Development of a Four-Wheel Steering Scale Vehicle for Research and Education on Autonomous Vehicle Motion Control

Christopher Rother, Zhaodong Zhou, and Jun Chen, *Senior Member, IEEE*

***Abstract*—Autonomous vehicle motion control development requires testing and evaluation at all stages of the process. The development phase involving the instrumentation and operation of a full-size vehicle can be especially costly. Scale vehicles have been developed in the literature to serve as a cost-effective transition from testing in a simulation environment to a physical system. However, the existing scale vehicle platforms do not support four-wheel steering and cannot isolate the performance of motion control algorithms from other modules such as perception and path planning. This letter closes this gap by proposing a new scale vehicle platform, called JetRacer-4WS, based on the open-source JetRacer autonomous vehicle with additional modifications to support four-wheel steering, model predictive control-based path following, and high-precision ultrasonic-based real-time positioning. The proposed JetRacer-4WS can be used as a low-cost platform for both research and education on motion control, path following, and vehicle dynamics. We describe the design of JetRacer-4WS, and experimentally demonstrate JetRacer-4WS' ability to perform controller auto-tuning and illustrate the advantage of four-wheel steering. We also show that JetRacer-4WS can be used as a validation platform for testing advanced control algorithms such as event-triggered model predictive control.**

***Index Terms*—Scale vehicles, autonomous vehicles, model predictive control, event-triggered control, motion control, path following, four-wheel steering.**

I. Introduction

MODEL predictive control (MPC) has been widely studied in the field of autonomous vehicle (AV) [1], [2], [3]. During the early stages in the design/development process, a relatively simple simulation environment is generally used to ensure functionality and give an indication of the expected performance of MPC. See [4], [5], [6] for example, where a simple MATLAB simulation environment or high-fidelity CARLA [5] are used to test and validate MPC.

Transitioning the testing and evaluation process of an MPC system into a physical environment by instrumenting full-size vehicles and using controlled testing facilities can be costly [7].

Manuscript received 17 March 2023; accepted 28 June 2023. Date of publication 4 July 2023; date of current version 10 July 2023. This letter was recommended for publication by Associate Editor J. Zhang and Editor J. P. Desai upon evaluation of the reviewers' comments. *(Corresponding author: Jun Chen.)*

The authors are with the ECE Department, Oakland University, Rochester, MI 48309 USA (e-mail: crother@oakland.edu; zhaodongzhou@oakland.edu; junchen@oakland.edu.).

The associated code is open-sourced and available at: https://github.com/jchenee2015/jetracer-4WS.

Digital Object Identifier 10.1109/LRA.2023.3291916

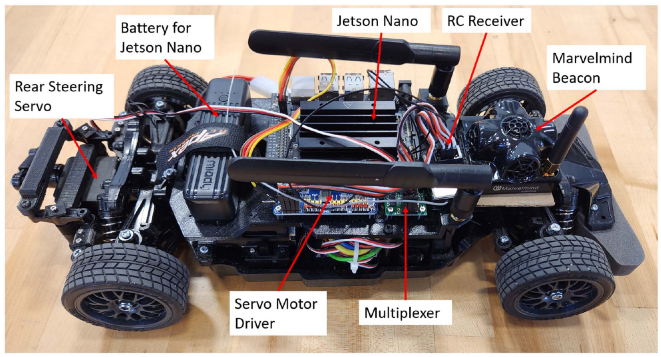

Fig. 1. Proposed scale AV platform with four-wheel steering capability.

To address this issue, scale vehicles have been developed in recent years to demonstrate autonomous driving (AD) capability [8], [9], [10], [11], [12], [13]. Using a scale vehicle platform allows engineers to perform initial hardware evaluations and to debug issues involved with using physical controllers, sensors, and actuators. For example, the scale vehicles presented in [11], [12], [13] feature end-to-end AD technology, where AI methods are used to generate control outputs directly from the sensor inputs. More traditional methods that involve separate environment perception, path planning, and motion control components are used in the scale vehicles presented in [8], [9], [10], [11], which results in a greater correlation to current automotive technology. Such scale vehicle platforms can also be used in several education activities due to their low-cost nature [8], [10].

However, there are several limitations associated with these existing platforms. *First*, existing platforms only include front steering mechanism [9], [10], [11], [12], [13]. As four-wheel steering can potentially provide better maneuverability at low speed and stability at high speed [14], it is very important to have a scale vehicle platform with four-wheel steering capability for research and education. *Second*, all the platforms found in literature [8], [9], [10], [11], [12], [13] focus on the system-level capability that integrates perception, path planning, and control (either modularly or end-to-end). Such platforms can be very helpful for system-level testing, but at the same time are of limited use for testing of motion control algorithms only. To address these limitations, this letter presents a new scale vehicle platform, based on the open-source JetRacer [13], for research and education on AV controls, specifically for MPC-based motion control. The proposed vehicle, which is shown in Fig. 1 and will be referred to as the JetRacer-4WS for the remainder of the letter, has four-wheel steering capability allowing early

TABLE I
COMPARISON OF THE PROPOSED JETRACER-4WS WITH EXISTING SCALE VEHICLE PLATFORMS

Platform	Steering	Use Case	Processor	Onboard Sensors	Cost
Duckiebot [8]	Two-wheel differential	PD lane keeping control	Raspberry Pi 2	• Camera w/ fisheye lens	$150
MIT Racecar [9]	Front-whell Ackermann	PID steering control	NVIDIA Jetson TX1	• RGB-D camera • Scanning Lidar • Stereo camera • IMU • Speedometer	∼$3,000 (estimated in [10])
ASIMcar [10]	Front-whell Ackermann	Stanley control method for lane keeping	NVIDIA Jetson TX2	• IMU • Front camera w/ fisheye lens • Rear camera • One-beam Lidar • Speedometer	$1,200
F1/10 [11]	Front-whell Ackermann	End-to-end AD and MPC	NVIDIA Jetson TX2	• Monocular USB web cam • Depth camera • Scanning Lidar • IMU	$3,480 [22]
CNN-based scale vehicle [12]	Front-whell Ackermann	End-to-end AD	Raspberry Pi 4	• Camera w/ fisheye lens • Ultrasonic sensor	N/A
JetRacer [13]	Front-whell Ackermann	End-to-end AD	NVIDIA Jetson Nano	• Wide angle camera	$825
JetRacer-4WS (ours)	Independent four-wheel Ackermann	• Controller auto-tuning • MPC-based path following	NVIDIA Jetson Nano	• Marvelmind mobile beacon (for positioning system)	$850

stage testing and validation of MPC-based motion control algorithms with active rear steering functionality. Moreover, the platform is designed primarily to test and evaluate motion control algorithms and therefore can isolate the error of the motion control module from those of perception and path planning.

To provide ground truth on real-time vehicle state information such as position and velocity, a Marvelmind positioning system [15] is added to the scale vehicle to provide vehicle localization as feedback to the control module. Note that using such a positioning system (as opposed to an onboard camera) can ensure the performance of control systems to be independent of the perception module. In addition, the proposed JetRacer-4WS is experimentally demonstrated for its ability to perform controller auto-tuning and to serve as a validation platform for advanced control algorithms such as event-triggered MPC. In particular, MPC is an optimal control technique that solves a constrained optimization problem for every control loop based on a dynamical prediction model over a finite prediction horizon. Moreover, event-triggered MPC [16], [17], [18], [19], [20], [21] is also demonstrated in the proposed platform, which aims to reduce the computational burden of conventional MPC by solving the OCP aperiodically. In other words, the event-triggered MPC solves the optimization problem only when an event is triggered, or equivalently when the control error exceeds a predefined threshold. The aforementioned references on event-triggered MPC have been focused on the theoretical development and simulation-based validation of event-triggered MPC. However, an experimental validation on event-triggered MPC-based path following has not been reported in literature. Here we demonstrate the capability of the JetRacer-4WS as a cost-effective testing and evaluation platform for advanced motion control methods, by experimentally validating event-triggered MPC-based path following.

The contribution of this letter is summarized as follows.

1) We develop a new scale vehicle platform with *four-wheel steering* capability, where front wheels and rear wheels can be independently controlled.
2) The performance of AV motion control is isolated from perception and path planning through the use of a high-precision positioning system.
3) We experimentally demonstrate the proposed platform's ability to perform controller auto-tuning and to serve as a validation platform for advanced control algorithms such as event-triggered MPC. The advantage of four-wheel steering is clearly shown, together with the benefits of using event-triggered MPC to balance computation and control performance.

The remainder of the letter is organized as follows. Section II discusses and compares several existing platforms, while Sections III and IV describe the hardware and control design of the proposed JetRacer-4WS, respectively. Section V presents several experiments to demonstrate the capability of JetRacer-4WS, while the letter is concluded in Section VI.

II. EXISTING PLATFORMS

In recent years, several scale vehicle platforms have been developed for AV research and education. As mentioned earlier, most of them focus on full AD capability, including perception, path planning, and motion control (either modularly or end-to-end). Table I lists and compares several existing scale vehicle platforms that are similar to JetRacer-4WS.

The Duckiebot [8] is an open-source platform for autonomy research and education. Each Duckiebot is equipped with onboard camera for perception and a lane-keeping control module. Duckietown, an emulated city, is also available for testing the navigation capability of Duckiebot. The steering mechanism of Duckiebot is, however, a simple two-wheel differential and therefore does not fully emulate vehicle dynamics, a critical aspect for AV motion control.

The MIT Racecar [9] is developed primarily for high school education, and makes utilization of high-end perception sensors including camera, Lidar, and IMU. Simple PID-based steering control is implemented for ease of use by high school students.

The cost, as estimated in [10], is roughly $3,000 and therefore can hinder wide use for a large group of users.

The ASIMcar [10] is a highly capable, open-source scale vehicle platform suitable to demonstrate full AD capability as well as motion control only. The onboard sensors include both a front and a rear camera with 1-d Lidar. In addition, an overhead high-speed camera is installed to serve as a vision-based positioning system. The cost is also comparable to JetRacer-4WS, but it lacks rear steering capability and the vision-based positioning system can only be used indoors.

The F1/10 [11] is a widely used platform and perhaps has the most community support among all the existing platforms. Similar to ASIMcar, F1/10 also uses NVIDIA Jetson TX2, providing powerful onboard computation for end-to-end AD techniques. The cost is the highest among all platforms reviewed, which is affordable for research but hinders a wider use for educational activities such as class projects.

Another scale vehicle platform based on 1/10 RC car is developed in [12], with the primary purpose of demonstrating end-to-end AD through convolutional neural network (CNN). Both camera and ultrasonic sensor are installed to proide input to the CNN model, which is run on a low-cost Raspberry Pi 4. The cost is unknown.

The JetRacer [13] is an open-source, flexible, and low-cost scale vehicle platform suitable for AV research and education. The onboard computation is through an NVIDIA Jetson Nano, which has low cost and wide community support. A wide angle camera is the only onboard perception sensor, providing raw video feed to the neural network model. The platform has been widely used for educational activities due to its intuitive interface to build customized end-to-end AD technology. As described in the next section, the proposed JetRacer-4WS is based on JetRacer, which will be extended to include additional functionality on rear steering, MPC-based path following, and ultrasonic-based positioning system.

There are other mobile robot platforms that are low-cost and designed for both research and education, such as Pheeno [23] and r-one [24]. However, since their primary purpose is on robotic manipulation rather than AD, their details are omitted here. Furthermore, as summarized in Table I, all these existing platforms use either two-wheel differential or front-wheel Ackermann as steering mechanisms and therefore are not suitable for research and education on four-wheel steering control. Furthermore, most of them focus on testing AD as a whole, and can not be used for testing motion control only. Note that ASIMcar [10] also allows testing motion control algorithms in isolation using the overhead camera as a global positioning system. However, it can only be used indoors, and therefore is not suitable for either high-speed testing or large track. The proposed JetRacer-4WS fills these gaps and can be used for indoor and outdoor research and education on four-wheel motion control. The detailed designs of JetRacer-4WS is given in the next section.

III. JetRacer-4WS Hardware Design

This section provides details related to the hardware design of the proposed JetRacer-4WS platform, including its chassis, steering, electronic, and positioning system.

a) Chassis: The JetRacer-4WS platform, based on the open-source JetRacer [13], is built from a 1/10th scale Tamiya TT-02 RC car. The overall configuration is depicted in Fig. 1.

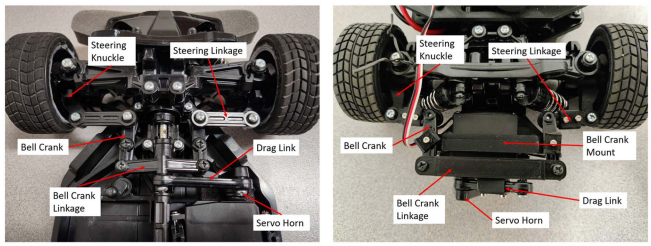

Fig. 2. (a) Standard front wheel steering assembly for the Tamiya TT-02 RC car. (b) Custom rear wheel steering assembly.

The vehicle is driven by a 72 W electric motor powering all four wheels through open differentials, and the 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.

b) Steering: Both front wheels and rear wheels use an Ackermann steering system controlled by a servo motor as shown in Fig. 2. Note that the rear steering components are mounted behind the rear axle, as shown in Fig. 2(b), due to the space limitation of the Tamiya TT-02 RC car chassis. To make sure the rear steering components are mounted tightly and durable, the original mounting points on the rear bumper are repurposed to fasten the bell cranks using a bracket and mounting pins that are 2D printed. 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. In addition, to fit the steering links and make sure that the rear wheels pointing straight forward when the rear steering linkage assembly is centered, both steering links and drag link have to be shortened by cutting out a middle section and reconnected with a 3D printed bracket. Since the space for rear steering is tight, the rear steering knuckles are also redesigned. To attach the custom rear knuckles, the original rear suspension arms is replaced with a copy of the front suspension. Finally, Fig. 2(b) illustrates the assembly of the rear steering servo, suspension arms, and steering linkage components.

c) Electronic: An NVIDIA Jetson Nano Developer Kit [13] is installed for real-time data processing and control. A separate computer is used to interface with the Jetson Nano through a WiFi connection using Jupyter Lab. The electronic components are mostly connected as specified for the standard JetRacer platform [13]. The newly added rear steering servo connector is attached to an output on the multiplexer board. The corresponding inputs on the multiplexer both come from the servo driver.

d) Positioning systems: As the primary purpose of JetRacer-4WS is to provide an affordable platform for research and education on AV motion control, it is important to provide ground truth of real-time vehicle state information to the control systems as to isolate the performance of control from those of perception and path planning, etc. In this regard, a Marvelmind positioning system [15] is installed to provide real-time measurement of vehicle position, velocity, heading angle, etc. The installed Marvelmind position system, which is shown in Fig. 3, is based on ultrasonic and has high precision of up to ± 2 cm error, accurate enough for scale vehicles. Note that alternative

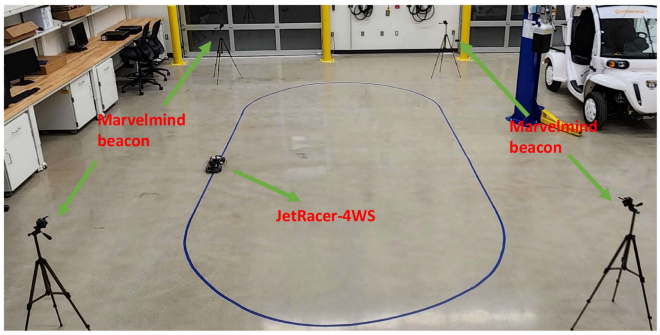

Fig. 3. Overview of the testing track with Marvelmind positioning system.

TABLE II
SUMMARY BILL OF MATERIALS FOR JETRACER-4WS

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**

positioning systems could be developed using an overhead camera to track the vehicle with computer vision methods [10] or an RTK GPS unit for outdoor testing [25], [26]. The selected Marvelmind system has the benefit of working in both indoors and outdoors, as well as being cost-effective, as discussed below.

Remark 1: All of the proposed JetRacer-4WS components can typically be found for around $850. Table II lists the bill of materials for the proposed scale vehicle platform.[1] Note that the Marvelmind system used for positioning is between $450-500, which is excluded from Table II since it can be used for a fleet of scale vehicles. Moreover, the cost-effectiveness of JetRacer-4WS is clearly shown by Table I. Therefore, the proposed JetRacer-4WS can be very advantageous in reducing cost of controls testing and evaluation as well as in serving as an educational platform for a large group of users.

IV. JETRACER-4WS CONTROL DESIGN

This section provides details related to the control design of the proposed JetRacer-4WS platform, including an MPC-based path following controller, an event-triggered MPC that reduces the computational load of conventional MPC approach, and a controller auto-tunning mechanism.

A. MPC-Based Path Following Design

To enable four-wheel steering control, a path following controller based on MPC is designed and implemented, which requires a prediction model to predict system evolution over a prediction horizon. For JetRacer-4WS, the following kinematic bicycle model [14] is used, as also shown in Fig. 4.

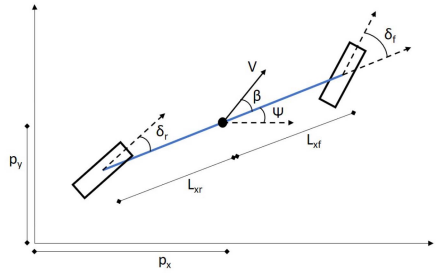

Fig. 4. Illustration of a kinematic bicycle model with rear steering.

$$\dot{p}_x = V\cos(\psi+\beta) \tag{1a}$$

$$\dot{p}_y = V\sin(\psi+\beta) \tag{1b}$$

$$\dot{\psi} = \frac{V\cos(\beta)}{L_{xf}+L_{xr}}(\tan(\delta_f)-\tan(\delta_r)) \tag{1c}$$

$$\beta = \arctan\left(\frac{L_{xf}\tan(\delta_r)+L_{xr}\tan(\delta_f)}{L_{xf}+L_{xr}}\right), \tag{1d}$$

where p_x and p_y are the vehicle longitudinal and lateral positions, respectively, and ψ is the vehicle yaw angle, all in the global frame; β 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; δ_f and δ_r are the front and rear steering angles.

The state vector can be compactly denoted as $x(t) = \left[p_x, p_y, \psi\right]^T$, and the control vector can be compactly denoted as $u(t) = \left[\delta_f, \delta_r\right]^T$. Note that for a conventional two-wheel steering system with front steering only, the rear steering angle δ_r is fixed to zero, and the control vector reduces to $u(t) = \delta_f$. Note that a dynamic vehicle model has been widely utilized in literature for MPC-based path following [27], [28] for high speed scenarios. However, since vehicle speed is usually low for the scale vehicle, a kinematic model without tire model is selected here. At each time step, MPC solves the following optimal control problem (OCP):

$$\min_{u} \quad \sum_{k=1}^{p} ||x_k - x_k^r||^2_{Q_x} + \sum_{k=0}^{p-1} ||u_k||^2_{Q_u} + \sum_{k=0}^{p-1} ||u_k - u_{k-1}||^2_{Q_d} \tag{2a}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k),\ k = 0,\dots,p-1 \tag{2b}$$

$$u_{\min} \le u_k \le u_{\max},\ k = 0,\dots,p-1 \tag{2c}$$

$$\Delta_{\min} \le u_k - u_{k-1} \le \Delta_{\max},\ k = 0,\dots,p-1 \tag{2d}$$

where the first term in the cost function (2a) penalizes deviation from the desired path x_k^r, the second term discourages large steering angles, and the third term minimizes the steering angle rate of change, p is the prediction horizon, matrices Q_x, Q_u, and Q_d are tuning parameters. Note that the last two terms are necessary to ensure stability. The system dynamic constraint (2b) can be obtained by discretizing the kinematic vehicle dynamics

[1] Based on the market price as of Fall 2021.

Algorithm 1: Event-Triggered MPC [4].

procedure EMPC(x, k, U_{t_1}, X_{t_1})
  $k \leftarrow k+1$;
  $e \leftarrow$ computing (3);
  if $e = 1$ **then**
    $k \leftarrow 0$;
    $(X_t, U_t) \leftarrow$ Solving OCP (2);
    $u \leftarrow U_t(1)$;
    $U_{t_1} \leftarrow U_t$;
    $X_{t_1} \leftarrow X_t$;
  else
    $u \leftarrow U_{t_1}(k+1)$;
  end if
  return u, k, U_{t_1}, X_{t_1}
end procedure

(1) through forward Euler [29]. The sampling time used in this letter is 200 ms. Finally, physical constraints (2c) and (2d) are the input constraint and input change rate constraint, respectively.

The above kinematic bicycle model (1), together with the OCP (2), are coded using Jupyter Lab. Note that the OCP (2) is solved by using the MPCTools package and CasADi, which are open-source optimization tools described in [29], [30]. Moreover, we have made the entire software of the proposed JetRacer-4WS open source and available at: https://github.com/jchenee2015/jetracer-4WS.

B. Event-Triggered MPC-Based Path Following

To reduce the computation complexity of MPC, event-triggered MPC has been proposed in literature [16], [17], [18], [19], [20], [21], especially for AV path following problem [4], [31], [32]. For event-triggered MPC, MPC is triggered to formulate and solve the OCP (2) only when it is needed, as opposed to being time-triggered at a fixed sampling rate. Here we consider the following event-trigger mechanism, which is adopted from [16], [31], [32]. At sampling time t, an event e is defined by

$$e = \begin{cases} 1 & \text{if } d_y > \sigma \text{ or } k > k_{\max} \\ 0 & \text{Otherwise} \end{cases}, \tag{3}$$

where k is such that $t = t_1 + kT_s$ with T_s being the sampling time, and σ and $k_{\max}$ are calibration parameters that influence the event frequency. In other words, the event trigger e is set only if the current step number k exceeds a threshold $k_{\max}$, or when the vehicle's displacement d_y from the nearest point in the desired path exceeds a predefined threshold σ.

Furthermore, an event-triggered MPC solves the OCP (2) only when $e = 1$. When $e = 0$, the control action can be determined using the optimal sequence U_{t_1} computed at last event [33], i.e.,

$$u = \begin{cases} \text{Solution of (2)} & \text{if} \quad e = 1 \\ U_{t_1}(k+1) & \text{Otherwise} \end{cases}. \tag{4}$$

Furthermore, it is trivial to see that $k_{\max} < p$, and otherwise (4) can be undefined. Algorithm 1 summarizes the event-triggered MPC for each sampling time t, where X and U denote the sequences of x_k and u_k, $k = 0, \ldots, p-1$, with their subscripts denoting the timing when X and U are calculated.

C. Controller Auto-Tuning

There are many parameters in an MPC-based motion control system (2) that can impact its control performance, including the length of the prediction horizon, calibration of the cost function (2a), and bounds for the control parameters. To auto-tune these parameters, a controller auto-tuning framework is developed, with a specific focus on auto-tuning of the gains Q_x, Q_u, and Q_d for the cost function (2a). Moreover, since only their relative magnitude impacts the solution of (2), Q_x can be assumed to be fixed, and only Q_u and Q_d need to be auto-tuned. Note that the JetRacer-4WS platform can function with either four-wheel steering or two-wheel steering modes (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.

To auto-tune the MPC-based motion controller, Design of Experiments (DoE) methods can be used to generate a test matrix, which includes a list of candidate calibrations to be tested. Several DoE methods can be used to generate such a test matrix. Full factorial design will provide the most coverage over design space. However, when the number of test points is small, it may place many points on the edges of the design space and cause the step sizes between points in the interior of the design space to be too coarse. In this work, Latin Hypercube [34] is used, which populates the design space by attempting to maximize the minimum distance between test points. Once the DoE test matrix is populated (see Section V for more details), an auto-tuning script is used to automatically iterate through the DoE test matrix and apply each calibration for a set time. Between each iteration, the first calibration (known to function reasonably well from hand calibration or previous DoE results) is re-applied for 20 seconds to allow the system to return to a reasonable starting position.

V. Experimental Results

Two types of experiments were conducted to evaluate JetRacer-4WS' ability to perform motion controller auto-tuning as well as to serve as a validation platform for testing advanced motion control algorithms such as event-triggered MPC. In both experiments, the JetRacer-4WS is set to track an oval in the counter-clockwise direction, as shown in Fig. 3. Note that the marker in Fig. 3 is for visual illustration only, as the reference path is predetermined using Marvelmind and saved to the control system to ensure consistency and comparability of subsequent test results. The maximum angle for the both front and rear wheels is ± 0.2 rd. The maximum front wheel change rate is ± 0.04 rd and the maximum rear wheel change rate is ± 0.02 rd per time step. The prediction horizon p is set to 10 according to prior work [4] to balance the computation and control performance.

A. Controller Auto-Tuning Results

Figs. 5 and 6 plot the test matrices generated by the Latin Hypercube method discussed in Section IV-C for two-wheel and four-wheel steering modes, respectively, which are tested by the auto-tuning script discussed earlier. Note that the list of

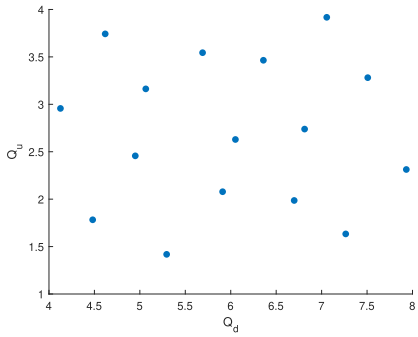

Fig. 5. Latin hypercube test matrix for two-wheel steering mode.

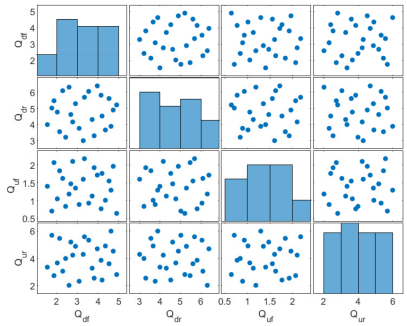

Fig. 6. Latin hypercube test matrix for four-wheel steering mode.

TABLE III
DOE RESULTS FOR TWO-WHEEL STEERING MPC. UNIT: METER

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**

calibration can also be found in Tables III and IV. Note also that the upper and lower bounds on Figs. 5 and 6 are chosen, based on manual calibration with the control system, so that a reasonably large space can be explored while focusing on testing points with reasonable performance.

Table III presents the experiment results for two-wheel steering and Table IV for four-wheel steering. Note that the root mean square error (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 calibrations, a simple index $I = \overline{RMSE} + \overline{MaxError}$ is calculated, where $\overline{RMSE}$ and $\overline{MaxError}$ are normalized with respect to the smallest

TABLE IV
DOE RESULTS FOR FOUR-WHEEL STEERING MPC. UNIT: METER

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

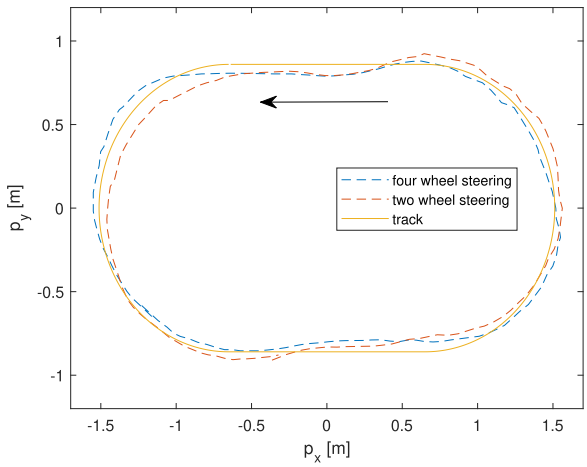

Fig. 7. Path following results for two-wheel and four-wheel steering modes.

value. Note that in both Tables III and IV, the top performing calibrations based on the cost index are marked in bold. In addition, Fig. 7 plots the path following performance for both two-wheel 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 Tables III, IV, and Fig. 7 show that the best calibrations from the four-wheel steering MPC-based path following controller are able to outperform the best calibrations from the two-wheel steering system, with the index I equals 2.01 for four-wheel steering and 2.94 for two-wheel steering. It is worth noting that, as shown in Fig. 7, the test track used in this letter has a small turning radius of 0.8 m, which approaches the maximum achievable turning radius by the

TABLE V
EXPERIMENTAL RESULTS FOR TWO-WHEEL EVENT-TRIGGERED MPC

σ	RMSE (m)	Max Error (m)	I (m)	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 VI
EXPERIMENTAL RESULTS FOR FOUR-WHEEL EVENT-TRIGGERED MPC

σ	RMSE (m)	Max Error (m)	I (m)	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

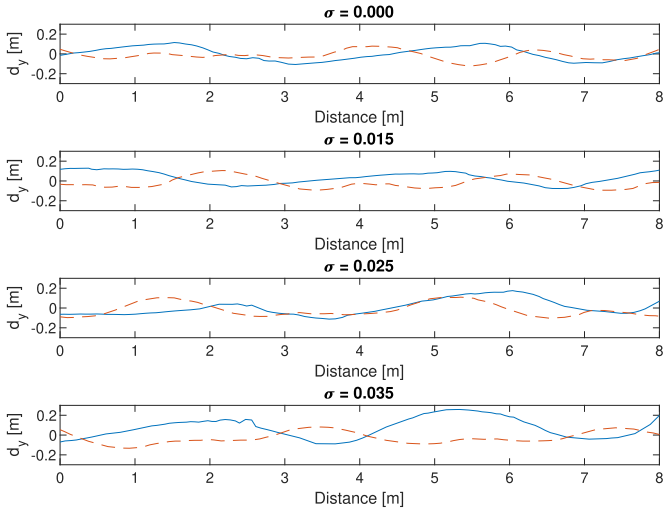

Fig. 8. Performance of two-wheel v.s. four-wheel event-triggered MPC. Solid: two-wheel steering; Dash: four-wheel steering.

two-wheel steering geometry. Therefore, the results presented in this letter suggest that implementing independent four-wheel steering MPC could see performance improvements in parking lots and other challenging driving scenarios that require high agility. A similar test is performed on a larger track with radius of 1.5 m, where four-wheel steering MPC still outperforms two-wheel steering MPC, with I being 2.02 with rear steering and 2.15 without. Therefore, for a normal driving scenario, the performances of two-wheel steering and four-wheel steering are very comparable.

B. Event-Triggered MPC-Based Path Following Results

Experiments on both two-wheel and four-wheel steering event-triggered MPC are tested with various threshold σ on a test track of radium of 0.8 m.[2] Results are shown in Tables V and VI and Figs. 8 and 9. Note that the same metrics are used to determine the path following performance, including RMSE, maximum offset, and the cost index calculated I from these values. In addition, the relative computational load can

[2]Same experiments on a large track with a radius of 1.5 m were also conducted with similar conclusion. Due to space limitation, the results on large track can be found at project Github page.

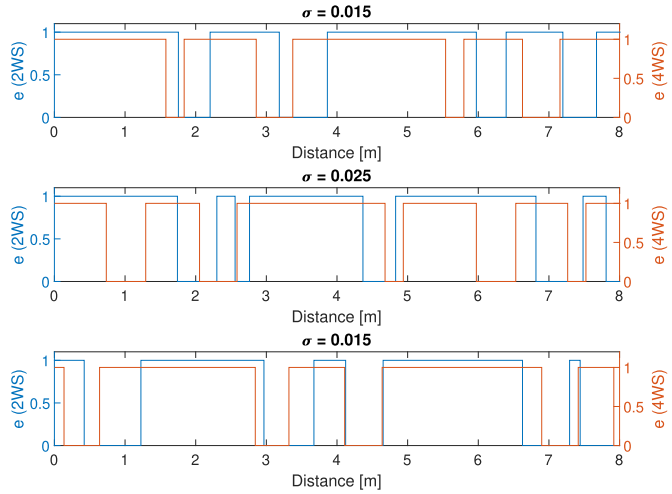

Fig. 9. Events of two-wheel event-triggered MPC. Left axis: two-wheel steering; Right axis: four-wheel steering.

be compared using the event-trigger frequency, which is the percentage that $e = 1$. Note that $\sigma = 0$ here corresponds to the case of time-triggered MPC path following, which are copied from Tables III and IV, respectively, to serve as a baseline. As can be seen, in most cases, the increase of event-trigger threshold σ will reduce MPC computation load with a lower event-trigger frequency, while negatively impacting the path following performance with worse RMSE and Max Error. In other words, as σ increases, both event-trigger frequency and performance index I increase, indicating an unavoidable performance degradation. However, as can be seen from Fig. 8, when $\sigma \leq 0.025$, such control performance degradation is very minimal compared to the baseline time-triggered MPC (i.e., when $\sigma = 0$), while up to 10% computation reduction can be expected (Tables III and IV and Fig. 9). Furthermore, according to Fig. 8, the advantage of using four-wheel steering is clearly illustrated, where four-wheel steering results in much smaller later errors in call cases. Finally, Fig. 9 compares the events in two-wheel and four-wheel modes. As can be seen, similar event patterns can be found in these two steering modes.

Remark 2: Compared to the simulation results in [4], where up to 50% computation reduction for event-triggered MPC path following can be expected, the experimental results presented here seem to be less promising. This can be attributed to the presence of random noise in a physical vehicle system, but at the same time this experiment provides more realistic validation results, which suggest that a simple threshold-based event-trigger mechanism may not be sufficient. Therefore, more sophisticated event-trigger mechanisms, such as reinforcement learning-based event-triggered MPC [31], will be tested in JetRacer-4WS as future work. Nonetheless, the experiments conducted here demonstrate the capability of JetRacer-4WS as a validation platform for advanced motion control algorithms, providing more realistic results compared to simulation and incurring less expense compared to full-scale vehicle testing. It is also worth mentioning that, reducing computation by 10% without incurring major control performance degradation can be significant in enhancing automotive systems as more advanced algorithm can now be implemented in low cost micro-controller.

VI. Conclusion

This letter presents a cost-effective scale vehicle platform (named as JetRacer-4WS) with four-wheel steering capability for research and education on autonomous vehicle motion control, path following, and vehicle dynamics. 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, and the proposed JetRacer-4WS addresses several limitations of existing scale vehicle platforms. Model predictive control (MPC) for both two-wheel and four-wheel steering are implemented using a kinematic bicycle model. Experiments on motion controller auto-tuning and event-triggered MPC validation are conducted to illustrate JetRacer-4WS' ability and versatility. Experimental results show that JetRacer-4WS can provide more realistic results compared to simulation and at the same time incur less expense compared to full-scale vehicle testing. The developed software is open-sourced and available at GitHub. Future work includes testing path following performance of additional motion control methods such as reinforcement learning, reinforcement learning-based event-triggered MPC, and a wholistic MPC to control both steering and throttle.

References

[1] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Veh. Symp.*, 2015, pp. 1094–1099.

[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. Technol.*, vol. 21, no. 4, pp. 1236–1248, Jul. 2013.

[3] R. Yu, H. Guo, Z. Sun, and H. Chen, "MPC-based regional path tracking controller design for autonomous ground vehicles," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 2510–2515.

[4] J. Chen and Z. Yi, "Comparison of event-triggered model predictive control for autonomous vehicle path tracking," in *Proc. IEEE Conf. Control Technol. Appl.*, 2021, pp. 8–13.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Conf. Robot Learn.*, 2017, pp. 1–16.

[6] Z. Zhou, C. Rother, and J. Chen, "Event-triggered model predictive control for autonomous vehicle path tracking: Validation using CARLA simulator," *IEEE Trans. Intell. Veh.*, early access, Apr. 13, 2023, doi: 10.1109/TIV.2023.3266941.

[7] N. Lutsey and M. Nicholas, "Update on electric vehicle costs in the United States through 2030," *Int. Council Clean Transp.*, vol. 12, pp. 1–12, 2019.

[8] L. Paull et al., "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1497–1504.

[9] S. Karaman et al., "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT," in *Proc. IEEE Integr. STEM Educ. Conf.*, 2017, pp. 195–203.

[10] X. Wu and A. Eskandarian, "An improved small-scale connected autonomous vehicle platform," in *Proc. Dyn. Syst. Control Conf.*, 2019, Art. no. V001T04A003.

[11] M. O'Kelly et al., "F1/10: An open-source autonomous cyber-physical platform," 2019, doi: 10.48550/arXiv.1901.08567.

[12] 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.

[13] Jetracer, "An autonomous AI racecar using NVIDIA jetson nano," 2019. [Online]. Available: https://github.com/NVIDIA-AI-IOT/jetracer

[14] R. Rajamani, *Vehicle Dynamics and Control*, (Series Mechanical Engineering Series). Berlin, Germany: Springer, 2012.

[15] M. Robotics, Accessed: Apr. 11, 2022. [Online]. Available: https://marvelmind.com/

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

[17] 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.

[18] R. Badawi and J. Chen, "Enhancing enumeration-based model predictive control for DC-DC boost converter with event-triggered control," in *Proc. IEEE Eur. Control Conf.*, 2022, pp. 723–728.

[19] H. Li, W. Yan, and Y. Shi, "Triggering and control codesign in self-triggered model predictive control of constrained systems: With guaranteed performance," *IEEE Trans. Autom. Control*, vol. 63, no. 11, pp. 4008–4015, Nov. 2018.

[20] C. Liu, H. Li, Y. Shi, and D. Xu, "Codesign of event trigger and feedback policy in robust model predictive control," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 302–309, Jan. 2020.

[21] R. Badawi and J. Chen, "Performance evaluation of event-triggered model predictive control for boost converter," in *Proc. IEEE Veh. Power Propulsion Conf.*, 2022, pp. 1–6.

[22] B. Zheng and J. Betz, "F1TEHTH project," 2022. [Online]. Available: https://f1tenth.org/build.html

[23] S. Wilson et al., "Pheeno, a versatile swarm robotic research and education platform," *IEEE Robot. Automat. Lett.*, vol. 1, no. 2, pp. 884–891, Jul. 2016.

[24] J. McLurkin et al., "A robot system design for low-cost multi-robot manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 912–918.

[25] 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, Art. no. 2703.

[26] S. Park, S. Ryu, J. Lim, and Y.-S. Lee, "A real-time high-speed autonomous driving based on a low-cost RTK-GPS," *J. Real-Time Image Process.*, vol. 18, pp. 1321–1330, 2021.

[27] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, May 2007.

[28] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *Int. J. Veh. Auton. Syst.*, vol. 3, pp. 265–291, 2005.

[29] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi–A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[30] J. Rawlings, "Model predictive control (MPC) tools package," 2019. [Online]. Available: https://sites.engineering.ucsb.edu/jbraw/software.html

[31] J. Chen, X. Meng, and Z. Li, "Reinforcement learning-based event-triggered model predictive control for autonomous vehicle path following," in *Proc. IEEE Amer. Control Conf.*, 2022, pp. 3342–3347.

[32] S. Huang and J. Chen, "Event-triggered model predictive control for autonomous vehicle with rear steering," Tech. Rep. 2022-01-0877, 2022.

[33] J. Yoo and K. H. Johansson, "Event-triggered model predictive control with a statistical learning," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 4, pp. 2571–2581, Apr. 2021.

[34] F. A. C. Viana, "A tutorial on latin hypercube design of experiments," *Qual. Rel. Eng. Int.*, vol. 32, pp. 1975–1985, 2016.