DonkieTown: a Low-cost Experimental Testbed for Research on Autonomous Cars

Emmanuel Larralde-Ortiz, Alberto Luviano-Juárez, Flabio Mirelez-Delgado and Diego Mercado-Ravell*

***Abstract*—In this work, *DonkieTown* is introduced, an affordable and scalable platform for research on autonomous vehicles. The experimental framework was developed in the Robot Operative System (ROS). The platform integrates multiple small scale autonomous vehicles called *Asinus Cars*, which are equipped with at least a camera, odometer, and onboard computer. The vehicles are Differential Drive Robots (DDR), forced by software to behave as car-like vehicles. DonkieTown incorporates a low-cost localization system to provide the real-time vehicles' pose, by means of external cameras which detect ArUco markers, then Kalman Filters (KF) are used to track and estimate the pose of each vehicle. The platform includes a base station computer with a graphical interface for monitoring the system. DonkieTown also includes a series of algorithms to facilitate autonomous driving, such as communication, tracking, object detection, obstacle avoidance, control, trajectory tracking, etc. Moreover, a centralized vehicular network is implemented to allow communication between the agents and the base station, where the agents can share information about their state, obstacles, maneuver intentions, etc. To facilitate the research on autonomous cars in Latin America, the developed libraries are released as open source. Real-time experiments demonstrate the performance of DonkieTown in autonomous driving missions, such as following a lane while avoiding Donkey-like obstacles, and collaborative autonomous driving in convoy.**

***Index Terms*—self-driving cars, low-cost testbed, mobile robotics, autonomous driving, ROS.**

I. Introduction

Autonomous cars are already assisting people by providing reliable and safe transportation services, handling parking problems, and eliminating a substantial number of accidents previously caused by human errors [1]. In addition, Connected and Automated Vehicles (CAVs) will reshape transportation and mobility by communicating and conveying real-time information (e.g., position, speed, sensor data, maneuver intentions, etc.). CAV technology also provides opportunities to improve vehicle energy efficiency [2], and introduces cooperative autonomous driving capabilities as explained in [3].

Even though CAVs will enable brand-new transportation functionalities, new collaborative driving architectures must be developed to ensure the reliability and serviceability of CAVs-based systems. In order to construct such architectures, and to explore more benefits of collaborative driving, numerous experiments should be performed in the real world. Nowadays using real-sized smart vehicles is still unfeasible, especially for educational institutions in underdeveloped countries, like in Latin America, hence there is the need to develop scalable and affordable experimental platforms with mobile robots instead of constructing a full-scale smart city testbed like in [4], which costed 10 million USD and covers 32 acres.

Fig. 1. DonkieTown, a low-cost scalable experimental platform for autonomous cars. Here, a lemniscate-like road is depicted, showing the *Asinus Cars* autonomously navigating while avoiding donkey-like obstacles.

Previously, some initiatives of platforms for education and research on mobile robotics have been adopted with success, such as [5], the open mobile robotics platform designed and marketed for the Robotic Operating System (ROS) [6]. Focusing on Automated Vehicles, we can encounter some recent projects [7]–[9], but most of them are too expensive to scale up for some cooperative autonomous driving since they use advanced sensors and high-performance computers. Taking that into account, other works have appeared with simplified mobile robot platforms, for example [10]–[12].

In [10] a fleet of 16 Cambridge Minicars operates in cooperative driving experiments and autonomous control strategies. The Cambridge Minicars are low-cost mobile robot platforms with Ackermann steering, each fits within a $75 \times 81 \times 197$ mm box and costs \$76.50 USD. The University of Delaware Scaled Smart City [11] is a scale-size smart city with $1:25$-scale Ackermann-steering mobile robots, pretty much like the Cambridge Minicar. Both use a central computer to drive each car. The major difference between those two robots is that the former uses no sensors and a Raspberry Pi Zero W as the main onboard computer, while the latter includes an ultrasonic sensor and a Raspberry Pi 3B+. The main concern about [10]

Emmanuel A. Larralde-Ortiz and Alberto Luviano-Juárez are with the Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas UPIITA, Instituto Politécnico Nacional IPN, Mexico e-mail: elarraldeo1700@alumno.ipn.mx, aluvianoj@ipn.mx.

Flabio Mirelez-Delgado is with the Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas UPIIZ, Instituto Politécnico Nacional IPN, Mexico e-mail: fmirelezd@ipn.mx

Diego Mercado-Ravell is with Investigadores CONACYT at Center for Research in Methematics, CIMAT AC, campus Zacatecas, Mexico. *Corresponding author e-mail: diego.mercado@cimat.mx

and [11] is that they rely on costly commercial 3D motion capture systems such as Vicon and OptiTrack, which most Latin American universities cannot afford.

The Duckietown platform [12] is a well-known worldwide initiative for AI and robotics education, its hardware is comprised of DuckieBots and DuckieTowns. Duckiebots are low-cost differential-drive robots with multiple onboard sensors and either a Raspberry Pi or NVIDIA Jetson Nano computer. DuckieTowns are the urban environments: roads and the signage that the robots use to navigate around. In DuckieTown, instead of using an expensive 3D tracking system, Duckiebots navigate independently by identifying simplified road patterns and signage marked with Augmented Reality Tags. Also, they can communicate maneuver intentions with LEDs like normal vehicles do with turn signal lights.

In this work, it is presented an affordable platform (Fig. 1) primarily conceived for education on robotics and autonomous driving and additionally for exploring different possibilities enabled by inter-vehicular communication. To do so:

1) a cheap 2D localization/tracking system has been implemented instead of an expensive 3D motion capture system.
2) The Asinus car is introduced, functionally similar to Duckiebots, but simplified to make them cheaper, while still being capable of performing all navigation operations onboard, contrasting with other cooperative driving platforms whose driving algorithms are computed by an external processor.
3) An artificial neural network is added on to detect potential donkey-like pedestrians and to perform reactive obstacle avoidance.
4) An inter-vehicle messaging protocol, based on international standardization initiatives, has been implemented by software, making DonkieTown the first scaled platform with such capability.

A video explaining the DonkieTown platform can be found at https://youtu.be/ZRRlJkVQ5IM.

Besides the implementation of a messaging protocol, the contribution of this work is the integration of multiple well-known techniques into an affordable and scalable platform, that not only could be used to teach the basics of mobile robotics but also to rapidly validate real-time experiments of sophisticated cooperative autonomous driving approaches.

The paper is organized as follows: in Section II, all elements of the platform (localization, vehicles, base station, road, and vehicular communication) are described; in Section III the driving software stack is addressed, including lane following, obstacle detection and avoidance, and the top layer behavioral who facilitates the cooperative driving feature; in Section IV a series of experimental results are analyzed. Finally, in Section V concluding remarks are discussed.

II. The Donkietown Platform

DonkieTown consists of one or more Differential-Drive Robots (DDR) called *Asinus cars*, a base station, a localization system, and a series of trusted techniques that easily allow the implementation and validation of different strategies for collaborative autonomous driving.

A. The Asinus car

To be aligned with the objective of building an affordable platform, the so-called *Asinus Cars* (AC) (Fig.2) were designed to limit the number of sensors and selecting low-cost but capable materials, actuators, and computing devices while aiming to build 1:10-scale compact cars.

Despite the fact that car-like robots using Ackermann Steering mimics real commercial cars, differential drive robots (DDRs) are cheaper, easier to design, and their motion may be limited by software to behave as car-like vehicles. This limitation is accomplished by saturating its control inputs taking into account a configurable minimum turning radius. Being DDRs, the ACs use one front Pololu caster wheel with 3/4" metal ball (pololu item #: 955) and two Bringsmart GB12-N20B micro metal brushed 6V DC gear motors with encoder, each coupled with one HobbyPark plastic wheel with a rubber tire. Wheel diameter and width are 60 mm and 26 mm each.

The chassis is a 0.15×0.27 m black acrylic sheet, structural elements are mounting screws, acrylic sheets, and 3D printed PLA plastic pieces; the sensor system is as simple as one RGB camera employing a Sony IMX219 sensor and 120° FOV (field of view) lens and the magnetic encoders that came fixed to each motor, and the onboard computer is a Nvidia Jetson nano 2GB Developer Kit. The onboard computer is connected to a Wireless Local Area Network (WLAN) via one TP-Link TL-WN725N USB Wi-Fi adapter. The onboard computer is wired to the RGB camera and one Arduino nano which controls the motors' spinning speed by means of the magnetic encoders and one DRV8833, a dual H-bridge motor driver, controlled by the Arduino with PWM (Pulse-Width Modulation). Note that the use of the Arduino is optional, and it can be replaced to use only the Jetson nano, but it allows an easier integration with other sensors. Additionally, each AC includes one unique ArUco Marker [13] for global localization, as explained later. See Fig.2 for more details.

Finally, all electronic components are supplied by a Luckso power bank featuring a 10,000 mAh battery (dedicating one of its two ports to power the motors and the other for the remaining electronic devices), and the total cost for each vehicle was less than $\$200.00$ USD, while the list-price of a similar platform as the one in [12] is around $\$369.00$ USD.

B. Base Station

A personal computer is used as Base Station for data visualization, inter-vehicle communication management, and to host a simulation environment. In order to allow data sharing and direct data visualization, ROS is used in every computer involved (i.e., upper camera's computer, vehicles' onboard computer, and Base Station). Although ROS Noetic is the latest version of ROS and ROS2 solves most of the problems that are not contemplated in ROS, ROS Melodic is favored to keep compatibility with former robotics platforms such as [7]. On the other hand, this base computer is used to run a simulation environment using Gazebo. The simulator was primarily developed to test beforehand the algorithms presented in this work, and to evaluate the performance of

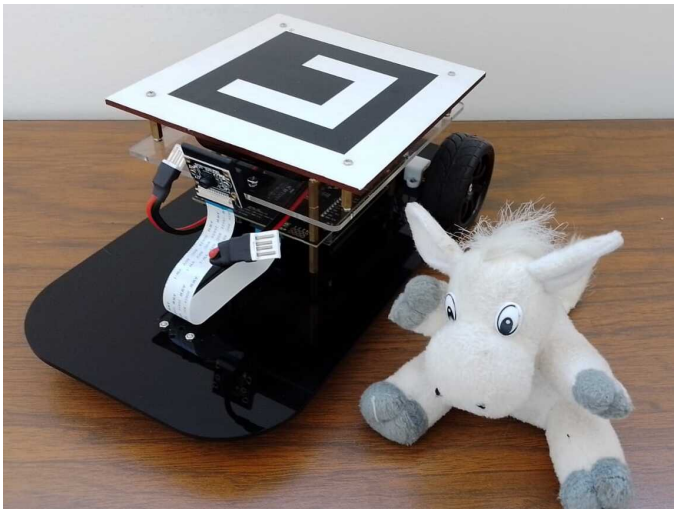


Fig. 2. The *Asinus Car*: A differential-drive robot with an ArUco marker at the top and a stacked circuity system for pedestrian detection and localization, state variables control, message production and communication, and general onboard computing. The top marker is used to compute vehicle's relative pose for the global localization function.

single-board computers (e.g., Jetson nano 2GB and Raspberry pi 3B+) while running those algorithms.

C. Localization

Since both communication and decisions depend on proximity, a reliable estimation of real-time absolute position is required for each vehicle.

Real-size Intelligent Vehicles use Global Positioning System (GPS) for global localization. In addition, several sensors (such as Lidars, RGB-D cameras, radars, etc.) are used to get their own and other agents' relative positions within the road. DonkieTown is a reduced and simplified world where roads are known beforehand, to not only let cars navigate with maps but also to estimate the global position of each car and the relative position with respect to other agents, such as Donkie-like pedestrians.

In contrast with a GPS, or expensive motion capture systems, the proposed localization system (Fig.3) requires an upper camera and a fixed number of reference markers. The upper camera is a static USB RGB (red, green, blue) camera connected to an embedded computer (e.g., Raspberry Pi 3B+), pointing towards the workspace; all the markers are ArUco Fiducial Markers [14], with reference markers fixed to the road and mobile markers fixed to each car. The work [15] has a detailed explanation of how the ArUco library works and how it can be implemented for mobile robot tracking.

As shown in Fig.3, let $\mathcal{F}_{RMi}$ denote the coordinate frame (also called coordinate system) of a reference marker, with $i \in \mathbb{Z}^+$. Similarly, let $\mathcal{F}_{MMv}$ be the coordinate frame of the mobile marker attached to the vehicle v; $\mathcal{F}_{UC}$ be the coordinate frame of the upper camera, and $\mathcal{F}_{RD}$ be the inertial coordinate frame fixed to the center of the workspace. Thus, $\mathbf{H}_A^B \in SE(3)$ defines the homogeneous transformation from any coordinate frame $\mathcal{F}_A$ to any frame $\mathcal{F}_B$, where $SE(3)$ is the so-called Special Euclidean group in 3 dimensions [16].

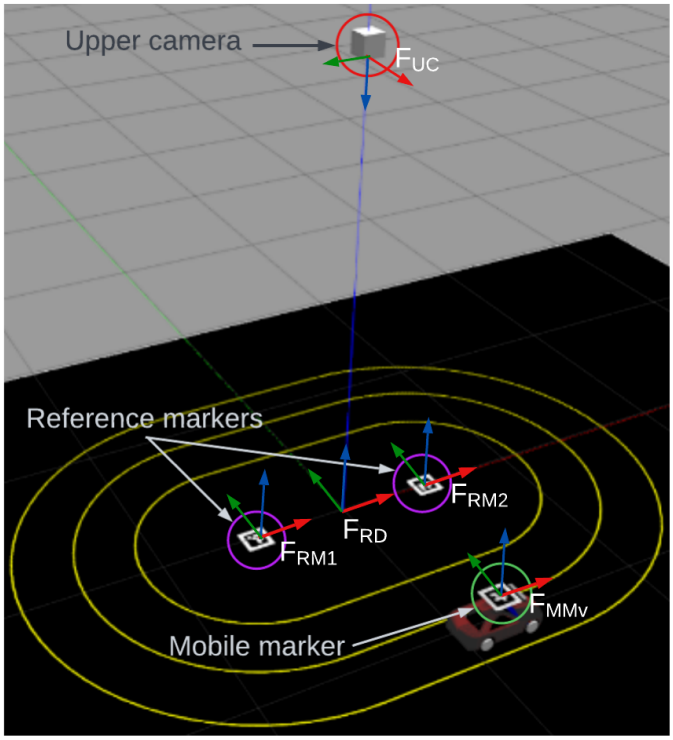


Fig. 3. DonkieTown's localization system. $\mathcal{F}_{RM1}$, $\mathcal{F}_{RM2}$ denote the coordinate frame attached to some of the markers stuck to the road (reference markers), while $\mathcal{F}_{MMv}$ and $\mathcal{F}_{UC}$ are for the coordinate frames of a car's marker (mobile marker) and the upper camera, respectively. Global localization system may consist of one or several upper cameras pointing towards different areas of the road's surface. Each upper camera is connected to a ground computer that processes images to detect mobile markers. *Asinus Cars* post-process upper camera's outcome to estimate their absolute position.

The pose estimation computation of each of the markers is decentralized by splitting the procedure as follows:

1) When a frame has been captured, the embedded computer connected to the camera (1) broadcasts the result of ArUco's *detectMarkers()* function; and (2) computes and broadcasts each $\mathbf{H}_{RMi}^{UC}$ transformation.

2) The vehicle's onboard computer periodically updates the Kalman Filters (KF) [17] for tracking the position of the four ordered corners of its mobile marker. If the marker is within the set of detected markers, the data will be taken for the next steps and will be used to correct the KF prediction, otherwise, the rejected square closest to the prediction is taken and used for filter correction. With the four corners' position, the transformation $\mathbf{H}_{MMv}^{UC}$ is computed.

Since reference markers are fixed to the road, the transformation $\mathbf{H}_{RD}^{RMi}$ is known. Hence, the transformation $\mathbf{H}_{RD}^{MMv}$ is calculated as follows:

$$\mathbf{H}_{RD}^{MMv} = \mathbf{H}_{RD}^{RMi}\mathbf{H}_{RMi}^{UC}\left(\mathbf{H}_{MMv}^{UC}\right)^{-1} \quad (1)$$

Although $\mathbf{H}_{RD}^{MMv}$ includes both the 3D orientation and 3D position of the mobile marker of vehicle v, every vehicle is considered to drive only on the $x-y$ plane. Therefore, vehicle´s orientation $\theta \in \mathbb{S}^1$ and absolute position coordinates $(x, y) \in \mathbb{R}^2$ are respectively taken from the rotation and translation parts of $\mathbf{H}_{RD}^{MMv}$.

The computation of the translation vector of a homogeneous transformation like $\mathbf{H}_{RD}^{MMv}$ is straightforward, but the rotation component is not that evident. It would be required to translate the orientation part of the homogeneous transformation to a set of angles and select the one which corresponds to the vehicle heading (for a comprehensive analysis of rigid motion in robotics, the reader is referred to [16]).

1) Corners Tracking: As stated before, 4 KFs are used to estimate the corners coordinates of the ArUco markers in the image space. Whenever a mobile marker is not found at a given time, corners estimations are matched with the rejected square corners that produce the minimum sum of Euclidean distances, while corners order is kept by means of the Hungarian Algorithm [17], [18].

A discrete state space model representation with uncertainty and no control inputs is given as follows for the KF:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}\mathbf{x}_{k-1} + \omega_{k-1} \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \nu_k \end{aligned} \tag{2}$$

where $\mathbf{F}$ stands for the transition matrix, $\mathbf{H}$ is the measurement matrix, $\mathbf{x}_k$ is the state variable vector at step k, $\mathbf{z}_k$ is the measurement vector at step k as well, and $\omega \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\nu \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ are the process and measurement noises, respectively, with normal distribution and covariance matrices $\mathbf{Q}$ and $\mathbf{R}$. Considering the first-order kinematics

$$\mathbf{x}_i = \begin{bmatrix} x_i & v_{xi} & y_i & v_{yi} \end{bmatrix}^T \tag{3}$$

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where $x_i, y_i \in \mathbb{R}$ are the corner image coordinates at step $i \in \mathbb{N}$; $v_{xi}, v_{yi} \in \mathbb{R}$ are the corner velocities defined by

$$\begin{aligned} v_{xi} &= \frac{x_i - x_{i-1}}{\Delta t} \\ v_{yi} &= \frac{y_i - y_{i-1}}{\Delta t} \end{aligned} \tag{5}$$

with $\Delta t \in \mathbb{R}$ as the fixed time step.

The ArUco's library returns the image coordinates (x, y) of the four ordered corners for each detected marker, hence the measurement matrix $\mathbf{H}$ is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6}$$

In order to speed up estimation, the process noise is treated as independent between different state variables. In this case

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & 0 & 0 \\ 0 & q_{22} & 0 & 0 \\ 0 & 0 & q_{33} & 0 \\ 0 & 0 & 0 & q_{44} \end{bmatrix} \tag{7}$$

The same assumption is applied to the measurement uncertainty matrix, i.e.

$$\mathbf{R} = \begin{bmatrix} r_{11} & 0 \\ 0 & r_{22} \end{bmatrix} \tag{8}$$

D. Road Assembly

The road is built from multiple 0.6×0.6 m puzzle foam mats with two lanes of 0.3 m width which lines are drawn by 2 cm width masking tape (see Fig. 1).

Different road topologies can be built easily by joining straight, turn-left, turn-right, and intersection road pieces. For example, an ellipsoid road, or a lemniscate-like road with an intersection crosswalk as the one depicted in Fig. 1.

E. Inter-Vehicle Communication

Instead of an inter-vehicle communication protocol, DonkieTown implements an inter-vehicle message inspired by the Cooperative Awareness Message (CAM) [19] and the Decentralized Environmental Notification Message (DENM) [20] from the European Telecommunications Standards Institute.

More precisely, DonkieTown's inter-vehicle message conveys the vehicle's absolute position, longitudinal speed, reference lane, driving direction (i.e., whether it is moving forward or in reverse), heading angle, driving state (detailed in Sec. III), maneuver intentions, identification number and message's time stamp. Messages are periodically produced and broadcasted by *Asinus Cars* while the Base Station collects and redirects them based on the following requirements:

- Point-to-point latency shall not exceed 0.1 s.
- To be considered as a message target, the distance from the message producer to the message target shall not exceed 3.0 m.

III. Autonomous Driving

The ACs' autonomous driving function architecture (or the decision-making architecture) is hierarchically decomposed into three components: A plain behavioral layer, a motion planning module, and a local feedback control system.

The behavioral layer is responsible for both selecting car's regulated speed and the most appropriate driving state at any time based on the driving task and the surrounding environment. Driving states are Finite State Machines (FSMs) which govern the vehicle's directions. All Asinus Cars are loaded with the same FSM, but the FSM should be enlarged if necessary when trying different use cases. Asinus Car's base FSM consists of four self-explanatory states: *Lane Following*, *Stop*, *Lane Change Request*, and *Lane Change Granted*. With these four states, each vehicle can start and finish a route with regulated speed, overtake other vehicles and avoid collisions.

The motion planning module chooses one vector field (e.g., Fig. 4) from a set of stored vector fields based on the AC state history. Let us say, the vehicle v is following lane A. Vehicle v's FSM is in the *lane following* state and the current vector field is centered in the lane A. Inside these vector fields, a displacement vector is assigned to each point of the workspace. Displacement vectors are calculated with a fixed step size and direct the vehicle from its current position to the next position the vehicle should reach.

In order to execute the path described by the current displacement vector, a local feedback control system is used to determine appropriate speeds for each wheel and correct tracking errors while two equal but independent feedback

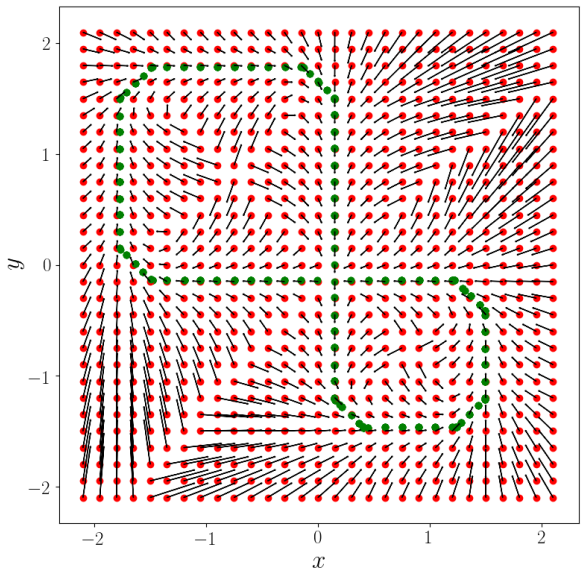


Fig. 4. Example of a vector field centered with one lane of an lemiscate-like road. Vectors direct vehicles from their current position to the position they should reach.

control algorithms are used to regulate the speed of each wheel.

To introduce the feedback control algorithm for the path following, let us define an eccentric point (p, q) as depicted in Fig. 5. μ is the eccentricity of (p, q) over the main axis of a Differential-Drive robot, thus, assuming from a unicycle kinematic model, the first-order kinematics of that very point could be expressed as follows [21]

$$\begin{bmatrix} \dot{p} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\mu\sin\theta \\ \sin\theta & \mu\cos\theta \end{bmatrix} \begin{bmatrix} s \\ \omega \end{bmatrix} \tag{9}$$

where $\theta \in \mathbb{S}^1$ is the same vehicle's heading as stated in Subsection II-C, and $s \in \mathbb{R}$, $\omega \in \mathbb{R}$ represent respectively the longitudinal and angular velocities of the DDR.

Given the model presented in Eq. (9), the suggested feedback control system is

$$\begin{bmatrix} s \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{1}{\mu}\sin\theta & \frac{1}{\mu}\cos\theta \end{bmatrix} \begin{bmatrix} \dot{p}^* - k_{p1}(p - p^*) \\ \dot{q}^* - k_{p2}(q - q^*) \end{bmatrix} \tag{10}$$

Here the superscript * denotes the desired value of a given variable at that time, e.g., p^* denotes the desired value of the coordinate p while p itself is its actual current value. In addition, both the symbols $\dot{p}^*$ and $\dot{q}^*$ denote the time derivative of the coordinates of the desired path. Finally, $k_{p1}, k_{p2} \in \mathbb{R}^+$ are positive constant control coefficients that are adjusted empirically.

The coordinates (p, q) and the orientation θ are directly taken from the outcome of our localization system, even though the center of the mobile marker may not coincide with the main axis of the Asinus Cars, the orientation θ is the same for both the mobile marker and the Asinus Car.

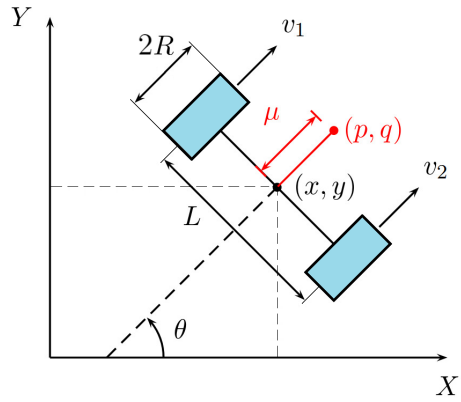


Fig. 5. Eccentric point over the main axis of a Differential-Drive Robot. Taking an eccentric point rather than taking the vehicle's axis midpoint for path following control leverages the usage of the non-singular feedback control system shown in Eq. (10).

The actual actuator inputs are evaluated from Eq. (10), considering the DDR geometry

$$\begin{aligned} s &= \frac{R}{2}(v_1 + v_2) \\ \omega &= \frac{R}{L}(v_2 - v_1) \end{aligned} \tag{11}$$

where R denotes the radius of each wheel, L is the longitude of the main axis or the distance between both wheels, and v_1 and v_2 are the angular speed of the left and right wheels, respectively.

Two independent PI (proportional-integral) controllers are used to compute the correspondent duty cycle of a PWM signal to regulate both angular speeds v_1 and v_2. Here, v_1 and v_2 play the role of reference values, while measurements from the wheel encoders, mentioned in Subsection II-A, are the actual current angular speeds. As done in Eq. (10), proportional and integral gains are tuned empirically.

A. Obstacle Detection

Lane change must be performed to avoid collisions with other vehicles and obstacles such as *pedestrians*. In DonkieTown, each vehicle conveys enough information to allow nearby vehicles to avoid vehicular collisions, however, *pedestrians* are passive obstacles that are not connected to anything. DonkieTown's pedestrians are donkey-like teddy bears (see Fig. 2) and any of the vehicles can detect and determine if there is a potential collision involving either *pedestrians* or other vehicles.

ACs are able to detect pedestrians via their frontal RGB cameras, and Single-Shot Detectors (SSD) [22] with a MobileNetv1 [23] Backbone Artificial Neural Network, initially trained with the PASCAL VOC Dataset [24] and retrained with a manually collected dataset of donkey-like teddy bears, to leverage transfer learning.

The Artificial Neural Network processes the video stream captured from the onboard camera and returns a bounding box for each detected *pedestrian*. Let's say that, in a given moment, one AC's camera captures a frame like the one in Fig. 6, which

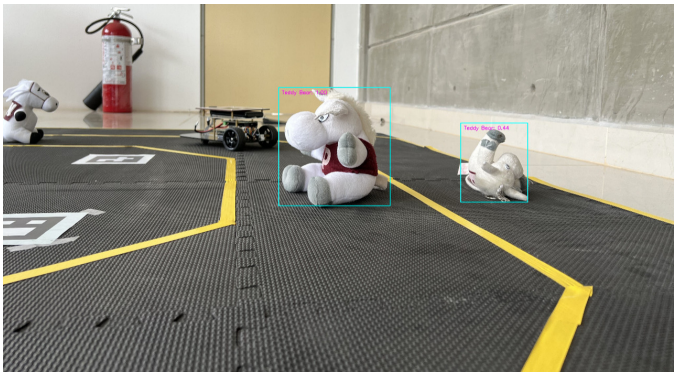


Fig. 6. Example of a video frame captured by an AC's onboard camera and post-processed with DonkieNet: a Mobilenetv1-SSD neural network pre-trained with an open dataset and re-trained with an in-house hand-labeled dataset of donkey-like teddy bears.

is a 1280×720 px RGB video frame. The Artificial Neural Network will return a collection of (x_c, y_c, w, h) per detected donkey-like teddy bears, where (x_c, y_c) is for the bounding box center while w and h denotes width and height of the same bounding box.

To compute the global position of each detected donkey-like *pedestrian*, distortion effects must be corrected. After that, the pinhole camera model is used to transform image coordinates into global coordinates. Let's say that a point lying in the workspace is located at (X, Y, Z), the capturing camera's global pose is represented with the transformation $\mathbf{H_C}$, the camera's intrinsic matrix is denoted as $\mathbf{K}$, and (u, v) corresponds to the image coordinates of the pixel where that point is in the image. Hence, the relation between a point in the 3D world and its image coordinate is given by

$$\left[\tilde{p}\right] = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H_C^{-1}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{12}$$

where $\tilde{p} = (\tilde{u}, \tilde{v}, \tilde{w})$ is the homogeneous coordinate of the world point $P = (X, Y, Z)$ in pixel coordinates. The non-homogeneous image-plane pixel coordinates are derived from their homogeneous counterpart as follows:

$$u = \frac{\tilde{u}}{\tilde{w}}, \; v = \frac{\tilde{v}}{\tilde{w}} \tag{13}$$

When considering that the midpoint of the lower edge of a bounding box corresponds to the base of a donkey-like teddy bear which is on the road surface, the third workspace coordinate (Z), corresponding to the height, turns out to be 0. At a point in time, $\mathbf{H_C}$ is constant and reproduced from the vehicle's current absolute pose and the relative pose of the camera with respect to the vehicle's frame. Therefore, Eq. (12) leads to:

$$\begin{bmatrix} C_{11} - C_{31}u & C_{12} - C_{32}u \\ C_{21} - C_{31}v & C_{22} - C_{32}v \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} C_{34}u - C_{14} \\ C_{34}v - C_{24} \end{bmatrix} \tag{14}$$

where, $\mathbf{C}$ denotes the projection matrix or the camera calibration matrix [25].

$$\mathbf{C} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{H_C^{-1}}$$

Finally, it is possible to calculate the global position $(X, Y, 0)$ of each detected *pedestrian* when solving Eq. (14). Since the map is already known and lanes are already represented in memory, (X, Y) coordinates are used to determine whether a *pedestrian* is blocking the road by simply getting the distance from (X, Y) to the road lanes.

IV. Experimental Results

In order to assess the usability of the DonkieTown testbed, two cases of study for autonomous driving were implemented and evaluated. For the first test, one AC was used for a lane-following task, without obstacles on the road. Two independent upper cameras covered half of the workspace, as shown in Fig.1. Both upper cameras had been configured to capture 1280×720 px RGB video at 30 FPS (frames per second). Fig. 7 depicts the path constructed from raw data (i.e., pose delivered by the localization system) alongside the path drawn with the outcome of the Kalman Filter. Any pose returned by the localization system representing a displacement greater than two standard-deviations of a series formed with the last 10 measurements is considered an aberrant result and ignored by the KF. The second experiment consists of a cooperative autonomous driving mission, where three AC travel through the road in a convoy formation. In the beginning, each car identifies its *leading car* (the nearest car in front), except for the front car, which acts as the *convoy leader*. The *convoy leader* is the only one with the *DonkieNet* enabled to detect Donkie-like *pedestrians*, and the only one allowed to change its driving state. When the *convoy leader* determines that the current lane is blocked, it will change its driving state to perform a lane change if possible, otherwise, it will change its driving state to stop. The *Convoy leader* driving state is periodically propagated, via inter-vehicle messages, from the *convoy leader* to the convoy tail through each *following car*. More information from inter-vehicle messages is taken into account to keep the convoy shape.

Fig. 8 shows the paths drawn by the three ACs employed in the convoy test, and the estimation of the absolute position of the *pedestrians* detected by the *convoy leader*, depicted as gray dots. Data was recorded after the convoy traveled a few laps and upper cameras were configured as in the lane following task. The AC was able to successfully travel the lemniscate shape road in an autonomous convoy formation while avoiding the obstacles. These experiments demonstrate the capability of the DonkieTown and the AC to easily implement and evaluate the performance of intelligent cars in autonomous missions. A video showing some experiments is provided at https://youtu.be/ZRRlJkVQ5IM.

V. Concluding Remarks

In this work, we have presented "DonkieTown", a non-expensive and scalable platform for education and research on autonomous cars and extensible to automated multi-car

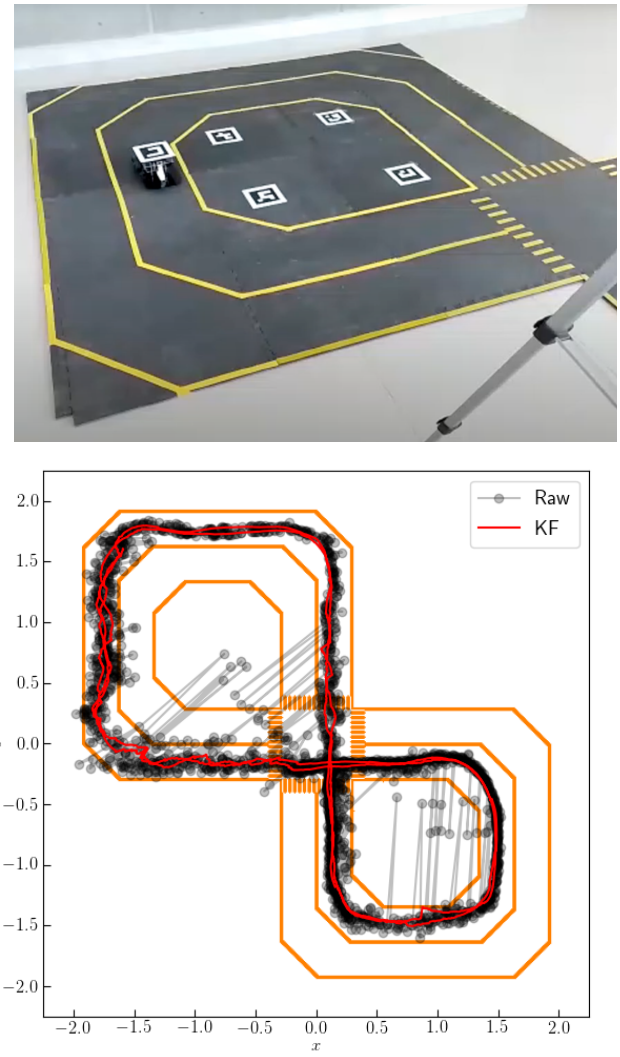


Fig. 7. Experimental results of a single AC autonomously following a road lane (left). Path followed by the AC (right) as estimated by the ArUco markers (gray dots) and the KF (red line). The KF filters out aberrant measurements and provides a good pose estimate in real time.

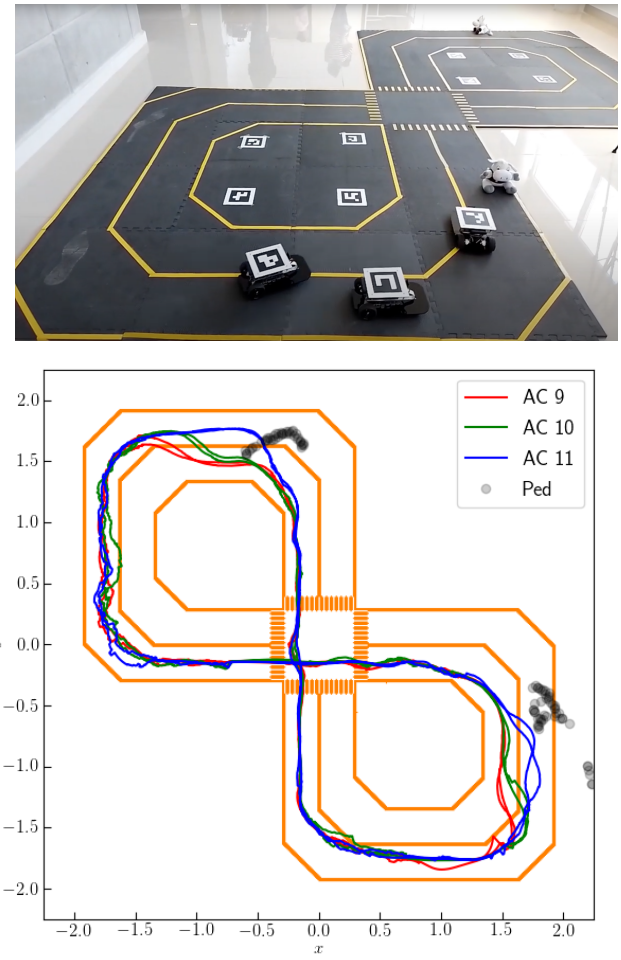


Fig. 8. Three-car convoy with maneuver intentions communication, avoiding a donkey-like pedestrian (left). Red, green, and blue lines represent the estimated path produced by the AC 9, 10, and 11, respectively; gray points are the estimated position of donkey-like obstacles detected by the convoy leader (right). No donkeys were harmed during the experiments.

scenarios. Our platform is particularly attractive for low-budget robotics labs since it does not require expensive motion capture infrastructure and our Differential-Drive Robot allows us to considerably reduce the cost of the platform, especially as the number of agents increases. "DonkieTown" is released as open-source software, becoming one of the very few openly available platforms that already implements inter-vehicle messaging. We have proposed a driving architecture which demonstrated to be sufficient for egocentric autonomous tasks such as lane following and obstacle avoidance, and demonstrated its applicability for cooperative autonomous driving in convoys by using inter-vehicle communication.

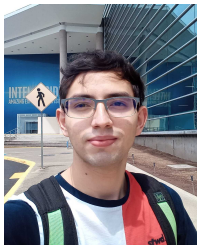
Hitherto, DonkieTown has been used in some workshops with the participation of undergraduate engineering students, graduate students, teachers, entrepreneurs and hobbyists from around Mexico. Any enthusiast with coding experience is a potential user since we have developed tools and software layers to simplify its usage. Furthermore, basic-education students could learn to program using DonkieTown and graduate students may develop from classic perception and planning algorithms up to light AI algorithms.

All materials are open source and available at our GitHub repository: https://github.com/L4rralde/DonkieTown. We seek for other people in the robotics community in Latin America to contribute to its enhancement and growth. As future work, we plan to use our platform for testing more interesting multi-car scenarios, which include intelligent road intersections, multi-lane convoys, and cooperative overtaking. Also, we aim at proposing more sophisticated autonomous driving algorithms, and testing state-of-the-art strategies for control, motion planning, obstacle avoidance, reinforcement learning, cooperative driving, etc.

REFERENCES

[1] I. Yaqoob, L. U. Khan, S. M. A. Kazmi, M. Imran, N. Guizani, and C. S. Hong, "Autonomous driving cars in smart cities: Recent advances, requirements, and challenges," *IEEE Network*, vol. 34, no. 1, pp. 174–181, 2020.

[2] M. Taiebat, A. L. Brown, H. R. Safford, S. Qu, and M. Xu, "A review on energy, environmental, and sustainability implications of connected and automated vehicles," *Environmental Science & Technology*, vol. 52, no. 20, pp. 11449–11465, 2018. PMID: 30192527.
[3] B. Häfner, V. Bajpai, J. Ott, and G. A. Schmitt, "A survey on cooperative architectures and maneuvers for connected and automated vehicles," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 380–403, 2022.
[4] J. R. Sayer, "Connected/automated vehicle and infrastructure research [michigan mobility transformation facility (mtf)]," tech. rep., University of Michigan Transportation Research Institute, 2021.
[5] D. Singh, E. Trivedi, Y. Sharma, and V. Niranjan, "Turtlebot: Design and hardware component selection," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pp. 805–809, 2018.
[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, (Kobe, Japan), May 2009.
[7] K. Alomari., R. Mendoza., S. Sundermann., D. Goehring., and R. Rojas., "Fuzzy logic-based adaptive cruise control for autonomous model car," in *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - ROBOVIS,*, pp. 121–130, INSTICC, SciTePress, 2020.
[8] J. Gonzales, F. Zhang, K. Li, and F. Borrelli, "Autonomous drifting with onboard sensors," in *Advanced Vehicle Control*, 2016.
[9] 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.
[10] N. Hyldmar, Y. He, and A. Prorok, "A fleet of miniature cars for experiments in cooperative driving," *CoRR*, vol. abs/1902.06133, 2019.
[11] L. Beaver, B. Chalaki, A. M. Mahbub, L. Zhao, R. Zayas, and A. Malikopoulos, "Demonstration of a time-efficient mobility system using a scaled smart city," *Vehicle System Dynamics*, vol. 58, no. 5, p. 787–804, 2020.
[12] 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.
[13] S. Roos-Hoefgeest, I. A. Garcia, and R. C. Gonzalez, "Mobile robot localization in industrial environments using a ring of cameras and aruco markers," in *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6, IEEE, 2021.
[14] S. Garrido-Jurado, R. Muñoz, F. Madrid, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, p. 2280–2292, 06 2014.
[15] A. Botta and G. Quaglia, "Performance analysis of low-cost tracking system for mobile robots," *Machines*, vol. 8, no. 2, 2020.
[16] M. W. Spong, *Robot Dynamics and Control*. USA: John Wiley & Sons, Inc., 1st ed., 1989.
[17] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," in *2016 6th International Conference on System Engineering and Technology (ICSET)*, pp. 109–115, 2016.
[18] M. García-Venegas, D. A. Mercado-Ravell, L. A. Pinedo-Sánchez, and C. A. Carballo-Monsivais, "On the safety of vulnerable road users by cyclist detection and tracking," *Machine Vision and Applications*, vol. 32, no. 5, p. 109, 2021.
[19] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basica Set of Applications; Part 2: Specific of Cooperative Awareness Basic Service.," standard, European Telecommunications Stadards Institute, Sofia Antipolis Cedex, France, Nov. 2014.
[20] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basica Set of Applications; Part 3: Specifications of Descentralized Environmental Notification Basic Service.," standard, European Telecommunications Stadards Institute, Sofia Antipolis Cedex, France, Nov. 2014.
[21] A. Lopez-Gonzalez, E. Ferreira, E. G. Hernández-Martínez, J.-J. Flores-Godoy, G. Fernandez-Anaya, and P. Paniagua-Contro, "Multi-robot formation control using distance and orientation," *Advanced Robotics*, vol. 30, no. 14, pp. 901–913, 2016.
[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 21–37, Springer International Publishing, 2016.
[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.
[24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.
[25] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Publishing Company, Incorporated, 1st ed., 2013.



Emmanuel Alejandro Larralde-Ortiz was born in Lazaro Cardenas, Michoacan, Mexico. He is a Recent College Graduate of UPIITA-IPN in Bachelor of Science in Mechatronics Engineering. In 2022, he worked as Technical Graduate Intern at Intel Guadalajara Design Center where he contributed to functional validation of hardware accelerators for next generation data center processors. Nowadays Emmanuel is a Hardware Engineer at Intel where he is now contributing to verify a novel CPU architecture. His research interests are mobile robotics, self-driving cars and digital electronics design.

Alberto Luviano-Juárez received the B.S. degree in mechatronics engineering from the National Polytechnic Institute (IPN), Mexico City, Mexico, in 2003, the M.Sc. degree in automatic control from the Department of Automatic Control, CINVESTAV-IPN, in 2006, and the Ph.D. degree in electrical engineering from the Department of Electrical Engineering, CINVESTAV, in 2011. He is currently with the Graduate and Research Section, UPIITA IPN. His current research interests include robust estimation and control in robotic and mechatronic systems, as well as energy storage systems.

Flabio Mirelez-Delgado was born in Zacatecas, Mexico. He completed his Engineering degree in Communications and Electronics at Universidad Autónoma de Zacatecas from 2005 to 2010, and his Master's degree in Robotics and Advanced Manufacturing at Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV) from 2010 to 2012. He worked for 3 years at the Universidad Tecnológica de Coahuila, and since 2018 he is full time associate professor at Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas IPN (UPIIZ-IPN). His main interests are mobile robotics, automatic control and artificial vision.

Diego Alberto Mercado-Ravell received the Ph.D. degree from the University of Technology of Compiègne UTC, France. He has held post-doctoral positions at the Mechanical and Aerospace Department at Rutgers, the State University of New Jersey, USA, and CINVESTAV Mexico. He is currently CONACYT researcher at CIMAT-Zacatecas, in Mexico, and member of the national research system (SNI), level I. His research topics include robotics, modeling and control, unmanned aerial/underwater vehicles, autonomous navigation, state estimation, data fusion, computer vision and deep learning applications.