figure 3.9, this relationship appears strongly linear. This result was confirmed by repetition of this experiment at various steer angles. This suggests that the system identification experiments performed excited only a small band of the vehicle's dynamics. Provided this assumption is true, it is then reasonable to assume that performing this test at more aggressive speeds would illustrate the non-linear nature of the relationship between curvature and lateral acceleration. Given safety considerations and the speed limits imposed by DARPA, however, this data is taken as representative of what will be encountered in the DUC. Indeed, because K_{us} varies with operating conditions, this is equivalent to approximating K_{us} for the relevant operating conditions only. For the data shown, the understeer coefficients for Talos-I and Talos-II were found to be 3° and $.34^\circ$ respectively.

3.6 Ackerman Steering vs. Dynamic Model

The steering angle required to negotiate a turn of a given radius for an Ackerman steered vehicle is

$$\delta = \tan^{-1} \frac{L_b}{R}. \quad (3.29)$$

If however, the radius of curvature is measured from the c.g. as opposed to the rear axle, the equation is the following:

$$\delta = \tan^{-1} \frac{L_b}{\sqrt{R^2 - l_r^2}} \quad (3.30)$$

Whether 3.29 or 3.30 is applied, the Ackerman-based prediction of the steering angle required to negotiate a turn of radius R differs from that of equation 3.24. Using the understeer coefficient identified for Talos-I, Figure 3.12 presents steering angle versus curvature for both equation 3.24 and equation 3.29. An identical plot is presented in Figure 3.13 for Talos-II.

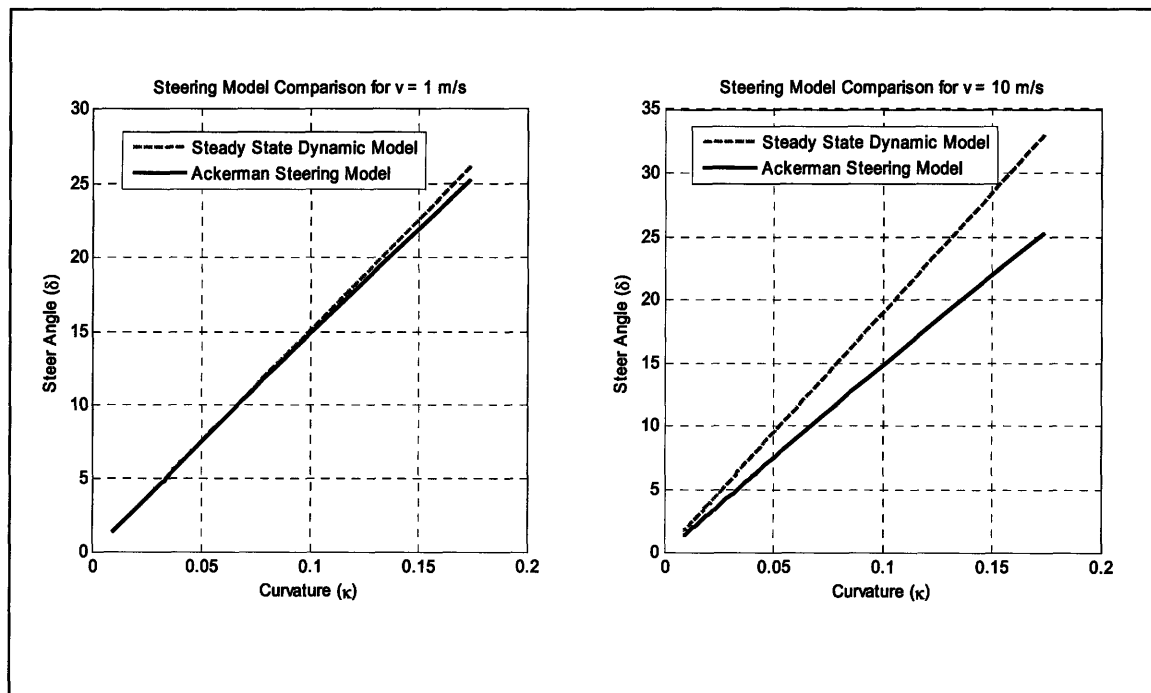


Figure 3.12: Comparison between Ackerman and Dynamics Steering Model for Talos-I

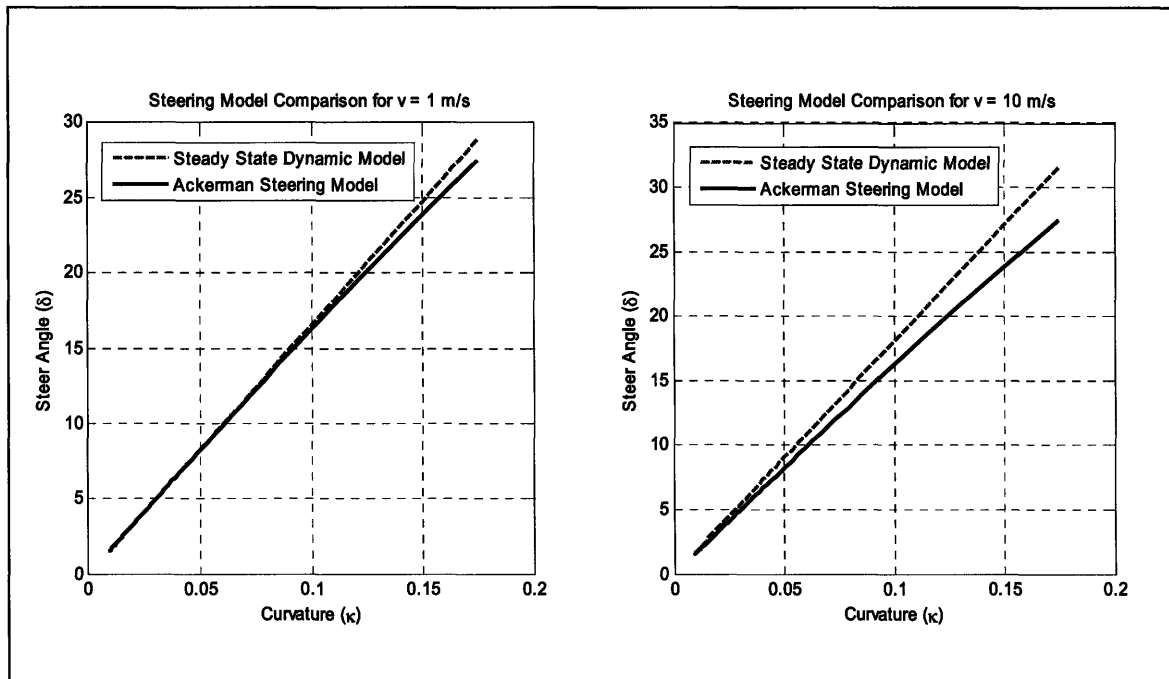


Figure 3.13: Comparison between Ackerman and Dynamic Steering Model for Talos-II

Figure 3.12 and Figure 3.13 illustrate several key points. At low speeds, the steady-state model of equation 3.24 and the Ackerman model of equation 3.29 are nearly identical. This is especially true for turns with a small curvature. The deviation only reaches the order of a single degree near the limit of the vehicle's turning radius. Conversely, at larger speeds, equations 3.24 and 3.29 differ significantly. This is expected as the Ackerman based model has no speed dependence where as equation 3.24 predicts that the steer angle should vary with the square of speed.

For forward driving, the model provided by equation 3.24 will be used to map from a desired curvature to the requisite steering angle. This decision is made for two reasons. First, at low speeds, equation 3.24 closely approximates Ackerman steering. Second, at higher speeds when side-slip dynamics invalidate the Ackerman steering assumption, equation 3.24 accounts for the presence of these increasing dynamics.

3.7 Steering System Model

An important issue in system modeling is the steering mechanism. In general, the steering linkage of the car was treated as a mechanical gearing system in which the relevant parameter is the coefficient K_{steer} , initially presented in equation 3.8. As noted there, this quantity is usually provided by the manufacturer or in standard literature on the vehicle. As a result of this assumption, the dynamics of the actuator driving the steering wheel are of primary concern. In the test vehicles employed in this work, the actuator is a servo mechanism mounted directly to the steering wheel. As is typical with servo mechanisms, it can be modeled as a second order system with the standard transfer function:

$$\frac{\delta_w(s)}{R(s)} = \frac{\omega_n}{s^2 + 2\xi\omega_n + \omega_n^2} \quad (3.31)$$

In practice, the system behaves as a second order system with a maximum achievable angular rate of rotation. To identify the model parameters of 3.31, data was collected for small step inputs (commands requiring less than 30° of steering wheel rotation) and standard approaches for matching both the peak time and maximum percent overshoot [18] were utilized. The resulting parameters were the following for Talos-I:

$$\begin{aligned} \xi &= .458 \\ \omega_n &= 23.27 \text{ rad/sec} \end{aligned}$$

The maximum slew rate (ω_{max}) was found by collecting position data from an encoder mounted to the actuator as the actuator was commanded from its neutral (center) position to either full left or full right. The maximum slew rate for the *Talos-I* steering actuator was found to be 6.17 *rad/sec*. For Talos-II, ζ , ω_n , and ω_{max} were found to be .391, 22.75

rad/sec, and *4.69 rad/sec* respectively. The maximum angular rate at the inside turning wheel is then determined by differentiating equation 3.8. This yields the following:

$$\dot{\delta}_i = \frac{\dot{\delta}_w}{K_{steer}} \quad (3.32)$$

Using equation 3.32, the maximum slew rate of the inside turning wheel was *.34 rad/sec* for Talos-I and *.265 rad/sec* for Talos-II.

Figure 3.14 and Figure 3.15 illustrate the step response of the actuator for Talos-I and the system model subject to identical inputs. Here it should be noted that the actuator is a discrete system in which commanded and actual positions are translated into counts on an encoder. In both figures, the steady-state offset of the actual motor position from the commanded position is a result of the discrete nature of the system. The motor's true position can actually be anywhere between its measured position and the commanded position as these two differ by only 1 encoder reading.

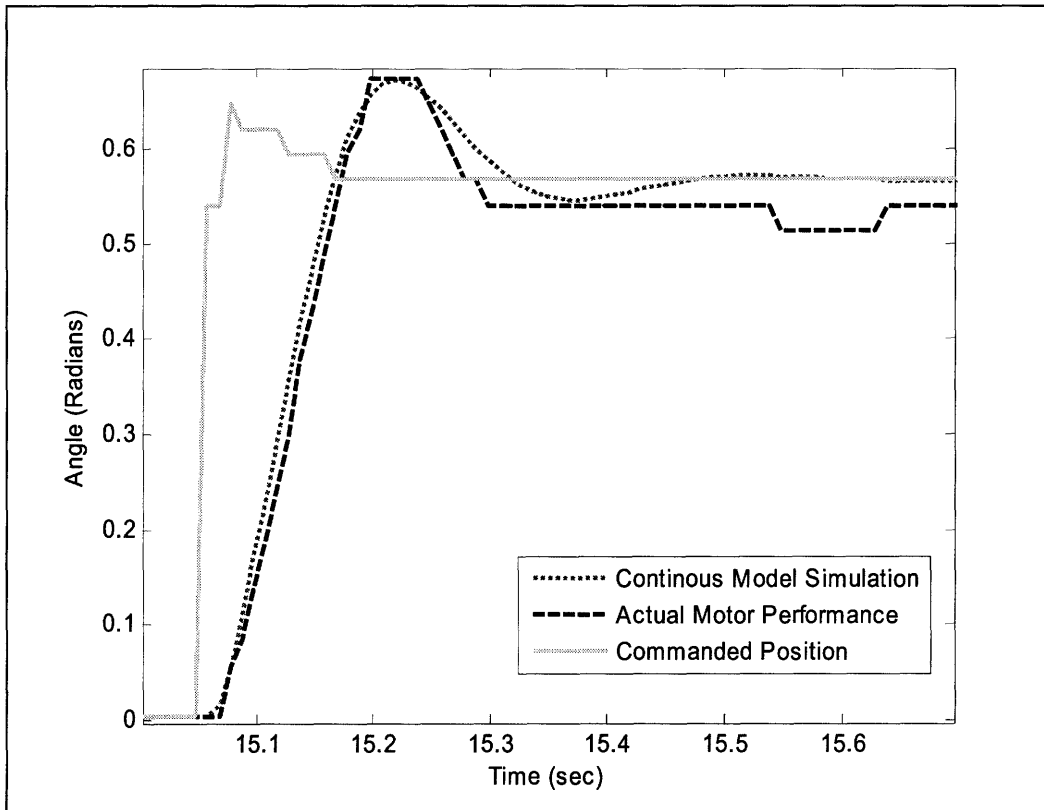


Figure 3.14: Small Step Response of Steering Wheel Actuator

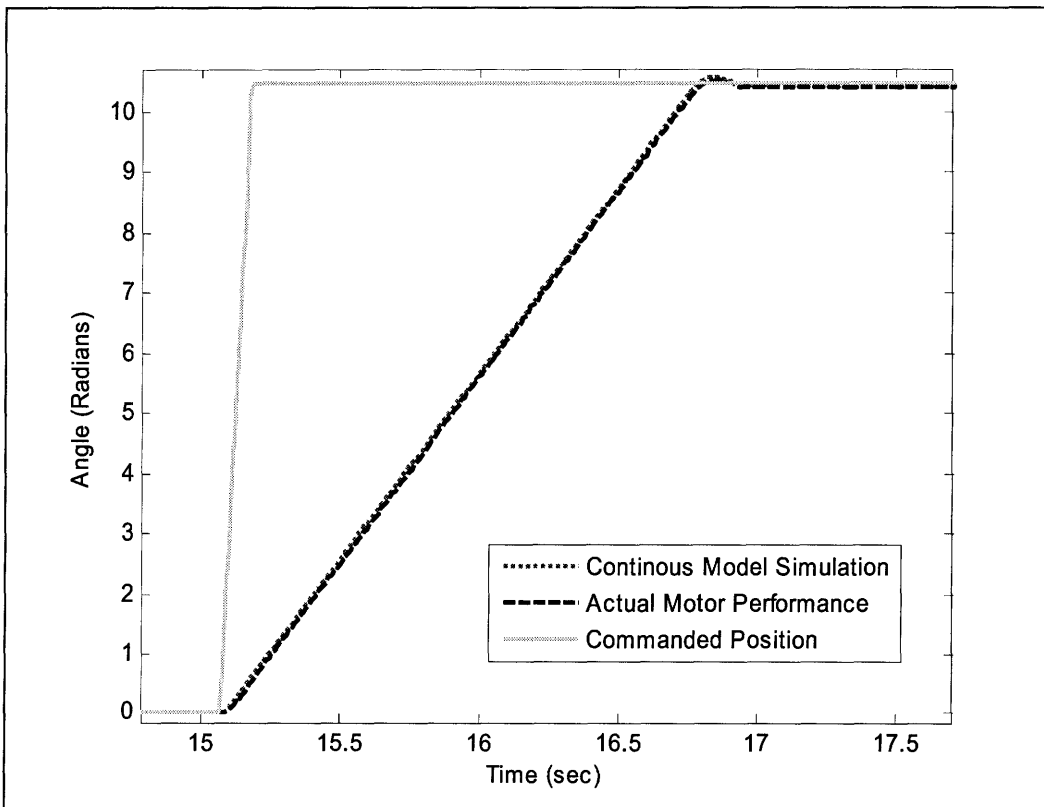


Figure 3.15: Large Step Response of Steering Wheel Actuator

3.8 Chapter Conclusions

This chapter presented some fundamental modeling concepts which will be used throughout the development and analysis of the pure pursuit controller. Specifically, a linear dynamic model for both forward and reverse driving was presented along with steady-state models of performance for both modes of operation. This was contrasted with simple kinematic models in which the Ackerman steering assumption was enforced. A simple comparison of these two approaches was given in section 3.6 and it was argued that the dynamic models are more capable of capturing the vehicle's true performance characteristics.

Additionally, a simple approach to system identification was used to determine the understeer coefficient for the vehicle. The simplicity of this test aligned with Team MIT's goal to develop a control system that required minimal system identification. Indeed, additional chapters will illustrate the usefulness of identifying this one parameter. The vehicle's remaining parameters were then taken from the manufacturer's published literature.

4 The Pure Pursuit Controller

This chapter will introduce the pure pursuit controller for a forward driven vehicle based on two approaches: a model based mapping approach and a simulation/search based approach. In the first, the pure pursuit controller outputs a desired lateral acceleration or path curvature and this is mapped directly to a steering angle. Here analysis will focus on linearizing the controller and developing a method for tuning the pure pursuit look-ahead distance. Moreover, two steering angle maps will be explored: one using a kinematic model and another using a dynamic model. Finally, alternatives to tuning the look-ahead distance that are not amenable to traditional control analysis will be presented. In the second approach, a simple kinematic model is used to generate a space of feasible trajectories based on possible steering inputs. The best trajectory is then selected using a simple cost function. Several simple search methods are described for finding the best trajectory and a basic simulation is used to demonstrate the utility of this approach.

4.1 Basic Pure Pursuit

The basic pure pursuit algorithm is introduced graphically in Figure 4.1. This control approach has a single tunable parameter L_a , which defines a virtual circle surrounding the vehicle. The intersection of this circle with the specified path defines a goal point. The pure pursuit algorithm then specifies either a desired curvature or a desired centripetal acceleration based on selecting an arc path to this goal point. This arc

is constrained to be tangential to the velocity vector found at the origin of the body fixed frame, as well as constrained to pass through both the origin of the body fixed frame and the goal point. This provides the three necessary conditions to define the coordinates of the arc center as well as the radius of curvature. In practice, this formality in defining the arc path will not be necessary, though it should be implicitly understood.

The obvious initial question is whether arc paths are the optimal choice for directing the vehicle towards the goal point. Though this is an issue that will motivate the second-half of this chapter, it should now be noted that in [3] two alternative approaches were considered: a classical controls approach and a quintic polynomial approach. In the first, a control law was formulated based on the cross-track error between the goal point and the desired orientation at the goal point. In the second approach, a quintic polynomial path was used to direct the vehicle towards the goal. In both cases the true pure pursuit algorithm (arc paths) offered superior performance.

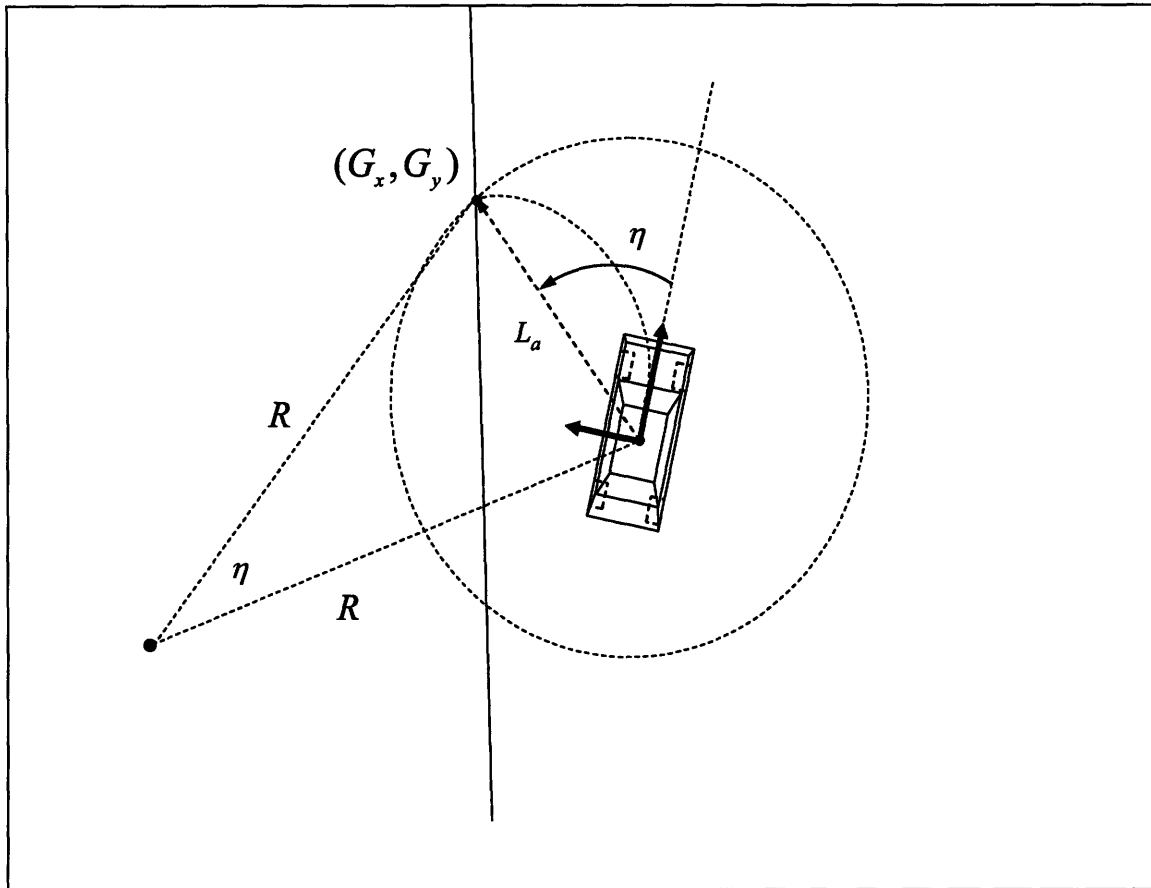


Figure 4.1: Pure Pursuit Goal Point

4.2 Steering Angle

The traditional approach to pure pursuit control affixes the coordinate system to the rear axle and the Ackerman steering assumption is applied. The curvature necessary to reach a desired goal point is given by equation 4.1 and the relevant quantities are illustrated in Figure 4.1.

$$\kappa = \frac{2 \sin \eta}{L_a} \quad (4.1)$$

The resulting steering angle, assuming a kinematic, single-track model, is given by the following:

$$\delta = \tan^{-1} \frac{2L_b \sin \eta}{L_a} \quad (4.2)$$

However, because dynamics are being neglected, there is no need to use a single-track model and this can be reformulated using Figure 3.1, i.e. a double track model. The resulting equation for the inside steer angle is the following:

$$\delta_i = \tan^{-1} \frac{2L_b \sin \eta}{L_a - L_{tw} \sin|\eta|} \quad (4.3)$$

It should be noted that if the front and rear track widths are different, L_{tw} should be taken as that of the front wheels. The alternative approach is to use a dynamic model (equation 3.24) to determine the average steer angle and then map this to the inside steering angle using equation 3.9 and the methodology described in section 3.3. Much of the remainder of this chapter will focus on comparing the use of this dynamic approach with the use of equation 4.2.

4.3 Linearized Straight Line Following

Sections 4.1 and 4.2 introduced the pure pursuit controller and its general implementation. This section will now introduce some basic tools for analysis. Specifically, the simple case of tracking a straight line is used to develop a linear state space model of the system.

Referring to Figure 4.2, the vehicle is initially located at the origin of the inertial XYZ frame and aligned with the X axis. This is equivalent to assuming zero initial conditions for the system. The line the vehicle is to track is located at an offset y_d . Building upon the linear model presented in equation 3.22, the model is expanded by several states. First the yaw (Ψ) of the vehicle is included as a state. Next, the y displacement of the vehicle's c.g. can be related to the existing states by the following equation:

$$\dot{y} = |\tilde{v}| \sin(\beta + \psi) \quad (4.4)$$

Assuming small angles then yields (refer to equations 3.14):

$$\dot{y} = v_x (\beta + \psi) \quad (4.5)$$

If the position of interest is the rear axle, as opposed to the c.g., the following linear equation would be applied:

$$\dot{y} = v_x (\beta + \psi) - l_r \dot{\psi} \quad (4.6)$$

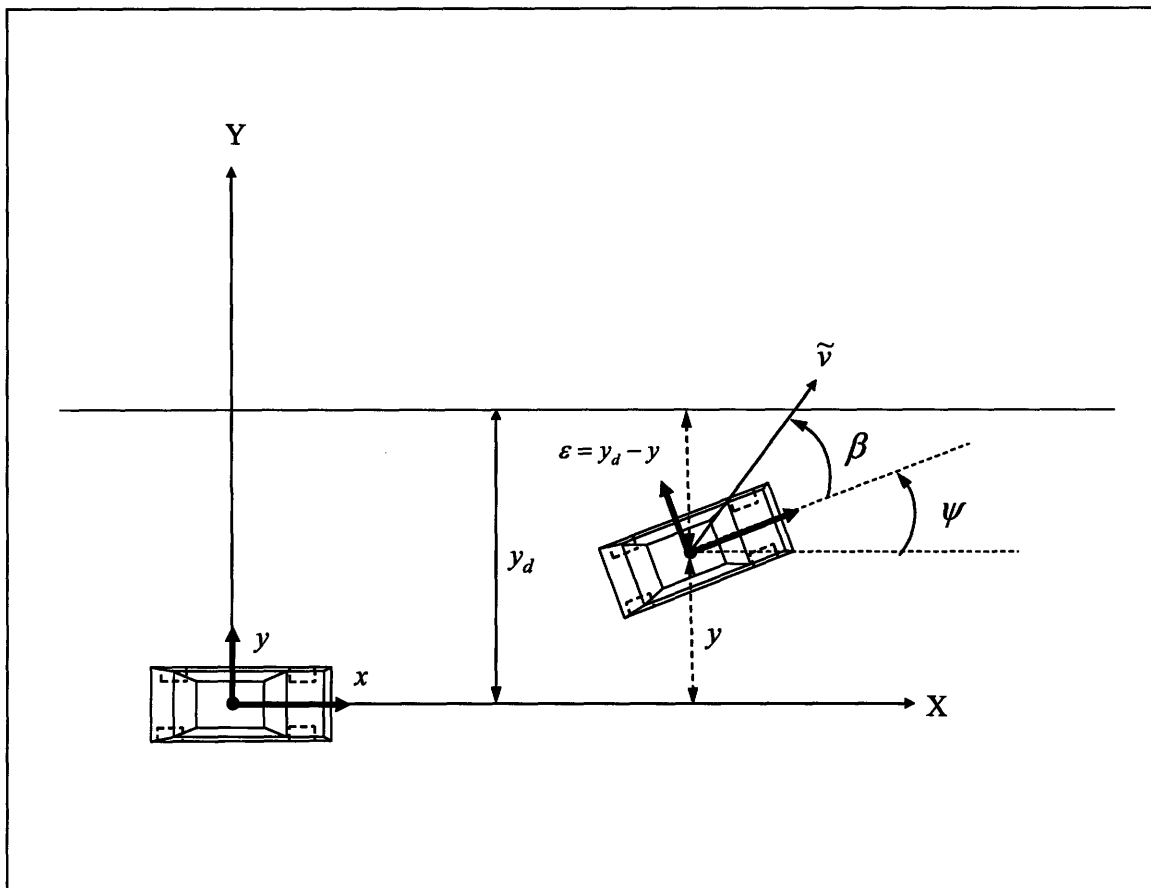


Figure 4.2: Pure Pursuit Straight Line Following

The system model can be augmented by the actuator model provided in equation 3.31.

The final result is given in equation 4.7. As alluded to above, the exact nature of this state space model will differ depending upon the point of interest chosen for cross-track error.

$$\begin{bmatrix} \dot{\beta} \\ \dot{\psi} \\ \dot{\psi} \\ \dot{y} \\ \dot{\delta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} a_{F11} & 0 & a_{F12} & 0 & b_{F11} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ a_{F21} & 0 & a_{F22} & 0 & b_{F12} & 0 \\ v_x & v_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\omega_n^2 & -2\xi\omega_n \end{bmatrix} \begin{bmatrix} \beta \\ \psi \\ \dot{\psi} \\ y \\ \delta \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{K_{act}\omega_n^2}{K_{steer}} \end{bmatrix} y_d \quad (4.7)$$

To facilitate a linear analysis, the non-linear controller needs to be linearized. Referring to Figure 4.3, the pure pursuit angle η can be decomposed as in [20]. The following results:

$$\eta_a \approx \frac{y_d - y}{L_a} \quad (4.8)$$

$$\eta_b \approx -\psi - \beta$$

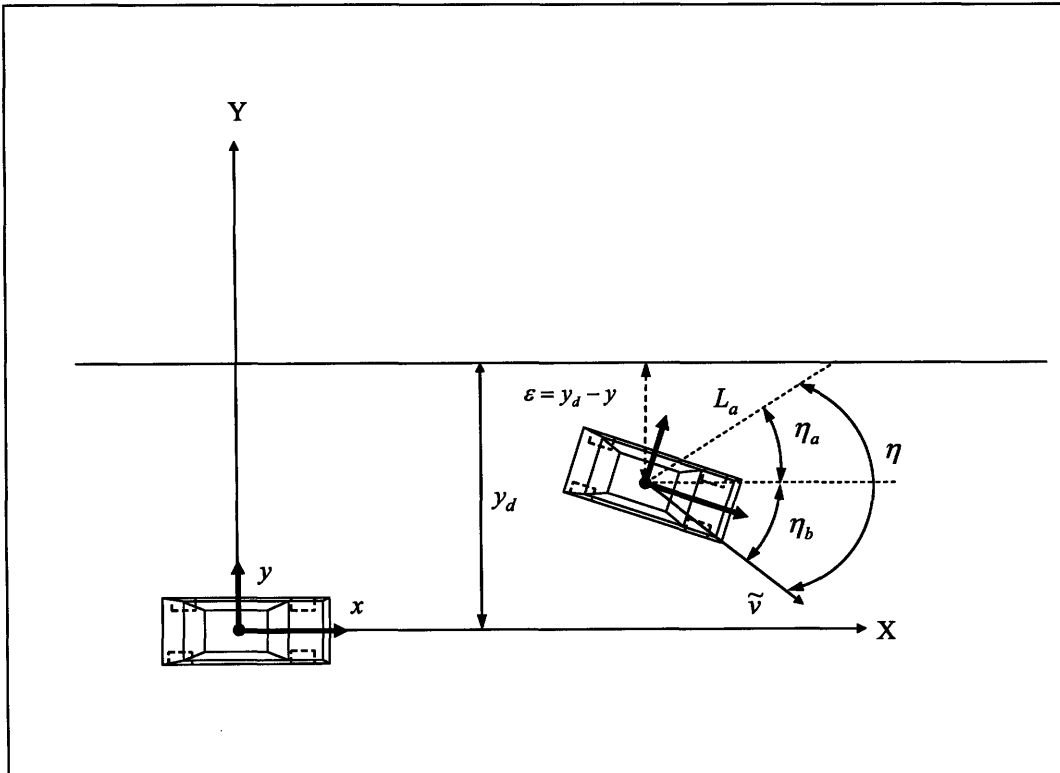


Figure 4.3: Pure Pursuit Angle Decomposition

The desired path curvature, after once again applying a small angle assumption, is then given by the following:

$$\kappa = 2 \left(\frac{y_d - y}{L_a^2} - \frac{\psi + \beta}{L_a} \right) \quad (4.9)$$

As is a consistent theme through this work, the control law has to be cast in different forms based on whether Ackerman steering is assumed or a dynamic approach is adopted. In the case of an Ackerman based approach, the body fixed frame is located at the rear axle and the velocity at this point is assumed along the longitudinal axis of the vehicle. As such, the linearized control law is derived from equation 4.2 as the following:

$$\delta_d = 2L_b \left(\frac{y_d - y}{L_a^2} - \frac{\psi}{L_a} \right) \quad (4.10)$$

Here a subscript d is added to the steering angle term to indicate that it is the desired value outputted from the controller. Adopting a dynamic approach, equation 3.24 and equation 4.9 are combined to yield equation 4.11 presented below.

$$\delta_d = 2 \left(\frac{y_d - y}{L_a^2} - \frac{\psi + \beta}{L_a} \right) \left(L_b + \frac{k_{us} v_x^2}{g} \right) \quad (4.11)$$

For additional clarity, Figure 4.4 presents the two approaches graphically.

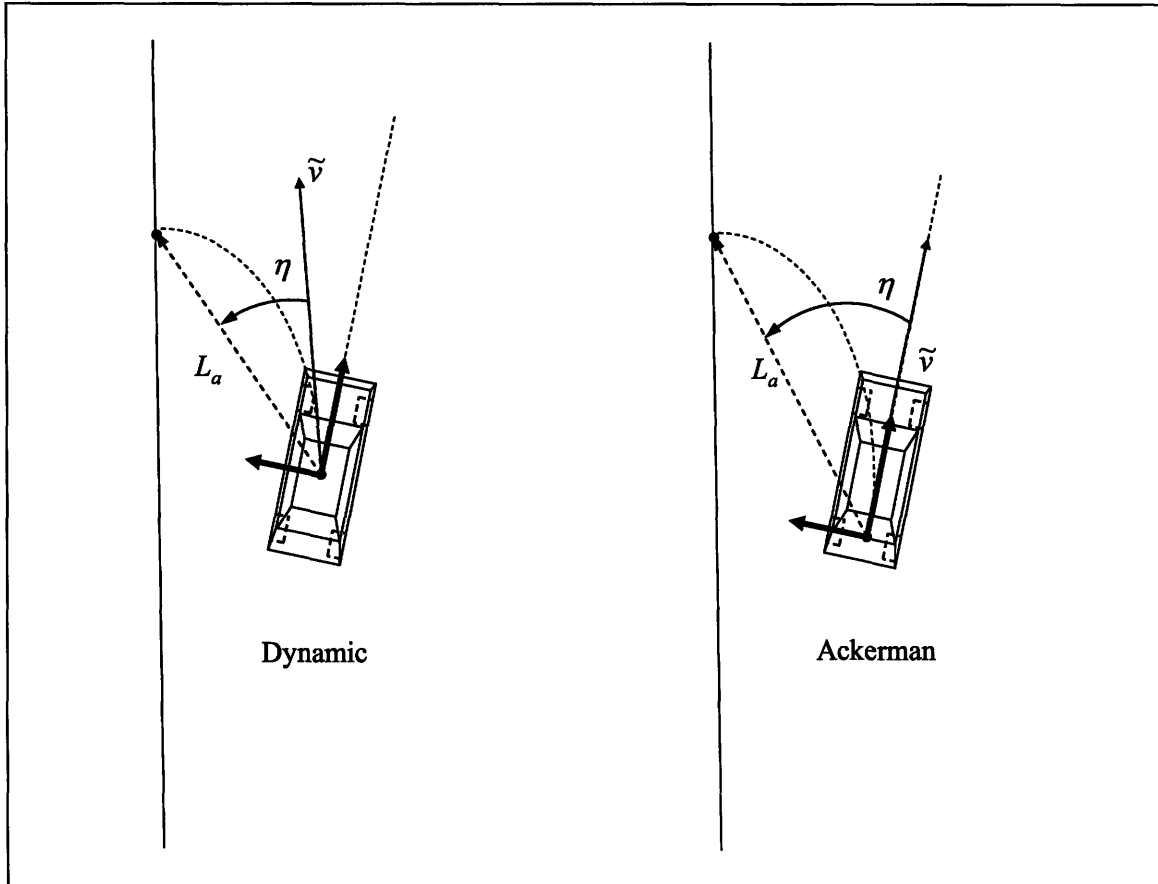


Figure 4.4: Ackerman Pure Pursuit and Dynamic Pure Pursuit

The linearizations given in equations 4.10 and 4.11 are valid, but offer little insight into the nature of the pure pursuit controller. The work of [20] presents an alternative linearization where the decomposition of the pure pursuit angle η is approximated as in equation 4.12. Here \dot{y}_d is taken as zero for straight line driving.

$$\eta_a \approx \frac{y_d - y}{L_a} \tag{4.12}$$

$$\eta_b \approx -\frac{\dot{y}}{v_x} - \beta$$

The linearized controller for an Ackerman based approach to pure pursuit is then recast as the following:

$$\delta_d = 2L_b \left(\frac{y_d - y}{L_a^2} - \frac{\dot{y}}{L_a v_x} \right) \quad (4.13)$$

For the dynamic approach, we now choose to assume side-slip is small and combine equations 3.24, 4.9, and 4.12 to yield the following:

$$\delta_d = 2 \left(\frac{y_d - y}{L_a^2} - \frac{\dot{y}}{L_a v_x} \right) \left(L_b + \frac{k_{us} v_x^2}{g} \right) \quad (4.14)$$

The linearizations of 4.13 and 4.14 are now represented by a more traditional controls block diagram in Figure 4.5. In essence, pure pursuit behaves as a PD controller on the vehicle's cross-track error.

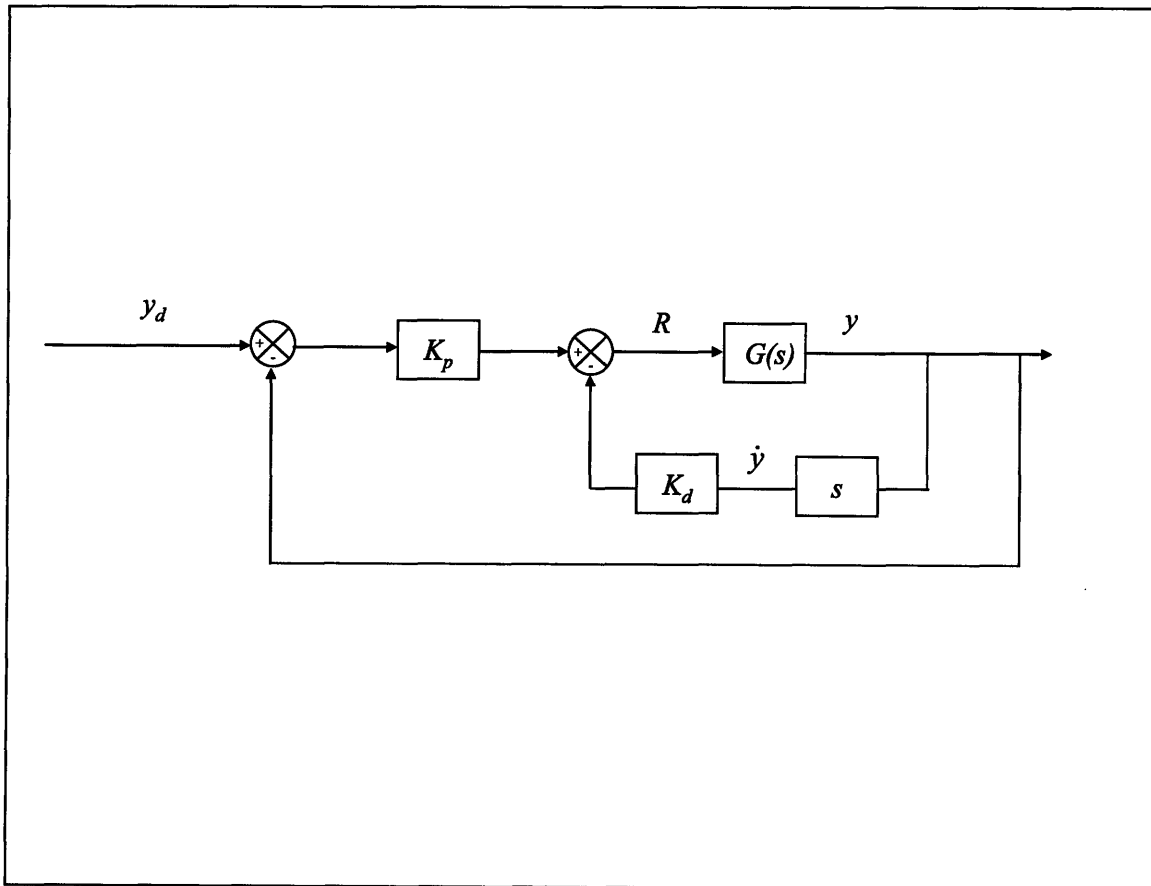


Figure 4.5: System Block Diagram

The gains in Figure 4.5 are given as the following for the Ackerman assumption case:

$$K_p = \frac{2L_b}{L_a^2}$$

$$K_d = \frac{2L_b}{L_a v_x}$$
(4.15)

For the dynamic model case the gains are given by the following:

$$K_p = \frac{2L_b}{L_a^2} + \frac{2K_{us} v_x^2}{gL_a^2}$$

$$K_d = \frac{2L_b}{L_a v_x} + \frac{2K_{us} v_x}{gL_a}$$
(4.16)

Two observations should be explicitly noted here. First, regardless of whether a dynamic based or an Akerman based approach is employed, the gains are functions of the look-ahead distance. Moreover, because the look-ahead distance appears as a quadratic term in the denominator of the proportional gain, a simple root locus is not possible. Second, the gains for equation 4.13 and equation 4.14 will be very nearly identical for low speeds. However, at larger speeds these gains will differ (assuming an identical look-ahead distance in both cases). This is an observation consistent with previous analysis (section 3.3).

Treating the controller as a PD controller also facilitates some basic steady-state analysis. For simplicity, we take $G(s)$ (the vehicle model) in Figure 4.5 as simply $V^2/(s^2 L_b)$. This is derived from a linearization of equation 3.2. By taking the vehicle transfer function as the above, it is assumed that the lateral acceleration of the vehicle and \ddot{y} (Figure 4.2) are approximately equivalent under small angle assumptions. The block diagram of Figure 4.5 is now transferred into the equivalent system illustrated in Figure 4.6.

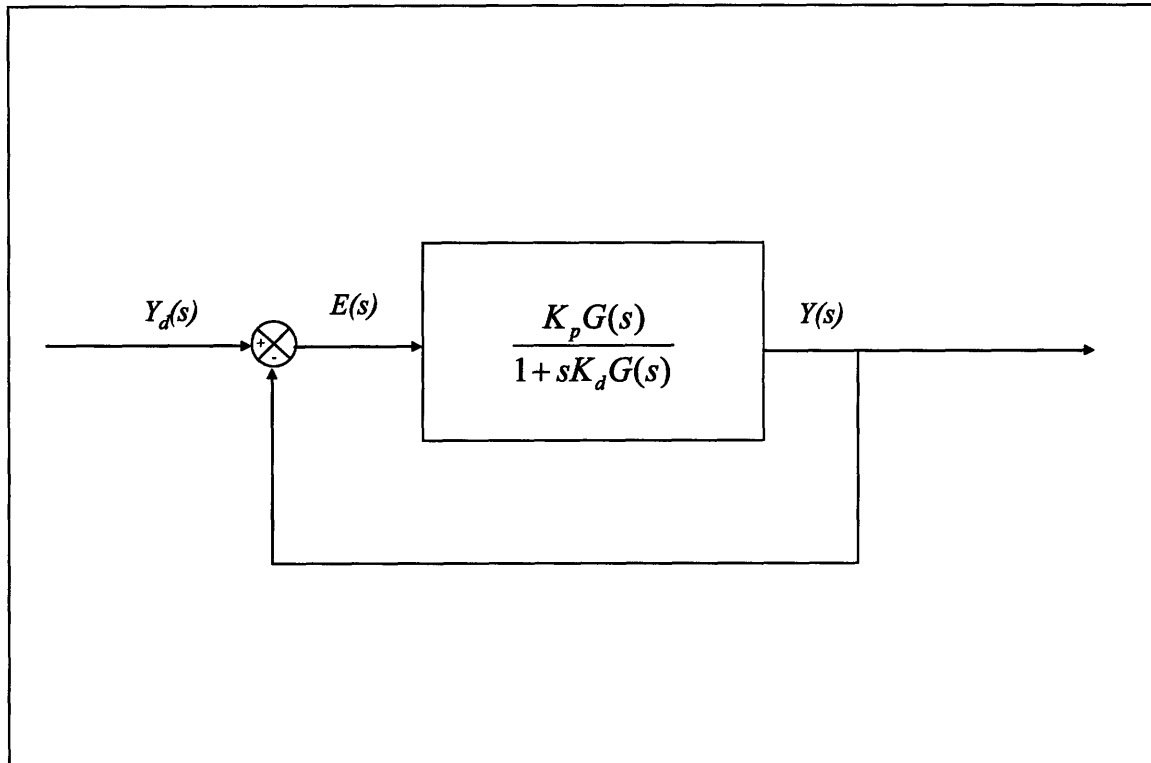


Figure 4.6: Simplified System Block Diagram

Here the error $E(s)$ is more explicitly incorporated into the system diagram. From this it can be easily shown that the following holds:

$$E(s) = \frac{Y_d(s)(1 + sK_d G(s))}{1 + sK_d G(s) + K_p G(s)} \quad (4.17)$$

Applying the final value theorem [18] will then give an indication of the anticipated steady-state error for various types of inputs. The result is the following for step inputs ($1/s$):

$$\lim_{s \rightarrow 0} s \frac{Y_d(s)(1 + sK_d G(s))}{1 + sK_d G(s) + K_p G(s)} = 0 \quad (4.18)$$

It can then be inferred that the pure pursuit controller will produce zero steady-state error for straight line following. Moreover, it can be shown that this is also true if the input function $Y_d(s)$ is taken as a ramp input. In this instance, however, it is necessary to incorporate the derivative of the setpoint (\dot{y}_d) as this is no longer the case of following a

horizontal line. In the case of a quadratic input, a steady-state error does persist and is given by the following (using the simplifying assumption on $G(s)$ of Figure 4.6):

$$E(s) = \frac{L_b}{K_p V^2} \quad (4.19)$$

As with traditional implementation of pure pursuit, it is assumed that it will be necessary to schedule the value of L_a with the current speed (s) of the vehicle. Though a search similar to a robust control approach could be done to establish an L_a that stabilizes the system over the whole range of expected speeds, this is unnecessary. Based on intuition, L_a is given a functional representation as illustrated in Figure 4.7.

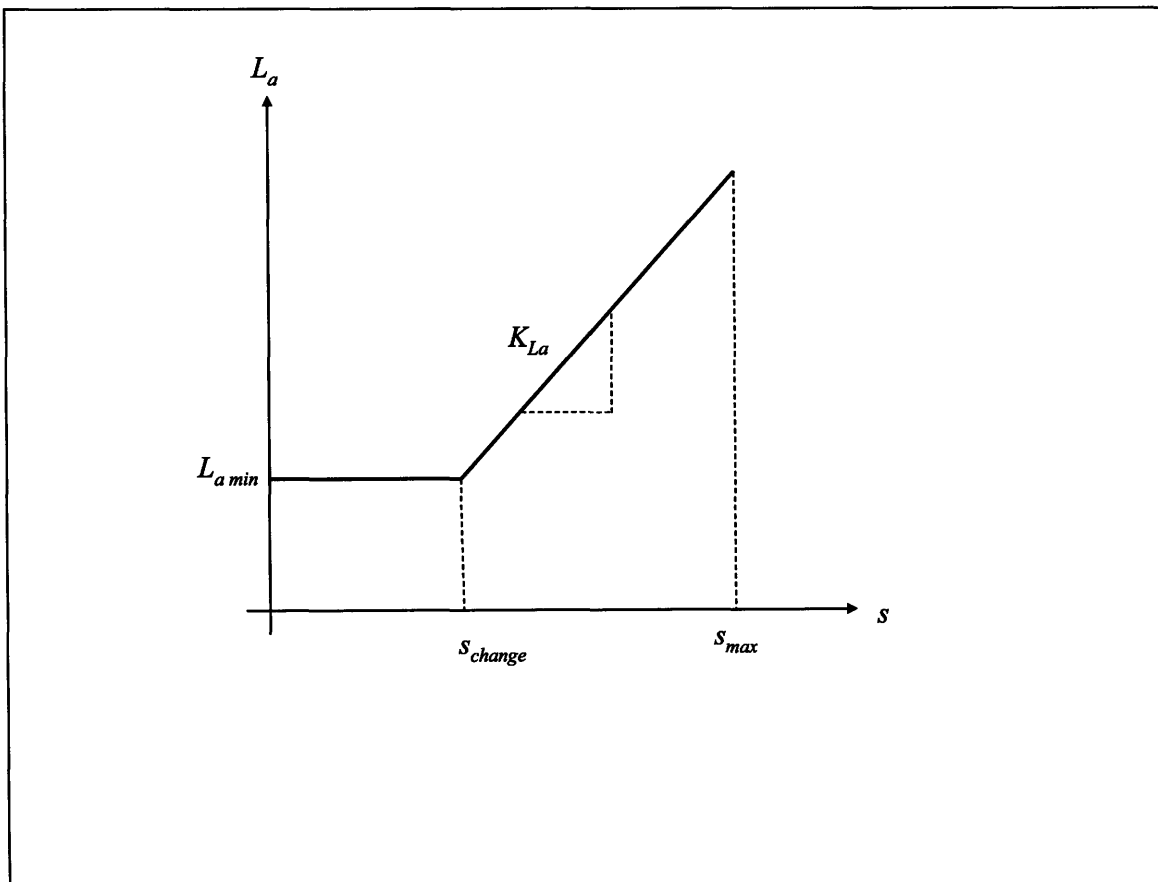


Figure 4.7: Look-ahead Distance as a Function of Speed

Functionally, this may be written as:

$$L_a = \begin{cases} L_{a \min} & s < s_{change} \\ K_{La} s & s \geq s_{change} \end{cases} \quad (4.20)$$

Where s_{change} is defined as the following:

$$s_{change} = \frac{L_{a \min}}{K_{La}} \quad (4.21)$$

This yields two tuning parameters for the look-ahead distance, the term K_{La} and the minimum value of the look-ahead distance $L_{a \min}$. The problem of selecting the appropriate look-ahead distance for every speed has now been recast into a problem of selecting the tuning parameters such that the system is stable across the range of anticipated speeds. It should be noted that this speed scheduling of L_a has the net effect of reducing the pure pursuit gains at higher speeds, effectively ensuring that the vehicle will not overcorrect for small cross-track errors.

To ensure system stability, a pole-zero plot for the closed-loop system can be found for every speed. Thus, if a maximum overshoot or a maximum settling time are specified for a vehicle traveling at top speed (or the top speed expected for the DUC) the tuning parameters for the look-ahead distance may be selected to ensure that the dominant poles of the system meet this criteria. This is illustrated below in Figure 4.8 where a pole-zero plot is presented for a system with an $L_{a \min}$ of 3 and a K_{La} of 2.25. This plot was generated using the dynamic pure pursuit gains (equation 4.16) with a linearized model of Talos-I. Using identical tuning parameters and gains, Figure 4.9 presents the pole-zero plot for a linearized model of Talos-II. Both plots show the dominant poles for speeds of .5 m/s to 13 m/s incremented by .5 m/s.

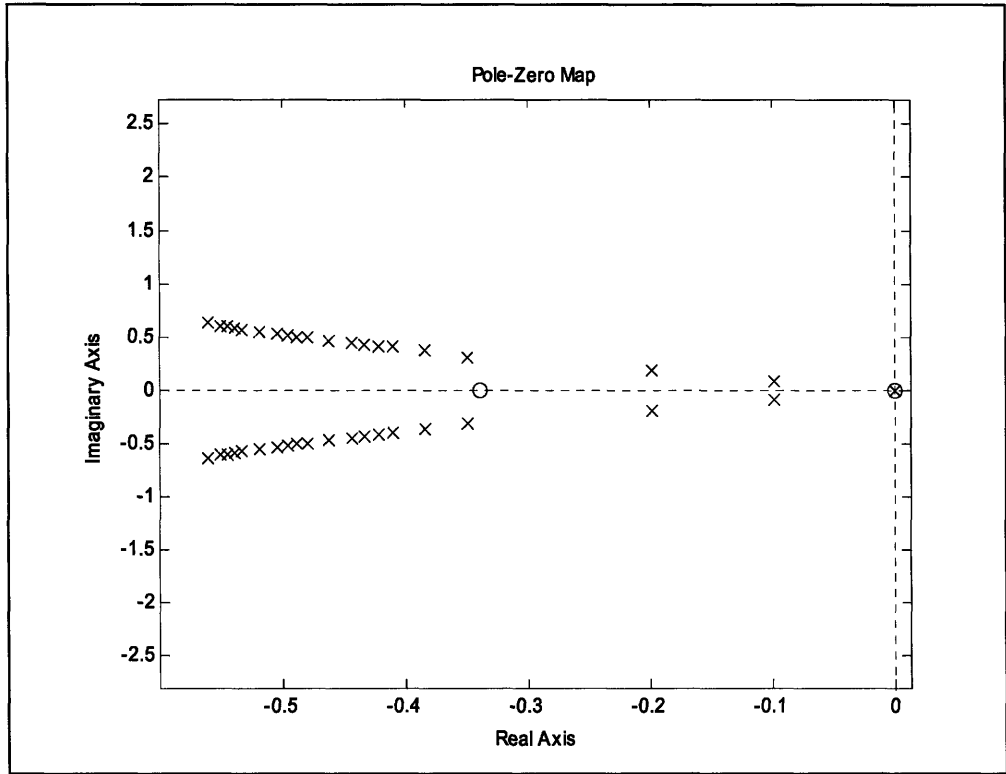


Figure 4.8: Linearized Talos-I Model with Dynamic Pure Pursuit Gains

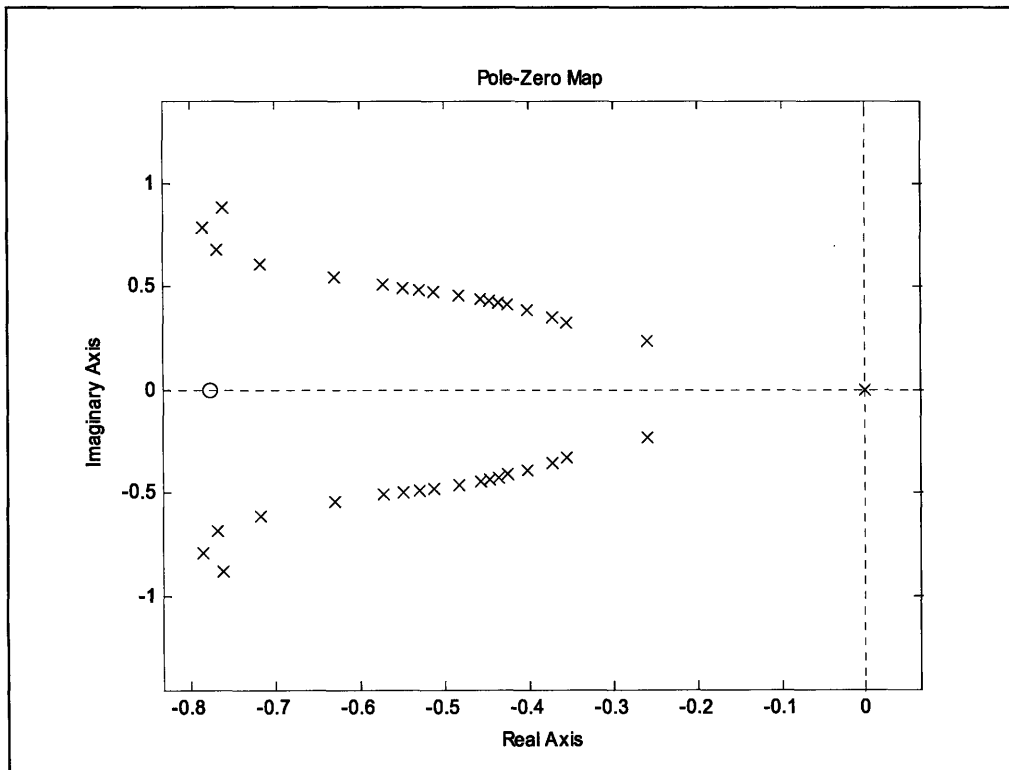


Figure 4.9: Linearized Talos-II Model with Dynamic Pure Pursuit Gains

At a speed of 13 *m/s* (29 *mph*) the linearized model predicts that Talos-I will exhibit a damping of .666 and an overshoot of 6.07%. At that same speed, the linearized model predicts that Talos-II will exhibit a damping of .6 and an overshoot of 9.51%. These results, assuming identical operating conditions and identical tuning parameters, can now be compared to those found using the Ackerman based gains (equation 4.15). Referring to Figures 4.10 and 4.11, the linearized model predicts a damping of .503 and an overshoot of 16.1% for Talos-I and a damping of .513 and an overshoot of 15.3% for Talos-II. It is clear from the results that using a dynamic map to determine the correct steering angle produces a significant improvement over using the traditional Ackerman assumption.

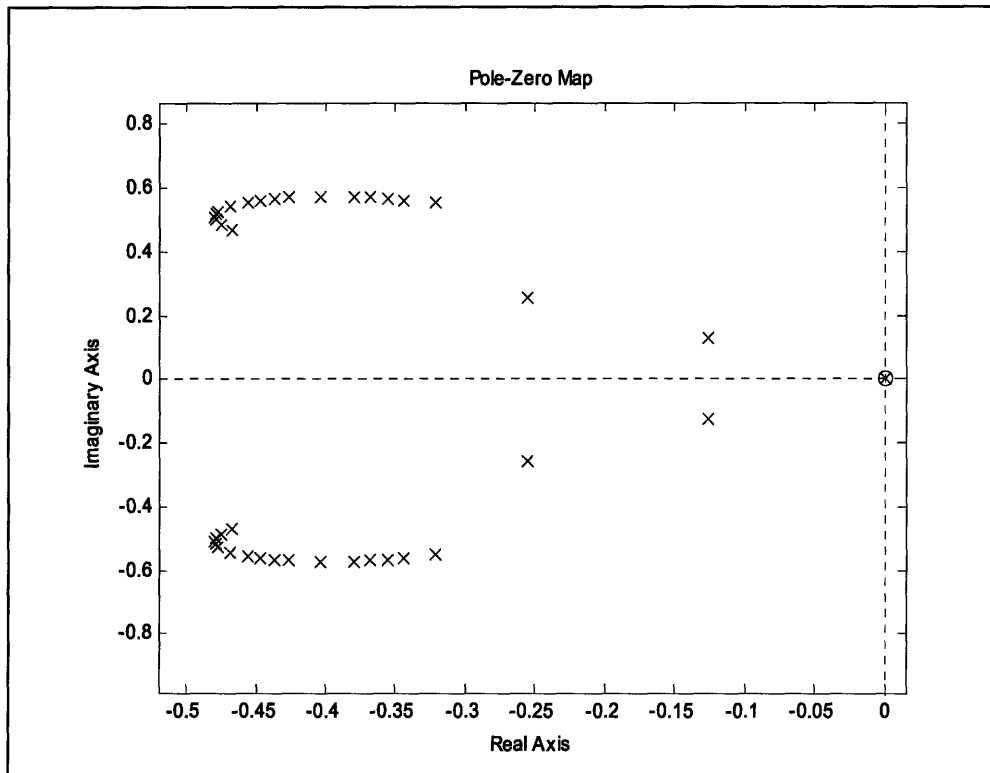


Figure 4.10: Linearized Talos-I Model with Ackerman Based Pure Pursuit Gains

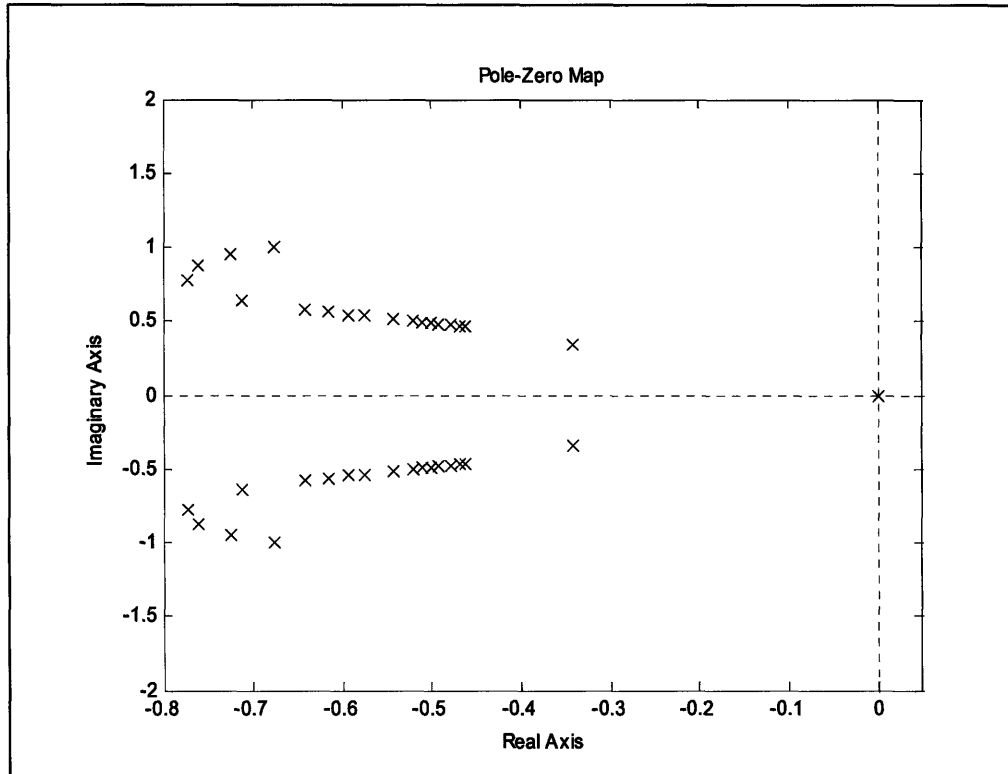


Figure 4.11: Linearized Talos-II Model with Ackerman Based Pure Pursuit Gains

4.4 Nonlinear Modifications to Pure Pursuit Algorithm

As noted in the introduction to this chapter, there are alternative approaches to the look-ahead distance tuning presented thus far. These alternatives were first explored in [15].

The first primary alternative is to not project the look-ahead distance from the vehicle's true location, but from the vehicle's projected location on the path. This can intuitively help stabilize the system as large offsets from the path will not result in large values of the pure pursuit angle. This approach is illustrated graphically in Figure 4.12. In practice, if the vehicle tracks the path with a reasonable fidelity, then this adjustment will not result in an appreciable performance change. However, if disturbances result in large cross-track errors, this approach can help prevent the system from going unstable.

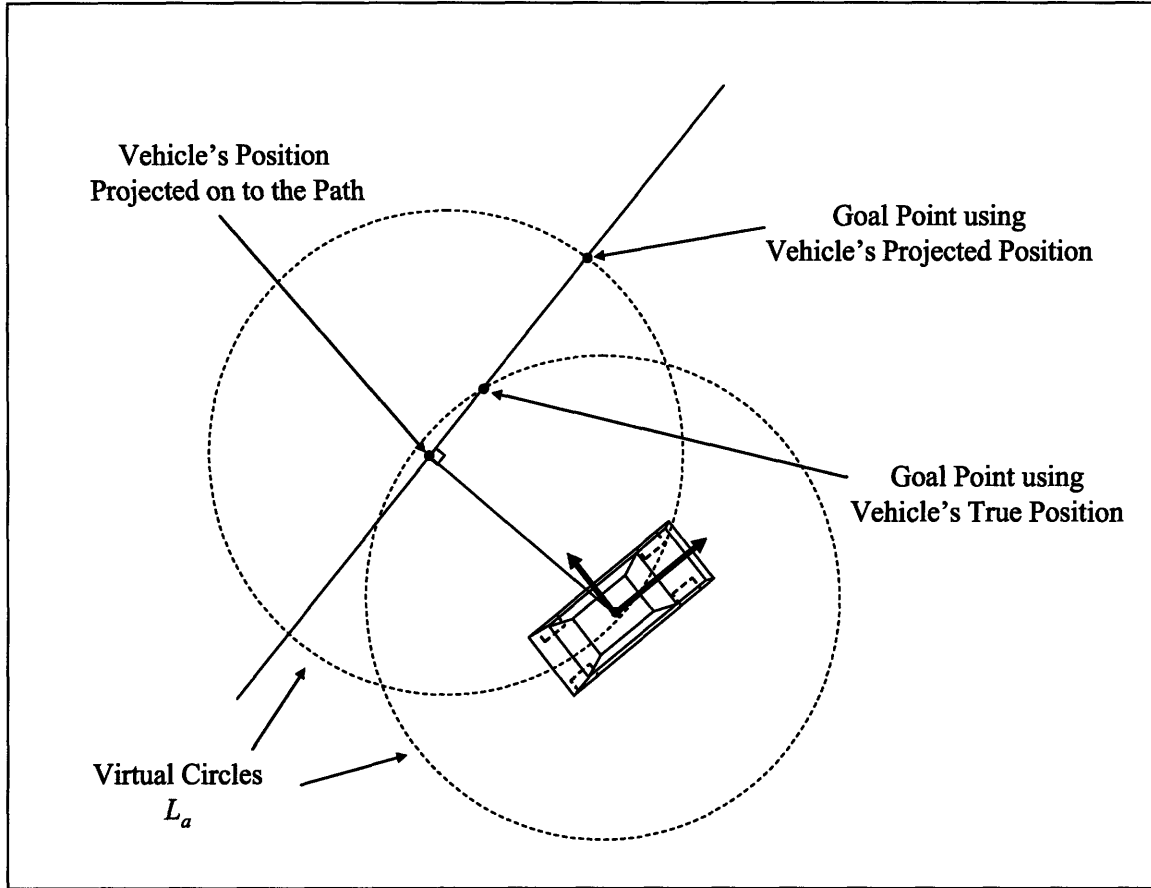


Figure 4.12: Using Vehicle's Position on Path to Define Pure Pursuit Goal Point

Also explored in [15] was an approach where a minimum L_a was selected and this value was increased proportionally with the cross-track error, as opposed to vehicle velocity. A natural extension is to then reformulate the look-ahead distance as a function of both speed and cross-track error. A simple implementation of this is to treat the cross-track error as an upward shift of the curve presented in Figure 4.7. Mathematically this is expressed as the following:

$$L_a = \left\{ \begin{array}{ll} L_{a\min} + K_{ct}|(y_d - y)| & s < s_{change} \\ K_{La}s + K_{ct}|(y_d - y)| & s \geq s_{change} \end{array} \right\} \quad (4.22)$$

Here K_{ct} is a tunable gain always greater than or equal to 0 and the definition of s_{change} remains unaltered.

As noted in the introduction, neither of these approaches are amenable to the analysis presented in section 4.3. In the case of using the vehicle's projected position on the path, a separate analysis is not entirely necessary as this will have a limited impact on the system when small cross-track errors are present. Adjusting the look-ahead distance with cross-track error, however, adds additional non-linearities that cannot be easily approximated. This is especially true considering the presence of the look-ahead distance in the denominators of the linearized gains presented in equations 4.13 and 4.14.

Ultimately, though these approaches are not well suited to analytical rigor, they still offer a powerful means to improve overall stability. For instance, the gain K_{ct} can cause degradation in the tracking performance, but will not cause the vehicle to go unstable. This is intuitively true as a larger look-ahead distance adds greater damping to the system. In instances in which the cross-track error is large, this approach will mitigate the tendency for the pure pursuit controller to output a sharp turn correction. However, as aforementioned, the problem lies in quantifying the degree to which K_{ct} can degrade overall tracking performance.

4.5 Simulation-based Approach to Steering Angle Selection

A vehicle's ability to execute the arc paths selected by the pure pursuit controller is severely limited by the bandwidth of the steering actuator. As pointed out in [15], the movement of the steering wheel from a position at time t to the desired position at time $t+1$ results in the vehicle following paths which are clothoidal, where a clothoid is a curve whose curvature is proportional to its arc length, as illustrated in Figure 4.13. More precisely, a vehicle follows clothoidal paths as the steering wheels transition from one

fixed position to another. When the steering wheels are held constant, the vehicle follows constant curvature arcs.

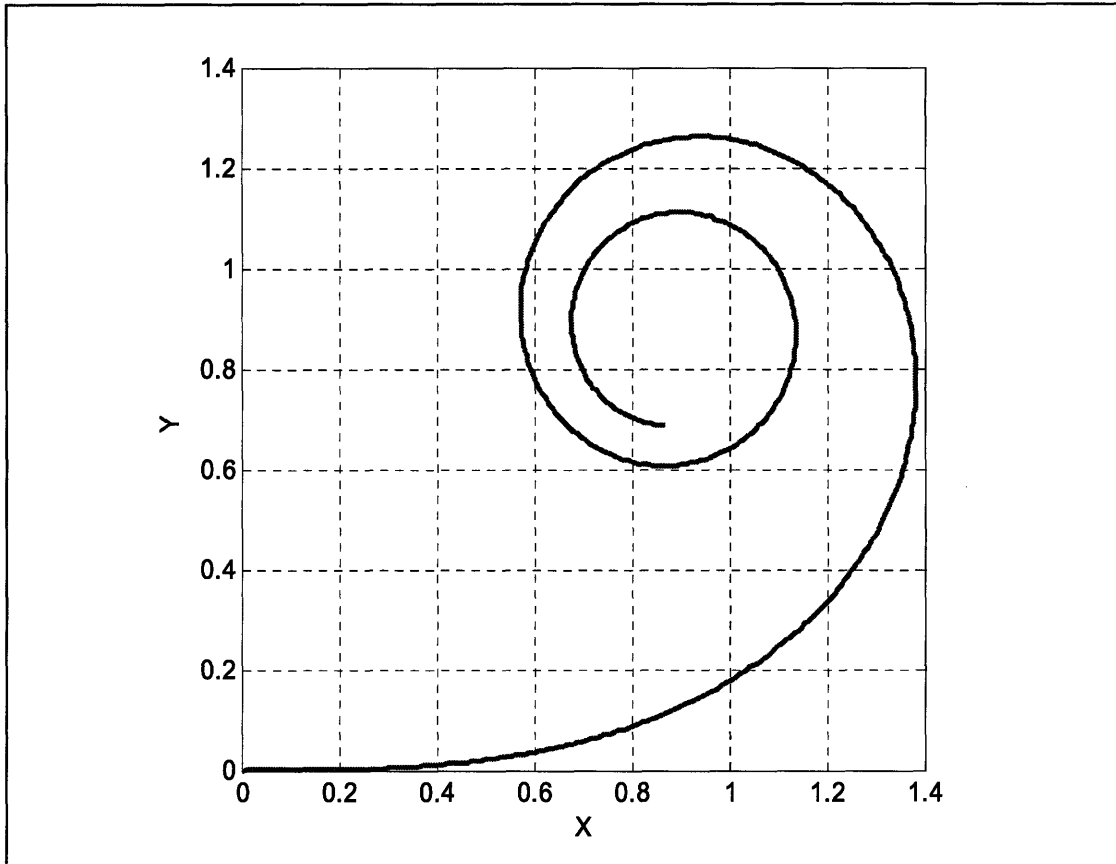


Figure 4.13: Clothoidal Curve

In practice, the arc based approach of sections 4.1 - 4.4 works quite well. However, as pointed out in [15], there is a serious failure mode that results because the pure pursuit controller is not able to take into account the current position of the wheels. As an illustration of this, consider the case presented in Figure 4.14 in which the vehicle has a heading along Y with the wheels turned full to the left. Given the marked goal point, the pure pursuit controller would output a desired path which would keep the wheels pointing left. In reality, however, the car will not reach the target point unless it is commanded to unwind the wheels with a nearly full right command. A solution to this problem is to then employ a simulation-based approach in which it is necessary to

discretize the space of possible steering inputs, forward simulate the vehicle's response based on a simple vehicle model, and finally select the input that produces the best performance. One might observe that this approach is quite similar to a model predictive optimal controller. However, because the formalism and rigor associated with such a controller is not presented here, this approach is termed a *simulation-based controller* throughout this work.

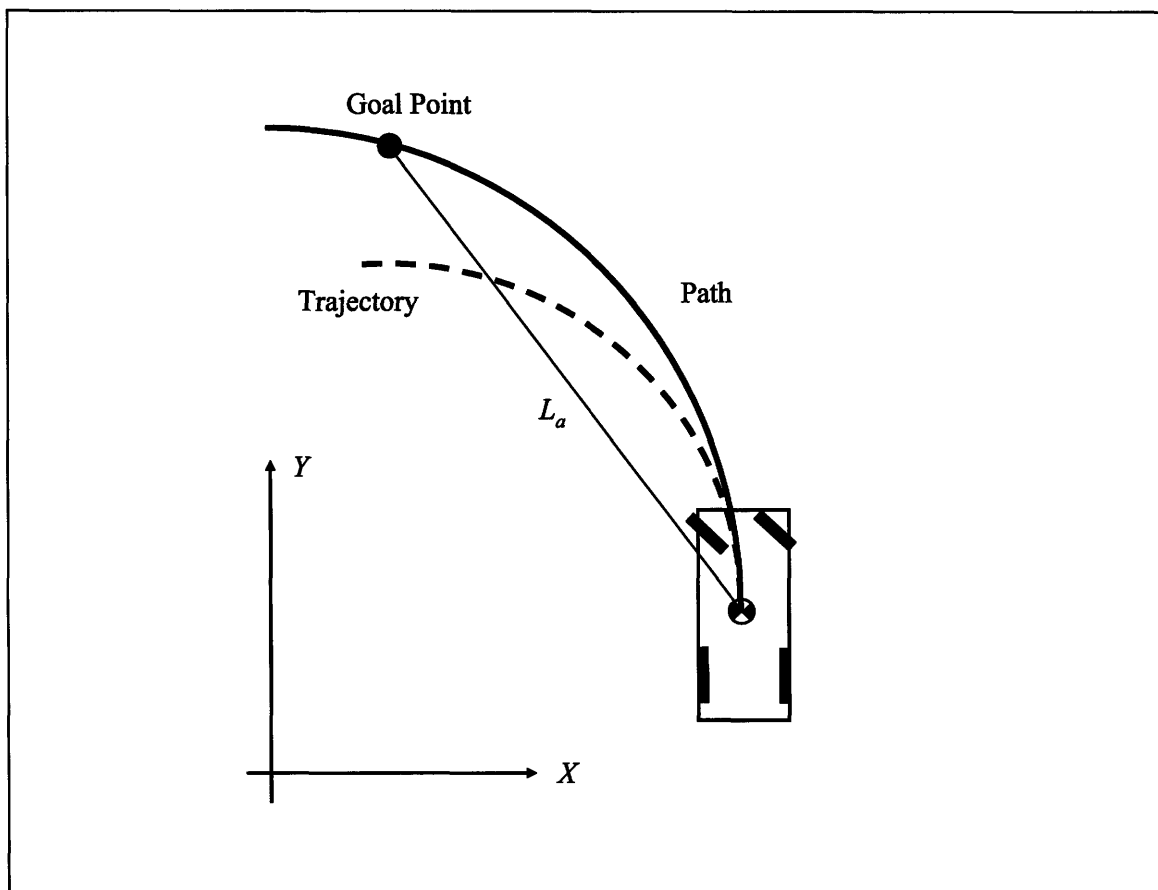


Figure 4.14: Pure Pursuit Failure Mode

For implementation of this simulation-based approach we rely upon an encoder mounted to the steering wheel actuator. This supplies position information for the steering wheel and thus the wheels on the ground. Additionally, differentiation of the steering wheel position provides angular velocity information for both the steering wheel and the inside turning wheel. Ideally a tachometer would have been employed to give this

information, but with a low pass filter, useful angular velocity information can still be obtained. In relating information regarding the steering wheel to information concerning the wheels on the ground, section 3.3 should be referenced.

For forward simulation, the linearized dynamic model could be employed. However, because this is slightly more complex to implement, a simple kinematic model is chosen instead. To alleviate the need for a mapping between the average steer angle and the inside steer angle (section 3.3), the double-track model introduced in section 3.1 is employed. For convenience, this model is reproduced below.

$$\begin{aligned}\dot{\psi} &= \frac{2V \tan \delta_i}{2L_b + L_{tw} \tan|\delta_i|} \\ \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi\end{aligned}\tag{4.23}$$

As aforementioned, the steering wheel model is initialized with the current position of the steering wheel and the steering wheel angular velocity. With regard to the vehicle model, the vehicle's true speed at the start of the forward simulation is taken as constant. The space of steering inputs is then discretized based on the range of motion for the actual wheels on the ground. For every point in this discrete range the vehicle's path is forward simulated using a Runge-Kutta numerical integration for the steering actuator and an Euler integration for the vehicle model. More generally, given the vehicle's current (X, Y) position, the vehicle's current velocity (V) , the orientation of the steering wheel δ_w , and the angular velocity of the steering wheel $\dot{\delta}_w$, the vehicle model presented in equation 4.23 is used to predict the vehicle's response to a series of different actuator command signals. The input that is predicted to get the vehicle closest to the desired goal point is then deemed optimal and selected for that time step.

Because the simulation cannot be continued indefinitely, a termination criterion must be enforced. For this criterion, it is decided that the simulation for an input will terminate when the vehicle's predicted position has crossed a line that passes through the goal point and is perpendicular to the path tangent at that location. This imaginary line is referred to as the *terminating line* and is illustrated in Figure 4.15. Of course, this can put the simulation in an infinite loop if the vehicle's simulated position will never cross this virtual boundary. To compensate for this, it is necessary to terminate the search if the predicted change in vehicle's orientation exceeds 180° .

To illustrate the nature of this simulation, Figure 4.15 presents achievable trajectories (using parameters from Talos-II) given that the vehicle is initially located at $(0,0)$, that the wheels are positioned 12° to the left (from the body fixed perspective), that the vehicle is moving at a speed of 14 m/s , and that the vehicle has an initial heading of 0° . Additionally, the target point is located at $(30,1)$ and the terminating line is at an orientation of 90° from the X axis. Given these initial conditions, the solution generated by traditional pure pursuit (provided this solution was allowed to propagate forward without feedback) is a desired steering wheel angle of 8.97° . The solution from the simulation-based approach is a desired steering wheel angle of -584° (full right). Clearly, because the simulation-based approach utilizes information concerning the initial wheel angles and a predictor of how the car will perform given a certain input, it is able to make a more informed decision about the command that should be sent at time t in order to get as close to the target point as possible.

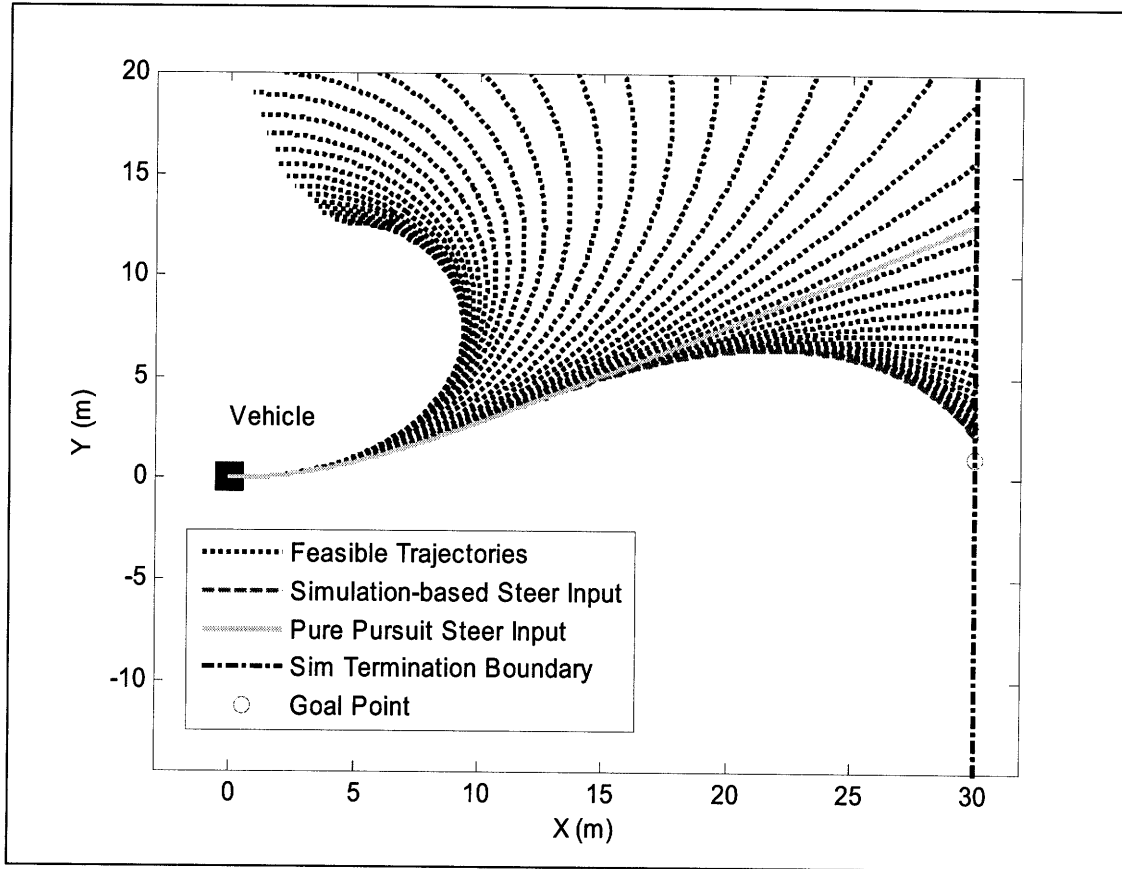


Figure 4.15: Simulation-based Approach to Steering Angle Selection

If the scenario illustrated in Figure 4.15 is repeated at a slower speed of 8 m/s with the wheels at an initial steer angle of only 3° , the simulation-based approach and pure pursuit will produce the results presented in Figure 4.16. The pure pursuit desired steer angle is 7.35° and the simulation-based desired steer angle is 0° . Here we see that the two solutions are nearly identical, as the actuator dynamics are not as significant at lower speeds with only small changes in steering required to execute a maneuver. Additionally, it is observable that the pure pursuit command actually produces the better input as it is not subject to the error that results over searching a discrete space of possible inputs.

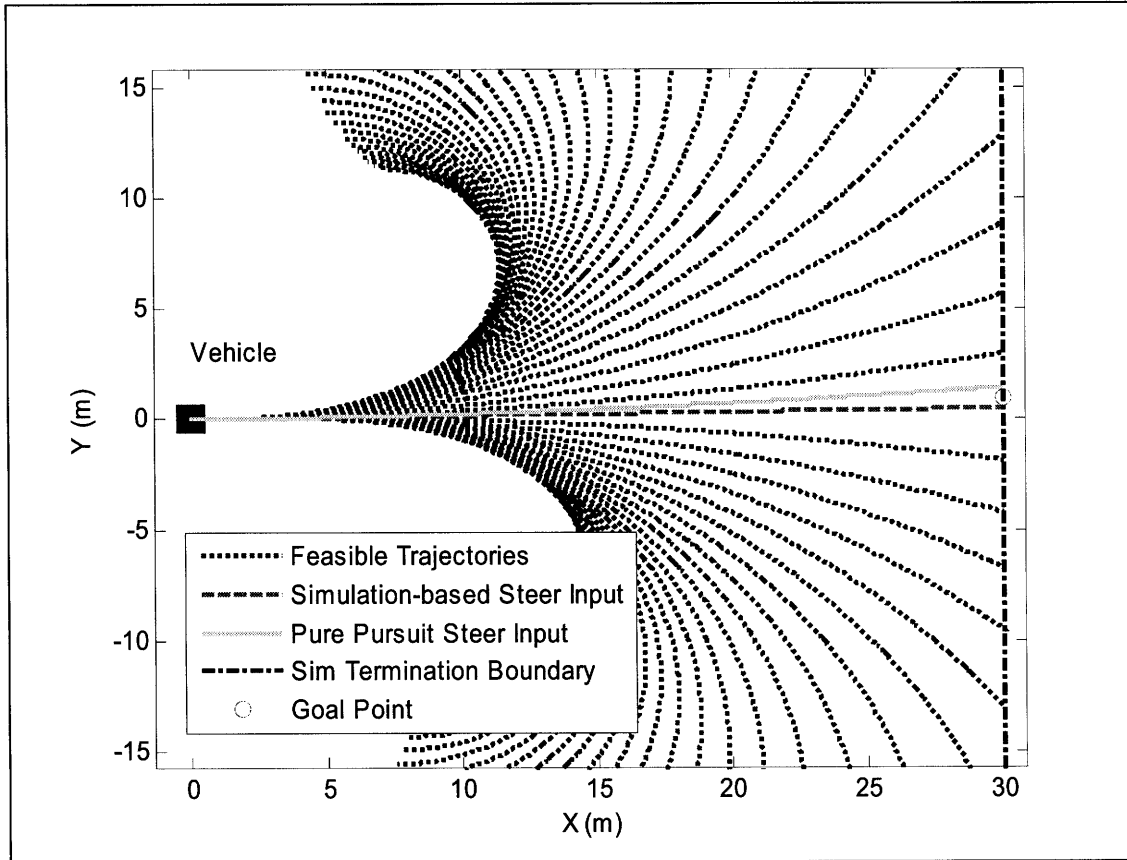


Figure 4.16: Simulation-based Approach to Steering Angle Selection

To solve the problem of searching a discrete space, we can employ a more efficient search algorithm, such as a *Bisection Search* or a *Golden Section Search* method. In terms of implementation, the algorithm would initially forward simulate using both the maximum and minimum values of the input range (the max left and right steering inputs). As before these simulations would terminate when the predicted vehicle position crossed the terminating line. If the bisection search method is utilized, the simulation would then be done for an input of half the maximum steering range such that the searchable space is divided in half. The portion of the original searchable space containing the goal point is now the new searchable space. This process then repeats continually halving the searchable space until the chosen input sends the car within some small, designer-specified distance of the goal point. The golden section search method is identical in

procedure, but the searchable space is continually reduced based on the golden ratio rather than by a factor of 1/2.

The challenge of implementing these traditional search methods is to determine whether the goal point is initially inside the searchable space or outside of the searchable space. This can be achieved efficiently by calculating unit vectors from the goal point to the end points of the maximum and minimum input trajectories (denoted u_{\min} and u_{\max} in Figure 4.17) and representing these vectors in a goal point coordinate frame. The goal point coordinate frame (denoted by x_g and y_g in Figure 4.17) is defined with its x_g axis pointed along the path tangent at the goal point location. As a result, if the y_g components of the aforementioned unit vectors have identical signs, then the goal point is out of range, and the steering input should default to either the maximum or minimum depending on which comes closer to the actual goal point. Conversely, if the y_g components of u_{\min} and u_{\max} have opposite signs, the goal point is within the searchable space and the search algorithm can proceed as normal, terminating when the goal point and the end point of the simulated path are within the designer-specified tolerance.

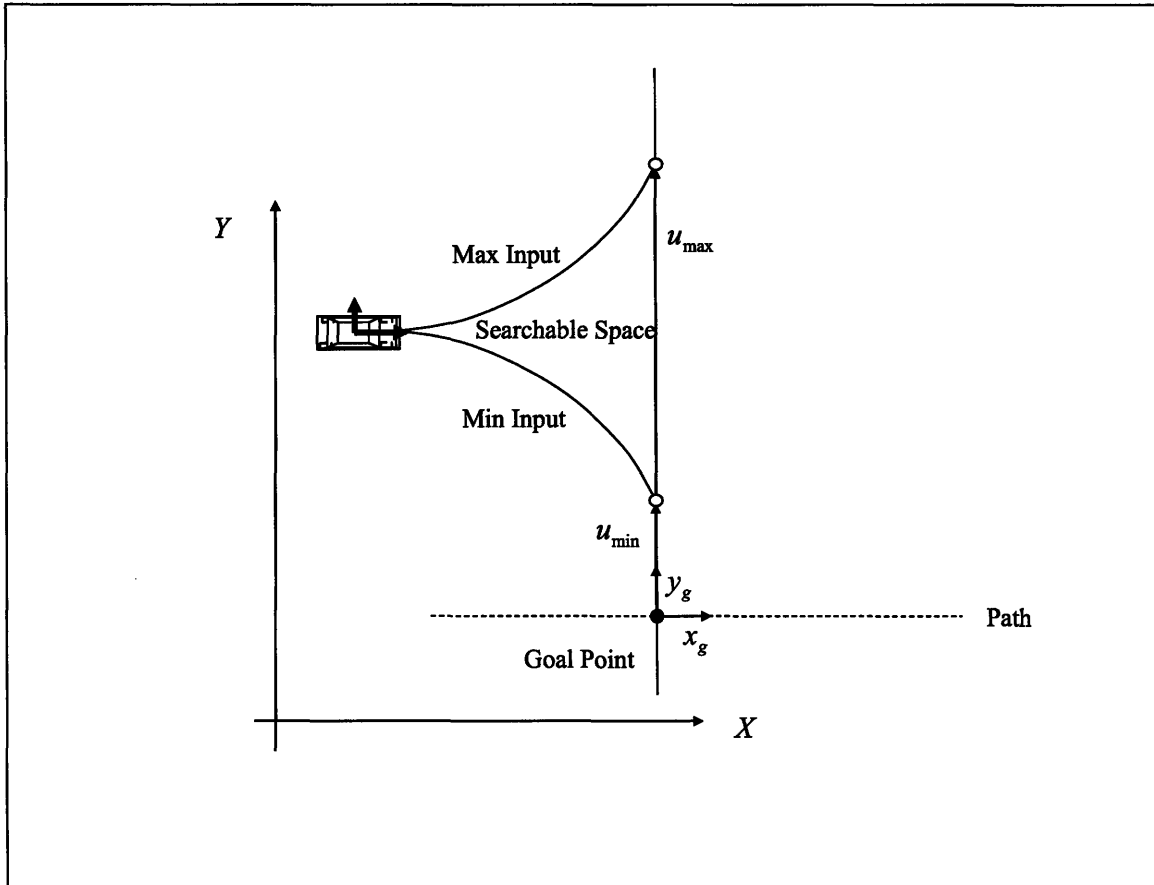


Figure 4.17: Evaluating Searchable Space for Presence of Goal Point

To this point, no discussion has been given to the selection of the look-ahead distance which would be used to find the goal point. Implicitly, the analysis from the previous sections of this chapter would be used to select the distance L_a as a function of velocity. The approach presented here serves as a substitute for the use of equation 3.24, obviating the need to explicitly calculate the desired radius of curvature as the simulation immediately searches for the best possible way to achieve the goal point.

4.6 Chapter Conclusions

This chapter was aimed at presenting analysis tools and analysis results for a pure pursuit controller. The analysis was aimed at using straight-line tracking as a means for gain scheduling the look-ahead distance as a function of velocity. Additional emphasis

was given to comparing the results of using a kinematic (i.e. Ackerman) approach versus a dynamic approach. Ultimately it was found that using a dynamic steering angle map produces a significant increase in system damping and overshoot when compared to the Ackerman based approach. A brief exploration was then given for different modifications that can be made to the look-ahead distance in order to further stabilize the controller.

Finally, a simulation-based approach for implementing the controller was presented. Here it was pointed out that more extreme maneuvers will result in poor vehicle performance as the basic pure pursuit controller does not account for the current position of the vehicle's wheels at each update. Additional attention was given to the details of implementing this controller. Specifically, a brute force approach of discretizing the space of possible inputs was presented, as well as an alternative using a more efficient Bisection search algorithm.

5 Vehicle Control in Reverse

This chapter will present two control approaches for path tracking in reverse. To begin, discussion will focus on the use of pure pursuit as a controller for reverse driving. In this discussion, an outline similar to that of chapter 4 will be followed. However, given certain simplifying assumptions and the nature of the reverse model, it will not be necessary to elaborate upon the discrepancy between using a dynamic or kinematic model. Discussion will then shift to an alternate controller that was explored during work at the University of California Berkeley [22]. This discussion will present the controller in an analytical fashion and show it to be a variant of pure pursuit. Though research in the field of ground vehicles driving in reverse is limited, the interested reader can find an alternative to the work presented here in [23].

5.1 Simplifying Assumption for Driving in Reverse

Here we assume that reverse driving will take place at low speeds, on the order of 2 to 3 *m/s*. This allows for the greater use of the Ackerman based vehicle models presented in section 2.1. Some analysis will be performed with a full dynamic model in order to gauge completeness, but in general the mapping equations from the pure pursuit angle to the steering angle will be done with kinematic models. Additionally, some analysis will ignore the use of actuator dynamics. Once again, because of the low speeds, it will be assumed that the dynamics of the actuator are sufficiently fast that they will not significantly impact the performance of the controller. This assumption will not be made

throughout the entirety of the chapter, but will be employed in key locations to gain insight into the nature of the pure pursuit controller in reverse as well as the mechanism-based approach that will be discussed in section 5.3.

Finally, it is assumed that the vehicle comes to a complete stop before entering reverse and that the desired path is initially well aligned with the vehicle once it begins moving. This assumption implies that the controller will only compensate for small offsets as the vehicle begins accelerating in reverse. However, this does not alleviate the need for the vehicle to compensate for step inputs once it is at cruising speed. In stating this assumption, we lend greater validity to designing around a specific speed and assume that the transient period in which the vehicle is accelerating to that speed is well behaved.

5.2 Pure Pursuit in Reverse

To begin the discussion of pure pursuit driving in reverse, a kinematic analysis provides useful information. Applying the Ackerman assumption, the velocity of the rear axle is taken as collinear with the longitudinal axis of the vehicle. Additionally, the distance between the rear axle and the position at which cross-track error will be measured is denoted by the vector quantity l_e in Figure 5.1 . Given the above definitions, the system can be modeled by the following equations:

$$\begin{aligned}\dot{\psi} &= -V \frac{\tan(\delta)}{L} \\ \dot{e} &= V \sin(\psi) + l_e \dot{\psi}\end{aligned}\tag{5.1}$$

From equation 5.1, the Laplace transform from cross-track error to steering input is given as the following:

$$\frac{\varepsilon(s)}{\delta(s)} = \frac{-V(sl_\varepsilon + V)}{Ls^2} \quad (5.2)$$

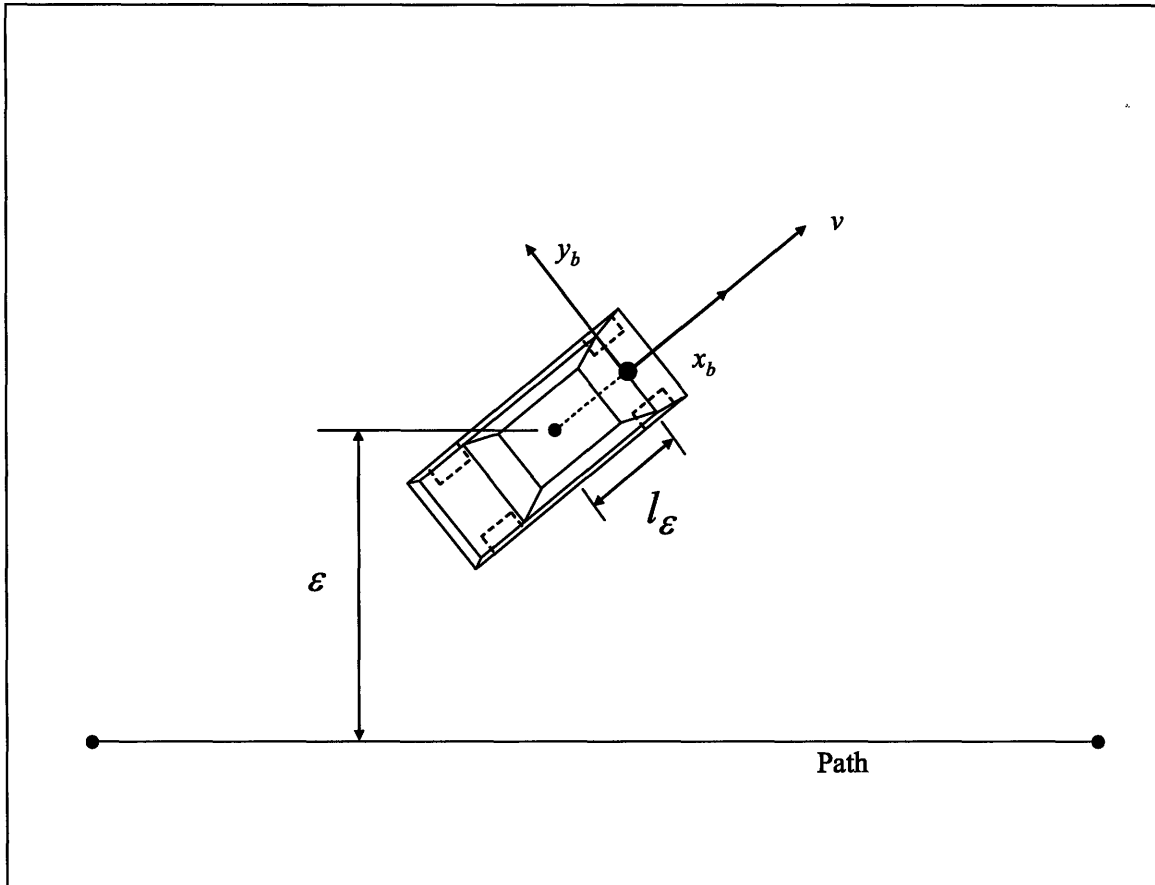


Figure 5.1: Cross-Track Error for Vehicle Moving in Reverse

From equation 5.2 it is clear that measuring cross-track error from a location between the front and rear axle produces an unstable zero in the linear system. Because pure pursuit linearizes to a PD controller on cross-track error, the virtual circle is centered at the rear axle in order to avoid this undesirable situation (i.e. set $l_\varepsilon = 0$). In the forward driving case, an unstable zero is not present and there was no penalty in shifting the center of the virtual circle to the c.g. of the vehicle. It should be noted that this result is presented using a kinematic model for simplicity; analysis of a full dynamic vehicle model in which the steering actuator dynamics are neglected and the mass moment of inertia is taken as in equation 3.25 will yield an identical result.

The linearized system can now be presented as the following:

$$\begin{bmatrix} \dot{\beta} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{y}_p \\ \dot{\delta} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} a_{R11} & 0 & a_{R12} & 0 & b_{R11} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ a_{R21} & 0 & a_{R22} & 0 & b_{R12} & 0 \\ v_x & v_x & l_r & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\omega_n^2 & -2\xi\omega_n \end{bmatrix} \begin{bmatrix} \beta \\ \psi \\ \dot{\psi} \\ y \\ \delta \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{K_{act}\omega_n^2}{K_{steer}} \end{bmatrix} y_d \quad (5.3)$$

The linear controller is the same as in equation 4.13 but with an additional negative sign to account for driving in reverse:

$$\delta_d = -2L_b \left(\frac{y_d - y}{L_a^2} - \frac{\dot{y}}{L_a v_x} \right) \quad (5.4)$$

Given that reverse driving will only be done over a very small range of speeds, the simplifying decision was made to not speed schedule the look-ahead distance L_a . Thus, the design approach is to select a look-ahead distance which is stabilizing over a range of $0 - 3.5 \text{ m/s}$ ($0 - 7.9 \text{ mph}$). This value was ultimately selected as 5 m . As with the design procedure presented for forward driving, a linear pole-zero plot can be generated for the various speeds and this can be used to evaluate the system performance. Figure 5.2, using the linearized model for Talos-I, presents the pole-zero plot for speeds from $.25 \text{ m/s}$ to 3.5 m/s in $.25 \text{ m/s}$ increments.

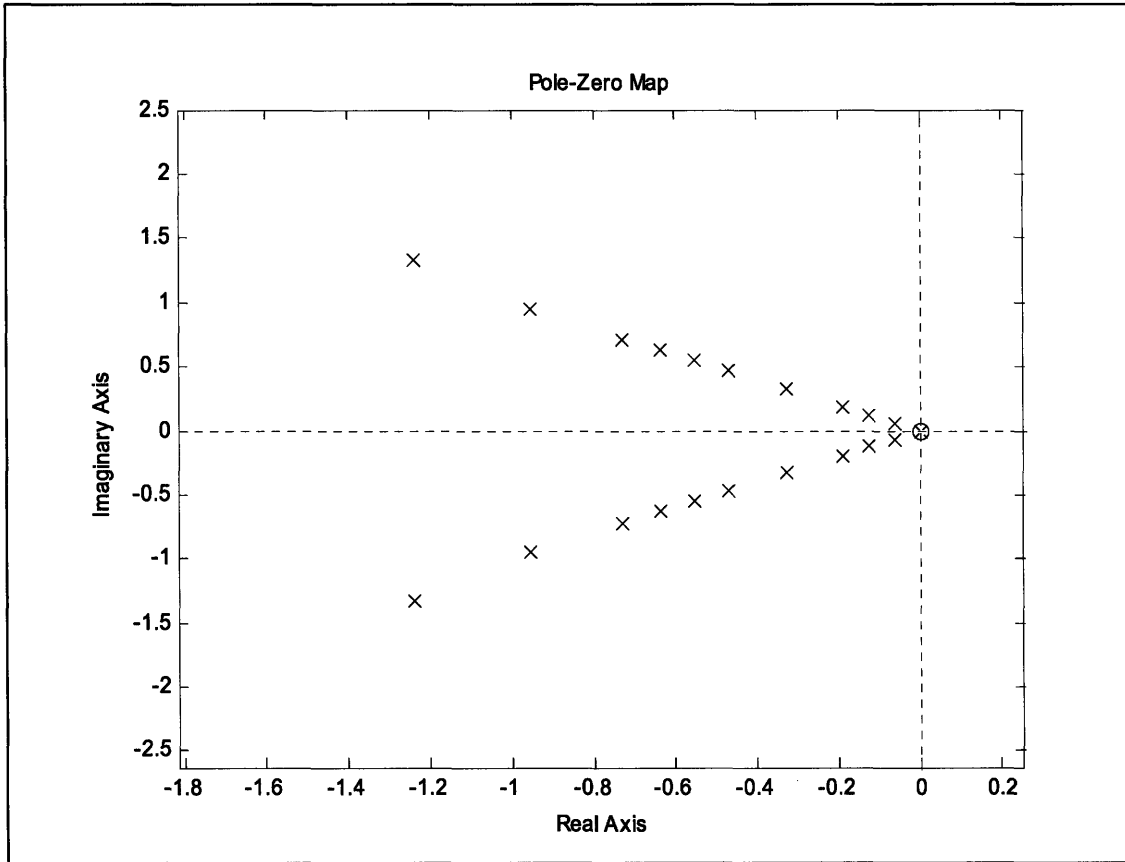


Figure 5.2: Talos-I Pole-Zero Plot for Pure Pursuit in Reverse

In this analysis, the linear model of Talos-I traveling at 3.5 *m/s* predicts a system damping of .681 and an overshoot of 5.4%. Figure 5.3 presents identical data for Talos-II. Here the linear model predicts that Talos-II will exhibit a damping of .697 with an overshoot of 4.7%.

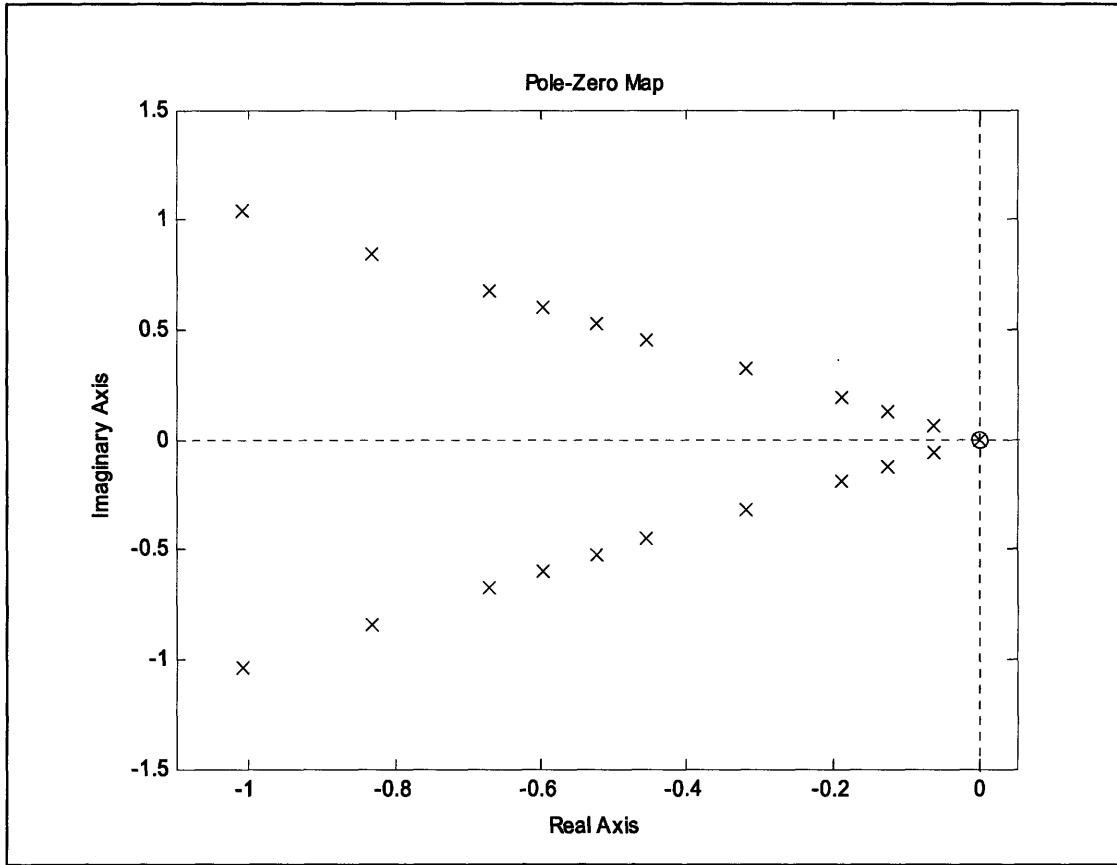


Figure 5.3: Talos-II Pole-Zero Plot for Pure Pursuit in Reverse

5.3 Simulation-based Approach to Steering Angle Selection

As with forward driving, it is possible to implement a simulation-based approach to selecting the optimal steering input. In practice, this is no different than the approach presented in section 4.5 except for the model used to forward simulate the vehicle's behavior. Here the double track model can be adjusted for reverse and implemented as the following:

$$\begin{aligned} \dot{\psi} &= \frac{-2V \tan \delta_i}{2L_b + L_{tw} \tan|\delta_i|} \\ \dot{X} &= V \cos \psi \\ \dot{Y} &= V \sin \psi \end{aligned} \tag{5.5}$$

As before, a discrete space may be utilized to search for the optimal input or a more efficient search algorithm may be employed.

5.4 Mechanism-based Approach

The controller presented in this section was first proposed in [22] and was developed by conceptualizing the control law as a mechanical linkage fixed to the car and the path. As illustrated below in Figure 5.4, the car can be thought of as having two mechanical links attached to it. The first link is attached to the path and passes through the rear axle of the vehicle. The length of this link is fixed. The second link is fixed at one end to the vehicle's front axle while the other end receives the first link. The end of the first link then slides on the second link, resulting in the second link having, effectively, a variable length. The control law is to then choose an inside wheel angle that is parallel to this second link.

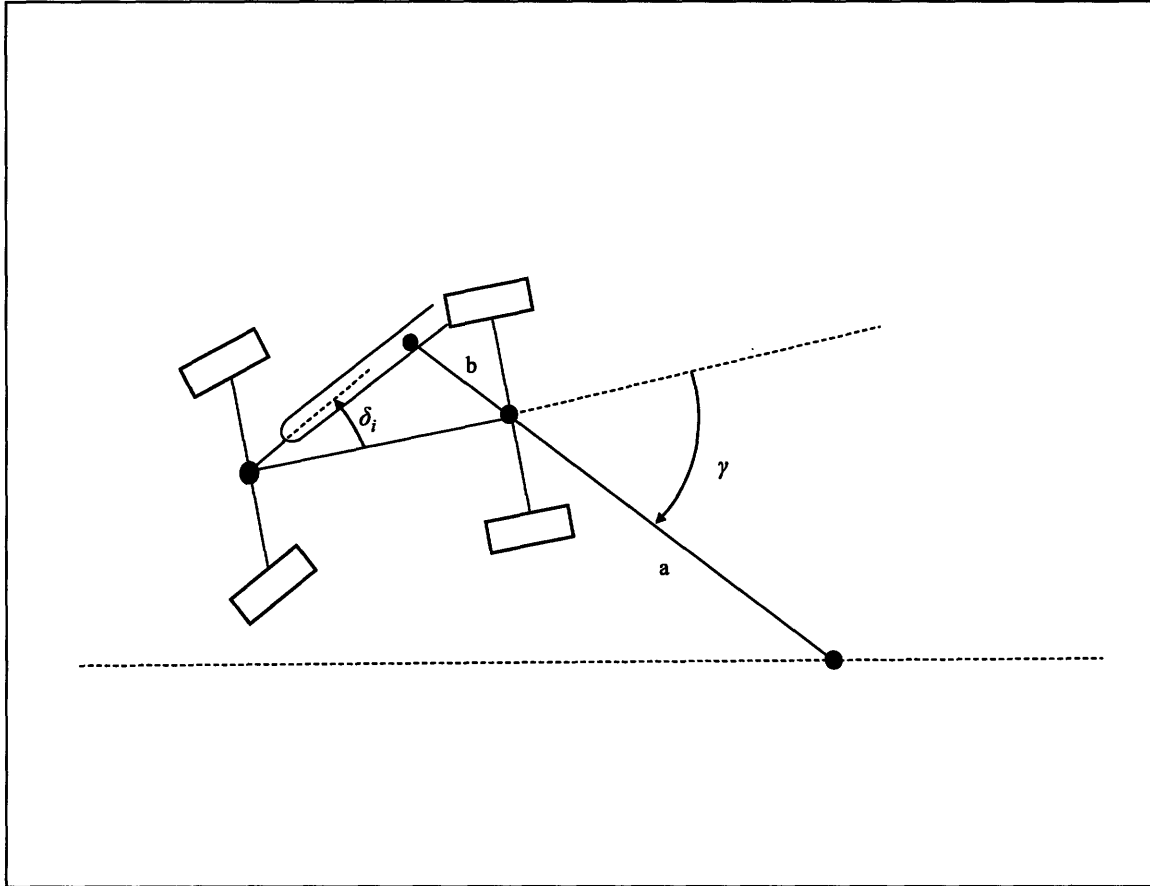


Figure 5.4: Mechanism-based Controller

Using the geometry from Figure 5.4, the control law is formulated as the following:

$$\delta_i = \tan^{-1} \left(\frac{-b \sin \gamma}{L_b - b \cos \gamma} \right) \quad (5.6)$$

In previous sections the output of the control law was denoted by δ_d and it was understood that this value would be mapped to the inside steer angle δ_i , as in section 3.3. Because the output of the mechanism-based controller does not require this mapping, the output of the control law is denoted by δ_i .

As in previous analysis, we can linearize the control law based on straight line driving. As before, the vehicle starts at the origin of the global coordinate frame with the line to be tracked located a distance y_d from the X axis. The vehicle's lateral position is, once again, denoted by the variable y . This is identical to the case illustrated in Figure 4.2

but now the vehicle is driving in reverse and the body fixed frame is located at the rear axle. The resulting linear control law is the following:

$$\delta_i = \frac{-b(y_d - y)}{a(L_b - b)} + \frac{b}{(L_b - b)}\psi \quad (5.7)$$

It should be noted that the control law has been linearized leaving the yaw as a state in the equation. Simplifying further by approximating the vehicle's yaw under small angle assumptions yields the following:

$$\delta_i = \frac{-b(y_d - y)}{a(L_b - b)} + \frac{b\dot{y}}{V(L_b - b)} \quad (5.8)$$

Through the remainder of this chapter, both linearizations will offer insightful information on the nature of the controller.

Using the simple single-track model of section 3.2, we can begin to approximate the manner in which the tuning parameters a and b affect the system's damping and natural frequency. Here the linearization presented in equation 5.7 will be used to preserve the yaw state. The system is first written in a linear state space form as illustrated below in equation 5.9. This is then converted into a transfer function relating the vehicle's vertical displacement y to the commanded input y_d .

$$\begin{bmatrix} \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & v \\ \frac{-vb}{L_b a(L_b - b)} & \frac{-vb}{L_b(L_b - b)} \end{bmatrix} \begin{bmatrix} y \\ \psi \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{vb}{L_b a(L_b - b)} \end{bmatrix} y_d \quad (5.9)$$

$$\frac{y(s)}{y_d(s)} = \frac{\frac{v^2 b}{L_b a(L_b - b)}}{s^2 + \frac{vb}{L_b(L_b - b)}s + \frac{v^2 b}{L_a a(L_b - b)}} \quad (5.10)$$

The transfer function presented in equation 5.10 then reveals that the systems natural frequency and damping are given by the following:

$$\omega_n = v \sqrt{\frac{b}{L_b a (L_b - b)}} \quad (5.11)$$

$$\xi = \frac{1}{2} \sqrt{\frac{ab}{L_b (L_b - b)}}$$

This formulation of the problem is kinematic and ignores actuator dynamics. As a result, equations 5.11 are limited in applicability to low speed maneuvers. Though equations 5.11 are useful, they do not explicitly provide the values of a and b which should be chosen for a desired damping or a desired natural frequency. However, if we specify the desired damping and the desired natural frequency as ω_d and ξ_d respectively, the required a and b parameters can be easily expressed by the following set of equations:

$$\bar{K} = \frac{4\omega_d^2 \xi_d^2 L_b^2}{v^2}$$

$$b = \frac{L_b (\bar{K} - \sqrt{\bar{K}})}{(\bar{K} - 1)} \quad (5.12)$$

$$a = \frac{4\xi_d^2 L_b (L_b - b)}{b}$$

Several things should be noted about this formulation. First, the term \bar{K} is an intermediate variable used for simplifying notation. Second, solving for b requires imposing the constraint that b is greater than 0 but less than L_b . Referring to equations 5.11, the natural frequency and the damping of the system would be imaginary if the latter of these constraints was not imposed. These constraints are also fairly intuitive given the mechanism-based justification for the control law. Finally, for completeness, it should be noted that it is also necessary for \bar{K} to be greater than 0. This ensures that b has a positive value.

If a performance variable that dictates the required ω_d and ξ_d is specified, then the appropriate a and b parameters may be approximated. For this design procedure, the system settling time is selected as this variable. The settling time is related to the desired natural frequency and damping through the standard equation [18]:

$$t_s = \frac{4}{\xi_d \omega_d} \quad (5.13)$$

The design process is to then select a settling time and assume a nearly critical damping ($\xi_d = .9$). This defines ω_d through equation 5.13. Equations 5.12 can then be utilized to find the required a and b to meet these specifications. Here a warning should be issued in terms of selecting the settling time as a design parameter. If the settling time is chosen too small, the actuator dynamics which were originally ignored may become significant. Specifically, the settling time should be selected such that the systems natural frequency is far less than that of the actuator (refer to section 3.7 for details on modeling the steering actuator) in order to justify the original assumptions of this analysis.

Here it is noted that these calculations should be done at the nominal cruising speed. For instance, if the vehicle is expected to operate between 2 m/s and 4 m/s while in reverse, 2 m/s should be the design speed. Evaluating equations 5.11, for a given a and b combination, it is clear that the damping will remain constant and that the natural frequency will increase with v . As a result, referencing equation 5.13, the settling time will decrease with greater speed. This ensures that the settling time is at or below the specified value for the operational range. If, however, 4 m/s is chosen as the design point, the settling time will increase with a reduction in v . As a result, the specified settling time will be exceeded if the vehicle is traveling at any speed lower than 4 m/s . As an alternative, a fixed settling time may be selected and the controller gains may be

scheduled with v , as opposed to selecting a fixed set of gains for the entire operating range.

If the linearization presented in equation 5.7 is used, this mechanism-based controller may be compared to the pure pursuit controller presented in section 4.2, repeated below for convenience.

$$\delta_d = -\frac{2L_b}{L_a} \left(\frac{-\dot{y}}{v} + \frac{(y_d - y)}{L_a} \right) \quad (5.14)$$

Comparing this equation to equation 5.7, it becomes apparent that there is a strong relationship between the two controllers. Indeed, the parameter a , referring to Figure 5.4, is analogous to a look-ahead distance. Thus, if a is set equal to L_a , the two controller are identical if b is defined as below.

$$b = \frac{2L_b^2}{(a + 2L_b)} \quad (5.15)$$

This is an interesting result in that it casts the mechanism-based controller as a variant on pure pursuit (pure pursuit under kinematic assumptions about vehicle performance). Indeed, one may interpret the mechanism-based controller as pure pursuit with the added flexibility of an extra tunable parameter.

Following the pattern for analysis set forth in chapter 4, it is now reasonable to validate the above by incorporating actuator dynamics and a dynamic model. The design speed was selected as 2 *m/s* with a settling time of 5.5 seconds and damping of .92. The resulting parameters a and b for Talos-I were then found to be 5.1 *m* and 1.67 *m* respectively. The a and b parameters for Talos-II, calculated under the same specifications, were 5.1 *m* and 1.9 *m* respectively. The resulting pole-zero plots for Talos-I and Talos-II are presented below in Figure 5.5 and Figure 5.6 respectively.

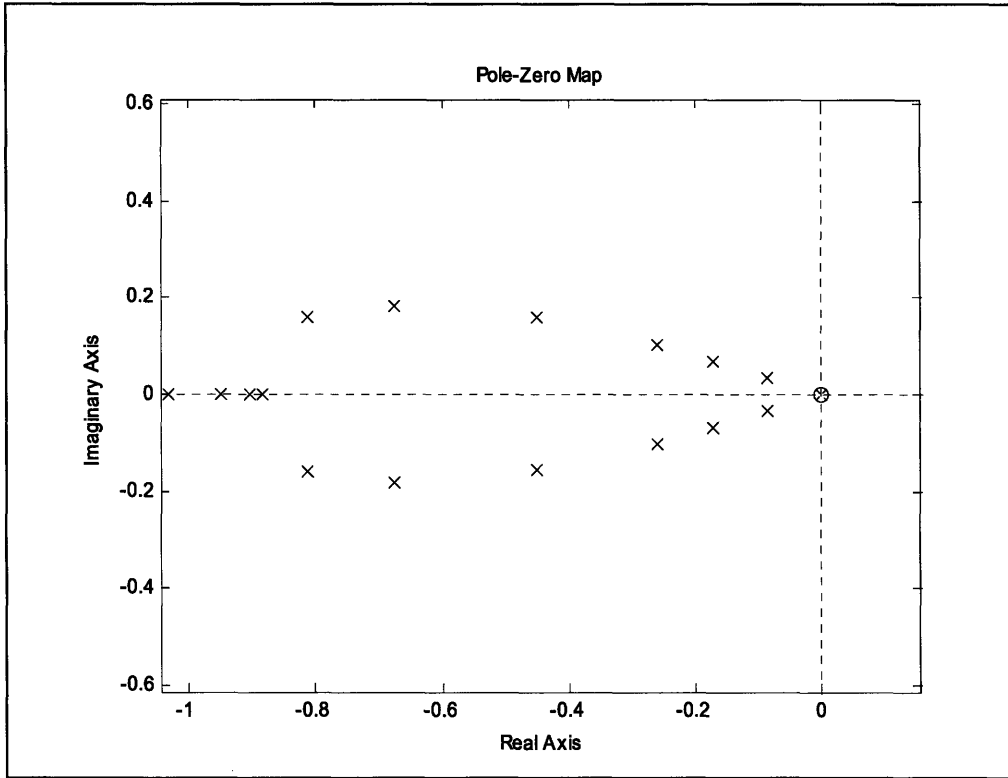


Figure 5.5: Talos-I Pole-Zero Plot for Mechanism-based Controller

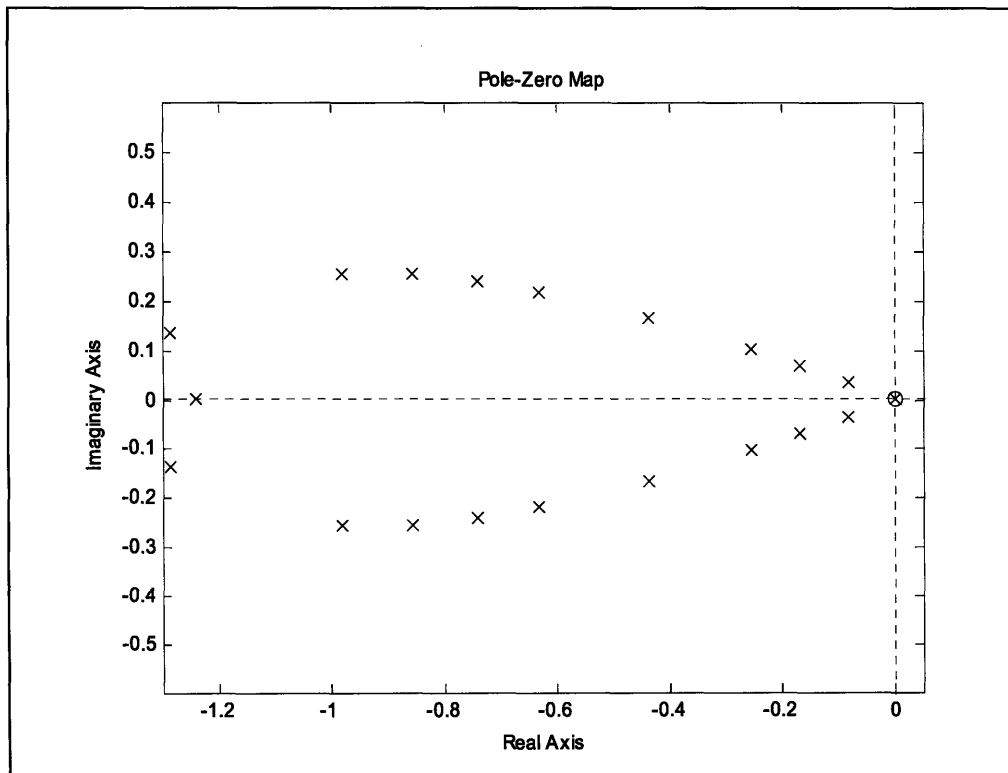


Figure 5.6: Talos-II Pole-Zero Plot for Mechanism-based Controller

Here the mechanism-based controller predicts a significant increase in performance over pure pursuit in reverse. Specifically, at the full range of speeds, system damping for both Talos-I and Talos-II is greater than .9 with an associated overshoot of less than .1% .

Several precautions must be taken when implementing this controller. First, if the controller was realized by a physical linkage as opposed to a software based system, the cross-track error between the vehicle and the desired path could not exceed the value of the parameter a . As a physical realization of the controller is not practical, a software approach is assumed in which the cross-track error could be greater than a . In such a case, the value of a is temporarily set to the value of the cross-track error. Drawing an analogy to the pure pursuit controller, the point at which the virtual mechanism is affixed to the path is equivalent to the pure pursuit goal point. It is essential that a goal point always be defined, otherwise the controller will have discontinuities in which it produces no response or an error. Temporarily increasing the parameter a to alleviate this condition is a reasonable decision as, referring to equation 5.11, a larger a value may degrade path tracking but will increase the system damping and reduce the system's natural frequency. Thus, this approach will not induce instability.

As illustrated in [22], the mechanism can reach a locked position in which the angle γ is equal to $\pm 180^\circ$. If the mechanism is realized physically, this is a failure mode. However, in software, this can be easily combated by limiting the angle γ to the range of $\pm 90^\circ$. Mathematically this can be expressed as the following:

$$\delta_i = \left\{ \begin{array}{l} \tan^{-1}\left(\frac{-b \sin(\gamma)}{L_b - b \cos(\gamma)}\right) \text{ if } -90^\circ < \gamma < 90^\circ \\ \tan^{-1}\left(\frac{-b}{L_b}\right) \text{ if } 90^\circ \leq \gamma < 180^\circ \\ \tan^{-1}\left(\frac{b}{L_b}\right) \text{ if } -180^\circ < \gamma \leq -90^\circ \end{array} \right\} \quad (5.16)$$

However, this formulation fails to handle the case in which γ is exactly $\pm 180^\circ$. In such a situation, the vehicle needs to execute a complete U-turn in order to realign itself with the path, yet a specification for turning either left or right is not clear. In such a case, the decision is made for the controller to choose right or left based on the last control signal. More precisely, given that δ_{i-old} is the last control signal and $\gamma = \pm 180^\circ$, then δ_i is given by the following:

$$\delta_i = \tan^{-1} \left(\frac{\text{sign}(\delta_{i-old})b}{L_b} \right) \quad (5.17)$$

If a previous control signal is not defined, such as in the case of a new path having just been sent by the higher level planner, then the controller should issue an error and fail to execute. In essence, the controller has no basis for making a logical decision about the correct action to execute. The purpose of being able to handle values of $|\gamma| > 90^\circ$, however, is the preservation of the ability for the planner to generate such paths based on its understanding of how the controller will respond. For instance, if it was desired for the vehicle to do a U-turn in reverse (though this would not presumably happen often) then the planner could create a precise clothoidal path or it could search for the linear path with $|\gamma| > 90^\circ$ such that the vehicle would produce the desired motion. This is a powerful capability that represents one of the key benefits of pure pursuit-like control schemes for ground vehicles.

5.5 Chapter Conclusions

This chapter investigated the problem of stabilizing a vehicle driving in reverse by building on the foundation presented in chapter 4. First, the pure pursuit controller was

investigated as a potential candidate. Here, this analysis made key assumptions concerning the operational range of speeds to simplify the design process. Specifically, it was assumed that both a constant look-ahead distance and an Ackerman based mapping from the pure pursuit curvature to steering angle would be sufficient. Developing a pole-zero plot of the system over .5 to 3.5 *m/s* then indicated that the system was stable and reasonably well damped. Second, a brief description of a simulated-based controller was provided. This approach was nearly identical to that presented in section 4.5 with a trivial difference in the simulation model. Finally, an alternative controller was presented based on the concept of a mechanical linkage. Analysis showed that this controller offered, through the specification of a settling time, a simple means for approximating the required tuning parameters. These parameters were then validated on a full dynamic model which included actuator dynamics.

6 Results

This chapter will present the results of implementing the controllers presented in chapters 4 and 5. These results will include pure pursuit for forward driving, the simulation-based controller for forward driving, pure pursuit for driving in reverse, the mechanism-based controller for driving in reverse, and the simulation-based controller for driving in reverse. In order to evaluate these controllers, results were collected with MIT's race vehicle Talos-II traversing pre-defined courses. In the specifications given by DARPA, a qualifying test was required prior to official entry into the Urban Challenge. The rules governing this qualifying test outlined the creation of a *test-site course*; for simplicity, those rules are not provided here, but instead Figure 6.1 presents the official test-site course designed for the MIT qualifying event. Many of the results that follow were collected from this course. Additional results were then collected using the figure eight test path presented in Figure 6.2. It should be noted prior to evaluating these results that it was necessary to low-pass filter all of the steering command signals. This was a result of noise from the velocity estimate, which impacted the calculation of the desired steer angle as well as the magnitude of the look-ahead distance. For Talos-II, the cutoff frequency for the low-pass filter was 1.67 Hz . This filter significantly reduced chatter and produced no compromise in performance.

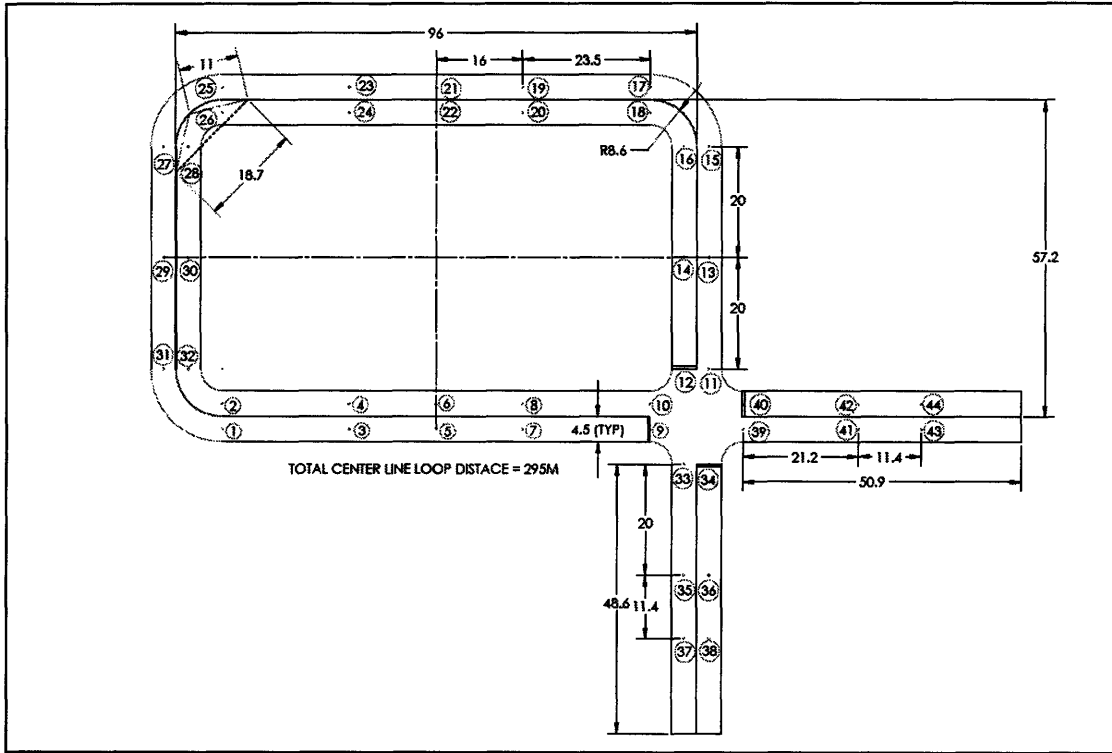


Figure 6.1: MIT Test-Site Course

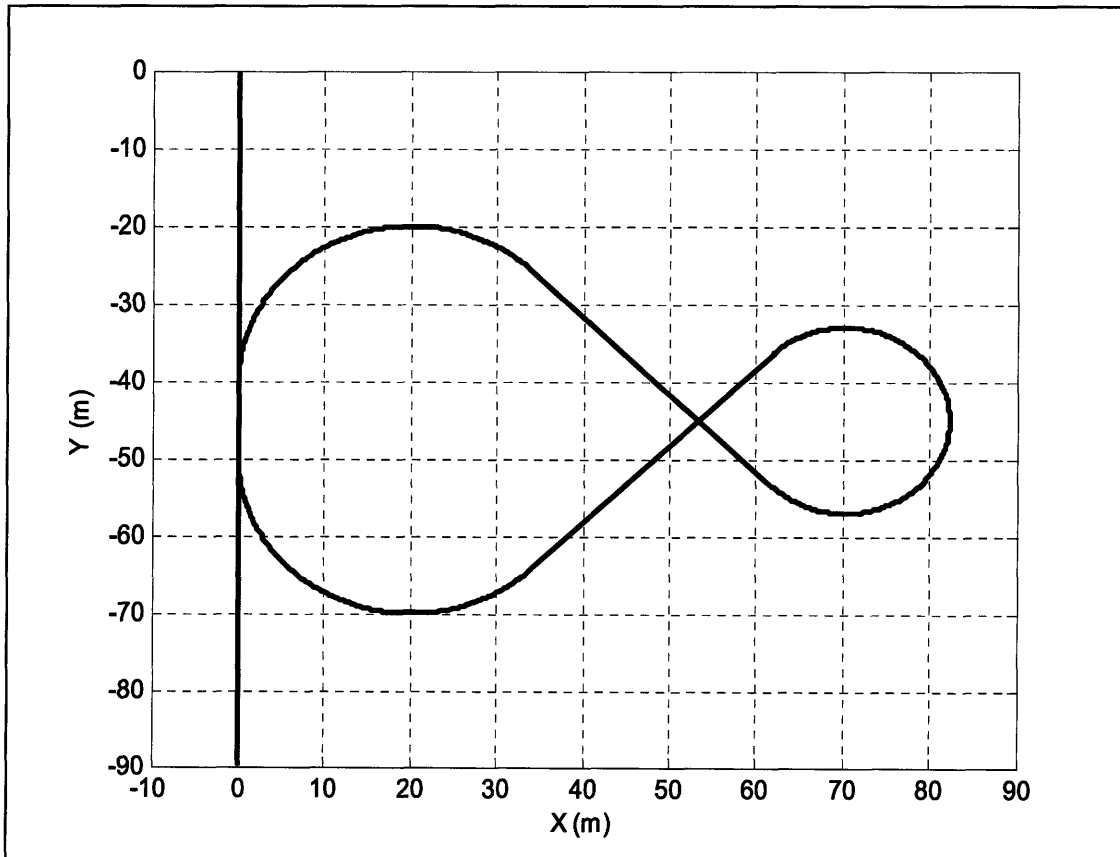


Figure 6.2: Figure-Eight Controller Evaluation Course

6.1 Pure Pursuit for Forward Driving

One of the key features of the pure pursuit algorithm is the ability to track paths which have discontinuous first and second derivatives. To evaluate this capability, the test site course of Figure 6.1 was approximated by a simple rectangular path. The results of traversing this coarse path at 4 *m/s* and 6 *m/s* are presented below in Figure 6.3. These results use the look-ahead tuning presented in section 4.3.

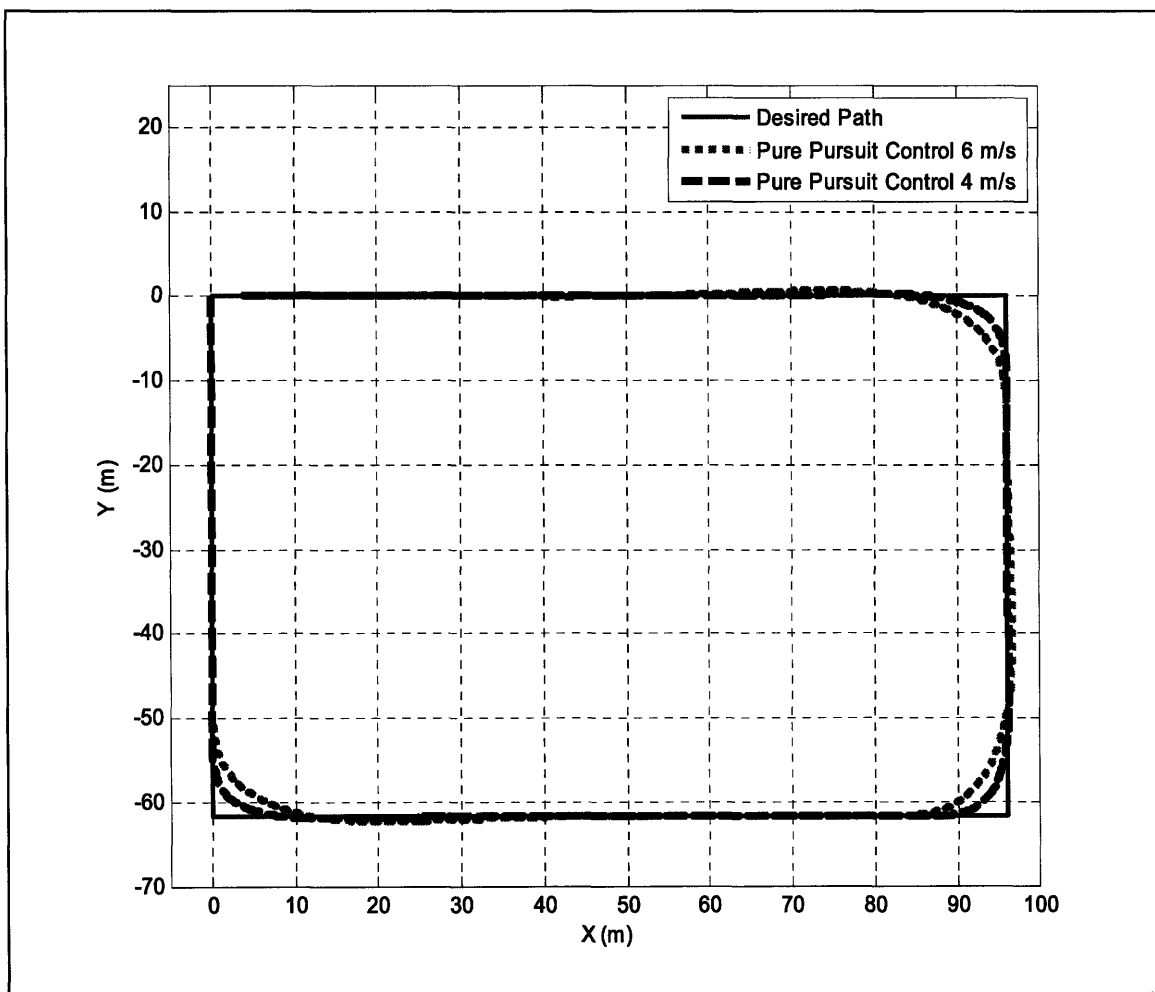


Figure 6.3: Talos-II Tracking a Coarse Rectangular Path

It is clear from this figure that the pure pursuit controller, by previewing the path ahead, is able to smoothly transition through the sharp 90° turns. Moreover, this figure clearly illustrates the impact that the look-ahead distance has on path tracking. At lower

speeds the look-ahead distance is smaller and Talos-II is much closer to the corner before a turn is initiated. Conversely, at higher speeds the look-ahead distance is larger and Talos-II will begin turning earlier, which results in a more gradual turn. To help quantify this feature, Figure 6.4 presents the desired steer angle specified by the pure pursuit controller as a function of the total distance traveled and Figure 6.5 presents the look-ahead distance, also as a function of the total distance traveled. Here it should be noted that the fluctuations in the look-ahead distance are a result of the noise associated with Talos-II's speed estimate. As expected, these figures show that the turn is much sharper at lower speeds than at higher speeds. This is a natural result as sharp turns are inherently safer when negotiated at lower speeds. Indeed, this is the driving force behind speed scheduling the look-ahead distance.

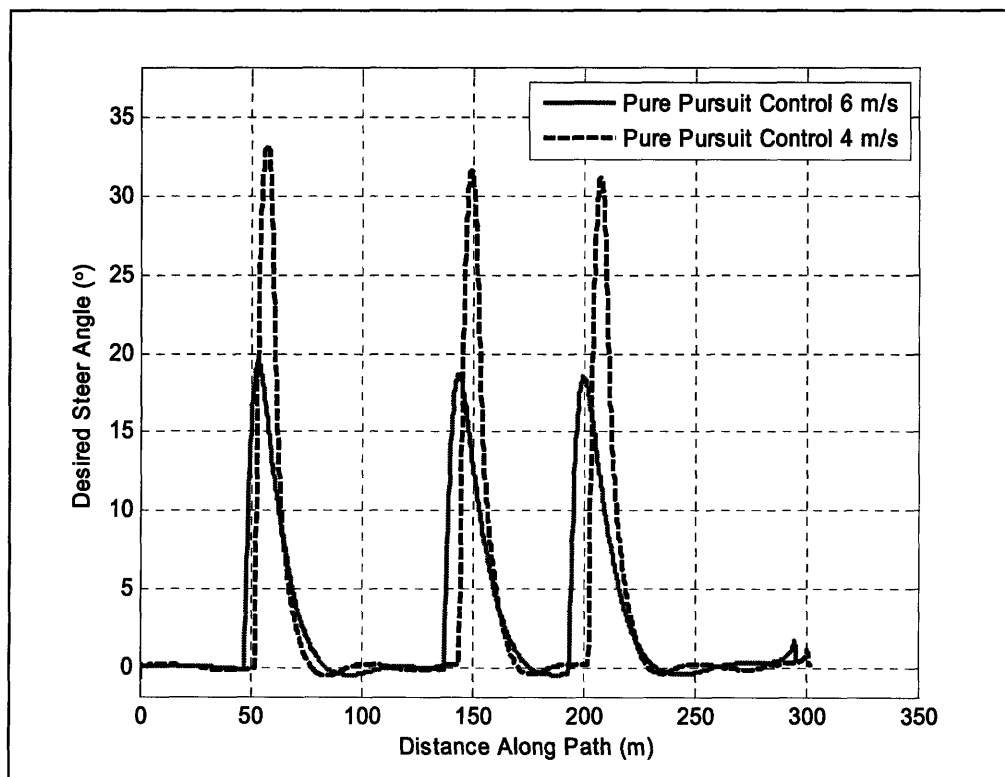


Figure 6.4: Desired Steering Angle for Traversing Rectangular Path

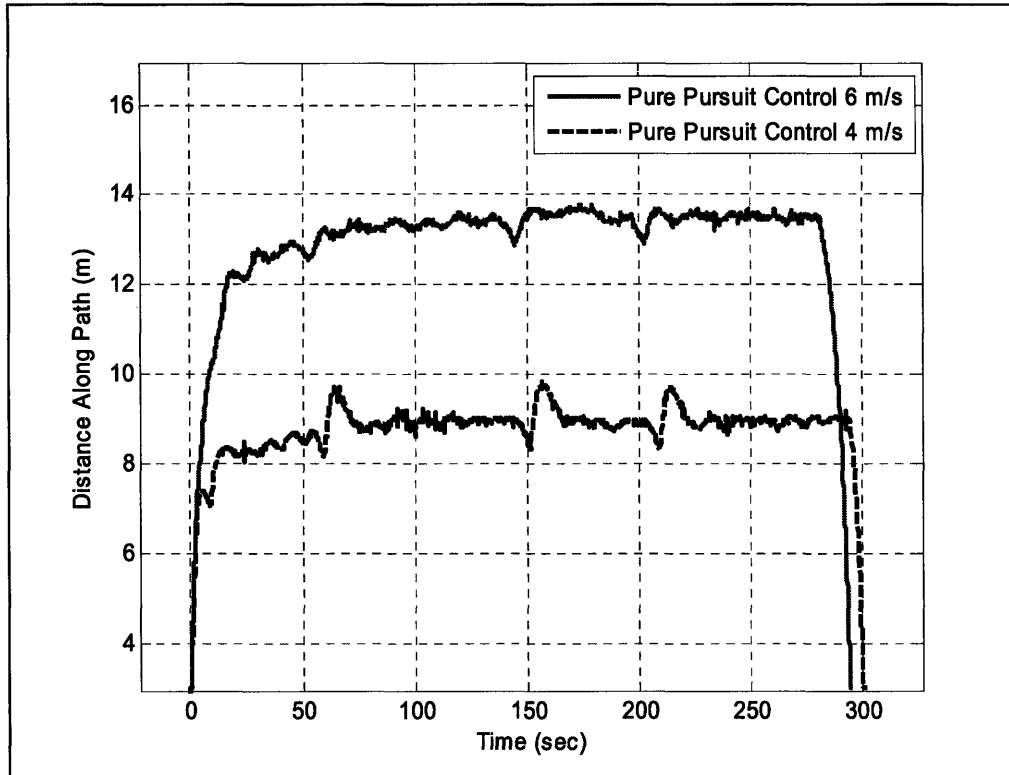


Figure 6.5: Look-Ahead Distance for Traversing Rectangular Path

The cross-track error associated with these runs is now presented below in Figure 6.6. Here the peaks in the cross-track error occur as Talos-II enters the corners of the specified rectangular path. As expected, at higher speeds the peak cross-track error is larger as a result of a more gradual turn. Moreover, as Talos-II exits a turn there is some nominal overshoot before it returns to a steady-state tracking of the path. At 6 *m/s* this overshoot is almost 60 *cm* while at a speed of 4 *m/s* the peak overshoot is only on the order of 25 *cm*. Additionally, at 4 *m/s* the root-mean-squared error (RMS) is .335 *m* while at 6 *m/s* the RMS error is .706 *m*. These results clearly illustrate the coupling between vehicle speed and tracking performance. Finally, it is also worth noting that the algorithms outlined in Chapter 2 yield a smooth specification for the cross-track error.

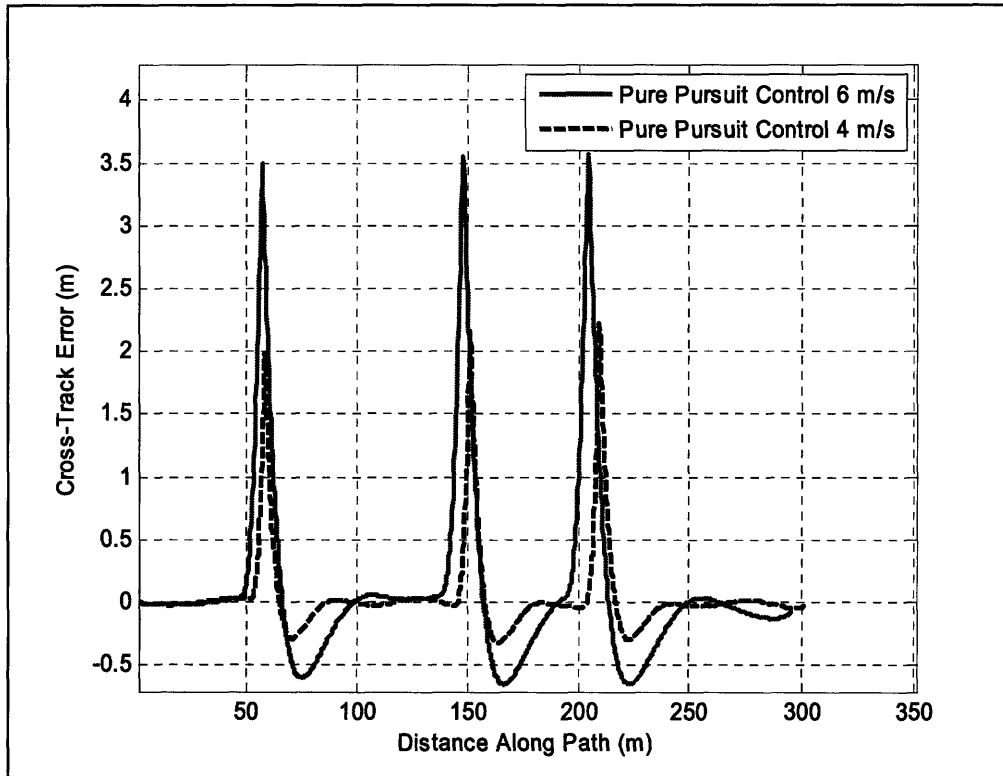


Figure 6.6: Cross-Track Error for Traversing a Rectangular Path

Attention is now given to the pure pursuit controller tracking smooth paths. Referring to Section 2.1, a smooth path is a dense discretization of a path having continuous first and second order derivatives. For this evaluation, the course of Figure 6.2 was used. Figure 6.7 presents results for Talos-II traversing the figure eight course at speeds of 2 m/s and 4 m/s . The associated cross-track errors for these runs are presented in Figure 6.8. The RMS errors associated with these runs were $.272\text{ m}$ and $.497\text{ m}$ respectively. Here all the errors associated with tracking the path once again increase with speed. Additionally, cross-track error arises as the pure pursuit controller transitions from one constant curvature path segment to another constant curvature path segment. The magnitude of this transitional cross-track error is also a function of speed. If better tracking is desired, the simple solution is to then vary the speed commands such that Talos-II would slow during these transitions.

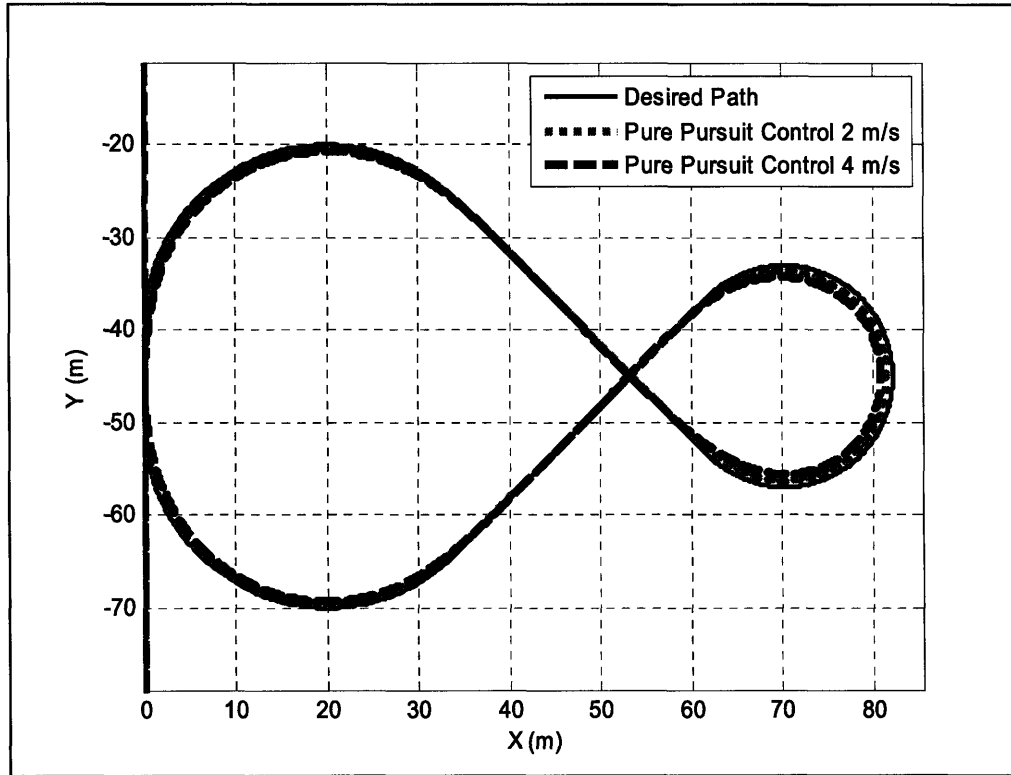


Figure 6.7: Talos-II Traversing Figure Eight Course with Pure Pursuit Controller

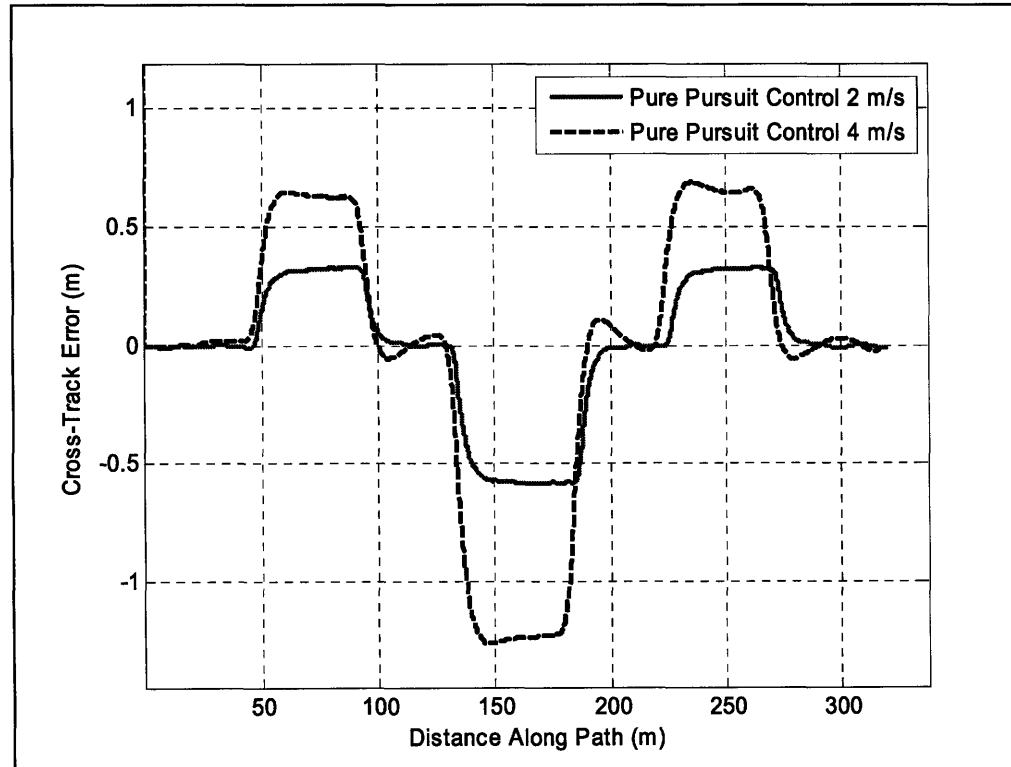


Figure 6.8: Cross-Track Error for Traversing Figure Eight Course

Though not the focus of this work, the selection of a desired speed (s_d) significantly impacts the performance of the pure pursuit controller. A reasonable approach to this problem is to assume that the path curvature (κ_{path}) should be less than or equal to $1/nL_a$, where n is a tunable factor of safety. Substituting in the speed dependence of L_a yields the following equation:

$$\kappa_{path} \leq \frac{1}{nK_{La}s} \quad (6.1)$$

The desired speed may then be found by solving the above equation for s , as in equation 6.2 below.

$$s_d = \frac{1}{n\kappa_{path}K_{La}} \quad (6.2)$$

In implementing this method, the path curvature should be based on the position of the goal point along the path, as opposed to the projected position of the vehicle onto the path. This will incorporate a certain level of feedforward that will provide the time necessary for the vehicle to slow down before it transitions from one path curvature to the next. To further illustrate the need to select the desired speed carefully, Figure 6.9 presents results for Talos-II traversing the figure eight course at a speed of 6 *m/s* while Figure 6.10 presents the associated cross-track error. Here the deviation from the path reaches a peak value of 2 *m* with an RMS error of .872 *m*. Though a less conservative look-ahead distance could be selected, this error should be interpreted as an indicator of poor speed selection as opposed to poor controller performance.

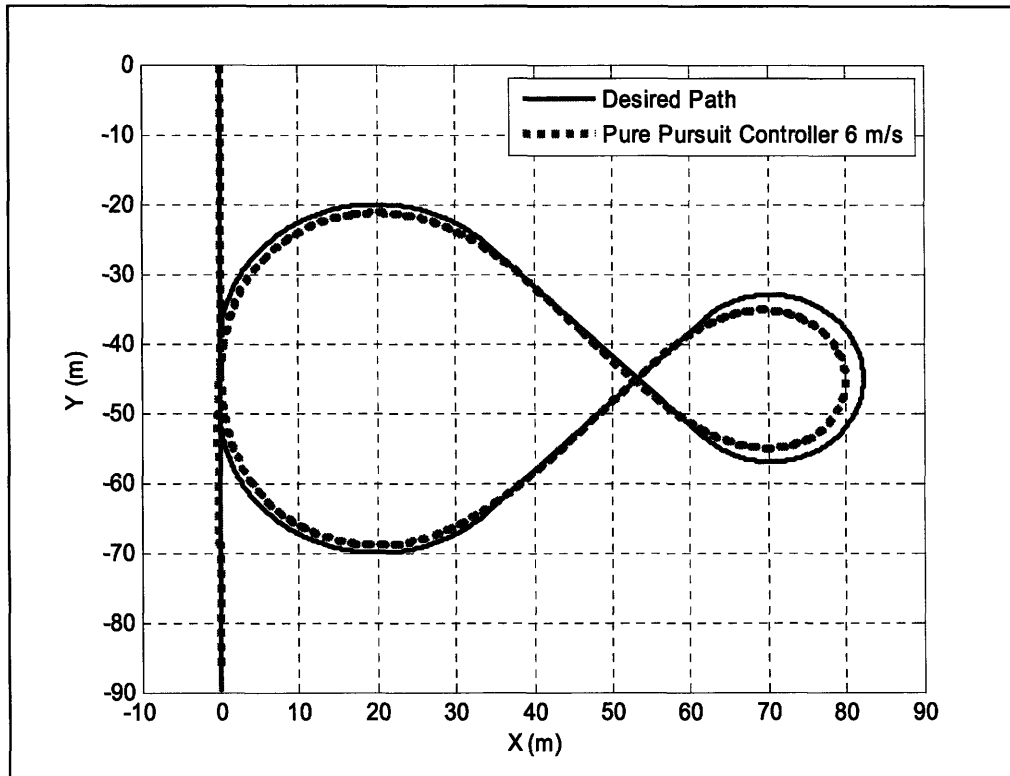


Figure 6.9: Talos-II Traversing Figure Eight Course at 6 m/s

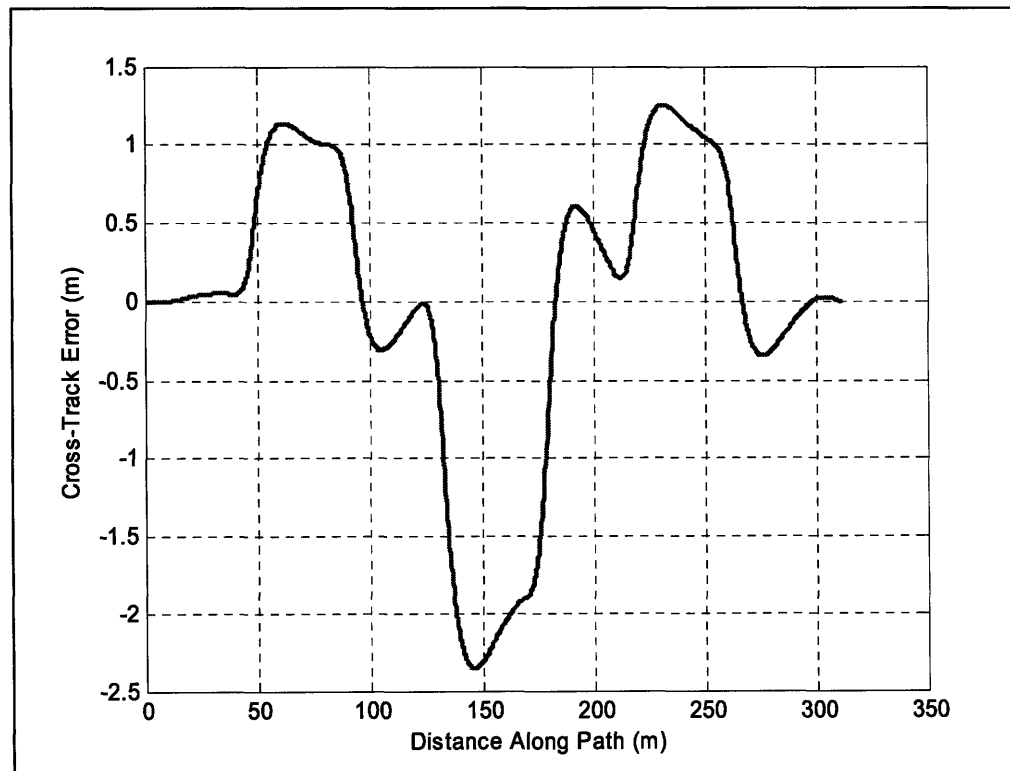


Figure 6.10: Cross-Track Error for Traversing Figure Eight Course at 6 m/s

To gauge the general robustness of the pure pursuit controller, step inputs were commanded at a speed of 10 *m/s*. These experiments were done by commanding Talos-II to track a straight line until it reached the steady-state speed, at which point the desired path was shifted to the left or right. For safety, the size of the step input was incrementally increased from 1 to 10 *m*. The results of this experimentation for a 10 *m* step are presented in Figure 6.11 and Figure 6.12. In general, this test was highly successful with less than a 5% overshoot. This result suggests that the actual system has slightly greater damping than expected based on the models of Chapter 4. Model accuracy would, of course, benefit from precise knowledge of many parameters, such as, cornering stiffness and mass distribution.

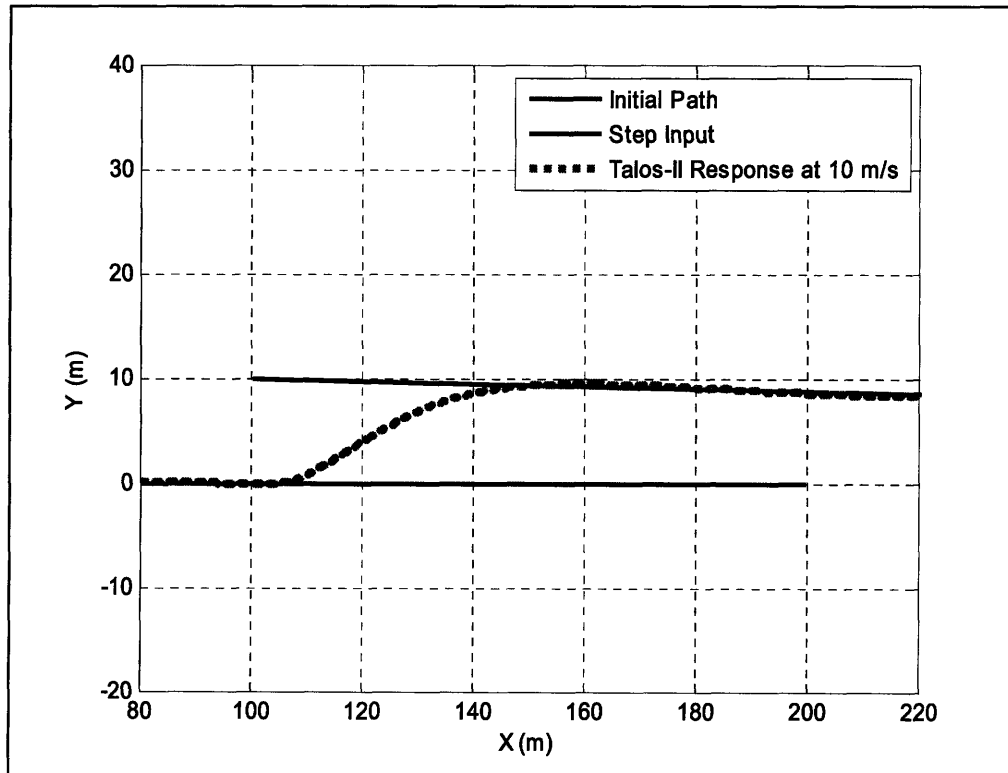


Figure 6.11: 10 m Step in Desired Path

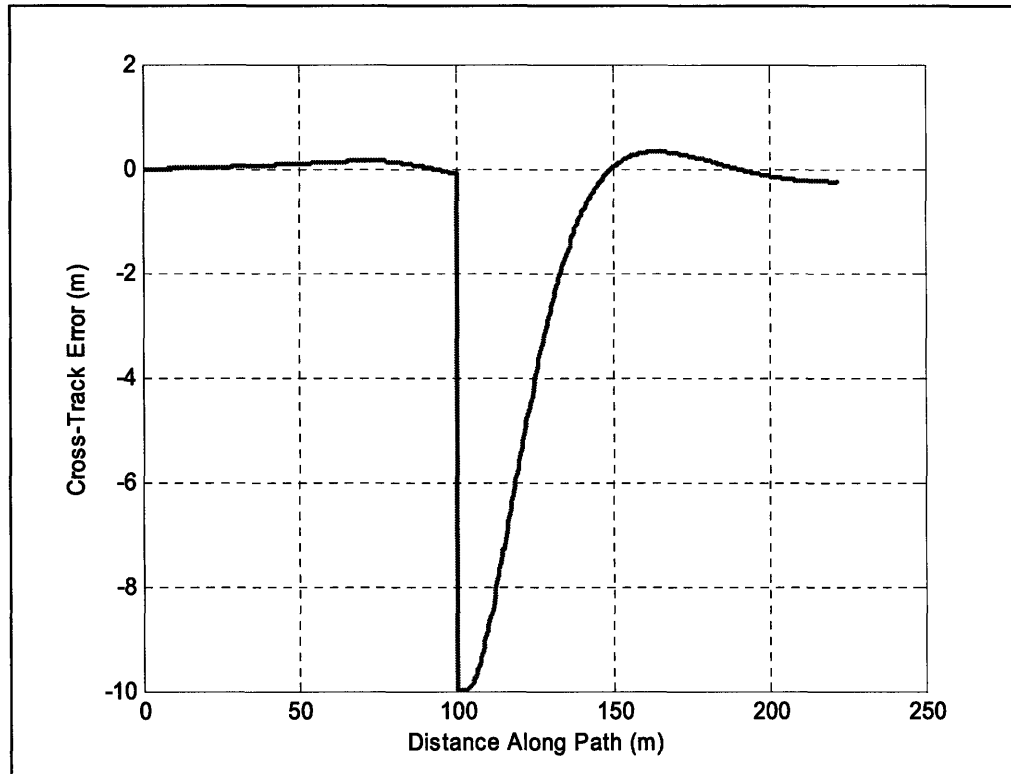


Figure 6.12: Cross-Track Error for 10 m Step Input

6.2 Simulation-based Control for Forward Driving

The simulation-based controller (SBC for short) offers slightly different performance characteristics than the traditional pure pursuit controller. Presented in Figure 6.13 are results for Talos-II traversing the figure eight course at 2 m/s and 4 m/s . The associated cross-track errors and desired steer angle are than presented in Figure 6.14 and Figure 6.15 respectively. For these runs the RMS errors were $.116\text{ m}$ and $.374\text{ m}$ respectively.

The simulation-based controller shows a clear improvement in tracking during the turns of the figure eight course. However, as the simulation-based controller transitions to the straight-aways, it converges more slowly than the traditional pure pursuit controller. Indeed, the simulation-based controller exhibits its peak deviations during these regions. Additionally, the simulation-based controller, at low speeds, produced a noisy command signal; this was a result of the noise associated with differentiating the steering wheel

encoders (refer to Section 4.5 for greater detail). Despite these factors, the RMS errors for the simulation-based controller were slightly smaller than those associated with the pure pursuit controller.

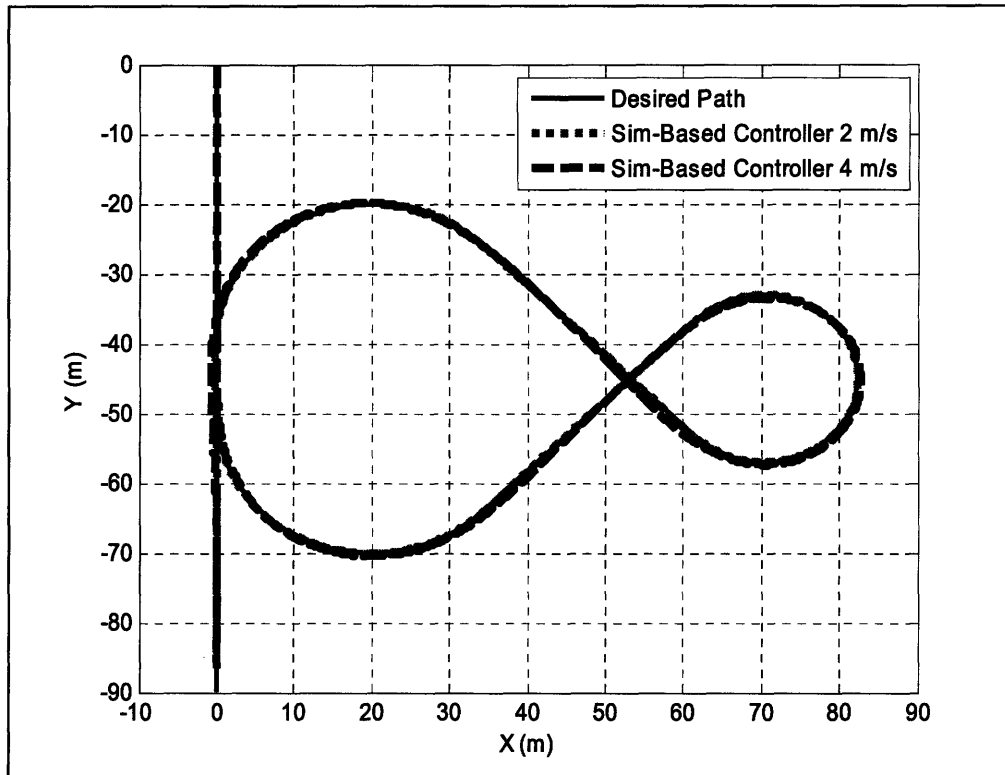


Figure 6.13: Talos-II Traversing Figure Eight Course with SBC

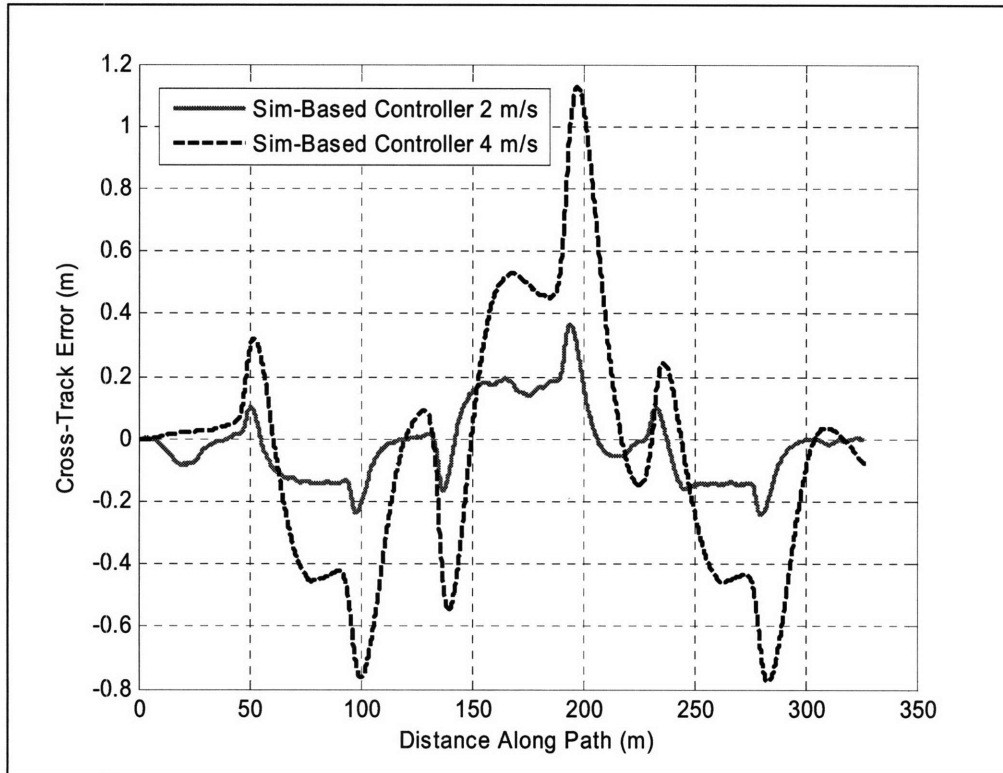


Figure 6.14: Cross-Track Error for Figure Eight Course with SBC

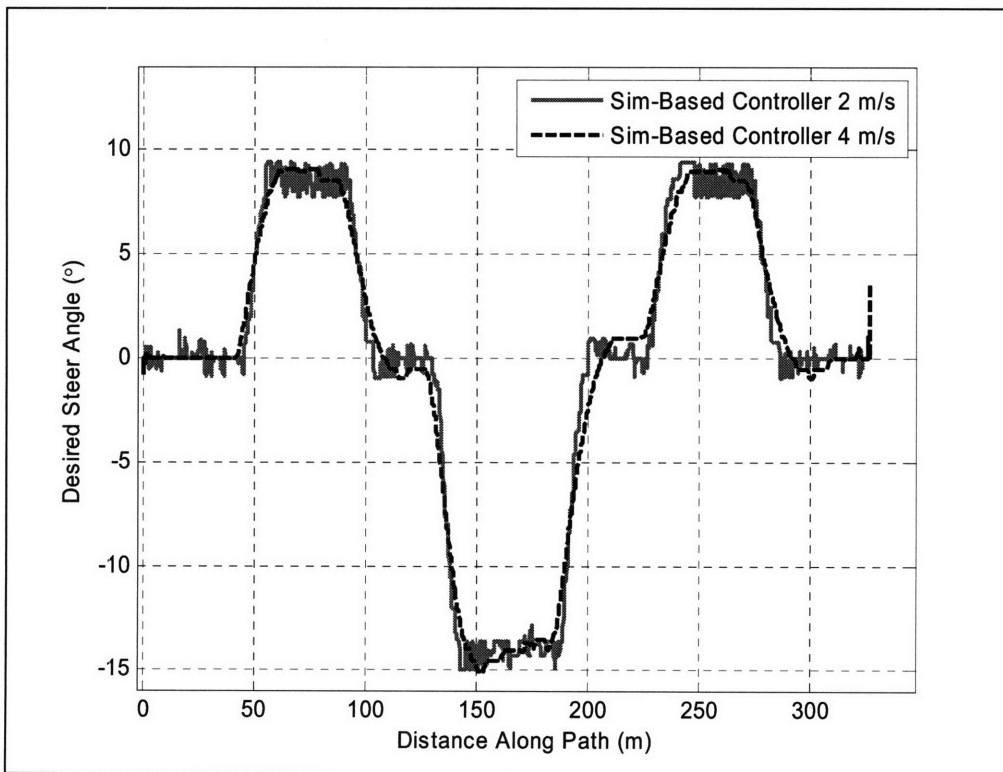


Figure 6.15: Desired Steer Angle for Figure Eight Course with MPC

Results for the simulation-based controller tracking the figure eight course at 6 m/s second are now presented in Figure 6.16 with the associated cross-track error presented in Figure 6.17. For this run the RMS error was .841 m. These figures more dramatically illustrate the overshoot associated with the simulation-based controller as Talos-II enters the course straight-aways.

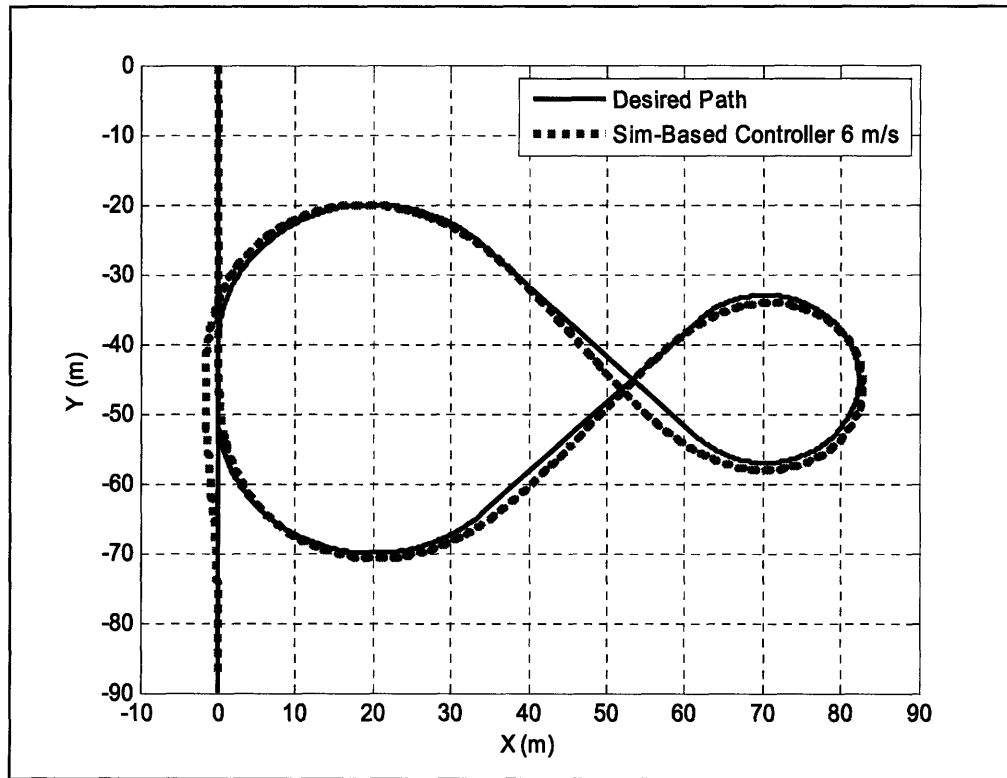


Figure 6.16: Talos-II Traversing Figure Eight Course at 6 m/s with SBC

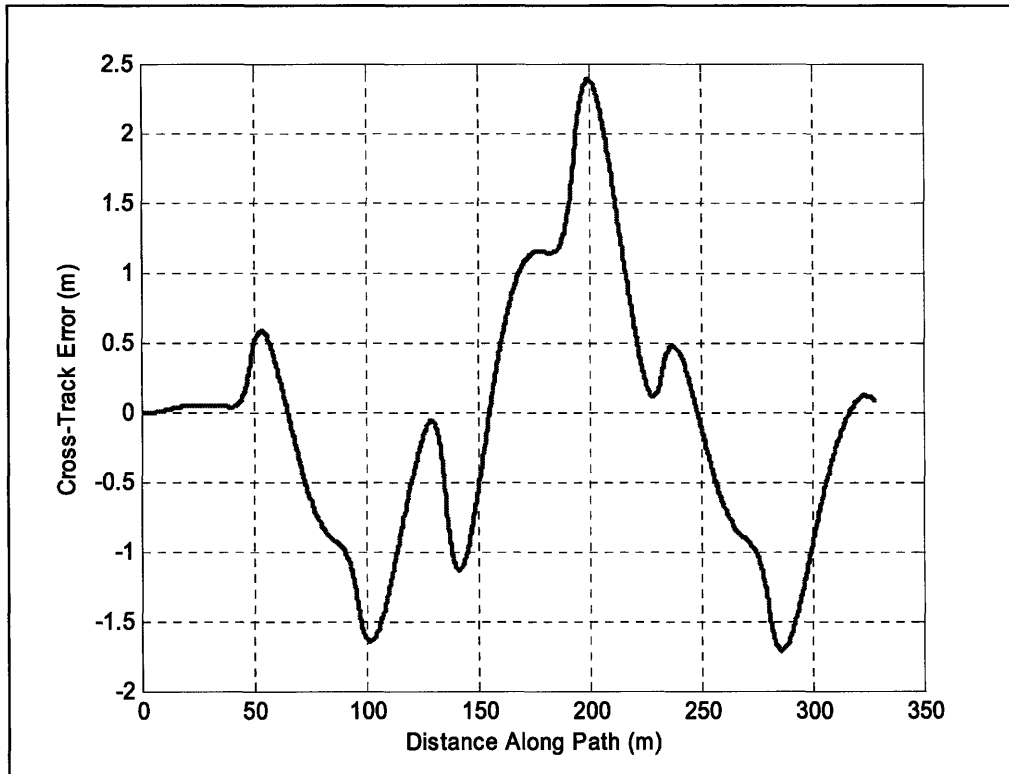


Figure 6.17: Cross-Track Error for Figure Eight Course at 6 m/s with SBC

As with the pure pursuit controller, the simulation-based controller is capable of tracking non-smooth paths. Presented in Figure 6.18 are results for the simulation-based controller tracking a rectangular path at 2 *m/s* and 4 *m/s*. The associated cross-track errors are presented in Figure 6.19. As compared to the pure pursuit controller, the simulation-based controller exhibits significantly greater overshoot as it exits the turns. At 6 *m/s* the pure pursuit controller overshoot is on the order of 60 *cm* while the simulation-based controller exhibits a deviation of approximately 2.5 *m*. Additionally, the RMS errors for these runs were .632 *m* (2 *m/s*) and 1.107 *m* (4 *m/s*) respectively, significantly larger than the errors associated with identical runs of the pure pursuit controller. These performance differences are most likely do to modeling error; the kinematic model does not capture system dynamics and the actuator model is a continuous approximation of a discrete system. These errors are seemingly more pronounced with non-smooth paths.

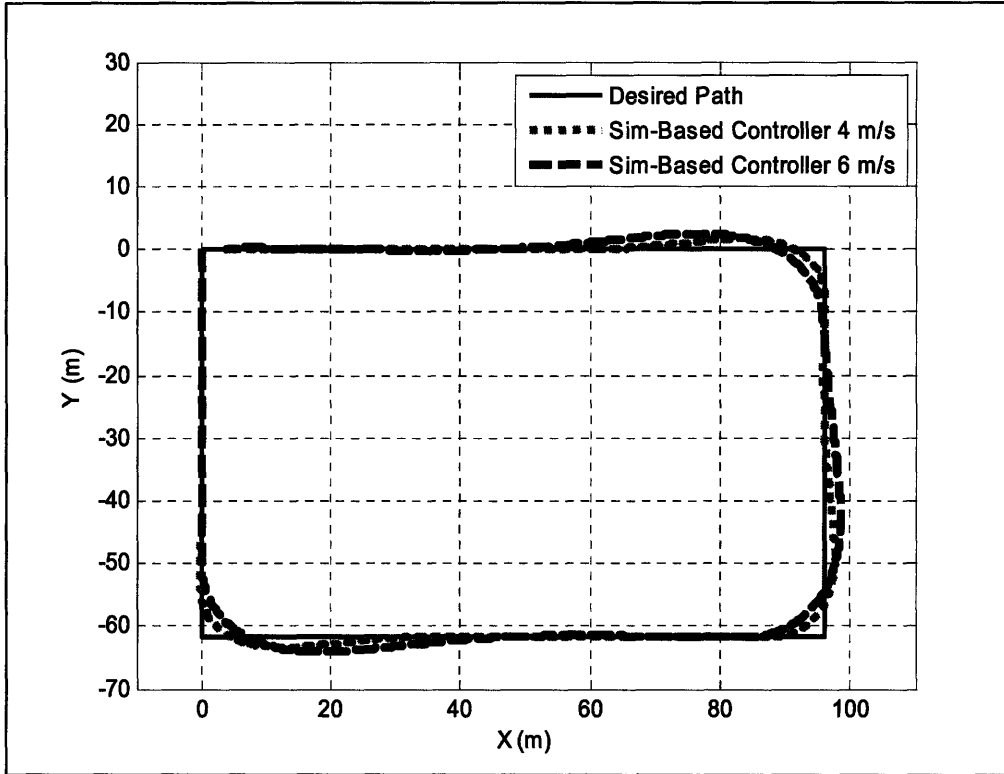


Figure 6.18: Talos-II Traversing Rectangular Path with SBC

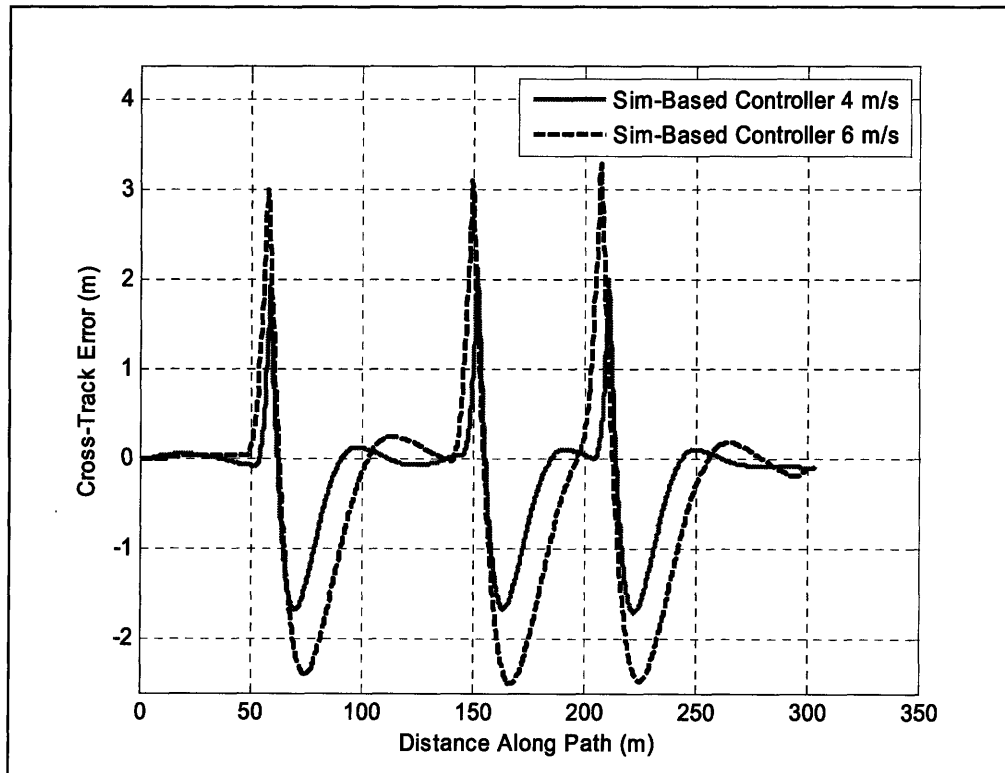


Figure 6.19: Cross-Track Error for SBC Tracking Rectangular Path

The advantage of using the simulation-based controller, however, is the ability to handle situations in which the controller needs to rapidly unwind the steering wheel from its current position in order to avoid overshooting the target. The traditional pure pursuit controller, because it lacks any awareness of where the steering wheels are at a given moment, fails to handle these situations. To test this capability, Talos-II was first commanded to turn its wheels full to the left (from the driver's perspective). Talos-II was then commanded to accelerate up to 8 m/s while tracking a path at an angle of 26° from its initial heading. The results of this experiment for both the simulation-based and pure pursuit controllers are presented in Figure 6.20. The associated cross-track errors and desired steering angle are then presented in Figure 6.21 and Figure 6.22 respectively. It is clear that the simulation-based controller starts unwinding the steering wheel much more rapidly, resulting in far less overshoot of the desired path. This represents a marked improvement over the pure pursuit controller and could successfully avoid situations in which instability would result.

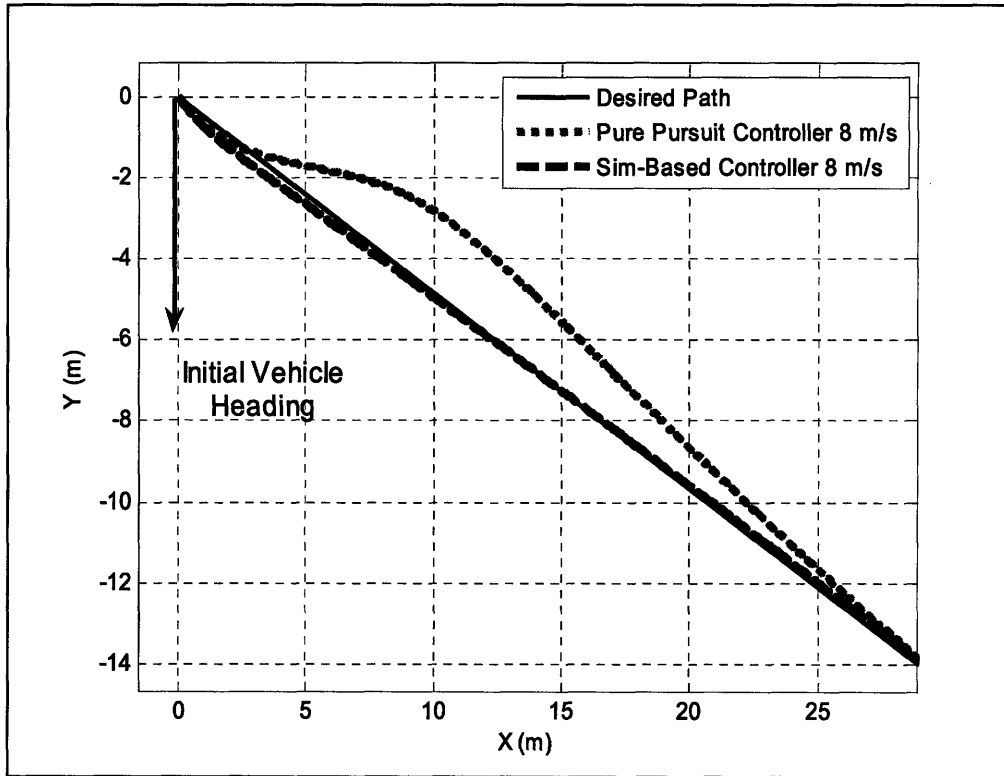


Figure 6.20: SBC vs. Pure Pursuit Controller

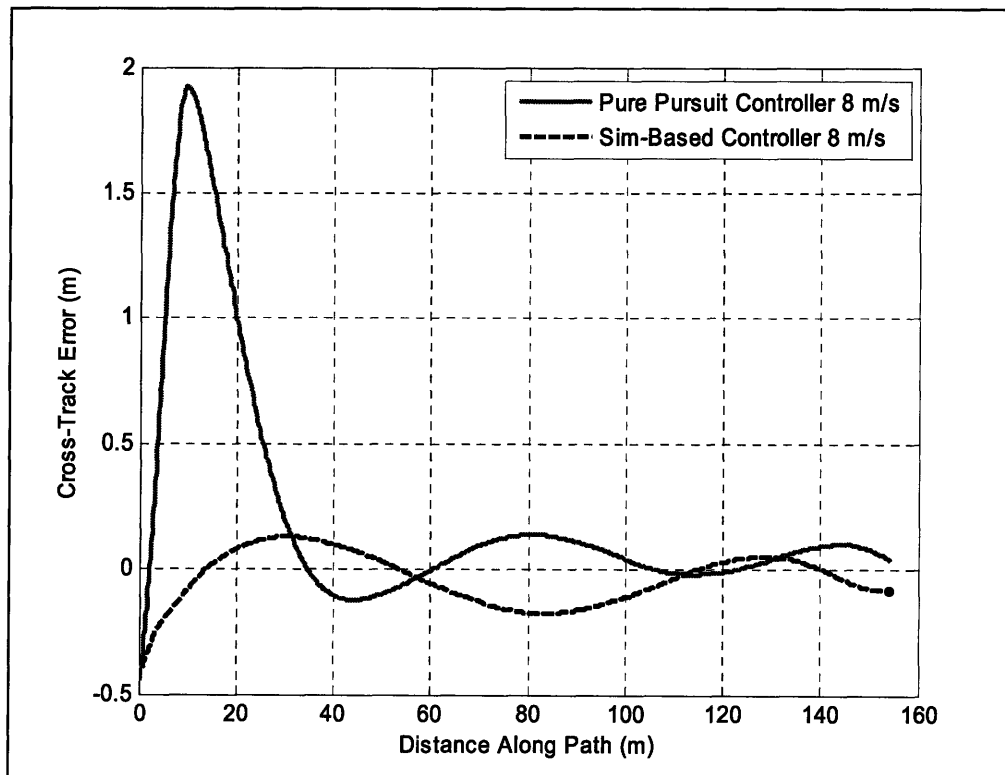


Figure 6.21: Cross-Track Error for Comparing SBC and Pure Pursuit Controller

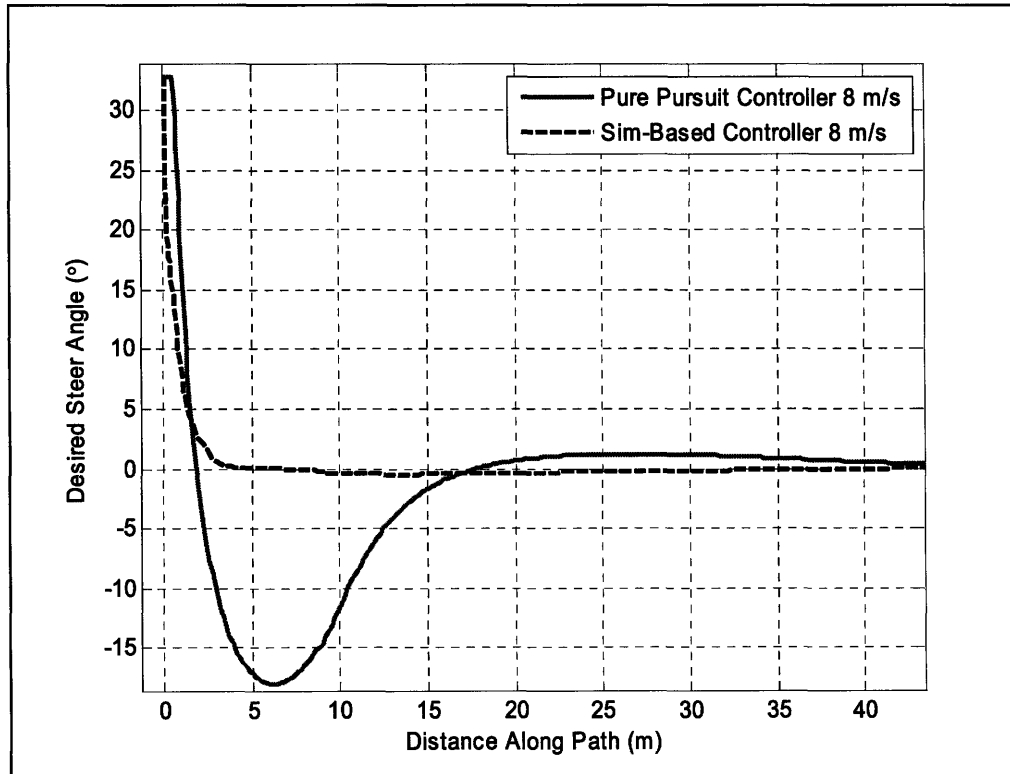


Figure 6.22: Desired Steer Angle for SBC vs. Pure Pursuit Controller

6.3 Pure Pursuit for Driving in Reverse

The pure pursuit controller for driving in reverse was evaluated on the figure eight test course with Talos-II traveling at a speed of 3 *m/s*. The results of this evaluation are presented in Figure 6.22 with the associated cross-track error and desired steer angle presented in Figure 6.23 and Figure 6.24 respectively. The pure pursuit controller exhibited excellent path tracking with less than 16 *cm* of peak path deviation and an RMS error of .0342 *m*. These results substantially validate the assumptions mad in Section 5.1.

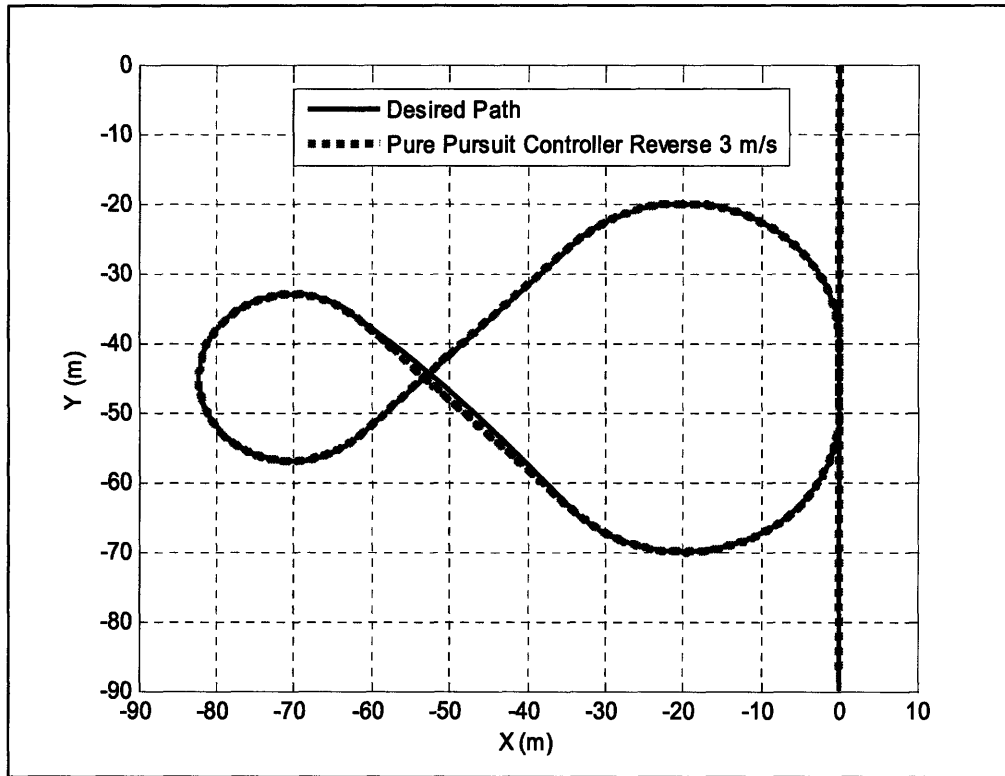


Figure 6.23: Pure Pursuit Controller for Reverse at 3 m/s

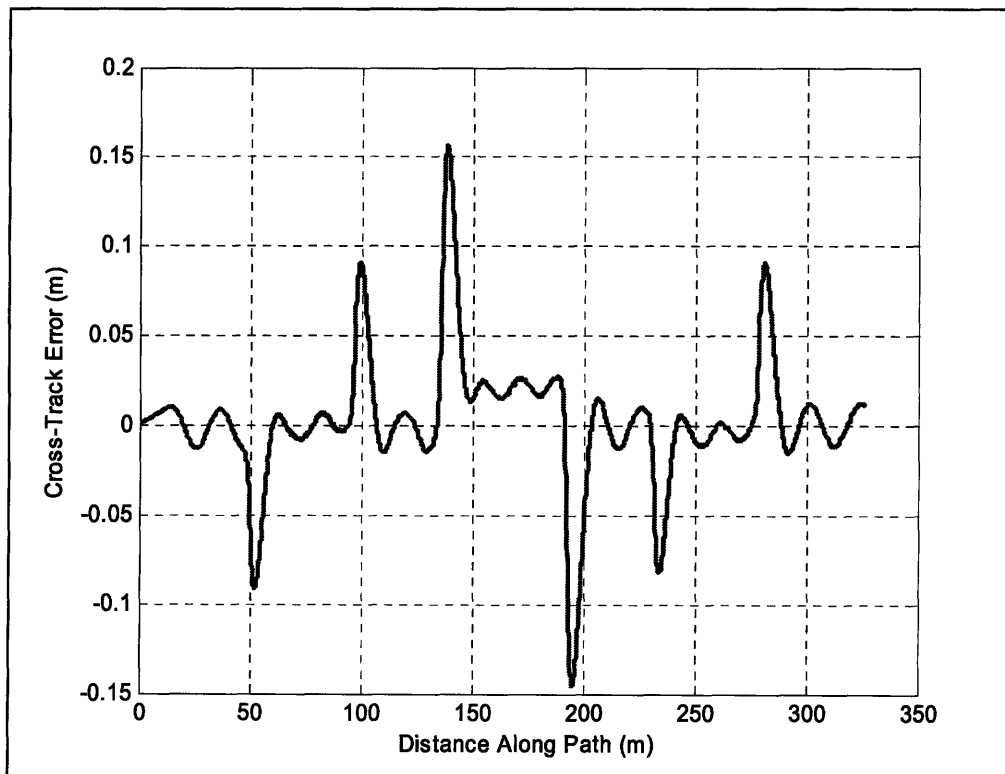


Figure 6.24: Cross-Track Error for Pure Pursuit Controller in Reverse

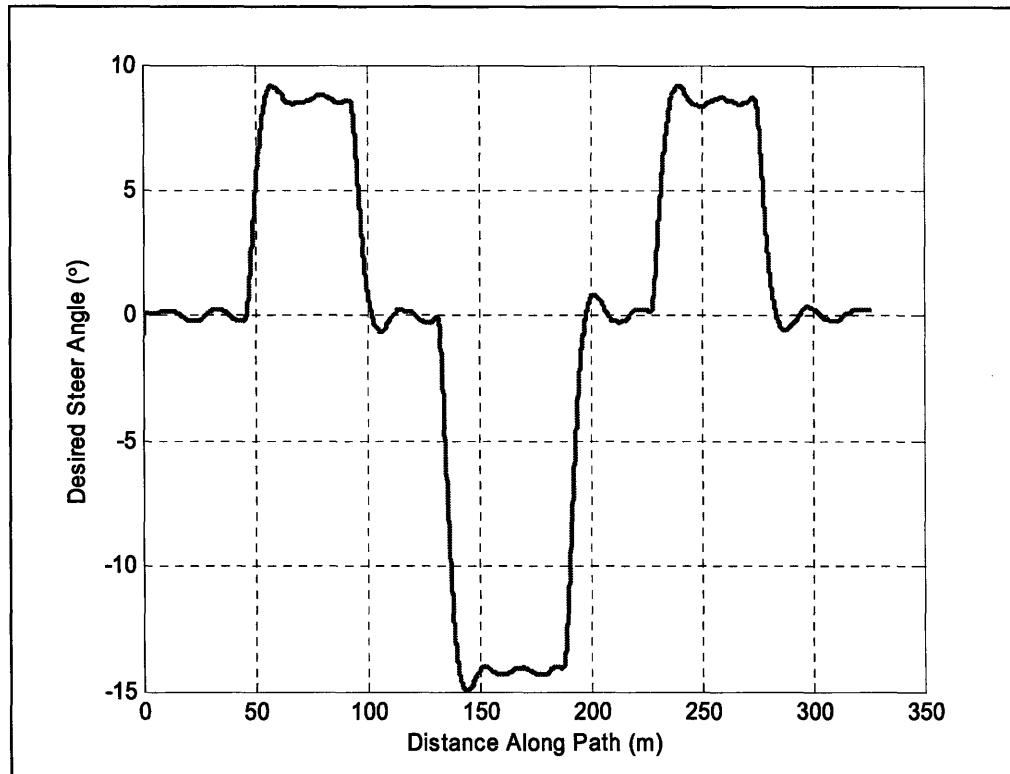


Figure 6.25: Desired Steer Angle for Pure Pursuit Controller in Reverse

6.4 Mechanism-based Control for Driving in Reverse

Results for the mechanism-based controller are presented in Figure 6.26, Figure 6.27, and Figure 6.28. Here the peak cross-track error is slightly greater than that for the pure pursuit controller in reverse (45 *cm* vs. 15 *cm*). Additionally, the RMS error for this run was .207 *m*. These results should not suggest that any strong conclusion be made as the mechanism-based controller exhibited slightly better performance on Talos-I than did the pure pursuit controller. Additionally, results were taken without any adjustments to the parameters designed in section 5.4. Slight modifications to these tuning parameters may increase performance. Ultimately, adjustments were not made as the results presented here are more than sufficient for Team MIT's entry into the DUC.

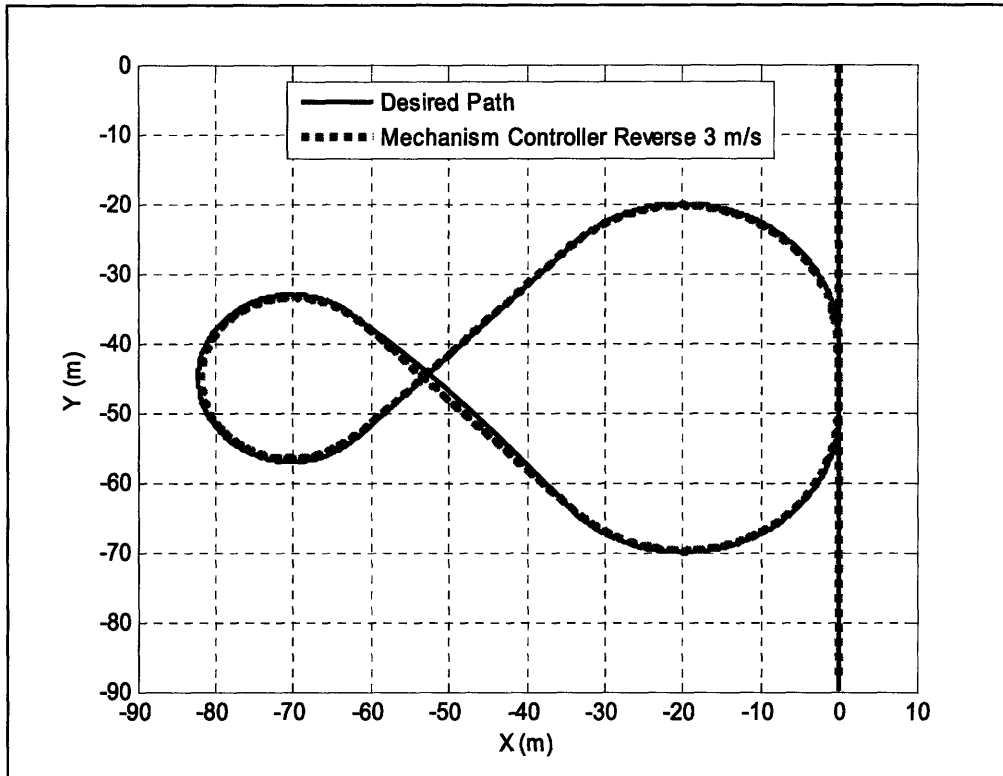


Figure 6.26: Mechanism-based Controller for Reverse at 3 m/s

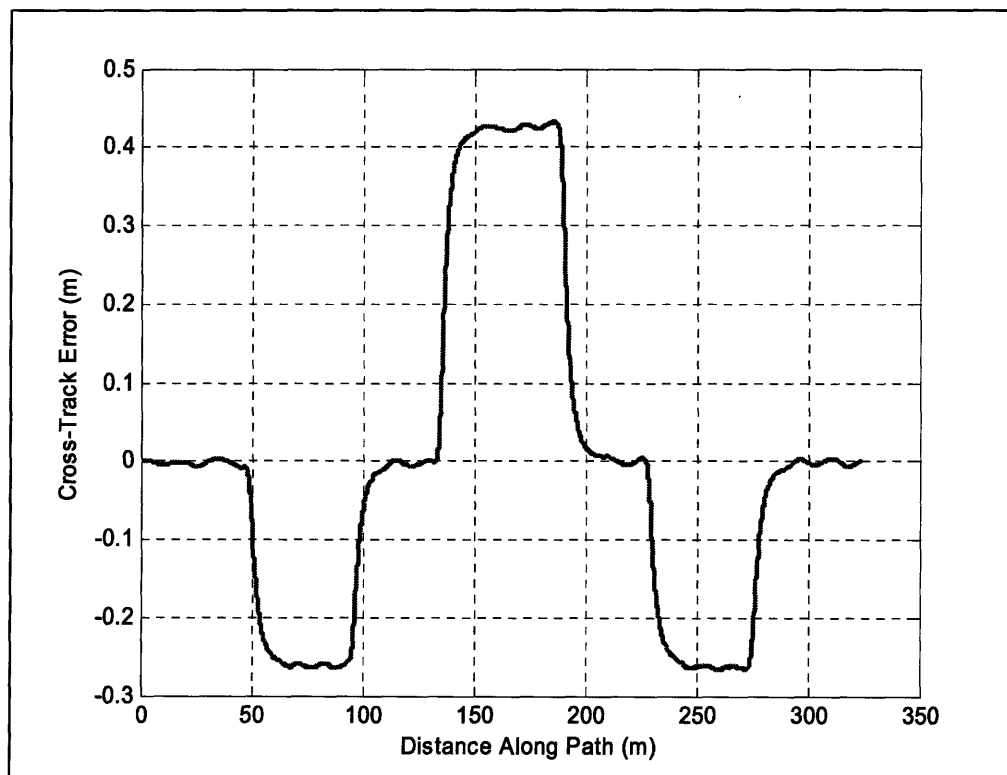


Figure 6.27: Cross-Track Error for Mechanism-based Controller

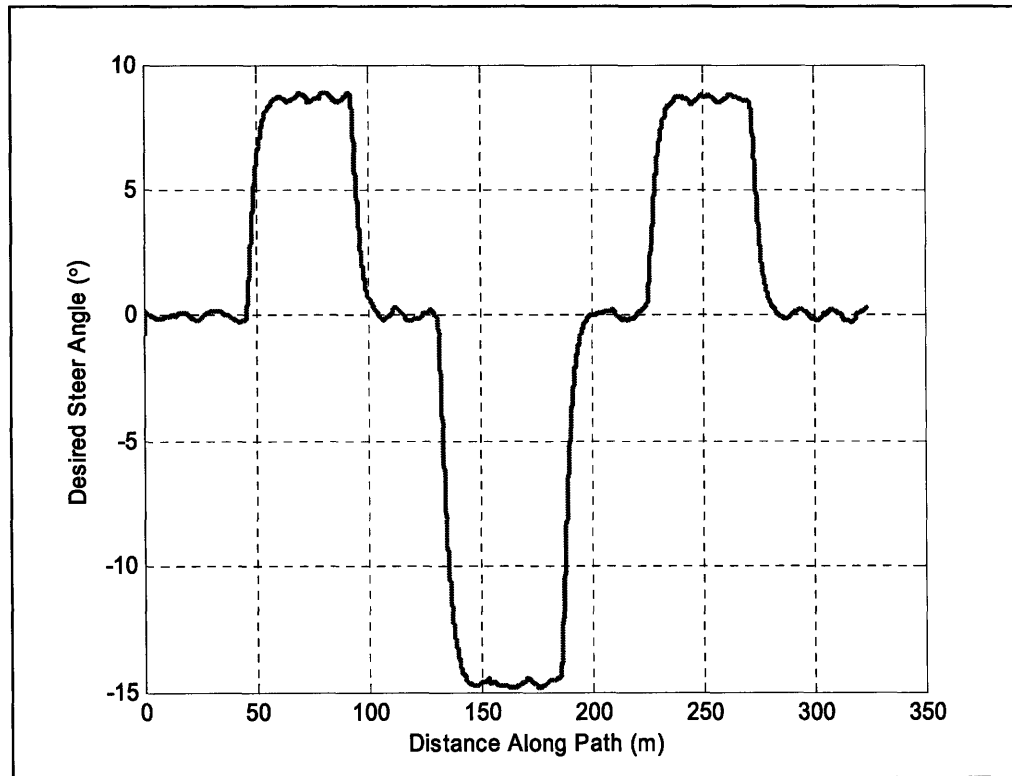


Figure 6.28: Desired Steer Angle for Mechanism-based Controller

6.5 Simulation-based Control for Driving in Reverse

Results for the simulation-based controller traversing the figure eight evaluation course at a speed of 3 *m/s* are presented in Figure 6.29, Figure 6.30, and Figure 6.31. Here the overall performance produced a peak deviation of 25 *cm* with an RMS error of .082 *m*. As noted in Section 6.3, the simulation-based controller once again exhibited noisy oscillations in the desired steer angle. Reducing the bandwidth of the low-pass filter used on the differentiation of the wheel encoder position would help reduce this phenomenon.

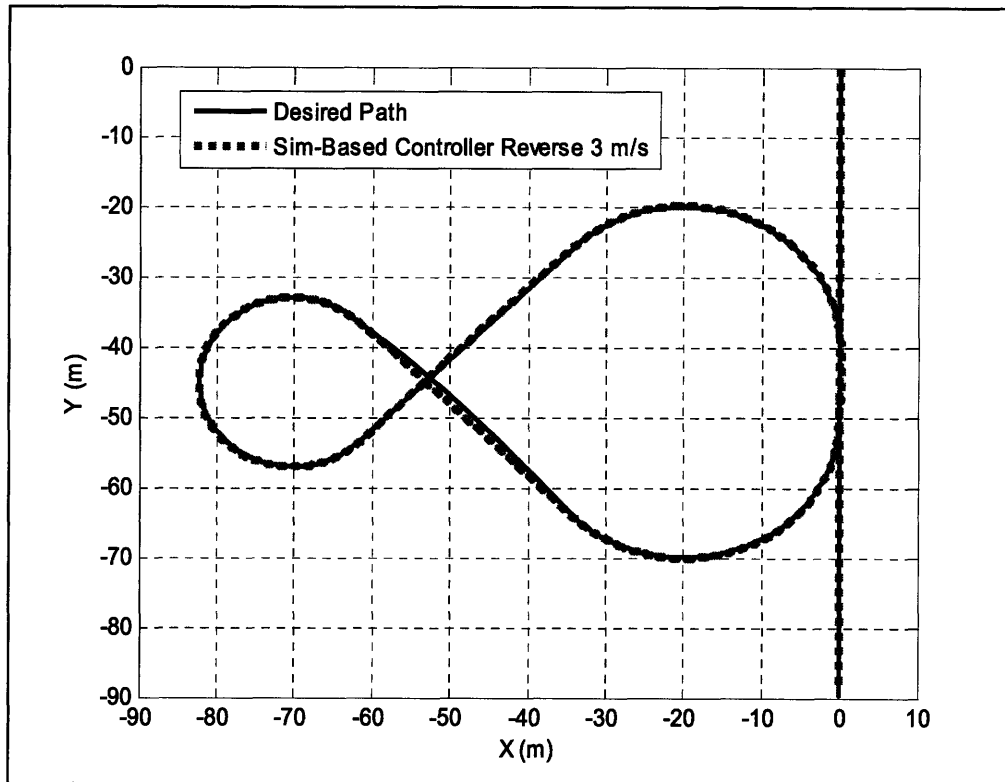


Figure 6.29: SBC for Reverse at 3 m/s

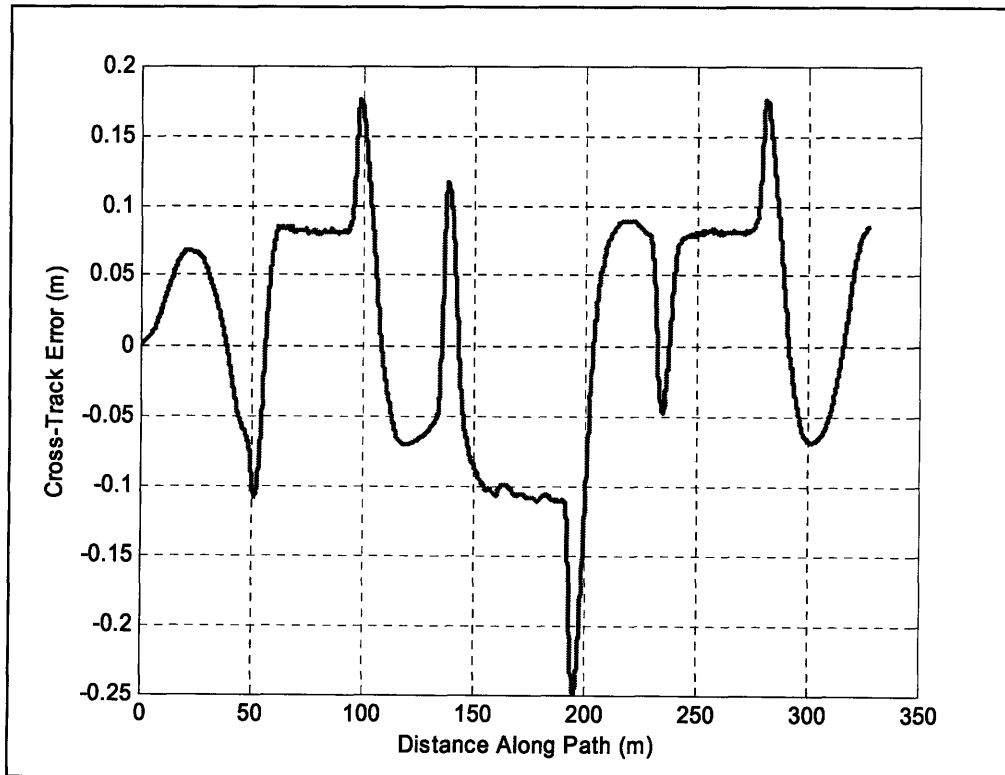


Figure 6.30: Cross-Track Error for SBC in Reverse

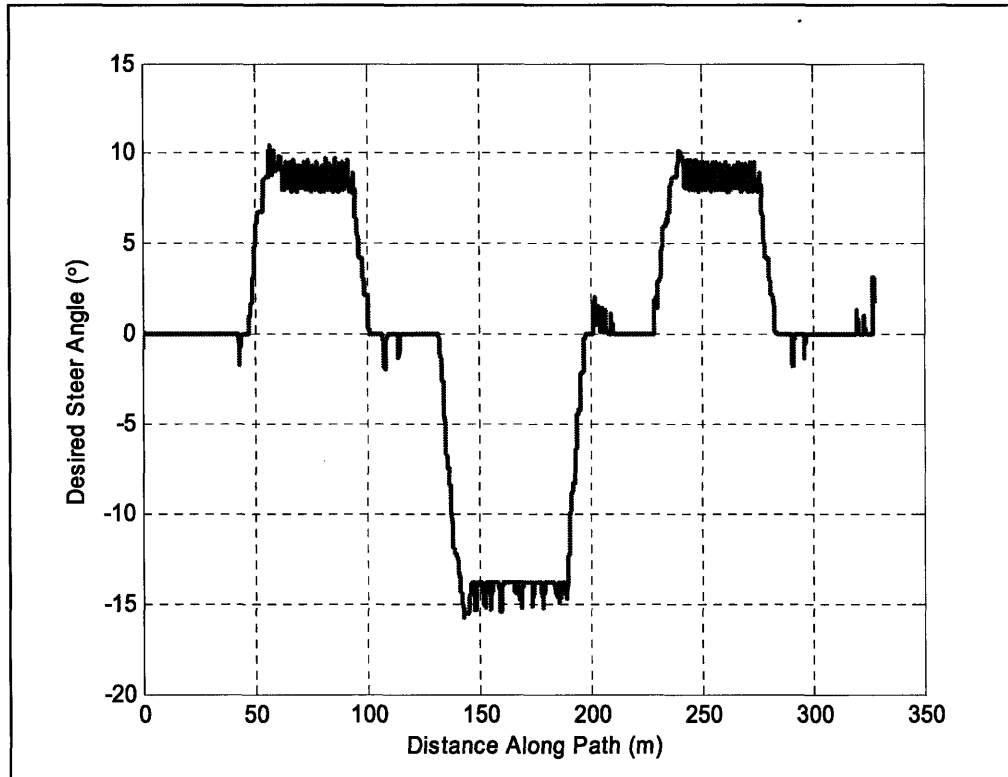


Figure 6.31: Desired Steer Angle for SBC in Reverse

As with forward driving, the simulation-based controller is especially well suited to handling situations in which it is necessary to counter steer vary rapidly in order to maintain stability. Like the test described in Section 6.2, Talos-II was initially aligned with its wheels positioned fully to the right. Talos-II was then commanded to accelerate up to 4 *m/s* while tracking a straight line oriented at 206° to the vehicle's initial forward heading. The results for conducting this experiment on the simulation-based controller and the mechanism-based controller are presented in Figure 6.32, Figure 6.33, and Figure 6.34. These figures clearly show that the simulation-based controller initially overshoot but was able to rapidly correct and reacquire the desired path. The mechanism-based controller exhibited the same overshoot, but was unable to reacquire the path and went entirely unstable. Admittedly, if some speed control logic was actively slowing Talos-II from 4 *m/s*, the mechanism-based controller may have regained stability.

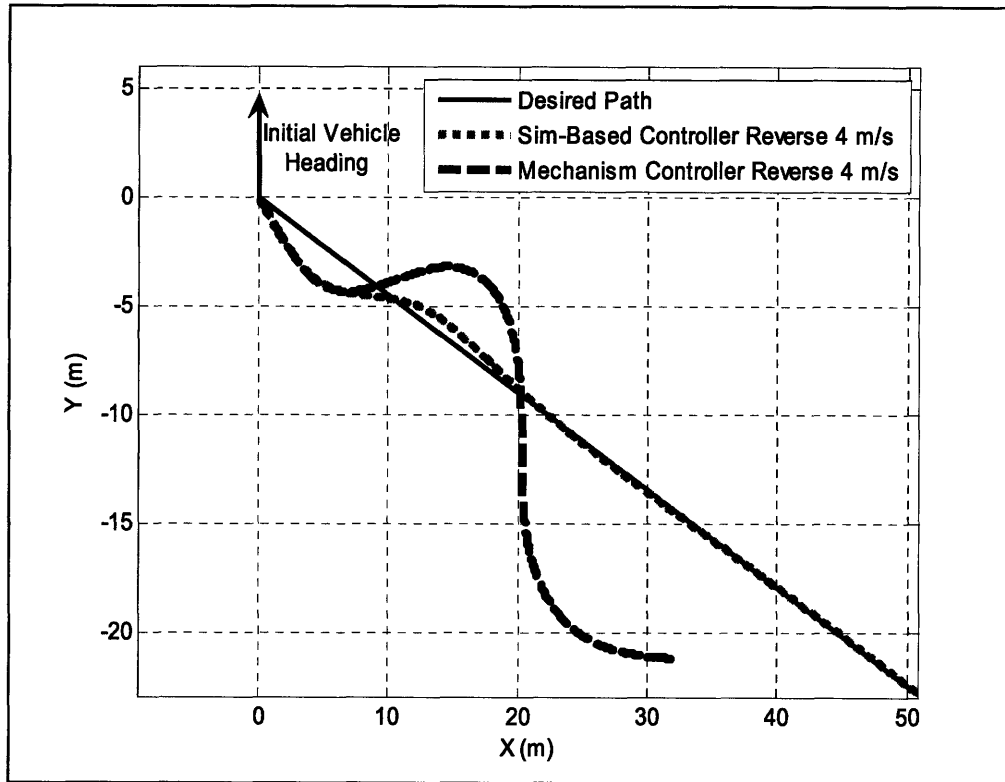


Figure 6.32: MPC vs. Pure Pursuit Style Controller

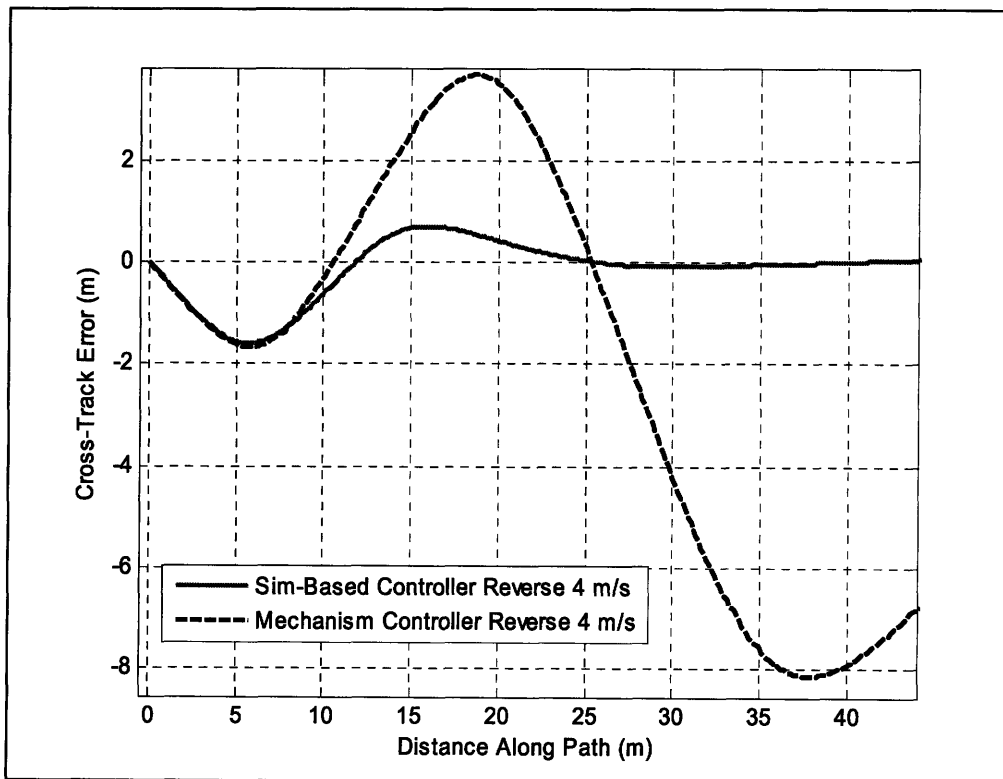


Figure 6.33: Cross-Track Error for MPC vs. Pure Pursuit Style Controller

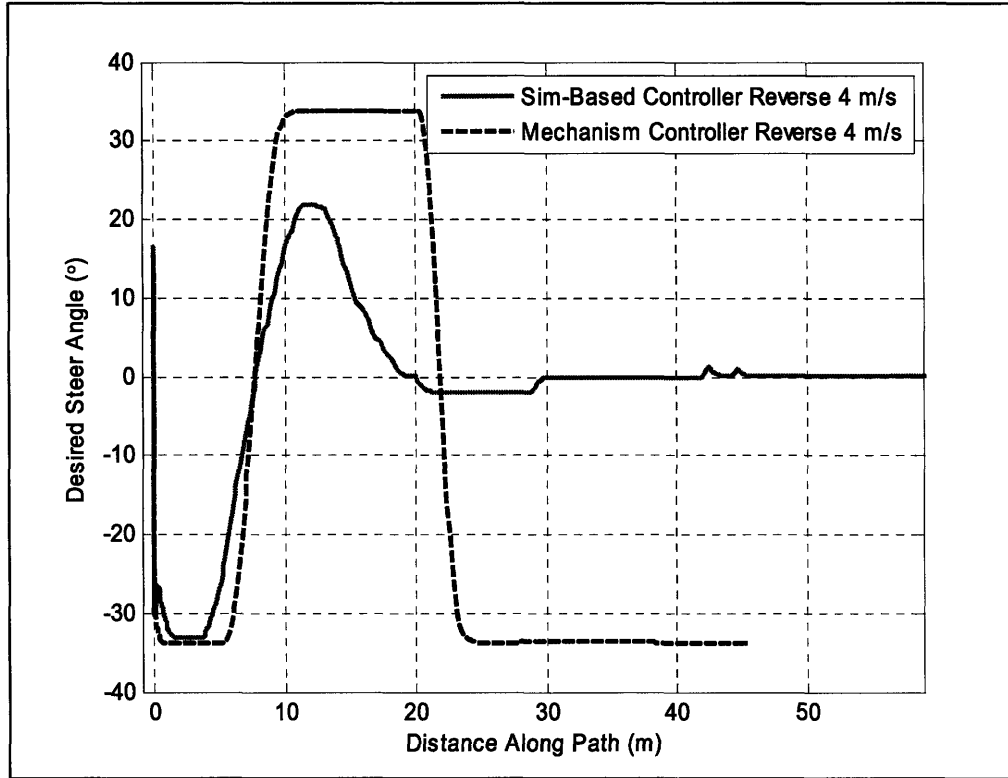


Figure 6.34: Desired Steer Angle for MPC vs. Pure Pursuit Style Controller

6.6 Conclusions

All of the controllers presented here exhibited reasonable to excellent tracking performance. In general, pure pursuit style controllers will exhibit excellent tracking performance as long as the vehicle's commanded speed is chosen appropriately. Indeed, the tuning procedures outlined in this thesis were generally rather conservative; at higher speeds this resulted in smooth driving with a tendency to tolerate some nominal cross-track error, typically less than $.5 m$. If a smaller K_{La} was selected, an improvement in tracking performance would result at the expense of some stability margin. Another key result of this work is that the simulation-based controllers are able to successfully handle many of the failure modes associated with typical pure pursuit. This was successfully demonstrated for both forward and reverse driving.

References

1. Ackerman, J., J. Guldner, W. Sienel, R. Steinhauser, and V.I. Utkin, "Linear and Nonlinear Controller Design for Robust Automatic Steering," *IEEE Transactions on Control Systems Technology*, vol. 3, no. 1, March 1995.
2. Ackerman, J., Robust Control: Systems with Uncertain Physical Parameters, London: Springer-Verlag, 1993.
3. Amidi, O., "Integrated Mobile Robot Control", *Robotics Institute, Carnegie Mellon University*, May 1990, CMU-RI-TR-90-17, Pittsburgh, PA.
4. Åström, Karl J., Tore Hägglund, PID Controllers, 2nd ed., Research Triangle Park, N.C.: International Society for Measurement and Control, c1995.
5. Bonnifait, P., P. Bouron, P. Crubillé, and D. Meizel, "Data Fusion of Four ABS Sensors and GPS for an Enhanced Localization of Car-like Vehicles," Proceedings of the 2001 *IEEE International Conference on Robotics & Automation*, May 2001, Seoul, Korea.
6. Bauer, Horst, ed., Automotive Handbook, 5th rev., Stuttgart: Robert Bosch GmbH; Warrendale, PA: Distributed by Society of Automotive Engineers, c2000.
7. Chen, C., H. Tan, "Experimental Study of Dynamic Look-Ahead Scheme for Vehicle Steering Control", Proceedings of the *American Control Conference*, June 1999.
8. Coulter, R.C., "Implementation of the Pure Pursuit Path Tracking Algorithm", *Robotics Institute, Carnegie Mellon University*, January 1992, CMU-RI-TR-92-01, Pittsburgh, PA.
9. Feng, K., H. Tan, and M. Tomizuka, "Automatic Steering Control of Vehicle Lateral Motion with the Effect of Roll Dynamics," Proceedings of the *American Control Conference*, Philadelphia, Pennsylvania, June 1998.
10. Gillespie, T.D., Fundamentals of Vehicle Dynamics, Warrendale, PA: Society of Automotive Engineers, 1992.
11. Guldner, J., V.I. Utkin, and J. Ackerman, "A Sliding Mode Control Approach to Automatic Car Steering," In Proceedings of the *American Control Conference*, Baltimore, June 1994.
12. Guldner, J., W. Sienel, H. Tan, J. Ackerman, S. Patwardhan, and T. Bünte, "Robust Automatic Steering Control for Look-Down Reference Systems with Front and Rear Sensors," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 1, January 1999.
13. Hernandez, J.I., and C.Y. Kuo, "Lateral Control of Higher Order Nonlinear Vehicle Model in Emergency Maneuvers Using Absolute Positioning GPS and Magnetic Markers," *IEEE Transactions on Vehicular Technology*, Vol. 53, No. 2, March 2004.
14. Hingwe, P., and M. Tomizuka, "A Variable Look-Ahead Controller for Lateral Guidance of Four Wheeled Vehicles," Proceedings of the *American Control Conference*, Philadelphia, Pennsylvania, June 1998.
15. Kelly, A., and A. Stentz, "An Approach to Rough Terrain Autonomous Mobility," *International Conference on Mobile Planetary Robots*, Santa Monica, January 1997.
16. Laumond, J.P., ed. Robot Motion Planning and Control. Great Britain: Springer-Verlag, 1998.
17. Meriam, J.L., and L.G. Kraige, Engineering Mechanics: Dynamics, 5th ed., New Jersey: John Wiley & Sons, Inc., 2001.

18. Ogata, K., Modern Control Engineering, 3rd ed., New Jersey: Prentice-Hall, Inc., 1997.
19. Omae, M., and T. Fujioka, "DGPS-Based Position Measurement and Steering Control for Automatic Driving," Proceedings of the *American Control Conference*, San Diego, California, June 1999.
20. Park, S., "Avionics and control system development for mid-air rendezvous of two unmanned aerial vehicles," Ph.D. dissertation, MIT, Feb. 2004.
21. Patwardhan, S., H. Tan, J. Guldner, and M. Tomizuka, "A General Framework for Automatic Steering Control: System Analysis," Proceedings of the *American Controls Conference*, June 1997, Albuquerque, NM.
22. Patwardhan, S., H. Tan, J. Guldner, and M. Tomizuka, "Lane Following During Backward Driving for Front Wheel Steered Vehicles," Proceedings of the *American Control Conference*, June 1997, Albuquerque, NM.
23. Rajamani, R., C. Zhu, and L. Alexander, "Lateral Control of a Backward Driven Front-Steering Vehicle," *Control Engineering Practice*, pp. 531-540, July 2002.
24. Sasiadek, J.Z., and Q. Wang, "Sensor Fusion Based on Fuzzy Kalman Filtering for Autonomous Robot Vehicle," Proceedings of the *IEEE International Conference on Robotics & Automation*, May 1999, Detroit, Michigan.
25. Slotine, J.J.E., L. Weiping, Applied Nonlinear Control, New Jersey: Prentice-Hall, Inc., 1991.
26. Shladover, S.E., C.A. Desoer, J.K. Hedrick, M. Tomizuka, J. Walrand, W. Zhang, D.H. McMahon, H. Peng, S. Sheikholeslam, and N. Mckeown, "Automatic Vehicle Control Developments in the PATH Program," *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 1, February 1991.
27. Smith, D.E., and J.M. Starkey, "Effect of Model Complexity on the Performance of Automated Vehicle Steering Controllers: Model Development, Validation and Comparison," *Vehicle System Dynamics*, Vol. 24, pp. 163-181, 1995.
28. Spenko, M.J., "Hazard Avoidance for High-Speed-Rough-Terrain Unmanned Ground Vehicles," Ph.D. dissertation, MIT, June 2005.
29. Thrun, S., M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Dievel, P. Fong, J. Gale, M. Halpenny, G. Hoffman, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohban, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The Robot that Won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, pp. 661-692, August 2006.
30. Travis, W., R. Daily, D.M. Bevly, K. Knodler, R. Behringer, H. Hemetsberger, J.Kogler, W. Kubinger, B. Alefs, "SciAutonics-Auburn Engineering's Low-Cost High-Speed ATV for the 2005 DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, pp. 580-597, August 2006.
31. Trepagnier, P.G., J. Nagel, P.M. Kinney, C. Koutsougeras, and M.Dooner, "KAT-5: Robust Systems for Autonomous Vehicle Navigation in Challenging and Unknown Terrain", *Journal of Field Robotics*, vol. 23, pp. 509-526, August 2006.
32. Wit, J., C.D. Crane III, and D. Armstrong, "Autonomous Ground Vehicle Path Tracking," *Journal of Robotic Systems*, vol. 21, pp. 439-449, October 2004.
33. Wong, J.Y., Theory of Ground Vehicles, New York: John Wiley, 2001.

Appendix A

This appendix introduces some simple models that may be used to prevent such catastrophic conditions as lateral sliding and rollover. Additionally, some discussion will be given to how this modeling may be used to help select a safe driving speed. This section borrows heavily from the work presented in [28].

A.1 Vehicle Rollover

Vehicle rollover can be a complicated condition to predict and much research is given to the topic. However, using a model of the vehicle in which the suspension system is effectively taken as rigid, a reasonable predictor of rollover can be developed with fundamental principles of mechanics. Figure A.1 presents a free-body diagram of a vehicle in which F and N denote the tire friction forces and normal forces respectively. Subscripts R and L then indicate whether the force acts on the left or right side of the vehicle. For this model, the distinction between front and rear wheels is ignored. Moreover, the yaw (Ψ), pitch (θ), and roll (ϕ) of the vehicle are used to resolve the force due to gravity into x_b , y_b , and z_b components relative to the vehicle's body fixed frame.

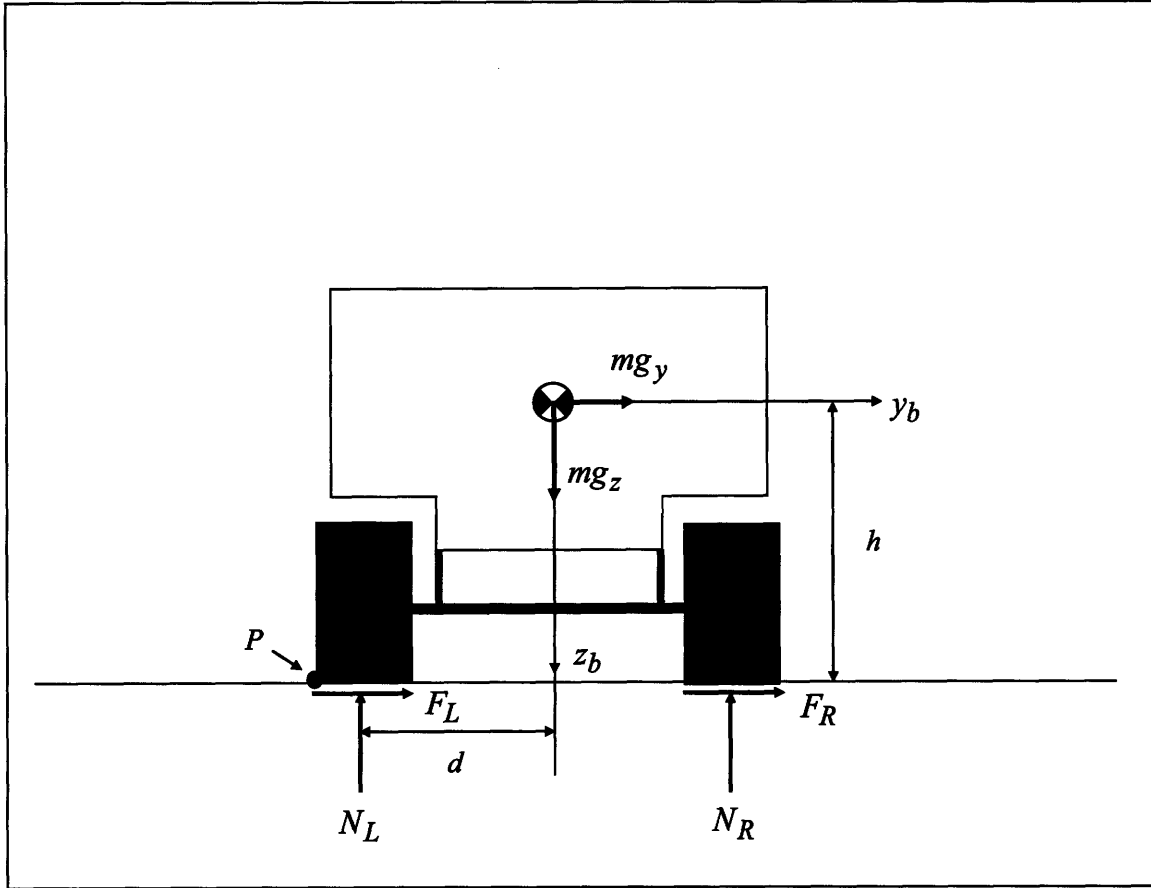


Figure A.1: Vehicle Free-Body Diagram (from Rear)

For a vehicle taking a turn, the tipping point is the contact location between either the edge of the left or the edge of the right tire and the road. For purposes of illustration, the left tire is chosen as the tipping point P . The sum of the moments about P is then given as the following:

$$\sum M_p = -mg_z d - mg_y h + 2N_R d \quad (\text{A.1})$$

The angular momentum about P is differentiated in order to determine the sum of the moments in terms of path curvature (k). The following results:

$$\sum M_p = mv^2 kh \quad (\text{A.2})$$

At the onset of rollover, N_R is identically zero. Combining equation A.1 and A.2 yields the simple rollover condition about the left tire as the following:

$$k = \frac{-g_z d - g_y h}{v^2 h} \quad (\text{A.3})$$

Repeating the same procedure for rolling over the right tire yields the following:

$$k = \frac{g_z d - g_y h}{v^2 h} \quad (\text{A.4})$$

Combining these results yields the maximum and minimum curvature constraints for a vehicle traveling at a speed v :

$$k^{\min, \max} = \frac{-g_y h \mp g_z d}{v^2 h} \quad (\text{A.5})$$

This can now be used to ensure that the steering command issued by the controller will produce a safe vehicle response. More specifically, equation 3.24 can be rewritten as the following:

$$\delta = \left(L_b + \frac{(l_r - l_f) m v_x^2}{C L_b} \right) \kappa \quad (\text{A.6})$$

This yields the following steering constraints:

$$\delta^{\min, \max} = \left(L_b + \frac{(l_r - l_f) m v^2}{C L_b} \right) \left(\frac{-g_y h \mp g_z d}{v^2 h} \right) \quad (\text{A.7})$$

It should be noted that there are alternative formulations for A.7 based on different models of the vehicle kinematics or dynamics. Chapter 3 provides a good summary of the most common vehicle models.

A.2 Lateral Sliding

Lateral sliding is another performance failure mode. This will occur when the maximum friction forces from the tires can no longer provide the centripetal force

necessary to maintain the vehicle's heading. The resultant maximum traction force is assumed to be μmg_z where μ is the coefficient of friction between the tires and the road surface and m is the mass of the vehicle. The coefficient of friction is usually taken as .8 or .9 for dry pavement [6]. Using the coordinate frame shown in figure A.1, for a positive turn (positive rate of rotation about the z_b axis) the sum of the forces in the lateral direction then yields the following:

$$\sum F_y = \mu mg_z + mg_y \quad (\text{A.8})$$

For a negative turn (negative rate of rotation about the z_b axis) the sum of the forces in the lateral direction yields the following:

$$\sum F_y = -\mu mg_z + mg_y \quad (\text{A.9})$$

Combining A.8 and A.9 with the required centripetal force ($mv^2\kappa$) yields the following constraint on allowable path curvature:

$$\kappa^{\min, \max} = \frac{g_y \mp \mu g_z}{v^2} \quad (\text{A.10})$$

As in equation A.7, these can be mapped to constraints on the steering input as in the following equation:

$$\delta^{\min, \max} = \left(L_b + \frac{(l_r - l_f)mv^2}{CL_b} \right) \left(\frac{g_y \mp \mu g_z}{v^2} \right) \quad (\text{A.11})$$

Figure A.2 illustrates the steering constraints for Talos-I traveling at a bank angle of 5°.

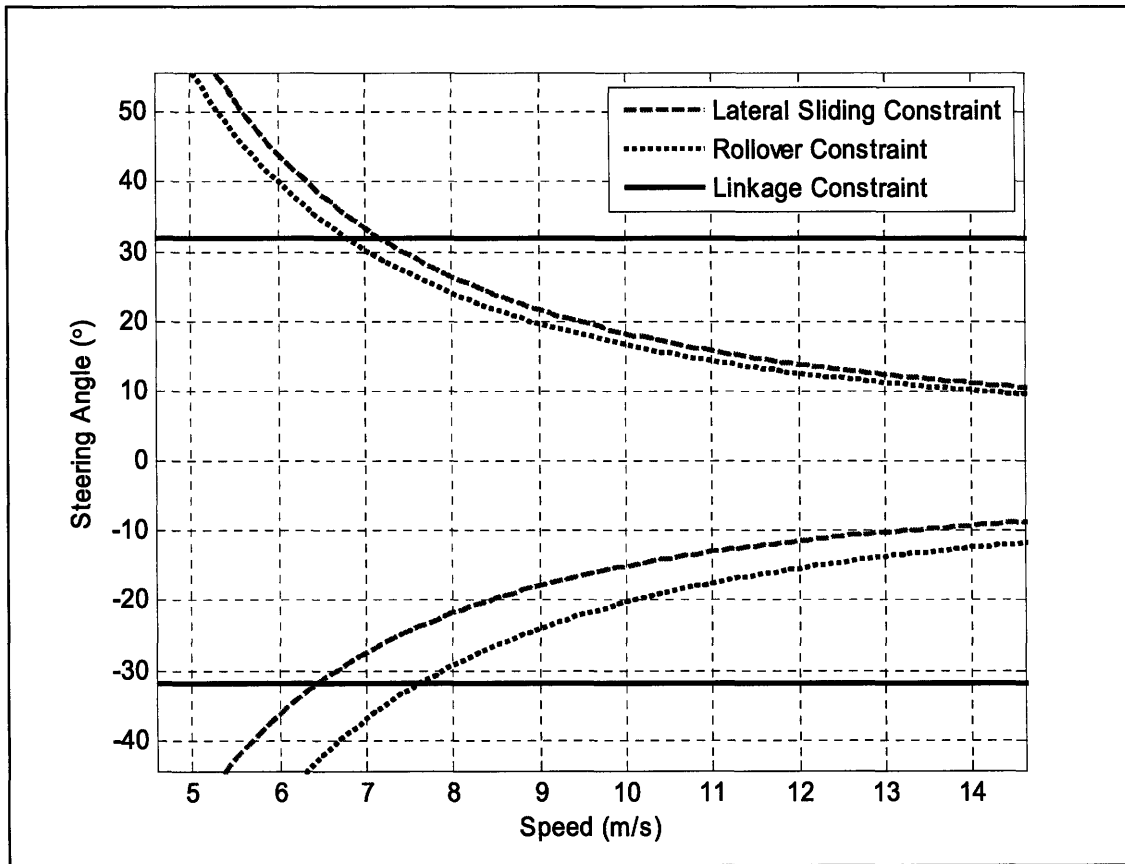


Figure A.2: Typical Steering Angle Constraints

An additional consideration that could prove beneficial to future research exploits is the use of these safety criteria for tuning the pure pursuit look-ahead distance. As shown in Chapter 6, the average cross-track error found for pure pursuit is a function of the look-ahead distance selected. By scheduling the pure pursuit look-ahead distance with vehicle speed, the designer has effectively ensured that high fidelity tracking at that speed is constrained to paths with a maximum curvature of $1/L_a$. As a substitute to the design procedure utilized in Chapter 4, the maximum and minimum allowable path curvatures could be used to select the look-ahead distance at every instant based on the vehicle's orientation. Assuming flat ground, the look-ahead distance would vary as presented

below in Figure A.3. Here it is also shown how the inclusion of a factor of safety (n) might impact the tuning of the look-ahead distance. As is clear from the figure, this approach can yield a far less conservative look-ahead distance. As such, this approach may be suitable for high performance driving, especially considering that this formulation could account for inclined terrain.

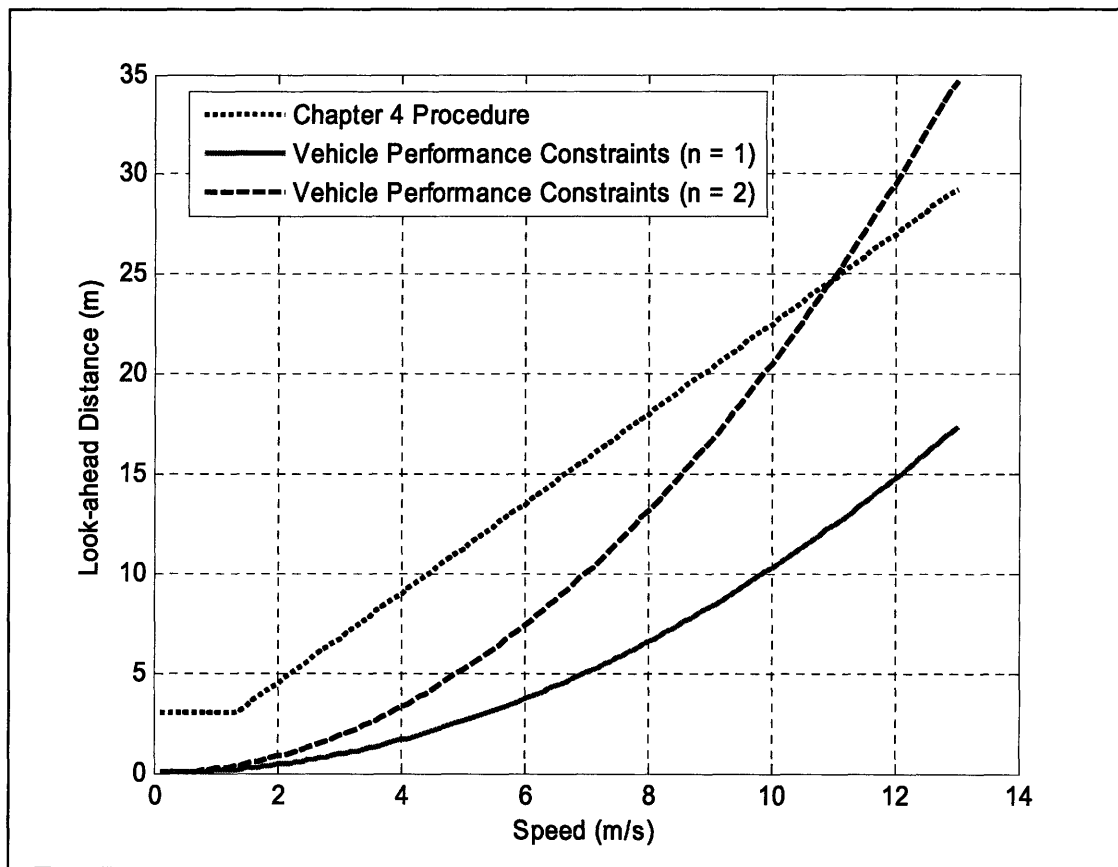


Figure A.3: Look-ahead Distance Tuning Using Vehicle Performance Constraints

A.3 Speed Regulation

As implemented for team MIT, the controller uses equations A.7 and A.11 to verify that the outputted steering command is safe for the vehicle's traveling speed. If the steering input is found unsafe, the steering command is set to the maximum or minimum (based on the direction of the turn) allowable steer angle for that speed. However, a more

sophisticated approach would be to also reduce the vehicle's desired speed such that the controller's originally requested steer angle would become safe. This logic is presented graphically in figure A.2. In this figure, v_{sc} is the speed command sent to the speed controller, κ_d is the curvature specified by the pure pursuit controller, and δ_d is the desired steering angle command (calculated from κ_d and the vehicle's current speed). The logic of this speed regulator is to first store the original pure pursuit specified steer angle in memory (stored as δ_o in figure A.2) and then proceed to check that δ_d is within the bounds set by equations A.7 and A.11. If this is not the case, δ_d is set to the maximum or minimum (once again based on the direction of the turn) allowable steering angle. The logic then compares δ_d with the originally specified steering angle to determine if the vehicle's speed command should be reduced. In the case that these values are not the same, the speed command is set to the speed that would have produced a feasible steering command had the vehicle been traveling at that speed when δ_d was calculated.

It should be understood that this is a precautionary approach to speed regulating. In the nominal case, the speed command specified by the planner is left unaltered. This approach merely attempts to coordinate speed and steering when the vehicle approaches a potential failure mode. It should also be stressed that this approach was not implemented or tested and that it is left as a future research pursuit. A particular application of this approach may be high-speed, agile driving.

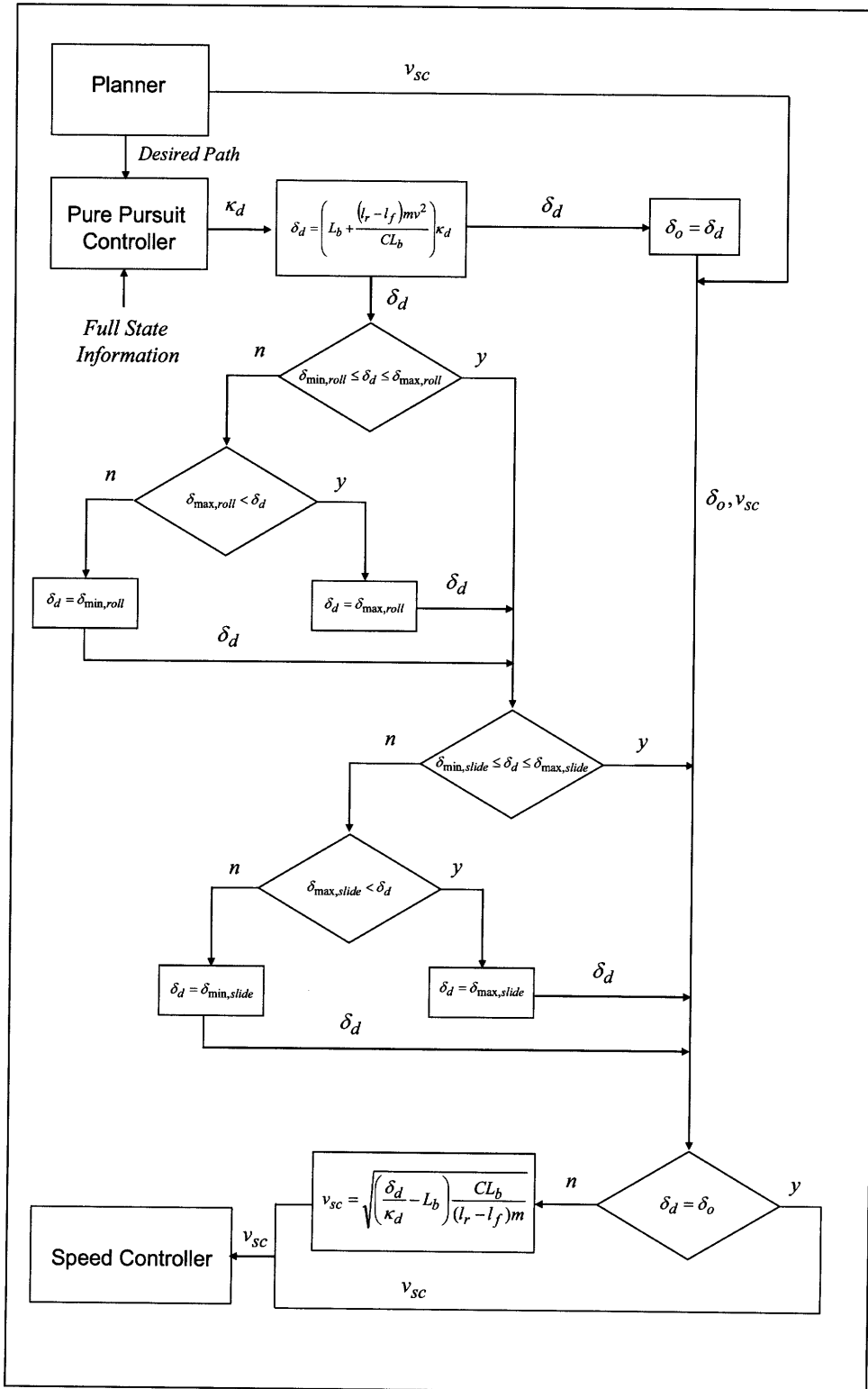


Figure A.4: Emergency Speed Regulator

Appendix B

This appendix presents an alternative controller for forward driving. Motivated by the analysis done in section 5.4, the mechanism-based controller is developed as an alternative to pure pursuit. As was discussed in section 5.4, the mechanism-based controller can be interpreted as a pure pursuit style controller with greater flexibility as a result of having an additional tuning parameter. The aim of this appendix will be to thus extend the mechanism controller to forward driving and then briefly present a methodology for tuning the controller parameters in a manner analogous to that done for reverse.

B.1 Mechanism-based Controller for Forward Driving

The mechanism-based controller for forward driving is presented below in Figure B.1.

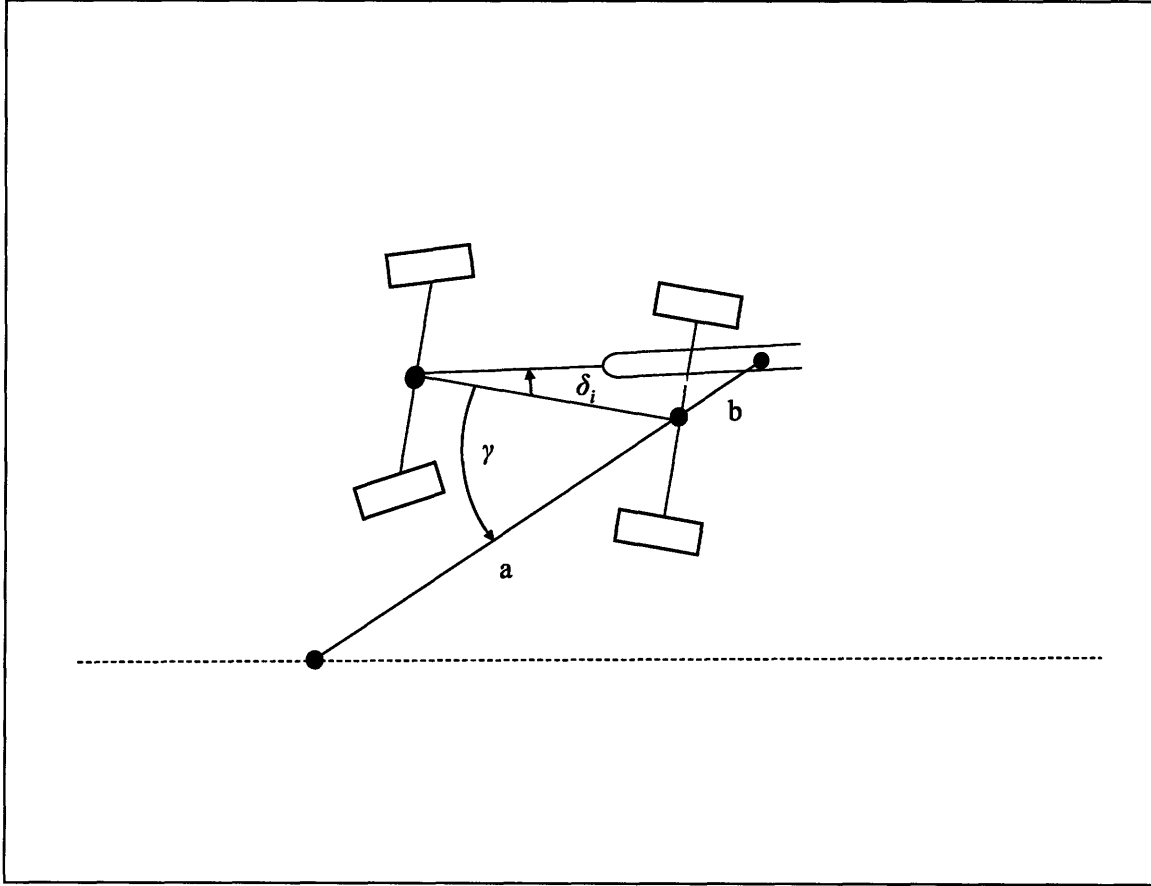


Figure B.1: Mechanism Controller for Forward Driving

As before the controller may be realized by a virtual mechanical linkage comprised of two links: a variable length link and fixed length link. The resulting control law is the following:

$$\delta_i = \tan^{-1} \left(\frac{b \sin \gamma}{L_b + b \cos \gamma} \right) \quad (\text{B.1})$$

Applying the same methodology of section 5.4, the linearized system for straight line following is given below:

$$\delta_i = \frac{b(y_d - y)}{a(L_b + b)} - \frac{b}{(L_b + b)} \psi \quad (\text{B.2})$$

An alternative linearization can be derived by assuming the yaw angle Ψ is small, which yields the following:

$$\delta_i = \frac{b(y_d - y)}{a(L_b + b)} - \frac{b\dot{y}}{V(L_b + b)} \quad (\text{B.3})$$

The full linear system is now given in equation B.4 with the transfer function from cross-track error y to desired lateral position y_d given in equation B.5.

$$\begin{bmatrix} \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & v \\ -vb & -vb \end{bmatrix} \begin{bmatrix} y \\ \psi \end{bmatrix} + \begin{bmatrix} 0 \\ vb \\ L_b a(L_b + b) \end{bmatrix} y_d \quad (\text{B.4})$$

$$\frac{y(s)}{y_d(s)} = \frac{\frac{v^2 b}{L_b a(L_b + b)}}{s^2 + \frac{vb}{L_b(L_b + b)}s + \frac{v^2 b}{L_b a(L_b + b)}} \quad (\text{B.5})$$

Here it is interesting to note that the full linear system is nearly identical to the linear system for reverse (equation 5.9) except that the term $(L_b - b)$ is replaced by the quantity $(L_b + b)$.

B.2 Parameter Tuning

Using the model from equation B.4, the natural frequency and damping of the system are given by the following:

$$\begin{aligned}\omega_n &= v \sqrt{\frac{b}{L_b a (L_b + b)}} \\ \xi &= \frac{1}{2} \sqrt{\frac{ab}{L_b (L_b + b)}}\end{aligned}\tag{B.6}$$

For a specified damping and natural frequency, ξ_d and ω_d respectively, the parameters a and b are then given by the following:

$$\begin{aligned}a &= \frac{2v\xi_d}{\omega_d} \\ b &= \frac{2\omega_d\xi_d L_b^2}{(v - 2\xi_d\omega_d L_b)}\end{aligned}\tag{C.7}$$

Here it is clear that the term b is now discontinuous and that for certain values of ξ_d and ω_d the value of b will not be positive. As a result, the design approach is the following: schedule the parameters a and b for the operating range of speeds v_{min} and v_{max} such that the parameter b is positive and continuous in this range. This is a surprisingly intuitive approach as most vehicles with an automatic transmission have a nominal cruising speed when there is zero throttle input and the DARPA challenge constraints all vehicles to a maximum speeds of 13 *m/s*. In the instances in which the vehicle is traveling at a speed below the specified minimum, the values of a and b for the speed v_{min} are selected. This is analogous to the approach selected for tuning the look-ahead distance for the pure pursuit controller in section 4.3.

Adopting the above approach and selecting a minimum speed v_{min} then yields the following condition for the acceptable value of ζ_d and ω_d :

$$\xi_d \omega_d < \frac{v_{min}}{2L_b} \quad (B.7)$$

Satisfying B.7 ensures that the parameter b will be positive and continuous for all speeds greater than v_{min} . Using equation 5.13 then yields the following condition on the system's settling time t_s :

$$t_s > \frac{8L_b}{v_{min}} \quad (B.8)$$

Selecting a v_{min} of 2.5 m/s, a ζ_d of .7, and a t_s of 11.0 sec (ω_d is defined through equation 5.13) yields the a and b velocity dependence that is illustrated in Figure B.2.

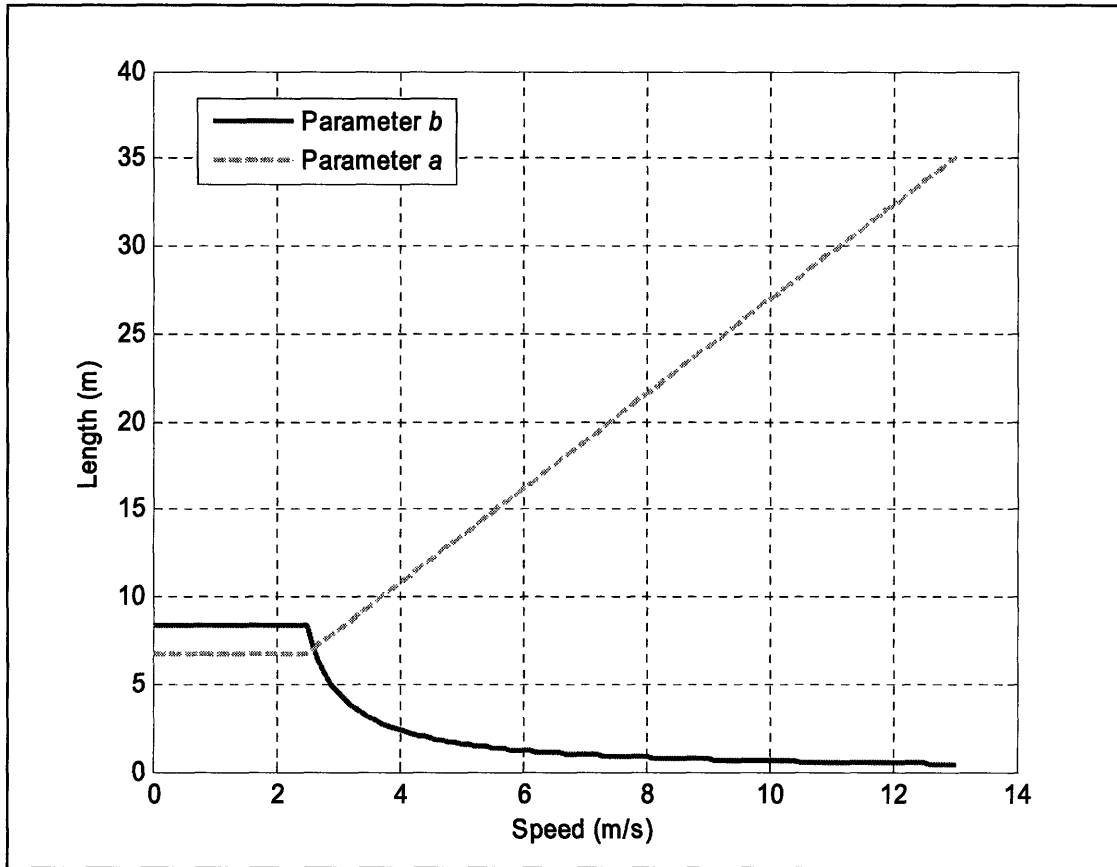


Figure B.2: Velocity Dependence of Parameters *a* and *b*

The results presented in Figure B.2 are somewhat intuitive as a controller with a larger *b* parameter will turn more sharply at a specified speed than will the same controller operating with a smaller value of *b*. Thus, at low speeds, a larger *b* is acceptable, if not desired, as sharper turns will not produce instability. However, at greater speeds it is necessary to decrease the value of the *b* parameter to ensure that system maintains a stable response. Similarly, equation B.7 clearly illustrates that the system needs an *a* parameter that increases with speed in order to maintain a constant ζ_d and ω_d .

Using the full dynamic model for Talos-I, including actuator dynamics, yields the dominant poles presented in Figure B.3. Here the poles are shown illustrated over a

range of speeds from .5 to 13 *m/s*, incremented by .5 *m/s*. At a speed of 13 *m/s* the dynamic model predicts a damping of .521 and an overshoot of 14.7 %.

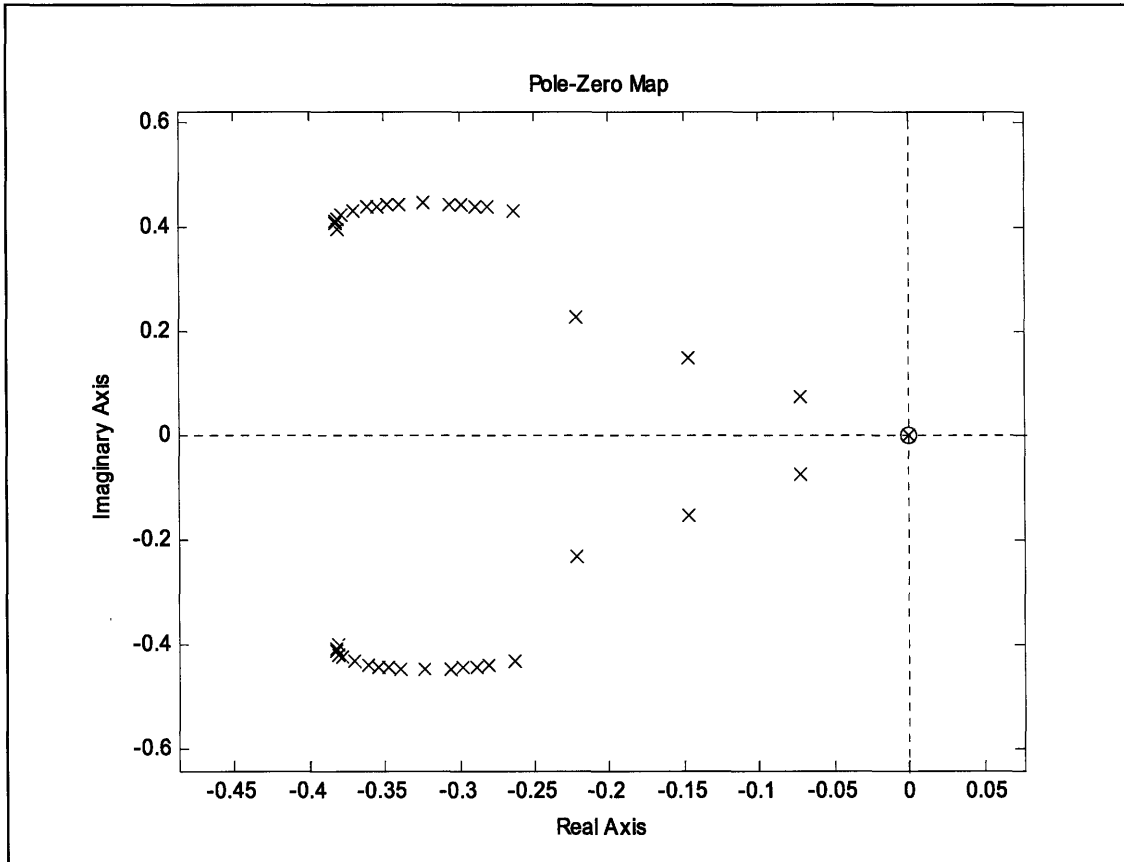


Figure B.3: Pole-Zero Map for Mechanism-based Forward Controller

Appendix C

Team MIT employed two vehicles for development: a test-bed 2006 Ford Escape and a 2006 Land Rover LR3 race vehicle. This section will provide general information on the sensor's associated with each vehicle as well as key technical specifications which were relevant to this work.

C.1 Vehicles

The 2006 Ford Escape used for initial testing was dubbed *Talos-1* and is presented below in Figure C.1.



Figure C.1: Talos-I (2006 Ford Escape) at Site Visit Course

Talos-I is equipped with an adjustable sensor suite that can be used to accommodate a large number of laser range finders (SICK LMS-291 S05 lidars) and radar systems (Delphi ACC3). Additionally, the Escape is quipped with a NAVCOM GPS unit and an XSens inertial measurement unit (IMU) that provides 6-DOF information on the vehicle's movement. Additionally an interface was established which acquired wheel encoder information from the vehicle's on board computer. Fusing the IMU information with speed estimates from the wheel encoders provided a dead-reckoned estimate of the vehicle's position.

Through a donation from Quanta Computer, Inc., the vehicle is equipped with a Fujitsu-Siemens BX600 Blade Cluster that can hold up to ten BX620 computers. Each BX620 computer contains four 2.3 GHz, 64 bit CPUs (quad-core). This provided the substantial computing power for the addition of Point Gray Firefly MV cameras to the external sensor suite. However, such a large computing center required significant power generation. This was handled by a rear mounted Honda EU3000 power generator. Alternative options were considered, such as an alternator based solution, but the generator provided cheap, reliable power that easily met system needs. Moreover, any solution relying upon the vehicle's internal electrical system would have been more expensive and would have required a significant installation time in which the vehicle would not be operational.

The official race vehicle, a Land Rover LR3, was dubbed *Talos-II*. This vehicle runs on the same computing architecture, sensor suite, and EMC actuator system as that of *Talos-I*. The vehicles differ in that *Talos-II* represents a refinement in vehicle engineering. First, the external Honda EU3000 generator was replaced by an internally mounted RV

generator that runs directly off of the vehicle's fuel supply. Second, the LR3 is equipped with an external air-conditioner which compensates for the immense heat produced by the BX600 Blade Cluster. Third, a more modular installation was achieved which allowed the inclusion of a firewall between vehicle occupants and the computer system and generator. This modular installation also preserved the rear seats, thereby allowing more engineers access to the car at a given time. Finally, the LR3 was outfitted with an Applanix POSLV 220 GPS/INS (inertial navigation system) that provides a complete state estimate by fusing wheel odometry, high-precision IMU data, and differential GPS. The precision of this system far exceeds that of Talos-I.

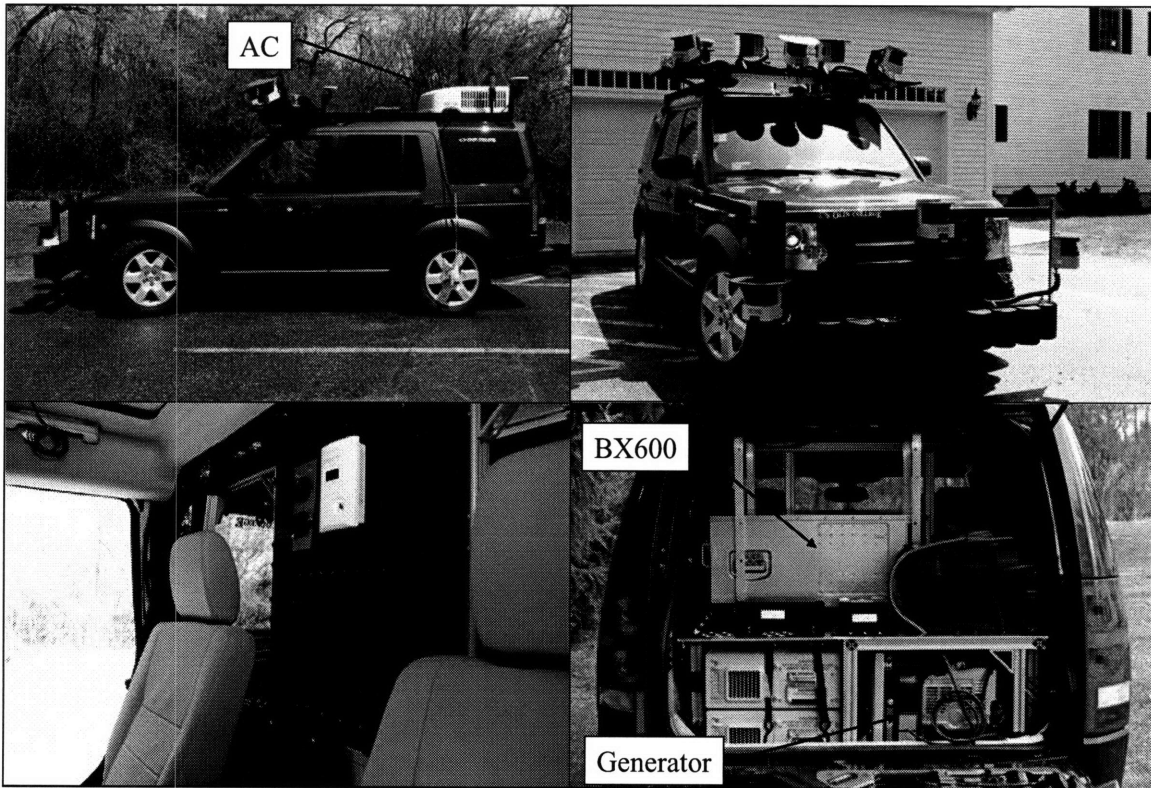


Figure C.2: Talos-II (Clock-Wise from Top-Left) Side View with Top Mounted AC Unit; Front Sensor Suite; Modular Rear Compartment Design; Rear Firewall and Access Panel.

As was briefly explained in the Introduction to this work, both vehicles are controlled by a drive-by-wire system provided by Electronic Mobility Controls (EMC).

The AEVIT (Advanced Electronic Vehicle Interface Technology) system provided by EMC consists of a servo mounted directly to the steering column and a single servo actuator for both brake and gas. This configuration provides a certain measure of safety as the brake and gas can never be actuated simultaneously. In addition to providing closed-loop motor control for the steering column and gas/brake vehicle inputs, AEVIT also provides electronic control of the drive-train; via a digital serial connection, the vehicle can be placed in Park, Drive, Reverse, and Neutral. Figure C.3 shows the AEVIT system installed on both Talos-I and Talos-II vehicle.

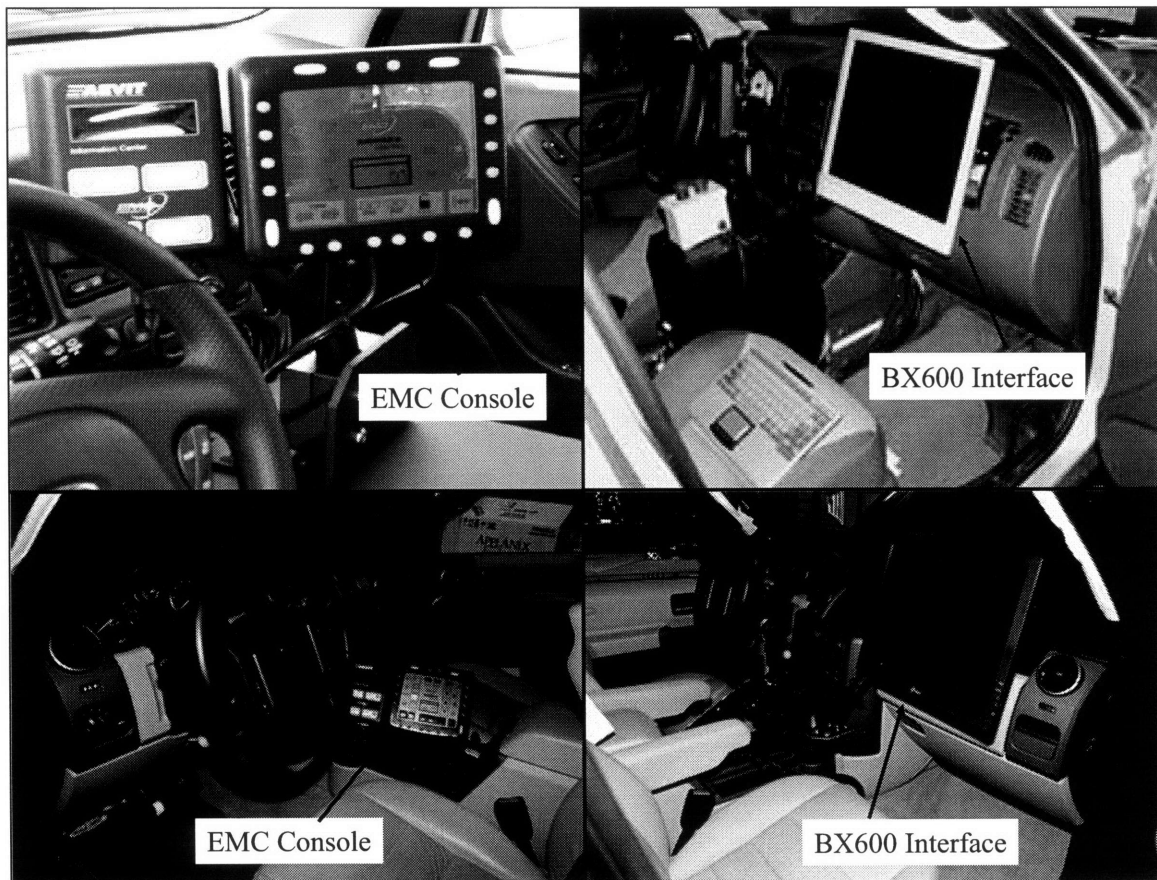


Figure C.3: (Clock-wise from Top-Left) Talos-I EMC Interface; Talos-I Front, Passenger Seat ; Talos-II Front, Passenger Seat; Talos-II EMC Interface

Relevant vehicle parameters for both *Talos-I* and *Talos-II* are presented below in Table

C.1. These parameters were taken from literature on each vehicle and were used to determine the parameters for the modeling done in Chapter 2.

| Parameter | Talos-I (2006 Ford Escape) | Talos-II (2006 Land Rover LR3) |
|---|-------------------------------|-----------------------------------|
| Wheelbase (<i>mm</i>) | 2621.28 | 2885.44 |
| Track Width - F/R (<i>mm</i>) | 15551.94/1534.16 | 1605.28/1612.9 |
| Length (<i>mm</i>) | 4427.22 | 4849.86 |
| Width (<i>mm</i>) | 1780.54 | 1915.16 |
| Height (<i>mm</i>) | 1788.16 | 1892.3 |
| Weight (<i>kg</i>) | 1720 | 2579 |
| Weight Dist. - F/R (%) | 56.4/43.6 | 48.8/51.2 |
| Turning Circle Dia. (<i>m</i>) | 11.49 | 11.460 |
| Steering Ratio (rack and pinion linkage) | 17.9 | 17.7 |
| Turns Lock-to-Lock | 3.3 | 3.33 |
| Drag Coefficient | .40 | .41 |
| Frontal Area (<i>m</i> ²) | 2.68 | 3.15 |
| Understeer Coefficient (°) | 3 | .34 |

Table C.1: Relevant Vehicle Parameters

It is noted here that *Talos-II* has a very small understeer coefficient and behaved very nearly like a neutral steer vehicle (Chapter 3). Though the parameters above were used for modeling and analysis, vehicle modifications caused changes to these parameters. Specifically, the mass and weight distribution of each vehicle was affected by the addition of a significant amount of equipment. Weighting a vehicle accurately, however, is a challenging task as weigh stations were not conveniently located near MIT and the purchase of weighting equipment was not a viable economic option. To help improve accuracy, the assumption was made that the additional weight added to the vehicles was roughly uniformly distributed over the entire vehicle. As such, the weight distribution of both vehicles was assumed to still be consistent with the original manufacturer's specifications. For the models used in Chapter 4 and 5, mass of *Talos-I* was adjusted to

2100 kg and mass of Talos-II was taken as 3000 kg. Using these assumptions, the cornering stiffness of both vehicles was approximated using equation C.1 below.

$$C = \frac{(l_r - l_f)m}{K_{us}gL_b} \quad (C.1)$$

The cornering stiffness for Talos-I was found to be 50 *kN/rad*. Talos-II was more difficult to estimate as the understeer coefficient, given the LR3's weight distribution, should have been slightly negative (oversteer). Experimentation shows that the modified vehicle is a slightly understeered vehicle. The decision was thus made to take the cornering stiffness as 100 *kN/rad* based on values used in other research experiments and the fact that the LR3 is equipped with rugged off-road tires. As illustrated in Chapter 6, these modeling parameters produced reasonable results. Given a greater length of time and greater financial resources, however, all of these parameters would have been validated for the modified vehicle.

C.2 EMC Control System for Steering

The EMC control system receives position commands in millivolts (*mV*). The manufacturer's recommended ranges are presented below in Table C.2.

| | <i>Talos-I</i> | <i>Talos-II</i> |
|--------------------------------|-----------------------------------|-----------------|
| Gas/Brake Signal | Corresponding Voltage (mV) | |
| Full Brake | 400 | 1000 |
| No Gas, No Brake | 2500 | 2500 |
| Full Gas | 3700 | 4200 |
| Steering Signal | | |
| Full Left (S_{\min}) | 600 | 200 |
| Center (S_{center}) | 2400 | 2450 |
| Full Right (S_{\max}) | 4400 | 4700 |

Table C.2: EMC Direct Current Voltage Signals

Using the manufacture's recommended values, the output of the controller (δ_d) is mapped to voltage commands via the following relationship:

$$V_{olt} = -\frac{(S_{\max} - S_{\min})}{\pi T_{tl}} (K_{steer} \delta_d) + S_{\text{center}} \quad (\text{C.1})$$

Here the variable T_{tl} is the total turns lock-to-lock of the steering wheel. Additionally, turning left (from the perspective of forward facing driver) is taken as the positive direction of rotation. This is equivalent to having a coordinate axis affixed to the steering wheel with the z axis pointing towards the driver. This is consistent with the orientation taken for the angle of the wheels on the ground (refer to Figure 3.2) throughout this work. Finally, the coefficient K_{steer} is used to map from the desired angle of the wheels on the ground to the steering wheel.

In practice, true center for the steering input can vary depending on the system setup. To elaborate, the EMC steering actuator can be engaged and disengaged from the vehicle's steering column. The procedure is as follows: first the driver manually centers the vehicle's steering wheel. Second, the EMC actuator is centered through a calibration

process specified by the manufacturer. Finally, the EMC actuator is engaged to the steering column. In this procedure it is difficult to ensure true alignment as the steering wheel can easily be centered 1° to 5° off true center. Moreover, there is the basic issue of vehicle wheel alignment degrading as a result of bumps in the road (i.e. the reason wheel alignment is a standard maintenance procedure).

An easy calibration procedure for correcting these errors is to have the vehicle track a straight line at a constant speed and record the voltage signal that the controller outputs at steady-state. Simply put, when an error is present in the calibration, the controller attempts to correct for the error but the outputted voltage results in straight driving parallel to the desired path. The net effect is that the error does not diminish and the control signal remains constant. This was the phenomenon observed in several tests. Given the setup of the vehicle, this steady-state voltage is now the voltage for true center. Equation C.1 should then be reformulated with this new voltage for center. In practice, this calibration procedure can be easily automated.

Appendix D

This appendix will present the speed controller used by Team MIT. This material is presented here as it is heavy on implementation and presents the details associated with the AEVIT (Advanced Electronic Vehicle Interface Technology) system designed and installed by Electronic Mobility Controls (EMC) of Baton Rouge, Louisiana. In general, Team MIT's philosophy was to adopt a similar approach to that taken for the steering control system. Of primary importance was that the controller chosen was simple and easy to implement with a history of robust operation. Of nearly equal importance was the specification that the controller require minimal system identification. This was particularly necessary as access to expensive automotive equipment (such as a dynamometer) was non-existent and obtaining detailed specifications from the vehicle manufacturer proved nearly impossible. Ultimately, the controller chosen was a PID controller that actuated both the gas and brake pedals via the single EMC actuator. This simple approach proved quite effective despite the large discrepancy between a vehicle's acceleration and braking dynamics. In composing this section, significant thanks must be given to Dr. Yoshi Kuwata, Dr. Jon How, and Dr. Emilio Frazzoli as they provided significant insight into efficiently implementing this system. Many of the ideas and details presented here arose from discussions with these individuals.

D.1 EMC Actuator

The EMC gas/brake actuator is a servomechanism that controls both the gas and brake pedal. Employing a system in which only one actuator controls both pedals eliminates the possibility that both the gas and brake pedal will be actuated simultaneously. To achieve this result, the servomechanism rotates a spindle that has an arm with a roller attached at one position and cable affixed at another position. The opposing end of this cable is attached to the underside of the gas pedal. To actuate the brake, the servo rotates counter-clockwise and the roller arm presses on the brake pedal. If the actuator rotates clockwise, the affixed cable is wound up and the gas pedal is pulled down. This is illustrated graphically in Figure D.1 below.

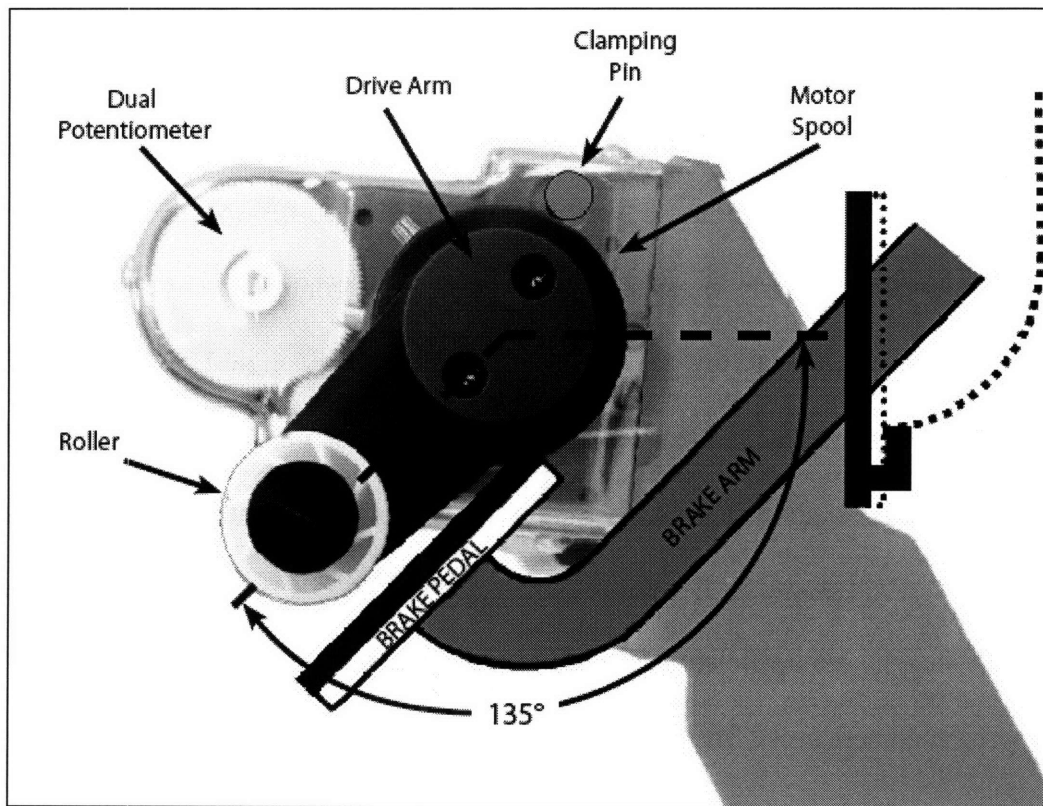


Figure D.1: EMC Gas/Brake Actuator

The position of the servomechanism, and thus the position of either the gas or brake pedal, is determined by the DC voltage sent to the mechanism itself. These voltage ranges are

specified in Table C.2. The manufacturer specified neutral position for the actuator is a voltage of 2500 mV. Voltages greater than 2500 mV depress the gas pedal while voltages below 2500 mV depress the brake pedal. The full gas and full brake ranges, however, differ from vehicle to vehicle as the geometry of the vehicle's gas and brake pedals as well as the servomechanism's mounting position are quite different.

D.2 Control Loop and Speed Selection

The control law selected for speed tracking was a basic PID controller augmented with an anti-windup logic on the integral term. The Laplace transform of the control law is presented below:

$$U_{action} = K_p(v_{cmd} - v) + K_d\left(a_{cmd} - \frac{vs}{Ts+1}\right) + \frac{K_i}{s}(v_{cmd} - v) \quad (D.1)$$

In general the derivative term is generated by a numerical differentiation of velocity measurements; the result is then smoothed by a first order filter with a time constant of T . To simplify things further, the vehicle receives commands to increase speed as step inputs and commands to decrease speed as ramp downs. This behavior is simple to implement as stopping behaviors can now be created automatically based on where the vehicle is in the path. As an example, the vehicle can track that it has completed 50 meters of a 100 meter path. If comfortable braking is defined as a deceleration of 2 meters a second, then the required stopping distance is specified. When the stopping distance and the length remaining in the path are identical, the speed command is automatically ramped down at 2 m/s. This is the a_{cmd} variable that is present in equation D.1. When step increases in speed are sent, a_{cmd} is taken as 0. In practice, the required stopping distance for a comfortable stop (D_c) is created from the following function:

$$D_c = K_{csd}v^2 + \alpha v + d_{extra} \quad (D.2)$$

This is derived from the fact that the distance required for a vehicle to come to a complete stop from an initial speed v_o at a constant deceleration (a_{cmd}) is given as the following:

$$d = \frac{-v_o^2}{2a_{cmd}} \quad (D.3)$$

The term α in equation D.2 is the average time lag from when a braking force is requested and the actuator actually begins applying that force (taken as .57 seconds). The distance traveled during that period is αv . The final term, d_{extra} accounts for half the length of the car and a prediction of how the car will overshoot the specified stopping point. The algorithm, thus works as the following: the higher level planner dictates, using the road speed limit, the max attainable speed the vehicle is allowed to achieve. This is sent as a step input to the vehicle and it will immediately begin accelerating. As the vehicle accelerates, it continually checks the value of D_c , as specified in equation D.2. When D_c and the distance remaining in the path are equal, the vehicle receives a constant deceleration command (a_{cmd}) of $-K_{csd}/2$.

In practice this worked well for long paths in which the vehicle could actually achieve the desired maximum speed. Often, however, small paths would create a surging and stopping behavior in which the vehicle rapidly accelerated and then immediately began decelerating. To combat this, a minimum steady-state cruising time is defined (T_{min}). Assuming the vehicle accelerates at a constant rate (a_c), the total distance the vehicle would travel accelerating up to speed, cruising, and then decelerating to a stop is given by the following:

$$d = \frac{(v_c^2 - v_o^2)}{2a_c} + vT_{\min} + K_{csd}v_c^2 + \alpha v + d_{extra} \quad (D.4)$$

Here v_c is the target cruising speed. If v_c is the maximum allowable speed and d is less than the length of the path, then the target speed is left as the maximum allowable. If, however, d is greater than the length of the path, equation D.4 is solved for the cruising speed which would allow the vehicle to approximately achieve the minimum cruising time of T_{\min} . This is the new specified speed command for the vehicle. If it is desired that the vehicle not receive step increases in speed, but rather ramp up in speed, a_c in equation D.4 should be replaced with the desired acceleration.

Given the above control law and the speed selection logic, it is necessary to provide some precaution against windup of the integral term in equation D.1. Windup, in essence, is the rapid growth of the integral term when an actuator is saturated. As an example, the gas pedal may be fully saturated by the control law and the integrator will continue to grow. This can cause significant overshoot as it may take a considerable amount of time to “unwind” the integrator. The solution is to monitor if the actuator is being saturated. If the actuator is not fully actuated, then no action is taken and the controller can run as normal. If the actuator is saturated, however, the anti-windup term is added to the integrator and is proportional (using gain K_{wup}) to difference in the controller output and the actuator limit. The control law with the anti-windup logic is presented below in Figure D.2. It should be stressed that anti-windup is only used when the actuator is saturated, otherwise this term is zero. Additionally, it is often the case that the output of the actuator is checked in real time to determine if it is saturated. For this implementation, however, the maximum and minimum allowable output of the control law is none a priori and can

be easily compared to the output of the control law at time t to determine if the anti-windup term should be constructed. This will be further explained in the next section.

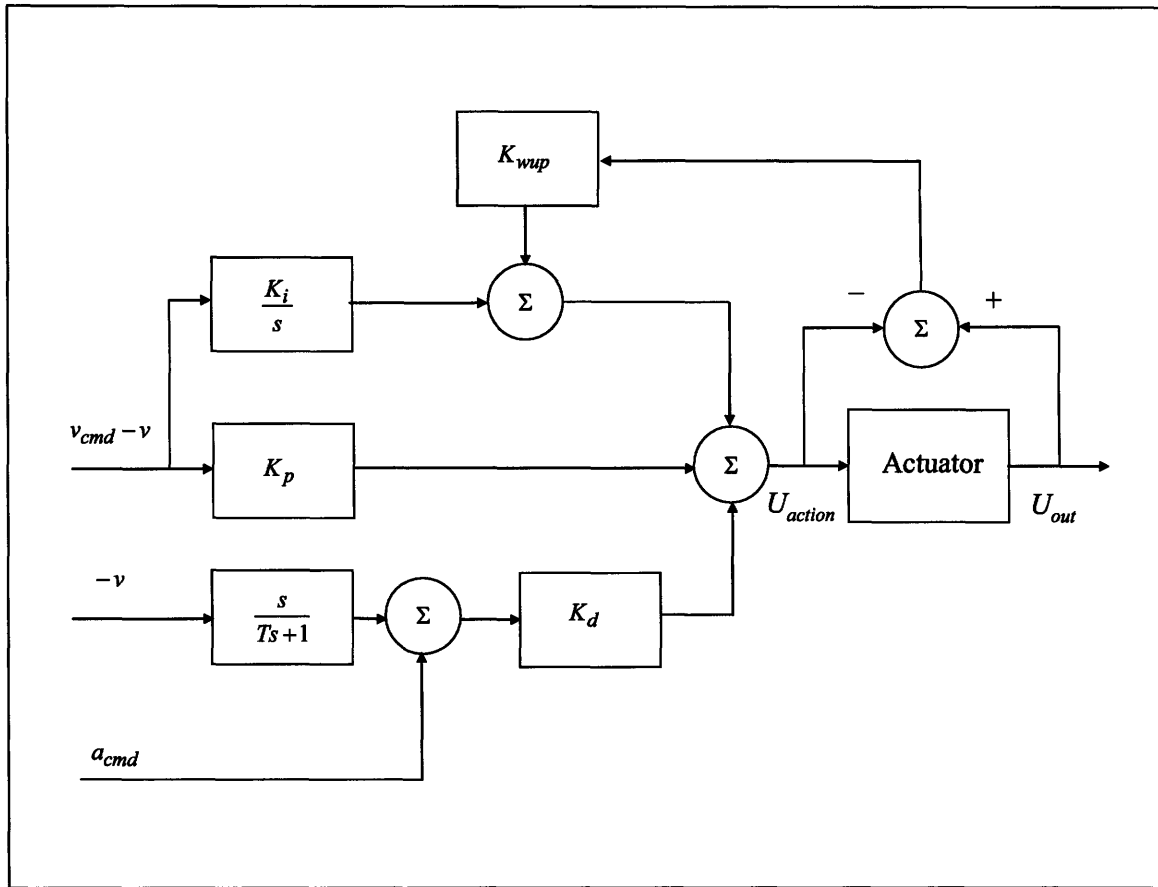


Figure D.2: Anti-Windup with PID Control Law

Even with anti-windup in place, it was found that the vehicle's response to a ramp down in speed was slow as it was necessary to still unwind the integral term, though the actuator was not saturated. To alleviate this problem, the integral term is reset every time the vehicle receives a commanded ramp down in speed.

D.3 Voltage Map

Given the PID control law, the output of the control law (U_{action}) is mapped to a voltage signal that is sent directly to the EMC servomechanism. Experimentation with the servomechanism and the gas and brake revealed that the servomechanism has a dead

zone in which it actuates neither the brake nor the gas pedal. This delay in the system is compounded by the fact that the brake and gas pedal, though physically pressed, have initial dead bands in which they provide neither a braking nor an accelerating force. Additionally, vehicles with automatic transmissions have an idling cruising speed in which they will move forward at a speed of 8 to 10 mph. To achieve speeds below this nominal cruising speed, it is necessary to actuate the brake. All of these factors impacted the development of a map from the control law output to the EMC actuator voltage.

In general, the dynamics of accelerating a vehicle and decelerating a vehicle are very different. This presents a challenge when tuning a single control law across both the brake and gas pedal. In designing the voltage map, the intention is to tune the gains K_p , K_d , and K_i around the vehicle's response to step increases in the speed command. A control law output U_o is to then be specified such that a U_{action} greater than U_o causes the vehicle to accelerate and a U_{action} less than U_o causes the vehicle to decelerate. To compensate for the difference in braking and acceleration dynamics, the map will be such that U_{max} and U_{min} produce an acceleration and deceleration which are approximately equal in magnitude. This is equivalent to multiplying the output of the control law by an additional gain when U_{action} is less than U_o .

The voltage map was then designed around five key voltages, V_{max} , V_{rest} , V_{brake} , V_{gas} , and V_{decel} . V_{max} is defined as the maximum allowable input voltage to the gas pedal. V_{brake} is the voltage that will just cause the brake pedal to begin decelerating the vehicle while V_{gas} is the voltage that will just press the gas pedal such that it causes the vehicle to accelerate. V_{rest} is then the voltage that will keep the vehicle at rest from initial startup. Generally, V_{brake} and V_{rest} have very similar values. Finally, V_{decel} is the voltage that

produces a deceleration that is approximately equal in magnitude to the acceleration produced by the input V_{max} .

The decision was made to design a voltage map that is piece-wise linear. Additionally, so that the vehicle will remain stationary when the controller is initiated, a control law output of zero is mapped to the voltage V_{rest} . The general form of the voltage map is then illustrated graphically in Figure D.3.

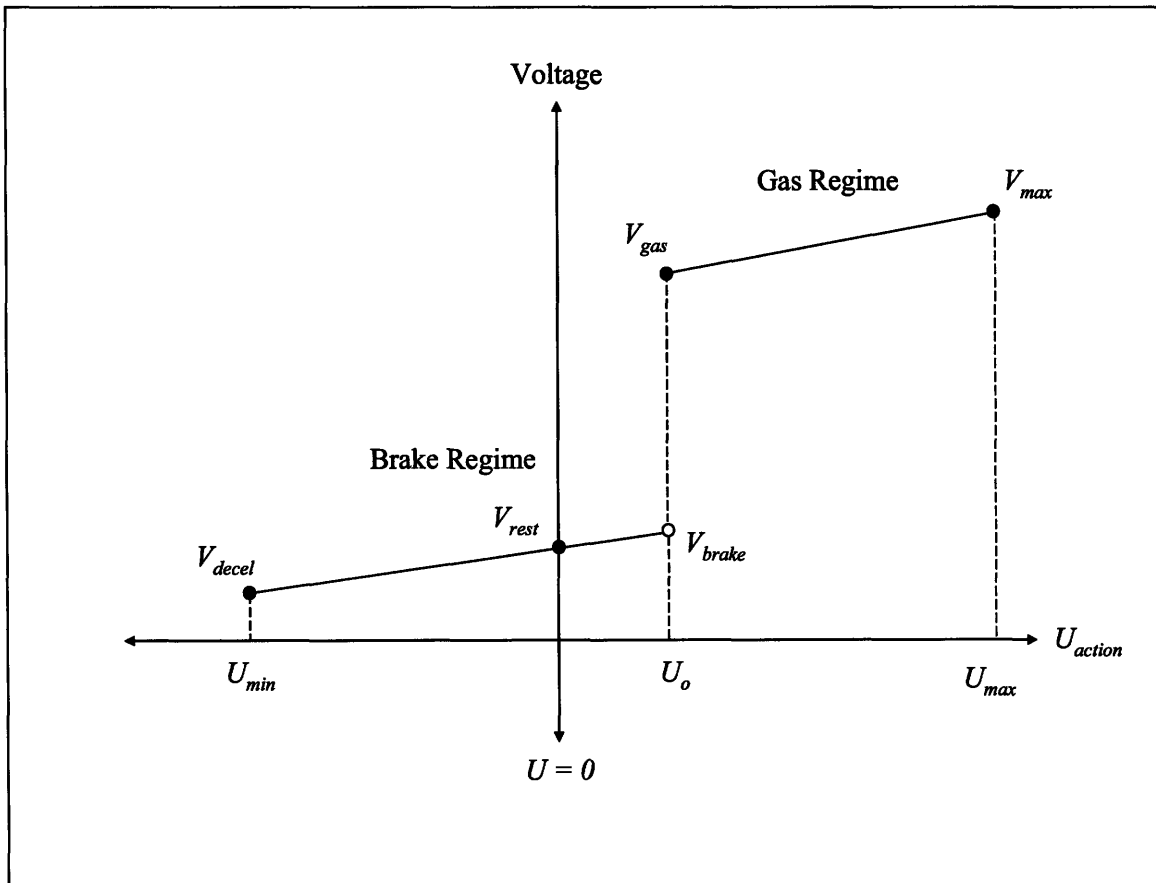


Figure D.3: General U_{action} to Voltage Map

A map of this form with the jump from v_{brake} to v_{gas} effectively attempts to remove the dead band from the actuators. Because the actuator can not move instantaneously from applying brake to applying throttle, some delay is present around a control law output of U_o . This results in a loss of phase margins in this region. However, the actuator dynamics

are significantly faster than those of the car, making it reasonable to ignore this delay in the system.

Given the design decisions in formulating this voltage map, the slope of the line joining V_{gas} and V_{max} is given as the following:

$$m_g = \frac{(V_{max} - V_{gas})}{U_{max} - U_o} \quad (D.5)$$

U_{min} is then defined as the following:

$$U_{min} = \frac{(V_{rest} - V_{decel})}{m_b} \quad (D.6)$$

In equation D.6, m_b is the slope of the brake regime. Additionally, U_{max} and U_{min} are taken as equal in magnitude. It should be noted here that U_{min} is a design parameter and not actually enforced; for safety reasons, the full range of motion of the brake pedal is preserved. The change over from applying brake to gas is then defined as the following:

$$U_o = \frac{(V_{brake} - V_{rest})}{m_b} \quad (D.7)$$

As a result of deciding to design the control law gains around the vehicle's response to step increases in the speed command, m_g is taken as 1. Combining D.6 and D.7 with D.5 and solving for m_b then yields the following:

$$m_b = \frac{(2V_{rest} - V_{decel} - V_{brake})}{(V_{max} - V_{gas})} \quad (D.8)$$

All parameters of the voltage map presented in Figure D.3 are now clearly defined. In practice, this map was quite effective at producing good acceleration and deceleration results. However, it was found that when the vehicle reaches the specified cruising speed, the inertia of the vehicle obviates the need for a significant throttle input to keep the vehicle at this speed. Indeed, the output of the engine needs to only compensate for

rolling friction and aerodynamic drag, resulting in a steady-state voltage that is very near V_{gas} . As a result, a small amount of noise in the velocity measurements and small disturbances in the road surface resulted in significant chatter from brake to gas as the output of the control law oscillated about U_o . Investigation of the voltage map reveals that this approach does not allow the vehicle to coast when neither gas nor brake are required to keep the vehicle rolling at the target speed. To compensate, some of the actuator's dead band was reintroduce as is illustrated in Figure D.4.

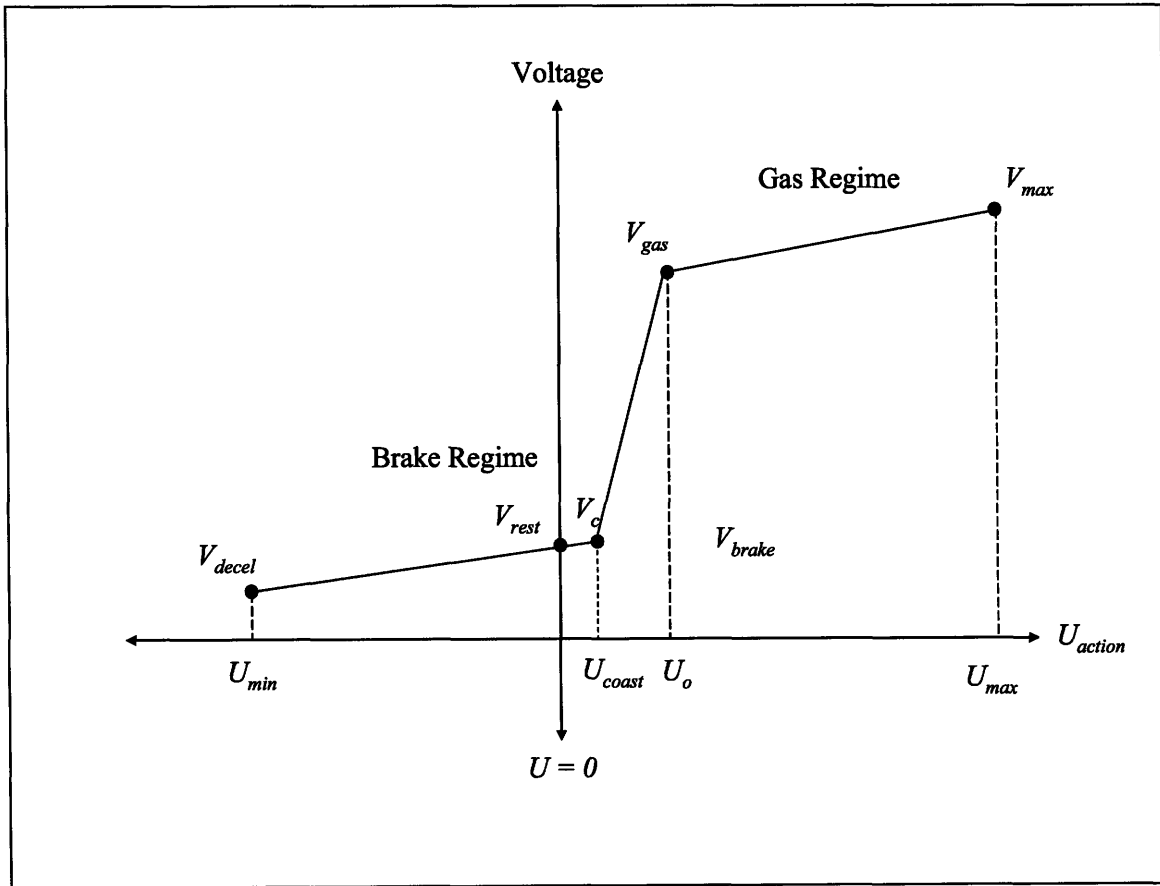


Figure D.4: Modified Voltage Map

U_{coast} is now introduced as a design parameter that determines the systems sensitivity to noise in the region near U_o . Once U_{coast} is specified, V_c follows below:

$$V_c = m_b U_{coast} + V_{rest} \quad (D.9)$$

The slope of the line joining V_c and V_{gas} is then given as the following:

$$m_c = \frac{V_{gas} - V_b}{U_o - U_{coast}} \quad (D.10)$$

In practice, U_{coast} was defined as a fraction of U_o . Values of all parameters required to define a voltage map for *Talos-I* and *Talos-II* are presented in Table D.1.

| | <i>Talos-I</i> | <i>Talos-II</i> |
|------------------|----------------|-----------------|
| V_{max} (mV) | 3900 | 4700 |
| V_{gas} (mV) | 2750 | 3250 |
| V_{brake} (mV) | 2300 | 1800 |
| V_{rest} (mV) | 2230 | 1700 |
| V_{decel} (mV) | 1500 | 1300 |
| U_{coast} | $.3U_o$ | $.3U_o$ |

Table D.1: Volatage Map Parameters for Team MIT

D.4 Speed Control Results

Presented in Figure D.5 and Figure D.6 are results from the speed controller. In both cases the controller produced nearly zero steady-state error. With a 6 m/s step input the rise time was approximately 4.15 seconds. Conversely, with a 13 m/s step input the rise time was approximately 9 seconds. This discrepancy is expected as the vehicle has a maximum acceleration limit that prevents consistent rise time across all step inputs. This is much the same way in which the steering actuator has a maximum rate of angular rotation.

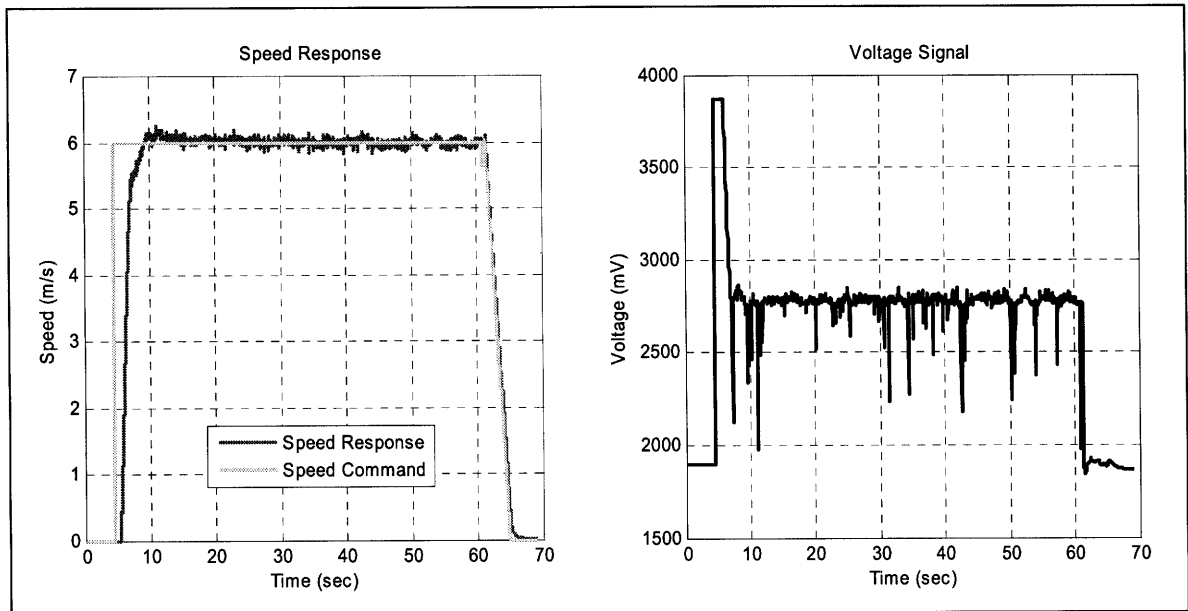


Figure D.5: Step Input of 6 m/s

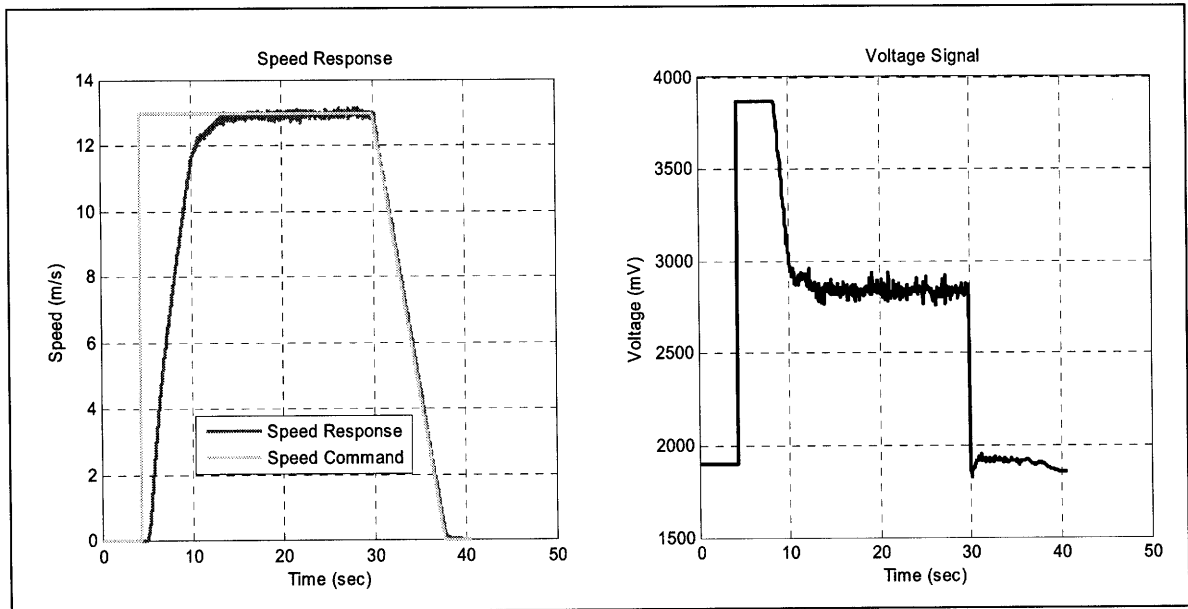


Figure D.6: Step Input of 13 m/s

The voltages for each run are also presented. With a 6 m/s step input it is clear that noise in the system causes the control signal to spike at many instances. However, because of the coasting regime in the voltage map, these spikes did not actually cause the brake to be actuated. Moreover, the in car driving was quite smooth. In the case of the 13 m/s step input the control signal is much smoother. Here the steady-state voltage necessary to keep

the vehicle moving forward is not as close to the U_o cross-over from brake to gas actuation. As a result, the noise in the system causes only minimal oscillation in the gas regime.

As a final consideration, because increases in speed are sent as step inputs, the controller exhibits a surging behavior when the vehicle accelerates from a stop. This results from the gas pedal initially being saturated. In Figures D.6, this occurs between 5 and 10 seconds when the voltage signal is at V_{max} . To help smooth driving for the occupants, the maximum allowable voltage was reduced from the manufacturer's specifications to one that produced a more natural acceleration. In practice, the need to do this would be obviated by sending commands to increase speed as ramp ups.