# DEVELOPMENT OF A FOUR-WHEEL STEERING SCALE VEHICLE FOR AUTONOMOUS VEHICLE MOTION CONTROL EVALUATION

by

CHRISTOPHER CARLYLE ROTHER

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHATRONIC SYSTEMS ENGINEERING

2023

Oakland University
Rochester, Michigan

Thesis Advisory Committee:

Jun Chen, Ph.D., Chair
Khalid Mirza, Ph.D.
Shadi Alawneh, Ph.D.

ACKNOWLEDGMENTS

The author wishes to thank his advisor, Jun Chen, Ph.D., for his support and expertise throughout the research process. Thanks are also due to committee members Khalid Mirza, Ph.D. and Shadi Alawneh, Ph.D. for their contributions to this thesis. Acknowledgment should also be extended to peers Zhaodong Zhou, Ali Irshayyid, and Luke Nucalaj.

Christopher Carlyle Rother

ABSTRACT

# DEVELOPMENT OF A FOUR-WHEEL STEERING SCALE VEHICLE FOR AUTONOMOUS VEHICLE MOTION CONTROL EVALUATION

by

Christopher Carlyle Rother

Adviser: Jun Chen, Ph.D.

The instrumentation and operation of a full-size vehicle for Autonomous Vehicle motion control development can be costly. This paper presents a scale vehicle platform that serves as a cost effective transition from testing in a simulation environment to a physical system. The proposed scale vehicle platform, called JetRacer-4WS, is based on the open-source JetRacer autonomous vehicle with additional modifications to support four-wheel steering. To demonstrate the effectiveness of the proposed platform, model predictive control-based motion controls are tested and calibrated using an ultrasonic indoor positioning system to provide vehicle state information to the controller. The developed scale vehicle demonstrates that four-wheel steering model predictive control can significantly improve performance in certain driving scenarios, i.e., by reducing the root mean square path tracking error by up to 21% for the testing included in this paper. Finally, event-triggered model predictive control is also implemented and validated using the proposed platform.

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

## LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES—Continued

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AV | Autonomous Vehicle |
| DoE | Design Of Experiments |
| ESC | Electronic Speed Controller |
| MPC | Model Predictive Control |
| RMSE | Root Mean Square Error |

# CHAPTER ONE

# INTRODUCTION

Model Predictive Control (MPC) has been widely studied in the field of Autonomous Vehicle (AV) control [1–3] and mechatronics [4–6]. During early stages in the design/development process of an MPC system, a relatively simple simulation environment is generally used to ensure functionality and give an indication of the performance that can be expected when the MPC system is fully implemented later in hardware. For example, in [7], a simple MATLAB simulation environment was created to evaluate the performance of four different MPC-based AV path following controllers. Automotive simulation tools such as CARLA [8] are also used for a more thorough virtual evaluation of MPC performance [9].

Transitioning the testing/evaluation process of an MPC system from simulation into a physical environment presents several challenges. Instrumenting full-size vehicles and using controlled testing facilities can introduce much higher costs. To address this issue, scale vehicles have been developed in recent years to demonstrate autonomous driving capability [10–15]. Several scale vehicle platforms from existing literature are compared in Table 1.1. Integrating a scale vehicle platform into the controls development process allows engineers to perform initial hardware evaluations and to debug issues involved with using physical controllers, sensors, and actuators. Many scale vehicles, such as the platform presented in [10], are primarily designed as educational tools, as the two-wheel differential steering method is not representative of the steering systems found in full-size vehicles. The Ackermann steering mechanisms used in [11–15] have greater potential to be introduced into the AV development process. The scale vehicles presented in [13–15] feature end-to-end autonomous driving technology, where AI methods are used to generate control outputs directly from the sensor inputs. This is a potentially powerful application of modern technology; however, it is currently rarely used in industry as "explained by the challenges of verifying system performance" [13]. More traditional methods that involve separate environment perception, path planning, and motion control components are used in the scale vehicles presented in [10–13]. This results in greater correlation to current automotive technology, which encourages integration into the AV development process.

This paper presents a new scale vehicle platform, based on the open-source JetRacer [15], for testing and evaluation of autonomous vehicle motion controls, specifically for MPC systems. The proposed vehicle, which will be referred to as the JetRacer-4WS for the remainder of the paper, has two significant differences from other platforms that support its research contribution. *First*, the vehicle has four-wheel steering capability, which allows early stage testing and validation of MPC-based motion control algorithms with active rear steering functionality. *Second*, the platform is designed primarily to test and evaluate motion control algorithms, rather than to demonstrate full autonomous driving capability. In order to isolate the motion controls and avoid error from other aspects of traditional AV operation, the platform does not include any onboard environment perception capabilities, and a predefined vehicle path is used in lieu of real-time path planning. Instead, vehicle localization is achieved using a Marvelmind indoor positioning system [16] to provide real-time vehicle state information, which is fed into the motion control module as feedback. Note that alternative positioning systems could be developed using an overhead camera to track the vehicle with computer vision methods [12] or an RTK GPS unit for outdoor testing [17, 18].

To demonstrate the effectiveness of the JetRacer-4WS platform and explore the viability of implementing four-wheel steering MPC in physical AV control systems,

Table 1.1: Comparison of the proposed JetRacer-4WS with a selection of existing scale vehicles.

| Platform | Steering | Use Case | Processor | Onboard Sensors | Cost |
|---|---|---|---|---|---|
| Duckiebot [10] | Two-wheel differential | PD lane keeping control | Raspberry Pi 2 | Camera w/ fisheye lens | \$150 |
| MIT Racecar [11] | Ackermann | PID steering control | NVIDIA Jetson TX1 | RGB-D camera<br>Scanning Lidar<br>Stereo camera<br>IMU<br>Speedometer | ~\$3000 estimated in [12] |
| ASIMcar [12] | Ackermann | Stanley control method for lane keeping | NVIDIA Jetson TX2 | IMU<br>Front camera w/ fisheye lens<br>Rear camera<br>One-beam Lidar<br>Speedometer | \$1200 |
| F1/10 [13] | Ackermann | End-to-end autonomous driving and MPC | NVIDIA Jetson TX2 | Monocular USB web cam<br>Depth camera<br>Scanning Lidar<br>IMU | \$3480 [19] |
| CNN-based scale vehicle [14] | Ackermann | End-to-end autonomous driving | Raspberry Pi 4 | Camera w/ fisheye lens<br>Ultrasonic sensor | N/A |
| JetRacer [15] | Ackermann | End-to-end autonomous driving | NVIDIA Jetson Nano | Wide angle camera | \$825 |
| JetRacer-4WS (ours) | Independent front and rear Ackermann | MPC path following evaluation | NVIDIA Jetson Nano | Mobile beacon for Marvelmind indoor positioning system | \$850 |

several experiments are conducted comparing the path following performance of two-wheel steering and four-wheel steering MPC systems. With a standard time-triggered MPC implementation, both systems are calibrated to comparable maturity by designing and running a series of calibrations using latin hypercube-based Design of Experiment (DoE) methods. The performance of each calibration is then indexed based on root mean square error and maximum error. The results of this scale vehicle research support the application of four-wheel steering motion controls in physical AV systems to improve path following performance in low-speed driving scenarios that benefit from greater agility and maneuverability. Then, an event-triggered MPC system [7, 20–27] is implemented and tested with several triggering thresholds. The results show that the four-wheel steering system continues to perform best for the given driving scenario, while the triggering threshold can be used to reduce computational load in exchange for reduced path following performance.

The remainder of the paper is organized as follows. Chapter Two covers the hardware design of the scale vehicle. Chapter Three introduces the MPC-based path following controllers that are used to demonstrate the scale AV platform. Chapter Four presents a systematic and automatic calibration procedure, together with experimental results, and Chapter Five presents another case study of validating event-triggered MPC with the proposed JetRacer-4WS. The paper is concluded in Chapter Six.

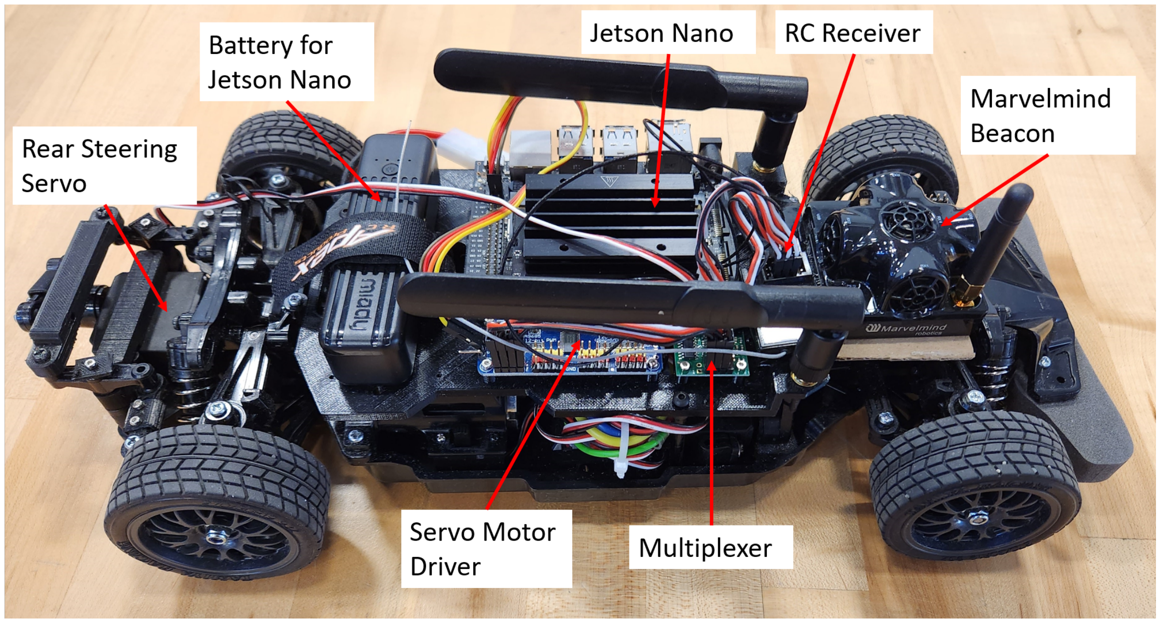


Figure 2.1: The proposed scale AV platform with four-wheel steering capability.

# CHAPTER TWO

# SCALE VEHICLE DESIGN

The JetRacer-4WS platform, based on the open-source JetRacer [15], is built from a 1/10th scale Tamiya TT-02 RC car. See Fig. 2.1. The vehicle is driven by an electric motor powering all four wheels through open differentials. Motor speed is controlled by a Tamiya Electronic Speed Controller (ESC). A servo multiplexer (switched from the RC transmitter) is used to select if the RC transmitter/receiver or the Jetson Nano/servo driver module supply the control signals for the ESC and steering servos.

An NVIDIA Jetson Nano Developer Kit is installed on the Jetracer-4WS for real-time data processing and control. Much of the code for the standard JetRacer platform is for end-to-end autonomous driving and is not useful for this application. However, the

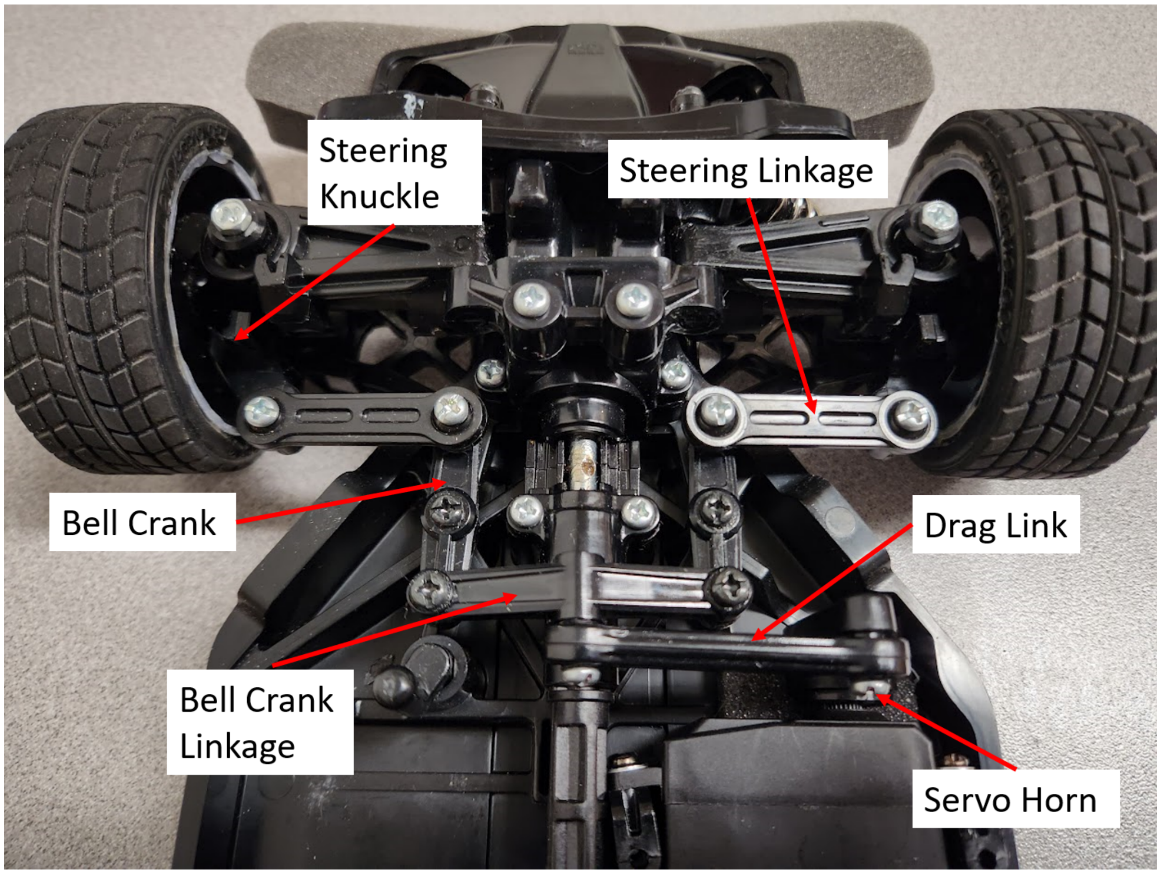
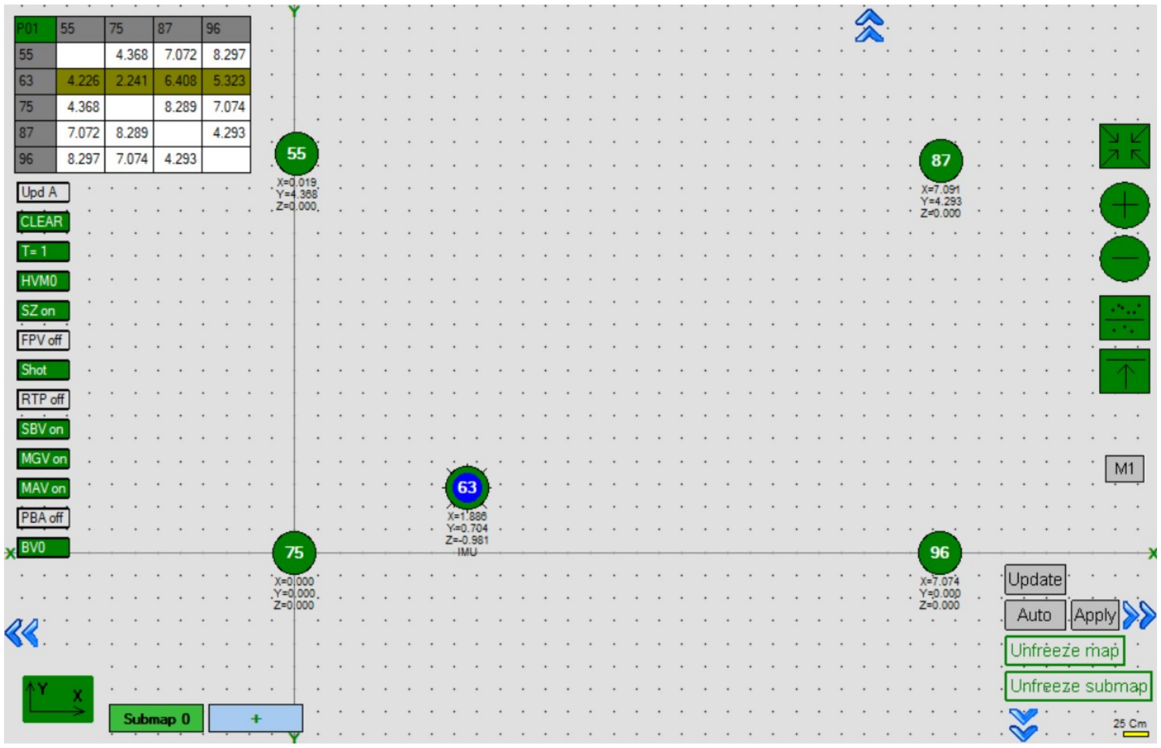


Figure 2.2: Standard front wheel steering assembly for the Tamiya TT-02 RC car.

codes setting up the control interfaces for the ESC and steering servo are reused. New instances of the interface are created to process the rear steering servo commands from additional sets of pins on the servo control board. One set of pins is used to keep the servo at zero degrees when the multiplexer passes through the RC receiver signals. The other set of pins is used to control the servo based on the MPC-generated steering command when the multiplexer passes through the control signals from the Jetson Nano.

For the testing presented in this paper, the MPC system runs on a separate computer. To calculate vehicle state information for the MPC-based motion controls, a Marvelmind indoor positioning system [16] is used. The system consists of five beacons and a modem. One of the beacons is mounted on the scale vehicle, and the Marvelmind software is used to modify the settings to identify it as the mobile beacon. The other four beacons are placed around the perimeter of the testing area, and they are set as stationary beacons. Fig. 2.3 shows the beacon positions in the Marvelmind software. The position of the mobile beacon is calculated based off of a time-of-flight calculation of an ultrasonic signal sent from the mobile beacon to the stationary beacons [16]. The modem communicates with the beacons wirelessly and sends data to the computer through a USB connection. The computer interfaces with the onboard Jetson Nano through a WiFi connection using Jupyter Lab. When the MPC system generates updated control commands, they are sent from the computer to the Jetson Nano using a socket created with the Python socket library.

The Jetracer-4WS platform's front wheels use an Ackermann steering system controlled by a servo motor as shown in Fig. 2.2. Originally, the rear wheels are designed to be fixed at zero degrees. In order to create a rear wheel steering system for the car, the front wheel steering design is modified to fit the different architecture around the rear axles. See Fig. 2.4. Because of the placement of the spur gear and electric motor, it is



| P01 | 55 | 75 | 87 | 96 |
|---|---|---|---|---|
| 55 | | 4.368 | 7.072 | 8.297 |
| 63 | 4.226 | 2.241 | 6.408 | 5.323 |
| 75 | 4.368 | | 8.289 | 7.074 |
| 87 | 7.072 | 8.289 | | 4.293 |
| 96 | 8.297 | 7.074 | 4.293 | |

55
X=0.019
Y=4.368
Z=0.000

87
X=7.091
Y=4.293
Z=0.000

63
X=1.886
Y=0.704
Z=-0.981
IMU

75
X=0.000
Y=0.000
Z=0.000

96
X=7.074
Y=0.000
Z=0.000

Figure 2.3: Marvelmind Dashboard software showing the stationary/mobile beacon locations.

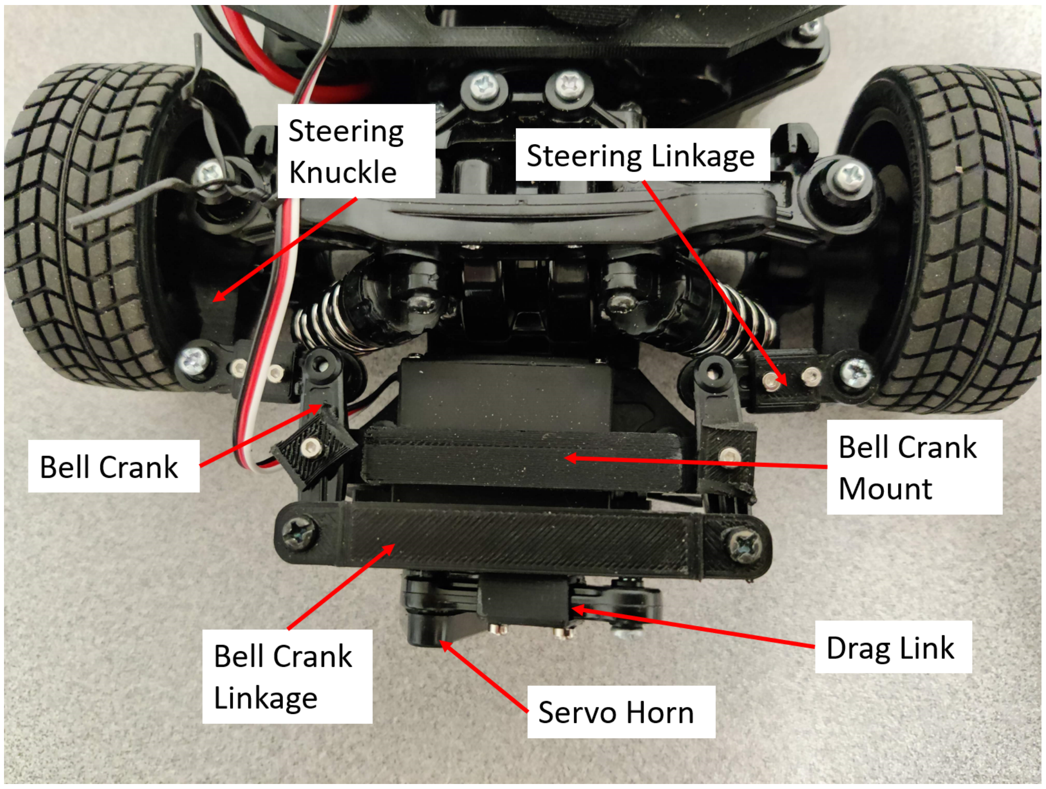


Figure 2.4: Custom rear wheel steering assembly.

simplest to mount the rear wheel steering components behind the rear axle. While original parts can be used for some sections of the mechanism, other parts have to be modified or specially designed and 3D printed to account for differing measurements.

Mounting points on the rear bumper originally used for cosmetic parts are repurposed to fasten replacement copies of the front wheel bell cranks using a 3D printed bracket with mounting pins. Because the bell cranks are positioned farther apart, the bell crank linkage design also has to be lengthened to match the separation of the mounting points and 3D printed. Replacement copies of the steering links and drag link both have to be shortened (by cutting out a section of the middle of the link) and reconnected with a 3D

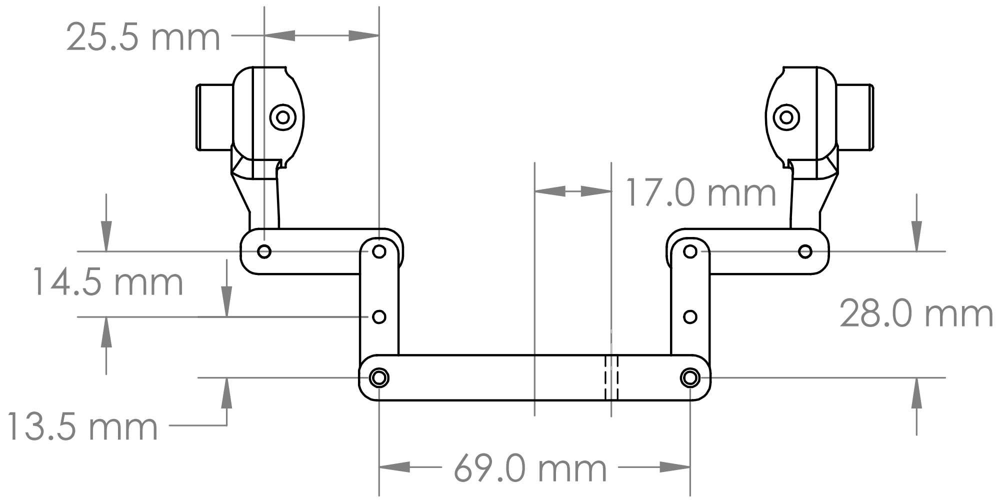


Figure 2.5: Custom rear wheel steering assembly annotated drawing.

printed bracket. The final lengths of the steering links must result in the wheels pointing straight forward when the steering linkage assembly is centered. The rear steering knuckles are also redesigned, as shown in Fig. 2.6, since exact copies of the front knuckles would interfere with the rear dampers. The rear suspension arms also have to be substituted for replacement copies of the front suspension arms. After drilling holes in the rear bumper and fastening a second steering servo in place, the custom rear knuckles are attached to the suspension arms, and the aforementioned steering linkage components are assembled as shown in Fig. 2.4 and Fig. 2.5.

The electronic components are mostly connected as specified for the standard JetRacer platform. See Fig. 2.7. The newly added rear steering servo connector is

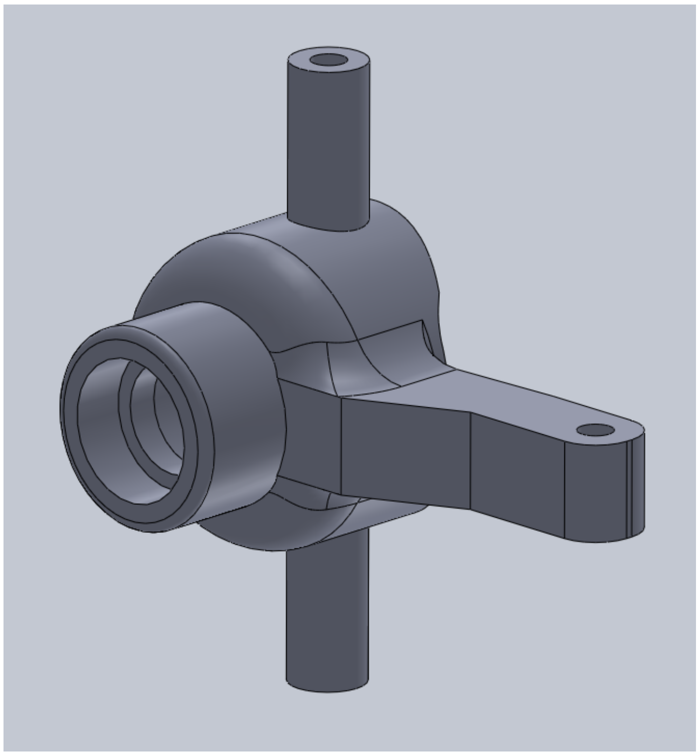


Figure 2.6: CAD drawing of the rear steering knuckle redesign.

Table 2.1: Summary bill of materials for scale vehicle

| Item Description | Cost ($) |
|---|---|
| Tamiya TT02 NSX Kit | 167.50 |
| Tamiya TT02 Spare Parts for Rear Steering | 24.75 |
| Futaba RC Transmitter | 162.35 |
| Futaba Ball Bearing Servo x 2 | 33.46 |
| Tenergy 7.2V Battery Pack | 33.29 |
| Intel Dual Band Wireless Card | 20.98 |
| TEU-105BK brushed ESC | 52.00 |
| Tenergy Battery Charger | 23.99 |
| 3D printed parts | 40.00 |
| Misc Additional Parts | 160.00 |
| Nvidia Jetson Nano Dev Kit - B01 | 118.75 |
| **Total** | 837.07 |

attached to an output on the multiplexer board. The corresponding inputs on the multiplexer both come from the servo driver.

**Remark 1** *Reducing cost of controls testing and evaluation is one of the primary reasons to include a scale vehicle in the development process. All of the proposed JetRacer-4WS components including the Jetson Nano Developer Kit can typically be found for around $850. Table 2.1 lists the bill of materials for the proposed scale vehicle platform. In addition, the Marvelmind system used for indoor positioning is currently between $450-500[1]. In addition, Table 1.1 compares the cost of existing small vehicle platforms.*

[1]Note that the cost information included in this chapter is based on the market price as of Fall 2021.

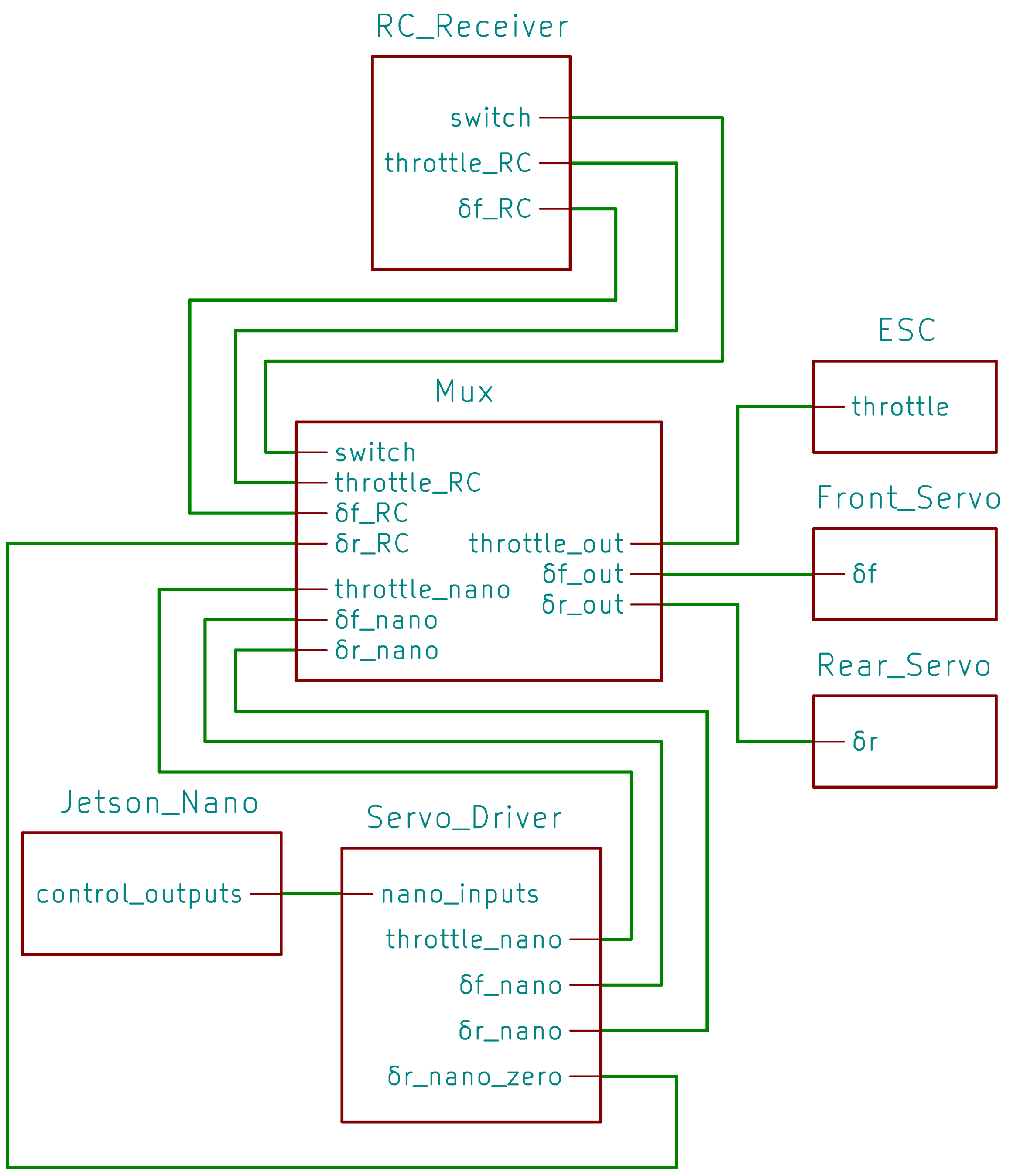


Figure 2.7: Wiring schematic for JetRacer-4WS.

# CHAPTER THREE

# MPC-BASED PATH FOLLOWING

To demonstrate the effectiveness of the JetRacer-4WS, an MPC path tracking controller is designed. This chapter presents details of the MPC-based path tracking. The control process is summarized in Fig. 3.1.

## 3.1 Vehicle Dynamic Model

MPC differs from traditional PID control because it incorporates a model of the system behavior to determine the control action, rather than simply calculating a control action based on the system's offset from a desired set point. In other words, MPC requires a model to predict the system evolution over the prediction horizon. This section briefly discusses the vehicle model. Please refer to [28] for more details.

In literature, a dynamic vehicle model has been widely utilized [29, 30]. However, since vehicle speed is usually low for the scale vehicle, a kinematic model is used in this paper, which is specified as follows:

$$\dot{p}_x = V\cos(\psi+\beta) \tag{3.1a}$$

$$\dot{p}_y = V\sin(\psi+\beta) \tag{3.1b}$$

$$\dot{\psi} = \frac{V\cos(\beta)}{L_{xf}+L_{xr}}(\tan(\delta_f)-\tan(\delta_r)) \tag{3.1c}$$

$$\beta = \arctan\left(\frac{L_{xf}\tan(\delta_r)+L_{xr}\tan(\delta_f)}{L_{xf}+L_{xr}}\right) \tag{3.1d}$$

where $p_x$ and $p_y$ are the vehicle longitudinal and lateral positions, respectively, and $\psi$ is the vehicle yaw angle, all in the global frame; $\beta$ is the vehicle slip angle; $V$ is the velocity of the vehicle's center of gravity; $L_{xf}$ and $L_{xr}$ are the distances from the center of gravity to the front and rear axles; $\delta_f$ and $\delta_r$ are the front and rear steering angles. Fig. 3.2 illustrates these vehicle parameters using a bicycle model. The Marvelmind indoor positioning system provides $p_x$ and $p_y$. The angle $\psi + \beta$ and the corresponding velocity $V$ are calculated from consecutive position measurements. Angle $\psi$ can be calculated from the $\psi + \beta$ measurement by solving for $\beta$ using equation 3.1d.

The state vector can be compactly denoted as

$$x(t) = \begin{bmatrix} p_x, p_y, \psi \end{bmatrix}^T,$$

and the control vector can be compactly denoted as

$$u(t) = \begin{bmatrix} \delta_f, \delta_r \end{bmatrix}^T$$

Note that for a four-wheel steering system, the front and rear steering angles can be independently commanded, while for a conventional two-wheel steering system with front steering only, the rear steering angle $\delta_r$ is fixed to zero, and the control vector reduces to $u(t) = \delta_f$.

## 3.2 Optimal Control Problem

MPC uses a prediction horizon $p$ of several future points in time. The control sequence $U$ is the series of control actions for each point in the prediction horizon. Each control sequence results in a state sequence $X$ according to the prediction model described above. A cost function can then be defined over the control sequence as well as the state sequence to quantify the performance of the control sequence over the prediction horizon. An optimization process is executed to determine the optimal control sequence that minimizes the cost function.

In the case of path tracking control, the primary objective is to minimize the offset between the vehicle's actual and desired position. However, to ensure stability, the cost function also includes other terms to penalize large steering angles and large changes in steering angle between time steps. These terms are implemented to minimize the impact of noise in the MPC inputs and to ensure smooth steering inputs that mimic a human driving style. More specifically, MPC solves the following optimal control problem at each time step:

$$\begin{aligned}
\min_{u} \quad & \sum_{k=1}^{p} ||x_k - x_k^r||_{Q_x}^2 + \sum_{k=0}^{p-1} ||u_k||_{Q_u}^2 \\
& \qquad + \sum_{k=0}^{p-1} ||u_k - u_{k-1}||_{Q_d}^2 & \text{(3.2a)} \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k),\ k = 0, \ldots, p-1 & \text{(3.2b)} \\
& u_{min} \le u_k \le u_{max},\ k = 0, \ldots, p-1 & \text{(3.2c)} \\
& \Delta_{min} \le u_k - u_{k-1} \le \Delta_{max},\ k = 0, \ldots, p-1 & \text{(3.2d)}
\end{aligned}$$

where the first term in the cost function penalizes deviation from the desired path, the second term discourages large steering angles, and the third term minimizes the steering angle rate of change. The constraint (3.2b) can be obtained by discretizing the kinematic vehicle dynamics (3.1).

The reference state term $x_k^r$ of the optimal control problem (3.2) is populated with coordinates representing the desired vehicle path over the prediction horizon. The coordinates are selected from the predetermined array of points that make up the track. The first point $x_0^r$ is set as the track point with the minimum distance to the initial vehicle position $x_0$. Note that this value may be different from the measured value of the vehicle's position if delay compensation is implemented, as described in Section 3.3. Then, the number of points that the vehicle will pass for each step in the prediction horizon assuming constant velocity is calculated by multiplying the scale vehicle velocity by the distance between track points and dividing by the time between steps. The rounded number of points is then used to calculate the indices of the remaining track points that populate $x_k^r$. Fig. 3.3 shows an example of a reference state sequence with the corresponding state sequence. Note that the term of the cost function (3.2a) that discourages large steering angles causes an increasing offset between the state sequence and the reference state sequence towards the end of the prediction horizon.

**Remark 2** *As a receding horizon control technique, after solving (3.2), a time-triggered MPC implementation then applies only the first action in the resulted control sequence, and then the process is repeated at the next time step with updated feedback measurement. In this paper, we will also demonstrate the effectiveness of the proposed JetRacer-4WS through by event-triggered MPC [7], which continues to apply subsequent actions in the resulted control sequence without resolving (3.2) for each time step until a threshold is exceeded. In this case, the threshold is implemented as the vehicle's deviation from the desired path. Event-triggered MPC is used to reduce computational load compared to standard time-triggered MPC. As the triggering threshold increases, new control sequences are calculated at a lower frequency, which generally trades reduced computational load for reduced system performance. See Chapter 5 for more details.*

Note that each term in the cost function (3.2a) has a gain that is calibratable to achieve ideal driving characteristics. Different values for $Q_x$, $Q_u$, and $Q_d$ can result in significantly different control performance. Therefore, a systematic approach to tune these parameters will be proposed and studied in Chapter Four.

## 3.3 Delay Compensation

Due to several factors contributing to delay in the system, the actual position of the scale vehicle will be somewhat ahead of the calculated vehicle state by the time the control action is applied and the vehicle reacts. Delay compensation is included in the motion controls to account for this issue. The main causes of the delay are the time the processor takes to update the control sequence and the latency of the Marvelmind position data caused by filtering methods designed to reduce noise. Several steps can be taken to significantly reduce, but not entirely remove, latency in the Marvelmind system. For example, the mobile beacon contains an IMU which allows the Marvelmind system to generate velocity data; however, configuring the system to include this processed data resulted in a significantly increased latency.

In this paper, delay compensation is achieved using the vehicle model to solve for an estimated vehicle state one time step in the future based on the measured vehicle state, the vehicle velocity, and the current control set point. The MPC system uses this estimated vehicle state as the initial state $x_0$, as it is expected to more accurately reflect the real vehicle state when the newly calculated control action is applied.

## 3.4 PID Velocity Control

While testing path tracking controls with the JetRacer-4WS platform, a constant throttle value can be used. However, implementing a PID-based velocity control system has multiple benefits. First, it allows the selection of a velocity set point, while the average velocity achieved with a constant throttle setting may shift over time due to factors such as the battery state of charge. In addition, implementing velocity control can somewhat reduce velocity fluctuations caused by large steering angles and other disturbances.

The PID controller is implemented by defining the velocity error as $e = V_r - V$ where $V_r$ is the reference velocity. The proportional term applies a control input based on the magnitude of the error signal. The integral term builds up over time based on the error magnitude. In this application, the integral term is mainly used to provide the steady-state throttle position. The derivative term produces a control output based on the rate of change of the error signal, which allows the control system to anticipate an account for a potential overshoot. However, as is typical with PID control, the derivative term results in an amplification of noise from the error signal. To minimize this issue, a discrete low pass filter is implemented using the form of the single-pole infinite impulse response filter presented in [31], both on the velocity signal and on the calculation of the derivative term. The raw velocity signal $v$ is filtered as follows:

$$V = (b * v) + (a * V_{-1}) \tag{3.3}$$

where $V_{-1}$ is the previous filtered velocity, and $a$ and $b$ are the filter coefficients. For this application, in addition to the standard PID terms, an additional control term $S$ is defined. This term is used to increase the throttle value as steering angles increase to account for sources of resistance such as reduced efficiency of u-joints. The proportional, integral, and derivative terms are calculated using the following equations:

$$P = K_p * e \tag{3.4a}$$

$$I = (K_i * e) + I_{-1} \tag{3.4b}$$

$$\frac{de}{dt} = \frac{e - e_{-1}}{dt} \tag{3.4c}$$

$$D = K_d * ((b * \frac{de}{dt}) + (a * (\frac{de}{dt})_{-1})) \tag{3.4d}$$

$$S = K_s * \delta_f^2 \tag{3.4e}$$

$$throttle = P + I + D + S \tag{3.4f}$$

where $K_p$, $K_i$, and $K_d$ are the PID weights, $I_{-1}$ is the previous integral term value, $e_{-1}$ is the previous velocity error value, $(\frac{de}{dt})_{-1}$ is the previous error derivative value used for filtering, and $K_s$ is the steering compensation weight. Initially, this velocity control system was implemented and calibrated for the JetRacer-4WS on the small track with $V_r = 1m/s$, as shown in Fig. 3.4. This set point is close to the minimum possible value that allows for effective control. The minimum throttle value that the ESC will apply results in a velocity of approximately $0.7m/s$. To function effectively, the velocity controller must have a reasonable throttle range below the set point to reduce velocity when required. In addition, the relationship between throttle and velocity becomes increasingly non-linear approaching the minimum value. As expected, the system was able to control to the $1.0m/s$ set point with an average velocity of $1.003m/s$. The system was also able to reduce the velocity fluctuations. The Root Mean Square Error (RMSE) values are calculated based on the error from the average velocity, as the constant throttle test did not have a velocity set point, resulting in an average velocity of $1.077m/s$. The RMSE value for the constant throttle test was $0.077m/s$, and the RMSE for the velocity controlled test was $0.060m/s$, or a 22% improvement. Next, the velocity controller set point was increased to $1.6m/s$ for testing on the larger track. After recalibration, the results (Fig. 3.5) showed that the velocity controller was able to achieve an average velocity of $1.591m/s$. The velocity controller achieved an RMSE value of $0.058m/s$, while the constant throttle test, with an average velocity of $1.672m/s$, resulted in an RMSE value of $0.065m/s$.

## 3.5 Track Generation

The JetRacer-4WS scale vehicle testing included in this paper uses an oval track (two half circles connected by straight sections) defined as an array of coordinates. In order to simplify the process of modifying the track, a Python code was written to generate the track based off of several inputs. Because the Marvelmind data is generated using one of the stationary beacons as the origin, it is useful to specify shifts in the x and y directions, as well as a rotation angle for the track ($\theta$). In addition, the code allows the user to set the radius of the turns ($R$), the length of the straight sections ($L$), and the number of points included in each half circle section ($N$).

First, coordinates for the upper half circle are generated. For each of the the points from $n = 0$ to $n = N - 1$, the coordinates are calculated using the following equations:

$$x = R * \cos\left(\frac{n * \pi}{N-1}\right) \tag{3.5a}$$

$$y = R * \sin\left(\frac{n * \pi}{N-1}\right) \tag{3.5b}$$

The lower half circle coordinates are the same, but with negated $x$ and $y$ values. Then, the distance between points ($d$) is calculated as $d = (\pi * r)/(N - 1)$. To ensure that the value of $d$ remains constant throughout the track, the requested length of the straight sections $l$ is modified to be the first multiple of $d$ that exceeds the requested length. The top/bottom half circles are shifted up/down by $L/2$. The left and right straight section coordinates are generated to fill the space between the half circle sections, while ensuring that the order of points results in a continuous track when the four sections are concatenated into a single array. After concatenation, the track can be rotated by multiplying each coordinate pair by the following rotation matrix:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.6}$$

Finally, all coordinates in the rotated track are shifted by the specified values to create the final array of track points used for JetRacer-4WS motion controls testing.

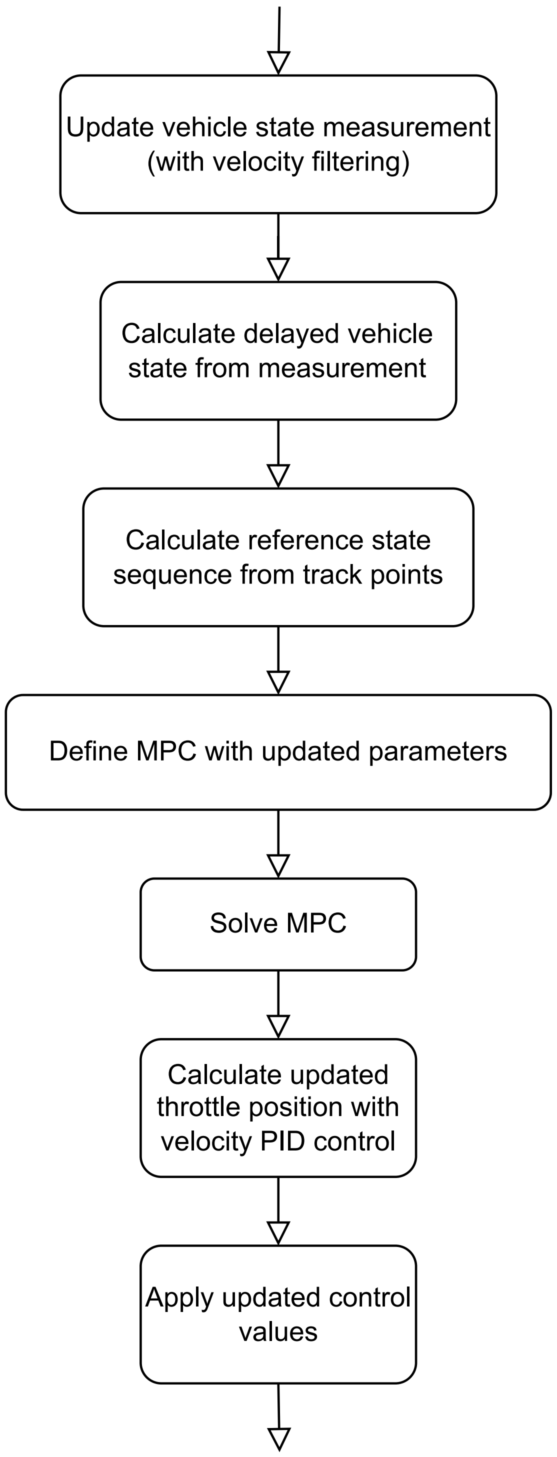

Update vehicle state measurement (with velocity filtering)
Calculate delayed vehicle state from measurement
Calculate reference state sequence from track points
Define MPC with updated parameters
Solve MPC
Calculate updated throttle position with velocity PID control
Apply updated control values

Figure 3.1: Overview of JetRacer-4WS motion control system.

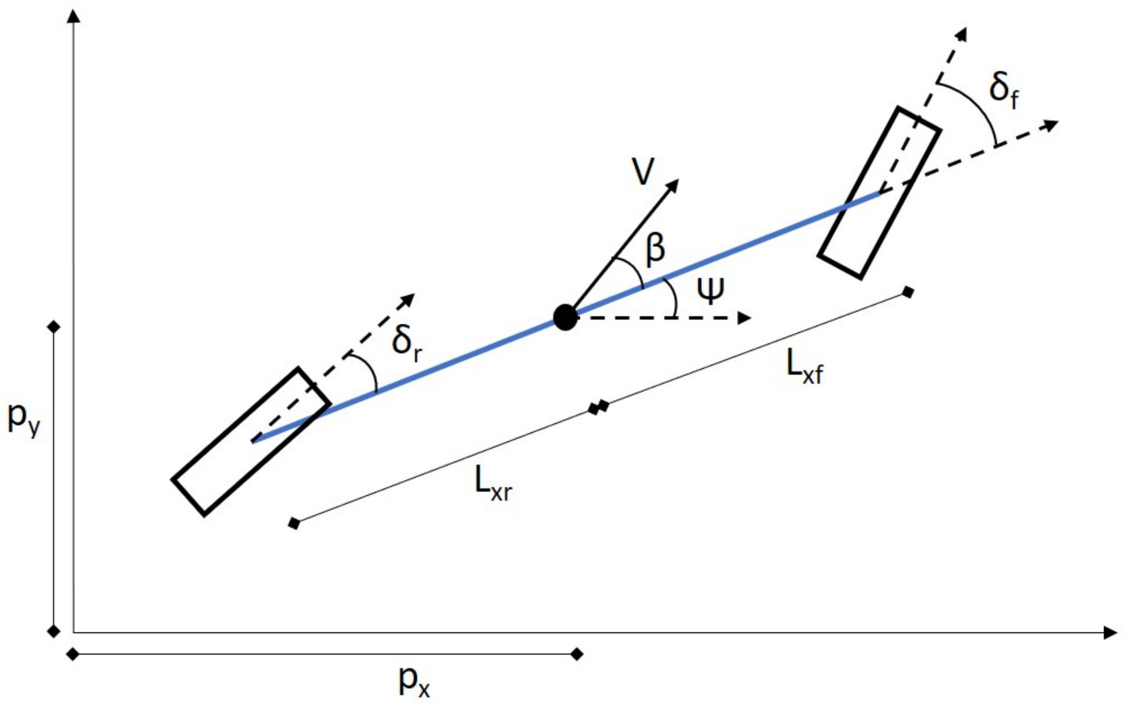


Figure 3.2: A bicycle model with relevant parameters for the vehicle dynamic model.

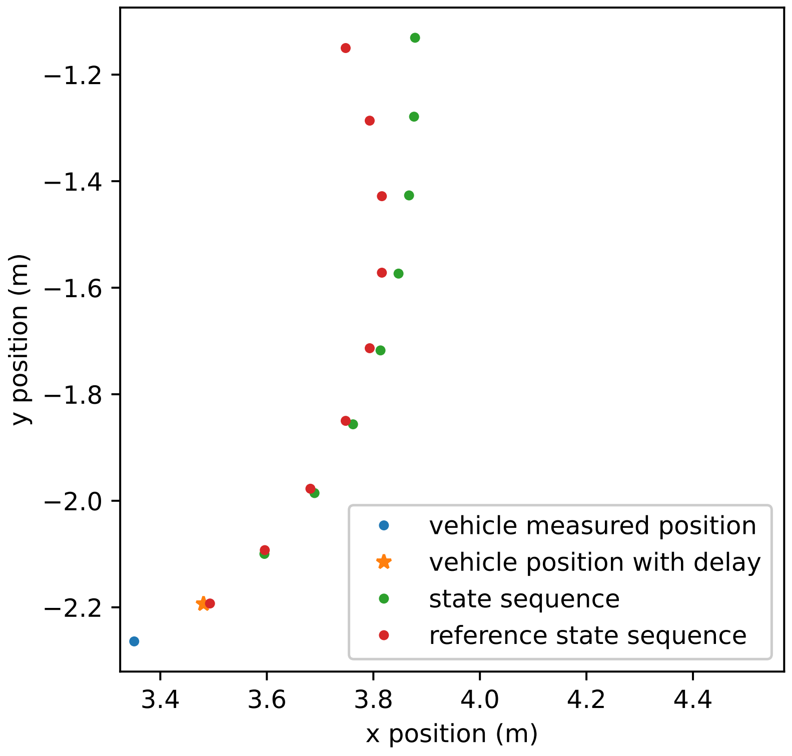


Figure 3.3: Reference state sequence coordinates plotted with the corresponding state sequence.

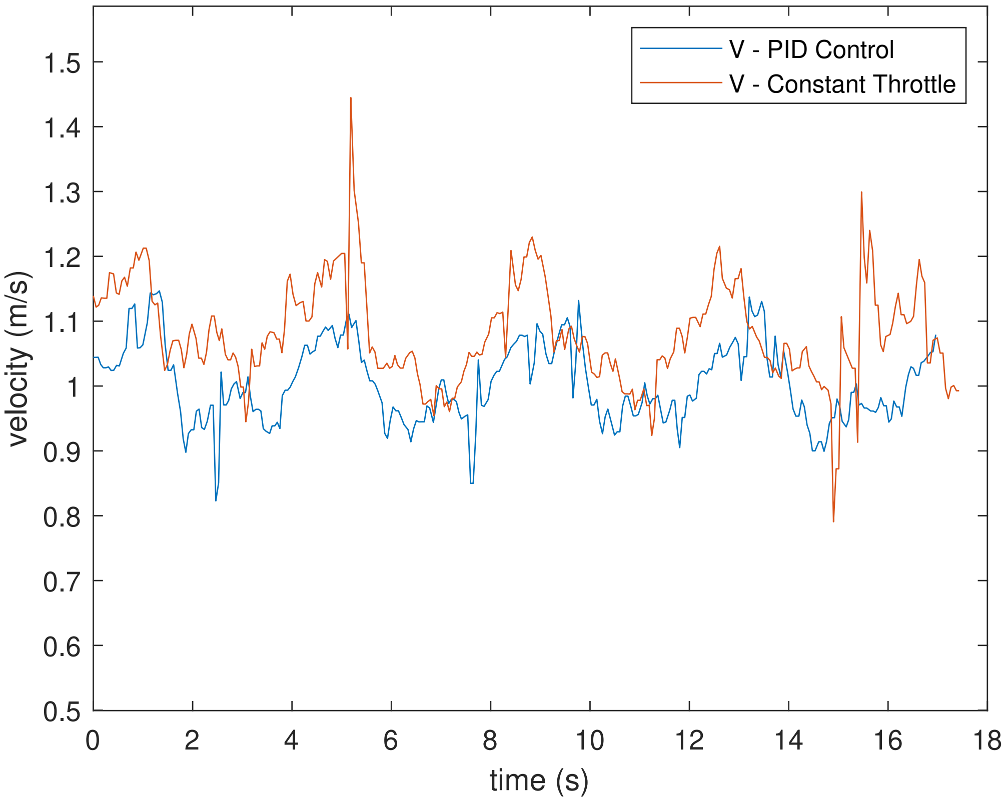


Figure 3.4: Comparison of velocity signals with PID control (1.0$m/s$ set point) and constant throttle.

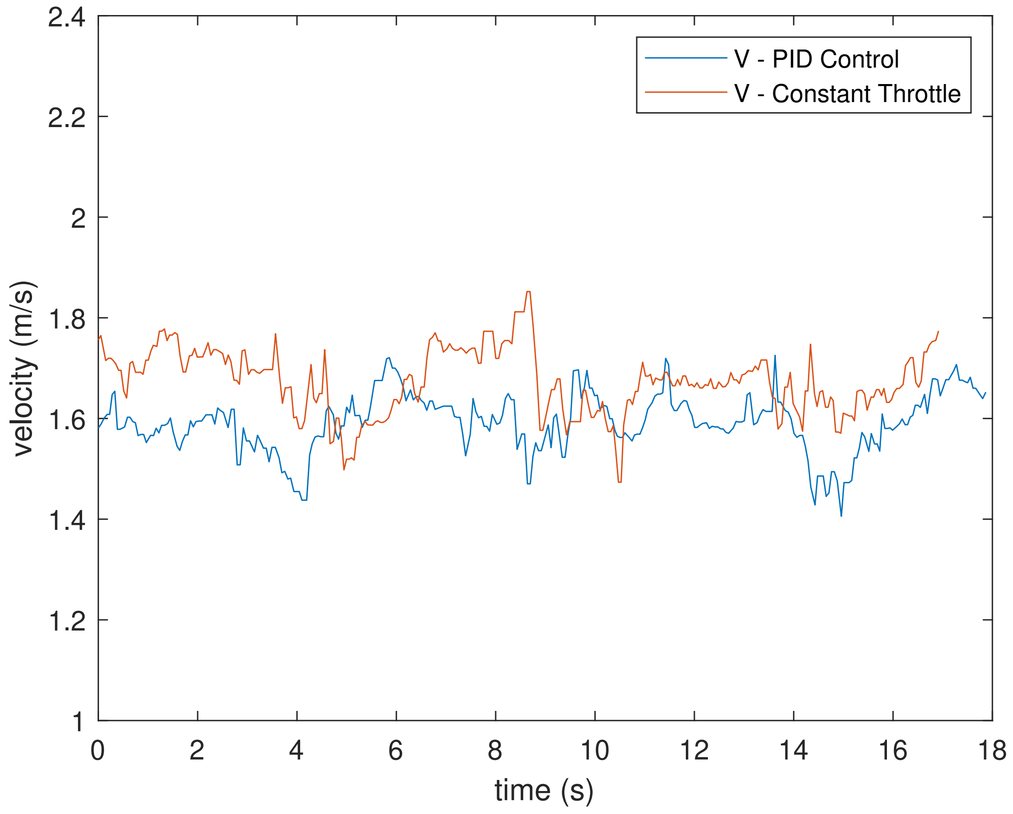


Figure 3.5: Comparison of velocity signals with PID control ($1.6 m/s$ set point) and constant throttle.

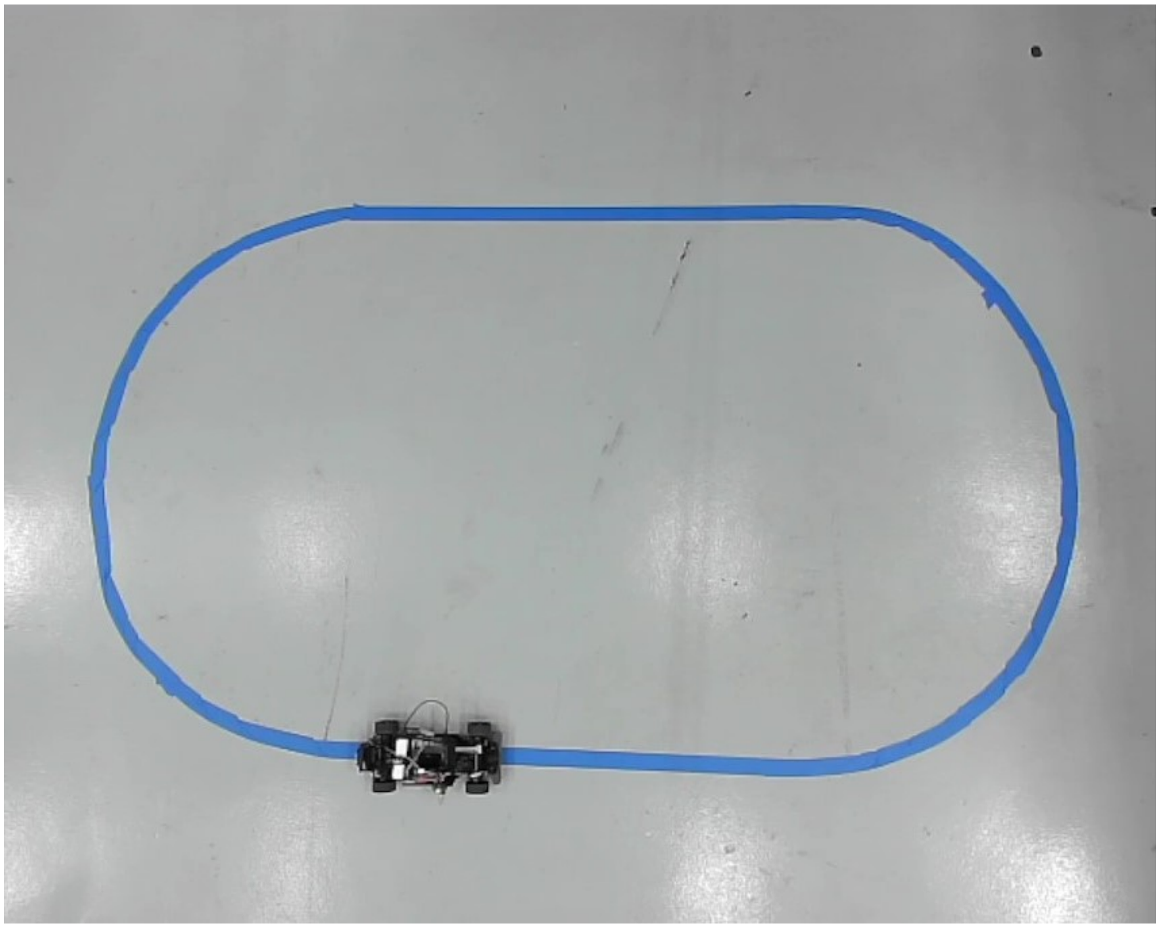


Figure 4.1: The JetRacer-4WS platform with the testing track.

# CHAPTER FOUR
# CASE STUDY: CALIBRATION WITH DOE

This chapter proposes a systematic and automatic calibration process to tune the time-triggered MPC implementation discussed above. The process is implemented for a smaller and larger sized track. Fig. 4.1 shows the JetRacer-4WS platform on the smaller track.

## 4.1 Calibration Process

There are many parameters in an MPC-based motion control system that can impact its control performance, including the length of the prediction horizon, distance between points in the path, and bounds for the control parameters. The calibration process presented in this paper focuses on the gains $Q_x$, $Q_u$, and $Q_d$ for the cost function (3.2a). Moreover, since only their relative magnitude impacts the solution of (3.2), $Q_x$ is assumed to be fixed, and only the calibrations of $Q_u$ and $Q_d$ are explored.

**Remark 3** *The JetRacer-4WS platform can function with either four-wheel steering or two-wheel steering models (by fixing the rear steering command at 0). For the two-wheel steering mode, the control input $u = \delta_f$ is a scalar, and so are $Q_u$ and $Q_d$. Therefore in this case, there are two calibration parameters to determine. On the other hand, for the four-wheel steering mode, the control input $u = \left[\delta_f, \delta_r\right]^T$ has a dimension of 2. By assuming both $Q_u$ and $Q_d$ are diagonal matrices, there are then 4 calibration parameters to determine in this case.*

There are several DoE methods to effectively populate the design space with calibrations to test. A full factorial design could work well with two variables in the two-wheel steering mode. However, for the four-wheel steering mode, when the number of test points is small, the full factorial design will place many points on the edges of the design space, causing the step sizes between points in the interior of the design space, where the optimal calibration is expected to be located, to be too coarse. An alternative DoE method that works better for systems with several variables is the latin hypercube [32]. It populates the design space by attempting to maximize the minimum

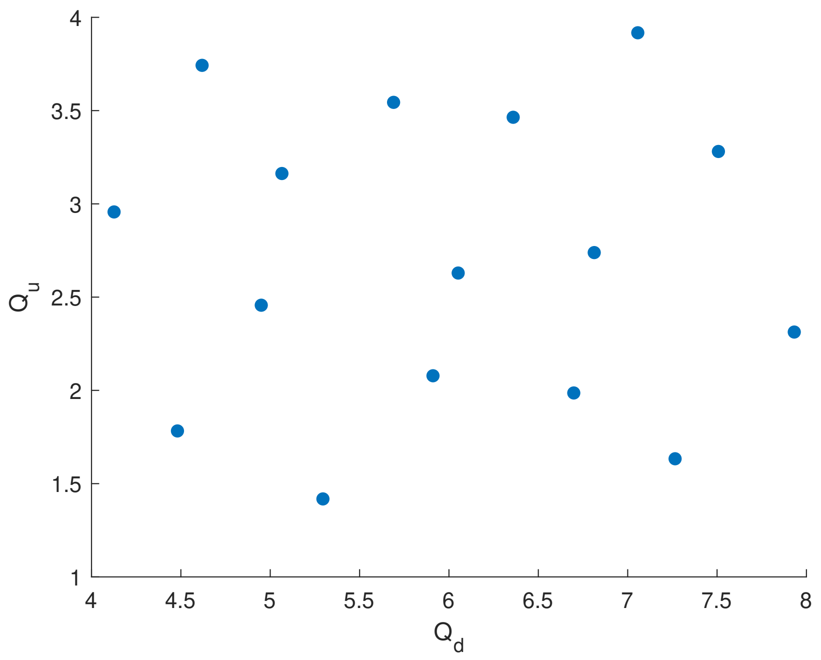


Figure 4.2: Latin hypercube calibration set for the small track two-wheel steering mode test.

distance between test points. Therefore in this paper, a latin hypercube feature included in the Python scipy library is used for the DoEs for both the two- and four-wheel steering modes. The resulting sets of calibrations for the smaller track are presented in Figs. 4.2 and 4.3, and the sets of calibrations for the larger track are presented in Figs. 4.4 and 4.5. All DoEs show reasonable distribution over the design space. Note that more points are generated in the four-wheel steering mode due to its larger design space.

In order to gather data for these calibration sets, the two- and four-wheel MPC controllers are implemented in separate scripts. The scripts automatically iterate through the DoEs, apply the calibration for a set time, and generate data files. Between each iteration, the first calibration (known to function reasonably well from hand calibration or

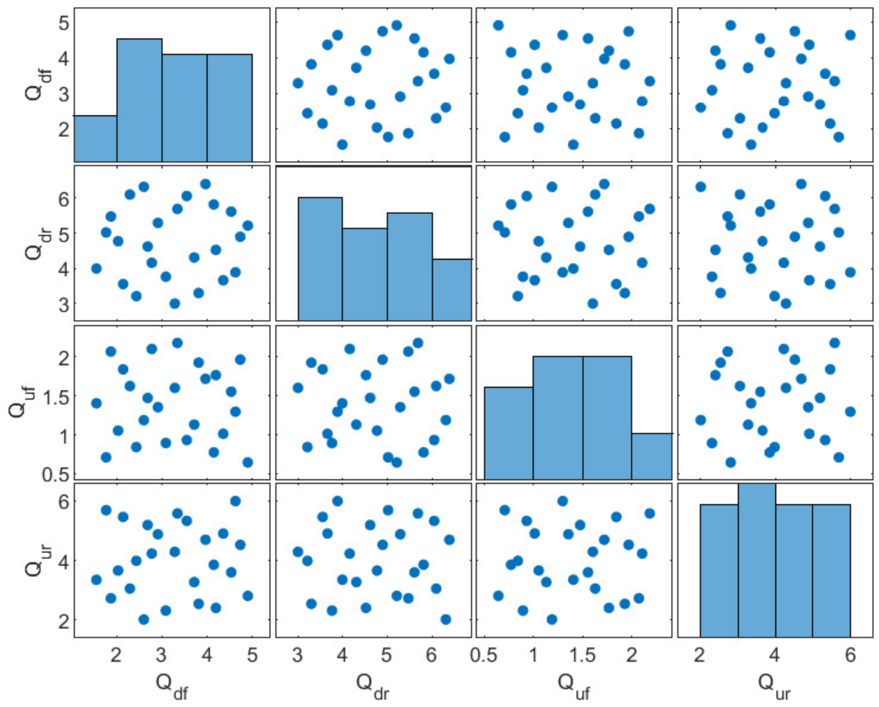


Figure 4.3: Latin hypercube calibration set for the small track four-wheel steering mode test.

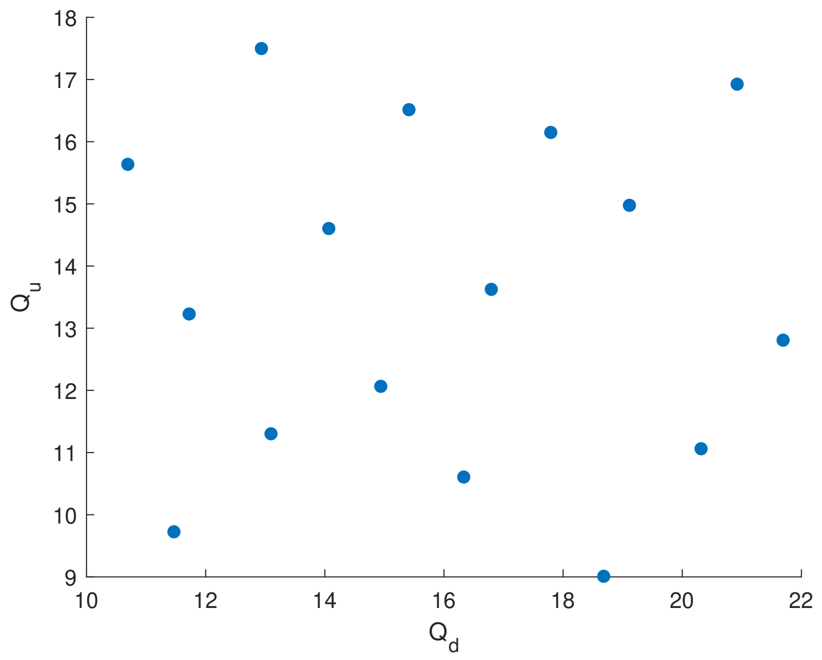


Figure 4.4: Latin hypercube calibration set for the large track two-wheel steering mode test.

previous DoE results) is reapplied to allow the system to recover if the previous DoE calibration performs poorly. In addition, data collection for a DoE calibration is ended early if the vehicle deviates from the desired path beyond a predefined threshold.

## 4.2 Small Track Test Results

Both the two- and four-wheel steering time-triggered MPC systems are calibrated using the calibration procedure discussed above. For the two-wheel mode, the DoE shown in Fig. 4.2 is used, with results shown in Table 4.1. For the four-wheel steering mode, the DoE shown in Fig. 4.3 is used, with results shown in Table 4.2. Note that in Tables 4.1 and 4.2, the RMSE and maximum error, both on lateral offset, are calculated for each calibration's best lap around the track. In order to determine the best performing

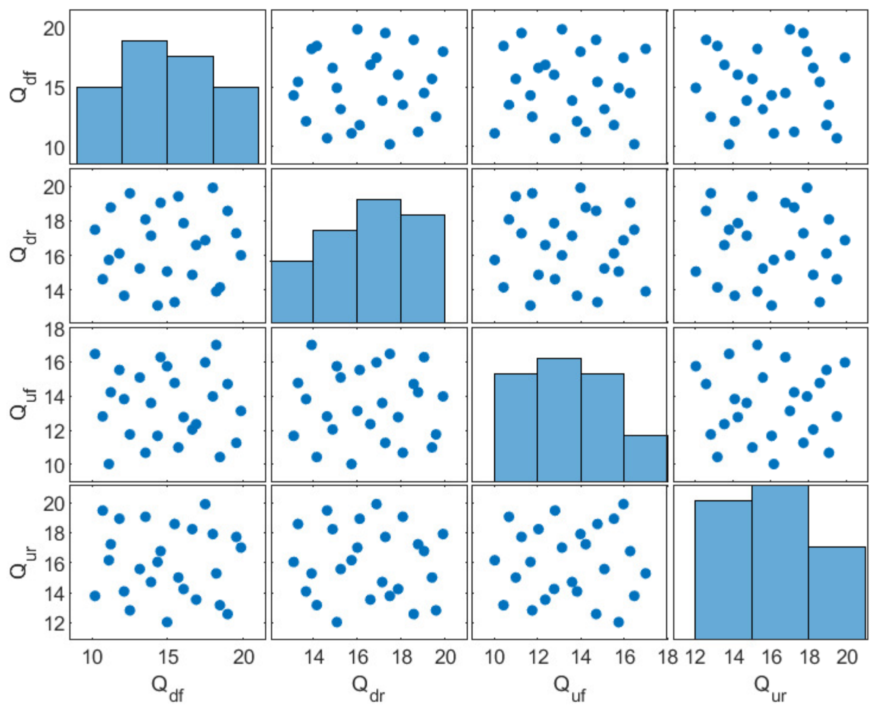


Figure 4.5: Latin hypercube calibration set for the large track four-wheel steering mode test.

Table 4.1: DoE results for two-wheel steering MPC for the small track.

| $Q_d$ | $Q_u$ | RMSE | Max Error | I |
|---|---|---|---|---|
| 5.69 | 3.54 | 0.092 | 0.182 | 4.45 |
| 7.27 | 1.63 | 0.085 | 0.180 | 4.27 |
| 4.48 | 1.78 | 0.069 | 0.147 | 3.48 |
| 6.05 | 2.63 | 0.082 | 0.165 | 4.01 |
| 4.13 | 2.96 | 0.078 | 0.156 | 3.80 |
| 6.81 | 2.74 | 0.083 | 0.174 | 4.15 |
| 6.36 | 3.46 | 0.078 | 0.150 | 3.73 |
| 5.91 | 2.08 | 0.073 | 0.160 | 3.74 |
| **5.29** | **1.42** | **0.065** | **0.139** | **3.27** |
| 5.07 | 3.16 | 0.071 | 0.150 | 3.56 |
| 7.06 | 3.92 | 0.096 | 0.191 | 4.67 |
| 4.95 | 2.46 | 0.062 | 0.153 | 3.40 |
| 7.93 | 2.31 | 0.070 | 0.146 | 3.49 |
| 4.62 | 3.74 | 0.078 | 0.208 | 4.51 |
| **6.70** | **1.99** | **0.060** | **0.128** | **3.02** |
| 7.51 | 3.28 | 0.079 | 0.162 | 3.89 |
| **5.60** | **2.20** | **0.058** | **0.124** | **2.94** |

calibrations, a simple equation for the cost index $I = \overline{RMSE} + \overline{MaxError}$ is created where $\overline{RMSE}$ and $\overline{MaxError}$ are normalized with respect to the smallest value. Note that in both Tables 4.1 and 4.2, the top performing calibrations based on the cost index are marked in bold. In addition, Fig. 4.6 plots the path tracking performance for both two- and four-wheel steering MPC with their respective best calibration, i.e., $Q_d = 5.60$ and $Q_u = 2.20$ for two-wheel steering MPC and $Q_{df} = 1.55$, $Q_{dr} = 4.00$, $Q_{uf} = 1.40$ and $Q_{ur} = 3.35$ for four-wheel steering MPC.

The test results presented in Table 4.1, Table 4.2, and Fig. 4.6 show that the best calibrations from the four-wheel steering time-triggered MPC test were able to outperform

Table 4.2: DoE results for four-wheel steering MPC for the small track.

| $Q_{df}$ | $Q_{dr}$ | $Q_{uf}$ | $Q_{ur}$ | RMSE | Max Error | I |
|---|---|---|---|---|---|---|
| 3.20 | 5.00 | 1.60 | 4.80 | 0.048 | 0.092 | 2.28 |
| 3.28 | 3.00 | 1.60 | 4.29 | 0.058 | 0.108 | 2.73 |
| **2.03** | **4.77** | **1.05** | **3.66** | **0.046** | **0.081** | **2.08** |
| 2.29 | 6.10 | 1.63 | 3.06 | 0.048 | 0.093 | 2.29 |
| 4.53 | 5.61 | 1.55 | 3.60 | 0.063 | 0.146 | 3.32 |
| **1.77** | **5.02** | **0.71** | **5.68** | **0.047** | **0.075** | **2.03** |
| 3.55 | 6.05 | 0.93 | 5.32 | 0.057 | 0.103 | 2.63 |
| 1.87 | 5.47 | 2.07 | 2.73 | 0.053 | 0.117 | 2.74 |
| 2.78 | 4.16 | 2.10 | 4.23 | 0.056 | 0.120 | 2.83 |
| 2.60 | 6.31 | 1.19 | 2.02 | 0.046 | 0.097 | 2.31 |
| **1.55** | **4.00** | **1.40** | **3.35** | **0.046** | **0.074** | **2.01** |
| 2.14 | 3.55 | 1.84 | 5.45 | 0.053 | 0.110 | 2.63 |
| 3.96 | 6.39 | 1.72 | 4.69 | 0.067 | 0.143 | 3.39 |
| 4.15 | 5.81 | 0.77 | 3.85 | 0.053 | 0.101 | 2.51 |
| 3.09 | 3.76 | 0.89 | 2.32 | 0.052 | 0.106 | 2.56 |
| 3.34 | 5.69 | 2.18 | 5.57 | 0.059 | 0.113 | 2.80 |
| 2.91 | 5.29 | 1.35 | 4.87 | 0.054 | 0.095 | 2.45 |
| 2.69 | 4.62 | 1.47 | 5.18 | 0.056 | 0.111 | 2.71 |
| 4.63 | 3.89 | 1.30 | 5.99 | 0.048 | 0.080 | 2.12 |
| 3.71 | 4.31 | 1.13 | 3.27 | 0.047 | 0.092 | 2.26 |
| 3.81 | 3.30 | 1.93 | 2.54 | 0.060 | 0.127 | 3.01 |
| 4.36 | 3.66 | 1.01 | 4.90 | 0.059 | 0.100 | 2.63 |
| 4.19 | 4.53 | 1.77 | 2.41 | 0.061 | 0.147 | 3.31 |
| 4.73 | 4.90 | 1.97 | 4.52 | 0.059 | 0.155 | 3.36 |
| 2.43 | 3.21 | 0.84 | 3.98 | 0.049 | 0.093 | 2.32 |
| 4.90 | 5.21 | 0.65 | 2.81 | 0.051 | 0.105 | 2.52 |

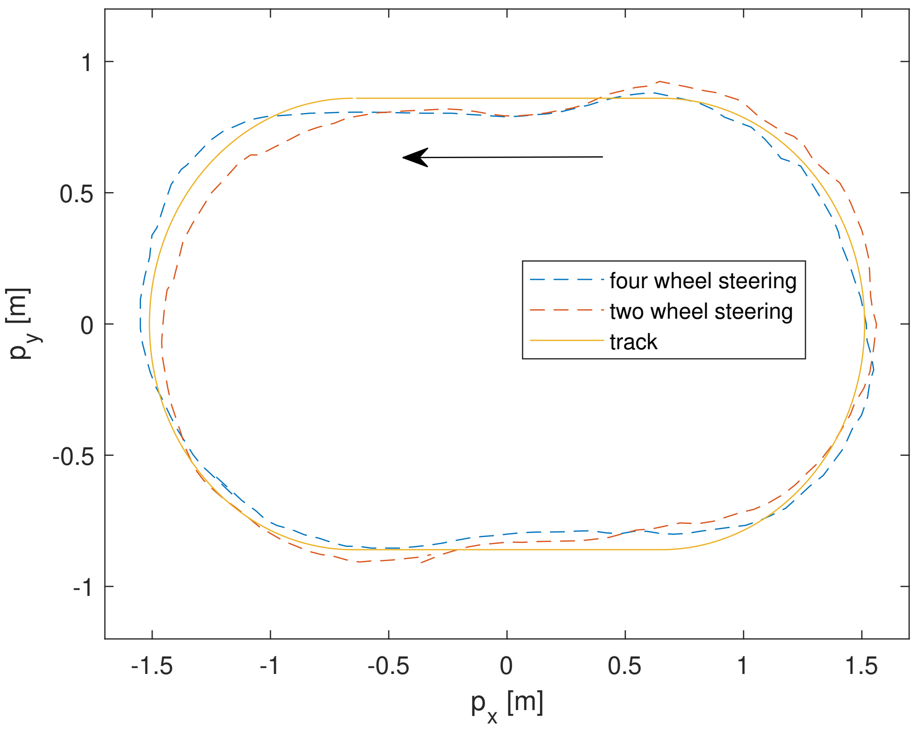


Figure 4.6: Path tracking performance for two- and four-wheel steering MPC systems on the small track.

the best calibrations from the two-wheel steering test. For example, the minimum RMSE values achieved by the four-wheel steering system show a 21% improvement over the two-wheel steering system. It is worth noting that, as shown in Fig. 4.6, the test track used in this paper has a small turning radius that approaches the maximum achievable by the two-wheel steering geometry. Therefore, the results suggest that implementing independent four-wheel steering MPC could see performance improvements in parking lots and other challenging driving scenarios that require high agility. However, with fully independent control of the front and rear steering angles, the complexity of (3.2) increases. Furthermore, the solution of (3.2) is also impacted by additional error sources from the rear wheel steering mechanism. Therefore, for most normal driving conditions, a two-wheel steering MPC may be preferable.

**Remark 4** *An alternative control strategy for four-wheel steering that may be worth further investigation is to determine the rear wheel steering angle based on the front steering angle. In other words, the ratio between $\delta_r$ and $\delta_f$ is predetermined as a function of vehicle speed, and MPC is set up to calculate $\delta_f$ only. Such a strategy could reduce MPC complexity while maintaining four-wheel steering improvements in agility at low speed and cornering stability at high speed [33].*

## 4.3 Large Track Test Results

In addition to testing the JetRacer-4WS platform on the small track with a distinct focus on maneuverability, path tracking results are also presented for testing on a larger track with a higher velocity set point ($1.6m/s$). Due to the significantly different driving style required for the larger radius turns and longer straight sections, recalibration of the cost function weights is necessary to achieve optimal performance. As a result, the DoEs for the larger track are generated using a different set of ranges for the calibration

Table 4.3: DoE results for two-wheel steering MPC for the large track.

| $Q_d$ | $Q_u$ | RMSE | Max Error | I |
|---|---|---|---|---|
| 11.47 | 9.73 | 0.057 | 0.146 | 2.65 |
| 10.69 | 15.64 | 0.075 | 0.180 | 3.37 |
| 21.69 | 12.81 | 0.076 | 0.157 | 3.15 |
| 14.94 | 12.07 | 0.070 | 0.171 | 3.17 |
| 20.92 | 16.93 | 0.090 | 0.214 | 4.03 |
| 19.11 | 14.98 | 0.089 | 0.183 | 3.68 |
| 12.94 | 17.50 | 0.086 | 0.187 | 3.66 |
| 13.10 | 11.30 | 0.061 | 0.149 | 2.77 |
| **11.72** | **13.23** | **0.061** | **0.103** | **2.28** |
| **20.32** | **11.06** | **0.050** | **0.110** | **2.15** |
| 16.79 | 13.62 | 0.065 | 0.180 | 3.18 |
| **18.68** | **9.01** | **0.057** | **0.109** | **2.28** |
| 15.41 | 16.51 | 0.063 | 0.164 | 2.97 |
| 16.33 | 10.61 | 0.062 | 0.157 | 2.86 |
| 17.79 | 16.15 | 0.064 | 0.128 | 2.60 |
| 14.07 | 14.60 | 0.053 | 0.121 | 2.31 |
| 15.60 | 13.20 | 0.058 | 0.147 | 2.69 |

variables that increases cost for large steering angles and fast changes in steering angle. The results for the two-wheel steering DoE shown in Fig. 4.4 is presented in Table 4.3, and the results for the four-wheel steering DoE shown in Fig. 4.5 is presented in Table 4.4. Fig. 4.7 shows the path tracking performance of the best calibrations, based on the calculated cost index values, from the two- and four-wheel steering tests, i.e., $Q_d = 20.32$ and $Q_u = 11.06$ for two-wheel steering MPC and $Q_{df} = 13.90$, $Q_{dr} = 17.14$, $Q_{uf} = 13.60$ and $Q_{ur} = 14.72$ for four-wheel steering MPC.

The JetRacer-4WS test results from the large track show that the four-wheel steering MPC slightly outperformed the two-wheel steering MPC. For example, the

Table 4.4: DoE results for four-wheel steering MPC for the large track.

| $Q_{df}$ | $Q_{dr}$ | $Q_{uf}$ | $Q_{ur}$ | RMSE | Max Error | I |
|---|---|---|---|---|---|---|
| **14.00** | **16.00** | **12.00** | **14.00** | **0.053** | **0.120** | **2.32** |
| 12.50 | 19.58 | 11.76 | 12.83 | 0.063 | 0.104 | 2.33 |
| 14.95 | 15.08 | 15.75 | 12.05 | 0.060 | 0.140 | 2.66 |
| 17.98 | 19.90 | 13.98 | 17.91 | 0.073 | 0.158 | 3.11 |
| 16.87 | 16.60 | 12.36 | 13.54 | 0.066 | 0.115 | 2.51 |
| **11.81** | **16.11** | **15.53** | **18.93** | **0.050** | **0.101** | **2.05** |
| 17.47 | 16.88 | 15.97 | 19.90 | 0.062 | 0.138 | 2.67 |
| 16.05 | 17.86 | 12.77 | 14.25 | 0.069 | 0.157 | 3.02 |
| 13.53 | 18.08 | 10.68 | 19.06 | 0.070 | 0.178 | 3.26 |
| 12.12 | 13.68 | 13.83 | 14.09 | 0.070 | 0.149 | 2.95 |
| 18.21 | 13.93 | 17.00 | 15.29 | 0.079 | 0.180 | 3.46 |
| 11.24 | 18.77 | 14.23 | 17.23 | 0.071 | 0.152 | 3.01 |
| 14.33 | 13.12 | 11.67 | 16.05 | 0.063 | 0.135 | 2.67 |
| 13.16 | 15.25 | 15.09 | 15.58 | 0.075 | 0.138 | 2.93 |
| **13.90** | **17.14** | **13.60** | **14.72** | **0.051** | **0.096** | **2.02** |
| 10.71 | 14.63 | 12.81 | 19.47 | 0.072 | 0.173 | 3.24 |
| 11.12 | 15.74 | 10.02 | 16.17 | 0.074 | 0.138 | 2.91 |
| 15.70 | 19.40 | 11.00 | 15.01 | 0.066 | 0.156 | 2.94 |
| 18.96 | 18.57 | 14.71 | 12.58 | 0.080 | 0.178 | 3.46 |
| 16.62 | 14.88 | 12.05 | 18.23 | 0.084 | 0.190 | 3.65 |
| 19.53 | 17.28 | 11.26 | 17.72 | 0.073 | 0.158 | 3.11 |
| 19.83 | 16.01 | 13.13 | 17.01 | 0.076 | 0.192 | 3.52 |
| 18.45 | 14.17 | 10.43 | 13.18 | 0.076 | 0.144 | 3.01 |
| 15.46 | 13.31 | 14.77 | 18.58 | 0.073 | 0.151 | 3.03 |
| 10.20 | 17.49 | 16.47 | 13.79 | 0.064 | 0.154 | 2.89 |
| 14.53 | 19.04 | 16.28 | 16.77 | 0.080 | 0.164 | 3.30 |

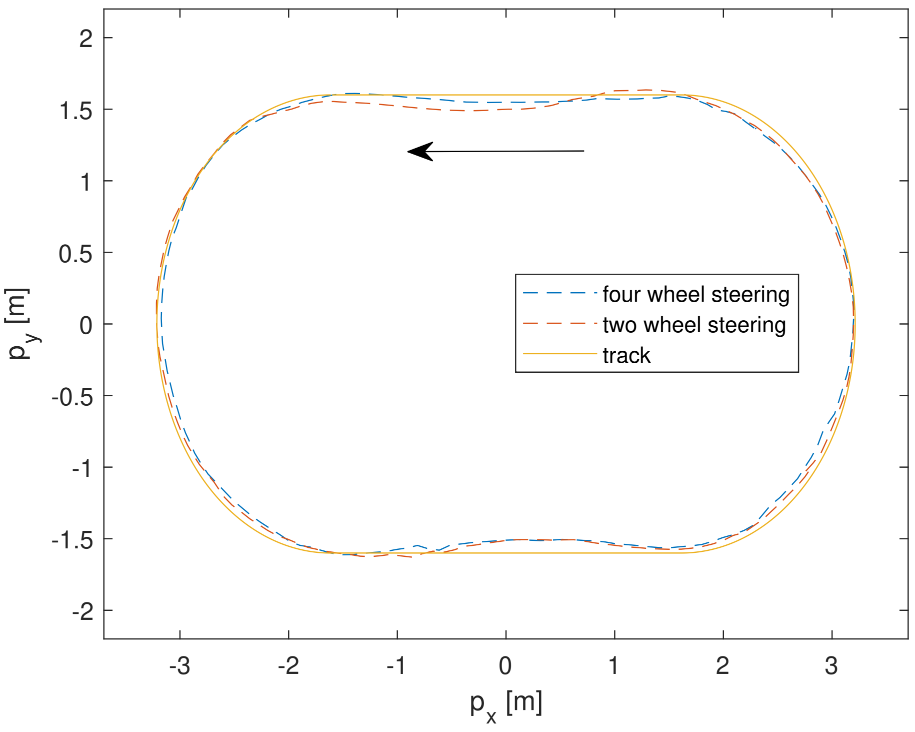


Figure 4.7: Path tracking performance for two- and four-wheel steering MPC systems on the large track.

minimum RMSE achieved by the two- and four-wheel steering systems are identical, so the four-wheel steering system's lower cost indexes stem primarily from the 12% improvement between the two lowest maximum offset values. While these results show that the four-wheel steering path tracking controller is effective, it is worth noting the results also show that the improvement over the two-wheel steering controller is significantly reduced compared to the small track test results. This is due to the decreased importance of the four-wheel steering system's superior maneuverability for the larger radius turns and longer straight sections.

# CHAPTER FIVE

# CASE STUDY: EVENT-TRIGGERED MPC

To further demonstrate the capability of the JetRacer-4WS platform for cost-effective testing and evaluation of various motion control methods, event-triggered MPC is demonstrated on the small track. Event-triggered MPC has been explored in literature [7, 20–27] as an attempt to reduce MPC computation without major degradation on control performance. One of the drawbacks of relevant literature is the lack of experimental validation. This chapter implements event-triggered MPC algorithm from [7] and performs experiment validation using the proposed JetRacer-4WS.

## 5.1 Event-Triggered MPC Path Following Control

In order to test the JetRacer-4WS platform with event-triggered MPC, the MPC architecture presented in Chapter Three is modified. To determine if the system should iterate through the current control sequence or calculate a new control sequence, an event-trigger variable $e$ is defined as follows:

$$e = \begin{cases} 1 & \text{if } Y > \sigma \text{ or } k > k_{max} \\ 0 & \text{Otherwise} \end{cases}, \tag{5.1}$$

The value of $e$ depends on two conditions. The event trigger is set if the current step number $k$ exceeds a threshold $k_{max}$, where $k_{max}$ must be less than the prediction horizon length $p$. The trigger also sets if the vehicle's displacement from the nearest point in the desired path $Y$ exceeds a predefined threshold $\sigma$.

The event-trigger MPC algorithm implemented is illustrated in Algorithm 5.1 and Fig. 5.1, which shows how the value of $e$ is used to determine the source of the control output.

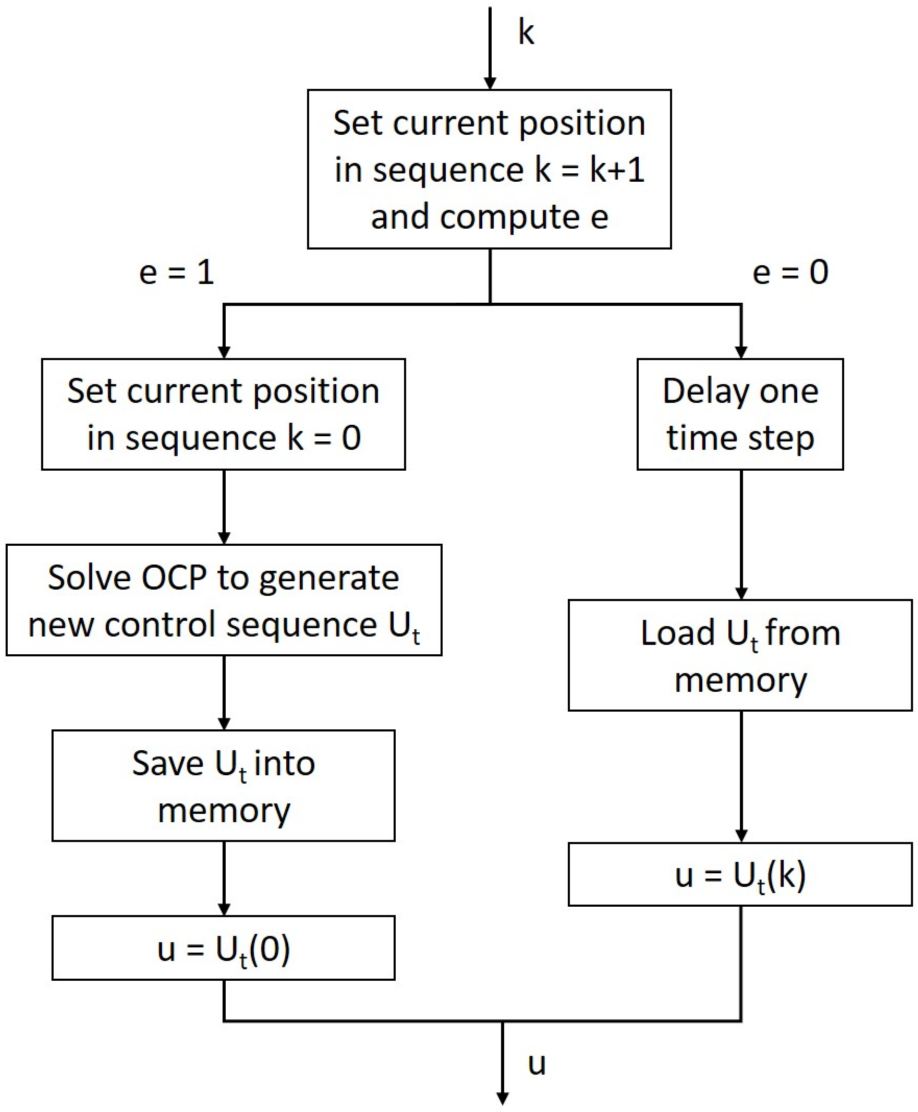

k
Set current position in sequence k = k+1 and compute e
e = 1
e = 0
Set current position in sequence k = 0
Delay one time step
Solve OCP to generate new control sequence $U_t$
Load $U_t$ from memory
Save $U_t$ into memory
$u = U_t(k)$
$u = U_t(0)$
u

Figure 5.1: A representation of the event-triggered MPC algorithm.

**Algorithm 5.1:** Event-Triggered NMPC [7]

```
1.  procedure eNMPC(x, k, $U_{t_1}$, $X_{t_1}$)
2.      $k \leftarrow k+1$
3.      $e \leftarrow$ computing(5.1)
4.      if $e = 1$ then
5.          $k \leftarrow 0$
6.          $(X_t, U_t) \leftarrow$ Solving OCP (3.2)
7.          $u \leftarrow U_t(1)$
8.          $U_{t_1} \leftarrow U_t$
9.          $X_{t_1} \leftarrow X_t$
10.     else
11.         $u \leftarrow U_{t_1}(k+1)$
12.     end
13.     return $u, k, U_{t_1}, Z_{t_1}$
14. end procedure
```

## 5.2 Small Track Test Results

Using the best performing calibrations from the small track two- and four-wheel steering time-triggered MPC tests, additional results are generated for event-triggered MPC with varying triggering thresholds ($\sigma$). For these tests, the same metrics are used to determine path following performance, including RMSE, maximum offset, and the cost index calculated from these values. In addition, relative computational load can be compared using the triggering frequency, which is the percentage of actions in which the control system calculates a new control sequence. Table 5.1 and Fig. 5.2 show the results of the two-wheel steering event-triggered MPC, and Table 5.2 and Fig. 5.3 show the results of the four-wheel steering event-triggered MPC. Note that the first row of both tables uses a triggering threshold of zero, which behaves like a time-triggered system. In most cases, the results confirm the expected trend that increasing the triggering threshold

Table 5.1: Small track results for two-wheel steering event triggered MPC.

| $\sigma$ | RMSE | Max Error | I | Frequency |
|---|---|---|---|---|
| 0.000 | 0.067 | 0.115 | 2.48 | 100.0 |
| 0.015 | 0.077 | 0.129 | 2.83 | 91.5 |
| 0.025 | 0.087 | 0.176 | 3.48 | 89.6 |
| 0.035 | 0.145 | 0.259 | 5.49 | 84.9 |

Table 5.2: Small track results for four-wheel steering event triggered MPC.

| $\sigma$ | RMSE | Max Error | I | Frequency |
|---|---|---|---|---|
| 0.000 | 0.048 | 0.120 | 2.14 | 100.0 |
| 0.015 | 0.060 | 0.105 | 2.25 | 91.5 |
| 0.025 | 0.072 | 0.110 | 2.53 | 90.0 |
| 0.035 | 0.066 | 0.133 | 2.64 | 87.7 |

will reduce computational load (lower triggering frequency) while negatively affecting path following performance metrics (higher RMSE, maximum offset, and cost index).

## 5.3 Large Track Test Results

Two- and four-wheel steering event-triggered MPC test results are also provided using the large track. Table 5.3 and Fig. 5.4 show the results of the two-wheel steering event-triggered MPC, and Table 5.4 and Fig. 5.5 show the results of the four-wheel steering event-triggered MPC. The results continue to show that four-wheel steering MPC provides a performance improvement over two-wheel steering MPC, and implementing event-triggering provides the ability to reduce computational load in exchange for reduced performance.

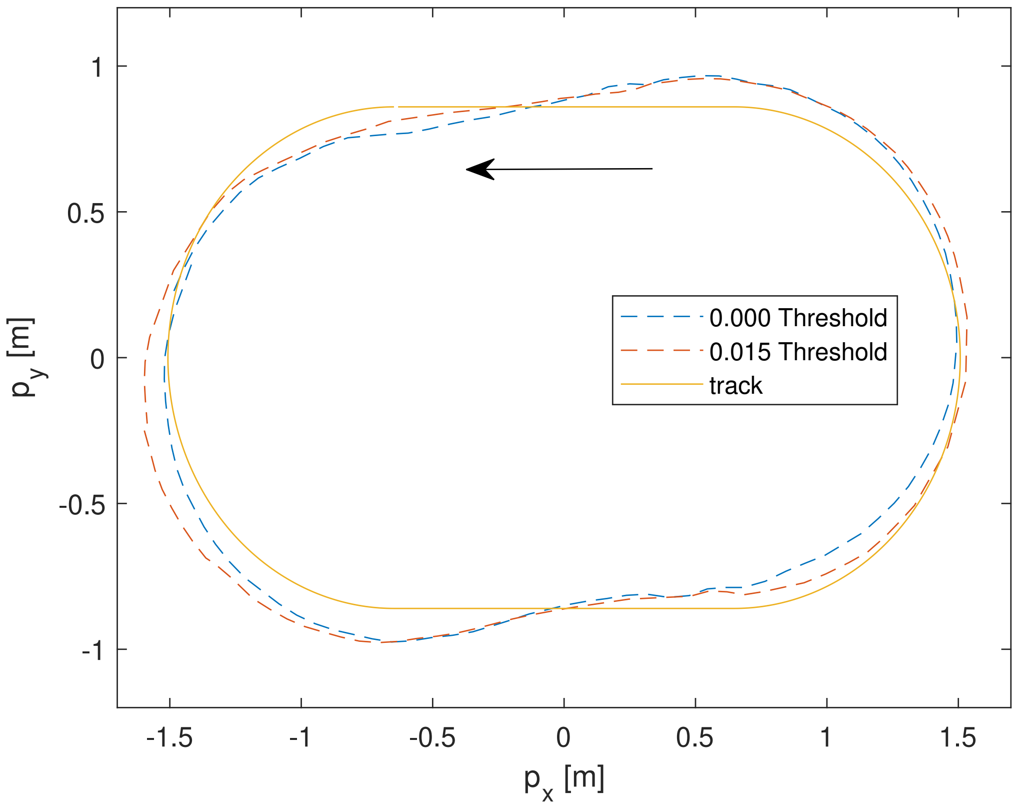


Figure 5.2: Small track two-wheel steering path tracking performance for event-triggered MPC.

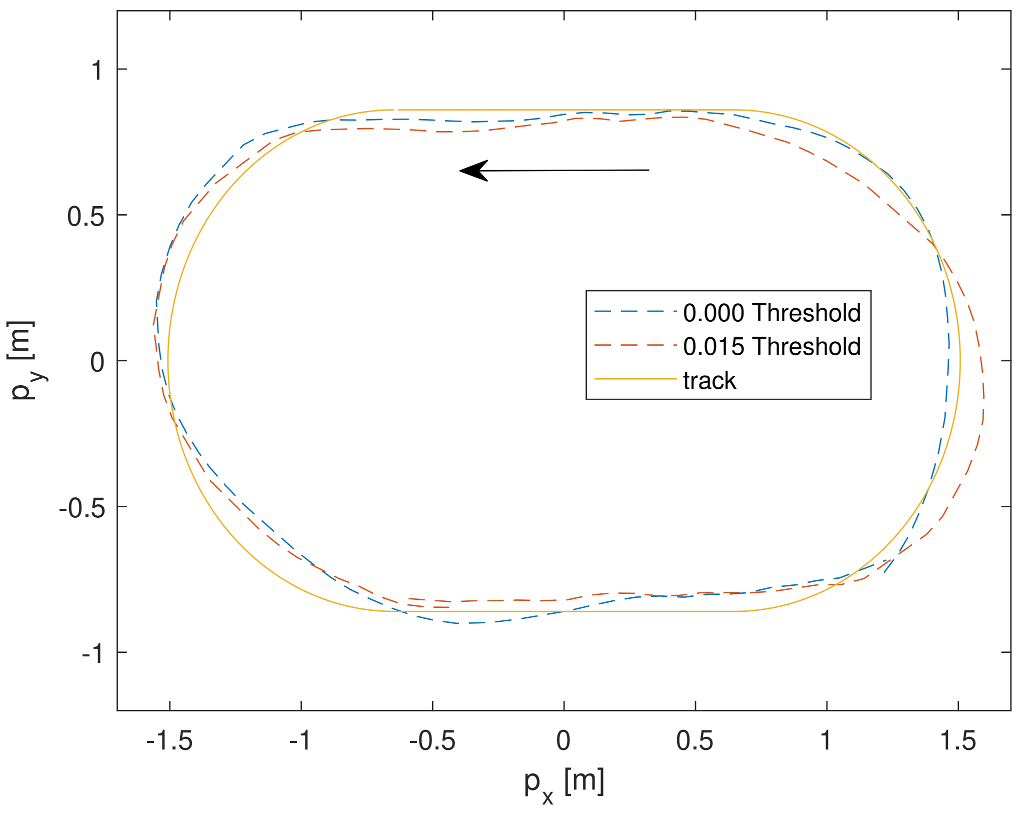


Figure 5.3: Small track four-wheel steering path tracking performance for event-triggered MPC.

Table 5.3: Large track results for two-wheel steering event triggered MPC.

| $\sigma$ | RMSE | Max Error | I | Frequency |
|---|---|---|---|---|
| 0.000 | 0.039 | 0.110 | 2.34 | 100.0 |
| 0.015 | 0.088 | 0.192 | 4.62 | 92.2 |
| 0.025 | 0.095 | 0.174 | 4.58 | 93.4 |
| 0.035 | 0.123 | 0.268 | 6.44 | 90.1 |

**Remark 5** *For the two-wheel steering test, the cost index of the 0.015 meter threshold test was similar to the 0.025 meter threshold test, which does not exactly align with the expected trend of gradually decreasing performance as the triggering threshold increases. In addition, the triggering frequencies for the non-zero triggering thresholds were similar instead of gradually decreasing. While this does not align with the expectation that a larger triggering threshold would require recalculation of the control sequence less frequently, the impact of the reduced performance can counteract this trend, resulting in the observed behavior. Also, the presence of random error in a physical vehicle system can produce unexpected results. In the future, multiple tests will be run to eliminate the impact of noise. The performance of the event-triggered MPC systems could likely be improved by reducing the time step value over the prediction horizon, although this would require recalibration of the cost function weights to approach optimal performance.*

Table 5.4: Large track results for four-wheel steering event triggered MPC.

| $\sigma$ | RMSE | Max Error | I | Frequency |
|---|---|---|---|---|
| 0.000 | 0.037 | 0.086 | 2.00 | 100.0 |
| 0.015 | 0.048 | 0.107 | 2.53 | 80.6 |
| 0.025 | 0.068 | 0.133 | 3.39 | 83.6 |
| 0.035 | 0.096 | 0.203 | 4.95 | 82.5 |

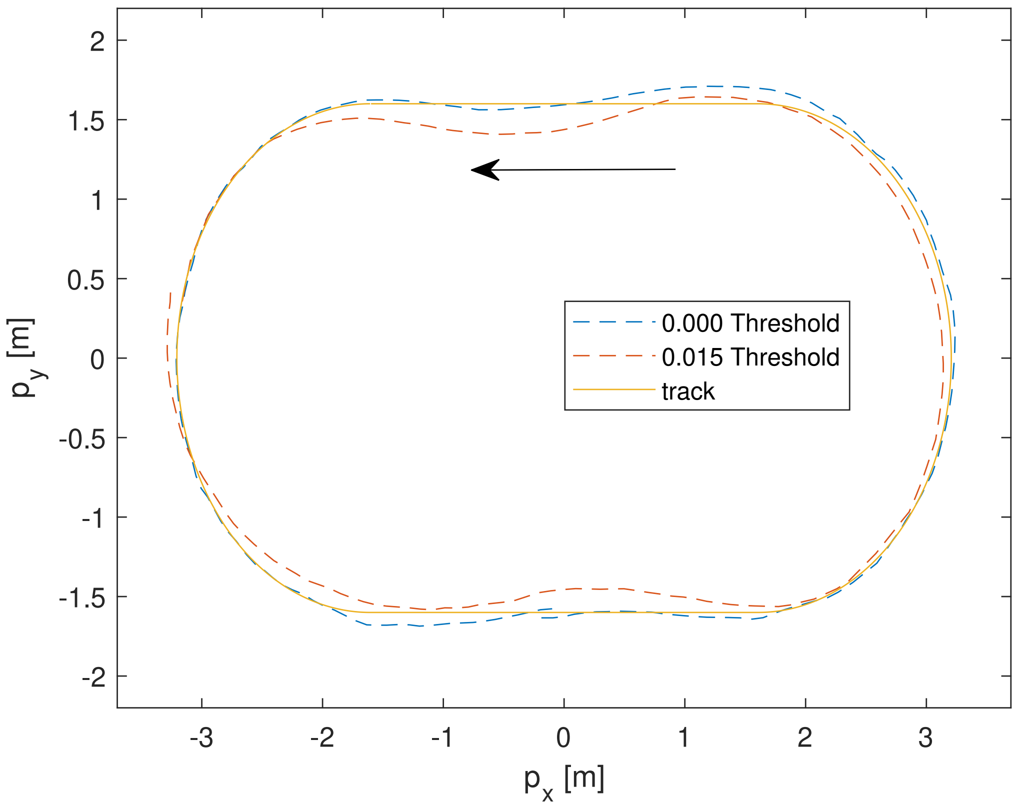


Figure 5.4: Large track two-wheel steering path tracking performance for event-triggered MPC.

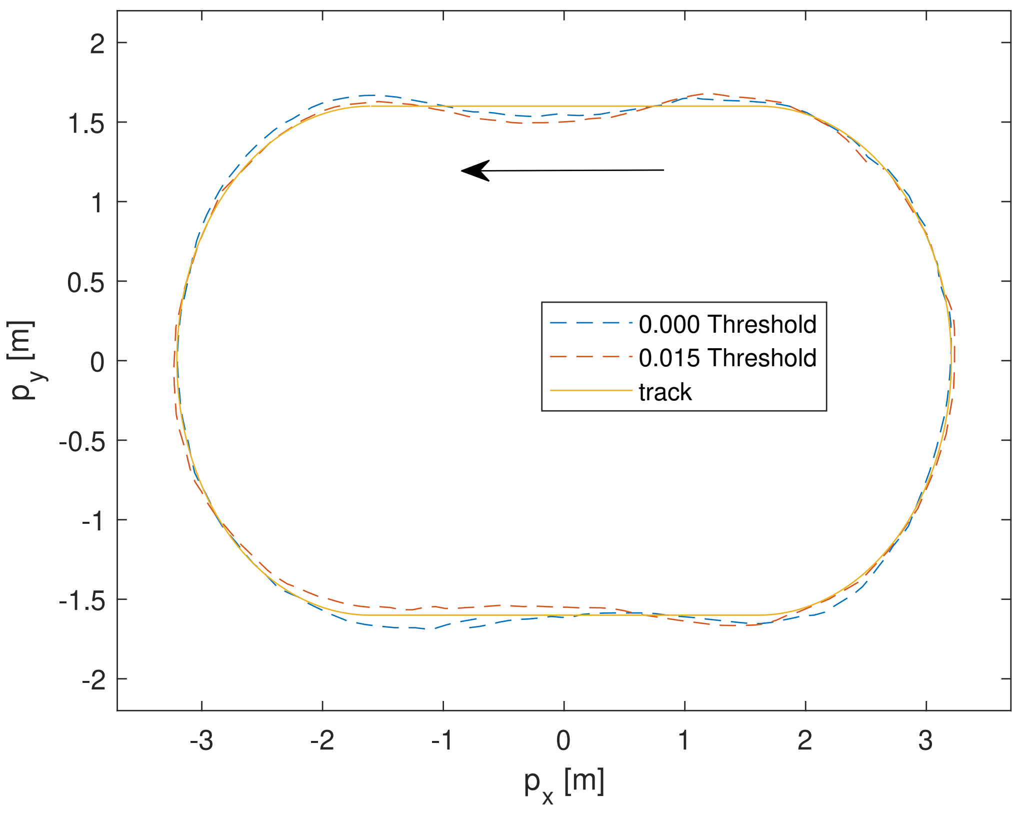


Figure 5.5: Large track four-wheel steering path tracking performance for event-triggered MPC.

# CHAPTER SIX

# CONCLUSION

This paper presents the JetRacer-4WS: a scale vehicle platform with four-wheel steering capability for testing autonomous vehicle motion controls. Introducing a scale vehicle into the controls development process is meant to reduce the time and expense associated with instrumenting and testing using a full-size vehicle. The JetRacer-4WS platform provides advantages over existing scale vehicle platforms for MPC path tracking control research based on its four-wheel steering capability, as well as its isolation of the path tracking portion of AV operation by eliminating onboard environment detection and real-time path planning. Model predictive control for both two- and four-wheel steering are implemented using a kinematic vehicle model of lateral motion. Test sequences of cost function tuning weights are generated using a latin hypercube-based Design of Experiments method to intelligently fill the design spaces. The tests are run for two- and four-wheel steering time-triggered MPC systems on a small and large track, and the results show that the four-wheel steering system significantly outperforms the two-wheel system in driving scenarios where its increased maneuverability provides a distinct advantage. In less extreme driving scenarios, the test results from the large track show that the four-wheel steering system performs only slightly better than the two-wheel steering system, as vehicle maneuverability is a less important factor in the path tracking control performance. Additional testing on the small track with event-triggered MPC shows that the JetRacer-4WS platform can be used to evaluate the effect of variations in the implemented control system. As the triggering threshold is increased, the computational load of the MPC system decreases, but the path tracking performance is reduced. Future work includes (1) testing path tracking performance of additional motion control methods,

(2) continuing to find and reduce sources of error contributing to path tracking error, and (3) modifying the scale vehicle platform positioning system to allow for even higher speed testing over a larger area with more variety of driving maneuvers.

## REFERENCES

[1] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium*, (Seoul, Korea), pp. 1094–1099, June 28–July 1, 2015.

[2] S. Di Cairano, H. E. Tseng, D. Bernardini, and A. Bemporad, "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain," *IEEE Trans. Control Syst. Tech.*, vol. 21, no. 4, pp. 1236–1248, 2012.

[3] R. Yu, H. Guo, Z. Sun, and H. Chen, "MPC-based regional path tracking controller design for autonomous ground vehicles," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2510–2515, IEEE, 2015.

[4] X. Sun, Y. Zhang, Y. Cai, and X. Tian, "Compensated deadbeat predictive current control considering disturbance and vsi nonlinearity for in-wheel pmsms," *IEEE/ASME Transactions on Mechatronics*, 2022.

[5] I. Jammeli, A. Chemori, H. Moon, S. Elloumi, and S. Mohammed, "An assistive explicit model predictive control framework for a knee rehabilitation exoskeleton," *IEEE/ASME Transactions on Mechatronics*, 2021.

[6] Q. Gong, J. Xu, J. Ye, H. Feng, and A. Shen, "Nonlinear model predictive control for premixed turbocharged natural gas engine," *IEEE/ASME Transactions on Mechatronics*, 2021.

[7] J. Chen and Z. Yi, "Comparison of event-triggered model predictive control for autonomous vehicle path tracking," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 808–813, IEEE, 2021.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[9] M. Kim, D. Lee, J. Ahn, M. Kim, and J. Park, "Model predictive control method for autonomous vehicles using time-varying and non-uniformly spaced horizon," *IEEE Access*, vol. 9, pp. 86475–86487, 2021.

[10] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H.-C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: An open, inexpensive and flexible platform for autonomy

education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, 2017.

[11] S. Karaman, A. Anders, M. Boulet, J. Connor, K. Gregson, W. Guerra, O. Guldner, M. Mohamoud, B. Plancher, R. Shin, and J. Vivilecchia, "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at mit," in *2017 IEEE Integrated STEM Education Conference (ISEC)*, pp. 195–203, 2017.

[12] X. Wu and A. Eskandarian, "An improved small-scale connected autonomous vehicle platform," in *Dynamic Systems and Control Conference*, vol. 59148, p. V001T04A003, 2019.

[13] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, *et al.*, "F1/10: An open-source autonomous cyber-physical platform," *arXiv preprint arXiv:1901.08567*, 2019.

[14] A. Seth, A. James, and S. C. Mukhopadhyay, "1/10th scale autonomous vehicle based on convolutional neural network," *Int. J. Smart Sens. Intell. Syst*, vol. 13, no. 1, pp. 1–17, 2020.

[15] "Jetracer." `https://github.com/NVIDIA-AI-IOT/jetracer`, 2019.

[16] M. Robotics. Accessed Apr. 11, 2022.

[17] J.-A. Yang and C.-H. Kuo, "Integrating vehicle positioning and path tracking practices for an autonomous vehicle prototype in campus environment," *Electronics*, vol. 10, no. 21, 2021.

[18] S. Park, S. Ryu, J. Lim, and Y.-S. Lee, "A real-time high-speed autonomous driving based on a low-cost rtk-gps," *Journal of Real-Time Image Processing*, vol. 18, 08 2021.

[19] "Bill of materials." https://f1tenth.org/build.html.

[20] F. D. Brunner, W. Heemels, and F. Allgöwer, "Robust event-triggered MPC with guaranteed asymptotic bound and average sampling rate," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 5694–5709, 2017.

[21] H. Li and Y. Shi, "Event-triggered robust model predictive control of continuous-time nonlinear systems," *Automatica*, vol. 50, no. 5, pp. 1507–1513, 2014.

[22] S. Huang and J. Chen, "Event-triggered model predictive control for autonomous vehicle with rear steering," *SAE Technical Paper*, no. 2022-01-0877, 2022.

[23] R. Badawi and J. Chen, “Enhancing enumeration-based model predictive control for dc-dc boost converter with event-triggered control,” in *2022 European Control Conference*, (London, UK), July 12–15, 2022.

[24] H. Li, W. Yan, and Y. Shi, “Triggering and control codesign in self-triggered model predictive control of constrained systems: With guaranteed performance,” *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 4008–4015, 2018.

[25] C. Liu, H. Li, Y. Shi, and D. Xu, “Codesign of event trigger and feedback policy in robust model predictive control,” *IEEE Transactions on Automatic Control*, vol. 65, no. 1, pp. 302–309, 2019.

[26] J. Chen, X. Meng, and Z. Li, “Reinforcement learning-based event-triggered model predictive control for autonomous vehicle path following,” in *2022 American Control Conference*, (Atlanta, GA), June 8–10, 2022.

[27] R. Badawi and J. Chen, “Performance evaluation of event-triggered model predictive control for boost converter,” in *2022 IEEE Vehicle Power and Propulsion Conference*, (Merced, CA), November 1–4, 2022.

[28] R. Rajamani, *Vehicle Dynamics and Control.* Mechanical Engineering Series, Springer US, 2012.

[29] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, “Predictive active steering control for autonomous vehicle systems,” *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[30] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, “MPC-based approach to active steering for autonomous vehicle systems,” *International Journal of Vehicle Autonomous Systems*, vol. 3, pp. 265–291, 2005.

[31] F. W. Isen, “Dsp for matlab and labview i: Fundamentals of discrete signal processing,” in *Synthesis Lectures on Signal Processing*, pp. 169–188, 2009.

[32] F. A. C. Viana, “A tutorial on latin hypercube design of experiments,” *Quality and Reliability Engineering International*, vol. 32, pp. 1975 – 1985, 2016.

[33] I. Besselink, T. Veldhuizen, and H. Nijmeijer, “Improving yaw dynamics by feedforward rear wheel steering,” in *2008 IEEE Intelligent Vehicles Symposium*, pp. 246–250, 2008.