Visual Urban Navigation for Mobile Robots: Implementation in the Duckietown Environment

Shima Akbari, Nima Akbari, Giuseppe Oriolo, Sergio Galeani

2025 International Conference on Control, Automation and Diagnosis (ICCAD) | 979-8-3315-1191-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICCAD64771.2025.11099311

***Abstract*—This paper presents a vision-based control framework for the autonomous navigation of wheeled mobile robots in city-like environments, including both straight roads and turns. The approach leverages Computer Vision techniques and *OpenCV* to extract lane line features and utilizes a previously established control law to compute the necessary steering commands. The proposed method enables the robot to accurately follow the lanes and seamlessly handle complex maneuvers such as consecutive turns. The framework has been rigorously validated through extensive simulations and real-world experiments using physical robots equipped with the *ROS* framework. Experimental evaluations were conducted at the DIAG Robotics Lab at Sapienza University of Rome, Italy, demonstrating the practicality of the proposed solution in realistic settings. This work bridges the gap between theoretical control strategies and their practical application, offering insights into vision-based navigation systems for autonomous robotics. A video demonstration of the experiments is available at https://youtu.be/tDvpwSj8X28.**

I. Introduction

Autonomous navigation in maze-like environments, such as rectilinear street networks with corners and junctions, presents a challenging problem for wheeled mobile robots. Specifically, the challenge lies in enabling a robot to move as close as possible to the center of the lanes while relying solely on visual information and without prior knowledge of the environment. This problem has significant implications for urban navigation and autonomous driving research.

Several prior studies have addressed indoor navigation and path-following tasks. For example, navigation based on odometric or map-based localization using preregistered images or known landmarks has been explored in [1], [2], and [3].

This work was partially supported by the European Union – Next Generation Eu - under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 1 Investment 4.1 - Decree No. 351 (09th April 2022) of Italian Ministry of University and Research - Concession Decree No. 2152 (28th December 2022) of the Italian Ministry of University and Research, Project code D93C22000850005, within the Italian National Program PhD Program in Autonomous Systems (DAuSy).This work also has been partially supported by the Italian Ministry for Research in the framework of the 2020 Program for Research Projects of National Interest (PRIN). Grant No.2020RTWES4

Shima Akbari is with Dept. of Electrical and Information Engineering, Polytechnic of Bari, Bari, Italy, and Dept. of Civil and Computer Engineering, Tor Vergata University, Rome, Italy, Email: s.akbari@phd.poliba.it, shima.akbari@uniroma2.it

Nima Akbari is with Dept. of Informatics and Mathematics, Basel University, Switzerland, Email: nima.akbari@unibas.ch

Giuseppe Oriolo is with Dept. of Information, Automation and Management Engineering (DIAG), Sapienza University of Rome, Rome, Italy, Email: oriolo@diag.uniroma1.it

Sergio Galeani is with Dept. of Civil and Computer Engineering, Tor Vergata University, Rome, Italy Email: sergio.galeani@uniroma2.it

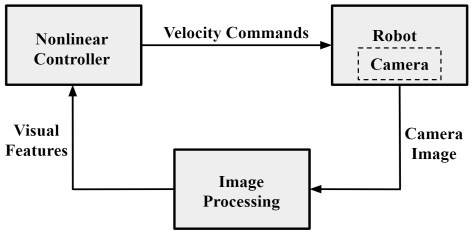


Fig. 1: General scheme

Planning techniques have been used for obstacle navigation in [4]. However, these approaches rely on localization systems, environment mapping, or predefined landmarks, which may not always be available in real-world scenarios. Alternative strategies are required for scenarios where localization systems are absent, and no a priori information about the environment exists.

In this paper, we propose a novel approach based on *Image-based Visual Servoing (IBVS)* to address these shortcomings. Our method generates steering velocity commands to maintain the robot position at the center of the lanes, using only visual information from the robot onboard camera. Unlike previous approaches, our method does not require localization systems, environmental maps, or predefined landmarks, making it a strategic solution for unstructured and unknown environments. The effectiveness of this method is demonstrated in both simulation in Duckietown simulator [5] and real-world experiments with Duckiebots [6] using the *ROS* programming framework [7].

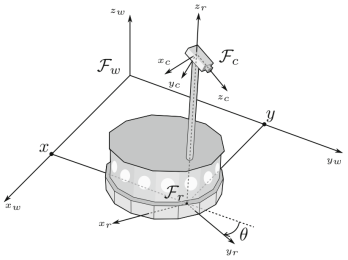
The remainder of this paper is organized as follows: Section II provides an overview of the Duckietown environment, highlighting its structural characteristics. Section III describes the feature-based controller, including the visual features model and control law. Section IV details the image processing steps used to extract relevant features from the camera footage. The lane-following control approach is presented in Section V, including the integration of visual data and control logic. Simulation results are discussed in Section VI, followed by experimental validations in Section VII, which demonstrate the effectiveness and practicality of the proposed method. By relying solely on visual feedback and leveraging natural features, our method represents a significant advancement in autonomous navigation for urban-like environments. This approach is implemented and tested using Duckiebots as well as the Duckietown simulator to validate the proposed

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on September 08,2025 at 13:08:18 UTC from IEEE Xplore. Restrictions apply.

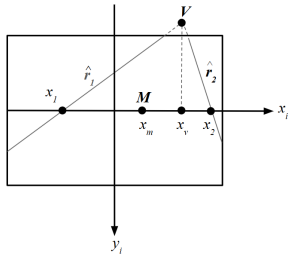
(a)

(b)

Fig. 2: (a) presents the Duckietown environment while (b) demonstrates the Duckietown simulator



(a)



(b)

Fig. 3: (a) demonstrates the frames of interest for derivation of the visual features model and design of the vision-based controller: world (F_w), robot (F_r) and camera frame (F_c) [1], (b) the two guidelines always intersect at the vanishing point V. The midpoint of the segment between the intersection points of the guidelines with the abscissa axis defines the middle point M [1].

method in a city-like environment featuring dynamic obstacles, intersections, and other challenges.

II. The Duckietown Environment

The Duckietown environment, depicted in Fig. 2a and Fig. 2b, is a structured, city-like setting comprising straight roads and turns. Within this environment, the robot is required to navigate while adhering to *right-hand traffic rules*. This behavior is facilitated by the design of the road markings: white shoulder lines delineate the edges of the roads, while a broken yellow line marks the centerline. Consequently, the robot must maintain the white line on its right and the broken yellow line on its left throughout navigation (see Fig. 2b). The primary objective is to enable the robot to autonomously navigate this environment by traveling along the lanes (*"lane navigation"*) and then performing appropriate maneuvers at intersections (*"turning"*). To address the problem effectively, it is essential to formulate it in a way that allows the extraction of relevant features from the camera footage. These features serve as critical inputs for deriving a control law to achieve the desired navigation objectives. The subsequent sections provide a detailed explanation of the methods employed to identify and extract these necessary features.

III. Feature-Based Controller

A. Mathematical Modeling

The wheeled mobile robot under consideration is modeled using unicycle kinematics [8]. Its configuration is represented by the generalized coordinates (x, y, θ), which describe the position and orientation of the robot frame F_r relative to the world reference frame F_w, as illustrated in Fig. 3a. The quantities v and ω correspond to the robot driving and steering velocities, respectively. v is assumed to be constant as in a *cruise velocity*, while ω is treated as the input variable for control [9]. A pinhole camera is rigidly mounted on the robot at a fixed height h from the ground, with its optical axis aligned with the robot heading and tilted by an angle γ. Importantly, the controller to be developed is robust to non-zero offsets in the camera position [1], as depicted in Fig. 3a.

B. Virtual Guidelines

The visual features utilized in the unicycle controller are the abscissa coordinates of the *vanishing point* and the *middle point*, denoted by x_v and x_m, respectively [9]. In the Cartesian space, the virtual guidelines associated with the street lines may intersect at a suitably far distance. Due to the effects of perspective projection, these guidelines always intersect on the image plane at the vanishing point V. The middle point M is defined as the midpoint of the segment formed by the intersection points of the guidelines with the abscissa axis [1] (see Fig. 3b). To compute these quantities, it is necessary to project the two corridor guidelines onto the image plane [9] and [1].

C. Control Law

A control law is required to ensure the fact that the robot should stay at the center of the lanes. Specifically, the robot x and θ coordinates must be stabilized to zero because F_w has been chosen so as to be aligned with the lane (so that $x = 0$ and $\theta = 0$ means staying at the center of the lane). In the Duckietown environment, there are lanes that are orthogonal to the x axis, so either we assume that F_w will be rotated at each turn or the control law needs to be adapted for achieving $y = 0$ and $\theta = \pi/2$. Stabilization of both the middle point and the vanishing point abscissas to the origin of the image plane ensures that the robot achieves the purpose of this project.

Given the virtual guidelines associated with the lanes, the dynamics of the middle point and vanishing point can be expressed as functions of the unicycle driving and steering velocities, v and ω, respectively [1]. Based on these dynamics, an expression for ω can be derived to ensure that $x_m \to 0$. To achieve this, a positive design constant k_p is introduced:

$$\omega = \frac{k_1}{k_1 k_3 + x_m x_v} \left(-\frac{k_2}{k_1} v x_v - k_p x_m \right) \tag{1}$$

It can be shown that the dynamics of x_m are exponentially stable, while the exponential convergence of x_v is proven using

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on September 08,2025 at 13:08:18 UTC from IEEE Xplore. Restrictions apply.

Lyapunov stability arguments. Alternatively, the convergence of x_m inherently implies the convergence of x_v due to the system non-holonomic constraints [1].

Using this control law, we first focus on solving the lane navigation problem and deal with turning later.

IV. Image Processing

To achieve the objectives of this project without explicit knowledge of the robot pose, two virtual guidelines must be defined. These guidelines represent the lines of the street on which the robot operates and are extracted using OpenCV. The image captured by the robot mounted camera is processed to extract the necessary features. The key image processing steps are outlined below:

A. Denoising the Frame

The initial step in most image processing pipelines is denoising the input frame, which is accomplished in OpenCV using kernels. A *Gaussian kernel* [10] is applied to blur the video stream captured by the robot camera, thereby reducing noise and artifacts.

B. Grayscaling

Grayscaling involves converting the image from its original color space (e.g., RGB, CMYK, or HSV) to shades of gray. This step reduces the computational complexity of subsequent image processing operations.

C. Edge Detection

After converting the camera footage to grayscale, edges are detected using the Canny edge detection algorithm, which is known for its robustness and flexibility [10]. To enhance the clarity and robustness of detected street lines, edge detection is applied separately to the yellow and white street lines because of the right-hand traffic rules features of the duckietown environment as it was explained in Sec. II. Therefore, the yellow (left line) and white (right line) colors are filtered separately in the HSV color space [10]. The Canny edge detection algorithm is then applied independently to these filtered segments, producing clearer and more distinct edges indicating each line.

D. Region of Interest (RoI) Selection

To further simplify the image, a Region of Interest (RoI) is defined to highlight only the areas containing street lines. This is achieved by creating a mask with the same dimensions as the original frame. A polygonal region, representing the area of interest, is filled with ones, while the remaining areas are filled with zeros. By applying a bitwise AND operation between the original image and the mask, only the region containing street lines is preserved, while other objects are filtered out.

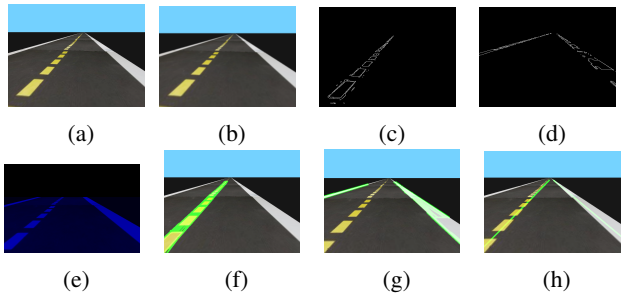


Fig. 4: Verification of the image processing steps in the Duckietown simulator. (a) shows the primary image, (b) depicts the denoised image, (c) and (d) illustrate the extraction of yellow and white edges, respectively, (e) highlights the region of interest, (f) and (g) display the extracted left and right segments, and (h) presents the optimized lines.

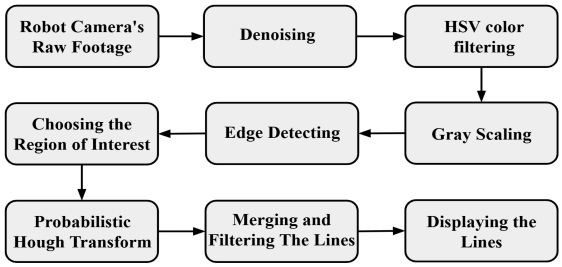


Fig. 5: All image processing steps in a nutshell

E. Line Detection

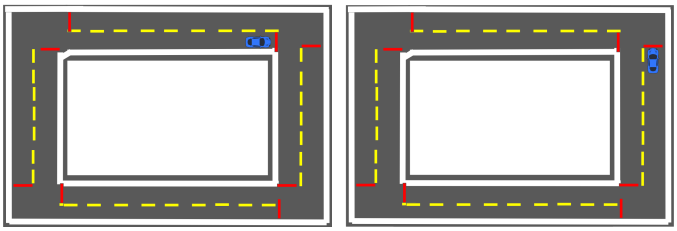
Lines are detected in the processed image using the *Probabilistic Hough Line Transform* [10], which outputs line segments. However, these segments are insufficient for directly computing the middle and vanishing points required for the controller design. To address this, an algorithm is implemented to merge these segments into solid lines. Each line segment is treated as an individual line, characterized by its slope and intercept. Line segments are grouped based on a threshold criterion (e.g., white segments always have positive slopes while the slope of the yellow segments are all negative). Once suitable candidates are identified, their slopes and intercepts are averaged to generate a single representative line for each line. To improve robustness, the computed slope and intercept can be further filtered to refine the final result.

The summarized image processing workflow is illustrated in Fig. 5, while Fig. 4 demonstrates the step-by-step process described in this section.

V. Lane Following Control

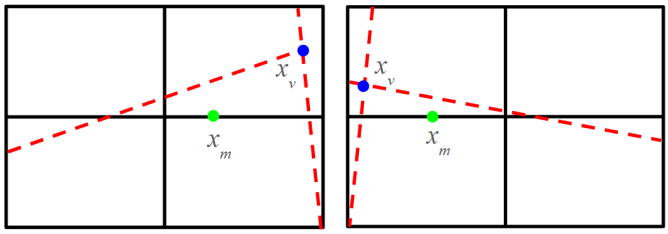
The image processing steps described in Section IV allow the extraction of virtual guidelines corresponding to the street lines from the robot camera footage. These guidelines enable the application of the control law introduced in Section III, which facilitates vision-based control to ensure the robot follows the lane and remains centered within the center of the two street lines. At each frame, the virtual guidelines produce

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on September 08,2025 at 13:08:18 UTC from IEEE Xplore. Restrictions apply.



(a) The time for turning right (b) The time for turning left

Fig. 6: The time for the robot to turn right(a) and left (b) schematically



(a) guidelines for turning right (b) guidelines for turning left

Fig. 7: Artificial guidelines for turning [1]

a middle point and a vanishing point, which in turn generate a suitable steering velocity according to Eq. 1, thereby achieving the primary objective of this paper. However, to address the challenge of turning within the Duckietown environment, a scenario-driven transition system (STS) has been developed. This system triggers the robot transition from lane navigation to a dedicated turning behavior upon detecting a specific environmental scenario. In this case, the scenario relies on a distinguishing feature: a red line orthogonal to the left and right street lines, marking the entry point of square-shaped turns (see Fig. 6a and Fig. 6b). The detection of the red line is performed using HSV-based color filtering via OpenCV. To ensure precise activation of the turning behavior, a region of interest (RoI) is applied to the image to mask areas further away from the robot current position. This ensures that the red line is only detected when it appears horizontal in the robot frame of view, which corresponds to the critical state transition point for initiating a turn. This transitional state is depicted schematically in Fig. 6a and Fig. 6b. Upon detecting the red line, the STS activates a temporary state in which two artificial visual guidelines are introduced. These guidelines are used to define a middle point and a vanishing point, which serve as inputs for calculating the steering velocity necessary for the turn. The artificial guidelines and their corresponding features are illustrated in Fig. 7b and Fig. 7a. Once the turn is complete, the robot transitions back to the lane-following state, where the actual street lines replace the artificial guidelines as the visual references. This seamless transition between states, governed by the STS, ensures continuous and reliable navigation throughout the Duckietown environment.

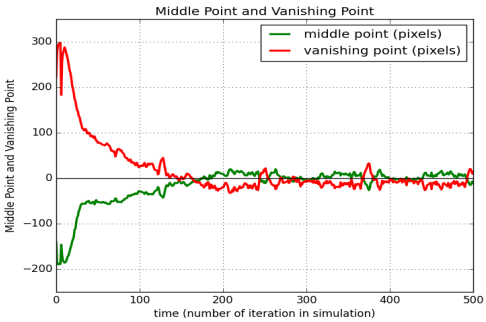


Fig. 8: Time evolution of middle and vanishing point and converging to the center of the lane (with $k_p = 2$)

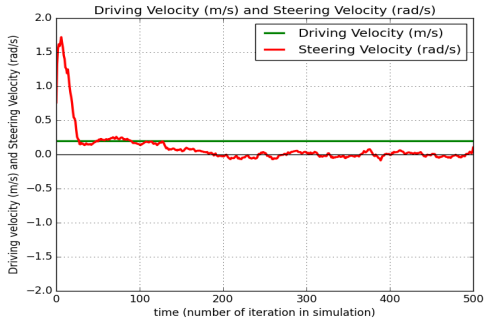


Fig. 9: Time evolution of the computed steering velocity and driving velocity of the robot and converging to the center of the lane. driving velocity is assumed constant and equal to 0.2. (with $k_p = 2$)

VI. Simulation Results

In the following, we have verified the explained method on two different scenarios.

A. Scenario 1: Converging to the Center of the Lane

In the first scenario, demonstrated in the simulator, the robot starts in an initial position slightly deviated from the center of the lane. Upon detecting the lines, the robot adjusts its position to align with the center.

The time evolution of the middle and vanishing points, along with the driving and steering velocities, is illustrated in Fig. 8 and Fig. 9, respectively. It is important to note that while a higher control gain k_p allows for faster response, excessively large values can lead to system instability.

B. Scenario 2: Straight Motion Followed by Consecutive Turns

This simulation demonstrates a scenario where the robot moves straight while maintaining its position as close as possible to the center of the lane. It then encounters two consecutive turns—a left turn followed by a right turn—and performs the required maneuvers based on the STS described in Section V.

The time evolution of the middle and vanishing points during this experiment is shown in Fig. 10, while Fig. 11

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on September 08,2025 at 13:08:18 UTC from IEEE Xplore. Restrictions apply.

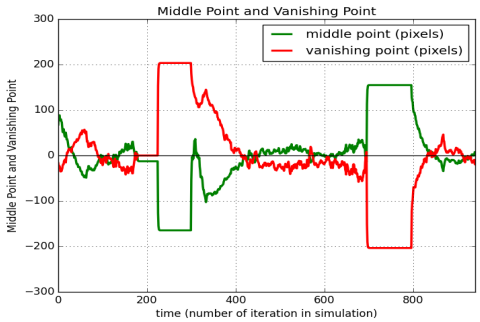


Fig. 10: Time evolution of middle and vanishing point when the robot negotiates one left and one right turn consecutively

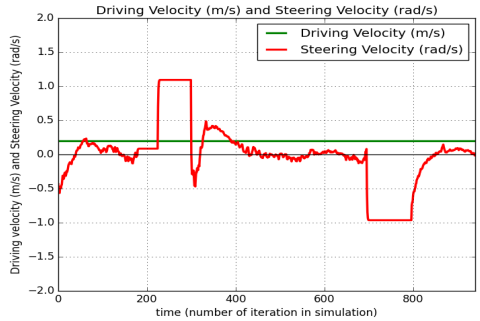


Fig. 11: Time evolution of the computed steering velocity and driving velocity of the robot when the robot negotiates one left and one right turn consecutively. (driving velocity is assumed constant and equal to 0.2)

illustrates the time evolution of the computed steering and driving velocities.

VII. Experimental Results

This project was implemented both in the Duckietown simulator using Python and on physical robots (Duckiebots). A Duckiebot is a wheeled mobile robot with unicycle kinematics, equipped with a Raspberry Pi $3B$, a camera, and two DC motors powered by rechargeable $10Ah$ batteries [6]. The experiments were conducted using the ROS programming framework.

The entire implementation for this project was encapsulated within a single ROS *node*, leveraging a *Publisher/Subscriber* architecture. The subscriber node processed the camera input, while the image processing pipeline computed the middle and vanishing points based on the virtual guidelines. Using these computed points, the control law calculated the appropriate steering velocity. The publisher node then sent this velocity as input to the robot wheels, enabling autonomous navigation along the lane while maintaining a position as close as possible to the center of the street lines. This implementation successfully achieved the project objectives.

The robot wheels and camera were calibrated in accordance with the robot documentation [11]. Additionally, since the

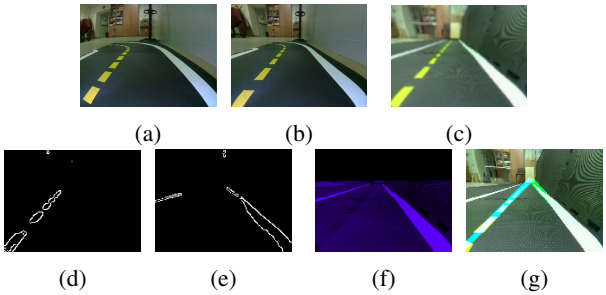


Fig. 12: Illustration of the image processing workflow on live footage from the robot's camera: (a) shows the original capture, (b) presents the rectified image, (c) depicts the denoised version, (d) extracts white edges, (e) extracts yellow edges, (f) delineates the region of interest, and (g) demonstrates the optimized lines

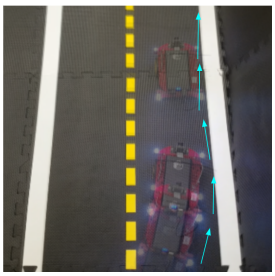


Fig. 13: The convergence of the robot to the center of the lane

camera images are distorted, rectification was performed before proceeding with the image processing pipeline [10], as shown in Fig. 12b.

A. Experiment 1: Converging to the Center of the Lane

This experiment validates the effectiveness of the proposed controller in guiding the robot to the center of the lane by driving the middle and vanishing points to zero. The robot was initially positioned with a deviation of $0 > \theta > -90°$. Upon activation, the robot adjusted its trajectory and successfully aligned itself with the lane center. The results of this experiment are depicted in Fig. 13.

Additionally, the time evolution of the computed steering and driving velocities was recorded using *rosbag* [7] and is shown in Fig. 14.

B. Experiment 2: Negotiating Turns

This experiment evaluates the robot ability to successfully navigate turns using the method discussed in this paper. The robot begins by following a straight road, maintaining its position as close as possible to the center of the lane. Upon reaching a turn, it executes the appropriate maneuver as described in Section V.

Both left and right turns were tested, as shown in Fig. 15a and Fig. 15b. The time evolution of the computed steering and driving velocities was recorded using a rosbag [7] and is illustrated in Fig. 16 and Fig. 17. As observed, the driving

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on September 08,2025 at 13:08:18 UTC from IEEE Xplore. Restrictions apply.

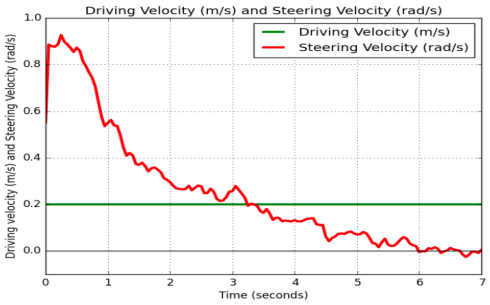
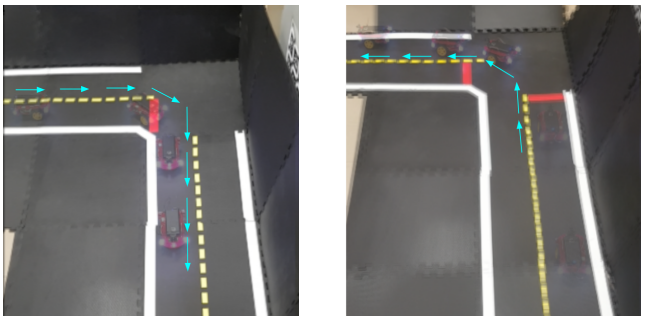


Fig. 14: Time evolution of the computed steering velocity and driving velocity of the robot in the experiment when the initial position is deviated with a $0 > \theta > -90°$. driving velocity is assumed constant and equal to 0.3 m/s



(a) (b)

Fig. 15: Robot negotiating different turns. (a) presents a right turn while in (b) the robot negotiates a left turn

velocity was kept constant while the steering velocity was dynamically computed using Eq. 1.

VIII. Conclusion

This paper proposed a vision-based control framework for lane-following tasks in wheeled mobile robots, validated through both simulations and real-world experiments. The approach effectively maintains the robot position at the center

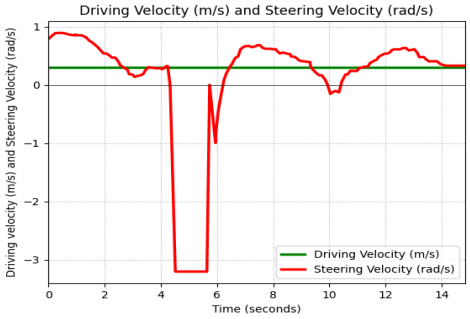


Fig. 16: Time evolution of the computed steering velocity and driving velocity when the robot negotiates a right turn. driving velocity is assumed constant and equal to 0.3 m/s

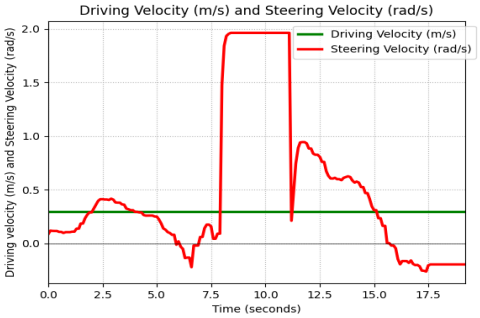


Fig. 17: Time evolution of the computed Steering velocity and driving velocity when the robot negotiates a left turn. driving velocity is assumed constant and equal to 0.3 m/s

of lanes and enables safe left and right turns by relying solely on visual feedback from onboard camera, without requiring external localization systems or pre-mapped environments. The system's modular design and simplicity allow for seamless integration with other robotic systems, making it versatile for diverse urban navigation scenarios. Future research will focus on enhancing the framework to handle complex scenarios, such as autonomous lane corrections, and incorporating obstacle detection and avoidance mechanisms for improved performance in dynamic, real-world environments. These advancements will expand the applicability of the proposed method, confirming its potential as a robust solution for autonomous navigation.

References

[1] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittelli, "Humanoid odometric localization integrating kinematic, inertial and visual information," *Autonomous Robots*, vol. 40, no. 5, pp. 867–879, Jun. 2016. [Online]. Available: http://link.springer.com/10.1007/s10514-015-9498-0

[2] E. A. Rodríguez Martínez, G. Caron, C. Pégard, and D. L. Alabazares, "Photometric Path Planning for Vision-Based Navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 9007–9013, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/document/9197091

[3] J. Ido, Y. Shimizu, Y. Matsumoto, and T. Ogasawara, "Indoor Navigation for a Humanoid Robot Using a View Sequence," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 315–325, Feb. 2009. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0278364908095841

[4] D. Maier, C. Stachniss, and M. Bennewitz, "Vision-based humanoid navigation using self-supervised obstacle detection," *International Journal of Humanoid Robotics*, vol. 10, no. 02, p. 1350016, Jun. 2013, publisher: World Scientific Publishing Co. [Online]. Available: https://www.worldscientific.com/doi/10.1142/S0219843613500163

[5] "Duckietown Github." [Online]. Available: http://github.com/duckietown/gym-duckietown

[6] "Duckietown." [Online]. Available: https://duckietown.com/

[7] "ROS." [Online]. Available: http://wiki.ros.org/

[8] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*, ser. Advanced Textbooks in Control and Signal Processing, M. J. Grimble and M. A. Johnson, Eds. London: Springer London, 2009. [Online]. Available: http://link.springer.com/10.1007/978-1-84628-642-1

[9] J. Toibero, C. Soria, F. Roberti, R. Carelli, and P. Fiorini, "Switching visual servoing approach for stable corridor navigation," in *2009 International Conference on Advanced Robotics*, Jun. 2009, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/5174702

[10] "OpenCV." [Online]. Available: http://opencv.org/

[11] "Duckietown lib." [Online]. Available: http://docs.duckietown.org/daffy

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on September 08,2025 at 13:08:18 UTC from IEEE Xplore. Restrictions apply.