DART

A Compact Platform for Autonomous Driving Research

Lyons, Lorenzo; Niesten, Thijs; Ferranti, Laura

DOI
10.1109/IV55156.2024.10588526

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the 35th IEEE Intelligent Vehicles Symposium, IV 2024

2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

DART: A Compact Platform for Autonomous Driving Research*

Lorenzo Lyons[1], Thijs Niesten[1], Laura Ferranti[1]

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588526

Abstract— **This paper presents the design of a research platform for autonomous driving applications, the Delft's Autonomous-driving Robotic Testbed (DART). Our goal was to design a small-scale car-like robot equipped with all the hardware needed for on-board navigation and control while keeping it cost-effective and easy to replicate. To develop DART, we built on an existing off-the-shelf model and augmented its sensor suite to improve its capabilities for control and motion planning tasks. We detail the hardware setup and the system identification challenges to derive the vehicle's models. Furthermore, we present some use cases where we used DART to test different motion planning applications to show the versatility of the platform. Finally, we provide a git repository with all the details to replicate DART, complete with a simulation environment and the data used for system identification.**

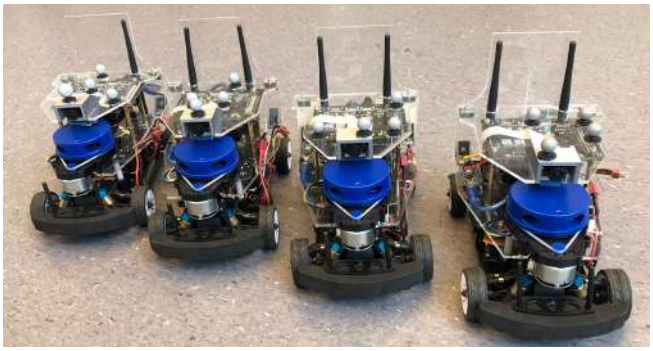


Fig. 1: DART, a small-scale robotic platform for autonomous driving research.

I. Introduction

Autonomous driving technologies have seen significant advancements in recent years leading to many companies around the world investing in, testing, and, in some cases, commercializing autonomous driving services, such as robo-taxis [1] and truck platooning [2]. Despite these encouraging signals, the day when fully autonomous cars will be a large percentage of traffic participants is decades ahead of us [3], and much research is sill needed to unlock the full benefits of autonomous driving.

To deploy these technologies in our cities, extensive testing and validation of tailored navigation algorithms are necessary steps to understand potential limitations, but also gain users' trust. Assessing these methods in simulation is a cost effective solution, yet despite the increasing number of available driving simulators in recent years [4], [5], issues concerning fidelity of sensor data and vehicle dynamics still remain a significant drawback [6]. Indeed it is quite challenging, if not impossible, to accurately capture and model all the interactions among the different components of an autonomous vehicle such as road-tire interactions, actuator and sensor dynamics, electronic and software components that give rise to measurement noise and delays. On the other hand, testing autonomous driving algorithms directly on full-scale platforms entails for significant costs and long time scales, not to mention safety and regulatory restrictions. Small-scale testing platforms provide an intermediate step (or a compromise) between a purely simulated system and a full-scale autonomous car. Compared to the latter, small-scale testbeds feature simpler hardware components with simpler dynamics and low-level embedded controllers, like braking systems, powertrain and suspensions, and they also typically feature cheaper sensors such as cameras, Lidars and Radars. Yet despite these differences they still need to fulfill all the functional requirements as their full-scale counterparts, and thus offer a holistic and more realistic testing environment compared to a simulator, at a fraction of the cost of a full-scale system.

Despite the advantages of using small-scale platforms in autonomous driving research, very few models are available on the market, most of which focus on machine-learning perception algorithms for racing, making them challenging to use for control and multi-agent applications. As a result many research groups have developed their own customized platform, leading to reproducibility and accessiblity issues.

With the previously mentioned issues in mind this paper presents the Delft's Autonomous-driving Robotic Testbed (DART), shown in Figure 1. This affordable and easy to build small-scale car-like robot is designed for both single and multi-vehicle autonomous driving experiments.

II. Related works

This section reviews small-scale car-like robots, roughly between 1:5 and 1:20 of a full vehicle. Full-scale research platforms and differential wheeled robots like TurtleBot [7] and Duckiebot [8] are considered out of scope for the present paper and will thus not be covered. We will describe the most well-known platforms starting from largest (1:5 scale) to smallest (1:16 scale) highlighting their main features, summarised in table I for convenience. Notice that the listed prices are indicative, as costs may vary over time and across different countries.

*This research is supported by the NWO-TTW Veni project HARMONIA (no. 18165).

[1]The authors are with the Reliable Robot Control Lab, Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands {l.lyons,T.E.J.Niesten,l.ferranti}@tudelft.nl

The AutoRally [9] is 1:5 scale vehicle-like robot platform developed for aggressive autonomous outdoor driving at the Georgia Tech Autonomous Racing Facility. The AutoRally platform is based on a 1/5 scale RC trophy truck. Due to its size, it can include a powerful mini desktop computer for state-of-the-art computationally heavy algorithms. The mini desktop can communicate through 2.4GHz/5GHz Dual-Band High-Speed WiFi and through an extra added 900 MHz XBee Pro providing a low latency, low-bandwidth wireless communication channel. Featuring high-end computation units, sensors, and actuators the total cost of this platform is around $5000\$$. To add these components to the main body and make them ready for outdoor conditions, several parts are customized.

The MIT RACECAR [10] (Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot) is a 1:10 scale vehicle-like robot platform developed by MIT. The RACECAR uses a Traxxis Rally Car platform with a powerful Nvidia Jetson TX1 computer to process all the data from the attached sensors. Communication is provided by 2.4GHz/5GHz Dual-Band High-Speed WiFi supported by the Nvidia Jetson TX1. Featuring high-end computation units, sensors, and actuators the total cost of this platform, including the lidar and Jetson TX1 computer, is $\approx 3700\$$. The system is also equipped with a self-made Vedder ESC that can prove difficult to replicate.

The BARC [11] (Berkeley Autonomous Race Car) is a 1:10 scale car-like robotic platform developed by UC Berkeley. The latest version, V4.0, consists of a Tamiya chassis and a Nvidia Jetson Nano, that supports the same Communication features as the more powerful TX1 computer. The platform, featuring an front facing Intel Depth camera and rear facing RGB camera, costs around $950\$$.

The MuSHR [12] (Multi-agent System for non-Holonomic Racing) is a 1:10 scale platform developed at the University of Washington's Paul G. Allen School of Computer Science Engineering. MuSHR can be built using the Redcat Racing Blackout SC 1:10 chassis and has two additional features compared to the BARC, which are a 2D Lidar and a bumper switch for collision detections. This platform comes at a cost of around $1050\$$.

The Donkey Car [13] is a 3D printed platform that can easily be attached to a 1:16 scale vehicle-like robot platform, with a few mentioned in the build instructions [13] but they are no longer available. The computing module on the Donkey Car is a Raspberry Pi that uses 2.4Ghz WiFi for communication. This platform features only one RGB camera and costs around $300\$$.

Contributions. The main factor that hinders the widespread adoption of the small-scale platforms available in literature is the need for custom parts. Indeed many prestigious research institutions have a team of skilled technicians specifically dedicated to design and maintain the robotic platforms, yet this may not be the case for other research labs. Some DIY robots, like the Donky Car project, solve this problem by relying on easily printable 3D parts, yet they feature a limited computation power and sensor-actuator suite, diminishing their appeal as research platforms. DART aims to fill this gap in the literature by presenting itself as a low-cost yet versatile platform. The contributions of this paper are threefold:

- DART. We present the result of the design process behind the platform, as well as a list of parts and indications on the level of skill required to build it.
- System identification. A reliable dynamic model is essential to accurately control the vehicle, yet in contrast to full-scale cars where a vast amount of literature is available, much fewer works focus on model-fitting practices for small-scale cars. In this paper we present a model identification process specifically designed for 1:10 scale car-like robots.
- Hardware and software setup. To ease reproducibility and speed up the setup of the platform the building instructions and the code relative to low level control, system identification and some examples of high level controllers, as well as a simulation environment can be found in the GitHub repository [14].

III. DART

This section presents the main features of the proposed robotic platform. For a quick overview of the functionalities and the relative required parts see Table II, while for a complete hardware and setup guide please refer to the GitHub repository [14].

DART's design adhers to the following criteria:

Accessibility: the total cost of the platform should be as low as possible, making it affordable for most research institutions, even for educational purposes.

Reproducibility: the platform should be based on a commercially available hardware and the custom parts should be as few as possible.

Versatility: the platform should lend itself to a broad variety of research fields, thus maximizing the number of researchers using the same platform, ultimately facilitating knowledge transfer.

With these criteria in mind, the robot is based on the commercially available JetracerPro AI kit [15]. This platform is originally designed for machine learning tasks and features a good computation unit (the Nvidia Jetson Nano), high reachable speeds due to the powerful brushed electric motor and 4-wheel drive.

Our goal is to use the platform to test navigation and control algorithms for autonomous and cooperative driving. Hence, to be able to use such a vehicle for these applications, we augmented the base kit as follows. We introduced a custom-made magnetic and infrared wheel encoder and an IMU necessary to produce odometry measurements, as well as an Arduino to process the raw sensor data. We also added a Lidar needed for localization and mapping (note that a camera with fish-eye lenses is already available in the base kit). Furthermore, after some testing, we upgraded the servomotor used for steering and included an external LiPo battery to power both the latter and the longitudinal driving electric motor. These two upgrades have proven to

Platform	Cost [$]	Scale	Computing	Sensors	Drawbacks
AutoRally [9]	5.000,00	1:5	Powerfull custom-build computer	Lord Microstrain 3DM-GX4-25 IMU Emisphere P307 GPS 2 RGB cameras Odemetry sensor	High number of custom parts
MIT RACECAR [10]	3.663,00	1:10	Nvidia Jetson TX1	Hokuyo UST-10LX Stereolabs ZED Structure.io depth camera	Self made ESC No shipping outside USA
BARC [11]	950,00	1:10	Jetson Nano	Intel Realsense D435i RGB camera	
MuSHR [12]	1050,00	1:10	Jetson Nano	Intel Realsense D435i YDlidar X4 VEX bumber switch	
Donkey Car [13]	300,00	1:16	Raspberry Pi	RGB camera	Low computational power One camera sensor

TABLE I: Main features of the small-scale car-like platforms available in the literature.

significantly improve the consistency of the measured longitudinal acceleration and steering angle, largely increasing control performances. The total cost of the platform is around €700 and requires little technical knowledge to assemble.

One challenge we faced was how to optimize the available space to accommodate the extra hardware given the size of the platform without compromising its driving performance. For this, we developed custom PVC plates to accommodate the augmented sensor suite and the encoder to measure wheel velocity. The PVC plates can be produced from the .stl files available in the GitHub repository [14] and many companies or university workshops offer online laser cutting services. The encoder requires fitting small magnets on the gear of the main shaft, yet no soldering is required and this operation requires low technical skills. Compared to platforms in the same price range, i.e. [12] and [11], DART features comparable hardware components and thus offers similar performance in terms of control accuracy and overall platform capabilities, yet relying more on off-the shelf components such as the JetracerPro AI base kit, it is much easier to replicate.

IV. System identification

Building reliable vehicle models is not only required to run model-based controllers such as MPC, but is also necessary to develop realistic simulators to test any kind of controller before deploying it on the real hardware.

A dynamic system, such as a robot, can be represented by a system of ordinary differential equations $\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}, \boldsymbol{u})$, where $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_x}, \boldsymbol{u} \in \mathbb{R}^{n_u}$ are the real state and control input, respectively. System identification refers to the problem of building a function $\boldsymbol{f}$, i.e., the model of the system, that is able to adequately approximate the real dynamics $\hat{\boldsymbol{f}}$ [16]. This can be achieved by recording measurements of the state and control action, collected in a matrix $\boldsymbol{X}$ of size $N \times M$, where N is the number of data points and M is the dimension of the input data. Note that the field of control the term "input" usually refers to the control action, while in system identification and machine learning the same term indicates the inputs to the model. In the remainder of this section we will follow the latter's terminology, i.e., $M = n_x + n_u$. For each row in $\boldsymbol{X}$, the corresponding measured system output is collected in a matrix $\boldsymbol{Y}$ of size $N \times n_x$. In this paper we will make use of parametric models, that is, we assume that the function $\boldsymbol{f}$ has a fixed structure where a set of parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$ can be chosen in order to better approximate the real dynamics, i.e., $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{p})$. The optimal set of parameters $\boldsymbol{p}$ is chosen as the solution the following minimization problem:

$$\min_{\boldsymbol{p}} \sum_{k=0}^{N} \mathcal{D}\left(\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{p}\right), \boldsymbol{Y}_k\right), \tag{1}$$

where $\boldsymbol{X}_k$ and $\boldsymbol{Y}_k$ indicate the k-th row in the repsective matrices, and $\mathcal{D}$ is a function that measures the distance between the observed data point $\boldsymbol{Y}_k$ and the model output $\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{p}\right)$. A typical choice for $\mathcal{D}$ is the squared error, i.e. $\mathcal{D}(\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{p}\right), \boldsymbol{Y}_k) = \left(\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{p}\right) - \boldsymbol{Y}_k\right)^\top \left(\boldsymbol{f}\left(\boldsymbol{X}_k, \boldsymbol{p}\right) - \boldsymbol{Y}_k\right)$.

The most critical aspect of parametric model identification is the design of the function $\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{p})$ and the definition of reasonable parameter bounds, since this will heavily influence the ability of the model to correctly represent the data. While for full-scale vehicles there is a rich literature featuring various models [17], empirical formulas like the Pacejka tire model [18] for tire forces estimation, best practices for data collection and reference values for the fitting parameters [19], much less is available for small-scale cars. Furthermore, the dynamic models used for full-scale cars do not easily transfer to their small-scale counterparts, since for example the latter do not feature pressurised tires and the weight scale is around 2-3 orders of magnitude different. As a result, the range of typical values for some important parameters (e.g., the cornering stiffness) can be significantly different between full-scale and small-scale cars, making model identification particularly challenging for the latter.

In the remaining part of this section we will describe how to obtain reliable kinematic and dynamic bicycle models [20] for these small-scale vehicles. We will describe how to obtain $\{\boldsymbol{X}, \boldsymbol{Y}\}$ in problem (1) from the raw sensor data and how to define reasonable parameter bounds. The values of the parameters are obtained by numerically solving problem (1) where $\mathcal{D}$ has been chosen as the square error, using the ADAM gradient descent based algorithm implemented in PyTorch. Table III shows the obtained parameter values. Notice that all the raw data and the code for data processing

Component	Functionality	Notes	Cost [€]
JetracerPro AI kit	Base robotic platform	18650 batteries are not included	420,00
YLidar X4 Lidar	Localization and mapping	-	100,00
Encoder	Wheel velocity measurement	Requires medium-low technical skills to build	10,00
IMU BNO055	Acceleration and yaw rate measurement	-	40,00
Arduino Nano	Collect data from encoder and IMU	Price includes jumper-wires and adaptors	30,00
LiPo battery	Power servo and driving motor	Price is for 2 batteries and a charger	30,00
Servomotor DS3225	Improved steering consistency	-	20,00
PVC mounting plates	Accommodate augmented sensor suite	Needs to be custom made	20,00
		Total cost	670,00

TABLE II: Overview of the parts needed to build DART.

and model fitting is available in the GitHub repository [14], as well as a simulation environment that uses the obtained vehicle models.

A. Kinematic bicycle model

The kinematic bicycle model consists of the following system of ordinary differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\eta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\eta \\ v\sin\eta \\ v\tan(\delta)/l \\ (F_m + F_f)/m \end{bmatrix} \tag{2}$$

Where the states x, y, η and v are the position of the rear axle, orientation, and longitudinal velocity of the vehicle, respectively. The mass and distance between the front and rear axles of the robot are indicated by m and l, respectively. F_m and F_f are the motor and friction forces, respectively. For full-scale cars, the motor characteristic curve that indicates the longitudinal force F_m transmitted to the wheels, and the steering angle δ as a function of the steering input are usually provided by the manufacturer. For small-scale cars, on the other hand, throttle τ and steering input s are provided to an ESC module and a servomotor, respectively, both of which accept normalized non-dimensional values between $[-1, 1]$. Identifying how these inputs are related to motor torque and steering angles represents an additional complication in the model fitting process.

The kinematic bicycle model (2) for small-scale cars requires the identification of the following sub-models: $F_f(v)$, $F_m(\tau, v)$, $\delta(s)$, as well as the actuation delays. Simply collecting driving data and fitting the model poses significant challenges since the fitting function for each sub-model needs to be user designed, and the relative parameters need to be initialized to some reasonable value in order to numerically solve Problem (1). For this reason our approach was to isolate each component and progressively build the full model. Attempting to identify one sub-model at a time allows us to tailor the type of test (e.g., step input or sinusoidal input test) to better highlight its contribution, while clearly visualizing the measured data aids in the design of the fitting function.

1) Longitudinal dynamics: We performed a series of step response tests for different throttle values on a smooth and level surface, while the steering is kept equals to zero. The collected raw data is a time series of throttle-velocity pairs. The acceleration is then obtained by numerically deriving the velocity, while the resulting longitudinal force $F = F_m + F_f$ acting on the vehicle is estimated by multiplying the acceleration by the mass of the vehicle $m = 1.67$ Kg that can be easily measured. The training inputs $\boldsymbol{X}$ are thus the throttle and velocity, while the training labels $\boldsymbol{Y}$ are the resulting measured longitudinal force. Figure 2 shows an example of the step response data.

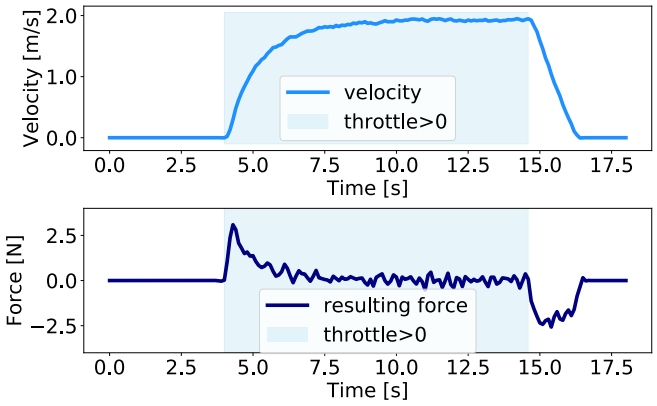


Fig. 2: The velocity profile (top) and the estimated resulting force acting on the vehicle (bottom) measured in response to a step throttle input (shaded area).

We start by modeling the friction force F_f. To isolate its contribution we selected the data with $\tau = 0$, that is, when the motor is switched off and vehicle is slowing down due to friction alone. By using the data showed in Figure 3, we designed the friction curve as:

$$F_f(v) = -(a\tanh(bv) + vc)$$

Where a, b, c are the fitting parameters. Notice that reasonable parameter initialization can be found by plotting the resulting friction curve over the data.

We now procede to model the motor force F_m. Since the friction term F_f has been successfully identified, we can define new training labels $\boldsymbol{Y}$ as the estimated motor force F_m, obtained as $\boldsymbol{Y} = F - F_f(\boldsymbol{X})$. We designed F_m in accordance to the characteristic curve of a brushed electric motor [21]:

$$F_m = (d - ve)\tilde{\tau}$$
$$\tilde{\tau} = (\tau + g)0.5(\tanh(100(\tau + g)) + 1)$$

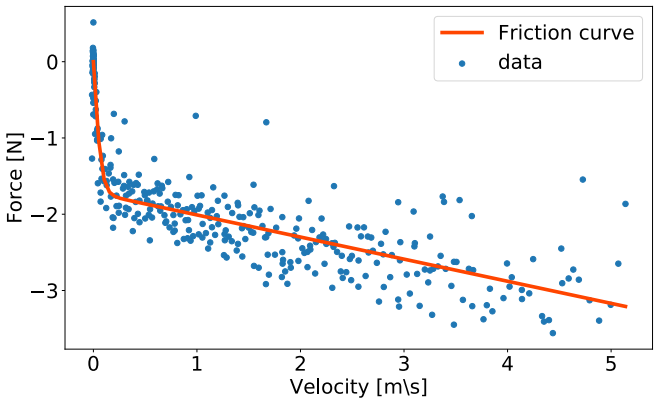


Fig. 3: Friction curve fitting results.

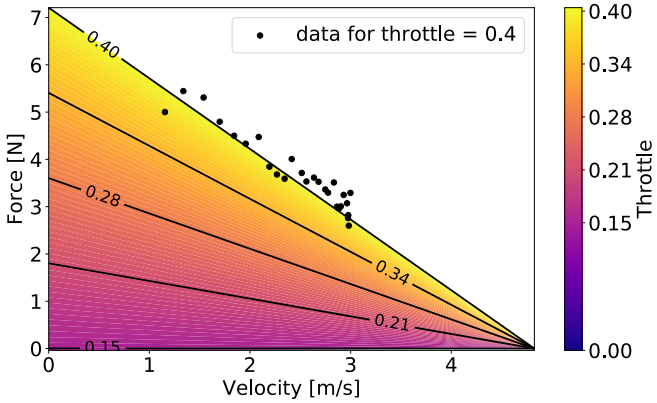


Fig. 4: Motor curve fitting results.

Where d, e, g are the fitting parameters. Note that the term $\tilde{\tau}$ is an approximation of the function $\tilde{\tau} = \max(0, \tau + g)$ but is continuously differentiable. This is highly desirable if the model will be used for model-based control such as MPC. Reasonable first guess parameter values for d and e can be obtained by plotting the measured step response data for a certain throttle value on the $\{v, F_m\}$, plane. The initial value for g was estimated by increasing the throttle from $\tau = 0$ until the vehicle begins moving. Figure 4 shows the step response data for $\tau = 0.4$ and $1 \leq v \leq 3\,\text{m/sec}$ used to initialize d and e, and the overall characteristic motor curve, note that the latter was obtained using the full dataset featuring τ in the range $\tau \in [0.15, 0.4]$.

2) Steering input to steering angle mapping: The servo-motor used for steering only accepts steering inputs in the range $s \in [-1, 1]$, we therefore need to identify how they relate to actual steering angles δ. To do so, we preformed a series of constant steering input tests while keeping the throttle at a constant value. The raw data consists of the resulting yaw rate $\dot{\eta}$ and the vehicle longitudinal speed v. By inverting the η dynamics (i.e., the third equation in (2)) we are able to estimate the steering angle as:

$$\delta = \arctan\left(\frac{l\dot{\eta}}{v}\right) \tag{3}$$

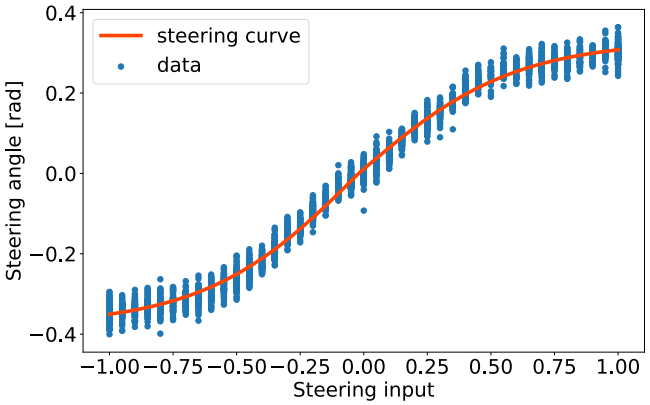


Fig. 5: Steering input to steering angle static mapping.

We can thus create a dataset using the steering input s as the training input $\boldsymbol{X}$ and the measured steering angle δ as the training labels $\boldsymbol{Y}$. Using the data shown in Figure 5 we define the steering input to steering angle mapping as the weighted sum of two sigmoidal curves. This is needed to capture the asymmetry between steering to the left and steering to the right. The curve $\delta(s)$ is defined as:

$$\delta(s) = \tilde{w}\tilde{a}\tanh(\tilde{b}(s + \tilde{c})) + (1 - \tilde{w})\tilde{d}\tanh(\tilde{e}(s + \tilde{c})) \tag{4}$$

$$\tilde{w} = 0.5(\tanh(30(s + \tilde{c})) + 1) \tag{5}$$

Reasonable first guess values were identified empirically from the shape of the measured data.

3) Actuation delays: The longitudinal actuation delay was measured by lifting the robot off the ground (so to reduce the system's inertia) and measuring the time delay between a throttle step input and a change in the wheel speed v. From these tests the longitudinal actuation delay was found to be negligible (in the order of $0.01\,\text{sec}$). The steering delay was instead obtained by providing the vehicle with a low frequency sinusoidal steering input. This allowed us to capture the reaction time of the steering, that is, the time interval before seeing a reaction after sending a steering command. The delay was then estimated by performing the cross correlation [22] between the steering angle input $\delta(s)$ and the measured steering angle obtained from equation (3). The steering delay was found to be around $0.15\,\text{sec}$ and is thus not negligible.

B. Dynamic bicycle model

DART allows one to perform highly dynamic maneuvers (e.g., racing). In such a context, a kinematic model is no longer a good representation of the vehicle dynamics, since it neglects the lateral dynamics of the vehicle. Indeed it relies on the assumption that no lateral slipping occurs, that is, there is no lateral motion in the vehicle body frame. This assumption holds until the tires are able to provide enough lateral friction force to counter the centrifugal force during curves. The centrifugal force can be evaluated as $F_y = m\,\dot{\eta}\,v$. For high speed and high yaw rate, the assumption that lateral slip is negligible is no longer valid and the kinematic bicycle

model fails. The dynamic bicycle model, on the other hand, does account for the lateral dynamics and relies on a tire model to predict tire force saturation, providing better model accuracy at high speeds.

The dynamic bicycle model is defined as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\eta} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x \cos\eta - v_y \sin\eta \\ v_x \sin\eta + v_y \cos\eta \\ \omega \\ (\hat{F}_{x,r} + \hat{F}_{x,f}\cos\delta - \hat{F}_{y,f}\sin\delta)/m + \omega v_y \\ (\hat{F}_{y,r} + \hat{F}_{y,f}\cos\delta + \hat{F}_{x,f}\sin\delta)/m - \omega v x \\ (l_f(\hat{F}_{y,f}\cos\delta + \hat{F}_{x,f}\sin\delta) - l_r\hat{F}_{y,r})/I_z \end{bmatrix}, \tag{6}$$

where v_x and v_y are the velocity components of the centre of mass measured in the vehicle body frame. l_f and l_r are the distances between the centre of mass and the front and rear axle, respectively. ω is the yaw rate and I_z is the moment of inertia around the vertical axes. The front and rear tire forces components $\hat{F}_{f,x}$, $\hat{F}_{f,y}$, $\hat{F}_{r,x}$ and $\hat{F}_{r,y}$ are measured in the respective tire's body frame. The evaluation of the tire forces is the most critical component of the dynamic bicycle model and a vast amount of literature is available for full-scale cars. One of the most famous tire models is the Pacejka magic formula [18], defined as:

$$\hat{F}_y = D\sin(C\arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))) \tag{7}$$

where α is the slip angle, defined as the angle from the direction of motion of the tire and the tire axis. Front and rear slip angles are defined as:

$$\alpha_f = -\arctan(v_y + \omega l f) + \delta \tag{8}$$

$$\alpha_r = -\arctan(v_y - \omega l r). \tag{9}$$

To identify the parameters D, C, B, E we collected driving data keeping a constant steering angle and gradually increasing the longitudinal velocity. Since we need to measure both longitudinal and lateral velocities in the vehicle body frame we used an external motion capture system, as the on-board sensors are not able to measure such quantities. The raw data thus consists of the vehicle's centre of mass position and orientation in the global reference frame. We then computed the time derivatives to measure the absolute velocity $\tilde{v}_x$, $\tilde{v}_y$ and acceleration $\dot{\tilde{v}}_x$, $\dot{\tilde{v}}_y$, as well as the yaw rate ω and it's time derivative $\dot{\omega}$. The training inputs $\boldsymbol{X}$ are thus the front and rear slip angles evaluated as in equations (8) and (9), where the velocities in the vehicle body frame are evaluated as:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos(-\eta) & -\sin(-\eta) \\ \sin(-\eta) & \cos(-\eta) \end{bmatrix} \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \end{bmatrix} \tag{10}$$

The training labels $\boldsymbol{Y}$ are the lateral forces in the tire body frame and have been evaluated by solving the equations of motion of the body in the absolute frame of reference:

$$\begin{bmatrix} F_x \\ F_{y,f} \\ F_{y,r} \end{bmatrix} = \begin{bmatrix} \cos\eta & -\sin\eta & -\sin\eta \\ \sin\eta & \cos\eta & \cos\eta \\ 0 & l_f & -l_r \end{bmatrix}^{-1} \begin{bmatrix} \dot{\tilde{v}}_x/m \\ \dot{\tilde{v}}_y/m \\ \dot{\omega}/I_z \end{bmatrix} \tag{11}$$

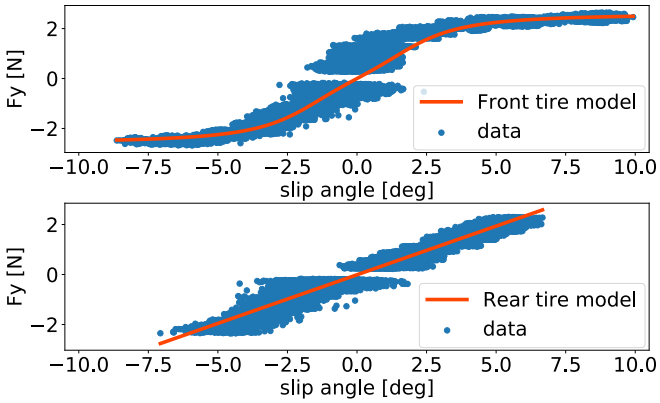


Fig. 6: Lateral tire force model for the front tire (top) and rear tire (bottom).

a	1.72	d	28.88	$\tilde{a}$	1.64	$\tilde{d}$	1.66
b	13.32	e	5.99	$\tilde{b}$	0.33	$\tilde{e}$	0.38
c	0.29	g	−0.15	$\tilde{c}$	0.02	C_r	0.39
D	2.98	C	0.69	B	0.29	E	−3.07

TABLE III: Identified parameter values.

Where F_x is the total longitudinal force in the vehicle frame, $F_{y,f}$ and $F_{y,r}$ are the front and lateral forces in vehicle frame, and $I_z \approx 0.006513\text{Kgm}^2$ was estimated considering the robot as a rectangle with uniformly distributed mass of size $l \times w$, where $w = 0.1\text{m}$ is the width of the vehicle. The lateral forces in the tire frame of reference are finally evaluated as:

$$\hat{F}_{y,f} = \sin(-\delta)F_{x,f} + \cos(-\delta)F_{y,f} \tag{12}$$

$$\hat{F}_{y,r} = F_{y,r} \tag{13}$$

Where $F_{x,f} = \frac{1}{2}F_x$ since the vehicle features a 4 wheel drive and we assume that the motor force is equally shared by front and rear axle. Figure 6 shows the obtained data and fitting results. Notice that since the rear tire does not reach saturation we chose to model it with a simple linear relation instead of equation (7), i.e., as:

$$F_{y,r} = C_r\alpha_r \tag{14}$$

V. Use Cases

This section presents an overview of work that featured DART as a test bed[1]. From a functional point of view all these applications can be seen as a tracking problem, where a vehicle needs to follow a path at a certain reference speed. This reference speed may be fixed, or may be evaluated at runtime based on the robot's global position or on the position and/or velocity of other robots. To perform this planning task the robot needs a good dynamic model in order to track the reference velocity, access to its own pose

[1]The aim of this section is to showcase the platform's capabilities thus we will not provide an in-depth discussion on the scientific merits of the described experiments.

(position and orientation) in the global reference frame, and access to the other robots' poses if needed. This can be achieved by means of an external motion capture system if it's available, or by means of on-board localization. The latter is however a viable option only for low to medium speeds, since the two key components are a sensor to perceive the environment, such as a camera or a lidar, and odometry data provided by the kinematic bicycle model (see section IV). In the remainder of the section we will describe the main features of each kind of experiment, focusing on the functional requirements the platform needed to meet.

1) Distributed MPCC [23]: We presented a distributed Model Predictive Contouring Control algorithm (D-MPCC) for a team of robots. Each robot aims at following a certain path at a given reference speed, while avoiding collisions with other agents. This is achieved through a distributed computation scheme that also accounts for possible packet loss over the communication network. The algorithm was tested in an intersection crossing, shown in Figure 7, and a lane merging scenario. In both cases the robots need to be aware of each other's position and intended future trajectory in order to avoid collisions. To achieve this a good vehicle model is required, to ensure a consistent behaviour.

2) Persistent monitoring [24]: A team of robots is tasked with monitoring a certain area. Each robot is equipped with omnidirectional sensors that are able to detect a target within a certain range. By relying on Lissajous curves and time-inverted Kuramoto dynamics, all vehicles follow the same smooth path within the designated area and adjust their speed based on the current position of the preceding and following vehicles, as shown in Figure 8. The emerging behaviour of the mobile sensor network is guaranteed to detect a moving target within bounded time and avoid collisions among agents. To successfully carry out the experiments the vehicles need to follow a highly curved path while accurately control their speed, since the latter needs to be adjusted according to the local path curvature and to the position of the other robots.

3) Vehicle platooning: Platooning refers to the problem of vehicles driving along a relatively straight path while maintaining a certain distance among each other. The typical application is heavy duty trucks driving on a highway, where for small inter-vehicle distances significant fuel efficiency can be gained due to air drag reduction. To achieve this behaviour the vehicles need to share information on their current speed and position. From a practical point of view, the main challenge is that experiments require a long straight path, thus an external motion capture system will typically not be large enough, requiring the robots to rely on on board sensors for localization. Thanks to the lidar and on board odometry data this can be achieved using standard ROS libraries for Simultaneous Localization And Mapping (SLAM) and Adaptive Monte Carlo Localization (AMCL). Another significant challenge is to carefully adjust the steering in order to limit lateral deviations from the path. This is why we upgraded the servomotor used to steer the robot. Figure 9 shows the robot using on-board localization and steering and

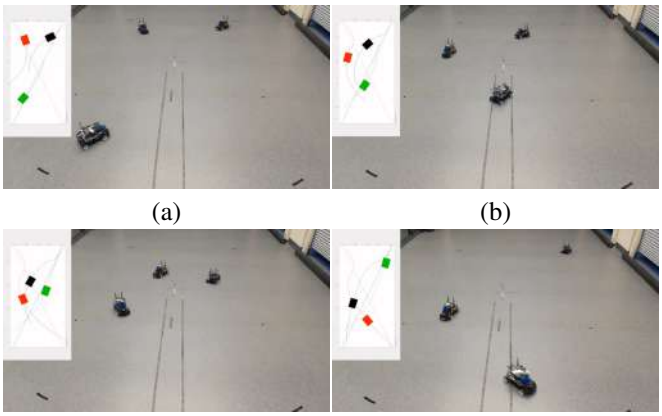


Fig. 7: A team of robots navigate through an unsupervised intersection crossing using a distributed MPC scheme. This figure has been taken from [23].

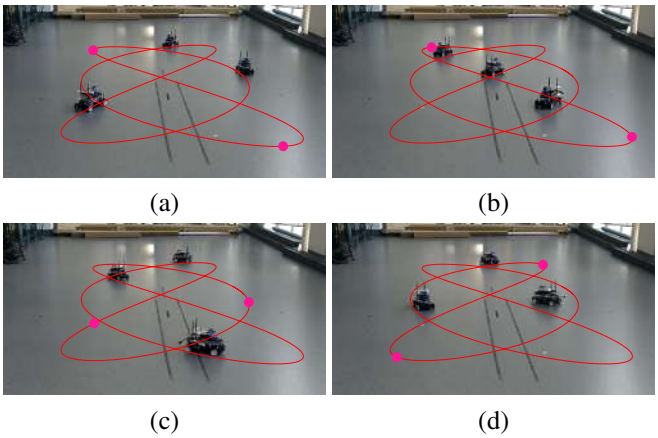


Fig. 8: A team of robots following a Lissajous curve under a time-inverted Kuramoto dynamics feedback controller. This figure has been taken from [24] .

velocity controllers to follow a straight path.

4) Contouring MPC [25]: We present a Model Predictive Contouring Control algorithm (MPCC) that also includes the information on the local path curvature, called Curvature-Aware MPCC. The new formulation features an improved estimation of the progress along the path and consequently more reliable lane boundary constraint satisfaction. Furthermore it features less cost function terms and is thus easier to tune. As far as experiments are concerned the main requirement for the platform is to exhibit consistent behaviour in order to highlight the differences due to the specific algorithm's formulation. This requires a good vehicle model. Figure 10 shows the robot following a highly curved path while avoiding collision with a virtual dynamic obstacle.

VI. Conclusions

This paper introduced the Delft's Autonomous-driving Robotic Test bed (DART), a small-scale car-like robot suitable for (multi-robot) motion planning and control. Compared to other available small-scale platforms DART maxi-

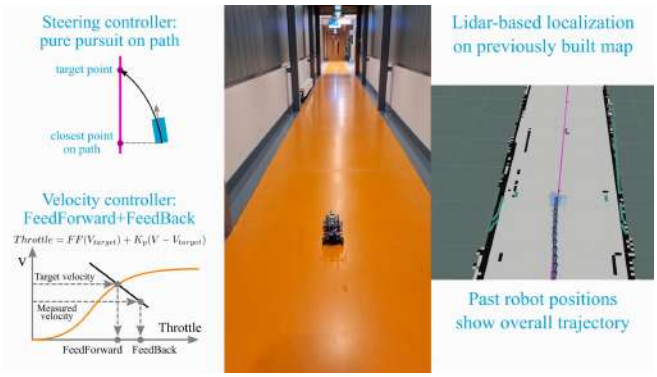
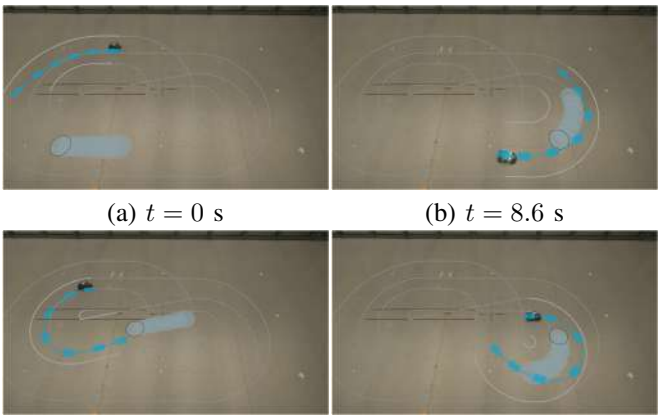


Fig. 9: A robot following a straight line using on-board sensors for both localization and state feedback controllers. This image was taken from [26].



(a) $t = 0$ s (b) $t = 8.6$ s

(c) $t = 16.6$ s (d) $t = 22.4$ s

Fig. 10: A robot following a highly curved path with the Curvature-Aware MPCC controller. This figure has been taken from [25].

mizes the use of available off-the-shelf hardware and features a lower number of custom parts, making it cost-effective and easier to reproduce. We provide a system identification procedure to obtain kinematic and dynamic bicycle models that allow the platform to be used for a wide range of applications, as demonstrated by the number of published works that featured DART as a test bed. Finally, we provide a GitHub repository containing building instructions, the data and code used for the identification, as well as a simulation environment and some readily-available low-level controllers.

REFERENCES

[1] E. Juliussen, "Robotaxis: What Is Going On?" 8 2023. [Online]. Available: https://www.eetimes.eu/robotaxis-what-is-going-on/

[2] Y. Rajan, "Top 5 truck platooning brands syncing convoy trucks via connective technology," 1 2022. [Online]. Available: https://www.verifiedmarketresearch.com/blog/top-truck-platooning-brands/

[3] N. Winton, "Computer Driven Autos Still Years Away Despite Massive Investment," 2 2022. [Online]. Available: https://www.forbes.com/sites/neilwinton/2022/02/27/computer-driven-autos-still-years-away-despite-massive-investment/

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[5] Y. R. Khusro, Y. Zheng, M. Grottoli, and B. Shyrokau, "MPC-based motion-cueing algorithm for a 6-DOF driving simulator with actuator constraints," *Vehicles*, vol. 2, no. 4, pp. 625–647, 2020.

[6] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang, and M. Yang, "Choose your simulator wisely: A review on open-source simulators for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, pp. 1–19, 2024.

[7] ROBOTIS, "TurtleBot." [Online]. Available: https://www.turtlebot.com/

[8] Duckietown, "Duckiebot." [Online]. Available: https://get.duckietown.com/products/duckiebot-db21?variant=40700056895663

[9] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, "Autorally: An open platform for aggressive autonomous driving," *Control Systems Magazine*, vol. 39, pp. 26–55, 2019.

[10] "MIT RACECAR," https://racecar.mit.edu/.

[11] "BARC," https://closestnum20.com/barc_v4-0/.

[12] S. S. Srinivasa, P. E. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi, "Mushr: A low-cost, open-source robotic racecar for education and research," *ArXiv*, vol. abs/1908.08031, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201125218

[13] "Dockey Car," https://http://docs.donkeycar.com/.

[14] L. Lorenzo, "DART," https://github.com/Lorenzo-Lyons/DART, 2024.

[15] "JetRacerProAI," https://www.waveshare.com/wiki/JetRacer_Pro_AI_Kit.

[16] L. Ljung, "Perspectives on system identification," *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[17] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[18] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.

[19] G. Baffet, A. Charara, and D. Lechner, "Estimation of vehicle sideslip, tire force and wheel cornering stiffness," *Control Engineering Practice*, vol. 17, no. 11, pp. 1255–1264, 2009.

[20] 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 (IV)*. IEEE, 2015, pp. 1094–1099.

[21] "Characteristics of Brushed DC Motors," https://techweb.rohm.com/product/motor/brushed-motor/brushed-motor-basic/209/.

[22] W. Menke, "Chapter 9 - detecting and understanding correlations among data," in *Environmental Data Analysis with MatLab® or Python (Third Edition)*, third edition ed., W. Menke, Ed. Academic Press, 2022, pp. 277–317. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323955768000064

[23] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed nonlinear trajectory optimization for multi-robot motion planning," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 809–824, 2022.

[24] M. Boldrer, L. Lyons, L. Palopoli, D. Fontanelli, and L. Ferranti, "Time-inverted kuramoto model meets lissajous curves: Multi-robot persistent monitoring and target detection," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 240–247, 2022.

[25] L. Lyons and L. Ferranti, "Curvature-aware model predictive contouring control," 2023.

[26] Reliable Robot Control lab, "Lidar-based lane following." [Online]. Available: https://youtu.be/QgXHbPtGkdc