Development of a Robust Reference Path Generator

for Lateral Vehicle Control

T. Talsma

Master of Science Thesis

Delft University of Technology

Development of a Robust Reference Path Generator

for Lateral Vehicle Control

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

T. Talsma

June 4, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Cognitive Robotics (CoR)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Development of a Robust Reference Path Generator

by

T. Talsma

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering

Dated: June 4, 2018

Supervisor(s):

Prof. dr. ir. J. Hellendoorn

Dr. M. Alirezaei

Ir. A. Teerhuis

Reader(s):

Dr. ir. A. Hegyi

Dr. B. Shyrokau

Abstract

Automated vehicle control provides advantages in transport efficiency, redundancy, human-safety, flexibility, and parallelism. Extensive research has been dedicated to direct-following lateral control methods, using a single preview point as reference signal. However, these methods give rise to significant tracking errors. As alternative to a single point, a continuous path can be used as reference signal for vehicle control. Using a continuous reference path, accurate and comfortable control actions can be computed, while avoiding reference tracking errors. Therefore, robust reference path generation is an essential part of lateral vehicle control.

The goal of this research is to develop a generic, robust reference path generator for lateral vehicle control. Currently in path generation, the state-of-the-art method is based on repetitive polynomial fitting. This method inherently contains two main weaknesses. Firstly, it is not robust to sensor noise and other real-world disturbances. Secondly, as a result of the repetitive fitting, a discontinuous path is generated. This is undesirable, because it leads to an unfeasible reference path, since vehicles can only produce and track continuous trajectories. Moreover, a discontinuous reference path results in the need for path smoothing when used in comfortable lateral vehicle control. Fundamentally, this smoothing leads to inaccurate control, caused by manipulation of the original, discontinuous reference path. To overcome these weaknesses, a new Model-based Path Generation method is presented.

The development of a new, generic method for robust path generation is the main contribution of this research. This new path generation method is capable of producing feasible vehicle trajectories based on unfeasible waypoints. The performance of this method is evaluated in simulations and experiments, benchmarking it against polynomial fitting path generation. Furthermore, path generation robustness is assessed based on the outcome of a disturbance sensitivity analysis. The results are in accordance with the hypothesis stating that the new method outperforms the benchmark method in terms of path accuracy, robustness, continuity, and general applicability.

Table of Contents

1 General Introduction **1**

1-1 Background . . . 1

1-2 Problem statement . . . 2

1-3 Outline . . . 3

2 Field of Application **5**

2-1 Vehicle and Scenario Assumptions . . . 5

2-2 Waypoint Ego-Motion Compensation . . . 6

2-3 Disturbances . . . 9

2-4 Summary . . . 11

3 Polynomial Path Generation **13**

3-1 Polynomial Fitting . . . 13

3-1-1 Objective . . . 14

3-1-2 Implementation . . . 14

3-2 Modified Polynomial Fitting . . . 15

3-2-1 Heading Constraint . . . 16

3-2-2 Curvature Constraint . . . 17

3-3 Conclusions . . . 18

4 Model-based Path Generation **19**

4-1 Introduction to Model-based Path Generation . . . 20

4-2 Virtual Leader Model . . . 23

4-2-1 Design . . . 23

4-2-2 Initialization . . . 24

4-3 Proportional Feedback Controller . . . 26

4-3-1 Waypoint Transformation . . . 26

4-3-2 Proportional Control Law 27
4-4 Model Predictive Controller 28
4-4-1 Objective Function 28
4-4-2 Stability 30
4-5 Tuning Parameters 31
4-6 Summary 32

5 Simulations **33**
5-1 Performance Criteria 33
5-1-1 Error Calculation for Polynomial Fitting 34
5-1-2 Error Calculation for Model-based Path Generation 35
5-1-3 Discontinuity Quantification 36
5-2 Simulator description 36
5-3 Simulation Results 37
5-3-1 Simulation in the Ideal World 38
5-3-2 Simulation with real-world disturbances 38
5-3-3 Simulation with user-defined waypoints 41
5-3-4 Computational Demand 46
5-4 Conclusions 47

6 Sensitivity Analysis **49**
6-1 Design 49
6-2 Results 50
6-2-1 Waypoint Noise 50
6-2-2 Motion Parameter Noise 50
6-2-3 Waypoint Delay 52
6-2-4 Waypoint Offset 52
6-3 Conclusions 55

7 Experimental Analysis **57**
7-1 Scaled Vehicles 57
7-1-1 Used experimental data 58
7-1-2 Results 60
7-2 TNO Carlab 64
7-2-1 Used experimental data 65
7-2-2 Results 67
7-3 Conclusions 68

8 Conclusions and Recommendations **71**
8-1 Conclusions 71
8-1-1 Field of Application 72
8-1-2 Polynomial Path Generation 72
8-1-3 Model-based Path Generation 73
8-1-4 Simulations and Experiments 73
8-2 Recommendations 74

A Waypoint Source Modeling **77**
A-1 Sensor Models . . . 77
A-2 Disturbance models . . . 79
A-2-1 Waypoint Noise . . . 79
A-2-2 Waypoint Delay . . . 79
A-2-3 Waypoint Offset . . . 80
A-2-4 Motion Parameter Noise . . . 80

B Simulation Parameters & Results **83**
B-1 Bicycle Model Parameters . . . 83
B-2 Steering Input . . . 84
B-3 Path Generation Implementation Parameters . . . 85
B-4 Results . . . 87
B-4-1 Reconstruction errors . . . 87
B-4-2 Path Discontinuity . . . 88

C Experimental Parameters & Results **91**
C-1 Bicycle Model Parameters . . . 91
C-2 Results . . . 92
C-2-1 Path Discontinuity . . . 92

Bibliography **95**

Glossary **99**
List of Acronyms . . . 99
List of Symbols . . . 99

List of Figures

2-1 Hierarchy of Path Generation. 5

2-2 A visualization of the generic vehicle-following field of application. On board of the black follower-vehicle, the Standard Sensor Systems provide the longitudinal velocity u, vehicle yaw rate r and lateral velocity v. In practice, v is not measured but can be estimated or neglected [23]. The local reference frame, attached to the follower-vehicle at time step b, is denoted by F_b. Finally, the Waypoint Source provides the waypoint w^{F_b} used for path generation. 6

2-3 A visualization of the storage of waypoints. Waypoint i is denoted by w_i. The local reference frame, attached to the follower-vehicle, is denoted by F_b. The uncertainty about the exact position of the waypoints is indicated by the red ellipse around the measured point. This uncertainty may be caused by Waypoint Noise. 7

2-4 A visualization of the ego-motion of the black follower-vehicle between two consecutive time instances. Moreover, the necessary coordinate transformation of the waypoints $w_i^{F_b}$ is visualized. 7

2-5 Waypoint Noise causes the measurements to deviate from the blue ground truth path of the leader-vehicle. The waypoint position uncertainty caused by this noise is represented by the red ellipses around the points. Waypoint Delay causes the latest waypoint (farthest away from the follower-vehicle) to be different from the current position of the leader-vehicle. Finally, Waypoint Offset is caused by the detection of the leader-vehicle rear. This offset is given by l_r. 9

2-6 Illustration of the Waypoint Delay effect in vehicle-following. Figure A represents the situation at the time of measuring waypoint w. Figure B shows the incorrect waypoint w being received with a delay. Waypoint w has become incorrect because the follower-vehicle has moved during the measurement delay. Figure B also shows the desired situation, where incorrect waypoint w has been compensated for the movement of the follower-vehicle, obtaining correct waypoint w^*. 10

3-1 Hierarchy of Polynomial Path Generation. 13

3-2 A visualization of polynomial path generation. The leader-vehicle is blue, the follower-vehicle is black. The waypoints from the Waypoint Source are stored in a list. This list of waypoints is represented by the red dots. The uncertainty about the exact position of the waypoints is indicated by the red ellipses around the waypoints. This uncertainty may caused by measurement noise. A polynomial y_b is fitted through a fixed amount of waypoints from the list. The fitted polynomial is represented by the smooth red line and is expressed in the reference frame F_b. 14

3-3 Visualization of polynomial fitting path generation at two consecutive time instances. The noise-corrupted waypoints are represented by the red points. The red polynomial $y_{b-1}(x_{b-1})$ is computed at time step $b-1$. Consequently, the follower-vehicle travels distance ΔX_b. Meanwhile, the list of waypoints is updated: a new noise-corrupted measurement is added, which is represented by the green point. The oldest measurement is discarded. At the next time instance b, a new polynomial $y_b(x_b)$ is fitted, based on the updated list of waypoints. The visualization clearly shows that this repetitive polynomial fitting approach results in a path that is discontinuous in position, heading and curvature. 16

4-1 Hierarchy of Model-based Path Generation. 19

4-2 The light-blue vehicle represents a virtual, computer-simulated, leader-vehicle. The aim of this virtual vehicle is to construct a reference path based on given waypoints. This simulated virtual leader-vehicle drives at a time headway of t_v (in seconds) with respect to the Waypoint Source, the real leader-vehicle. The time headway t_v is directly related to the amount of waypoints between the virtual vehicle and the real leader-vehicle. The smooth, light-blue dotted line represents the path driven by the virtual leader-vehicle. Note that the virtual leader-vehicle must not exactly follow the red waypoints. The waypoints are noise-corrupted and do not exactly represent the desired path of Waypoint Source. The waypoint uncertainty caused by this noise is represented by the red ellipses around the waypoints. 20

4-3 Visualization of the different control objectives. Figure A represents the ideal vehicle-following scenario, where the waypoints are noise-free. In this scenario, the given waypoints reference should be tracked accurately. Note that Figure A also represents the control objective when the proposed Model-based Path Generation method is deployed outside the generic vehicle-following use-case, when a different Waypoint Source provides precise and accurate waypoints. Figure B represents the real-world case, where the given waypoints are noise-corrupted and should not be tracked exactly. Instead, the light-blue virtual leader-vehicle should drive a smooth trajectory, accurately imitating the dark-blue, real leader-vehicle. For both objectives, the virtual vehicle must produce a feasible path, meaning that the trajectory is in accordance with vehicle dynamics. 21

4-4 The steering control inputs are compared. A real leader-vehicle is assumed to perform a sine steering maneuver starting after 5 seconds. Consequently, the computed control inputs for both the Virtual Driver implementations are shown. The control inputs are time-shifted versions of the real leader-vehicle steering input, since the virtual vehicle drives at a certain time headway t_v (in seconds) with respect to the real leader-vehicle. 22

4-5 Virtual leader-vehicle model frame and state definitions 23

4-6 Visualization of follower-vehicle ego-motion compensation for virtual leader-vehicle modeling. 25

4-7 Visualization of Virtual Driver Proportional Control variable y_m. 27

4-8 A visualization of Model Predictive Control (MPC) for virtual leader-vehicle modeling. The future Virtual Leader Model state predictions are shown in transparent light-blue. MPC is based on minimizing future position state offsets from the waypoints. These offsets are denoted by $y_e^{M_{k+1}}$ ... $y_e^{M_{k+N}}$, where N is the prediction horizon. This minimization is realized by computing a sequence of optimal control inputs $\tilde{\delta}_k^*$. Only the first control input of this sequence is applied, visualized by the solid line in the control input optimization graph. Consequently, at the next time step, the optimization is run again and a new sequence of optimized control inputs $\tilde{\delta}_{k+1}^*$ is calculated. 29

5-1 A visualization of the reconstruction error y_e^b, which is given by the lateral distance between the real leader-vehicle path y_{gt} and the generated path y_b, evaluated at $x_b = 0$. The generated path y_b is the result of either Polynomial Fitting or Model-based Path Generation. 34
5-2 A visualization of the simulated scenario. The vehicles drive at separate lanes at $100\,\mathrm{km\,h^{-1}}$. The follower-vehicle follows the leader-vehicle at a headway time of $1.3\,\mathrm{s}$. In meters, this headway is: $1.3\,\mathrm{s} \cdot 27.78\,\mathrm{m\,s^{-1}} = 36.11\,\mathrm{m}$. 37
5-3 The computed Virtual Driver Steering inputs in the ideal-world are shown and compared to the real leader-vehicle steering input. In the scenario of the left plot, the leader-vehicle performs the given sine steering input (and the follower-vehicle drives straight). Clearly, both Virtual Drivers approach the real leader-vehicle's steering input, shifted in time. In the scenario of the right plot, the leader-vehicle drives straight (and the follower-vehicle performs the given sine steering input). Note the different scale on the vertical axis. 39
5-4 The performance of the different path generators in the ideal-world is shown. The real-world disturbances are absent and the leader-vehicle is performing the sine steering maneuver, while the follower-vehicle drives straight. The reconstruction errors are plotted as functions of simulation time. All graphs start at the time instance at which the Waypoint Storage subsystem contains sufficient waypoints. 40
5-5 The computed Virtual Driver Steering inputs in the real-world simulation are shown, compared to the real leader-vehicle steering input. In the scenario of the left plot, the leader-vehicle performs the given sine steering input (and the follower-vehicle drives straight). In the scenario of the right plot, the leader-vehicle drives straight (and the follower-vehicle performs the given sine steering input). Due to the real-world disturbances, the Virtual Driver inputs differ from the real leader-vehicle's steering input. 41
5-6 The performance of the different path generators in the real-world simulation is shown. The real-world disturbances are Waypoint Noise, Delay, Offset and Motion Parameter Noise. The leader-vehicle is performing the sine steering maneuver, while the follower-vehicle drives straight. The reconstruction errors are plotted as functions of simulation time. Clearly, the new Model-based Path Generation Methods 4 and 5 provide continuous paths with and small reconstruction errors, in contrast to the Polynomial Fitting Methods 1, 2 and 3. All graphs start at the time instance at which the Waypoint Storage subsystem contains sufficient waypoints. 42
5-7 The generated path variables y_b, ψ_b, and κ_b, from simulating the real-world scenario, are shown. Method 2 clearly shows a continuous heading variable ψ_b, which was expected from the heading constrained polynomial fitting method. Similarly, curvature constrained fitting Method 3 clearly shows a continuous curvature variable κ_b. Overall, the plots show the that new Model-based Path Generation Methods 4 and 5 provide more continuous paths, compared to the Polynomial Fitting Methods 1, 2 and 3. 43
5-8 This figure shows the incapability of the polynomial path generation methods to provide a smooth, continuous path, based on user-defined (extreme maneuver) unfeasible waypoints. The path resulting from the repetitive polynomial fitting is discontinuous because of the jumps in computed polynomial coefficients. This phenomenon was visualized in Fig. 3-3 44
5-9 The computed Virtual Driver steering actions are shown. Clearly, the MPC Virtual Driver starts steering earlier, anticipating on future offsets from the user-defined waypoint reference path. The Proportional Control lets errors (at a virtual look-ahead-distance) occur first, prior to taking action. 45
5-10 This figure shows the capability of the Model-based Path Generation methods to provide a smooth, continuous path, based on user-defined (extreme maneuver) unfeasible waypoints. The Proportional Control Virtual Driver implementation results in a feasible path with a small overshoot, whereas the MPC implementation provides the desired reference tracking behavior. 45

5-11 A snapshot of the simulated scenario. The user-defined (unfeasible) waypoints are given by the black dotted line. An instantaneous unconstrained polynomial fit is shown by the red curve. The new Model-based Path Generation with the MPC Virtual Driver provides the desired path generation functionality. The resulting Virtual Leader Model trajectory is given by the light-blue line, representing the state-history of the light-blue virtual-vehicle. 46

6-1 Waypoint Noise Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied. 51

6-2 Motion Parameter Noise Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied. 53

6-3 Waypoint Delay Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied. 54

6-4 Waypoint Offset Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied. 56

7-1 The Delft Scaled Vehicle (DSV), with 1: Arduino, 2: Lidar, 3: Onboard computer, 4: Inertial Measurement Unit (IMU), 5: Router, 6: Battery, 7: Servo, 8: Motor controller, 9: Motor, 10: Encoders. Visualization courtesy of Mart Baars. 58

7-2 A visualization of the paths driven by the scaled vehicles, obtained by the Motion Capture system with an average sampling time of $0.0127\,\mathrm{s}$. During the experiment, the leader-vehicle drove in front of the follower-vehicle. The graph is to be interpreted from left to right, with increasing global X-coordinate. 58

7-3 A visualization of the heading of the scaled vehicle paths, obtained by the Motion Capture system with an average sampling time of $0.0127\,\mathrm{s}$. 59

7-4 A visualization of the Waypoint Source data, obtained using relative position information from the Motion Capture system. The sampling time is set to $0.1\,\mathrm{s}$. . . 59

7-5 A visualization of the logged data of the longitudinal velocity and yaw rate of the follower-vehicle. The average sampling times of the longitudinal velocity and yaw rate data are $0.0261\,\mathrm{s}$ and $0.0099\,\mathrm{s}$ respectively. 60

7-6 The computed Virtual Driver steering inputs for the Virtual Leader Model are shown. The Proportional Control inputs are limited to a maximum absolute value of $1\,\mathrm{rad}$ for stability purposes. 60

7-7 A visualization of the scaled vehicle experiment, where the follower-vehicle overtakes the virtual vehicle. The follower-vehicle is black, the virtual vehicle is light-blue and the leader-vehicle is dark blue. Furthermore, the ground truth leader-vehicle path is given by the green line, obtained from the Motion Capture system. The waypoints from the Waypoint Storage subsystem are visualized by the red dots. A single unconstrained polynomial fit of Method 1 is visualized by the solid red curve. Furthermore, the path resulting from Model-based Path Generation Method 5 is given by the black line, representing the position state history of the virtual vehicle. The visualization also shows encoder latency, as the waypoints do not exactly lie on the green ground truth path. Moreover, packet loss causes a varying distance between the waypoints. 61

7-8 The performance of the different path generators in the scaled vehicle experiment. The reconstruction errors are plotted as functions of simulation time. All graphs at the time instance at which the Waypoint Storage subsystem contains sufficient waypoints. 62

7-9 TNO Carlab Prius vehicles. 64

7-10 A visualization of the path driven by the follower-vehicle, i.e., the second Toyota Prius vehicle. The figure is generated using MATLAB code by TNO. 65

7-11 A visualization of the logged data of the relative position measurement of the leader-vehicle with respect to the follower-vehicle. 66

7-12 A visualization of the logged data of longitudinal velocity and yaw rate of the follower-vehicle. 66

7-13 A visualization of the logged data from the lane-marking detection system, on board of the follower-vehicle. 67

7-14 The computed Virtual Driver steering inputs for the Virtual Leader Model are shown, which are based on the waypoints obtained using TNO's experimental data. 68

7-15 The generated path variables y_b, ψ_b, and κ_b are shown. The results were obtained by running the path generators based on TNO's experimental data. 69

A-1 This figure visualizes the black follower-vehicle and the blue leader-vehicle, with their local reference frames F and L respectively. The position of the frames, with respect to the global frame G, is given by P_F and P_L. The orientation of the vehicles, with respect to the global frame G, is given by ψ_F and ψ_L. The relative position of the leader-vehicle, with respect to the follower-vehicle, expressed in frame F is denoted by $p_L^{F,F}$. 78

B-1 This figure visualizes the steering input given in the simulation scenarios. For the leader-vehicle, this input is mirrored in the horizontal axis, to let it initiate a lane-change to the right, rather than to the left. 84

B-2 This figure shows the difference in generated path variables y_b, ψ_b, and κ_b, from simulating the real-world scenario. The real-world disturbances are Waypoint Noise, Delay, Offset and Motion Parameter Noise. The leader-vehicle is performing the sine steering maneuver, while the follower-vehicle drives straight. Clearly, the new Model-based Path Generation Methods 4 and 5 provide more continuous paths, compared to the Polynomial Fitting Methods 1, 2 and 3. 89

C-1 The differences in generated path variables y_b, ψ_b, and κ_b are shown. The results were obtained by running the path generators based on the scaled vehicle experimental data. 93

C-2 The differences in generated path variables y_b, ψ_b, and κ_b are shown. The results were obtained by running the path generators based on TNO's experimental data. 94

List of Tables

5-1 Shorthand notation for different path generation methods. 38

5-2 The computation time is shown, required for $20\,\mathrm{seconds}$ of simulated driving. . . 46

7-1 Maximum occurring reconstruction errors with y_e in m, ψ_e in rad. 61

7-2 Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}, using the scaled vehicle experimental data. The ground truth curvature difference is undefined, since the path curvature is not measured by the Motion Capture system. 63

7-3 Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}. The results are obtained by the approach of running the path generators based on TNO's experimental data. 68

A-1 Expected real-world disturbance values. 80

B-1 Bicycle model parameters used for the vehicles in the custom-built simulator. . . 83

B-2 Virtual Leader Model single-track parameters used in the scaled vehicle experiments. These values are obtained by system identification performed by Mart Baars for the DSV. 84

B-3 Maximum occurring reconstruction errors with y_e in m, ψ_e in rad and κ_e in m^{-1}. 87

B-4 Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}. The simulation is given by the real-world, leader sine maneuvering scenario (simulation time 15 to 20 seconds). Comparing Method 5 to Method 1, for lateral position (y_b), the maximum occurring jump is smaller by one order of magnitude. For the heading (ψ_b) and curvature (κ_b), the decrease in discontinuity is by two orders of magnitude. 88

B-5 Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}. The simulation is given by the extreme maneuver scenario with user-defined waypoints. Comparing Method 5 to Method 1, for lateral position (y_b), the maximum occurring jump is smaller by one order of magnitude. For the heading (ψ_b) and curvature (κ_b), the decrease in discontinuity is by three orders of magnitude. 88

C-1 Virtual Leader Model single-track parameters used in the scaled vehicle experiments. These values are obtained by system identification performed by Mart Baars for the DSV. 91

Chapter 1

General Introduction

This chapter gives an introduction to the rest of this thesis. Section 1-1 starts with background information on the subject of path generation. Furthermore, Section 1-2 describes the main problem that is addressed in this research. Finally, Section 1-3 outlines the structure of the rest of this thesis.

1-1 Background

Nowadays, the pressure on existing road networks is increasing. Possible solutions are increasing the road capacity or increasing the efficiency of road usage [24]. However, increasing the road capacity may not be possible, due to spacial constraints, especially in urban areas. Moreover, increasing road capacity could lead to intensifying road usage [16], which leads to environmental concerns. This indicates the symptom fighting character of the capacity increasing solution. Therefore, increasing the efficiency of the road usage is the most tenable solution [5]. Increased road usage efficiency can be realized by reducing the inter-vehicular distances, without reducing operating speeds. However, with a human driver, the effect is limited and the human workload of driving is increased, due to the slow human reaction time [12]. The solution to this is given by automated vehicle control, which is realized by systems that autonomously control the steering, throttle and braking actions in a vehicle [18]. The longitudinal control problem is solved already, and current research focuses mostly on reliability issues of the longitudinal control [8], [19]. However, the lateral guidance problem is still the subject of ongoing research [17]. Extensive research has been dedicated to direct-following lateral control methods, using a single preview point as reference signal [1]. These approaches are effective when the reference point is close to the current position of the vehicle [20]. In other scenarios, these approaches will give rise to significant tracking errors, e.g., cutting corners [14]. Thus, for the general case of automated vehicle control, direct point-following approaches do not provide optimal performance. As alternative to a single point, a continuous path can be used as reference signal for lateral vehicle control. Using a continuous reference path, the tracking errors can be avoided. Based on this path, the lateral vehicle control systems can compute continuous steering, throttle and braking actions. These actions will be such that the deviation from the reference path is minimized, while preserving driving comfort.

The ambition of the research-oriented company TNO is to help create livable, sustainable cities through clean, safe, reliable and affordable mobility and logistics. With this aim, TNO has experimented with reference path generation for automated vehicle-following. Automated vehicle-following is one of the many applications requiring lateral vehicle control. The development of automated vehicle-following will improve the public transportation networks, and allows the conception of automated highways, providing transportation efficiency, reduced fuel consumption and decreased pollution [11]. Vehicle-following can also be used in freight ports where transport vehicles can travel closely, yet safely, to carry containers from ships to docks [25]. However, not only transportation, but also other operations, such as exploration [15], search and rescue [27] and agriculture operations [26] will benefit from automated vehicle-following, due to the advantages it gives in redundancy, human-safety, flexibility, and parallelism.

This research deals with the development of a generic, robust reference path generator for lateral vehicle control. TNO's state-of-the-art polynomial fitting path generator is used as benchmark method. The method is based on fitting a third order polynomial to a set of given waypoints.

1-2 Problem statement

The main issue at hand is that the state-of-the-art polynomial fitting path generation solution does not provide desirable results [10]. This is caused by its two main weaknesses. Firstly, the current system is not robust to sensor noise and other real-world disturbances. Secondly, the system is based on repetitive least-squares polynomial fitting, resulting in discontinuous path generation. This is undesirable, because it leads to an unfeasible reference path, since vehicles can only track and produce continuous trajectories. Moreover, a discontinuous reference path results in the need for path smoothing when used in comfortable lateral vehicle control. Fundamentally, this smoothing leads to inaccurate vehicle control, caused by manipulation of the original reference path. These problems lead to the definition of the main research question, addressed in this thesis:

How can a feasible, continuous vehicle reference path be generated, based on a set of given waypoints?

The goal of this research is to develop a generic, robust reference path generator for lateral vehicle control. This path generator must meet a number of minimal requirements. Firstly, it must provide an accurate and feasible path. Secondly, it must be robust in the sense that is has proper disturbance rejection characteristics. Finally, the resulting path must be continuous to allow for accurate and comfortable lateral vehicle control. It is hypothesized that this generic, robust reference path generator outperforms the benchmark method.

1-3 Outline

Chapter 2 describes the generic field of application for which the main problem is solved. In addition, the corresponding challenges are addressed.

Chapter 3 presents the benchmark path generation method. The benchmark method is given by the state-of-the-art polynomial fitting path generation method. In addition, two approaches to improve the benchmark method are proposed.

Just like any method, polynomial fitting path generation has its limitations. It is hypothesized that a new path generation method can be conceived, which can outperform polynomial fitting path generation. The new method will not rely on fitting a mathematical function. Instead, the method will be based on vehicle modeling.

Chapter 4 presents a new model-based solution to the path generation problem. Two different implementations for Model-based Path Generation are presented, using proportional feedback and the Model Predictive Control (MPC) framework.

Next is the challenge of selecting the best path generation method. For this, a comparing analysis is needed. To assess the performance of different path generating methods, performance criteria must be defined. Furthermore, the sensitivity of the systems to real-world disturbances must be evaluated, to assess the feasibility and robustness of real-world implementation. The evaluation of the performance criteria, as well as the sensitivity analysis, will be based on numerical simulations and experiments.

Chapter 5 presents the design of the path generation performance criteria. Furthermore, the results from path generation simulations are presented.

Chapter 6 presents the design of the path generation sensitivity analysis. Next, the sensitivity analysis results are reported, identifying and comparing the robustness of the different path generators.

Chapter 7 elaborates on the experiments that confirm the feasibility of the path generators and verify the results obtained from simulations.

Finally, Chapter 8 concludes this thesis by discussing the main findings and presenting recommendations.

Chapter 2

Field of Application

This chapter describes the generic path generation field of application. Section 2-1 describes the assumptions regarding the generic field of application. Subsequently, corresponding path generation challenges are addressed in Sections 2-2 and 2-3. Finally, Section 2-4 presents a summary of this chapter.

2-1 Vehicle and Scenario Assumptions

A path generation system consists of three main subsystems. The hierarchy of the subsystems is visualized in Fig. 2-1. To compute a reference path, the Path Generation subsystem uses a set of discrete waypoints obtained from the Waypoint Storage subsystem. This set of waypoints can originate from various Waypoint Sources. For instance, in automated vehicle-following, the source is given by sensor systems that detect a leader-vehicle. Alternatively, the waypoints may be defined by a human operator or originate from a strategic path planning layer [4]. Fundamentally, waypoints obtained from sensor systems are noise-corrupted. Therefore, a generic path generation system must have noise-rejecting characteristics. In this way, the path generation system is applicable irrespective of the specific Waypoint Source.

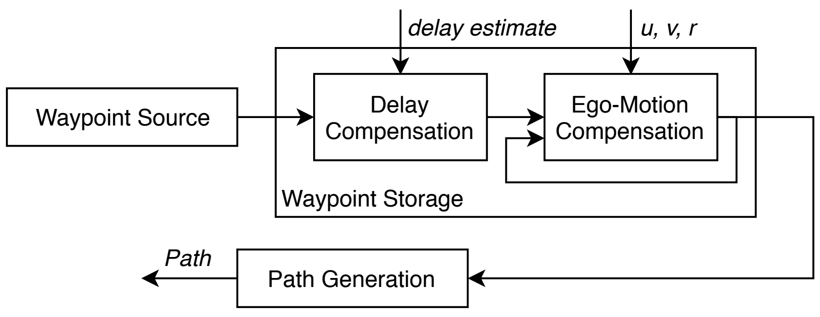


Fig. 2-1: Hierarchy of Path Generation.

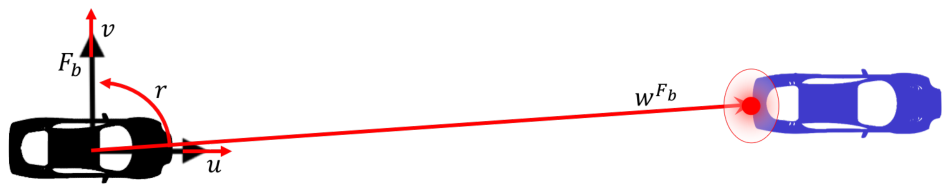


Fig. 2-2: A visualization of the generic vehicle-following field of application. On board of the black follower-vehicle, the Standard Sensor Systems provide the longitudinal velocity u, vehicle yaw rate r and lateral velocity v. In practice, v is not measured but can be estimated or neglected [23]. The local reference frame, attached to the follower-vehicle at time step b, is denoted by F_b. Finally, the Waypoint Source provides the waypoint w^{F_b} used for path generation.

In this research, the path generation problem is solved for the generic vehicle-following field of application. This application entails a continuously changing set of waypoints and a continuously changing, vehicle-attached, local frame of reference. Consequently, the path generation solution is also applicable in use-cases where the set of waypoints and reference frame are stationary.

In the generic vehicle-following field of application, the path generation system is assumed to use discrete relative leader-vehicle position measurements as waypoints. This assumption is made because it allows for evaluation of path generation performance. The generated path can be interpreted as a reconstruction of the leader-vehicle's factual path. Consequently, path generation performance is related to the accuracy of reconstruction. This approach does not impose any limitations on the system; that is, the applicability remains irrespective of the specific Waypoint Source. The waypoints may be obtained from radar and camera sensor systems on board of the follower-vehicle, detecting the leader-vehicle. Next to a Waypoint Source, onboard ego-motion sensor systems (e.g., wheel-speed, steering angle and gyro sensors) are assumed. These systems are required to compensate the waypoints for the ego-motion of the follower-vehicle, which is further explained in Section 2-2. Current commercially available vehicles are assumed to be equipped with these sensor systems. Therefore, these sensors will be referred to as Standard Sensor Systems. No specific assumptions are made regarding the leader-vehicle, other than its longitudinal position with respect to the follower-vehicle, forming the illustrating Waypoint Source. The availability of information from GPS, roadside or inter-vehicular communication is not assumed. Note that in use-cases where this information is available, it could be used to improve the accuracy of the generated path. The generic field of application is visualized in Fig. 2-2.

2-2 Waypoint Ego-Motion Compensation

The Waypoint Source provides the waypoints used for onboard path generation, expressed in the local, vehicle-attached, planar coordinate frame. For path generation, multiple waypoints are required. Therefore, the waypoints need to be stored on board of the follower-vehicle. This is visualized in Fig. 2-3. The waypoint storing functionality is covered by the Waypoint Storage subsystem. The waypoints can be mathematically described by position coordinates expressed in the local follower-vehicle reference frame F_b. Let $\tilde{\mathbf{w}}^{F_b}$ denote a matrix of $n+1$

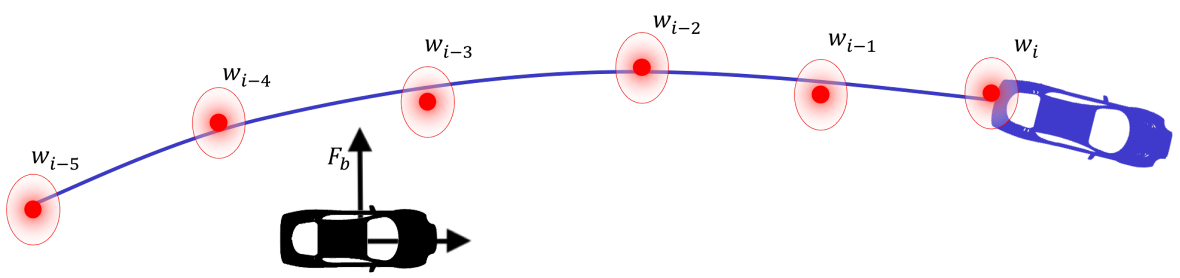


Fig. 2-3: A visualization of the storage of waypoints. Waypoint i is denoted by w_i. The local reference frame, attached to the follower-vehicle, is denoted by F_b. The uncertainty about the exact position of the waypoints is indicated by the red ellipse around the measured point. This uncertainty may be caused by Waypoint Noise.

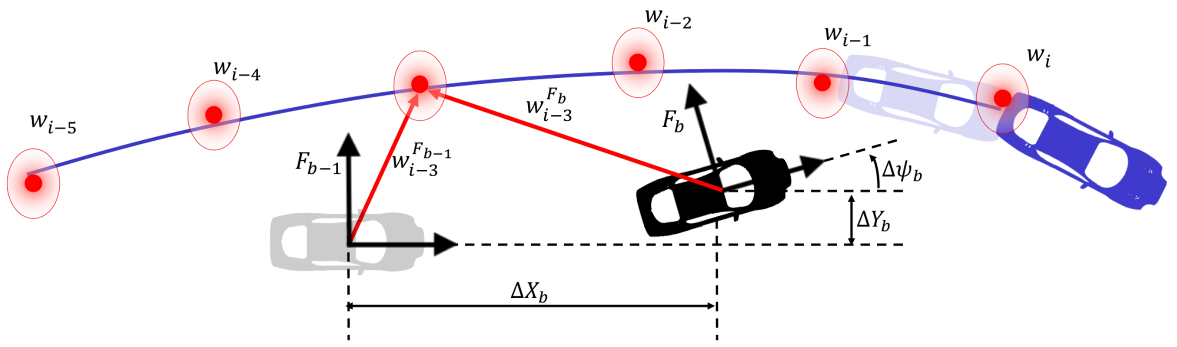


Fig. 2-4: A visualization of the ego-motion of the black follower-vehicle between two consecutive time instances. Moreover, the necessary coordinate transformation of the waypoints $w_i^{F_b}$ is visualized.

waypoints expressed in F_b, given by:

$$\tilde{\mathbf{w}}^{F_b} = \begin{bmatrix} w_{i,x} & w_{i-1,x} & \dots & w_{i-n,x} \\ w_{i,y} & w_{i-1,y} & \dots & w_{i-n,y} \\ 1 & 1 & \dots & 1 \end{bmatrix}, \tag{2-1}$$

where $w_{i,x}$ and $w_{i,y}$ represent the x- and y-coordinate of waypoint i respectively. The distance between two consecutive waypoints depends on the specific Waypoint Source. For instance, in vehicle-following, it depends on the sampling frequency of the Standard Sensor Systems and the speed of the leader-vehicle. For example, assuming a sampling frequency of 10 Hz and a speed of $100\,\mathrm{km\,h^{-1}}$, this distance is approximately 2.78 m.

In driving situations, vehicles are in motion. This means that the position and the orientation of local reference frame F_b changes over time. As a result, the waypoints must continuously be remapped to the latest local frame F_b. Let Δt_b denote time difference between time steps $b-1$ and b. The Waypoint Storage subsystem manages this mapping, by compensating the stored waypoints for the ego-motion of the follower-vehicle, using its ego-motion parameters obtained from the Standard Sensor Systems. The mapping is given by:

$$\tilde{\mathbf{w}}^{F_b} = \mathbf{T}^{F_b}_{F_{b-1}} \tilde{\mathbf{w}}^{F_{b-1}}, \tag{2-2}$$

where $\mathbf{T}^{F_b}_{F_{b-1}}$ represents the homogeneous transformation matrix, mapping vectors from follower-vehicle frame F_{b-1} to F_b. This matrix is given by:

$$\mathbf{T}^{F_b}_{F_{b-1}} = \begin{bmatrix} \mathbf{R}^{F_b}_{F_{b-1}} & -\mathbf{R}^{F_b}_{F_{b-1}} \cdot \Delta\mathbf{O}^{F_{b-1}}_{F_b} \\ 0_{1\times 2} & 1 \end{bmatrix}. \tag{2-3}$$

The rotation matrix $\mathbf{R}^{F_b}_{F_{b-1}}$ and the relative relative displacement of the coordinate frame $\Delta\mathbf{O}^{F_{b-1}}_{F_b}$ are given by the following equations:

$$\mathbf{R}^{F_b}_{F_{b-1}} = \begin{bmatrix} \cos\Delta\psi_b & \sin\Delta\psi_b \\ -\sin\Delta\psi_b & \cos\Delta\psi_b \end{bmatrix}, \tag{2-4}$$

$$\Delta\mathbf{O}^{F_{b-1}}_{F_b} = \begin{bmatrix} \Delta X_b \\ \Delta Y_b \end{bmatrix}. \tag{2-5}$$

The relative coordinate frame changes $\Delta\psi_b$, ΔX_b and ΔY_b are defined in Fig. 2-4. These changes can be computed by using the motion parameters from the Standard Sensor Systems:

$$\Delta\psi_b = \int_t^{t+\Delta t_b} r \cdot dt, \tag{2-6}$$

$$\Delta X_b = \int_t^{t+\Delta t_b} u \cdot dt, \tag{2-7}$$

$$\Delta Y_b = \int_t^{t+\Delta t_b} v \cdot dt, \tag{2-8}$$

where r represents the yaw rate and u the longitudinal speed of the vehicle. In practice, the lateral velocity v is not measured; instead, it can either be estimated or neglected [23].

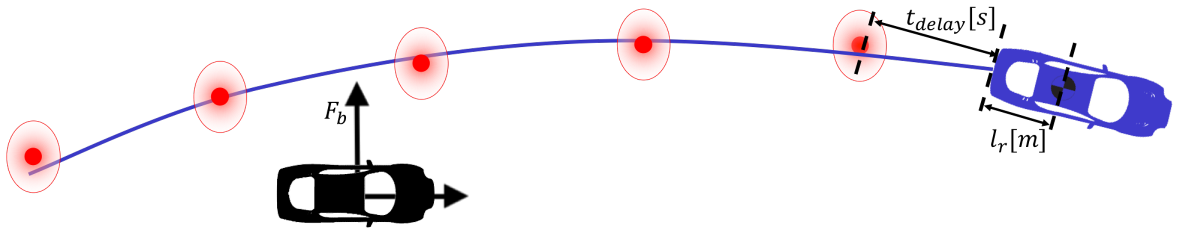


Fig. 2-5: Waypoint Noise causes the measurements to deviate from the blue ground truth path of the leader-vehicle. The waypoint position uncertainty caused by this noise is represented by the red ellipses around the points. Waypoint Delay causes the latest waypoint (farthest away from the follower-vehicle) to be different from the current position of the leader-vehicle. Finally, Waypoint Offset is caused by the detection of the leader-vehicle rear. This offset is given by l_r.

2-3 Disturbances

The input to a path generation system is given by a set of waypoints from the Waypoint Source. In an ideal situation, these waypoints are precise and accurate. However, in reality, they may be uncertain or corrupted with (sensor) Waypoint Noise. In addition, the path generation system may be subject to Waypoint Delay and Waypoint Offset. These disturbances are visualized in Fig. 2-5. Moreover, in the real world, the ego-motion parameters from the Standard Sensor Systems are generally noise-corrupted. These disturbances are further explained in the next paragraphs. The impact of other types of disturbances (e.g. road grade or bank angle) is not investigated here, and is left for future research, because it can reasonably be expected that it is small compared to the impact of Waypoint Noise, Offset, Delay, and Motion Parameter Noise, due to the dominant planar motion of vehicles. The distinction between ideal- and real-world is made to be able to analyze the performance and robustness of path generation systems. The performance and robustness aspects are further discussed in Chapters 5 and 6.

Waypoint Noise

Due to the possible existence of uncertainty or sensor noise on a single waypoint, there may exist an error between the measured (or desired) position of the waypoint and its actual, given position. This phenomenon is illustrated in Figure 2-5. As a result, Waypoint Noise is considered the first real-world disturbance on a generic path generation system.

Waypoint Delay

The Waypoint Source may provide the waypoints with a certain time delay t_{delay}. For instance, in vehicle-following, this means that the time instant of measuring differs from the time instant at which the waypoint is available to the path generation system. The time instant at which the measuring takes place and waypoint w is obtained is visualized in Fig. 2-6A. The time instance at which this w is actually received is visualized in Fig. 2-6B. Because of the follower-vehicle ego-motion, during the time gap between A and B, the waypoint w has become incorrect. As a result, Waypoint Delay is considered the second real-world disturbance

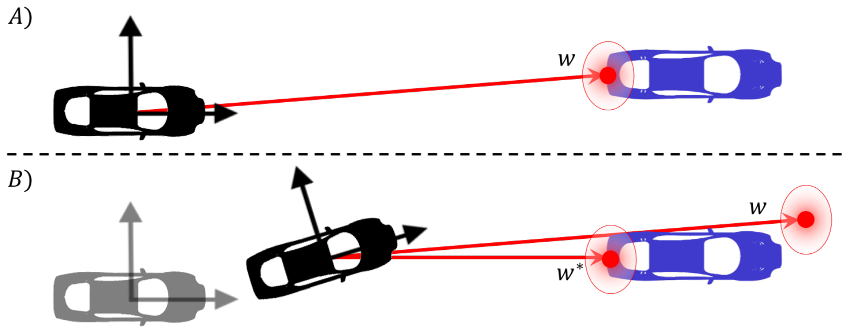


Fig. 2-6: Illustration of the Waypoint Delay effect in vehicle-following. Figure A represents the situation at the time of measuring waypoint w. Figure B shows the incorrect waypoint w being received with a delay. Waypoint w has become incorrect because the follower-vehicle has moved during the measurement delay. Figure B also shows the desired situation, where incorrect waypoint w has been compensated for the movement of the follower-vehicle, obtaining correct waypoint w^*.

on a generic path generation system. The correct position of the waypoint is visualized in Fig. 2-6B by w^*. To compensate w, and obtain the correct waypoint w^*, remapping is needed; in fact, this challenge was already addressed in Section 2-2. The only difference for Waypoint Delay compensation is the time duration of integration of ego-motion parameters, given by (2-6) to (2-8). To compensate for Waypoint Delay, this time duration Δt_b is set equal to the delay estimate t_{delay}.

Waypoint Offset

Typically, the trajectory of the rear of a vehicle differs from the trajectory of its Center of Gravity (CoG). From the vehicle-following scenario, it follows that the sensor systems on board of the follower-vehicle detect the rear of the leader-vehicle. Therefore, the vehicle-following path generation system can only be based on the relative position measurements of the rear of the leader-vehicle. As a result, the reconstructed path will be that of the leader-vehicle rear, rather than its CoG. As a result, Waypoint Offset is considered the third real-world disturbance.

Motion Parameter Measurement Noise

To be able to use the waypoints for path generation purposes, the waypoints must continuously be compensated for ego-motion of the follower-vehicle. This was already discussed in Section 2-2. For this compensation, the ego-motion parameters of the follower-vehicle are used. These parameters are follower-vehicle longitudinal speed u, its lateral velocity v and yaw rate r (also known as rate of change of heading angle $\dot{\psi}$). In practice, the lateral velocity v is not measured; instead, it can either be estimated or neglected [23]. The Standard Sensor Systems introduce noise on the longitudinal speed u and yaw rate r. Moreover, the vehicle state estimation can only provide the lateral velocity v with limited certainty. These phenomena introduce error when the ego-motion parameters are used to map the list of waypoints. The older the waypoints are, the more often they have to be remapped. Therefore, the waypoints get less accurate over time, because of noise-induced error accumulation. As a result, noise on the follower-vehicle ego-motion parameters is considered the fourth and final real-world disturbance on a generic path generation system.

2-4 Summary

Path generation is deployed to obtain a smooth and continuous path, based on given waypoints. The waypoints can originate from various sources. For instance, in automated vehicle-following, the source is a system that detects a leader-vehicle. Alternatively, the waypoints may be defined by a human operator or originate from a strategic path planning layer. Fundamentally, waypoints obtained from sensor systems are noise-corrupted. Therefore, a generic path generation system must have noise-rejecting characteristics. In this way, the path generation system is applicable irrespective of the specific Waypoint Source.

Within this research, the path generation problem is solved for the generic vehicle-following field of application. This field of application results in a general situation with a continuously changing set of waypoints and a continuously changing, vehicle-attached, local frame of reference. Consequently, the path generation solution is also applicable in situations where the set of waypoints and reference frame are stationary. Moreover, in this generic field of application, the generated path can be interpreted as a reconstruction of the leader-vehicle's factual path. As a consequence, path generation performance can be evaluated, since it is related to the accuracy of reconstruction. This approach does not impose any limitations on the system; that is, the applicability remains irrespective of the specific Waypoint Source.

In an ideal situation, the input waypoints are precise and accurate. However, in reality, they may be uncertain or corrupted with sensor noise. Next to Waypoint Noise, a path generation system is subject to other real-world disturbances, e.g., Waypoint Delay, Waypoint Offset and Motion Parameter Measurement Noise.

Chapter 3

Polynomial Path Generation

A path generation system consists of three main subsystems. The hierarchy of the subsystems is visualized in Fig. 3-1. Based on the output of the Waypoint Storage subsystem, a Path Generation method can be deployed. This chapter analyzes Path Generation by polynomial fitting.

3-1 Polynomial Fitting

The benchmark polynomial fitting path generation method is visualized in Fig. 3-2. From the Waypoint Storage subsystem, a list of (possibly noise-corrupted) waypoints is obtained. From this list, a fixed amount of N waypoints is selected, behind and in front of the follower-vehicle. Subsequently, a polynomial can be fitted to this window of waypoints.

The polynomial coefficients are computed while minimizing the mean-square error between the polynomial and the waypoints. In this way, a mathematical expression is obtained, which represents the waypoints. This mathematical expression is C^n continuous, where n is the degree of the polynomial. It can be interpreted as a reconstruction of the path driven by the leader-vehicle, which is instantaneously continuous by design. In contrast to this, obtaining a

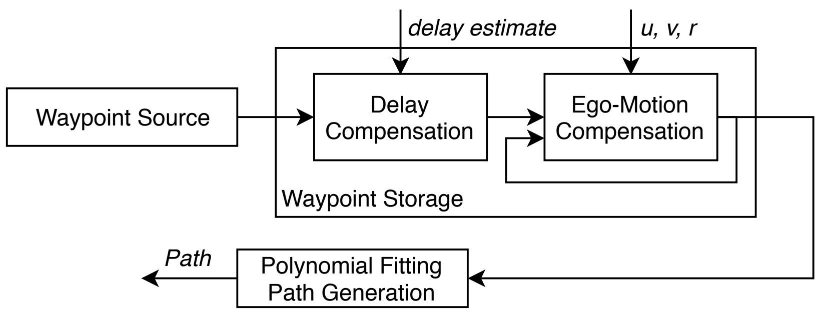


Fig. 3-1: Hierarchy of Polynomial Path Generation.

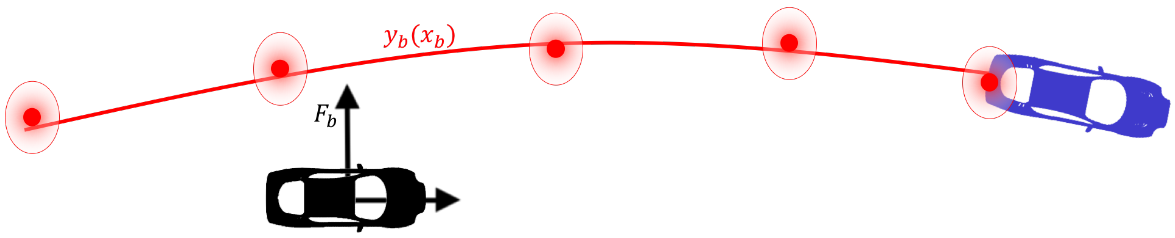


Fig. 3-2: A visualization of polynomial path generation. The leader-vehicle is blue, the follower-vehicle is black. The waypoints from the Waypoint Source are stored in a list. This list of waypoints is represented by the red dots. The uncertainty about the exact position of the waypoints is indicated by the red ellipses around the waypoints. This uncertainty may caused by measurement noise. A polynomial y_b is fitted through a fixed amount of waypoints from the list. The fitted polynomial is represented by the smooth red line and is expressed in the reference frame F_b.

path by linear interpolation of the waypoints would lead to an instantaneously discontinuous path. In addition, due to the least-squares function fitting approach, the negative effect of the Waypoint Noise disturbance is attenuated.

3-1-1 Objective

The mathematical expression of a polynomial in Cartesian coordinates x_b (longitudinal) and y_b (lateral) of order n is given by:

$$y_b(x_b) = c_n \cdot x_b^n + c_{n-1} \cdot x_b^{n-1} + \ldots c_1 \cdot x_b + c_0. \tag{3-1}$$

Polynomial fitting is defined by the task of finding coefficients $c_0 \ldots c_n$ such that the following expression is minimized:

$$S = \sum_{i=1}^{N} (y_i - y_b(x_i))^2. \tag{3-2}$$

Here, y_i is the lateral position (dependent variable) of the waypoint set, with $i = 1 \ldots N$, to which the polynomial is fitted. Furthermore, $y_b(x_i)$ is the function value of (3-1), evaluated at x_i, the corresponding longitudinal position value (independent variable). Minimizing the expression given by (3-2) results in a polynomial function that fits the waypoints in the least-squares sense.

3-1-2 Implementation

To provide appropriate feedforward in lateral vehicle control, path curvature is used [18]. Therefore, in the implementation of polynomial fitting path generation, a third order polynomial is considered. This is the simplest polynomial that provides a continuous curvature. The polynomial expression is given by:

$$y_b = c_3 \cdot x_b^3 + c_2 \cdot x_b^2 + c_1 \cdot x_b + c_0, \tag{3-3}$$

where y_b and x_b represent the Cartesian coordinates of the polynomial, expressed in an instantaneous follower-vehicle reference frame F_b. The coefficients $c_0 \ldots c_3$ are calculated based on

N noise-corrupted waypoints $\mathbf{w}_i^F$ with $i = 1 \dots N$. This results in N equations with a number of unknowns equal to the number of polynomial coefficients. The least-squares solution is obtained by the linear programming:

$$\mathbf{X} = \begin{bmatrix} (\tilde{\mathbf{w}}_x^F)^3 & (\tilde{\mathbf{w}}_x^F)^2 & (\tilde{\mathbf{w}}_x^F) & \mathbf{1} \end{bmatrix}, \tag{3-4}$$

$$\mathbf{p} = \begin{bmatrix} c_3 & c_2 & c_1 & c_0 \end{bmatrix}^T, \tag{3-5}$$

$$\mathbf{X}\,\mathbf{p} = \tilde{\mathbf{w}}_y^F, \tag{3-6}$$

where $\tilde{\mathbf{w}}_x^F$ is a column vector containing the (longitudinal) x-coordinates of all waypoints $\mathbf{w}_i^F$ with $i = 1 \dots N$. Furthermore, $\tilde{\mathbf{w}}_y^F$ is the column vector containing the (lateral) y-coordinates of all waypoints. The power operations are computed element-wise. Consequently, the polynomial coefficients, for a least-squares fit, are obtained by:

$$\mathbf{p} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\tilde{\mathbf{w}}_y^F. \tag{3-7}$$

3-2 Modified Polynomial Fitting

An advantage of polynomial fitting path generation is the intrinsic noise filtering characteristic. Fitting a third order polynomial to more than four waypoints, in the least-squares sense, leads to attenuating the negative effects of Waypoint Noise. However, this filtering behavior can also be interpreted as a disadvantage, since there is no direct, structured way of tuning the algorithm or incorporating knowledge on noise characteristics. Moreover, the resulting path is not necessarily feasible to track by a vehicle, because it was obtained without incorporating the non-holonomic vehicle constraints. The order of the polynomial and the amount of used waypoints are the only direct tools available to influence the algorithm's behavior. To obtain extra tuning parameters, weights can be assigned to future offsets from the waypoints. Consequently, by weighted least-squares polynomial fitting, the noise-filtering characteristics of the polynomial fitting method can be influenced. However, there is no structured or intuitive way of tuning these parameters for the path generation use-case.

A single third order polynomial by itself is smooth and C^3-continuous. However, a sequence of computed polynomials is not, when evaluated at the longitudinal coordinate of the local follower-vehicle reference frame F_b ($x_b = 0$). Instead, it provides a path with a discontinuous position, heading and curvature. This is visualized in Fig. 3-3. This phenomenon is confirmed and quantified by the results from simulations, presented in Section 5-3. The discontinuous character is caused by the required repetitive re-fitting of the polynomial. The reference frame, in which the polynomial must be expressed, is moving because of the ego-motion of the follower-vehicle. Moreover, the set of waypoints may be updated, for instance when a new measurement is available. The oldest waypoint is discarded and the new waypoint is added. This results in the need for repetitive re-fitting of the polynomial to the updated list of waypoints.

Theoretically, this basic polynomial fitting method can be improved by imposing constraints on the linear programming process. These constraints can be based on the characteristics of a previously computed polynomial. For instance, they can be set on the first or second derivative of a new polynomial y_b, depending on the previous polynomial y_{b-1}. This method is investigated in the next paragraphs.

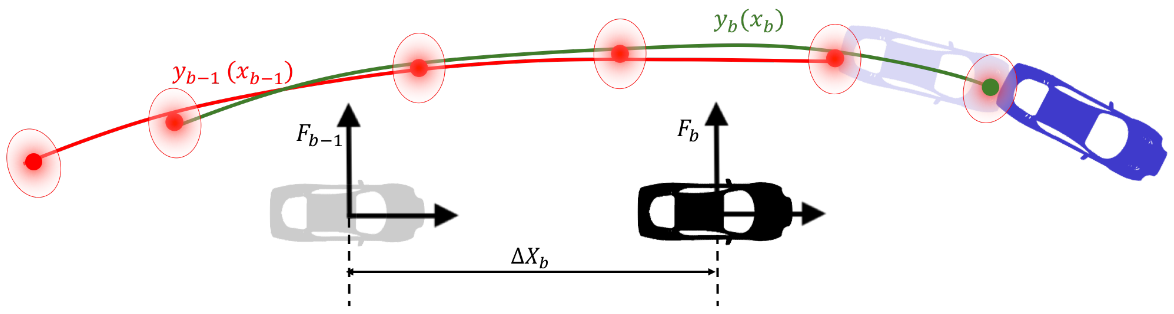


Fig. 3-3: Visualization of polynomial fitting path generation at two consecutive time instances. The noise-corrupted waypoints are represented by the red points. The red polynomial $y_{b-1}(x_{b-1})$ is computed at time step $b-1$. Consequently, the follower-vehicle travels distance ΔX_b. Meanwhile, the list of waypoints is updated: a new noise-corrupted measurement is added, which is represented by the green point. The oldest measurement is discarded. At the next time instance b, a new polynomial $y_b(x_b)$ is fitted, based on the updated list of waypoints. The visualization clearly shows that this repetitive polynomial fitting approach results in a path that is discontinuous in position, heading and curvature.

3-2-1 Heading Constraint

In real driving situations, a vehicle drives a path with a continuous heading. During polynomial path generation, the heading of the leader-vehicle is being estimated by the heading of the resulting polynomial path ψ_b. This heading is given by:

$$\psi_b = \tan^{-1}(\frac{dy_b}{dx_b}). \tag{3-8}$$

To force the resulting path to have a continuous heading angle, the transition in heading angle from polynomial y_{b-1} to polynomial y_b must be continuous. To achieve this, a constraint on the heading of the newly computed polynomial $y_b(x_b)$ can be set. A direct heading constraint in the form of (3-8) would result in a nonlinear constrained fitting process. Therefore, it is more sensible to set a constraint on $\frac{dy_b}{dx_b}$ only. For a third order polynomial, this derivative is given by:

$$\frac{dy_b}{dx_b} = 3c_3x_b^2 + 2c_2x_b + c_1. \tag{3-9}$$

A constraint on this derivative results in a linear constraint on the process of finding polynomial coefficients c_0 ... c_n. As a result, the constrained least-squares fitting is still solvable by linear programming.

It is proposed to set the heading at the origin of the new polynomial y_b equal to the origin's heading of the previously computed polynomial y_{b-1}. In this way, the heading is forced to be continuous, on the condition that the traveled origin distance from F_{b-1} to F_b is accurate. Mathematically, this is translated into the following linear equality constraint:

$$\frac{dy_{b-1}}{dx_{b-1}}|_{x_{b-1}=\Delta X_b} = \frac{dy_b}{dx_b}|_{x_b=0}, \tag{3-10}$$

where ΔX_b is the traveled origin distance of the local follower-vehicle reference frame, during the time between the two polynomial fitting time steps. This is visualized in Fig. 3-3. This constraint approximates the forcing of the derivative of the polynomial, and therefore the heading of the path, to be continuous. Note that, in general, the vehicle may also move in the lateral direction or change its orientation. Therefore ΔX_b is merely an estimate of the movement of frame F_b.

In this way, the previously computed polynomial y_{b-1} is incorporated in the computation of the new polynomial y_b. However, this results in a new polynomial with a certain heading at $x_b = 0$, which is not necessarily the real leader-vehicle heading. Although it is not an intuitive tuning parameter, ΔX_b can be tuned to mitigate this problem. Summarizing, it is hypothesized that the heading-constrained fitting method provides paths with a more continuous heading, without necessarily resulting in a superior reconstruction of the true path driven by the leader-vehicle. This hypothesis is confirmed by the results from simulations and experiments, presented in Chapters 5 and 7.

3-2-2 Curvature Constraint

In real driving situations, a vehicle drives a path with a continuous curvature. During polynomial path generation, the curvature of the path driven by the leader-vehicle is being estimated by the curvature of the resulting polynomial path κ_b. This curvature is given by the nonlinear combination of the first and second order derivative of the polynomial:

$$\kappa_b = \frac{\frac{d^2 y_b}{(dx_b)^2}}{(1 + (\frac{dy_b}{dx_b})^2)^{\frac{3}{2}}}. \tag{3-11}$$

In normal driving, $(\frac{dy_b}{dx_b})^2$ is very small compared to unity. This is confirmed by simulations in Chapter 5. Therefore, the curvature can be estimated by:

$$\kappa_b = \frac{d^2 y_b}{(dx_b)^2}. \tag{3-12}$$

For a third order polynomial, the second order derivative is given by:

$$\frac{d^2 y_b}{(dx_b)^2} = 6c_3 x_b + 2c_2. \tag{3-13}$$

As a result, a constraint on $\frac{d^2 y_b}{(dx_b)^2}$ brings about a linear constraint on the process of finding the polynomial coefficients $c_0 \ldots c_n$. With a linear constraint added, the least-squares polynomial fitting is still solvable by linear programming. It is proposed to set the second order derivative, at the origin of the new polynomial y_b, equal the origin's second order derivative of the previously computed polynomial y_{b-1}. In this way, the estimated curvature is forced to be continuous, resembling the real driving situation; that is, on the condition that the traveled origin distance from F_{b-1} to F_b is accurate. Mathematically, this is translated into the following linear equality constraint:

$$\frac{d^2 y_{b-1}}{(dx_{b-1})^2}|_{x_{b-1}=\Delta X_b} = \frac{d^2 y_b}{(dx_b)^2}|_{x_b=0}, \tag{3-14}$$

where ΔX_b is again the traveled origin distance of the local follower-vehicle reference frame, during the time between two consecutive polynomial fitting steps. This is a second way in which the previously computed polynomial y_{b-1} can be incorporated in the computation of the new polynomial y_b. Note that this framework can also be used to incorporate curvature information from inter-vehicular or roadside communication when available. However, the new polynomial, computed with this method, has a certain curvature at $x_b = 0$, which is not necessarily the real leader-vehicle curvature. Again, ΔX_b can be interpreted as the tuning parameter. Therefore, it is hypothesized that the curvature-constrained fitting method provides paths with a more continuous curvature, without necessarily resulting in a superior reconstruction of the true path driven by the leader-vehicle. This hypothesis is confirmed by simulations and experiments, presented in Chapters 5 and 7, as well.

Apart from constraining the heading and curvature of consecutive polynomials, bounds can directly be imposed on the polynomial coefficients. In a third order polynomial, c_1 is equal to the function value of the first derivative at $x_b = 0$. Therefore, a bound on c_1 represents a bound on the derivative of the polynomial at $x_b = 0$. The bound margins determine how fast the polynomial coefficients can change. This approach boils down to knowing how fast a vehicle can turn or change heading, and incorporating this information in the polynomial fitting. However, this is an indirect way of using vehicle-modeling for path generation.

3-3 Conclusions

From theoretical analysis, it is concluded that repetitive polynomial fitting methods produce discontinuous paths, which is inherent to the function fitting approach. Firstly, the discontinuous character of the paths makes them fundamentally unfeasible, since vehicles can only produce and track continuous trajectories. Moreover, a discontinuous reference path results in the need for path smoothing, when used in comfortable lateral vehicle control. This smoothing leads to inaccurate vehicle control, caused by manipulation of the original, discontinuous reference path. To mitigate these drawbacks, modified polynomial path generation methods are proposed, resulting in a total of three different methods. The first method is an unmodified version of the unconstrained benchmark polynomial path fitting method. The other two methods are based on constrained polynomial fitting. Heading constrained and curvature constrained polynomial fitting methods are presented. However, when constrained fitting methods are applied in vehicle-following, the resulting paths can obtain a heading or curvature which is not necessarily the correct, real leader-vehicle heading or curvature. Therefore, it is hypothesized that the constrained fitting methods provide more continuous paths, without necessarily resulting in superior reconstructions of the true path driven by the leader-vehicle. Additionally, polynomial fitting methods require tuning of parameters that have indirect connections with the vehicle dynamics. In other words, the tuning process attempts to model the vehicle and its trajectory characteristics with tuning parameters that are not intuitive. All in all, it is concluded that a polynomial fitting path generation approach is fundamentally sub-optimal.

Chapter 4

Model-based Path Generation

This chapter presents a new Model-Based Path Generation method. The new Model-based Path Generation method consists of four main subsystems. The hierarchy of the subsystems is visualized in Fig. 4-1. Based on the output of the Waypoint Storage subsystem, the new method can be deployed. The two subsystems representing the new method are given by the Virtual Leader Model and the Virtual Driver. A visualization of the new method is given in Fig. 4-2.

First, an introduction to Model-based Path generation is given in Section 4-1. Consequently, Section 4-2 elaborates on the Virtual Leader Model. The Virtual Leader Model is driven by the Virtual Driver subsystem. Sections 4-3 and 4-4 provide the mathematical framework for two different implementations of the Virtual Driver subsystem. Finally, Section 4-6 presents a summary of the new Model-based Path Generation method.

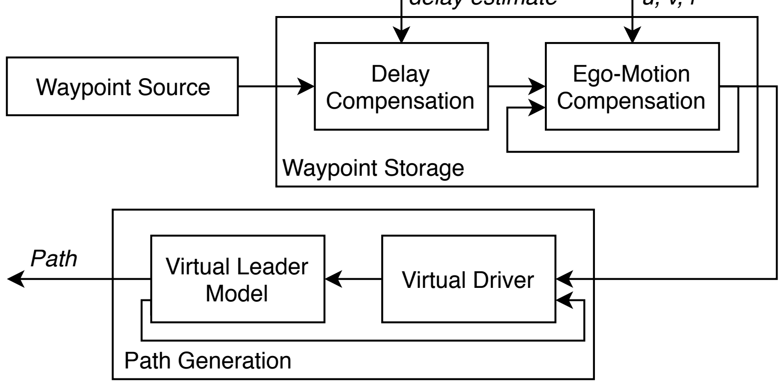


Fig. 4-1: Hierarchy of Model-based Path Generation.

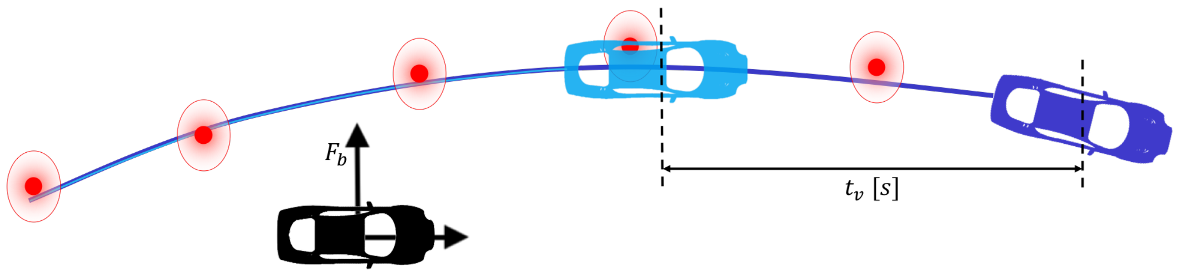


Fig. 4-2: The light-blue vehicle represents a virtual, computer-simulated, leader-vehicle. The aim of this virtual vehicle is to construct a reference path based on given waypoints. This simulated virtual leader-vehicle drives at a time headway of t_v (in seconds) with respect to the Waypoint Source, the real leader-vehicle. The time headway t_v is directly related to the amount of waypoints between the virtual vehicle and the real leader-vehicle. The smooth, light-blue dotted line represents the path driven by the virtual leader-vehicle. Note that the virtual leader-vehicle must not exactly follow the red waypoints. The waypoints are noise-corrupted and do not exactly represent the desired path of Waypoint Source. The waypoint uncertainty caused by this noise is represented by the red ellipses around the waypoints.

4-1 Introduction to Model-based Path Generation

The model-based path generation method is visualized in Fig. 4-2. The method is based on simulating a virtual leader-vehicle, on board of the host follower-vehicle. From simulating this virtual leader-vehicle, a reference path based on the waypoints can be obtained. The behavior of the virtual leader-vehicle depends on the (state transition) Virtual Leader Model and on the applied Virtual Leader Model inputs. These inputs are given by the Virtual Driver subsystem. The goal in the vehicle-following field of application is to reconstruct the behavior of real leader-vehicle as accurate as possible. As a result, naturally, the Virtual Driver needs to accurately imitate the actual real leader-vehicle's control actions. This can be achieved by controlling the Virtual Driver based on the list of waypoints, obtained from the Waypoint Storage.

The Virtual Leader Model subsystem is discussed in Section 4-2. The goal of this subsystem is to generate a robust, continuous reference path, based on given model inputs. This reference path can be extracted from the state description of the Virtual Leader Model.

The inputs to the Virtual Leader Model are given by the Virtual Driver subsystem. Its objective is to steer the virtual leader-vehicle in such a way, that it accurately tracks the waypoints from the Waypoint Source. In the ideal case, without Waypoint Noise and Center of Gravity (CoG) Offset, this corresponds to the objective of exactly driving through the obtained list of waypoints. The control objective is different for a real-world scenario, where real-world disturbances are present. In this case, the Virtual Driver should not exactly track the reference waypoints. Instead, it should realize a feasible, smooth trajectory for the virtual leader-vehicle, attenuating the effects of the disturbances. This is visualized in Fig. 4-3.

The Virtual Driver subsystem can be realized in different ways. A concept based on proportional control, using a look-ahead distance, is implemented and discussed in Section 4-3. An illustrating example of the behavior of this controller implementation is visualized in Fig. 4-4. In this example, the model of the virtual leader-vehicle is an accurate vehicle model, representing the real leader-vehicle's system dynamics. Additionally, in this example, the real

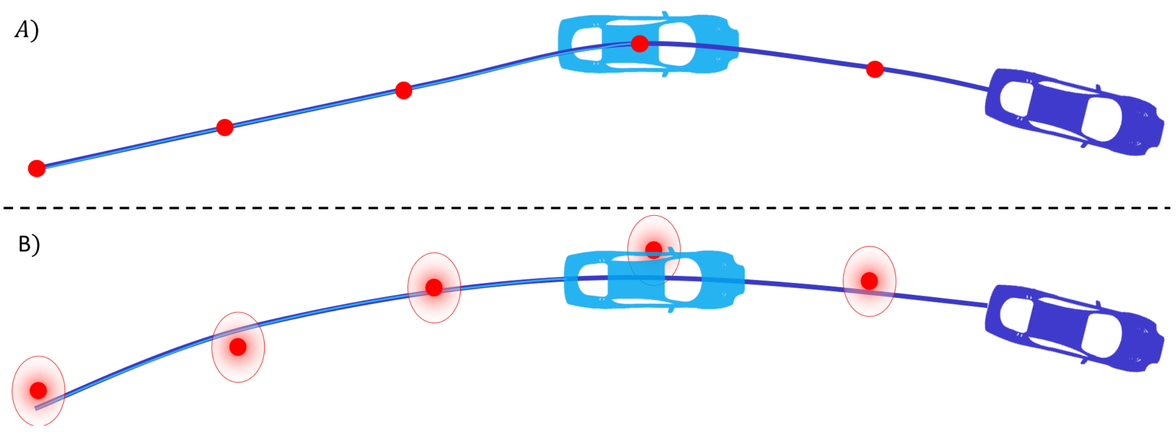


Fig. 4-3: Visualization of the different control objectives. Figure A represents the ideal vehicle-following scenario, where the waypoints are noise-free. In this scenario, the given waypoints reference should be tracked accurately. Note that Figure A also represents the control objective when the proposed Model-based Path Generation method is deployed outside the generic vehicle-following use-case, when a different Waypoint Source provides precise and accurate waypoints. Figure B represents the real-world case, where the given waypoints are noise-corrupted and should not be tracked exactly. Instead, the light-blue virtual leader-vehicle should drive a smooth trajectory, accurately imitating the dark-blue, real leader-vehicle. For both objectives, the virtual vehicle must produce a feasible path, meaning that the trajectory is in accordance with vehicle dynamics.

leader-vehicle performs a sine steering maneuver. The computed control input is shifted in time, compared to the real leader-vehicle steering input. This time shift is not because of control limitations, but because the real leader-vehicle drives at a certain time headway t_v (in seconds) with respect to the virtual leader-vehicle. This time headway t_v is required to realize a positive prediction horizon, which is further explained in Sections 4-3 and 4-4. On top of this time-shift, the control-signal is lags behind. This is because a proportional feedback framework needs to allow error, before it computes control actions. Another downside of proportional control is given by the possibility of control overshoot. Because of these weaknesses an additional proposal is done for the Virtual Driver subsystem. This additional proposal is based the framework of Model Predictive Control (MPC). The MPC methodology is further explained in the Section 4-4. The behavior of this MPC implementation is also visualized in Fig. 4-4, using the aforementioned illustrating example. It is known that the MPC strategy generally has a higher computational burden compared to a simple proportional feedback framework. This is caused by the fact that MPC is based on prediction and function optimization. Therefore, it is hypothesized that the computed control input of MPC is more discrete, compared to the control input obtained by proportional control. This is visualized by the staircase-shape of the virtual leader control input computed by MPC in Fig. 4-4. Note that the Virtual Driver framework allows incorporation of additional information, when available. For instance, from inter-vehicular communication, steering information can be used in the Virtual Driver subsystem. Alternatively, camera systems or roadside communication could provide road curvature information to provide appropriate feedforward [18].

The proposed methodology allows for explicit incorporation of knowledge about the vehicle dynamics in path generation, resulting in feasible vehicle paths. For instance, in vehicle-

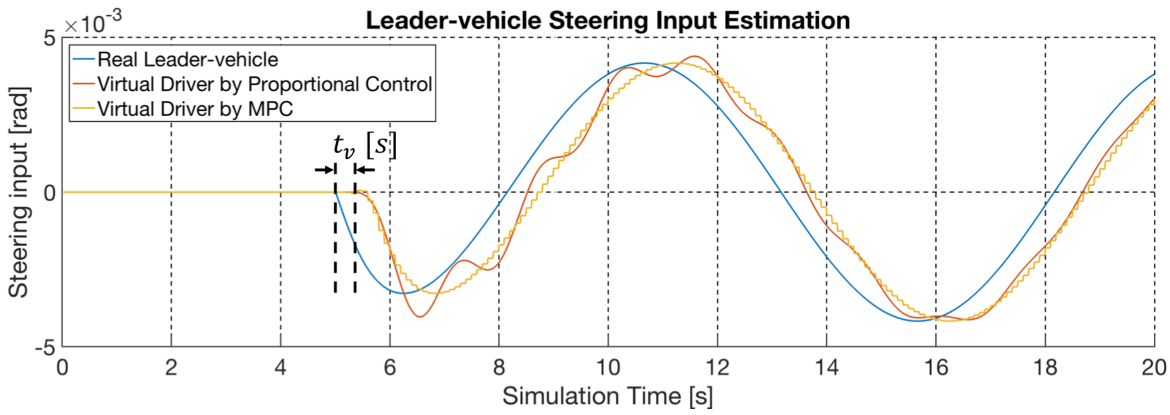


Fig. 4-4: The steering control inputs are compared. A real leader-vehicle is assumed to perform a sine steering maneuver starting after 5 seconds. Consequently, the computed control inputs for both the Virtual Driver implementations are shown. The control inputs are time-shifted versions of the real leader-vehicle steering input, since the virtual vehicle drives at a certain time headway t_v (in seconds) with respect to the real leader-vehicle.

following, the preceding object is a vehicle: the leader-vehicle. Object classification techniques can extend this knowledge by providing the type, brand or model of the vehicle. This knowledge can be translated into a description of the leader-vehicle's dynamic capabilities, which can be described mathematically in the framework of a state transition model. This approach allows for intuitive path generator tuning by parameters that have a direct connection with the vehicle dynamics. This is in contrast to the polynomial path generation method, which was discussed in Chapter 3.

Model-based Path Generation in literature

With the methodology proposed in this chapter, a new approach to path generation is conceived. The approach is different from state-of-the-art path generation or path filtering methods found in literature. In literature, a Kalman Filter (KF) or Particle Filter (PF) (or one of their variants) are often deployed for path filtering. However, for the generic field of application, these filters do not provide a solution, since the system input is not available. Instead, a virtual vehicle modeling approach is presented, where the model input is estimated by proportional feedback (Section 4-3) or MPC (Section 4-4).

Alternatively, in literature, model-based path generation is done by using motion primitives. For aerial vehicle motion planning, Bottasso et al. propose a novel method of smoothing given waypoints [2]. The smooth path is obtained by using a transcribed version of the vehicle dynamics expressed in terms of motion primitives. With this approach, the path planning issue is solved by examining a limited set of motion templates that can be adopted in sequence, such that the waypoints are reached [3]. To select a specific sequence of motion templates, reaching the desired waypoints, a node search algorithm, e.g., Rapidly-exploring Random Tree (RRT) [13], can be used. The method of computing the motion templates resembles MPC, in the sense that it uses a system transition model, and computes a series of control inputs for a certain time horizon, such that an objective function is minimized and certain constraints are met. However, in the motion primitive method the predictions are computed offline and the

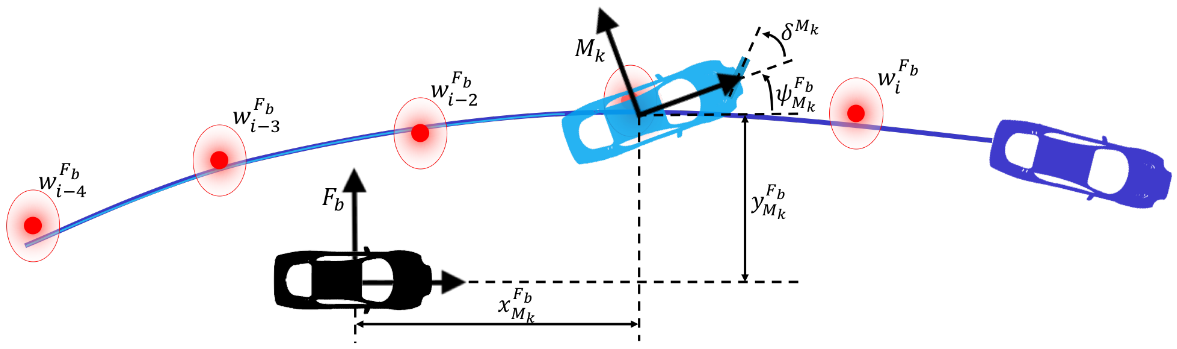


Fig. 4-5: Virtual leader-vehicle model frame and state definitions

control inputs are not executed. Instead, the optimized feasible system state series is stored in a library as motion templates. The resulting trajectories are compatible by design with the vehicle model embedded in the motion primitives. The motion primitives consist of trims and maneuvers. Trims are trajectories that operate at steady state model equilibria. Maneuvers are the transitions between two trim trajectories [6]. The number of trims is a design choice. Its selection affects the size of the motion primitive library. A large motion primitive library can generate more exquisite trajectories, but the search process for an optimal trajectory is then likely to take more time [3]. For the vehicle-following field of application, the motion primitive method would lead to the necessity of having different libraries of motion primitives for different types of vehicles. Consequently, matching a library to the specific leader-vehicle at hand could be done by object classification techniques. However, this approach is sub-optimal, since it would require a large amount of different pre-computed libraries containing motion primitives for different types of vehicles. Clearly, this limits the usage of a path generation system based on motion primitives. Moreover, a motion primitive-based method would limit the generality of path generation, because the motion primitive framework lacks tuning capabilities: the method does not deal with the existence of uncertainty on the exact position of the waypoints, in contrast to the new methodology proposed in this chapter.

4-2 Virtual Leader Model

This section elaborates on the mathematical framework for the Virtual Leader Model subsystem. The subsystem hierarchy was visualized in Fig. 4-1.

4-2-1 Design

The process model of the virtual leader-vehicle is used to evolve and predict the state behavior of the virtual leader vehicle. The process model will be simulated on board of the follower-vehicle. From the simulated virtual leader-vehicle, the state trajectory can be extracted. With this state trajectory, a feasible reference path is obtained, which meets the non-holonomic vehicle constraints. For the vehicle-following use-case, it is desirable that this state trajectory is an accurate imitation of the real leader-vehicle's trajectory.

Let F_b denote the local reference frame attached to the follower-vehicle, at time instance b. This is visualized in Fig. 4-5. Furthermore, let M_k denote the reference frame attached to the virtual leader-vehicle, at time step k. The position of M_k is given by $x^{F_b}_{M_k}$ and $y^{F_b}_{M_k}$, which represent the x- and y-coordinate respectively of the origin of the virtual leader-vehicle-attached reference frame M_k. Moreover, let $\psi^{F_b}_{M_k}$ and δ^{M_k} denote the heading and steering angle respectively of the virtual leader-vehicle. The states $x^{F_b}_{M_k}$, $y^{F_b}_{M_k}$, $\psi^{F_b}_{M_k}$ and δ^{M_k} together define the state of the virtual leader-vehicle at time step k. They form the model state vector $\mathbf{s}^{F_b}_{M_k}$, given by:

$$\mathbf{s}^{F_b}_{M_k} = \begin{bmatrix} x^{F_b}_{M_k} \; y^{F_b}_{M_k} \; \psi^{F_b}_{M_k} \; \delta^{M_k} \end{bmatrix}^T . \tag{4-1}$$

Assuming zero lateral velocity of the virtual vehicle is justified by the fact that the path generation method will be used for the regular driving regime. Consequently, with a forward velocity denoted by u and using a kinematic model, the time derivative of the virtual leader-vehicle position is given by:

$$\begin{bmatrix} \dot{x}^{F_b}_{M_k} \\ \dot{y}^{F_b}_{M_k} \end{bmatrix} = \begin{bmatrix} u \cdot \cos(\psi^{F_b}_{M_k}) \\ u \cdot \sin(\psi^{F_b}_{M_k}) \end{bmatrix} . \tag{4-2}$$

To incorporate vehicle dynamics, the kinematic model is extended with yaw and steering dynamics. The yaw dynamics are modeled by the steady-state equation of the well-known single-track bicycle model [18], with the assumptions of constant forward velocity, no longitudinal or lateral load transfer, linear range tires, no roll, pitch or vertical motion, no suspension and no compliance effects [21]. These assumptions are justified, since the path generation method will be used for the regular driving regime. The time derivative of heading angle $\psi^{F_b}_{M_k}$ is then given by the yaw velocity response:

$$\dot{\psi}^{F_b}_{M_k} = \frac{u \cdot \delta^{M_k}}{L + k_{us} \cdot u^2}, \tag{4-3}$$

where u is the forward velocity, L the wheelbase and k_{us} the understeer gradient. Furthermore, δ^{M_k} is the steering angle state, also visualized in Fig. 4-5. The steering dynamics are modeled by the following first order system:

$$\dot{\delta}^{M_k} = \frac{\delta_d - \delta^{M_k}}{\tau}, \tag{4-4}$$

where δ_d is the driver input steering angle and τ the time constant characterizing the steering state response.

Summarizing, the Virtual Leader Model dynamics are given by:

$$\dot{\mathbf{s}}^{F_b}_{M_k} = f(\mathbf{s}^{F_b}_{M_k}, u, \delta_d) = \begin{bmatrix} u \cdot \cos(\psi^{F_b}_{M_k}) \\ u \cdot \sin(\psi^{F_b}_{M_k}) \\ \frac{u \cdot \delta^{M_k}}{L + k_{us} \cdot u^2} \\ (\delta_d - \delta^{M_k})/\tau \end{bmatrix} . \tag{4-5}$$

4-2-2 Initialization

To initialize the simulation of the virtual leader-vehicle, initial states are required. These states are based on using the final states from the previous optimization time step. Let the

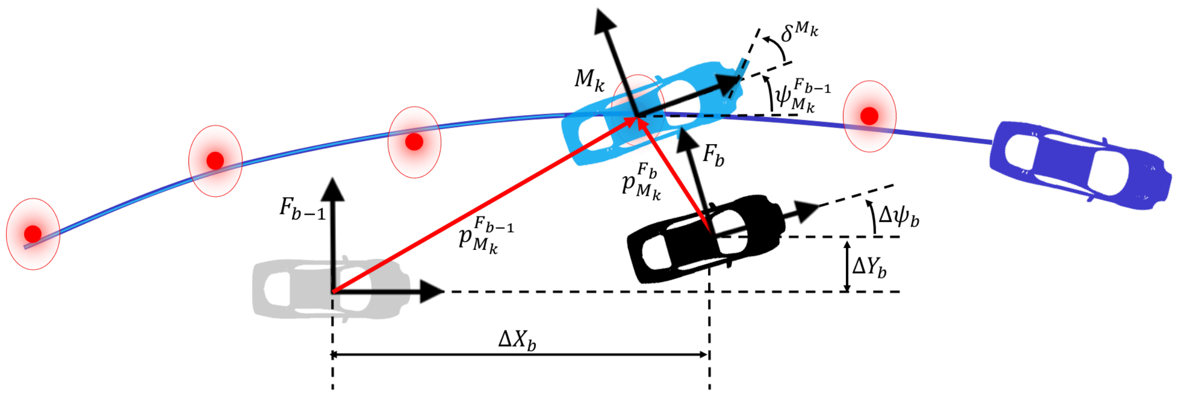


Fig. 4-6: Visualization of follower-vehicle ego-motion compensation for virtual leader-vehicle modeling.

position of frame M_k, which is incorporated in the state vector $\mathbf{s}^{F_b}_{M_k}$, also be denoted by the following vector:

$$\mathbf{p}^{F_b}_{M_k} = \begin{bmatrix} x^{F_b}_{M_k} & y^{F_b}_{M_k} \end{bmatrix}^T. \tag{4-6}$$

Due to follower-vehicle ego-motion, this state vector must continuously be transformed from reference frame F_{b-1} to F_b. Therefore, for every initialization in a new frame F_b, the initial state $\mathbf{s}^{F_b}_{M_{k=0}}$ of the virtual leader-vehicle process model is given by its final state $\mathbf{s}^{F_{b-1}}_{M_{k=k_{final}}}$. This mapping is visualized in Fig. 4-6 and performed by:

$$\mathbf{s}^{F_b}_{M_{k=0}} = g(\mathbf{s}^{F_{b-1}}_{M_{k=k_{final}}}) = \begin{bmatrix} \mathbf{R}^{F_b}_{F_{b-1}} \cdot (\mathbf{p}^{F_{b-1}}_{M_k} - \Delta\mathbf{O}^{F_{b-1}}_{F_b}) \\ \psi^{F_{b-1}}_{M_k} - \Delta\psi_b \\ \delta^{M_k} \end{bmatrix}, \tag{4-7}$$

where $\Delta\mathbf{O}^{F_{b-1}}_{F_b}$ is the origin of the local follower-vehicle reference frame F_b, with respect to F_{b-1}. This is given by the horizontal and vertical displacement of the follower-vehicle: ΔX_b and ΔY_b respectively. Moreover, $\Delta\psi_b$ is the change in heading angle of follower-vehicle frame F_{b-1} to F_b. These variables, representing the ego-motion of the follower-vehicle, are visualized in Fig. 4-6. Note that the steering state δ^{M_k} does not require follower-vehicle ego-motion compensation, since it is expressed in the local virtual vehicle frame M_k. Furthermore, $\mathbf{R}^{F_b}_{F_{b-1}}$ is the rotation matrix, mapping vectors from F_{b-1} to F_b, given by:

$$\mathbf{R}^{F_b}_{F_{b-1}} = \begin{bmatrix} \cos\Delta\psi_b & \sin\Delta\psi_b \\ -\sin\Delta\psi_b & \cos\Delta\psi_b \end{bmatrix}. \tag{4-8}$$

For the first initialization of the virtual leader-vehicle simulation in frame F_b, no previous virtual leader-vehicle state is available. Therefore the first waypoint from the Waypoint Storage $\tilde{\mathbf{w}}^{F_b}$ is assumed as true initial position of M_k. Moreover, if the other state variables can be assumed to be zero, then:

$$\mathbf{s}^{F_b}_{M_{k=0}} = \begin{bmatrix} \mathbf{w}^{F_b}_1 & 0 & 0 \end{bmatrix}^T. \tag{4-9}$$

This finalizes the definition of the mathematical framework for the Virtual Leader Model subsystem. For the proposed process model, control inputs must be calculated, such that the model states approach or track desired reference values. For path generation, these reference values are given by the waypoint matrix $\tilde{\mathbf{w}}^{F_b}$ from the Waypoint Storage subsystem. Note that the proposed Model-based Path Generation method can also be deployed outside the vehicle-following use-case, e.g., when the list of waypoints is user-defined or coming from a strategic path planning layer. For any case, the virtual vehicle produces a feasible reference path, meaning that it is in accordance with vehicle dynamics.

4-3 Proportional Feedback Controller

This section elaborates on an implementation of the Virtual Driver subsystem. The subsystem hierarchy was visualized in Fig. 4-1. The implementation proposed in this section is based on proportional feedback. The control objective was introduced in Section 4-1 and visualized for different use-cases in Fig. 4-3. The Virtual Driver subsystem computes steering inputs for the Virtual Leader Model, based on given waypoints. First, the set of waypoints needs to be transformed. This is explained in section 4-3-1. Next, the transformed set of waypoints can be used to compute the steering input for the Virtual Leader Model. For this, the proportional control law is given in Section 4-3-2.

4-3-1 Waypoint Transformation

Given is the matrix of $n+1$ waypoints in the coordinate frame of the virtual leader-vehicle, denoted by $\tilde{\mathbf{w}}^{M_k}$:

$$\tilde{\mathbf{w}}^{M_k} = \begin{bmatrix} w_{i,x} & w_{i-1,x} & \dots & w_{i-n,x} \\ w_{i,y} & w_{i-1,y} & \dots & w_{i-n,y} \\ 1 & 1 & \dots & 1 \end{bmatrix}, \tag{4-10}$$

where $w_{i,x}$ and $w_{i,y}$ represent the x- and y-coordinate of waypoint i respectively. The individual waypoints in (4-10) are visualized in Fig. 4-7. The control action will be based on the mismatch between the waypoints at a look-ahead distance, and the current state $\mathbf{s}_{M_k}^{F_b}$ of the virtual leader-vehicle [18]. For this purpose, the control variable y_m is introduced. y_m is defined as the lateral distance error between the current position of the virtual vehicle and the lateral position of $\mathbf{w}_i^{M_k}$ at a look-ahead distance x_{la}. This is also visualized in Fig. 4-7. If no waypoint is available at x_{la}, the nearest waypoints are interpolated to obtain an expression for y_m.

The matrix of waypoints $\tilde{\mathbf{w}}^{M_k}$ is obtained by continuously remapping of $\tilde{\mathbf{w}}^{F_b}$ to the virtual vehicle frame M_k. This mapping is necessary because the Waypoint Storage subsystem provides the waypoints in the follower-vehicle reference frame F_b, rather than in M_k. Note that this mapping is time variant, because of the virtual vehicle's ego-motion during the computation of its states. The mapping is given by:

$$\tilde{\mathbf{w}}^{M_k} = \mathbf{T}_{F_b}^{M_k} \tilde{\mathbf{w}}^{F_b}, \tag{4-11}$$

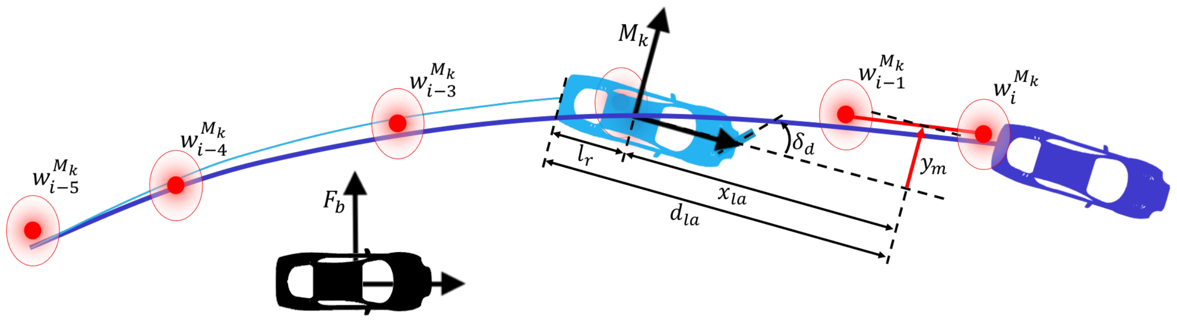


Fig. 4-7: Visualization of Virtual Driver Proportional Control variable y_m.

where $\mathbf{T}_{F_b}^{M_k}$ represents the homogeneous transformation matrix, mapping vectors from follower-vehicle frame F_b to virtual leader-vehicle frame M_k. This matrix is given by:

$$\mathbf{T}_{F_b}^{M_k} = \begin{bmatrix} \mathbf{R}_{F_b}^{M_k} & -\mathbf{R}_{F_b}^{M_k} \cdot \mathbf{p}_{M_k}^{F_b} \\ 0_{1\times 2} & 1 \end{bmatrix}, \tag{4-12}$$

where the relative virtual leader-vehicle position $\mathbf{p}_{M_k}^{F_b}$ is given by (4-6). Furthermore, the rotation matrix $\mathbf{R}_{F_b}^{M_k}$ is given by:

$$\mathbf{R}_{F_b}^{M_k} = \begin{bmatrix} \cos \psi_{M_k}^{F_b} & \sin \psi_{M_k}^{F_b} \\ -\sin \psi_{M_k}^{F_b} & \cos \psi_{M_k}^{F_b} \end{bmatrix}. \tag{4-13}$$

The required virtual leader-vehicle model states $x_{M_k}^{F_b}$, $y_{M_k}^{F_b}$ and $\psi_{M_k}^{F_b}$ were previously defined and visualized in Fig. 4-5.

4-3-2 Proportional Control Law

The control law is given by the following equation:

$$\delta_d = K_p \cdot y_m, \tag{4-14}$$

where δ_d represents the control steering input of the Virtual Driver. Moreover, K_p is the proportional gain, which is determined by realizing a certain driving radius that reduces the lateral distance error to zero at the virtual look-ahead distance x_{la}. This control law allows computation of steering actions based on the given set of transformed waypoints. Furthermore, in analogy with human driving, the look-ahead distance should increase with velocity [18]. For this reason, a look-ahead time t_{la} is introduced. Consequently, the proportional gain K_p is given by:

$$K_p = 2 \cdot \frac{L + k_{us} \cdot u^2}{d_{la}^2}, \tag{4-15}$$

where d_{la} is visualized in Fig. 4-7 and is computed by (4-16). Furthermore, u is the assumed forward velocity, L the wheelbase and k_{us} the understeer gradient of the Virtual Leader Model.

$$d_{la} = l_r + u \cdot t_{la} \tag{4-16}$$

This finalizes the first proposal for the implementation of the Virtual Driver subsystem. The behavior of this controller implementation was already discussed in Section 4-1. Because of the weaknesses inherent to proportional control, an additional proposal is done for the Virtual Driver subsystem implementation. This additional proposal is based on the MPC framework. This approach is further explained in the Section 4-4.

4-4 Model Predictive Controller

This section elaborates on a second implementation of the Virtual Driver subsystem. The implementation proposed in this section is based on the MPC framework. Section 4-3 described a control system that generates necessary inputs for the Virtual Leader Model: steering the vehicle in such a way that the resulting position states approach the given waypoints. This approach is not optimal because the steering input is only computed once an error already occurred. This is a fundamental characteristic of the proportional feedback control strategy. In practice, a human driver does not let error occur first, only to compensate proportionally to this error afterwards. Instead, the driver anticipates on future errors and undertakes action to avoid these future errors. This approach is very well represented by the MPC methodology. The goal is to find the optimal sequence of control actions for the Virtual Leader Model, such that it minimizes current and future errors $y_e^{M_k}$. These errors are defined as the offset between the predicted, future virtual vehicle states and given waypoints. This is visualized in Fig. 4-8. The errors $y_e^{M_k}$ are used in the optimization of an mathematical objective function J, leading to a sequence of optimal control actions. This will be explained in Section 4-4-1.

MPC is a methodology of controller design [22]. For path generation, the resulting Virtual Driver computes model inputs for the virtual leader-vehicle. The MPC framework is selected, because it can handle constraints on control actions and system states in a systematic way during design and implementation [22]. Moreover, MPC allows to compute optimal steering control actions based on prediction, analogue to how a human drives a vehicle. At a certain time step, a forward predicting optimization is run, minimizing an objective function J. The function J is minimized by computing a sequence of future control actions $\tilde{\delta}_d$. MPC is based on a receding horizon principle. This principle results in only the first control action being applied, instead of the complete sequence of optimized control actions $\tilde{\delta}_k^*$ at time step k. Consequently, at the next time step, the optimization is run again. A new sequence of optimized control inputs $\tilde{\delta}_{k+1}^*$ is calculated, minimizing the objective function J. Again, only the first input of the sequence is applied. This process is visualized in Fig. 4-8. The amount of control inputs $\delta_d(k)$, in the computed sequence $\tilde{\delta}_k^*$, depends on the control horizon N_c. This is a tuning parameter that is further discussed in Section 4-5. After the control horizon, for $k > N_c$, the control signal $\delta_d(k)$, in the sequence $\tilde{\delta}_k^*$, is taken to be constant [7].

4-4-1 Objective Function

The goal is to make the virtual leader-vehicle, with associated frame M_k, track the waypoints. The waypoints are used as the reference signal. The steering input to the model is used as a control signal to evolve the states $\mathbf{s}_{M_k}^{F_b}$ in such a way that M_k tracks the reference signal.

Define the sampling time of the integration of the Virtual Leader Model equations of motion to be T_i. Let T_m denote the sampling time of Waypoint Source and the optimization of MPC.

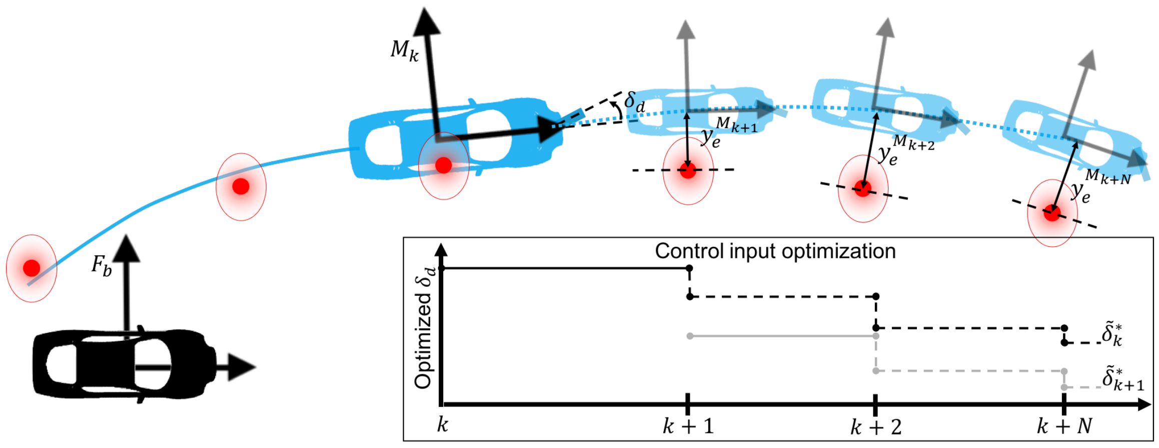


Fig. 4-8: A visualization of MPC for virtual leader-vehicle modeling. The future Virtual Leader Model state predictions are shown in transparent light-blue. MPC is based on minimizing future position state offsets from the waypoints. These offsets are denoted by $y_e^{M_{k+1}}$ ... $y_e^{M_{k+N}}$, where N is the prediction horizon. This minimization is realized by computing a sequence of optimal control inputs $\tilde{\delta}_k^*$. Only the first control input of this sequence is applied, visualized by the solid line in the control input optimization graph. Consequently, at the next time step, the optimization is run again and a new sequence of optimized control inputs $\tilde{\delta}_{k+1}^*$ is calculated.

The sampling time ratio is defined by:

$$q = \frac{T_m}{T_i}. \tag{4-17}$$

Furthermore, let $\mathbf{w}_z^{F_b}$ with $z = 1 \dots N$ denote a sequence of the first N waypoints from the Waypoint Storage matrix $\tilde{\mathbf{w}}^{F_b}$. The MPC optimization problem is given by the minimization of objective function J. This objective function consists of the squared sum of the current and predicted lateral errors $y_e^{M_k}$, given by:

$$J(\mathbf{p}_{M_k}^{F_b}, \mathbf{w}_z^{F_b}) = \sum_{z=N_m}^{N} (\mathbf{p}_{M_{q\cdot z+1}}^{F_b} - \mathbf{w}_z^{F_b})^T \cdot (\mathbf{p}_{M_{q\cdot z+1}}^{F_b} - \mathbf{w}_z^{F_b}), \tag{4-18}$$

where the subscript operation with sampling time ratio q is necessary to synchronize the Virtual Leader Model states and the waypoints. Note that Fig. 4-8 only shows the simplest scenario, where the Virtual Leader Model states and waypoints are already synchronized. Furthermore, N_m is the minimum cost horizon tuning parameter, which is further explained in Section 4-5. Now let the sequence of model inputs at time step k be given by:

$$\tilde{\delta}_k = \begin{bmatrix} \delta_d(k) & \delta_d(k+1) & \dots & \delta_d(k+N_c) \end{bmatrix}^T. \tag{4-19}$$

The optimal control sequence $\tilde{\delta}_k^*$, is then given by:

$$\tilde{\delta}_k^* = \arg\min_{\tilde{\delta}_k} J(\mathbf{p}_{M_k}^{F_b}, \mathbf{w}_z^{F_b}) \tag{4-20}$$

subject to the Virtual Leader Model equations (4-5) and (4-7). For $N_c < k < N$ the control input is kept constant with a value equal to $\delta_{k=N_c}^*$. This is related to smoothing the control signal and improving the robustness of the algorithm, for which N_c is the tuning parameter. The resulting optimized control sequence $\tilde{\delta}_k^*$ forces the virtual leader-vehicle to track the reference signal, which was given by the waypoints.

To set operating limits on the range of available control inputs, inequality constraints are used. The optimization problem of (4-20) will be solved subject to these constraints. The constraints model the physical phenomenon of actuator saturation, e.g., minimum and maximum allowable steering angle. The following constraints can be considered:

$$\begin{bmatrix} \delta_d(k) \\ -\delta_d(k) \end{bmatrix} \leq \begin{bmatrix} \delta_{max} \\ -\delta_{min} \end{bmatrix} \forall k, \tag{4-21}$$

where δ_{max} and δ_{min} represent the minimum and maximum achievable steering angles respectively. This mathematical framework of optimization subject to inequality constraints can also be used to limit the rate of change of the steering angle input, when its limit is known:

$$\begin{bmatrix} \delta_d(k+1) - \delta_d(k) \\ -(\delta_d(k+1) - \delta_d(k)) \end{bmatrix} \leq \begin{bmatrix} \dot{\delta}_{max} T_m \\ -\dot{\delta}_{min} T_m \end{bmatrix} \forall k, \tag{4-22}$$

where $\dot{\delta}_{max}$ and $\dot{\delta}_{min}$ represent the minimum and maximum achievable rate of change of steering angle respectively and T_m the sampling time of the optimization.

Note that for the non-ideal case, with Waypoint Noise, the control signal should be conservative, realizing imperfect reference tracking. For this non-ideal case, the imperfect reference tracking is required because it results in filtering the noise-corrupted waypoints with the Virtual Leader Model, resulting in a feasible vehicle path. This different control objective was explained in Section 4-1 and visualized in Fig. 4-3. The different control objectives can be handled by tuning the MPC algorithm, which is further explained in Section 4-5. Alternatively, a variation to the proposed objective function can be designed, for instance, by incorporating knowledge on the uncertainty of the waypoints. In this way, the control objective of not exactly tracking the noise-corrupted waypoints can be incorporated in the design phase, rather than obtaining the desired noise-rejecting characteristics by tuning. This alternative objective function design left for future research.

4-4-2 Stability

The drawback of MPC is that although, in practice, stability and robustness are easily obtained by accurate tuning, theoretical analysis of stability and robustness properties are difficult to derive [22]. Currently, the development of a general stability and robustness theory for predictive control is an important research topic. For the path generation functionality, the stability of the model-based controller can be forced by explicitly defining an end-point constraint, in the form of:

$$\mathbf{s}_{M_{k=N}}^{F_b} = \mathbf{s}_{ss}^{F_b}. \tag{4-23}$$

When (4-20) is solved subject to this constraint, the state at the prediction horizon N is forced to a steady-state value $\mathbf{s}_{ss}$. Defining and imposing such an end-point constraint would be in favor of stability. However, it may also lead to infeasibility of the optimization problem [22]. For path generation, infeasibility of the optimization is more likely to occur when the Virtual Leader Model headway time t_v decreases. The reason for this is that the maximum possible prediction horizon N decreases with decreasing virtual vehicle time headway t_v, since less waypoints will be available in front of the virtual vehicle (see Fig. 4-2). Another downside of the end-point constraint is that the tuning capability associated with the prediction horizon N is lost. This is because the system will be forced to steady state at the end of the control horizon N_c. Alternatively, a soft end-point constraint can be considered. A soft end-point constraint can be realized by adding it to the objective function as terminal constraint set [22]. This alternative objective function design left for future research, since it is expected that stability (with bounded generated paths) is easily obtained by tuning. This expectation is confirmed by numerical simulations (Chapter 5).

4-5 Tuning Parameters

The values for the Virtual Leader Model parameters u, L, k_{us} and τ determine the dynamic capability of the virtual vehicle. As a result, the Model-based Path Generation method allows for intuitive path generation tuning by parameters that have a direct connection with vehicle dynamics. Clearly, for the vehicle-follow use-case, the parameter values should represent the real leader-vehicle as accurate as possible. For instance, forward velocity u can be set based on leader-vehicle speed measurements from radar systems.

The Proportional Control implementation of the Virtual Driver results in the availability of look-ahead time t_{la} as extra tuning parameter. This parameter defines the look-ahead distance of the controller, influencing the proportional feedback gain.

The proposed MPC-based implementation results in the availability of the following three extra tuning parameters: N_m, N, and N_c, representing the minimum-cost, prediction and control horizon respectively. Tuning these parameters is necessary to obtain accurate and robust tracking of the given waypoints.

The setting of minimum cost horizon N_m determines when the errors $y_e^{M_k}$ start to influence the control input optimization. The optimization is only being influenced by occurring position errors $y_e^{M_k}$ for $k > N_m$. Clearly, for vehicle-following, it is desirable to set N_m equal to one, resulting in all position errors $y_e^{M_k}$ being incorporated in the optimization. For other use-cases, where only the end-position of the Virtual Leader Model matters, a different choice of N_m is necessary.

Prediction horizon N determines the number of prediction samples being used in the optimization of control input sequence $\tilde{\delta}_k^*$. In other words: N sets how far the algorithm is looking ahead. As a result, N determines based on how much information the sequence $\tilde{\delta}_k^*$ is computed. In the optimization, a compromise is found between all predicted errors $y_e^{M_k}$, such that their squared sum is minimized. This means that the larger N is, the more errors $y_e^{M_k}$ will be involved in the found compromise. As a result, N also has an effect on the Waypoint Noise rejection characteristics.

Control horizon N_c determines how many different control inputs are calculated in the sequence of optimized control inputs $\tilde{\delta}_k^*$. For $N_c > k > N$ the control input is kept constant with a value equal to $\delta_{k=N_c}^*$. This has a direct influence on the control nervousness and robustness. Moreover, setting N_c smaller than N reduces the dimension of the optimization search-region. As a result, N_c also has a large influence on the computational demand of the optimization.

Moreover, with the MPC Virtual Driver, the path generation behavior can be influenced by the operating limits on the range of available control inputs. The values set for δ_{max}, δ_{min}, $\dot{\delta}_{max}$ and $\dot{\delta}_{min}$ determine the allowed steering aggressiveness of the Virtual Driver. Consequently, they also determine the turbulence of the generated reference path, i.e. the trajectory of the virtual vehicle.

This finalizes the analysis of the new Model-based Path Generator. The reference path is obtained by extracting it from the Virtual Leader Model state history. Note that for this extraction, the state history must also continuously be compensated for ego-motion of the follower-vehicle. This compensation was given by (4-7). By design, the Model-based Path Generation methodology results in continuous paths that are compatible with the vehicle dynamics. Moreover, the resulting path can be tuned by intuitive tuning parameters. The complete mathematical framework is implemented in the custom-built simulator. This simulator will be used to emulate the path generation functionality of different path generators. With simulations, the performance, robustness, and continuity of the path generators can be evaluated. The design and the results of this evaluation are given in Chapters 5 and 6.

4-6 Summary

As alternative to polynomial fitting, a new Model-based Path Generation method is presented. The new method consists of modeling a virtual leader-vehicle: the Virtual Leader Model. The steering inputs for this Virtual Leader Model are given by a control system: the Virtual Driver. Two different control implementations of the Virtual Driver are presented. The first implementation is given by Proportional Control, using a virtual look-ahead distance. Based on given waypoints at this look-ahead distance, the control actions are calculated. The second Virtual Driver implementation is given by MPC. Using model predictions, an optimal control input sequence is computed, based on real-time optimization of an objective function. Consequently, the reference path is obtained by extracting it from the Virtual Leader Model state history. By design, the Model-based Path Generation methodology results in continuous paths that are compatible with the vehicle dynamics. Moreover, the resulting path can be tuned by intuitive tuning parameters.

Chapter 5

Simulations

In preceding chapters, three different polynomial fitting and two model-based path generation methods are presented. To validate and assess the performance of the five different methods, performance criteria must be defined. Section 5-1 presents the definitions of these performance criteria.

The evaluation of the performance criteria is based on numerical simulations and experiments. The numerical simulations are carried out using a custom-built simulator. The mathematical models defined in preceding chapters form the basis of this custom-built simulator in Matlab & Simulink. The simulator is further explained in Section 5-2.

The results from emulating different scenarios are given in Section 5-3. Finally, in Section 5-4, conclusions are drawn based on the obtained results.

5-1 Performance Criteria

To validate and assess the performance of the five different path generating methods, the resulting paths must be evaluated. The generic vehicle-following field of application allows for this evaluation, since the generated path can be interpreted as a reconstruction of the leader-vehicle's factual path. Consequently, path generation performance is related to the accuracy of reconstruction. An error between the path driven by the real leader-vehicle and the path resulting from the path generator, will be referred to as a reconstruction error.

In lateral vehicle control, path distance, heading, and curvature are used. The path curvature is required to provide appropriate feedforward [18]. Therefore, as performance criteria, three different types of reconstruction errors are defined. The reconstruction errors will be denoted by y_e, ψ_e and κ_e. These errors represent the lateral distance, heading and curvature error respectively, evaluated at the origin of the follower-vehicle reference frame F_b. For clarity, the lateral distance error at time step b (y_e^b) is visualized in Fig. 5-1. Let y_{gt} denote the relative lateral distance of the ground truth, real leader-vehicle path. Furthermore, let ψ_{gt} be the

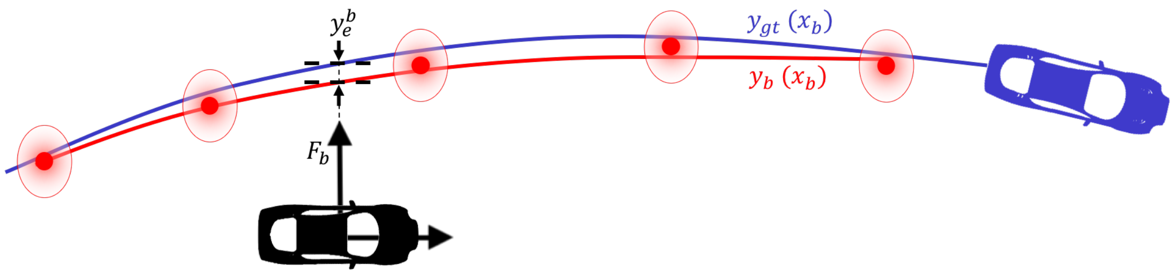


Fig. 5-1: A visualization of the reconstruction error y_e^b, which is given by the lateral distance between the real leader-vehicle path y_{gt} and the generated path y_b, evaluated at $x_b = 0$. The generated path y_b is the result of either Polynomial Fitting or Model-based Path Generation.

ground truth heading angle of this path. Moreover, the ground truth curvature is obtained by:

$$\kappa_{gt} = \frac{r_{gt}}{u_{gt}}, \tag{5-1}$$

where r_{gt} is the yaw rate of the real simulated leader-vehicle and u_{gt} its longitudinal velocity. This is an estimation of the curvature that neglects lateral acceleration, which is justified by the fact that a normal driving scenario is considered. The ground truth variables y_{gt}, ψ_{gt}, and κ_{gt} are all obtained by extracting the real leader-vehicle states from the Waypoint Source simulation, which is further explained in Section 5-2.

5-1-1 Error Calculation for Polynomial Fitting

Polynomial path generation results in obtaining the polynomial coefficients $c_0 \dots c_n$, expressed in the follower-vehicle coordinate frame F_b. The third order polynomial path is given by:

$$y_b(x_b) = c_3 \cdot x_b^3 + c_2 \cdot x_b^2 + c_1 \cdot x_b + c_0, \tag{5-2}$$

where the polynomial coefficients c_0 ... c_n vary as reference frame F_b is in motion and the polynomial is repetitively re-fitted.

Calculation of y_e

Let y_e at time instance b be denoted by y_e^b. For a third order polynomial path, y_e^b is calculated by evaluating the polynomial coefficients. The error is given by the difference between the polynomial function value at $x_b = 0$ and the lateral distance of the ground truth leader-vehicle path. This difference is given by:

$$y_e^b = y_{gt}|_{x_b=0} - y_b|_{x_b=0}. \tag{5-3}$$

Calculation of ψ_e

Similarly, let ψ_e^b denote the heading error at time instance b. The error is given by the difference between the heading of the polynomial path and the heading of the ground-truth path, both evaluated at $x_b = 0$. The polynomial path heading is given by:

$$\psi_b|_{x_b=0} = \tan^{-1}(\frac{dy_b}{dx_b}|_{x_b=0}), \tag{5-4}$$

where

$$\frac{dy_b}{dx_b} = 3c_3x_b^2 + 2c_2x_b + c_1. \tag{5-5}$$

Consequently, the heading reconstruction error is given by:

$$\psi_e^b = \psi_{gt}|_{x_b=0} - \psi_b|_{x_b=0}. \tag{5-6}$$

Calculation of κ_e

Finally, let κ_e^b denote the curvature error at time instance b. The polynomial path curvature at $x_b = 0$ is computed by:

$$\kappa_b|_{x_b=0} = \frac{\frac{d^2y_b}{(dx_b)^2}|_{x_b=0}}{(1 + (\frac{dy_b}{dx_b}|_{x_b=0})^2)^{\frac{3}{2}}}, \tag{5-7}$$

where $\frac{dy_b}{dx_b}$ is given by (5-5) and the second derivative of the polynomial is given by:

$$\frac{d^2y_b}{(dx_b)^2} = 6c_3x_b + 2c_2. \tag{5-8}$$

Consequently, the curvature reconstruction error κ_e^b is given by:

$$\kappa_e^b = \kappa_{gt}|_{x_b=0} - \kappa_b|_{x_b=0}. \tag{5-9}$$

5-1-2 Error Calculation for Model-based Path Generation

For Model-based Path Generation, let the lateral distance of the generated path at time step b also be denoted by y_b. Similarly, let ψ_b and κ_b denote the virtual leader's path heading and curvature respectively. y_b is obtained by storing and extracting the state vector $\mathbf{s}_{M_k}^{F_b}$ from the Virtual Leader Model. More specifically, y_b is obtained from state $y_{M_k}^{F_b}$, which was defined in Section 4-2.

The lateral distance error y_e^b is given by the difference between the ground truth leader-vehicle path y_{gt} and the virtual leader-position history y_b, both evaluated at $x_b = 0$. Consequently, the lateral distance error y_e^b is found by (5-3).

Similarly, ψ_e^b is obtained from the virtual leader-vehicle's heading state and computing the difference given by (5-6).

The curvature of the virtual leader-vehicle's path κ_b can be obtained by:

$$\kappa_b = \frac{\dot{\psi}_{M_k}^{F_b}}{u_b}, \tag{5-10}$$

where $\dot{\psi}_{M_k}^{F_b}$ is the yaw rate, also extracted from the virtual leader-vehicle's state history. Moreover, u_b represents the longitudinal virtual vehicle's velocity. Consequently the curvature reconstruction error κ_e^b is given by (5-9).

Note that computing the reconstruction errors is based on evaluating different function values at the longitudinal location of the follower-vehicle, i.e. at $x_b = 0$. When there is no function value defined for exactly $x_b = 0$, it is found by linear interpolation.

5-1-3 Discontinuity Quantification

Another performance criterion is obtained when quantifying the discontinuity of a generated path. This quantification is done by computing the differences in generated path variables y_b, ψ_b, and κ_b between two consecutive samples as follows:

$$\begin{aligned} \text{diff}(y_b) &= y_b(k+1) - y_b(k), \\ \text{diff}(\psi_b) &= \psi_b(k+1) - \psi_b(k), \\ \text{diff}(\kappa_b) &= \kappa_b(k+1) - \kappa_b(k), \end{aligned} \tag{5-11}$$

where k is a simulation time step. As a result, the values of $\text{diff}(y_b)$, $\text{diff}(\psi_b)$ and $\text{diff}(\kappa_b)$ are related to the discontinuity of the path. This approach is motivated by the intuitive physical meaning of the computed differences. For instance, $\text{diff}(y_b)$ represents the jump in lateral position of the path in units of meter. This jump is directly related to the discontinuity of the generated path and can be put into perspective by comparing it to the width of the road or the discontinuity of the ground truth path (when available). Paths with large jumps in the path variables will be referred to as discontinuous paths. In contrast to this, paths with smaller jumps in the path variables will be referred to as more continuous paths. Note that this continuity interpretation deviates from the formal continuity definition, which is related to differentiability.

5-2 Simulator description

The mathematical models defined in preceding chapters form the basis of this custom-built simulator in Matlab & Simulink. The simulator is based on simulating two vehicles, where the first vehicle is the leader-vehicle and the second is the follower-vehicle. The follower-vehicle has onboard radar and camera systems, providing the relative leader-vehicle position measurements. This illustrating Waypoint Source is given by the Standard Sensor Systems. The mathematical models required for this Waypoint Source are reported in Appendix A. Based on analysis of TNO's Carlab, the waypoint sampling frequency is set to 10 Hz.

Using the output of the Waypoint Source, the implementation of the Waypoint Storage subsystem is run (see Fig. 2-1). The output of this subsystem is a list of (possibly noise-corrupted) waypoints. The waypoints are expressed in the local coordinate frame attached to the follower-vehicle F_b. Based on these waypoints, a path generation system is deployed.

In the custom-built simulator, all five different path generation systems from Chapters 3 and 4 are implemented. The output of the polynomial fitting methods, are lists of polynomial coefficients c_0 ... c_n. The output of the Model-based Path Generation implementations, are

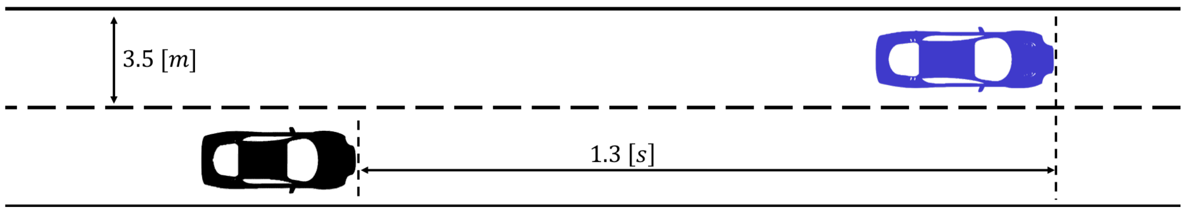


Fig. 5-2: A visualization of the simulated scenario. The vehicles drive at separate lanes at $100\,\mathrm{km\,h^{-1}}$. The follower-vehicle follows the leader-vehicle at a headway time of $1.3\,\mathrm{s}$. In meters, this headway is: $1.3\,\mathrm{s} \cdot 27.78\,\mathrm{m\,s^{-1}} = 36.11\,\mathrm{m}$.

lists of states $\mathbf{s}_{M_k}^{F_b}$ of the Virtual Leader Model. The values of the required path generation tuning parameters are reported in Section B-3 of Appendix B.

Simulated Scenarios

A general path generation scenario is simulated, analogue to the generic field of application described in Chapter 2. The vehicles are modeled by single-track bicycle models with parameters given in Table B-1 of Appendix B. The vehicles drive on a simulated straight highway with two $3.5\,\mathrm{m}$ wide lanes. The leader-vehicle is assumed to drive on the left lane. The follower-vehicle is assumed to drive on the right lane, behind the leader-vehicle, at a headway time of $1.3\,\mathrm{s}$. Both vehicles drive at a speed of $100\,\mathrm{km\,h^{-1}}$. This simulation scenario is visualized in Fig. 5-2.

The steer inputs to the vehicles are designed such that the vehicles perform lane-changes, staying on the two-lane road, and keeping their lateral accelerations limited. The designed steer input to the vehicles is shown in Fig. B-1 of Appendix B. This steering input results in a maximum occurring heading angle of $0.0377\,\mathrm{rad}$ with respect to the road. Moreover, the maximum occurring steering rate and lateral acceleration are $5.2 \times 10^{-3}\,\mathrm{rad\,s^{-1}}$ and $0.7\,\mathrm{m\,s^{-2}}$ respectively. This represents a normal driving scenario. This main scenario is split into two sub-scenarios. The first sub-scenario is given by the leader-vehicle performing the lane-changes, while the follower-vehicle drives straight. The second sub-scenario is given by the follower-vehicle performing the lane-changes, while the leader-vehicle drives straight.

5-3 Simulation Results

First, the two sub-scenarios are simulated with and without real-world disturbances, defined in Section 2-3. The simulated real-world scenarios include the expected magnitudes of the real-world disturbances. The results for two simulated ideal-world sub-scenarios, without real-world disturbances, are presented in Section 5-3-1. Subsequently, the results for two simulated real-world sub-scenarios, with real-world disturbances, are presented in Section 5-3-2. The maximum occurring reconstruction errors for all scenarios are given in Table B-3 of Appendix B. Finally, an extra scenario with user-defined waypoints is emulated. This extra scenario is presented in Section 5-3-3. For notation purposes, let the different path generation methods be denoted by numbers: Unconstrained Polynomial Fitting (Method 1), Heading Constrained Polynomial Fitting (Method 2), Curvature Constrained Polynomial Fitting (Method 3), Model-based Path Generation with Proportional Control (Method 4), and

Model-based Path Generation with MPC (Method 5). An overview of this notation is also given in Table 5-1.

Path Generation Method	**Notation**
Unconstrained Polynomial Fitting	Method 1
Heading constrained Polynomial Fitting	Method 2
Curvature constrained Polynomial Fitting	Method 3
Model-based Path Generation with Proportional Control	Method 4
Model-based Path Generation with Model Predictive Control (MPC)	Method 5

Table 5-1: Shorthand notation for different path generation methods.

5-3-1 Simulation in the Ideal World

In the first ideal-world sub-scenario, the leader-vehicle is performing the sine steering maneuver. The computed Virtual Driver steer inputs are shown in the left plot of Fig. 5-3. Both computed Virtual Driver steering inputs show similarity with the real leader-vehicle steering input, but shifted in time. This time-shift was explained in Section 4-1: it resulted from the virtual vehicle driving at a certain headway time t_v with respect to the real leader-vehicle. For this sub-scenario, the maximum occurring path generation reconstruction errors for all path generation methods are given in Table B-3a of Appendix B. To illustrate the overall performance of all five path generation methods, the reconstruction errors are plotted in Fig. 5-4. The main observation is that largest error is given by Method 4, which is 0.0379 m. This error can be regarded as negligible, considering the road lane width of 3.5 m.

In the second ideal-world sub-scenario, the follower-vehicle is performing the sine steering maneuver. The computed Virtual Driver steer inputs are shown in the right plot of Fig. 5-3. In this scenario version, the real leader-vehicle drove straight, without steering. All computed steering inputs are smaller than 6×10^{-4} rad, approaching zero and resembling the real leader-vehicle's steering input. For this sub-scenario, the resulting maximum occurring path generation reconstruction errors are given in Table B-3b of Appendix B. The main observation is that largest error is given by Method 2, which is 0.0324 m.

5-3-2 Simulation with real-world disturbances

Now real-world disturbances, as described in Section 2-3, are introduced. In the first real-world sub-scenario, the leader-vehicle is performing the sine steering maneuver. The computed steering inputs of the two Virtual Driver implementations for Model-based Path Generation, Methods 4 and 5, are shown in the left plot of Fig. 5-5. The figure clearly shows that the steering inputs of Method 4 are more turbulent and less bounded than the inputs of Method 5. In the second real-world sub-scenario, the follower-vehicle is performing the sine steering maneuver. For this case, the computed steer inputs of Methods 4 and 5 are shown in the right plot of Fig. 5-5. Clearly, both Virtual Drivers do not exactly imitate the real leader-vehicle's steering input in these real-world scenarios. This is caused by the fact that the

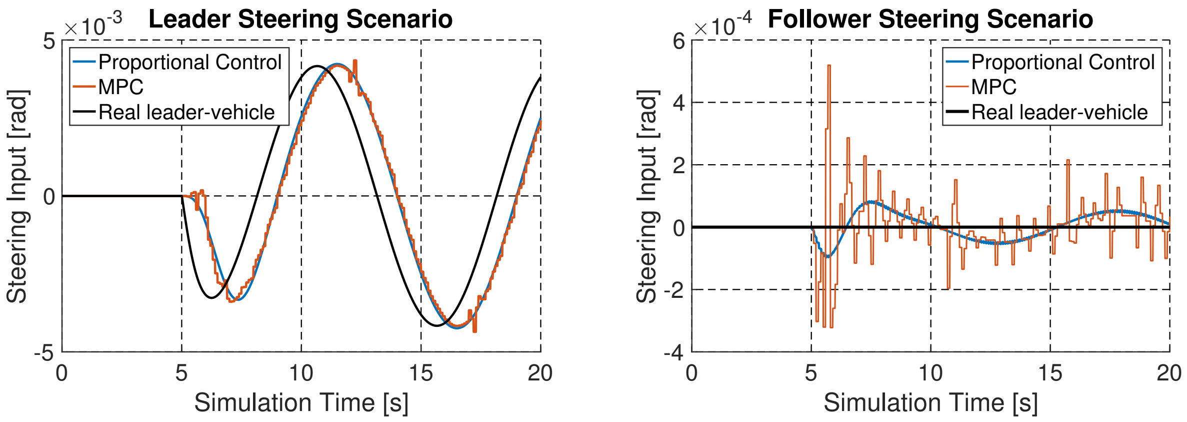


Fig. 5-3: The computed Virtual Driver Steering inputs in the ideal-world are shown and compared to the real leader-vehicle steering input. In the scenario of the left plot, the leader-vehicle performs the given sine steering input (and the follower-vehicle drives straight). Clearly, both Virtual Drivers approach the real leader-vehicle's steering input, shifted in time. In the scenario of the right plot, the leader-vehicle drives straight (and the follower-vehicle performs the given sine steering input). Note the different scale on the vertical axis.

Virtual Drivers now have to deal with the real-world disturbances. The resulting maximum reconstruction errors for all methods are given in Table B-3c and B-3d of Appendix B. Despite the fact that the real leader-vehicle steer input is not precisely imitated, the resulting Virtual Leader Models lead to accurate path reconstruction. To illustrate the overall performance of all five path generation methods, the reconstruction errors are plotted in Fig. 5-6. From the figures, it is clear that all three polynomial fitting methods (Methods 1, 2, and 3) result in paths with discontinuities in the error variables. Here, the informal interpretation of (dis)continuity is meant, which was explained in Section 5-1-3. Besides that, note that the paths merely look C^1-continuous (differentiable), due to the line plot, connecting the data. In practice, the paths are not continuous in terms of differentiability.

The unconstrained polynomial fit shows discontinuity in both ψ_e and κ_e. The heading constrained polynomial fit shows continuity in ψ_e, but discontinuity in κ_e. The curvature constrained polynomial fit shows continuity in κ_e, but discontinuity in ψ_e. In contrast to this, the plots show that the new Model-based Path Generation Methods 4 and 5 provides continuous results for all three error variables. This was to be expected since vehicle dynamics were explicitly incorporated in the Virtual Leader Model that has produced the trajectories. Moreover, Methods 4 and 5 show small reconstruction errors, compared to Methods 1, 2, and 3. The maximum occurring reconstruction errors are shown in Tables B-3c and B-3d of Appendix B. With Methods 4 and 5, the maximum occurring reconstruction error y_e is 0.108 m, compared to 0.245 m for Method 1 (in the second sub-scenario). Moreover, considering the path heading, the maximum occurring ψ_e is 0.008 rad, compared to 0.049 rad for Method 1. Finally, also the path curvature reconstruction error κ_e of Method 5 is smaller: $0.0009\,\mathrm{m}^{-1}$, compared to $0.0080\,\mathrm{m}^{-1}$ for Method 1. Summarizing, the paths resulting from the new Model-based Path Generation methods are not only more continuous, but also have minimum reconstruction error, compared to the polynomial fitting benchmark method. The continuity in path generation is further investigated and quantified in the next paragraph.

Fig. 5-7 shows the generated path in the simulation time interval from 15 s to 20 s. This

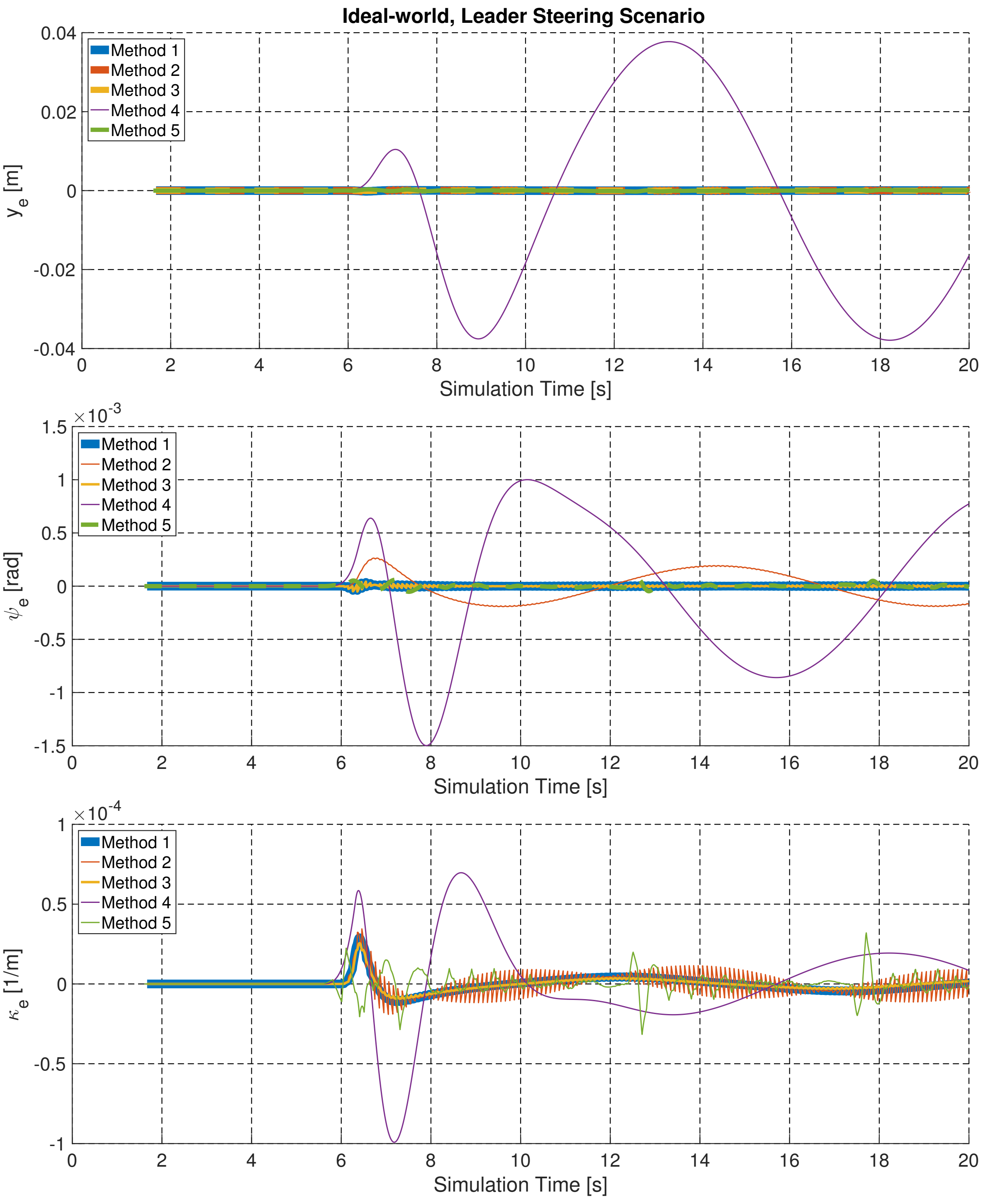


Fig. 5-4: The performance of the different path generators in the ideal-world is shown. The real-world disturbances are absent and the leader-vehicle is performing the sine steering maneuver, while the follower-vehicle drives straight. The reconstruction errors are plotted as functions of simulation time. All graphs start at the time instance at which the Waypoint Storage subsystem contains sufficient waypoints.

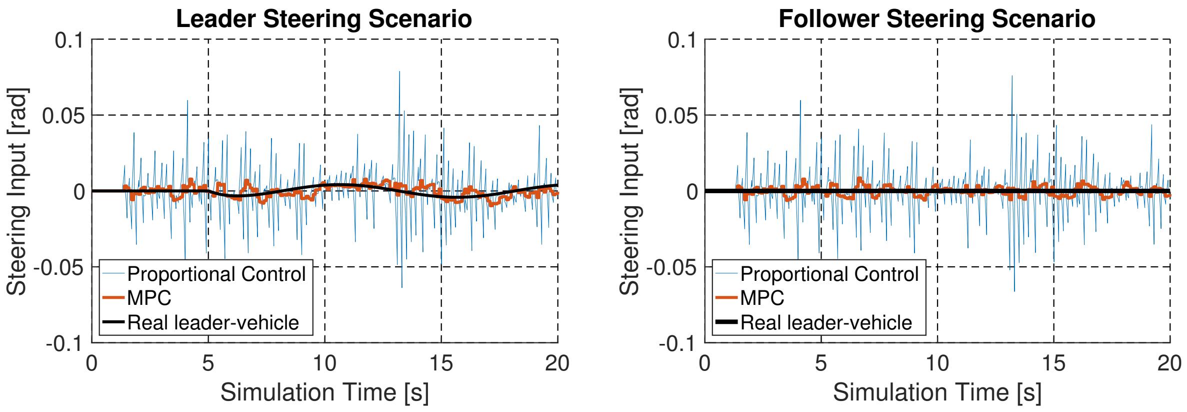


Fig. 5-5: The computed Virtual Driver Steering inputs in the real-world simulation are shown, compared to the real leader-vehicle steering input. In the scenario of the left plot, the leader-vehicle performs the given sine steering input (and the follower-vehicle drives straight). In the scenario of the right plot, the leader-vehicle drives straight (and the follower-vehicle performs the given sine steering input). Due to the real-world disturbances, the Virtual Driver inputs differ from the real leader-vehicle's steering input.

time interval is chosen because it makes the figure more clear to read. Moreover, the goal of the figure is to show the (dis)continuity of the resulting paths. For this, the chosen time interval suffices. The generated path is expressed in y_b, the lateral distance with respect to the local follower-vehicle reference frame F_b. Also, the path heading ψ_b and curvature κ_b are shown in Fig. 5-7. The discontinuous character of polynomial fitting path generation is clearly visible in this figure. To quantify this discontinuity, Fig. B-2 of Appendix B shows the differences in generated path variables y_b, ψ_b and κ_b, between two consecutive samples. These differences are calculated by (5-11). In Fig. B-2, the discontinuities are given by the jumps in the graphs. The maximum occurring jump, i.e., the maximum occurring difference in path variables y_b, ψ_b and κ_b, are summarized in Table B-4 of Appendix B. Observed is that polynomial fitting with heading constraint (Method 2) and with curvature constraint (Method 3), provide smaller jumps in their corresponding constrained path variable. This was theoretically substantiated in Section 3-2. The main observation is that the new Model-based Path Generation Methods 4 and 5 provide paths with the smallest jumps in path variables. For instance, comparing Method 5 to Method 1, the maximum occurring jumps in y_b are smaller by one order of magnitude. For the heading ψ_b and curvature κ_b, Method 5 realizes a decrease in discontinuity by two orders of magnitude.

5-3-3 Simulation with user-defined waypoints

Finally, an extra scenario is simulated, in which the Waypoint Source is not given by the Standard Sensor Systems detecting a leader-vehicle. Instead, the waypoints are user-defined, based on an extreme maneuver. Evaluating the performance of all path generators for this maneuver scenario identifies robustness of the path generation methods for extreme maneuver vehicle-following. Moreover, it demonstrates the generality of the new model-based path generation method. The new method is capable of producing feasible trajectories based on unfeasible waypoints. Unfeasible waypoints are waypoints that are not (re)producible by a vehicle, e.g., because they do not satisfy the non-holonomic motion constraints.

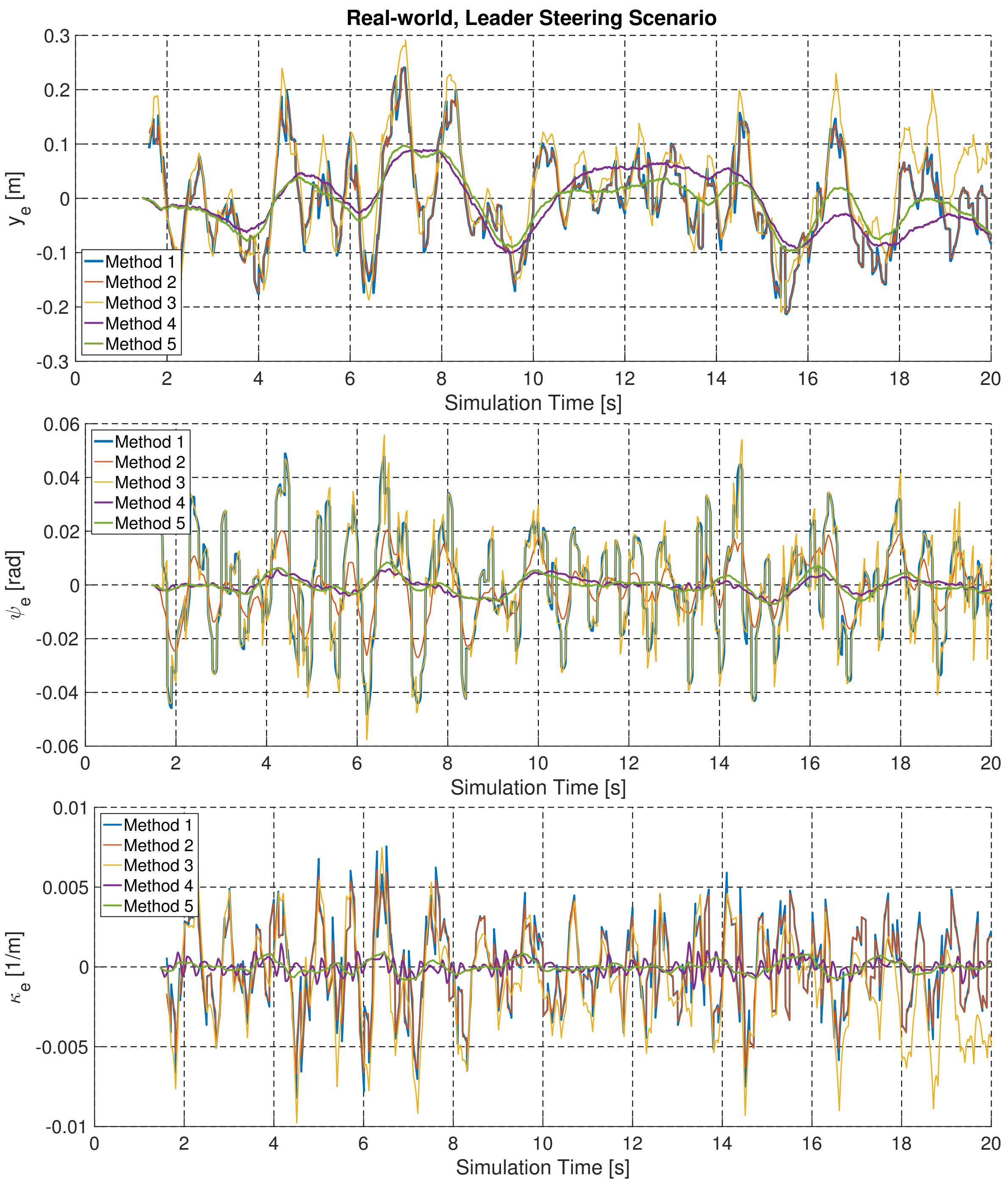


Fig. 5-6: The performance of the different path generators in the real-world simulation is shown. The real-world disturbances are Waypoint Noise, Delay, Offset and Motion Parameter Noise. The leader-vehicle is performing the sine steering maneuver, while the follower-vehicle drives straight. The reconstruction errors are plotted as functions of simulation time. Clearly, the new Model-based Path Generation Methods 4 and 5 provide continuous paths with and small reconstruction errors, in contrast to the Polynomial Fitting Methods 1, 2 and 3. All graphs start at the time instance at which the Waypoint Storage subsystem contains sufficient waypoints.

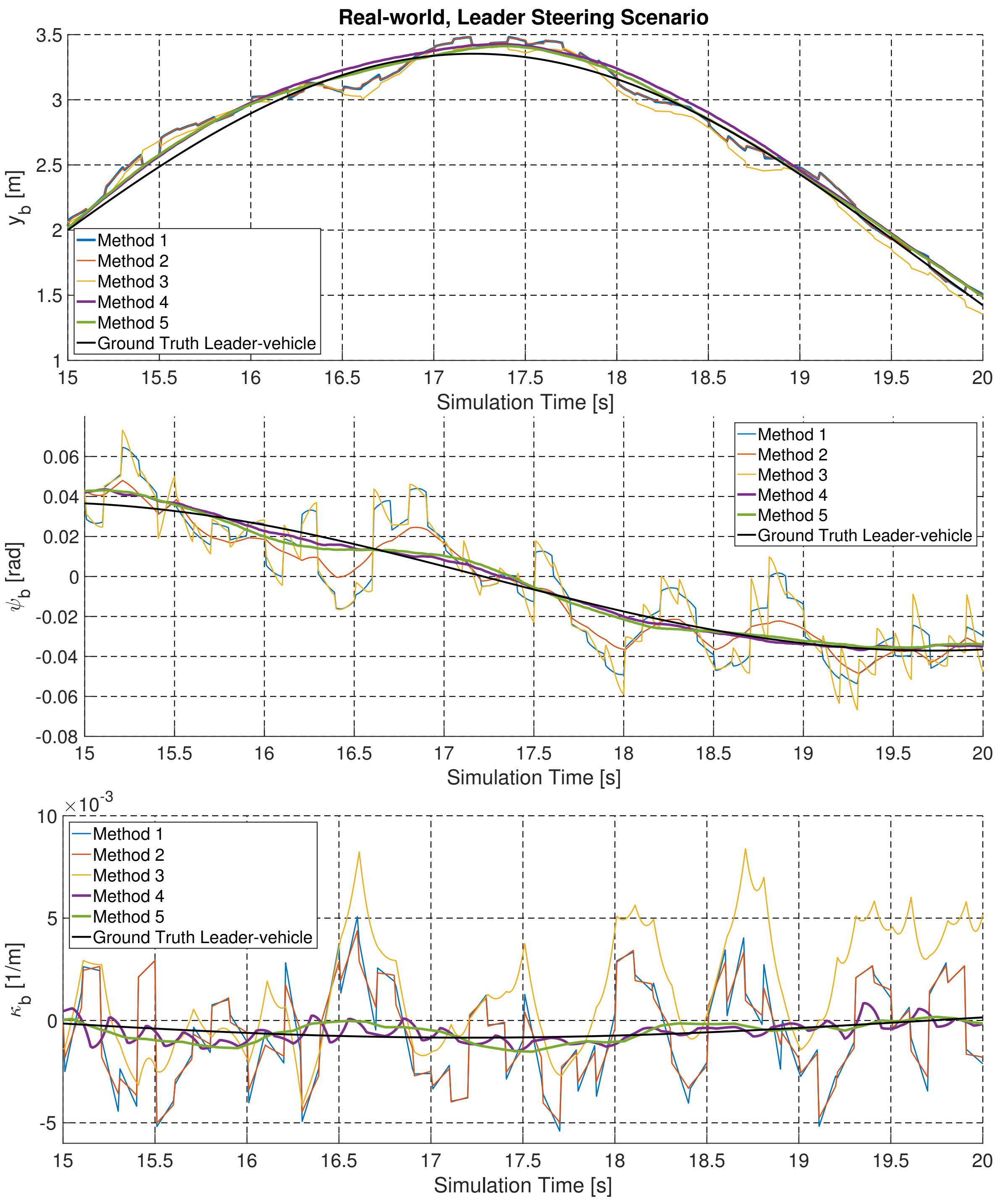


Fig. 5-7: The generated path variables y_b, ψ_b, and κ_b, from simulating the real-world scenario, are shown. Method 2 clearly shows a continuous heading variable ψ_b, which was expected from the heading constrained polynomial fitting method. Similarly, curvature constrained fitting Method 3 clearly shows a continuous curvature variable κ_b. Overall, the plots show the that new Model-based Path Generation Methods 4 and 5 provide more continuous paths, compared to the Polynomial Fitting Methods 1, 2 and 3.

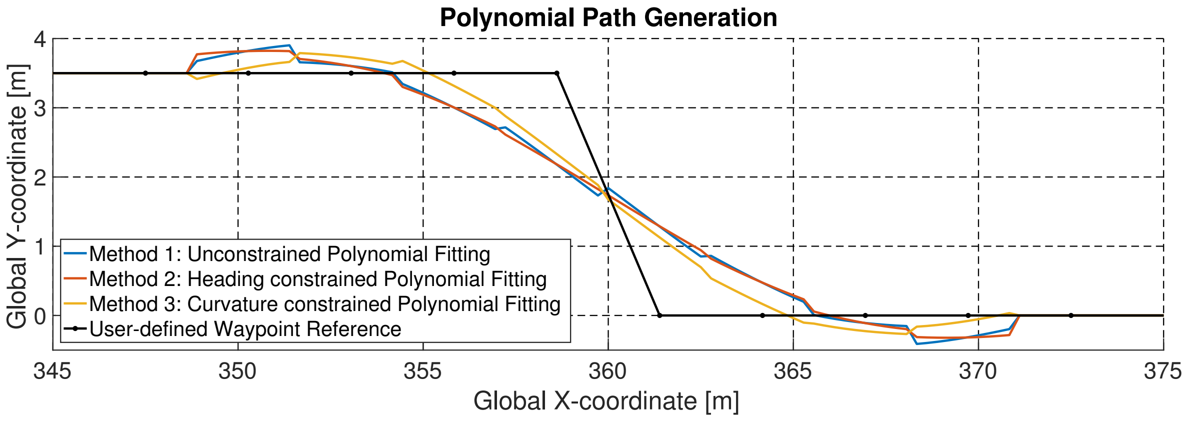


Fig. 5-8: This figure shows the incapability of the polynomial path generation methods to provide a smooth, continuous path, based on user-defined (extreme maneuver) unfeasible waypoints. The path resulting from the repetitive polynomial fitting is discontinuous because of the jumps in computed polynomial coefficients. This phenomenon was visualized in Fig. 3-3

.

The user-defined waypoints represent a sudden, discontinuous change of lanes, starting after ten seconds of straight driving. These waypoints are fed to the different path generation methods. For this specific scenario, the Model-based Path Generation implementations were re-tuned: the look-ahead-time t_{la} for Proportional Control was set to 2.0 s and the prediction horizon for both implementations was set to $N = 35$, which corresponds (with waypoints sampled at approximately 2.78 m distance) to approximately 100 m. The intuition behind this is given by how a human would perform the lane-change: by anticipating on the future, making forward predictions in the same order of magnitude as the chosen values for t_{la} and N. The values of the rest of the tuning parameters remained unchanged (see Section B-3 of Appendix B).

The resulting Polynomial Fitting paths of Methods 1, 2, and 3 are given in Fig. 5-8. The figure clearly shows the incapability of these methods to provide a continuous path, based on the user-defined, unfeasible waypoints (extreme maneuver). In Model-based Path Generation, the Virtual Driver implementations compute steering actions for the virtual vehicle. In this case, the steering actions are based on the user-defined waypoints. The computed steering actions are shown in Fig. 5-9. The resulting paths are given in Fig. 5-10. The figure clearly shows the capability of Methods 4 and 5 to provide a continuous path, based on a user-defined extreme maneuver. The Proportional Control implementation of the Virtual Driver shows the overshoot that was to be expected from a proportional feedback system. The Proportional Control lets errors (at a virtual look-ahead-distance) occur first, prior to taking action. In contrast to this, the MPC implementation of the Virtual Driver shows the desired behavior. The resulting path is not only continuous, but also imitates the behavior of a human-driver performing the lane-change. Similar to the human-driver, the MPC Virtual Driver performs forward predictions, minimizing potential look-ahead errors. The continuity of the resulting paths is quantified by calculating the differences in generated path variables y_b, ψ_b and κ_b, between two consecutive samples, given by (5-11). The maximum occurring difference in path variables y_b, ψ_b and κ_b, are summarized in Table B-5 of Appendix B. Here, the main observation is that Methods 4 and 5 provide more continuous reference paths than Methods

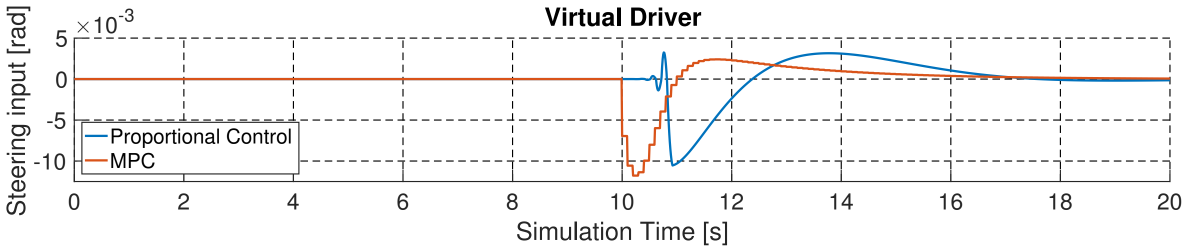


Fig. 5-9: The computed Virtual Driver steering actions are shown. Clearly, the MPC Virtual Driver starts steering earlier, anticipating on future offsets from the user-defined waypoint reference path. The Proportional Control lets errors (at a virtual look-ahead-distance) occur first, prior to taking action.

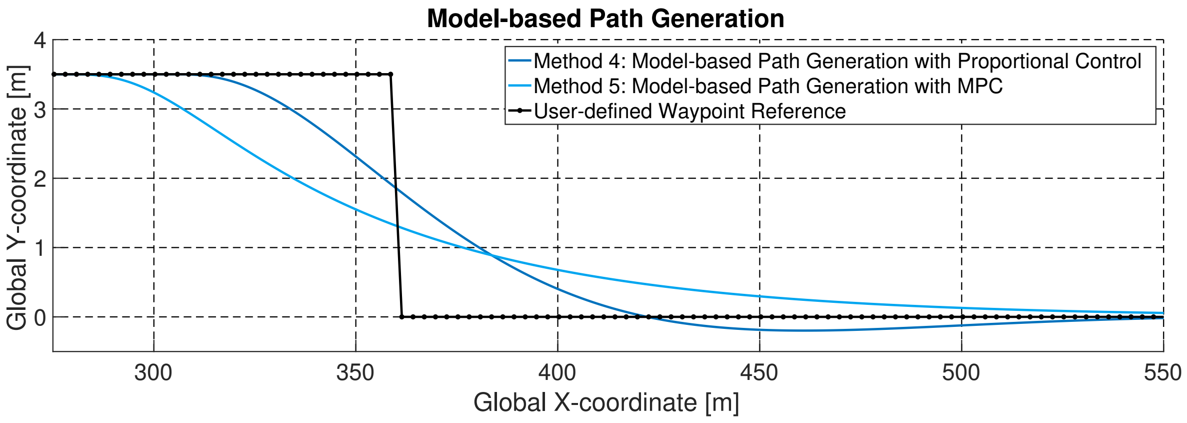


Fig. 5-10: This figure shows the capability of the Model-based Path Generation methods to provide a smooth, continuous path, based on user-defined (extreme maneuver) unfeasible waypoints. The Proportional Control Virtual Driver implementation results in a feasible path with a small overshoot, whereas the MPC implementation provides the desired reference tracking behavior.

1, 2, and 3.

For completeness, a snapshot of the simulation is shown in Fig. 5-11. The sequence of user-defined waypoints is visualized by the black dotted line in the figure. An instantaneous unconstrained polynomial fit is given by the smooth red curve. Note that this red curve is instantaneously continuous, because it is a snapshot of one single polynomial, rather than the path resulting from a sequence of polynomials. This was explained in Section 3-2 and visualized in Fig. 3-3. The desired reference path result is given by the new Model-based Path Generation method with the MPC Virtual Driver implementation. The resulting Virtual Leader Model trajectory is given by the light-blue line, representing the state-history of the virtual-vehicle. This example scenario demonstrates the general applicability of the new Model-based Path Generation method with the MPC Virtual Driver implementation. Clearly, the method can be used to generate feasible trajectories based on unfeasible waypoints, irrespective of the specific Waypoint Source. Moreover, this example shows the path generating robustness for the vehicle-following application, considering an extreme maneuvering leader-vehicle.

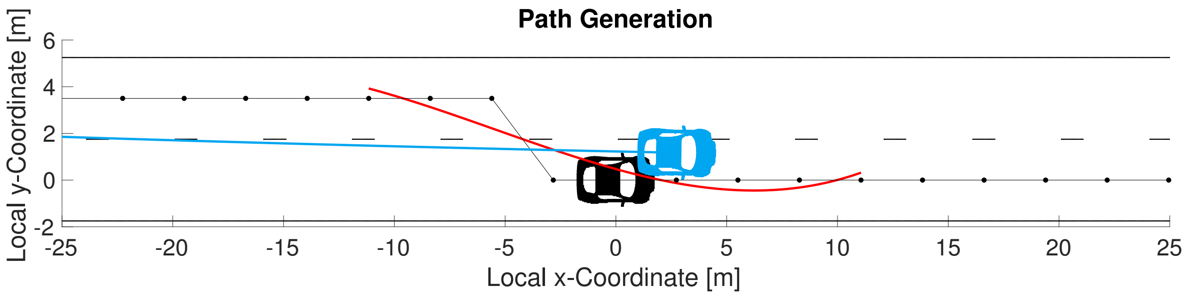


Fig. 5-11: A snapshot of the simulated scenario. The user-defined (unfeasible) waypoints are given by the black dotted line. An instantaneous unconstrained polynomial fit is shown by the red curve. The new Model-based Path Generation with the MPC Virtual Driver provides the desired path generation functionality. The resulting Virtual Leader Model trajectory is given by the light-blue line, representing the state-history of the light-blue virtual-vehicle.

Path Generation Method	**Computation Time**
Simulator without Path Generation	3.4880 s
Unconstrained Polynomial Fitting	4.4615 s
Heading constrained Polynomial Fitting	6.8084 s
Curvature constrained Polynomial Fitting	6.8560 s
Model-based Path Generation by Proportional Control Virtual Driver	4.4042 s
Model-based Path Generation by MPC Virtual Driver	10.8057 s
Simulator with all path generation methods active at the same time	17.2655 s

Table 5-2: The computation time is shown, required for 20 seconds of simulated driving.

5-3-4 Computational Demand

All aforementioned scenarios are emulated with the custom-built simulator in Matlab & Simulink, representing a driving scenario with a duration of 20 seconds. Using the build-in Matlab functions *tic* and *toc* the computation time of the different simulations is logged. This is done to be able to compare the computational demand of the different path generation implementations. Running the custom-built simulator without any path generation method active takes approximately 3.488 seconds for a driving scenario of 20 seconds. This was measured by disabling all path generation systems, running the Waypoint Source and Waypoint Storage subsystems for 10 times and taking the average computation time returned by *tic* and *toc*. This approach was repeated, running the custom-built simulator with different path generators active. The results are summarized in Table 5-2. This table clearly shows that Model-based Path Generation with the MPC Virtual Driver is the most computation demanding implementation. This was to be expected, since it makes use of control-input optimization based on forward predictions. Nevertheless, this method is still suitable for real-time implementation, since the simulations require less computation time than the real-time duration that is being simulated (20 seconds).

5-4 Conclusions

Results have shown that paths resulting from any polynomial fitting method are discontinuous. The discontinuous character was quantified by computing the maximum occurring difference in the position, heading, and curvature path variables (y_b, ψ_b, and κ_b). As an exception, the constrained polynomial fitting methods provided a more continuous path only in terms of their corresponding constrained path variable: heading constrained fitting provided jumps of smaller by one order of magnitude in ψ_b, compared to unconstrained fitting. Similarly, curvature constrained fitting provided jumps smaller by one order of magnitude in κ_b, compared to unconstrained fitting. However, the heading and curvature constrained fitting methods resulted in larger reconstruction errors, compared to unconstrained polynomial fitting, confirming the hypotheses regarding constrained polynomial fitting. In contrast to this, the new Model-based Path Generation, with MPC Virtual Driver, showed improved path reconstruction performance. Results showed that the maximum occurring position, heading and curvature reconstruction errors (y_e, ψ_e and κ_e) were significantly reduced, compared to polynomial fitting path generation. From this, it can be concluded that Model-based Path Generation with MPC outperforms the benchmark method in terms of accurateness.

Also for Model-based Path generation, the discontinuous character was quantified by computing the maximum occurring difference in the path variables (y_b, ψ_b, and κ_b). Model-based Path Generation with MPC, benchmarked against polynomial fitting path generation, provided a decrease by at least one order of magnitude in maximum occurring jumps in the path variables. From this, it can be concluded that the new Model-based Path Generation method also outperforms the benchmark method in terms of path continuity.

Next to that, user-defined waypoints, representing an extreme lane-change maneuver, were successfully processed by the new Model-based Path Generation method: a smooth reference path was realized, compatible with the vehicle dynamics. In contrast to this, the polynomial fitting methods resulted in discontinuous, unfeasible paths. This demonstrated the new method's robustness against unfeasible waypoints in vehicle-following. Moreover, it shows the method's applicability for path generation in general: the new method, with the MPC Virtual Driver, is capable of producing feasible trajectories based on unfeasible waypoints from any Waypoint Source.

Finally, results indicate that all proposed methods are suitable for real-time implementation. This conclusion is based on the fact that the required computation time is less than the simulated driving time.

Summarizing, it is concluded that the new Model-based Path Generation method outperforms polynomial fitting path generation in terms of accurateness, continuity, and general applicability. Hereby, the goal of this research is partly achieved: a generic reference path generator for lateral vehicle control is developed. Further verifying the path generator's robustness against real-world disturbances remains. This challenge is addressed in Chapter 6.

Chapter 6

Sensitivity Analysis

In preceding chapters, three different polynomial fitting and two model-based path generation methods are presented. In this chapter, the sensitivity of the methods towards real-world disturbances is evaluated. This is important to assess the feasibility and robustness of real-world implementation. Section 6-1 elaborates on the design of the sensitivity analysis.

The evaluation of the sensitivity analysis is based on numerical simulations. These simulations are carried out using the custom-built simulator, which was explained in Section 5-2.

The results from the sensitivity analysis are reported in Section 6-2. Finally, in Section 6-3, conclusions are drawn based on the obtained results.

6-1 Design

Four real-world disturbances were defined in Section 2-3. The corresponding mathematical models are given in Section A-2 of Appendix A. To test the sensitivity of all methods towards these real-world disturbances, multiple simulations are carried out. In the simulations, the magnitude of one disturbance is varied, while keeping the other disturbances absent.

The sensitivity towards Waypoint Noise is tested by the following steps:

1. The magnitude of the Waypoint Noise disturbance is set to a non-zero value. The magnitudes of all other disturbances are set to zero.
2. The vehicle-following scenario is simulated and the maxima of the occurring reconstruction errors (y_e^b, ψ_e^b, κ_e^b) are stored.
3. The magnitude of the Waypoint Noise disturbance is changed, according to (6-1), and the simulation is run again.

$$\text{Disturbance Magnitude} = \text{Expected Magnitude} \times \text{Disturbance Level} \tag{6-1}$$

In (6-1), the Disturbance Level is given by a sequence of multiplication factors from 0 to 2 in steps of 0.1. For each iteration of step 3 described above, a different Disturbance Level multiplication factor is set. The Expected Magnitude for the different disturbances are given by Table A-1. Consequently, the Disturbance Magnitude is the specific magnitude of the disturbance used in the simulation. By varying this magnitude, the maxima of the occurring reconstruction errors (y_e^b, ψ_e^b, κ_e^b) change. Based on this change of maximum errors, the sensitivity of a path generation method is identified. The maximum reconstruction errors are plotted in a figure with the disturbance level on the horizontal axis and the maximum occurring reconstruction errors on the vertical axis. As a result, the slope of the graph represents the disturbance sensitivity.

6-2 Results

Four real-world disturbances were defined in Section 2-3. For all path generation methods, the sensitivity towards these real-world disturbances is tested by the procedure explained in Section 6-1. This section presents the corresponding results. To refer to the five different path generation methods, the same shorthand notation of Table 5-1 is used: Unconstrained Polynomial Fitting (Method 1), Heading Constrained Polynomial Fitting (Method 2), Curvature Constrained Polynomial Fitting (Method 3), Model-based Path Generation with Proportional Control (Method 4), and Model-based Path Generation with MPC (Method 5).

6-2-1 Waypoint Noise

First, the sub-scenario where the leader-vehicle performs the given sine steering maneuver is emulated. The follower-vehicle drives straight. The sensitivity of the path generation methods towards Waypoint Noise is shown in the left column of Fig. 6-1. From the plots it can be concluded that Methods 4 and 5 have the best sensitivity characteristics for y_e, i.e., the slopes are the least steep. Furthermore, also for reconstruction errors ψ_e and κ_e, the best slopes are given by Methods 4 and 5.

Secondly, the sub-scenario where the follower-vehicle performs the given sine steering maneuver is emulated. The leader-vehicle drives straight. The Waypoint Noise sensitivity for this scenario is shown in the right column of Fig. 6-1. The results are similar to the results from the left column. Also for this scenario, Methods 4 and 5 show to be the least sensitive towards the Waypoint Noise disturbance; that is, they provide the least steep slopes for all reconstruction errors. For ψ_e, Method 2 shows improved robustness compared to other polynomial fitting Methods 1 and 3. This can be explained by the fact that Method 2 is based on constraining the heading of the fitted polynomial.

6-2-2 Motion Parameter Noise

For the first sub-scenario, the sensitivity of the path generation methods towards Motion Parameter Noise is shown in the right column of Fig. 6-2. For y_e with noise levels below 1, the figure shows that the magnitude of reconstruction error contribution is largest for the Model-based Path Generation implementations: Methods 4 and 5. This can be explained by

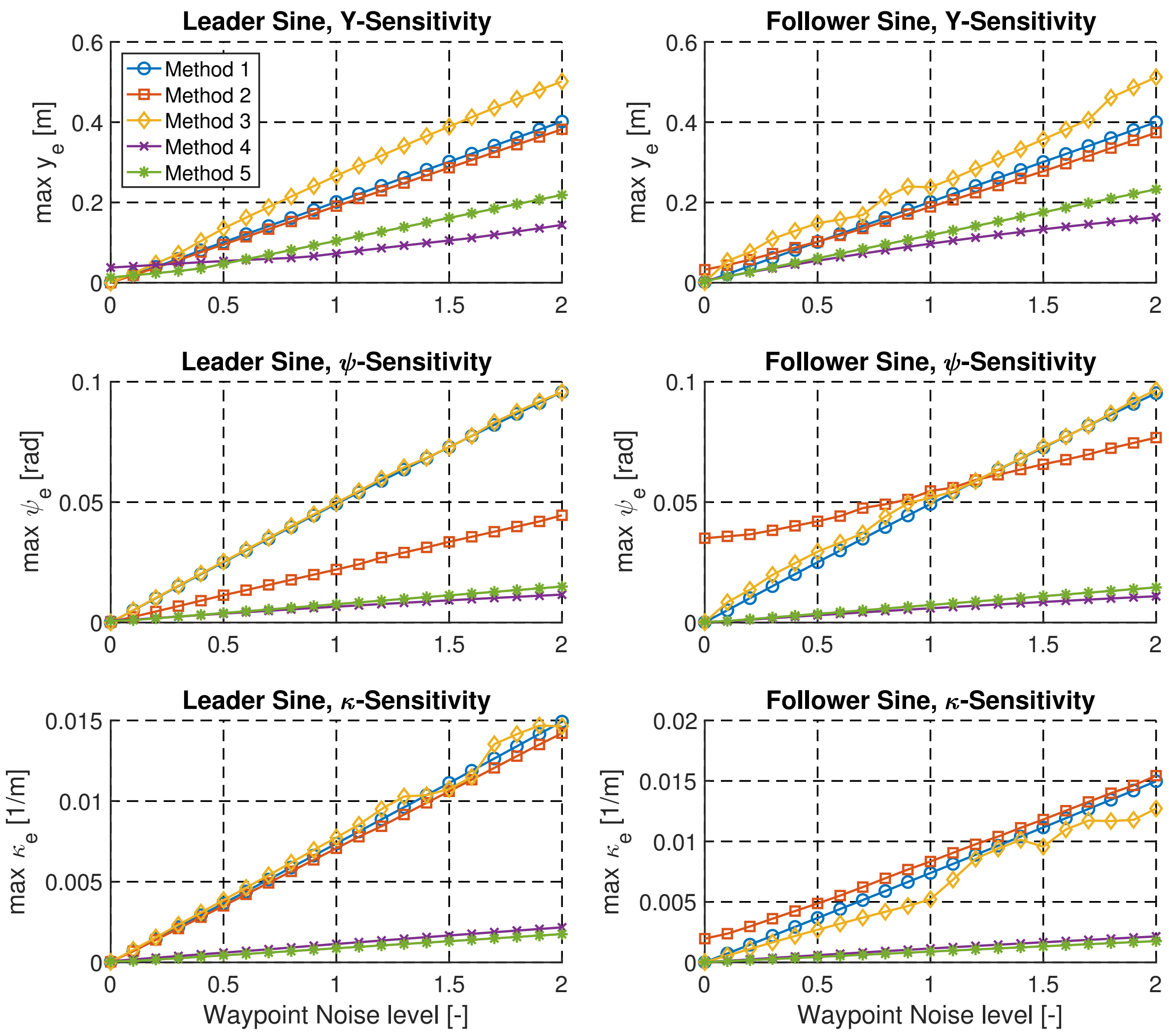


Fig. 6-1: Waypoint Noise Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied.

the fact that these methods use the motion parameters twice for vector mapping: first for waypoint mapping and secondly for Virtual Leader Model state mapping. For polynomial path generation, the motion parameters are only used for waypoint mapping. Despite this fact, the difference in the magnitude of error contribution is negligible compared to the Waypoint Noise induces error. Overall, the best robustness is again given by Methods 4 and 5: the slopes of the lines in the figures of ψ_e and κ_e are the least steep.

For the second sub-scenario, the motion parameter noise sensitivity is shown in right column of Fig. 6-2. For this scenario, the sensitivities of Methods 1, 4, and 5 are similar for all three types of reconstruction errors. For ψ_e, Method 2 shows insensitivity: the slope of the line is approximately zero. This is not surprising, since Method 2 was using a constraint on the first order derivative of the polynomial, representing a constraint on the heading of the polynomial. For κ_e, Method 2 shows the same behavior. This indicates that the heading constraint effectively makes the polynomial fitting process robust against Motion Parameter Noise, but only for this scenario and only for ψ_e, κ_e and y_e with noise levels below 0.5. Method 3 shows deviating sensitivity for κ_e only, which can be explained by the fact that it is based on constraining the curvature of the fitted polynomial.

Overall, Model-based Path Generation (Methods 4 and 5) again show to be the most robust, especially for κ_e.

6-2-3 Waypoint Delay

The sensitivity analysis results for Waypoint Delay are reported in Fig. 6-3. This figure shows that all methods are robust against Waypoint Delay. This was expected since the Waypoint Delay is compensated for, using the motion parameters. The necessity of the delay compensation was explained in Section 2-3.

6-2-4 Waypoint Offset

For both sub-scenarios, the results are shown in Fig. 6-4. In the right column, all slopes are approximately zero, as expected for the sub-scenario with the follower-vehicle performing the sine maneuver and the leader-vehicle driving straight. In the left column, the three polynomial fitting Methods 1, 2, and 3 have similar Waypoint Offset sensitivity for y_e. This can be explained by the fact that they are all based on polynomial fitting. Methods 4 and 5 show approximately zero slope for Waypoint Offset levels up to 1. With Waypoint Offset levels larger than 1, the methods show negative slope for ψ_e and κ_e, indicating negative sensitivity. It is expected that this is caused by the settings for prediction horizon N and control horizon N_c. In the absence of the other disturbances, the reconstruction error is increased due to forward prediction with $N_c < N$. This causes reconstruction error because less different steer inputs are calculated than the number of incorporated future offsets. Having a larger Waypoint Offset compensates this prediction-induced error, because the algorithm anticipates on the rear of the preceding vehicle, rather than its Center of Gravity (CoG). This results in a smaller effective prediction horizon. Methods 1, 2, and 3 show similar behavior in the plots of maximum occurring ψ_e and κ_e. For these polynomial fitting methods, this behavior can be explained by the intrinsic filtering, given by the least-squares fitting. Depending on the Waypoint Offset level, this filtering leads to polynomial fitting behavior that results in

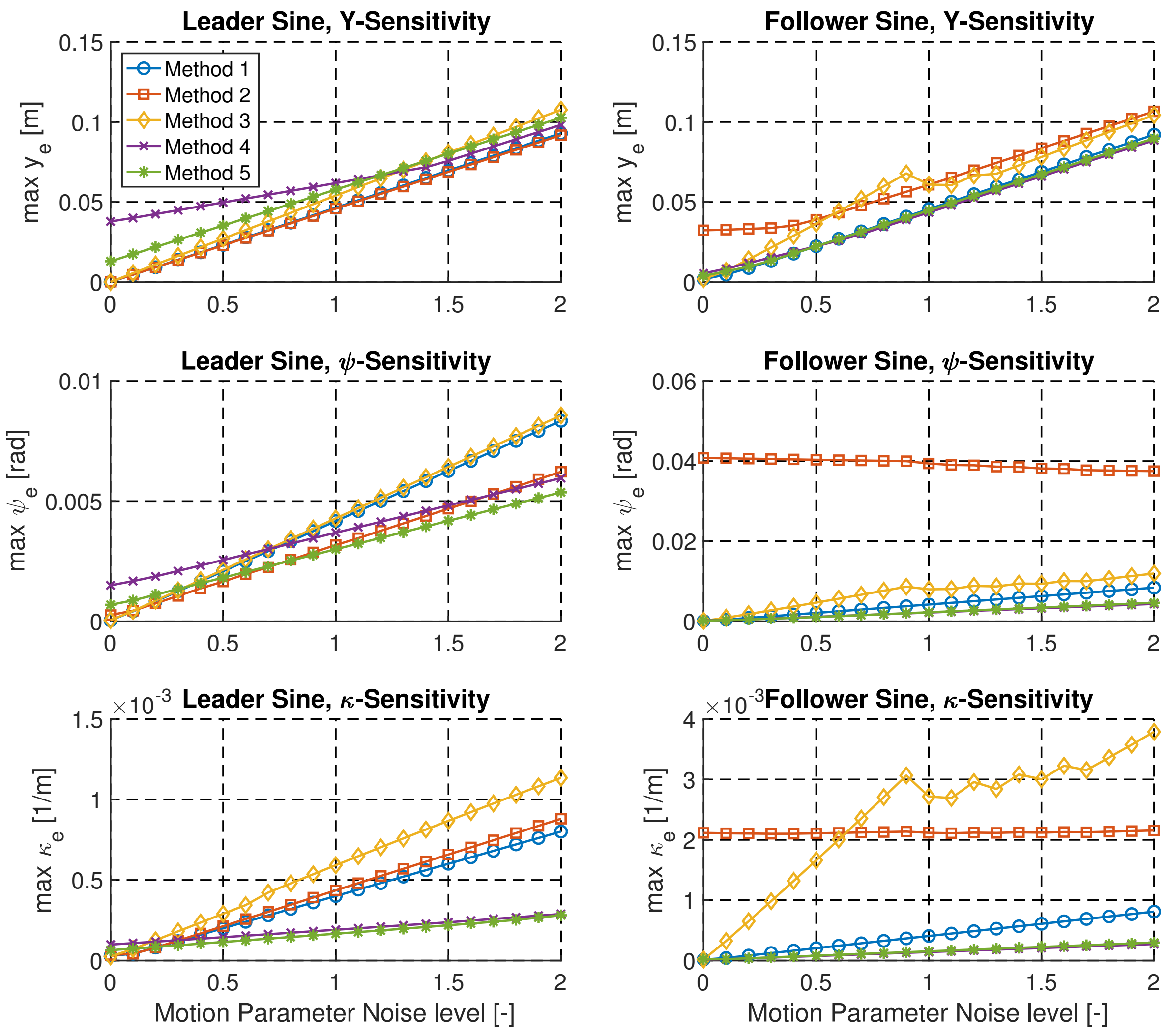


Fig. 6-2: Motion Parameter Noise Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied.

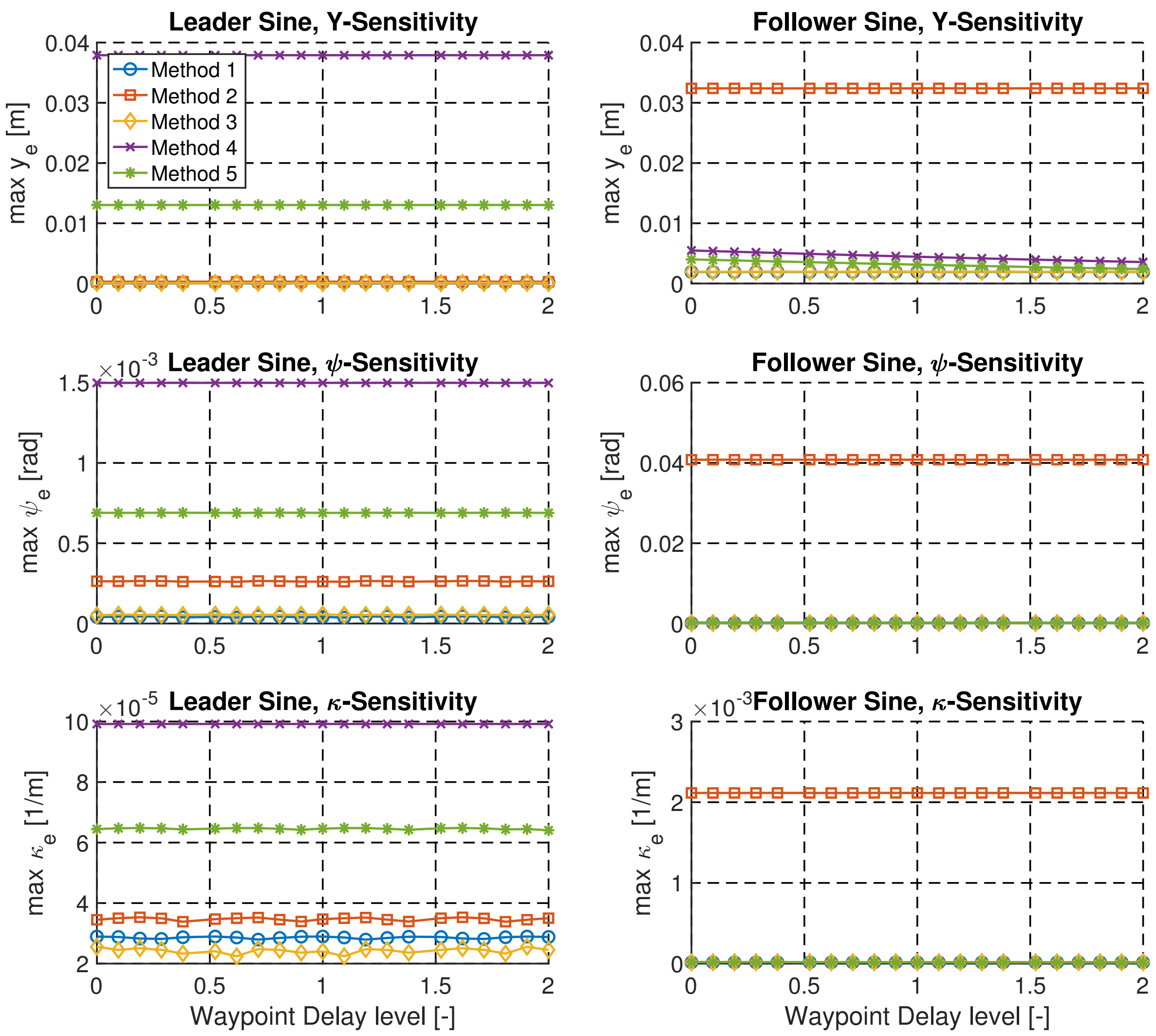


Fig. 6-3: Waypoint Delay Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied.

an accurate path reconstruction of the vehicle rear, rather than its CoG. As a result, the reconstruction errors may get smaller, as the Waypoint Offset level is varied.

In general, all slopes are approximately zero. Moreover, the absolute error contribution of Waypoint Offset shows to be insignificant, compared to Waypoint Noise (see Fig. 6-1). Consequently, the main observation is that the sensitivity of all methods towards Waypoint Offset is negligible.

6-3 Conclusions

The performed sensitivity analysis showed the relative robustness of the five different path generation implementations. For the Waypoint Noise disturbance, Model-based Path Generation (Methods 4 and 5) not only showed the lowest maximum occurring reconstruction errors, but also the best robustness. The robustness was visualized by the slope of the graphs in Fig. 6-1. The steeper the slope, the more sensitive the method is towards increasing the magnitude of the specific disturbance. The more sensitive the method is, the less robust it is. Also for the Motion Parameter Noise, Methods 4 and 5 outperformed the rest of the methods in terms of robustness. Furthermore, results showed that all methods are robust against Waypoint Delay, because this delay was successfully compensated for, using the methodology proposed in 2-2. Finally, the sensitivities towards Waypoint Offset were analyzed. The observed variations in occurring reconstruction errors are negligible, compared to the variations in error induced by Waypoint Noise. Therefore, it can be concluded that all methods are robust against the Waypoint Offset disturbance.

Summarizing, it is concluded that the new Model-based Path Generation method outperforms the benchmark polynomial fitting method in terms of robustness. Hereby, the goal of this research is achieved: a generic, robust reference path generator for lateral vehicle control is developed.

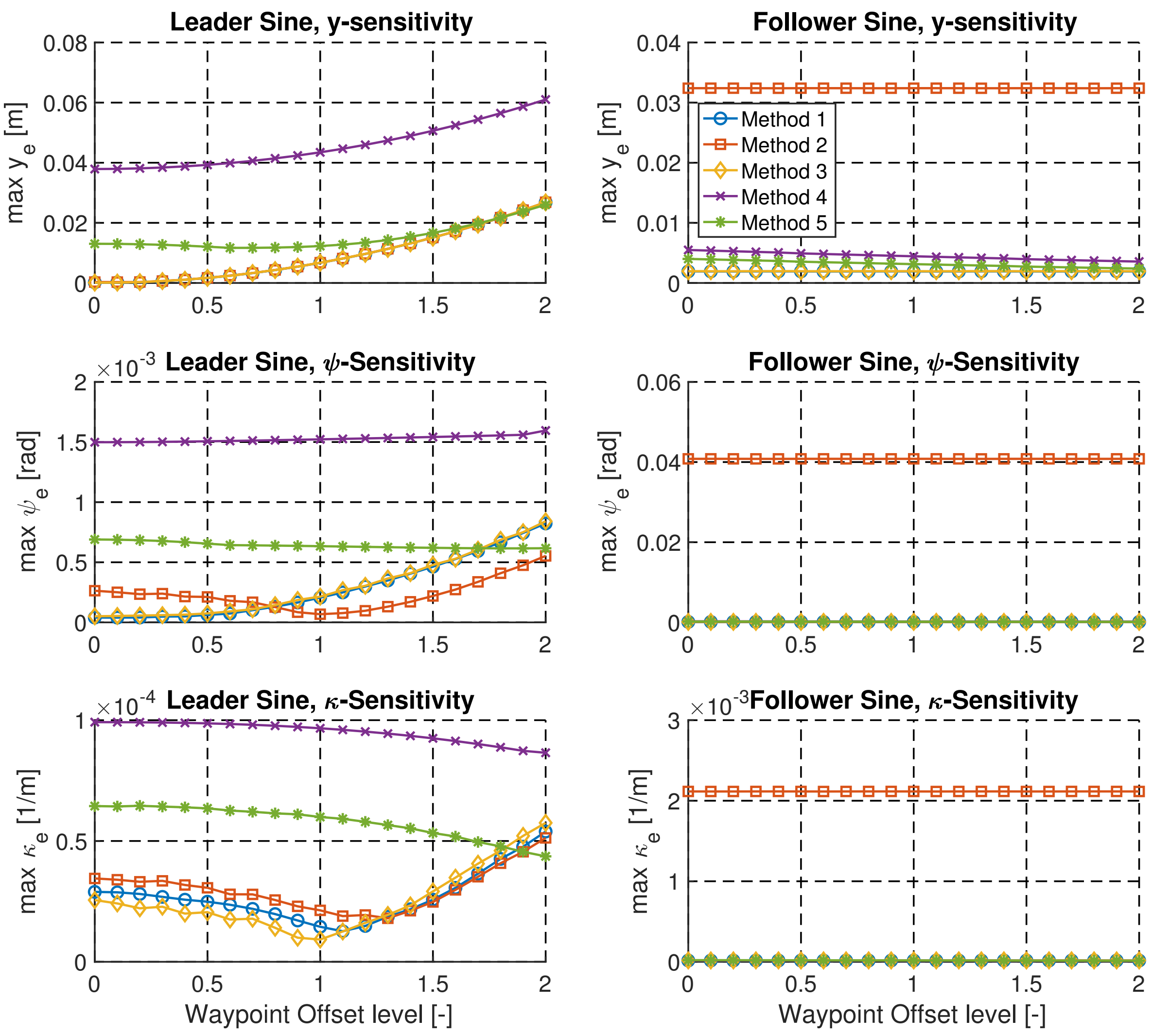


Fig. 6-4: Waypoint Offset Sensitivity. The plots in the left column are generated by simulating the first sub-scenario, where the leader-vehicle performs the sine steering maneuver. The right column corresponds to second sub-scenario, in which the follower-vehicle performs the sine steering. The sensitivity is given by the slope of the graph: the rate of change of the maximum occurring reconstruction error, as the disturbance level is varied.

Chapter 7

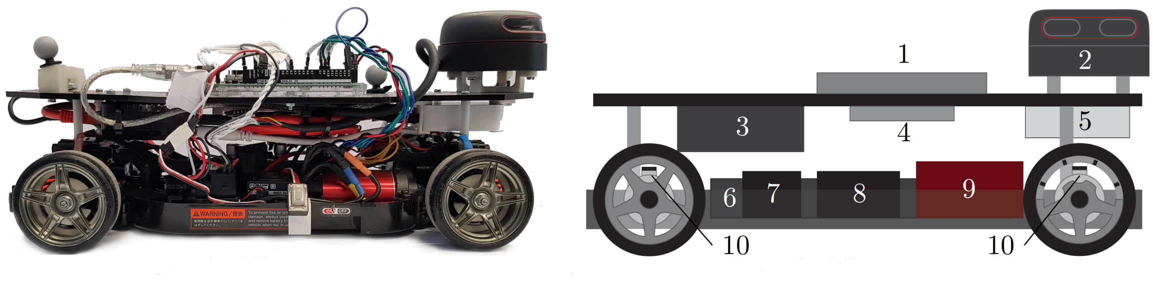
Experimental Analysis

A total of five different path generation systems were obtained: three different polynomial fitting methods and two model-based path generation methods. To validate these five different path generating methods, they are implemented using two sets of experimental data. The first set of experimental data was obtained by driving scaled vehicles in the lab of the Delft Center for Systems and Control (DCSC), discussed in Section 7-1. The second set of experimental data was obtained by TNO's Carlab, discussed in Section 7-2. Finally, in Section 7-3 conclusions are drawn based on the obtained results.

7-1 Scaled Vehicles

To obtain the first set of experimental data, two 1:10 scaled vehicles were driven in the lab of DCSC. Using the lab's Motion Capture system, the position and heading data of the vehicles was logged. During this experiment, the leader-vehicle performed a sine steering maneuver. The Delft Scaled Vehicle (DSV) was used as a follower-vehicle, which drove behind the leader-vehicle. Both scaled vehicles were operated manually by a remote control. In Fig. 7-1 the DSV is shown, along with a schematic overview of its components. The encoders were used to measure the longitudinal velocity. Additionally, the Inertial Measurement Unit (IMU) measured the yaw rate of the follower-vehicle. The lidar was not used, because it is unsuitable for leader-vehicle detection. The sensors and actuators were connected to an onboard computer, which runs the Robotic Operating System (ROS). This operating system enabled synchronized data logging with the Motion Capture system.

The resulting experimental data is presented in Section 7-1-1. In the simulations of Chapter 5, the vehicle and sensor models of Appendix A were used to emulate a Waypoint Source and the Standard Sensor Systems. Here, within the experimental analysis, the experimental data is used instead. Subsequently, the different path generation methods are implemented and validated. The results are presented in Section 7-1-2.



(a) The DSV **(b)** Schematic of the DSV

Fig. 7-1: The DSV, with 1: Arduino, 2: Lidar, 3: Onboard computer, 4: IMU, 5: Router, 6: Battery, 7: Servo, 8: Motor controller, 9: Motor, 10: Encoders. Visualization courtesy of Mart Baars.

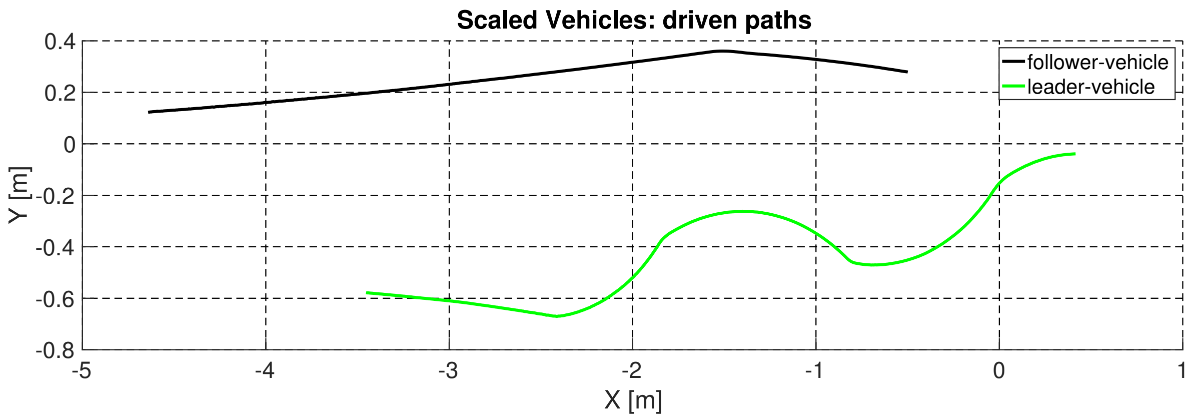


Fig. 7-2: A visualization of the paths driven by the scaled vehicles, obtained by the Motion Capture system with an average sampling time of $0.0127\,\mathrm{s}$. During the experiment, the leader-vehicle drove in front of the follower-vehicle. The graph is to be interpreted from left to right, with increasing global X-coordinate.

7-1-1 Used experimental data

The paths driven by the scaled vehicles are visualized in Fig. 7-2 and Fig. 7-3. This data was obtained using the Motion Capture system of the lab of DCSC, serving as ground truth data of the scaled vehicles.

Within this experimental analysis, the Waypoint Source is provided by calculations of the relative position of the leader-vehicle with respect to the follower-vehicle, using the Motion Capture system's position data at a rate of $10\,\mathrm{Hz}$. The resulting Waypoint Source data is presented in Fig. 7-4. Note that this Waypoint Source implementation results in relatively accurate leader-vehicle position measurements, compared to a radar Waypoint Source for instance.

During the experiment, the ego-motion parameters of the follower-vehicle (DSV) were measured by onboard sensor systems. The longitudinal velocity was obtained from wheel-speed sensors. Furthermore, the follower-vehicle's yaw rate was measured using an IMU. The logged ego-motion parameter data is shown in Fig. 7-5. This finalizes the discussion of the experimental data that serves as Waypoint Source and Standard Sensor System in the path

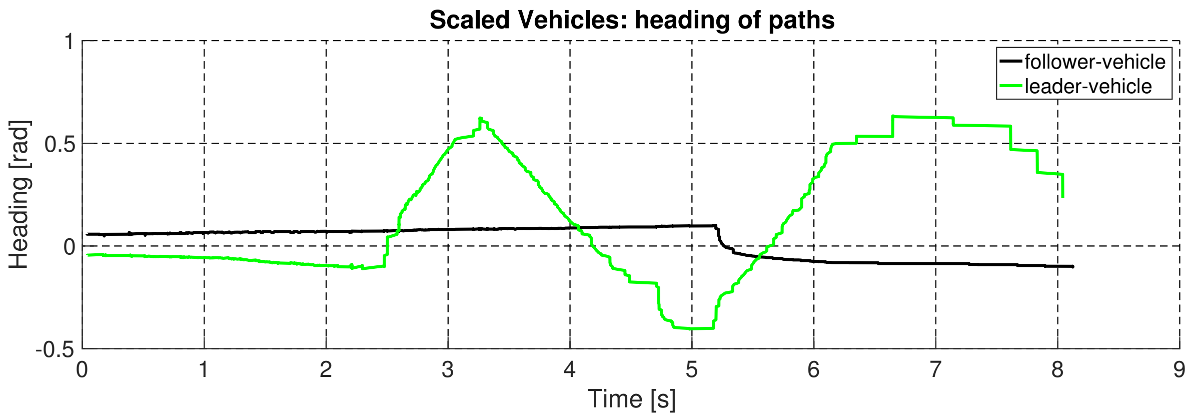


Fig. 7-3: A visualization of the heading of the scaled vehicle paths, obtained by the Motion Capture system with an average sampling time of $0.0127\,\mathrm{s}$.

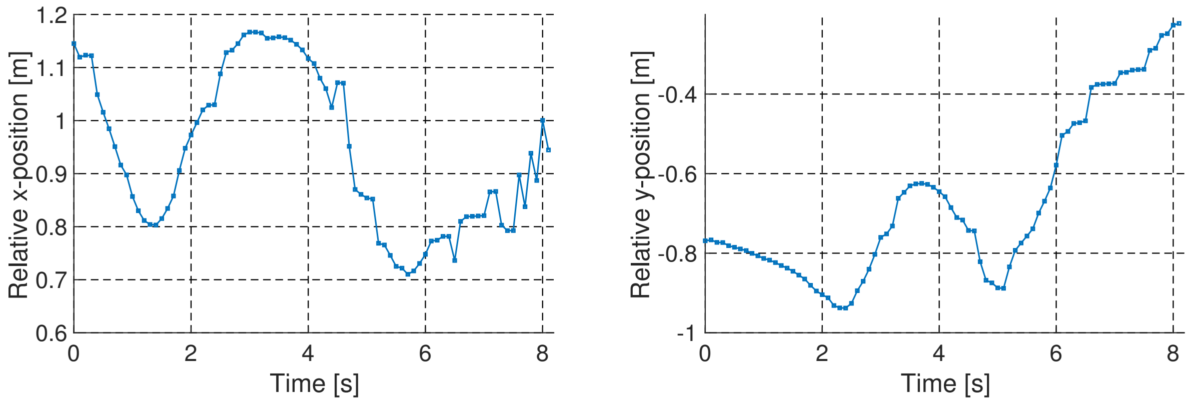


Fig. 7-4: A visualization of the Waypoint Source data, obtained using relative position information from the Motion Capture system. The sampling time is set to $0.1\,\mathrm{s}$.

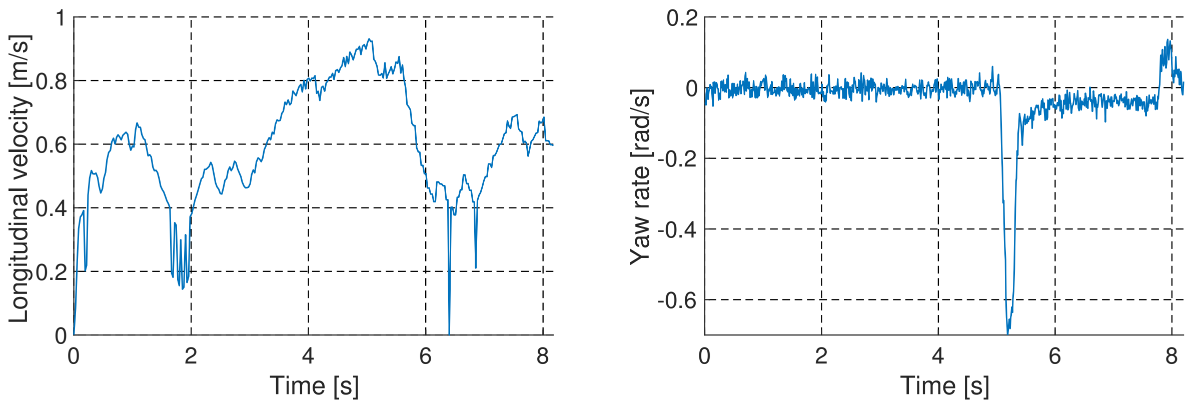


Fig. 7-5: A visualization of the logged data of the longitudinal velocity and yaw rate of the follower-vehicle. The average sampling times of the longitudinal velocity and yaw rate data are $0.0261\,\mathrm{s}$ and $0.0099\,\mathrm{s}$ respectively.

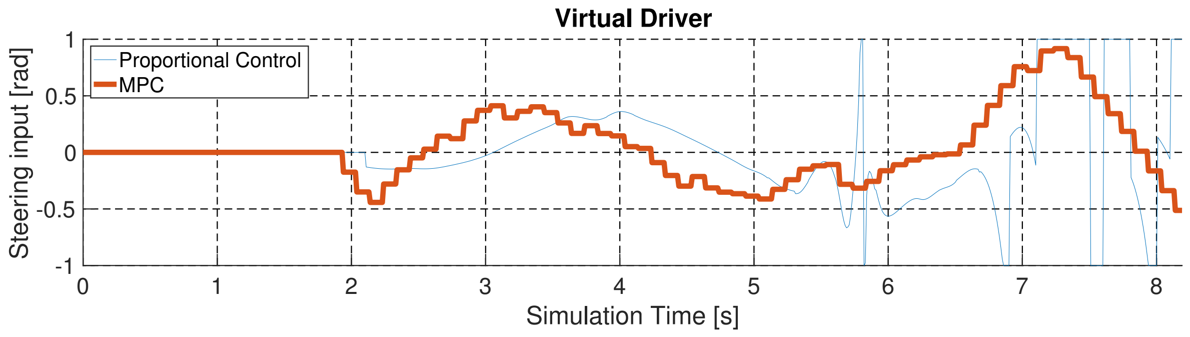


Fig. 7-6: The computed Virtual Driver steering inputs for the Virtual Leader Model are shown. The Proportional Control inputs are limited to a maximum absolute value of $1\,\mathrm{rad}$ for stability purposes.

generation simulator. Based on the relative position measurements depicted in Fig. 7-4, the Waypoint Storage subsystem provides a list of waypoints. Using these waypoints, the different path generation methods are deployed, reconstructing the path driven by the leader-vehicle.

7-1-2 Results

In the new Model-Based Path Generation implementations, the single-track model parameters given in Table C-1 of Appendix C are used. Additionally, the tuning parameters reported in Section B-3 of Appendix B are used. The resulting Virtual Driver steering inputs are shown in Fig. 7-6. This figure clearly shows that Proportional Control (Method 4) provides more turbulent and less bounded steering inputs than Model Predictive Control (MPC) (Method 5) does.

The reconstruction errors are calculated using the path variables y_b, ψ_b, and κ_b. The definitions of these reconstruction errors were given in Section 5-1. The reconstruction errors of the generated paths are shown in Fig. 7-8. When the follower-vehicle overtakes the virtual vehicle, the path variables are undefined at the local x-coordinate of the follower-vehicle. This phenomenon is visualized in Fig. 7-7. As a result, the reconstruction errors cannot be

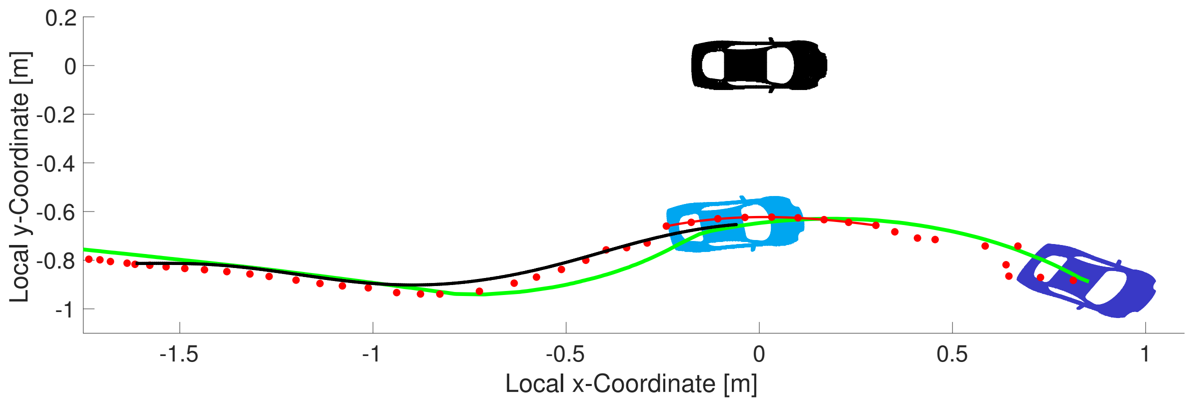


Fig. 7-7: A visualization of the scaled vehicle experiment, where the follower-vehicle overtakes the virtual vehicle. The follower-vehicle is black, the virtual vehicle is light-blue and the leader-vehicle is dark blue. Furthermore, the ground truth leader-vehicle path is given by the green line, obtained from the Motion Capture system. The waypoints from the Waypoint Storage subsystem are visualized by the red dots. A single unconstrained polynomial fit of Method 1 is visualized by the solid red curve. Furthermore, the path resulting from Model-based Path Generation Method 5 is given by the black line, representing the position state history of the virtual vehicle. The visualization also shows encoder latency, as the waypoints do not exactly lie on the green ground truth path. Moreover, packet loss causes a varying distance between the waypoints.

calculated for this short period. To mitigate this, Model-based Path Generation Methods 4 and 5 may either use extrapolation of the path variables or future state predictions, in order to compute the reconstruction errors. This mitigation is left for future implementation. Furthermore, note that there is no ground truth measurement of the path curvature, since the Motion Capture system only measures vehicle position and heading (pose). As a result, the curvature reconstruction error κ_e is undefined. The maximum occurring reconstruction errors are summarized in Table 7-1. The main observation from this experimental scenario is that all methods have similar reconstruction performance in terms of y_e. The resulting relative performances are different from the results obtained by simulations, which can be explained by model mismatch, sensor latency, and packet loss (see Fig. 7-7). However, when considering heading error ψ_e, Model-based Path Generation Methods 4 and 5 clearly outperform polynomial fitting Methods 1, 2, and 3.

Table 7-1: Maximum occurring reconstruction errors with y_e in m, ψ_e in rad.

Error	Method 1	Method 2	Method 3	Method 4	Method 5
max y_e	0.12721	0.13253	0.12739	0.17335	0.12772
max ψ_e	0.44152	0.53019	0.52299	0.25431	0.25053

To quantify the discontinuity of the resulting paths, the differences in generated path variables y_b, ψ_b, and κ_b are calculated by (5-11). By doing so, the continuity of all path generation methods is quantified. Severe discontinuities result in large values of the calculated differences in path variables, which are reported in Fig. C-1 of Appendix C. The maximum occurring differences in path variables y_b, ψ_b and κ_b, are summarized in Table 7-2. From this table it becomes clear that all three polynomial fitting methods (Methods 1, 2, and 3) result in paths with larger discontinuities in the error variables, compared to Model-based Path Generation

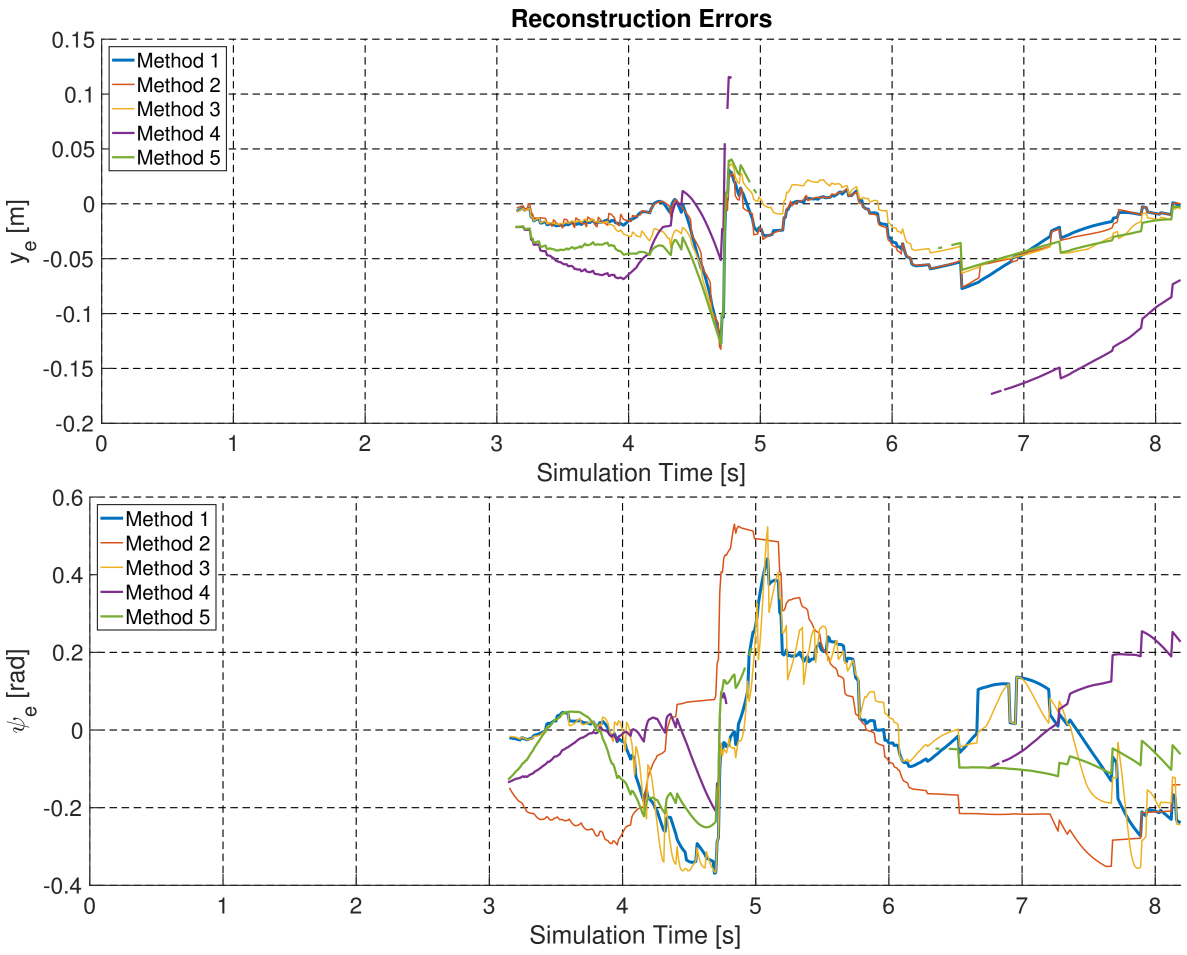


Fig. 7-8: The performance of the different path generators in the scaled vehicle experiment. The reconstruction errors are plotted as functions of simulation time. All graphs at the time instance at which the Waypoint Storage subsystem contains sufficient waypoints.

Table 7-2: Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}, using the scaled vehicle experimental data. The ground truth curvature difference is undefined, since the path curvature is not measured by the Motion Capture system.

Path Variable	Method 1	Method 2	Method 3	Method 4	Method 5	Ground Truth
max diff(y_b)	0.0077861	0.025788	0.0096984	0.0038729	0.0039941	0.082885
max diff(ψ_b)	0.12205	0.026238	0.21945	0.009781	0.011003	0.17255
max diff(κ_b)	1.612	2.6766	0.17186	0.038254	0.070535	undefined

methods. This is in accordance with the observation made based on the simulations presented in Chapter 5.

7-2 TNO Carlab

The second set of experimental data was obtained by TNO. The purpose of the executed experiment was to test different lateral control systems in real vehicles, realizing path- and vehicle-following. In the experiment, two modified Toyota Prius vehicles (depicted in Fig. 7-9), drove on the Ford Lommel test track in Belgium. One Prius leader-vehicle drove in front of the second Prius follower-vehicle. Both vehicles were equipped with various systems logging GPS, radar, camera, inertial, and wheel-speed data. The logged data is presented in Section 7-2-1 and is used to replace the Waypoint Source and Standard Sensor Systems in the custom-built simulator. Subsequently, the Waypoint Storage and Path Generation implementations are run based on this real-world experimental data. The results are presented in Section 7-2-2.

Ideally, the resulting paths (computed on board of the follower-vehicle) are compared to the ground truth GPS-data from the leader-vehicle. Unfortunately, the data logging was erroneous. As a result, the GPS-data from the leader-vehicle is not available. This makes the Ford Lommel data set sub-optimal for testing the different path generators. Currently, generating a new real vehicle data set is not a viable option, because of the large financial costs of the required Real Time Kinematic (RTK) GPS systems. Therefore, the existing data set is used in a creative way; that is, instead of comparing the resulting paths to ground truth leader-vehicle GPS data, they are compared to the center of the lane. This is further explained in Section 7-2-2.

Fig. 7-9: TNO Carlab Prius vehicles.

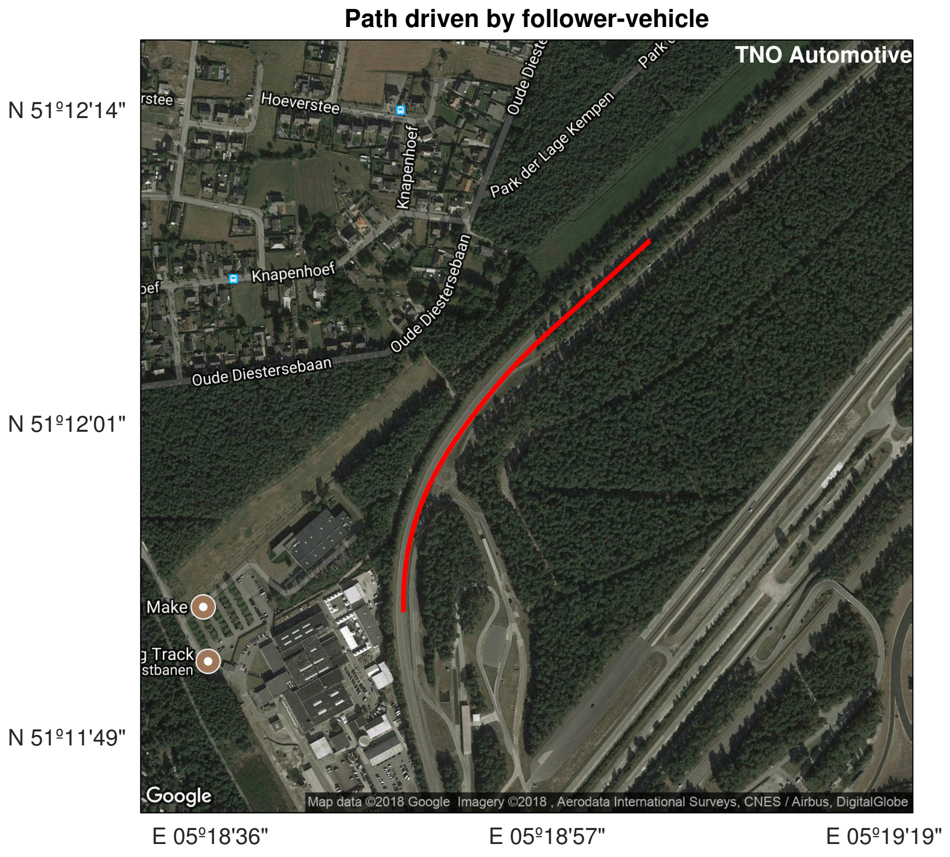


Fig. 7-10: A visualization of the path driven by the follower-vehicle, i.e., the second Toyota Prius vehicle. The figure is generated using MATLAB code by TNO.

7-2-1 Used experimental data

The path driven by the follower-vehicle is logged by an RTK GPS-system at a rate of 100 Hz. The resulting driven path is depicted in Fig. 7-10.

While driving, the onboard radar- and camera system computed a fused estimate of the relative leader-vehicle position at 10 Hz. This position data is converted into a relative (longitudinal and lateral) x- and y-position, expressed in the local follower-vehicle coordinate frame. This data is visualized in Fig. 7-11. Simultaneously, wheel-speed and inertial sensors systems logged the longitudinal velocity and yaw rate of the follower-vehicle at 100 Hz. This ego-motion parameter data is visualized in Fig. 7-12.

Additionally, the onboard Mobileye camera system was detecting the lane markings of the Ford Lommel test-track at 100 Hz. The data from this lane-marking detection is used to compare the generated paths with, which is further explained in Section 7-2-2. The camera system detects lateral offsets of the left and right lane-marking with respect to the follower-vehicle. The center of the lane can be reconstructed by computing the mean of these offsets. Similarly, estimations of the heading and curvature of the lane are obtained. This lane-marking detection data is visualized in Fig. 7-13. This finalizes the discussion of the experimental data that serves as Waypoint Source and Standard Sensor System in the path generation simulator. Based on the relative position measurements of Fig. 7-11, the Waypoint Storage subsystem

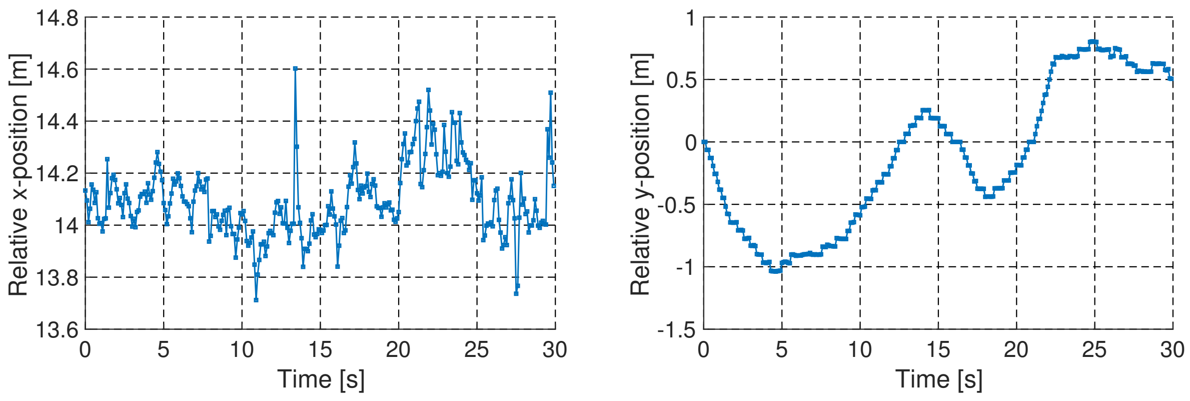


Fig. 7-11: A visualization of the logged data of the relative position measurement of the leader-vehicle with respect to the follower-vehicle.

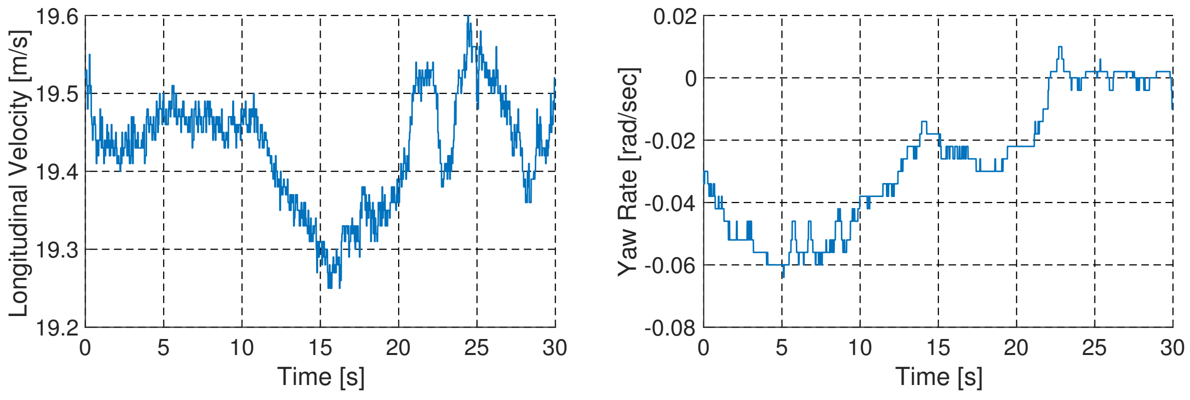


Fig. 7-12: A visualization of the logged data of longitudinal velocity and yaw rate of the follower-vehicle.

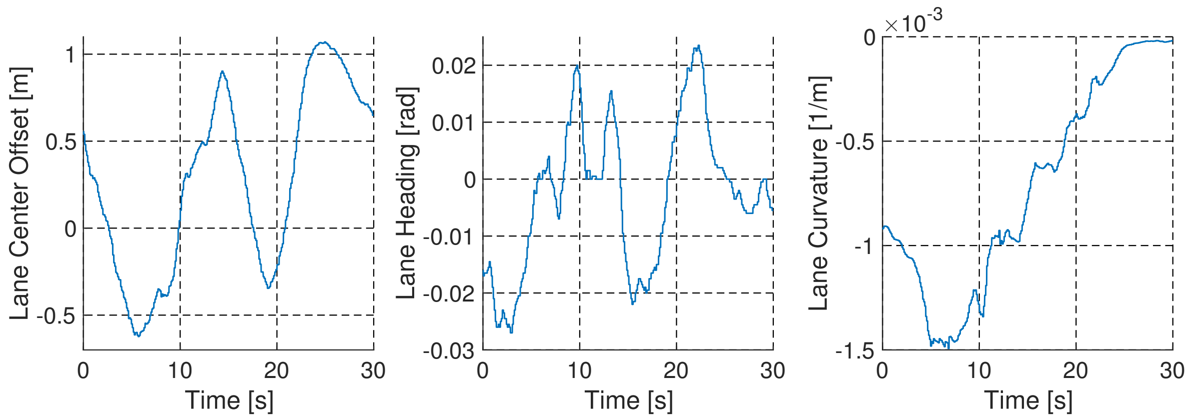


Fig. 7-13: A visualization of the logged data from the lane-marking detection system, on board of the follower-vehicle.

provides a list of waypoints. Using these waypoints, the different path generation methods are deployed, reconstructing the path driven by the leader-vehicle.

7-2-2 Results

In the new Model-Based Path Generation implementations, the single-track model parameters given in Table B-1 of Appendix B are used. Additionally, the tuning parameters reported in Section B-3 of Appendix B are used. The resulting Virtual Driver steering inputs are shown in Fig. 7-14. This figure clearly shows that Proportional Control (Method 4) provides more turbulent and less bounded steering inputs than MPC (Method 5) does. The generated paths are shown in Fig. 7-15. These paths are expressed in y_b, the lateral position of the path, expressed in the local follower-vehicle reference frame F_b. Also, the path heading ψ_b and curvature κ_b are shown. In the same figures, the path variables y_b, ψ_b, and κ_b, obtained from the lane-marking detection system, are shown: the Lane Center, Lane Heading, and Lane Curvature. These lane-marking detection variables represent the relative offset, heading, and curvature of the center of the driven lane. Based upon the comparison of the Lane Center path with the reconstructed leader-vehicle paths, it is concluded that the leader-vehicle was not driving on the center of the lane. Fig. 7-15 shows that the generated paths all lie below the Lane Center in terms of local lateral position. This indicates that the detected leader-vehicle was driving to the right of the Lane Center.

In Section 3-2, the statement was made that by Heading Constrained Polynomial Fitting (Method 2), the previously computed polynomial y_{b-1} is incorporated in the computation of the new polynomial y_b. However, this results in a new polynomial with a certain heading at $x_b = 0$, which is not necessarily the real leader-vehicle heading. It was hypothesized that Method 2 does not necessarily outperform Unconstrained Polynomial Fitting (Method 1) in terms of estimating the true path driven by a leader-vehicle. By implementation with real-world experimental data, this hypothesis is confirmed: Fig. 7-15 clearly shows a deviating polynomial heading ψ_b. Curvature Constrained Polynomial Fitting (Method 3) suffers from the same phenomenon in path variable κ_b.

Unfortunately, the results do not allow for computation of the reconstruction errors y_e, ψ_e

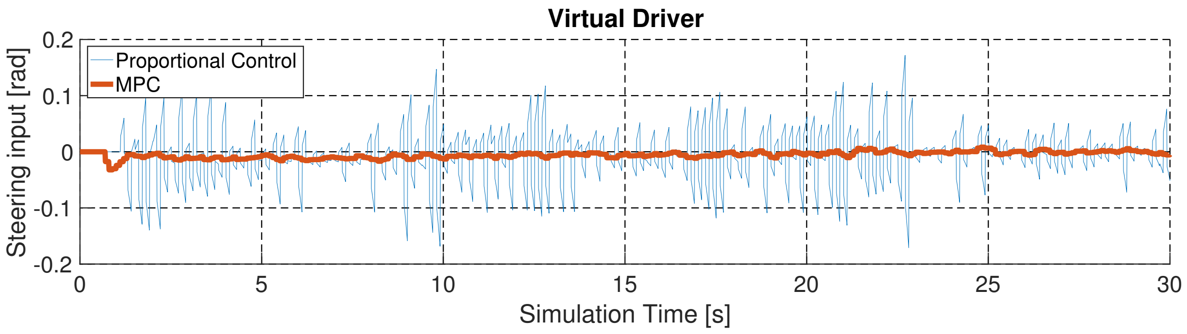


Fig. 7-14: The computed Virtual Driver steering inputs for the Virtual Leader Model are shown, which are based on the waypoints obtained using TNO's experimental data.

Table 7-3: Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}. The results are obtained by the approach of running the path generators based on TNO's experimental data.

Path Variable	Method 1	Method 2	Method 3	Method 4	Method 5
max diff(y_b)	0.020497	0.40414	0.043185	0.012709	0.011739
max diff(ψ_b)	0.0076849	0.009731	0.046576	0.0024589	0.0010102
max diff(κ_b)	0.001788	0.036164	0.0010563	0.0026842	0.0002678

or κ_e (see Section 5-1), because the ground truth GPS-data of the leader-vehicle is not available. However, the simulation with the experimental data does verify the discontinuous, sub-optimal nature of the polynomial path generation methods. Additionally, the analysis confirms that the new Model-based Path Generation implementations are capable of producing more continuous paths, compared to polynomial path generation. In order to quantify this, the differences in generated path variables y_b, ψ_b and κ_b are calculated by (5-11). The calculated differences in path variables are reported in Fig. C-2 of Appendix C. The maximum occurring differences in path variables y_b, ψ_b, and κ_b are summarized in Table 7-3. All in all, it is clear that Model-based Path Generation with MPC (Method 5) provides the smallest differences in the path variables. This is in accordance with the simulation and experimental results presented in Chapter 5 and Section 7-1-2.

7-3 Conclusions

Results from the scaled vehicle experiments have shown that paths resulting from any polynomial fitting method are discontinuous. The discontinuous character was quantified by computing the maximum occurring differences in position, heading, and curvature path variables (y_b, ψ_b, and κ_b). As an exception, the constrained polynomial fitting methods provided a more continuous path only in terms of their corresponding constrained path variable: heading constrained fitting provided deviations smaller by one order of magnitude in ψ_b, compared to unconstrained fitting. Similarly, curvature constrained fitting provided deviations smaller by one order of magnitude in κ_b, compared to unconstrained fitting. However, the heading and curvature constrained fitting methods resulted in larger reconstruction errors, compared to unconstrained polynomial fitting. In contrast to this, Model-based Path Generation showed

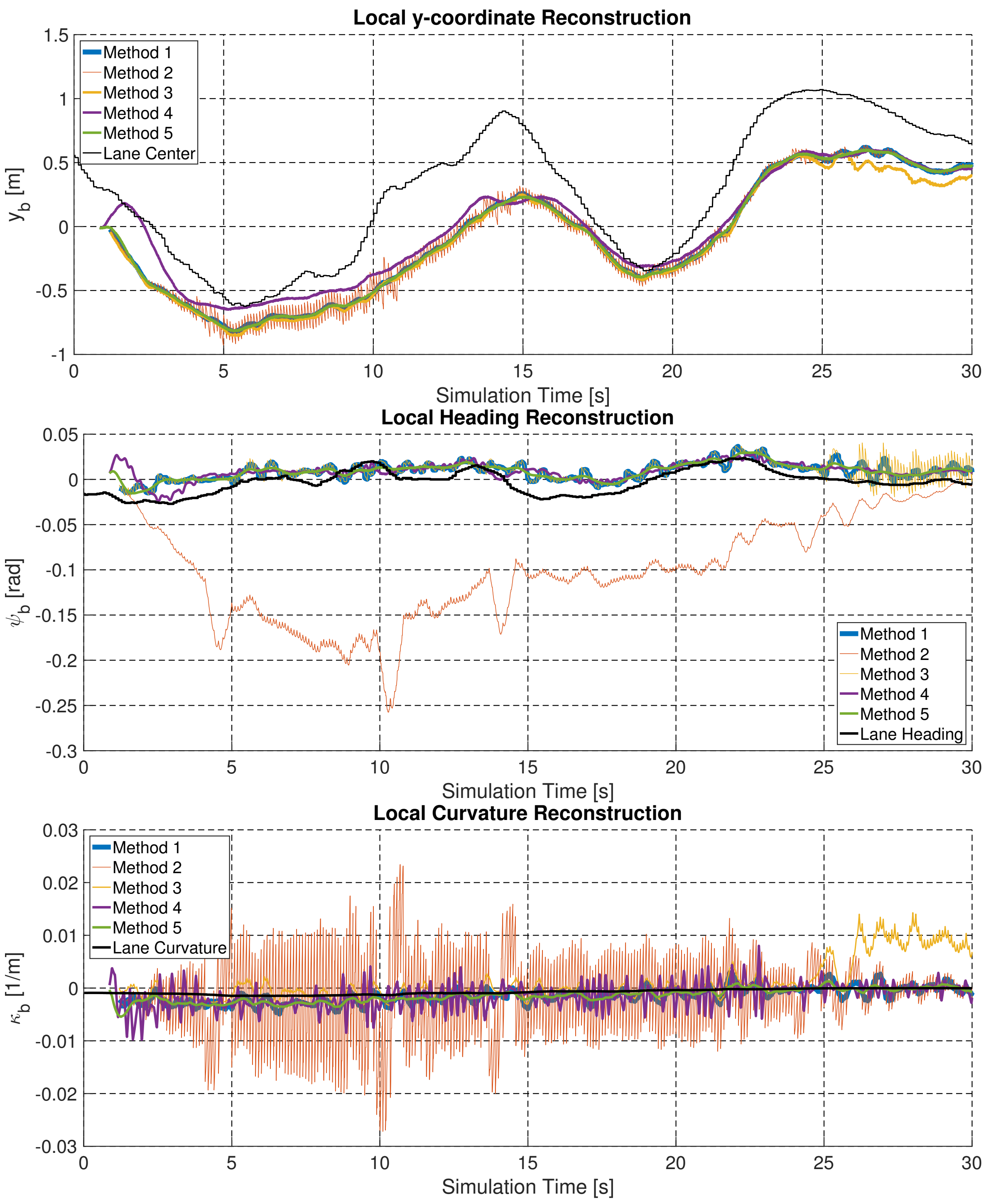


Fig. 7-15: The generated path variables y_b, ψ_b, and κ_b are shown. The results were obtained by running the path generators based on TNO's experimental data.

improved path heading reconstruction performance. Additionally, the discontinuous character of Model-based Path Generation was quantified by computing the maximum occurring difference in the path variables (y_b, ψ_b and κ_b). Model-based Path Generation with MPC, benchmarked against polynomial fitting path generation, provided a decrease by at least one order of magnitude in maximum occurring deviations in the heading and curvature path variables. From this, it can be concluded that the new Model-based Path Generation method outperforms the benchmark method in terms of path continuity. These conclusions are in accordance with the conclusions drawn in Chapter 5.

Unfortunately, the results from the experimental analysis using the TNO Carlab data do not allow for conclusions to be drawn regarding the path reconstruction performance. This is because the ground truth GPS-data of the leader-vehicle is not available. However, the experimental analysis does verify the discontinuous, sub-optimal nature of the polynomial fitting path generation methods. Additionally, the results have shown that the Model-based Path Generation methods outperform the benchmark method in terms of path continuity. This is in accordance with the conclusions drawn from the scaled vehicle experimental analysis.

Altogether, it is concluded that the experimental analysis substantiates the outperforming abilities of Model-based Path Generation, in terms of path continuity and path heading accurateness, compared to polynomial fitting path generation.

Chapter 8

Conclusions and Recommendations

In Chapter 1, the main path generation problem was defined. Through Chapters 2 to 7, this main problem was solved. This final chapter presents the corresponding conclusions in Section 8-1. Subsequently, this thesis is concluded in Section 8-2 by providing various recommendations for future research.

8-1 Conclusions

This research addressed the development of a generic, robust reference path generator for lateral vehicle control. TNO's state-of-the-art polynomial fitting path generator was used as a benchmark method.

The main issue at hand was that the state-of-the-art polynomial fitting path generation solution does not provide desirable results. As described, this is inherently caused by its two main weaknesses. Firstly, the current system is not robust to sensor noise and other real-world disturbances. Secondly, the current system is based on repetitive least-squares polynomial fitting, resulting in discontinuous path generation. This is undesirable, because it leads to an unfeasible vehicle path, since vehicles can only produce and track continuous trajectories. Moreover, a discontinuous reference path results in the need for path smoothing when used in comfortable lateral vehicle control. Fundamentally, this smoothing leads to inaccurate vehicle control, caused by manipulation of the original, discontinuous reference path. These problems led to the definition of the main research question, addressed in this thesis:

How can a feasible, continuous vehicle reference path be generated, based on a set of given waypoints?

The goal of this research was to develop a generic, robust reference path generator for lateral vehicle control. This path generator must meet a number of minimal requirements. Firstly, it must provide an accurate and feasible path. Secondly, it must be robust in the sense that is has proper disturbance rejection characteristics. Finally, the resulting path must be continuous to allow for accurate and comfortable lateral vehicle control. The preceding chapters addressed the main research question, contributing to achieving the research goal. The resulting conclusions are summarized in the next sections.

8-1-1 Field of Application

Path generation is deployed to obtain a smooth and continuous path, based on given waypoints. The waypoints can originate from various sources. For instance, in automated vehicle-following the Waypoint Source is a system that detects a leader-vehicle. Alternatively, the waypoints may be defined by a human operator or originate from a strategic path planning layer. Fundamentally, waypoints obtained from sensor systems are noise-corrupted. Therefore, a generic path generation system must have noise-rejecting characteristics. In this way, the path generation system is applicable irrespective of the specific Waypoint Source.

Within this research, the path generation problem was solved for the generic vehicle-following field of application. This field of application resulted in a general situation with a continuously changing set of waypoints and a continuously changing, local reference frame. Consequently, the path generation solution is also applicable in situations where the set of waypoints and reference frame are stationary. Moreover, in this generic field of application the generated path was interpreted as a reconstruction of the leader-vehicle's factual path. As a consequence, path generation performance can be evaluated, since it was related to the accuracy of reconstruction. This approach did not impose any limitations on the system; that is, the applicability remained irrespective of the specific Waypoint Source.

In an ideal situation, the input waypoints are precise and accurate. However, in reality they may be uncertain or corrupted with sensor noise. Next to Waypoint Noise, a path generation system is subject to other real-world disturbances; namely, Waypoint Delay, Waypoint Offset, and Motion Parameter Measurement Noise.

8-1-2 Polynomial Path Generation

From theoretical analysis, it was concluded that the repetitive polynomial fitting methods produce discontinuous paths, which is inherent to the function fitting approach. Firstly, the discontinuous character of the paths makes them fundamentally unfeasible, since vehicles can only produce and track continuous trajectories. Moreover, a discontinuous reference path results in the need for path smoothing when used in comfortable lateral vehicle control. This smoothing leads to inaccurate vehicle control, caused by manipulation of the original, discontinuous reference path. To mitigate these drawbacks, modified polynomial path generation methods were proposed, which resulted in a total of three different polynomial fitting methods. The first method was an unmodified version of the unconstrained benchmark polynomial path fitting method. The other two methods were based on constrained polynomial fitting. Heading constrained and curvature constrained polynomial fitting methods were presented. However, when the constrained fitting methods are applied in vehicle-following, the resulting paths can obtain a heading or curvature, which is not necessarily the correct, leader-vehicle's factual heading or curvature. Therefore, it was hypothesized that the constrained fitting methods are superior in terms of providing continuous paths, but not necessarily superior in reconstructing the factual path driven by a leader-vehicle. Moreover, polynomial fitting methods require tuning of parameters that have indirect connections with vehicle dynamics. In other words, the tuning process attempts to model the vehicle and its trajectory characteristics with tuning parameters that are not intuitive. All in all, it was concluded that a polynomial fitting path generation approach is fundamentally sub-optimal.

8-1-3 Model-based Path Generation

As an alternative to polynomial fitting, a new Model-based Path Generation method was presented. This new method is based on modeling a virtual leader-vehicle: the Virtual Leader Model. The steering inputs for this Virtual Leader Model are given by a control system: the Virtual Driver. Two different control implementations of the Virtual Driver were presented. The first implementation was given by Proportional Control, using a virtual look-ahead distance. Based on given waypoints at this look-ahead-distance, the control actions are calculated. The second Virtual Driver implementation was based on the Model Predictive Control (MPC) framework. Using model predictions, an optimal control input sequence is computed based on real-time optimization of an objective function. Consequently, the reference path is obtained by extracting it from the Virtual Leader Model state history. By design, the Model-based Path Generation methodology results in continuous paths that are compatible with vehicle dynamics. Moreover, the resulting path can be tuned by intuitive tuning parameters.

8-1-4 Simulations and Experiments

Simulation and experimental results have shown that paths resulting from polynomial fitting methods are discontinuous. The discontinuous character was quantified by computing the maximum occurring differences in position, heading, and curvature path variables (y_b, ψ_b, and κ_b). As an exception, the constrained polynomial fitting methods provided a more continuous path only in terms of their corresponding constrained path variable: heading constrained fitting provided deviations smaller by one order of magnitude in ψ_b, compared to unconstrained fitting. Similarly, curvature constrained fitting provided deviations smaller by one order of magnitude smaller in κ_b, compared to unconstrained fitting. However, the heading and curvature constrained fitting methods resulted in larger reconstruction errors when compared to unconstrained polynomial fitting, confirming the hypotheses regarding constrained polynomial fitting. In contrast to this, the new Model-based Path Generation method, with MPC Virtual Driver, showed improved path reconstruction performance. Results have shown that the maximum occurring position, heading, and curvature reconstruction errors (y_e, ψ_e, and κ_e) were significantly reduced, compared to polynomial path generation. From this, it was concluded that Model-based Path Generation with MPC outperforms the benchmark method in terms of accurateness.

Also for Model-based Path generation, the discontinuous character was quantified by computing the maximum occurring differences in the path variables (y_b, ψ_b, and κ_b). Model-based Path Generation with MPC, benchmarked against polynomial fitting path generation, provided a decrease by at least one order of magnitude in maximum occurring deviations in the path variables. Experimental analyses resulted in the same observation. From this, it was concluded that the new Model-based Path Generation method also outperforms the benchmark method in terms of path continuity.

In addition, the new Model-based Path Generation method successfully processed user-defined waypoints representing an extreme lane-change maneuver. A smooth reference path was realized, compatible with the vehicle dynamics. In contrast to this, the polynomial fitting methods resulted in discontinuous, unfeasible paths. This demonstrated the new method's applicability for path generation in general: it is capable of producing feasible trajectories based on unfeasible waypoints, irrespective of the specific Waypoint Source. Moreover, it showed the new method's robustness against unfeasible waypoints in vehicle-following. Additionally, results from the sensitivity analysis have shown that Model-based Path Generation also outperforms the benchmark method in terms of robustness against real-world disturbances.

Furthermore, results have indicated that all proposed methods are suitable for real-time implementation. This conclusion was based on the fact that the required computation time is less than the simulated driving time.

In summary, it was concluded that the new Model-based Path Generation method outperforms polynomial fitting path generation in terms of accurateness, robustness, continuity, and general applicability. With this, the goal of this research was achieved: a generic, robust reference path generator for lateral vehicle control is developed.

8-2 Recommendations

The first recommendation for future research is to design variations to the proposed MPC framework, incorporating different types of objective functions. Currently, optimization of the proposed objective function results in minimizing future offsets from given waypoints. Variations or extensions to this objective function can be designed. For instance, by incorporating knowledge on the uncertainty of the waypoints. In this way, the control objective of not exactly tracking the noise-corrupted waypoints can be incorporated in the design phase, rather than obtaining the desired noise-rejecting characteristics by tuning the prediction and control horizons. Alternatively, passenger comfort can be incorporated by including comfort criteria in the objective function.

Regarding the continuity of the resulting path, Model-based Path Generation using MPC showed significant improvement compared to polynomial fitting path generation. Clearly, further continuity improvement is limited by the fact that the path generator is implemented on a discrete-time computer. As a result, further continuity improvement can be achieved by increasing the relevant sampling frequencies. Results regarding the computational demand have shown that there is sufficient margin for this, considering the speed of the current implementation. Therefore, the second recommendation for future research is to explore the feasibility of improving path continuity by increasing relevant MPC frequencies. For instance, the model prediction and optimization frequencies can be increased.

Currently, only the position states from the Virtual Leader Model are used for path generation. Therefore, a third recommendation for future research is to explore the feasibility of improving the path generation functionality by using the rest of the Virtual Leader Model states. For instance, the heading angle state can be used. In vehicle-following, this angle is a reconstruction of the leader-vehicle's factual heading angle. Potentially, this enables the use of the heading estimate to compensate for the disturbance induced by the leader-vehicle Center of Gravity (CoG)-detection mismatch (Waypoint Offset).

Finally, the fourth recommendation for future research is to implement and test the new method in various fields of application. In vehicle-following, the method can be tested by using ground-truth path information when available. Additionally, it can be used for world-modeling in general. Road users can be modeled using the proposed Model-based Path Generation framework. In this way, computer-simulated virtual versions of the road users will be obtained. Using these virtual versions, past road-user trajectories can be modeled and future trajectories can be predicted. Subsequently, the obtained information can be used by decision-making units or collision avoidance systems. In addition, the new method can be used to model road lane-markings. To this end, lane-marking position information from camera systems can be used as input waypoints. With these waypoints, a virtual vehicle driving on the lane-markings can be simulated. Consequently, the lane-marking path is given by the trajectory of the virtual vehicle. This lane-marking path can be used as reference path to realize automated lane-keeping.

Appendix A

Waypoint Source Modeling

A path generation system consists of three main subsystems. The hierarchy of the subsystems was visualized in Fig. 2-1. This appendix presents the mathematical simulation models for the Waypoint Source in the generic vehicle-following field of application.

The mathematical models form the basis of the custom-built simulator. This simulator is used to emulate the path generation functionality of different path generators. With simulations, the performance, robustness and continuity of the path generators can be evaluated. The design and the results of this evaluation are given in Chapters 5 and 6.

In the generic vehicle-following field of application, the Waypoint Source makes use of the assumed Standard Sensor Systems. These systems provide the relative position of a leader-vehicle and the ego-motion parameters of the follower-vehicle, as explained in Section 2-1 and illustrated in Fig. 2-2. First the sensor models are explained in Section A-1. To be able to simulate a real-world scenario, the mathematical models for the real-world disturbances must be defined. These disturbances were introduced in Section 2-3. The corresponding mathematical models for simulation are defined in Section A-2.

A-1 Sensor Models

In the custom-built simulator, the Standard Sensor Systems provide the relative position of the leader-vehicle and the ego-motion parameters of the follower-vehicle. These outputs are obtained by simulating two separate vehicles and extracting their internal vehicle states. To model the leader- and follower-vehicle, single-track bicycle models are used [18]. The model equations are given by:

$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{(C_f+C_r)}{(m\cdot u)} & \frac{(l_rC_r-l_fC_f)}{(m\cdot u)} - u \\ \frac{(l_rC_r-l_fC_f)}{(I_zu)} & -\frac{(l_f^2C_f+l_r^2C_r)}{(I_zu)} \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} + \begin{bmatrix} \frac{C_f}{m} \\ \frac{l_fC_f}{I_z} \end{bmatrix} \cdot \delta, \tag{A-1}$$

where C_f and C_r are front and rear cornering stiffness respectively, m is the total vehicle mass, l_f and l_r the distances from the Center of Gravity (CoG) of the vehicle to the front

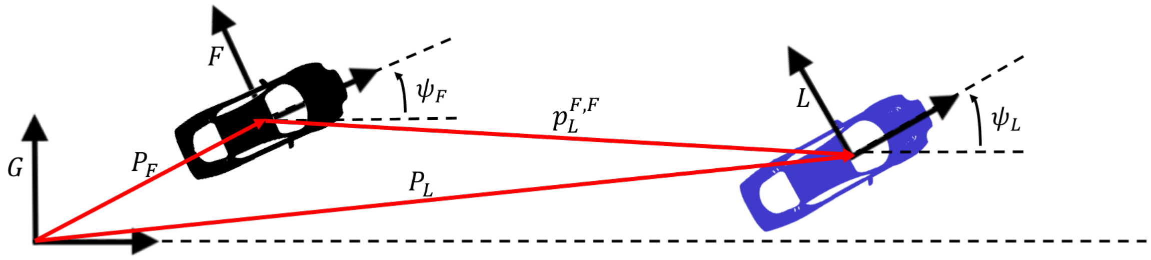


Fig. A-1: This figure visualizes the black follower-vehicle and the blue leader-vehicle, with their local reference frames F and L respectively. The position of the frames, with respect to the global frame G, is given by P_F and P_L. The orientation of the vehicles, with respect to the global frame G, is given by ψ_F and ψ_L. The relative position of the leader-vehicle, with respect to the follower-vehicle, expressed in frame F is denoted by $p_L^{F,F}$.

and rear wheel-position of the vehicle respectively. Generally, distances l_f and l_r are not equal for passenger cars or trucks, because of an uneven mass distribution. Furthermore, I_z is the mass moment of inertia around the vertical axis through the CoG. Moreover, u is the longitudinal forward velocity, v the lateral velocity and r the yaw rate of the vehicle. Finally, δ is the input to the model, representing the steering angle of the front wheel. This single-track model assumes constant forward velocity, no longitudinal or lateral load transfer, linear range tires, no roll, pitch or vertical motion, no suspension and no compliance effects [21]. These assumptions are justified, since the custom-built simulator will be used for the normal driving regime.

The position of the leader-vehicle in a global reference frame can be obtained by mapping and integrating the local leader-vehicle longitudinal and lateral velocities u_l and v_l. For the follower-vehicle, the global position is obtained in the same way: by mapping and integrating the local follower-vehicle longitudinal and lateral velocities u_f and v_f. From now on, the subscripts l and f are dropped for variables in equations that hold for both the leader- and the follower-vehicle.

To obtain the global vehicle positions, first the model state r is integrated to obtain the relative vehicle heading $\psi(t)$, expressed in a global reference frame G:

$$\psi(t) = \int_0^t r(t) \cdot dt + \psi_0, \tag{A-2}$$

where ψ_0 is the initial heading of the vehicle model. The global reference frame G, follower-vehicle reference frame F, and leader-vehicle reference frame L are visualized in Fig. A-1. Subscripts l or f denote leader-vehicle or follower-vehicle variables respectively.

Consequently, the local velocities u and v are transformed to obtain the vehicle velocity vector V in the global reference frame G:

$$\mathbf{V}(t) = \begin{bmatrix} \cos(\psi(t)) & -\sin(\psi(t)) \\ \sin(\psi(t)) & \cos(\psi(t)) \end{bmatrix} \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}. \tag{A-3}$$

The global position $\mathbf{P}$ of a simulated vehicle is then obtained by integrating this global velocity vector:

$$\mathbf{P}(t) = \int \mathbf{V}(t) + \mathbf{P}_0, \tag{A-4}$$

where $\mathbf{P}_0$ is the initial position of the vehicle. Consequently, let $\mathbf{P}^*_L(t)$ and $\mathbf{P}_F(t)$ denote the global position of the leader- and follower-vehicle respectively. The reason behind the asterisk superscript for the global position of the leader-vehicle will become clear in Section A-2-3. Let $\mathbf{p}_L^{G,F}(t)$ denote the position of frame L, with respect to frame F, expressed in frame G. This is the relative position of the leader-vehicle. This relative position is obtained by:

$$\mathbf{p}_L^{G,F}(t) = \mathbf{P}^*_L(t) - \mathbf{P}_F(t). \tag{A-5}$$

Finally, the relative position of the leader-vehicle expressed in the local follower-vehicle reference frame is denoted by $\mathbf{p}_L^{F,F}$. This relative position is visualized in Fig. A-1. It is given by:

$$\mathbf{p}_L^{F,F}(t) = \begin{bmatrix} \cos(\psi_F(t)) & \sin(\psi_F(t)) \\ -\sin(\psi_F(t)) & \cos(\psi_F(t)) \end{bmatrix} \mathbf{p}_L^{G,F}(t), \tag{A-6}$$

where $\psi_F(t)$ denotes the heading angle of the follower-vehicle with respect to the global reference frame. Note that this transformation matrix is the transpose of the matrix used to map the velocity vector from the local vehicle-attached reference frame to the global reference frame. This finalizes the mathematical framework of the Standard Sensor Systems for the ideal world. The resulting outputs are the ego-motion parameters (u, v and r) of the follower vehicle and the position measurements of the leader-vehicle with respect to the follower-vehicle, given by $\mathbf{p}_L^{F,F}$. Additionally, to simulate a real-world scenario, disturbances must be incorporated. These disturbances were introduced in Section 2-3. The corresponding mathematical models are defined in Section A-2.

A-2 Disturbance models

As defined in Section 2-3, there are four relevant real-world disturbances to consider. The first three disturbances are related to the relative position measurements $\mathbf{p}_L^{F,F}(t)$. These measurements will be referred to as waypoints. The disturbances are caused by Waypoint Noise, Delay and Offset. These disturbances are visualized in Fig. 2-5. The fourth disturbance is caused by noise on the motion parameters u, v and r. The expected magnitude of the different disturbances are obtained from analyzing data from TNO's Carlab. The variances are given in Table A-1. How these disturbance variances are used in the simulator, will be explained in the next sections.

A-2-1 Waypoint Noise

The Waypoint Source gives position measurements of a preceding leader-vehicle. For simulation purposes, the uncertainty or noise on these measurements can be modeled by a zero-mean Gaussian distribution with a variance in Cartesian coordinates x and y of the local follower-vehicle reference frame, given by σ_x^2 and σ_y^2 respectively.

A-2-2 Waypoint Delay

The Waypoint Source provides the waypoints with a certain time delay. In simulation, this time delay is easily implemented by holding the measurement signal value for a time equal to

Disturbance	**Value**	**Unit**
σ_x^2	0.0044	m
σ_y^2	0.0278	m
t_{delay}	0.210	s
l_r	4.0	m
σ_u^2	4.0000e-04	$\mathrm{m\,s^{-1}}$
σ_β^2	3.3846e-05	rad
σ_r^2	4.4444e-05	$\mathrm{rad\,s^{-1}}$

Table A-1: Expected real-world disturbance values.

the estimated time delay of the Waypoint Source:

$$\mathbf{p}_{delayed}(t) = \mathbf{p}_L^{F,F}(t - t_{delay}), \tag{A-7}$$

where $\mathbf{p}_L^{F,F}$ is the undelayed waypoint, obtained from the sensor models of Section A-1. Furthermore, t_{delay} is the estimated time delay of the sensor systems and $\mathbf{p}_{delayed}(t)$ the delayed waypoint that is available as input for the path generation system. Based on TNO's Carlab, t_{delay} is estimated to be approximately 210 milliseconds [9]. Theoretically, when the Waypoint Delay is properly compensated for, the induced disturbance can be fully eliminated. This is verified by numerical simulations in Chapter 6. In practice, the degree of this disturbance elimination depends on the accuracy of the estimate of t_{delay}.

A-2-3 Waypoint Offset

As discussed in Section 2-3, the path generation system can only be based on the relative position measurements of the rear of the leader-vehicle, considering the vehicle-following field of application. As a result, the reconstructed path will be that of the rear, rather than the CoG of the leader-vehicle. To model this Waypoint Offset in the simulator, the global leader-vehicle position $\mathbf{P}_L(t)$ is modified, prior to being used in the equations of Section A-1. The modification is given by:

$$\mathbf{P}_L^*(t) = \mathbf{P}_L(t) - \begin{bmatrix} l_r \cdot \cos(\psi_L(t)) \\ l_r \cdot \sin(\psi_L(t)) \end{bmatrix}, \tag{A-8}$$

where $\psi_L(t)$ represents the heading of the leader-vehicle with respect to the global reference frame. Moreover, $\mathbf{P}_L(t)$ is the position of the CoG of the leader-vehicle, and $\mathbf{P}_L^*(t)$ is the position of the rear of the leader-vehicle, both expressed in the global reference frame. The CoG-offset is denoted by l_r. This is the distance from the rear to the CoG of the leader-vehicle.

A-2-4 Motion Parameter Noise

In practice, the ego-motion parameters of the follower-vehicle are obtained from different sensors. Wheel-speed sensors provide the longitudinal velocity u of the vehicle. An Inertial

Measurement Unit (IMU) provides the yaw rate r of the vehicle. The lateral velocity v can be estimated [23]. The noise introduced by the wheel-speed sensors and IMU are given by the variances σ_u^2 and σ_r^2 respectively. The lateral velocity v uncertainty is given in terms of the estimation variance of the body-slip angle σ_β^2, where the body-slip angle is given by:

$$\beta = \tan^{-1}(\frac{v}{u}). \tag{A-9}$$

Similar to the waypoint noise, the noise on the motion parameter measurements can be modeled by considering additive zero-mean Gaussian noise with variances of σ_u^2, σ_β^2 and σ_r^2.

This finalizes the mathematical framework of the Waypoint Source for path generation, including models of the four real-world disturbances.

Appendix B

Simulation Parameters & Results

B-1 Bicycle Model Parameters

Parameter	**Symbol**	**Unit**	**Value**
Wheelbase	L	m	2.89
Distance front axle to CoG	l_f	m	1.48
Distance rear axle to CoG	l_r	m	1.41
Total mass	m	kg	1900
Body inertia around vertical axis	I_z	$\mathrm{kg\,m^2}$	3500
Front cornering stiffness	C_f	$\mathrm{N\,rad^{-1}}$	120000
Rear cornering stiffness	C_r	$\mathrm{N\,rad^{-1}}$	190000
Longitudinal vehicle velocity	V	$\mathrm{m\,s^{-1}}$	27.78

Table B-1: Bicycle model parameters used for the vehicles in the custom-built simulator.

Parameter	Symbol	Unit	Value
Wheelbase	L	m	2.89
Distance front axle to CoG	l_f	m	1.48
Distance rear axle to CoG	l_r	m	1.41
Total mass	m	kg	1900
Body inertia around vertical axis	I_z	$\mathrm{kg\,m^2}$	3500
Front cornering stiffness	C_f	$\mathrm{N\,rad^{-1}}$	120000
Rear cornering stiffness	C_r	$\mathrm{N\,rad^{-1}}$	190000
Longitudinal vehicle velocity	V	$\mathrm{m\,s^{-1}}$	27.78

Table B-2: Virtual Leader Model single-track parameters used in the scaled vehicle experiments. These values are obtained by system identification performed by Mart Baars for the DSV.

B-2 Steering Input

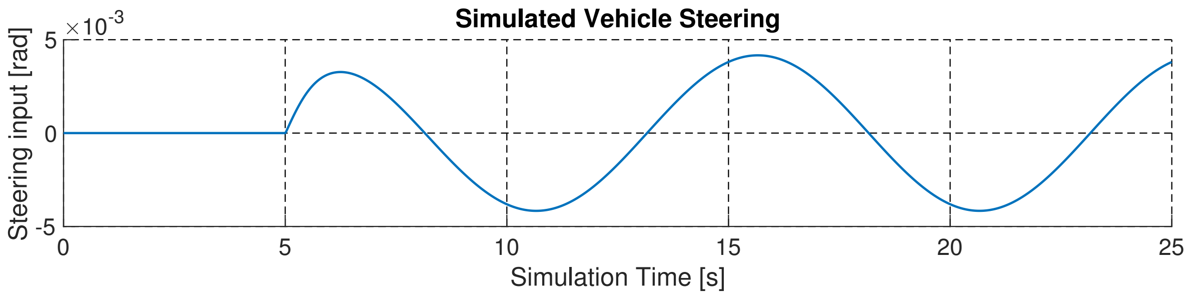


Fig. B-1: This figure visualizes the steering input given in the simulation scenarios. For the leader-vehicle, this input is mirrored in the horizontal axis, to let it initiate a lane-change to the right, rather than to the left.

B-3 Path Generation Implementation Parameters

Polynomial Path Generation

For unconstrained polynomial fitting, one tuning parameter is available: the number of used waypoints. For all polynomial fitting methods, the number of used waypoints is set to 9. This number is chosen because using this amount of waypoints lead to the best possible path reconstruction by the resulting polynomials, in terms of maximum occurring reconstruction errors. With constrained polynomial fitting, another tuning parameter is available: ΔX_b. In theory, explained in Section 3-2, this parameter should represent the traveled origin distance of the local follower-vehicle reference frame, during the time between two consecutive polynomial fitting steps. In practice, it can be used to tune the constrained polynomial fitting behavior. For the simulations, $\Delta X_b = \int u \cdot dt$ is chosen, in correspondence with theory. The reason for this is given by the conclusion drawn at the end of Chapter 3: the polynomial fitting methods require tuning of parameters that have indirect connections to the vehicle dynamics. The tuning process comes down to trying to model the leader-vehicle and its trajectory characteristics, but with tuning parameters that are not intuitive.

Model-based Path Generation

For Model-based Path Generation, more tuning parameters are available. The effects of these intuitive parameters were theoretically substantiated in Sections 4-3-2 and 4-5. For the model-predictions, the Virtual Leader Model of Section 4-2 is used. This model is integrated using the classical Runge-Kutta method (RK4) with a sampling time of $T_i = 0.01\,\mathrm{s}$. With $T_m = 0.1\,\mathrm{s}$, this defines the sampling time ratio q to be 10, given by (4-17). For the Virtual Leader Model parameters, the bicycle model parameters of Table B-1 are used. For the Model Predictive Control (MPC) optimization, δ_{max} and δ_{min} are set to $0.1\,\mathrm{rad}$ and $-0.1\,\mathrm{rad}$ respectively. Moreover, $\dot{\delta}_{max}$ and $\dot{\delta}_{min}$ are set to $0.175\,\mathrm{rad\,s^{-1}}$ and $-0.175\,\mathrm{rad\,s^{-1}}$ respectively. Note that these values define the allowed steering behavior of the Virtual Driver. This steering behavior can be modified by defining different bounds, influencing the resulting Virtual Leader Model's trajectory. The aforementioned values for the bounds are chosen because, in combination with the other tuning parameters, they result in superior leader-vehicle path reconstruction. This was shown by the simulation results.

In Section 4-3-2, the look-ahead-time t_{la} was introduced as tuning parameter for the Proportional Control implementation for the Virtual Driver. For all scenarios, a fixed look-ahead-time t_{la} of $0.9\,\mathrm{s}$ is chosen. The intuition behind this is given by how a human would perform the lane-change: by anticipating on the future, making forward predictions in the same order of magnitude as the chosen value for t_{la}. The Virtual Leader Model drives at a certain time headway t_v with respect to the real leader-vehicle. This time headway is defined by setting a prediction horizon N for the Virtual Leader Model. This prediction horizon N corresponds to the amount of waypoints from the Waypoint Source that are available to the virtual leader-vehicle. These waypoints are located between the virtual leader-vehicle and the real leader-vehicle and lead to a certain time headway t_v.

In Section 4-5, the MPC tuning parameters N, N_m and N_c were defined. For all scenarios, the minimum cost horizon N_m is set to 1, making all offsets y_e^b contribute to the objective function. Moreover, the prediction horizon N is set to 10. This means that the virtual leader-vehicle uses 10 waypoints in front of it to determine its control actions. A Waypoint Source, sampling waypoints at 10 Hz, in combination with a vehicle speed of $100\,\mathrm{km\,h^{-1}}$, results in a distance between each waypoint of approximately 2.78 m. Consequently, $N = 10$ corresponds to a distance 27.8 m.

The aforementioned values of t_{la} and N are chosen based on how a human would perform the vehicle-following: by anticipating on the future, making forward predictions in the same order of magnitude. In the simulated scenarios, the set value for N realizes that the Virtual Leader Model drives between the follower- and leader-vehicle. This way, the generated reference path is obtained in front of the follower-vehicle. As a result, the reference path can potentially be used by a control system on board of the follower-vehicle.

For ideal-world scenarios, the control objective is given by the goal of tracking the given waypoints accurately. Therefore, the control horizon for these scenarios is set to the maximum ($N_c = N$), realizing a Virtual Driver that effectively tracks the waypoints. For real-world scenarios, where real-world waypoint disturbances are introduced, the control objective is given by the goal of tracking the given waypoints in a more conservative way, realizing an accurate reconstruction of the path driven by the real leader-vehicle. This change of objective is implemented by reducing the control horizon N_c. This leads to the Virtual Driver realizing a smooth, feasible vehicle trajectory, based on the noise-corrupted or unfeasible waypoints. Besides the filtering effect, a reduced control horizon N_c gives the advantage of reducing the dimension of the optimization search-region, leading to a smaller computational demand. For these reasons, in simulating real-world scenarios, a control horizon $N_c = 1$ is chosen. The same control horizon is used for simulating the scenario with user-defined waypoints, which was explained in Section 5-3-3.

B-4 Results

B-4-1 Reconstruction errors

Table B-3: Maximum occurring reconstruction errors with y_e in m, ψ_e in rad and κ_e in m^{-1}.

(a) Ideal World - Leader Sine Driving

Error	Method 1	Method 2	Method 3	Method 4	Method 5
max y_e	0.00018793	0.00030355	0.00010095	0.037904	0.00030395
max ψ_e	4.0285e-05	0.00026335	5.0621e-05	0.0014985	6.2451e-05
max κ_e	2.8941e-05	3.4498e-05	2.5556e-05	9.919e-05	3.1993e-05

(b) Ideal World - Follower Sine Driving

Error	Method 1	Method 2	Method 3	Method 4	Method 5
max y_e	0.0019225	0.032398	0.0019586	0.0054849	0.0022609
max ψ_e	0.00012225	0.040807	0.00012326	0.00027088	0.00019536
max κ_e	1.597e-05	0.0021153	1.7262e-05	1.6809e-05	3.7281e-05

(c) Real World - Leader Sine Driving

Error	Method 1	Method 2	Method 3	Method 4	Method 5
max y_e	0.24057	0.23982	0.29136	0.1001	0.097159
max ψ_e	0.048872	0.027011	0.057649	0.0075581	0.0084076
max κ_e	0.0082191	0.006687	0.0097673	0.0015352	0.00088326

(d) Real World - Follower Sine Driving

Error	Method 1	Method 2	Method 3	Method 4	Method 5
max y_e	0.2449	0.25863	0.25968	0.10821	0.10532
max ψ_e	0.048872	0.051299	0.05733	0.0066565	0.0079502
max κ_e	0.0082191	0.0073581	0.0097673	0.0016034	0.00083815

B-4-2 Path Discontinuity

Table B-4: Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}. The simulation is given by the real-world, leader sine maneuvering scenario (simulation time 15 to 20 seconds). Comparing Method 5 to Method 1, for lateral position (y_b), the maximum occurring jump is smaller by one order of magnitude. For the heading (ψ_b) and curvature (κ_b), the decrease in discontinuity is by two orders of magnitude.

Path Variable	Method 1	Method 2	Method 3	Method 4	Method 5	Ground Truth
max diff(y_b)	0.13105	0.12832	0.0612	0.013341	0.013471	0.010306
max diff(ψ_b)	0.034362	0.0013938	0.050691	0.0005101	0.00053334	0.00023319
max diff(κ_b)	0.0081176	0.0079337	0.00083033	0.00034089	4.9128e-05	5.2789e-06

Table B-5: Table of maximum occurring differences in path generation variables with y_b in m, ψ_b in rad and κ_b in m^{-1}. The simulation is given by the extreme maneuver scenario with user-defined waypoints. Comparing Method 5 to Method 1, for lateral position (y_b), the maximum occurring jump is smaller by one order of magnitude. For the heading (ψ_b) and curvature (κ_b), the decrease in discontinuity is by three orders of magnitude.

Path Variable	Method 1	Method 2	Method 3	Method 4	Method 5
max diff(y_b)	0.25778	0.28298	0.20868	0.012774	0.010676
max diff(ψ_b)	0.11778	0.010437	0.16484	0.00044154	0.00051211
max diff(κ_b)	0.015257	0.019715	0.0046075	7.004e-05	6.6206e-05

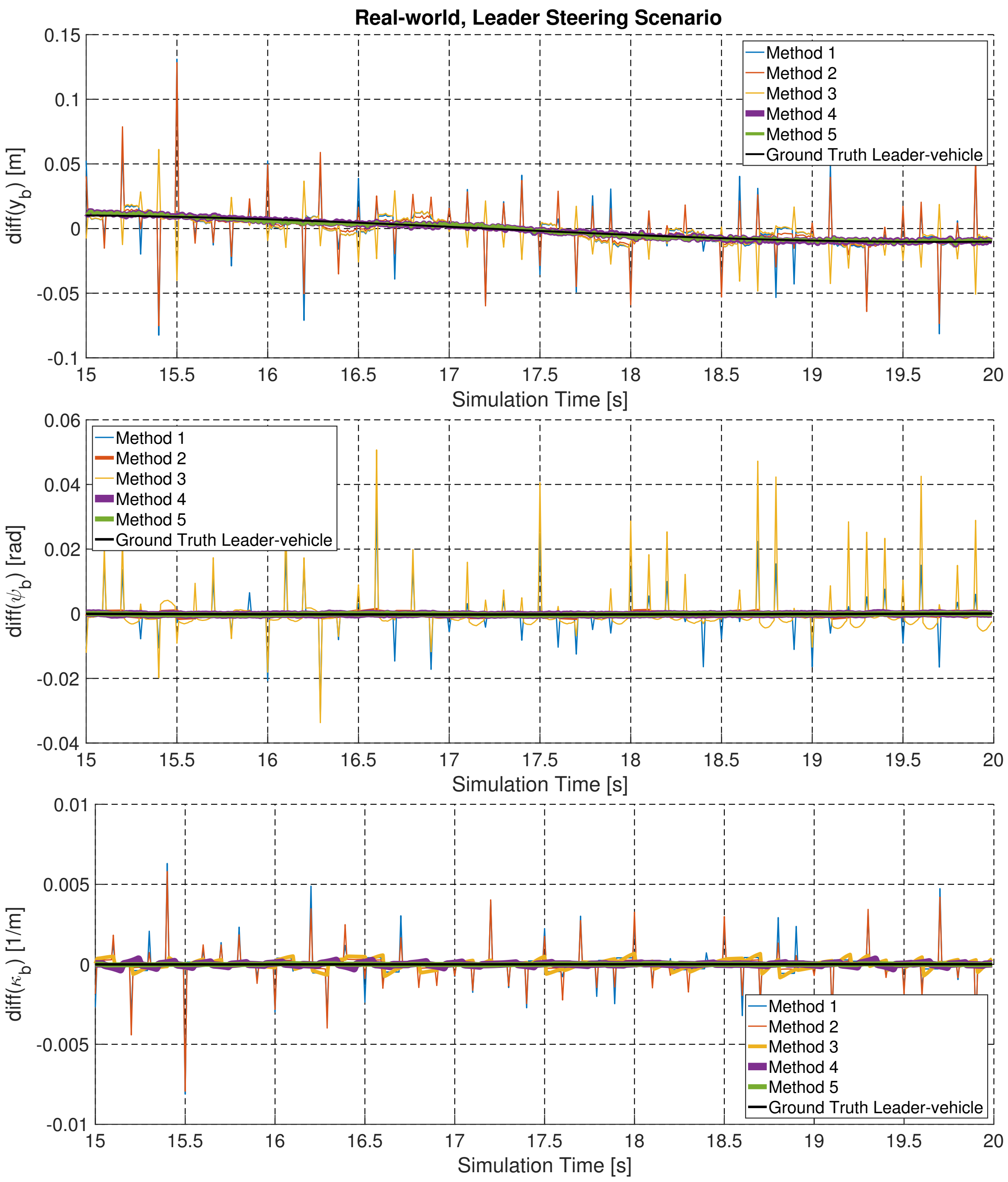


Fig. B-2: This figure shows the difference in generated path variables y_b, ψ_b, and κ_b, from simulating the real-world scenario. The real-world disturbances are Waypoint Noise, Delay, Offset and Motion Parameter Noise. The leader-vehicle is performing the sine steering maneuver, while the follower-vehicle drives straight. Clearly, the new Model-based Path Generation Methods 4 and 5 provide more continuous paths, compared to the Polynomial Fitting Methods 1, 2 and 3.

Appendix C

Experimental Parameters & Results

C-1 Bicycle Model Parameters

Parameter	**Symbol**	**Unit**	**Value**
Wheelbase	L	m	0.26
Distance front axle to CoG	l_f	m	0.15
Distance rear axle to CoG	l_r	m	0.11
Total mass	m	kg	2.286
Body inertia around vertical axis	I_z	$\mathrm{kg\,m^2}$	0.042
Front cornering stiffness	C_f	$\mathrm{N\,rad^{-1}}$	18.13
Rear cornering stiffness	C_r	$\mathrm{N\,rad^{-1}}$	30.08
Longitudinal vehicle velocity	V	$\mathrm{m\,s^{-1}}$	0.50

Table C-1: Virtual Leader Model single-track parameters used in the scaled vehicle experiments. These values are obtained by system identification performed by Mart Baars for the DSV.

C-2 Results

C-2-1 Path Discontinuity

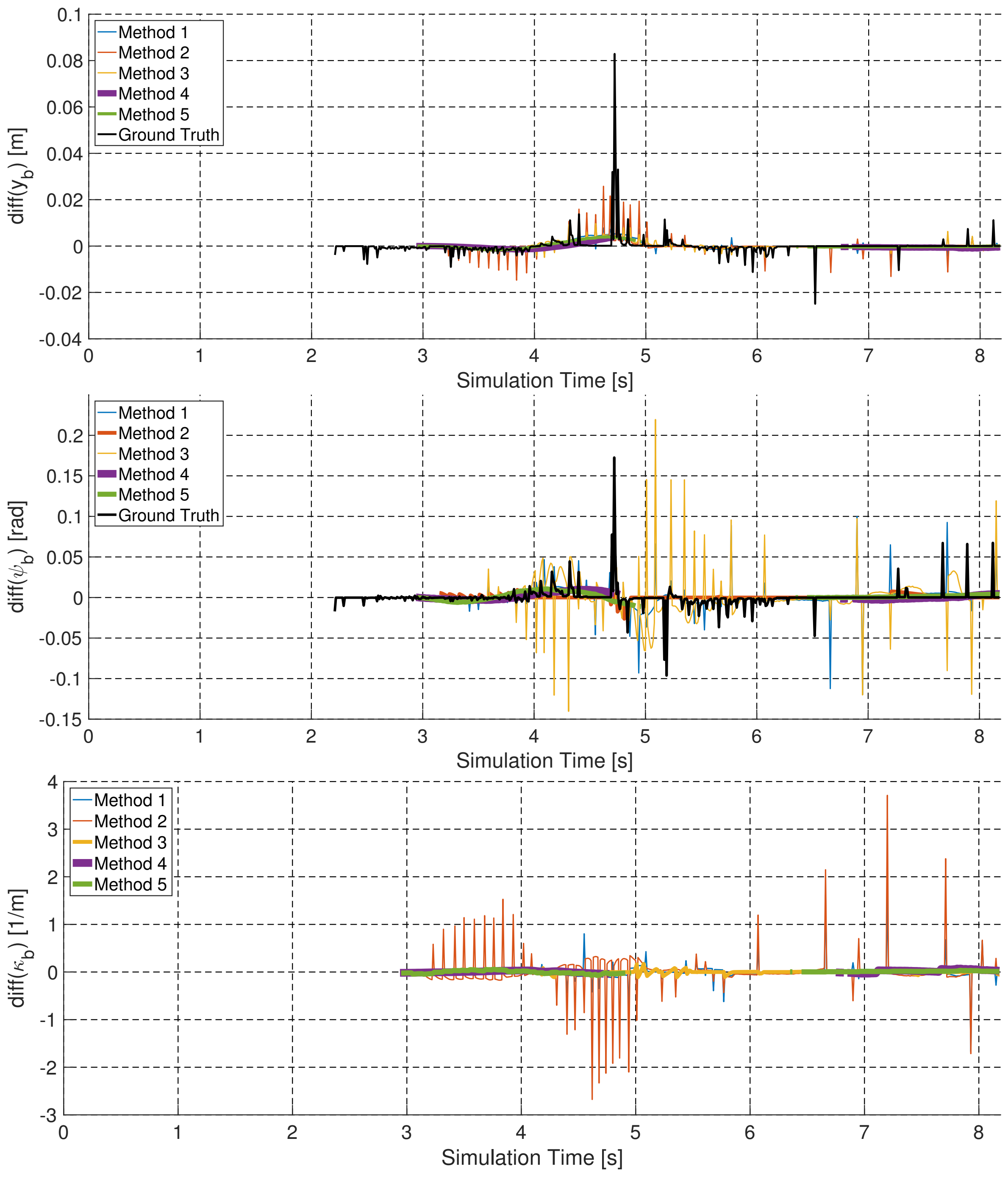


Fig. C-1: The differences in generated path variables y_b, ψ_b, and κ_b are shown. The results were obtained by running the path generators based on the scaled vehicle experimental data.

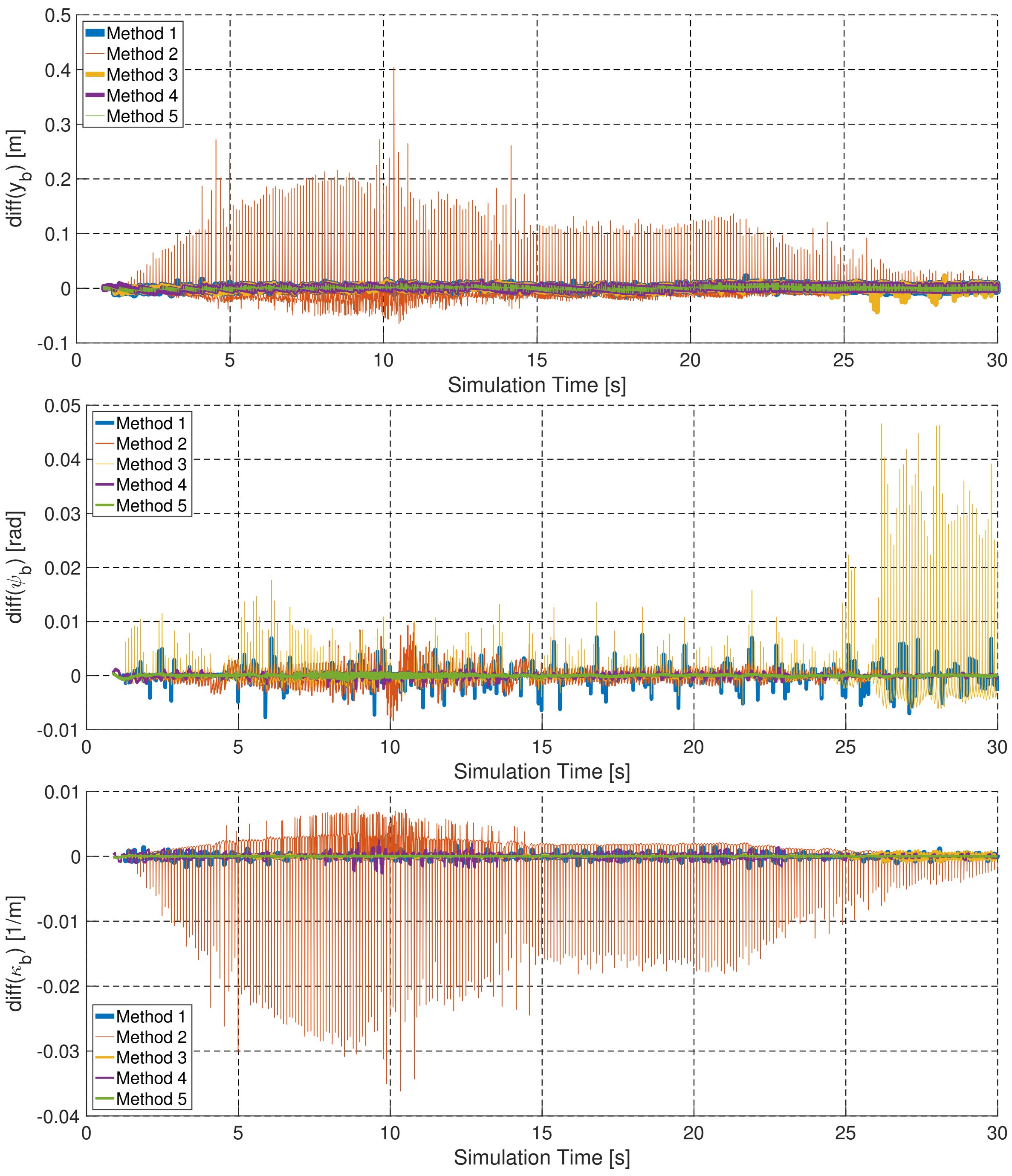


Fig. C-2: The differences in generated path variables y_b, ψ_b, and κ_b are shown. The results were obtained by running the path generators based on TNO's experimental data.

Bibliography

[1] M. Alirezaei, M. Corno, A. Ghaffari, and R. Kazemi, "A new approach to the design of coordinated road departure avoidance systems," *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, vol. 226, no. 1, pp. 45–60, 2012.

[2] C. L. Bottasso, D. Leonello, and B. Savini, "Path planning for autonomous vehicles by trajectory smoothing using motion primitives," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1152–1168, 2008.

[3] Y. Chen, H. Peng, and J. Grizzle, "Fast trajectory planning and robust trajectory tracking for pedestrian avoidance," *IEEE Access*, 2017.

[4] F. Esposto, J. Goos, A. Teerhuis, and M. Alirezaei, "Hybrid path planning for nonholonomic autonomous vehicles: An experimental evaluation," in *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on.* IEEE, 2017, pp. 25–30.

[5] P. Fernandes and U. Nunes, "Platooning of autonomous vehicles with intervehicle communications in sumo traffic simulator," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on.* IEEE, 2010, pp. 1313–1318.

[6] E. Frazzoli, M. A. Dahleh, and E. Feron, "Robust hybrid control for autonomous vehicle motion planning," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 1. IEEE, 2000, pp. 821–826.

[7] A. Hegyi, B. De Schutter, and H. Hellendoorn, "Model predictive control for optimal coordination of ramp metering and variable speed limits," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 3, pp. 185–209, 2005.

[8] P. A. Ioannou and C.-C. Chien, "Autonomous intelligent cruise control," *IEEE Transactions on Vehicular technology*, vol. 42, no. 4, pp. 657–672, 1993.

[9] W. Jansen, "Lateral path-following control of automated vehicle platoons," 2016.

[10] S. Kanarachos and M. Alirezaei, “An adaptive finite element method for computing emergency manoeuvres of ground vehicles in complex driving scenarios,” *International Journal of Vehicle Systems Modelling and Testing*, vol. 10, no. 3, pp. 239–262, 2015.

[11] S. Kato and S. Tsugawa, “Lateral and longitudinal control algorithms for visual platooning of autonomous vehicles,” in *Proc: F1SITA World Automotive Congress*, 2000.

[12] K. D. Kusano and H. C. Gabler, “Safety benefits of forward collision warning, brake assist, and autonomous braking systems in rear-end collisions,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1546–1555, 2012.

[13] S. M. LaValle and J. J. Kuffner Jr, “Randomized kinodynamic planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[14] T. C. Ng, M. Adams, and J. Ibañez-Guzmán, “Bayesian estimation of follower and leader vehicle poses with a virtual trailer link model,” *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 91–106, 2008.

[15] T. Nguyen, G. K. Mann, and R. G. Gosine, “Vision-based qualitative path-following control of quadrotor aerial vehicle,” in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on.* IEEE, 2014, pp. 412–417.

[16] R. B. Noland, “Relationships between highway capacity and induced vehicle travel,” *Transportation Research Part A: Policy and Practice*, vol. 35, no. 1, pp. 47–72, 2001.

[17] I. Papadimitriou and M. Tomizuka, “Lateral control of platoons of vehicles on highways: the autonomous following based approach,” *International journal of vehicle design*, vol. 36, no. 1, pp. 24–37, 2004.

[18] A. Schmeitz, J. Zegers, J. Ploeg, and M. Alirezaei, “Towards a generic lateral control concept for cooperative automated driving theoretical and experimental evaluation,” in *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on.* IEEE, 2017, pp. 134–139.

[19] E. Semsar-Kazerooni, J. Verhaegh, J. Ploeg, and M. Alirezaei, “Cooperative adaptive cruise control: An artificial potential field approach,” in *Intelligent Vehicles Symposium (IV), 2016 IEEE.* IEEE, 2016, pp. 361–367.

[20] M. Shan, Y. Zou, M. Guan, C. Wen, K.-Y. Lim, C.-L. Ng, and P. Tan, “Probabilistic trajectory estimation based leader following for multi-robot systems,” in *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on.* IEEE, 2016, pp. 1–6.

[21] B. Shyrokau, “Lecture notes vehicle dynamics course me41100,” Januari 2017.

[22] T. Van Den Boom, “Additional lecture notes model predictive control course ee4c04,” October 2016.

[23] R. van Hoek, M. Alirezaei, A. Schmeitz, and H. Nijmeijer, “Vehicle state estimation using a state dependent riccati equation,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3388–3393, 2017.

[24] P. Varaiya, "Smart cars on smart roads: problems of control," *IEEE Transactions on automatic control*, vol. 38, no. 2, pp. 195–207, 1993.

[25] J. Yazbeck, "Decentralized local approach for lateral control of platoons," 2010.

[26] L. Zhang, T. Ahamed, Y. Zhang, P. Gao, and T. Takigawa, "Vision-based leader vehicle trajectory tracking for multiple agricultural vehicles," *Sensors*, vol. 16, no. 4, p. 578, 2016.

[27] Y. Zou, M. Shan, M. Guan, C. Wen, and K.-Y. Lim, "A trajectory reconstruction approach for leader-following of multi-robot system."

Glossary

List of Acronyms

CoG	Center of Gravity
DCSC	Delft Center for Systems and Control
DSV	Delft Scaled Vehicle
IMU	Inertial Measurement Unit
KF	Kalman Filter
Method 1	Unconstrained Polynomial Fitting
Method 2	Heading Constrained Polynomial Fitting
Method 3	Curvature Constrained Polynomial Fitting
Method 4	Model-based Path Generation with Proportional Control
Method 5	Model-based Path Generation with MPC
MPC	Model Predictive Control
PF	Particle Filter
RRT	Rapidly-exploring Random Tree
RTK	Real Time Kinematic