Development Platform for Autonomous Driving in Real and Simulated Environments

Cristian Allione, Santiago Pincin, and Sebastián Zudaire

Instituto Balseiro - Universidad Nacional de Cuyo, Argentina
{cristian.allione, santiago.pincin, sebastian.zudaire}@ib.edu.ar

Abstract. Autonomous car driving has gained popularity in recent years both in industrial and academic environments, leading to numerous commercial and research platforms being constructed for development and testing. Moreover, due to advantages at design time and for testing purposes, several simulation environments have been proposed to simulate the complex dynamics and real-world problems the driving algorithms will face. However, most platforms and simulation environments are not suitable in costs and complexity for research groups beginning to work in the area or for educational purposes. In this work we propose a low-cost 3D-printed hardware 1:10 scale car that allows for development and testing of near SAE level 2 driving automation in software-in-the-loop realistic simulations together with real world scenarios. We demonstrate this by running a lane-centering and a simplified version of adaptive cruise control algorithm in a simulated and real highway-inspired track.

Keywords: autonomous driving · development platform · software-in-the-loop.

1 Introduction

Design and development of autonomous cars has been ongoing for the last decades for commercial products in industrial environments, as well as experimental and testing tools for research purposes (e.g., [3,9]). One of the many challenges researched both by the software and robotics community is to develop safe and correct algorithm implementations of Advanced Driver-Assistance Systems (ADAS) that are sound and bug-free [13]. Different approaches have been proposed to ensure these characteristics on the autonomous driving capabilities. In [11] an overview of several state-of-the-art Deep Learning approaches is shown. A design strategy is presented in [21] which aids in assuring a certain safety level in the overall system. Other works explain in detail their algorithmic implementations that aim at safer handling of real on-road situations [20,29].

The common denominator of these works is that an experimental platform either real or simulated is required to validate (or train in some cases) the proposed algorithms and solutions. Having a full-scaled real or prototype car as in [11,17,20] may seem the preferable option as it allows for validation in the real environment. However, for training and testing of Deep Learning algorithms a realistic simulation of a full-scaled car may be a better option given the reduced time in development to obtain datasets [2,28]. Other works [14,27] have shown that a 1:10 scale real car can be a good platform for development and testing of localization and navigation algorithms. Similarly, one may use a simplified specialized (e.g., [10]) or general purpose (e.g., [12]) simulation environment to develop and test control algorithms.

Not every platform is suitable in costs, building time and/or complexity to be usable by researchers or educators starting to work in this field. Thus, open-source simulated [10] and scale real [23] platforms have been developed to provide easy access to a testing and development environment for these groups. However, existing simulated and real platforms may have an over-simplification of the dynamics or hardware that must be used to fully develop and test ADAS algorithms (e.g., [23] has no camera). Moreover, results in simulation may not transfer fully to the scale real cars as in [12].

In this work we present an open-source, open-hardware [8] research and educational platform that allows for software-in-the-loop simulation and real-world testing, using a simple, yet realistic, simulator and a low-cost 3D-printed scale car. Albeit missing some of the components commonly present in autonomous cars (e.g., LiDAR sensors), we demonstrate how this platform can be used to develop near SAE level 2 autonomous driving capabilities, namely a lane centering and a simplified version of adaptive cruise control algorithms. We present the architecture of the proposed platform and evaluate the ADAS system in a simulated and real track, comparing the results in both scenarios.

This paper is structured as follows. Section 2 presents the related work, and later Section 3 shows an overview of the system and its architecture. We explain the driving capabilities in Section 4 and describe their implementation in real world and simulation in Section 5, which are validated in Section 6. Finally, Section 7 concludes.

2 Related Work

Safety and economic reasons make it convenient to research and develop autonomous capabilities over scaled platforms instead real large-scale vehicles. There are many interesting and successfully projects that reach advanced autonomous driving systems in this way (see [11] for an overview). However, most of them use high-cost components and complex platforms, making them inadequate for beginners in this area. The work in [7] proposes a low-cost platform equipped with lane centering and velocity cruise control. [22] consists on an even lower cost, open 3D-printed and easier to assemble platform which is capable of person following and real-time autonomous navigation.

On the other hand, there are several works that propose a realistic simulation environment to test and validate driving capabilities both in real scale [10] as in small scale prototypes [5,26]. Our proposed platform differs from these works in that it is a simple 3D-printed, affordable and open-source platform that allows for developing and testing of lane centering and speed control algorithms, and also provides a realistic simulation environment with software-in-the-loop and using the main processing hardware as in the real system.

3 System Overview and Architecture

In this section, we discuss the target applications of our proposed system and then present an overview of the developed hardware and software architecture.

As mentioned in Section 1, which car platform is adequate for an application depends on the application itself. Our focus is educators or researchers starting to work on autonomous driving systems, and for this reason we aim at a simple simulated/real platform, but without losing the ability to test autonomous driving capabilities. We aim to demonstrate this ability by implementing control algorithms that maintain vehicle centered on the track and adjust its velocity to the curvature of the road. Thus, for the real vehicle we selected the following specifications:

– 1:10 scale vehicle, which are low-cost and widely used in competitions like *CaroloCup* [4] and many autonomous car researches [7,25,27,33].
– Affordable and readily available electronic components as in [3,22].
– 3D-printed chassis and steering mechanism (e.g., [22,27]).
– Obstacle-free highway-inspired track, similar to [5].
– From the different localization technologies, we choose to only use a front facing monocular camera as in [33]. A common choice in these platforms is a 2D or 3D LiDAR sensor [18,24], but we choose not include them due to the added cost and complexity of the overall system, without being a necessary addition taking into account the previous item.

For the simulated environment, we choose to work with the general-purpose simulator CoppeliaSim, which has seen plenty of use in recent robotic research [30,34]. This simulator has a fully functional educational version which adds to the low-cost philosophy behind the proposed platform.

The main hardware and software components are presented in Fig. 1. Here we see that the *Autonomous Driving Algorithms* will run aboard a Raspberry Pi 3B+ single board computer. This is a common choice [3,6,33] in hardware for many mobile robots as a good compromise between computing resources and monetary cost. These algorithms are able to run with both the real and simulated systems thanks to an Interface API that implements the *Camera & Serial Interface Library* to communicate with the real system, and the *Simulation Interface Library* to communicate with the simulated environment.

On the right side of the architecture we see the similarities between both platforms. The monocular camera (PS3 Eye) can be modeled with a vision sensor that has configurable resolution and Field of View (FOV). 3V Motors, H-bridge (LM2596) and MOC7811 optical encoders are simulated with a revolute joint working as a motor. We programmed into the simulated motor a first-order differential equation which we carefully identified as described in Section 5.1. The steering mechanism translates the angular movement from a MG90S servomotor

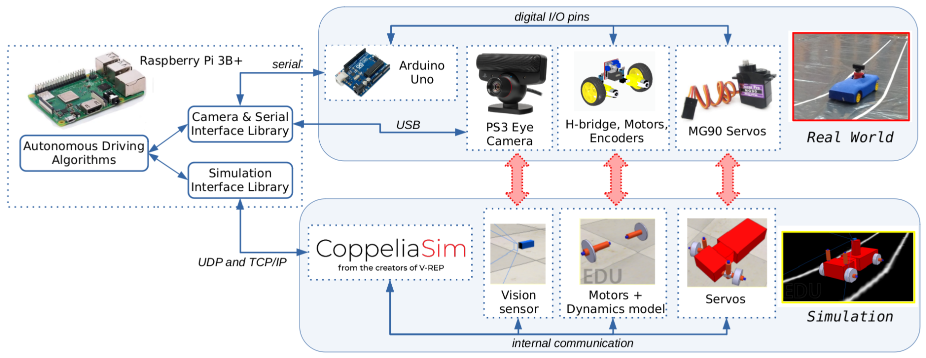


Fig. 1: Architecture of our proposed real and simulated platform. Red arrows indicate similarity relations between simulated and real components. Blue arrows indicate data flow.

to front wheels, with a rate 1:1. It can be easily simulated with a combination of revolute joints. Finally, the Arduino Uno, which handles the communication with most of the electronic components of the system, does not need to be modeled as the simulated environment communicates in a similar way with the simulated components.

The resulting platform allows the autonomous driving algorithms to run in the real world and in a software-in-the-loop simulation using the same computing resources as the real system. Given how we modeled the components from the real system in the simulated environment we are able to produce realistic simulations as we will show in Section 6.

4 Driving Capabilities

For this work, we aim to show that the proposed platform is adequate for near SAE level-2 automation [31], which means that the driver does not need to perform longitudinal and lateral operational driving, although the driver should be ready to intervene at any time. This is a common goal pursued by other projects [17,19]. We will implement lateral driving capabilities through a lane centering algorithm and longitudinal through an adaptive speed controller. There are many methods for autonomous lane centering [32], from which we will use the *Stanley algorithm*, a geometric nonlinear algorithm that was developed and validated on a *DARPA challenge* [35]. For the adaptive speed control we propose controlling the vehicle's velocity based on road curvature measurements (i.e., higher curvature translates to lower target vehicle speed), similar to the safety speed recommender in [15]. With these two algorithms running at the same time we achieve a high level of autonomy as we will see in Section 6. However, we say that we achieve *near* SAE level-2 automation due to the simplified nature of our chosen algorithms.

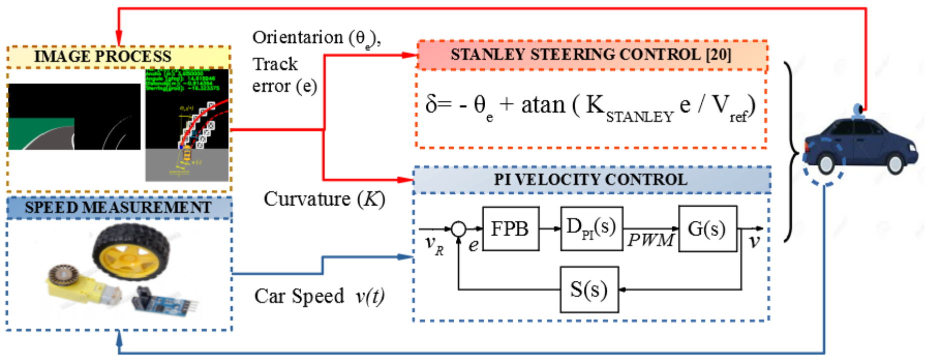


IMAGE PROCESS
SPEED MEASUREMENT
Orientarion (θ_e), Track error (e)
Curvature (K)
Car Speed $v(t)$
STANLEY STEERING CONTROL [20]
$\delta = -\theta_e + \text{atan}(K_{STANLEY}\, e / V_{ref})$
PI VELOCITY CONTROL
v_R
e
FPB
$D_{PI}(s)$
PWM
G(s)
v
S(s)

Fig. 2: Diagram of the driving capabilities.

In Fig. 2 we present the main components of the driving capabilities that we will implement to validate and compare the behaviour between the simulated and real system. The driving algorithms require different measurements about the state of the vehicle, most of which can be obtained from the front facing camera (i.e., orientation, track error and curvature) but the vehicle's speed is obtained by measuring rotation of the wheels on the rear axle (see Fig. 2).

The image process algorithm begins by obtaining an image from the front facing camera of the car. This image is then converted to grayscale in order to apply a threshold filter, which was calibrated to the lighting conditions of the indoor workspace. The result is a binary image where white pixels should represent the road that must be followed. We proceed to warp the image to obtain an aerial view (see *Image procces* block of Fig. 2). To the resulting frame, we apply the *stacking boxes* algorithm proposed by [16], allowing us to obtain the (x, y) coordinates of each of the road lines which are later used to compute the vehicle's orientation, track error and road curvature. The *stacking boxes* algorithm is also used in other lane centering projects like [1,7].

5 Design and Implementation

Here we will first present the characteristics of the real system, followed by the modeling process in the simulated environment.

5.1 Real System

In this section we present details of the construction of the scaled robot and track in the real world.

To test the lane centering system we must provide an appropriate physical model for the track. Our proposal, shown in Fig. 3, consists of a large straight lane segment and three types of curvatures: low curvature ($R = 1\,\text{m}$), medium curvature ($R = 0.5\,\text{m}$) and high irregular curvature. The track was created using

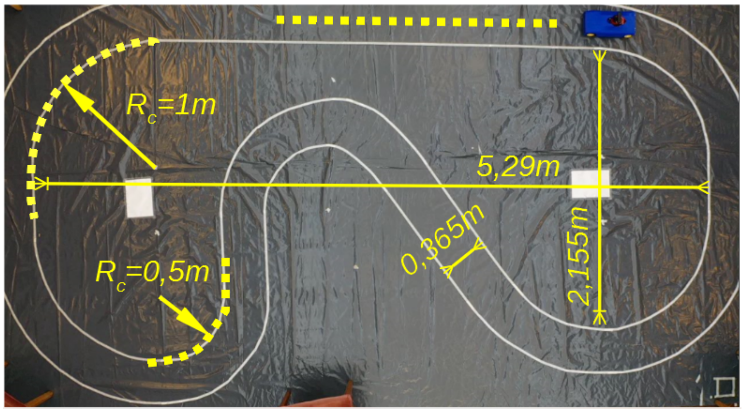
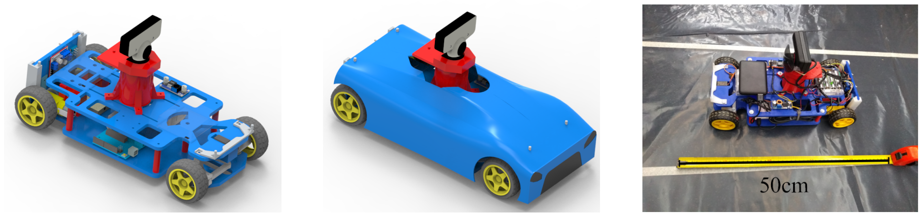


Fig. 3: Aerial view of the scaled track for the validation of our proposed system.



(a) (b) (c)

Fig. 4: (a) Two-level structure CAD model, (b) Casing CAD model, (c) Two-level structure of the real system with all the hardware components.

a black surface with white colored lines for the road. The lane width is 365 mm, representing a 1:10 scale of real highway lanes in Argentina [37]. The parameters of this track are similar to others used by 1:10 scale cars. For instance, the "CaroloCup" [4] is an autonomous scaled cars competition and they propose radios of curvature and lane width of 1.2 m and 400 mm, respectively.

The vehicle platform is shown in Fig. 4. A casing (see Fig. 4b) covers the two-level structure (Fig. 4a) that houses the electronic components mentioned in Section 3. Top level is reserved for rechargeable batteries and the camera while the bottom level contains the rest of electronic components and their connectors. The steering system was designed to directly translate the angle from an *MG90S* servomotor to the front wheels, with a 1:1 fixed rate. The longitudinal position of the camera stand on the top level and can be regulated in a range of ± 50 mm. The structure components in blue and red were 3D-printed using a PLA polymer, while the ones in white use a Nylon polymer because its lower friction coefficient ideal for pieces with relative movement. The constructed vehicle can be seen in Fig. 4c, with an estimated total weight and dimensions of: 1.7 kg – $355 \times 185 \times 200$mm. 3D CAD models can be accessed in [8].

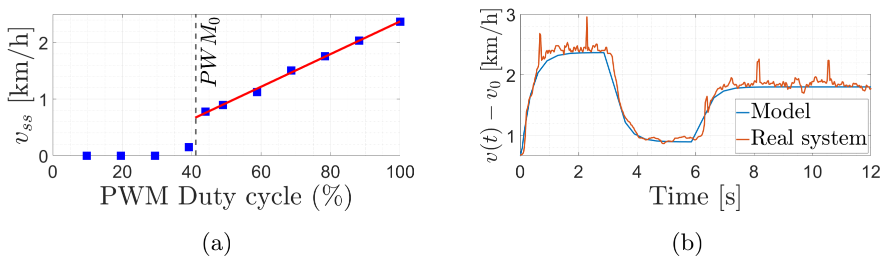


Fig. 5: (a) Stationary car velocity for different PWM inputs, where the red line shows the portion of the system's response with linear behaviour, (b) Comparison between the response of the real system and the first order model G.

In order to implement a velocity controller as described in Section 4 we first require modeling the dynamics of the vehicle. In Fig. 5a we show in blue points the measurements of stationary velocity for different PWM inputs on the driven wheels of the vehicle. On one hand, we can see in the left zone (i.e., low values of PWM) that the vehicle doesn't move. On the other hand, after a certain PWM_0 value, the stationary velocity begins to grow linearly with the PWM, meaning that in this region we have a linear behaviour of the system. Thus, we propose a first order linear model for it: $G(s) = \frac{K_{ss}}{\tau s+1}$, where K_{ss} corresponds to the slope of the linear zone of the system's response and τ depends on the rise time. From experiments we obtained the following parameters: $G(s) = \frac{2{,}89}{1+0.35s}$. In Fig. 5b we see that it correctly models the behaviour of the system in its linear zone.

Using this model, we designed a proportional integral (PI) velocity controller as in [10,15], where the reference velocity was set according to the road's curvature. We adjusted the roadline with a second order polynomial function and used the quadratic coefficient to estimate curvature, defining three zones and safe velocities: $2.3\,\mathrm{km\,h^{-1}}$ for a straight road, $2.0\,\mathrm{km\,h^{-1}}$ for a smooth curve and $1.0\,\mathrm{km\,h^{-1}}$ for a strong curve. The car speed can be measured from the time between two pulses of an encoder mounted over a grooved wheel connected to the rear axle, and knowing the amount of marks in the wheel as well as the car wheel diameter. A low-pass filter was used to reduce the measurement's noise and the integral term of controller was saturated to prevent windup.

Regarding the camera, we found that incorporating a polarizing lens in front of the camera could significantly improve the captured images. Since our proposed image processing algorithm obtains the roadlines by applying a static threshold filter, it is unable to distinguish white lines from the light reflections and other noise on the track's surface. Fig. 6 shows how for an input image with and without polarizing lens, the output of the roadline identification algorithm differs significantly.

Finally, since for the Stanley algorithm we must only tune the K parameter, we determined it experimentally. Our criteria was that the convergence to the middle of the road should be as fast as possible (high values of K) without producing undesired oscillations and magnification of measurement noises (see [36]

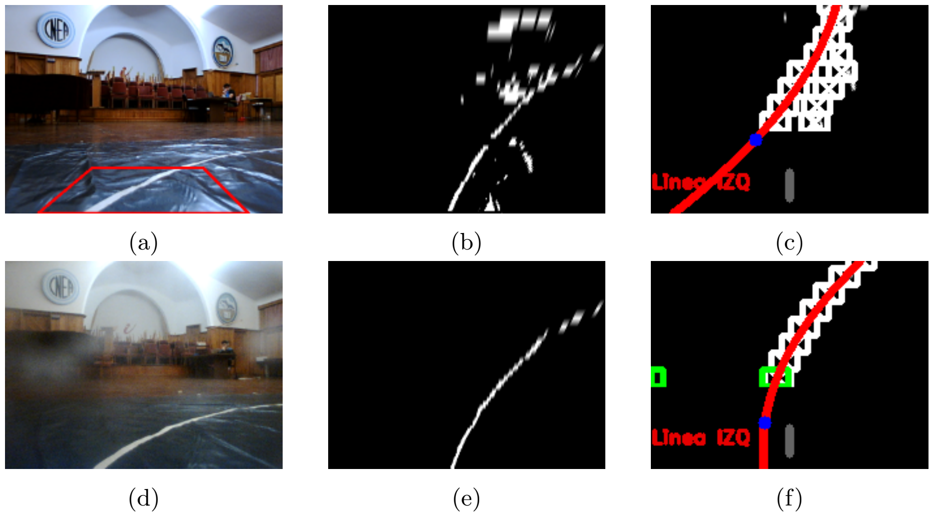


Fig. 6: (a) Input image without polarising lens, (b) Filtered and warped image of (a), (c) Processed image of (b). The sequence (d), (e) and (f) is similar but for the image with the polarising lens.

for a more in-depth analysis of the stability of the Stanley algorithm). We arrived at a value of $K = 1.0$.

5.2 Simulated System

In this section we show how we recreated the real vehicle and experimental track in the simulated environment.

In Fig. 7 we show the tree scheme which indicates how components in CoppeliaSim will interact with each other. The root of the structure is the car chassis (1) containing the car frame and the camera stand. Employing the 3D CAD model created on *Solid Edge* we measured moments of inertia, weight, and dimensions of the car and included them in the simulation as physical parameters. The rest of the components are connected with the chassis object. Right (3) and left (9) rear axles consist of a motorized joint and an attached wheel. However, front axle must provide the steering mechanism so its components are different. Looking at the right front axle, it consists of a motorized joint for steering (5) with position control loop enabled, a cube shaped rigid body as the hub (6) which connects the steering axle with wheel turning joint (7) (i.e., motor disabled) and finally a wheel (8).

The digitization of the track was made by taking an aerial view capture of it, as shown in Fig. 3. We then proceeded to binarise this image to obtain clear white roadlines over a black background, to better represent the kind of images that are obtained from the front facing camera of the real vehicle. In CoppeliaSim

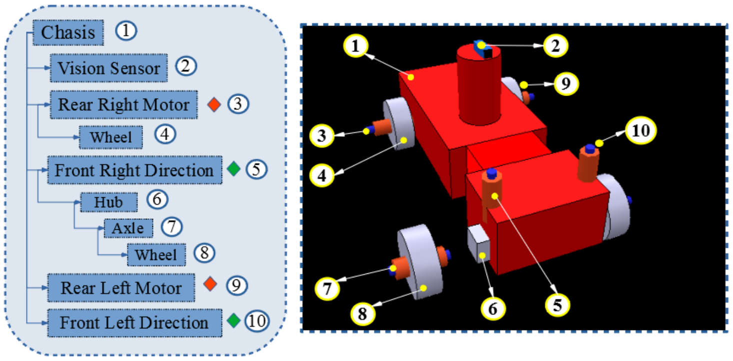


Fig. 7: Components employed in CoppeliSim to recreate the real vehicle hardware. Left and right components are the same for both rear and front axles.

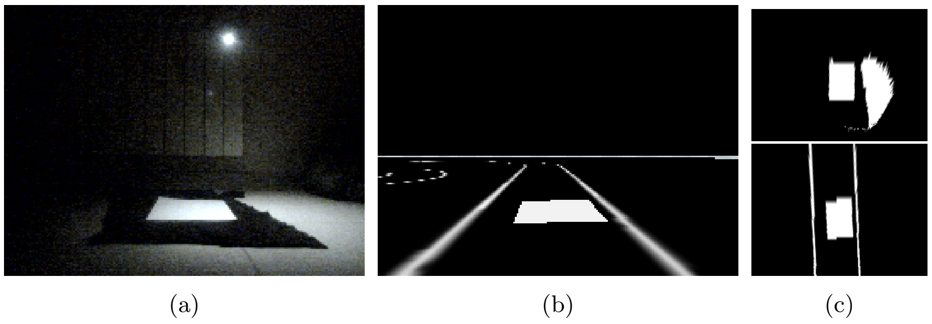
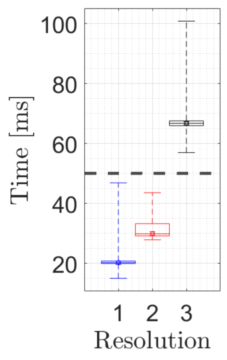


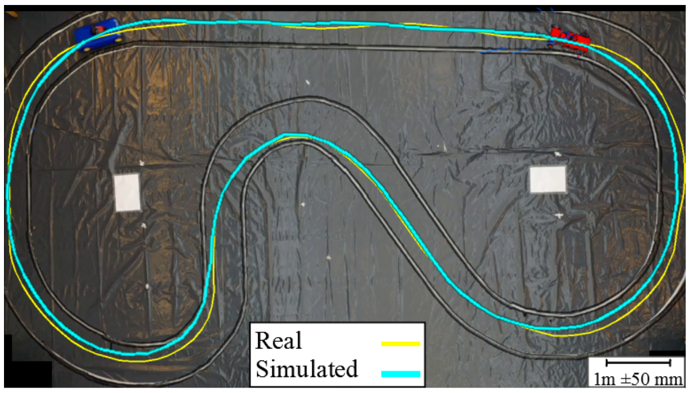
Fig. 8: Camera view of a same-dimensions object at fixed location in front the real (a) and simulated (b) car. In (c) we see the warped and filtered aerial view of (a) in the top image and of (b) in the bottom one.

we used a plane object to add the digitalised track as a texture. We then scaled the texture object to the correct dimensions of the real track.

There are two aspects that we had to focus on to reach the desired level of realism between the simulated and real environments. The first aspect is centered around the camera (or *vision sensor* in the simulator). Since the velocity and lane-centering algorithms greatly depend on what is sensed by the camera, one must ensure that the captured images between the simulator and the real vehicle are as similar as possible. For this, we put a white rectangular object of known dimensions in front of the real and simulated vehicle at a fixed distance. We then proceeded to adjust the Field of View (FOV) and orientation of the simulated camera until the warped aerial view of the captured images were as similar as possible, as shown in Fig. 8.

The second aspect was in regards the velocity control algorithm. Despite the fact that CoppeliaSim allows tuning the engine properties of its simulated motors, we found it easier to implement with custom LUA scripts the first-





(a) (b)

Fig. 9: (a) Boxplot of processing times for 3 different captured image resolutions: 213×160 (blue), 320×240 (red) and 640×480 (black). (b) Aerial view of the superposed real and simulated vehicle trajectories.

order model we obtained from the real system. Similarly, instead of modeling the encoders, we wrote a simple function that measures the rotating speed of the wheels in the simulator and calculates de corresponding time between pulses that the real encoder would have produced. This way, we can directly connect the velocity control algorithm's input and output with the simulator.

6 Validation

In this section we aim to validate the two key aspects of our proposed system: (a) the real and simulated platforms are adequate for the driving capabilities we described in Section 4, (b) the simulated platform produces very similar results when compared with the real system.

For the first aspect, the main concern is that the computational resources available in a Raspberry Pi 3B+ may not be enough for the driving algorithms we proposed, considering that these algorithms require processing images from the front facing camera. To test this potential problem we conducted the following experiment: since the image processing algorithms complexity depend on the captured image resolution, we evaluated the processing time required by the entire sensing and control processes for different input image resolutions. We show the results in Fig. 9a with a box plot of the processing time (T_S) for a sample of 1899 images captured by the real vehicle at one same position on the track with three different resolutions: 640×480, 320×240 and 213×160. Here we can see that for resolution sizes of 320×240 and 213×160 we can set a sample time of 50 ms for the sensing and control loop, similar to the sample size in other works [3]. For the following experiments we set the captured image resolution to 213×160 for both the real and simulated systems.

ASAI, Simposio Argentino de Inteligencia Artificial

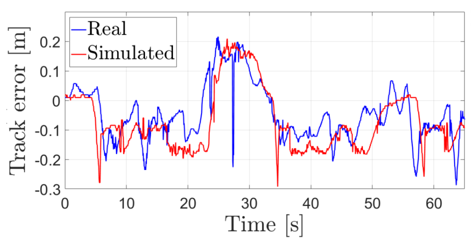


Fig. 10: Road tracking error.

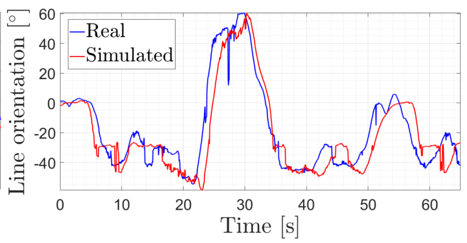


Fig. 11: Roadline orientation.

For the second aspect, we proceeded to do test runs for the real and simulated car platform using similar initial conditions, using a software-in-the-loop approach on the same Raspberry Pi 3B+ hardware as shown in Fig. 1. To ensure that the simulation process does not limit the available computational resources on the Raspberry Pi, we run the simulations on a separate PC with a Intel Core i7-8550U CPU of 1.80 GHz and 12 GB of RAM, communicating via UDP and TCP/IP with the Raspberry Pi.

To compare the trajectories of both systems, we took an aerial view video of the respective runs: in the simulation by capturing the screen on the PC running CoppeliaSim, and in the real system by using a camera-equipped DJI Mavic Pro 2 drone. The videos from these runs can be found in [8]. From these videos we extracted the vehicle's position throughout the run via an image processing algorithm that stabilizes the drone's captured image, later identifies the vehicle's position through a red mask filter, isolating the camera stand of the real vehicle (red coloured) and the car chassis in the simulator, and then calculates the average position of these red pixels. To be able to present both trajectories in the same graph, we plotted each of these trajectories in a sample frame from their respective video, and later resized, rotated and translated the resulting images until the roadlines from the track were superposed.

In Fig.9b we can see the reconstructed trajectories of the real and simulated vehicles as they complete a full turn around the track. As the graph shows, the driving capabilities allow for the real and simulated vehicle to correctly follow the track, with very similar trajectories. However, we can see some discrepancies between them, where the real vehicle follows the track closer to the outer roadline than the simulated vehicle. These differences may be originated from a variety of different sources: reflections in the real track causing roadlines to be identified differently to the simulator, delay in the real system from the moment the image is captured until it is handled to the image processing module leading to a delayed response in the real system, among others.

We present the measured track error in Fig. 10 and roadline orientation in Fig. 11 of the real and simulated runs, where we can observe both the correct behaviour of the driving algorithms and the similarity between the runs. In Fig. 10 we can see that the track error is always inside the range ± 0.3 m thanks to the implemented lane centering algorithm. Fig. 11 shows how the simulated measured line orientation matches closely to the real counterpart. Finally, from the video and these two graphs we see that simulated system completes the lap around 8% slower than the real one, which could again be explained by the several sources of modelation error.

Given the simplicity of our simulated environment we believe that the similarity between plotted trajectories and graphs is enough to use the simulated platform as a realistic testing and development platform.

7 Conclusions

In this paper we presented a low-cost 3D-printed 1:10 scale car that integrates with a realistic, yet simple, simulator allowing for software-in-the-loop simulations for development and testing on the same processing hardware as the real system. This platform was designed for researchers and educators beginning to work in the area of autonomous driving. For this reason, we implemented on this system near SAE level 2 driving capabilities and demonstrate that it is suitable to develop and test advanced driving algorithms on real and simulated tracks.

In future work, it would be interesting to analyse the possibility of incorporating other sensors common in the autonomous driving literature (e.g., a LiDAR sensor), to be able to implement full adaptive cruise controllers and obstacle avoidance algorithms.

References

1. Aditham, H., Nadar, R.: Real time lane detection using ross kippenbrock method and object detection using yolo. In: IEJRD - International Multidisciplinary Journal. vol. 5, p. 8 (2020)
2. Agnihotri, A., Saraf, P., Bapnad, K.R.: A convolutional neural network approach towards self-driving cars. In: 2019 IEEE 16th India Council International Conference (INDICON). pp. 1–4 (2019)
3. Bechtel, M.G., Mcellhiney, E., Kim, M., Yun, H.: Deeppicar: A low-cost deep neural network-based autonomous car. In: 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). pp. 11–21 (2018)
4. Berger, C.: From a competition for self-driving miniature cars to a standardized experimental platform: Concept, models, architecture, and evaluation. Journal of Software Engineering for Robotics **5**(1), 63–79 (2014)
5. Berger, C., Chaudron, M., Heldal, R., Landsiedel, O., Schiller, E.M.: Model-based, composable simulation for the development of autonomous miniature vehicles. In: Proceedings of the Symposium on Theory of Modeling Simulation - DEVS Integrative MS Symposium. pp. 1–8 (2013)
6. Blaga, B.C.Z., Deac, M.A., Al-doori, R.W.Y., Negru, M., Dănescu, R.: Miniature autonomous vehicle development on raspberry pi. In: 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP). pp. 229–236 (2018)

7. Bégin, W., Duquette, S., Lavallière, M.: Real application for a low-cost low-power self-driving 1/10 scale car. Journal of Robotics and Automation **4**(1), 195–201 (2020)
8. C. Allione: Opensource platform. `https://bitbucket.org/CristianAllione/opensource_plattform`, [Online; accessed 21-August-2021]
9. Cho, M.g.: A study on the obstacle recognition for autonomous driving rc car using lidar and thermal infrared camera. In: 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN). pp. 544–546 (2019)
10. Cumali, K., Armagan, E.: Steering control of a vehicle equipped with automated lane centering system. In: 2019 11th International Conference on Electrical and Electronics Engineering (ELECO). pp. 820–824 (2019)
11. Curiel-Ramirez, L.A., Ramirez-Mendoza, R.A., Bautista-Montesano, R., Bustamante-Bello, M.R., Gonzalez-Hernandez, H.G., Reyes-Avedaño, J.A., Gallardo-Medina, E.C.: End-to-end automated guided modular vehicle. Applied Sciences **10**(12) (2020)
12. Fusic, S.J., Karlmarx, M., Leando, I., Hariharan, K.: Path planning for car like mobile robot using robot operating system. In: 2018 National Power Engineering Conference (NPEC). pp. 1–5 (2018)
13. Garcia, J., Feng, Y., Shen, J., Almanee, S., Xia, Y., Chen, Q.A.: A comprehensive study of autonomous vehicle bugs. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). pp. 385–396 (2020)
14. Gotlib, A., Łukojć, K., Szczygielski, M.: Localization-based software architecture for 1:10 scale autonomous car. In: 2019 International Interdisciplinary PhD Workshop (IIPhDW). pp. 7–11 (2019)
15. Hoffmann, G.M., Tomlin, C.J., Montemerlo, M., Thrun, S.: Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In: 2007 American Control Conference. pp. 2296–2301 (2007)
16. Keenan, R., Kippenbrok, R., Brucholtz, B., Reichelt, M.: `https://github.com/rkipp1210/advanced-lane-lines` (2017), [Online; accessed 25-May-2021]
17. Kim, C.i., Kim, M.s., Lee, K.s., Jang, H.S., Park, T.S.: Development of a full speed range path-following system for the autonomous vehicle. In: 2015 15th International Conference on Control, Automation and Systems (ICCAS). pp. 710–715 (2015)
18. Kim, J.K., Kim, J.W., Kim, J.H., Jung, T.H., Park, Y.J., Ko, Y.H., Jung, S.: Experimental studies of autonomous driving of a vehicle on the road using lidar and dgps. In: 2015 15th International Conference on Control, Automation and Systems (ICCAS). pp. 1366–1369 (2015)
19. Knoop, V.L., Wang, M., Wilmink, I., Hoedemaeker, D.M., Maaskant, M., der Meer, E.J.V.: Platoon of sae level-2 automated vehicles on public roads: Setup, traffic interactions, and stability. Transportation Research Record **2673**(9), 311–322 (2019)
20. Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., Thrun, S.: Towards fully autonomous driving: Systems and algorithms. In: 2011 IEEE Intelligent Vehicles Symposium (IV). pp. 163–168 (2011)
21. Molina, C.B.S.T., Almeida, J.R.d., Vismari, L.F., González, R.I.R., Naufal, J.K., Camargo, J.: Assuring fully autonomous vehicles safety by design: The autonomous vehicle control (avc) module strategy. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). pp. 16–21 (2017)

22. Müller, M., Koltun, V.: Openbot: Turning smartphones into robots. In: 2021 IEEE International Conference on Robotics and Automation (ICRA) (2021), in press.
23. Nakamoto, N., Kobayashi, H.: Development of an open-source educational and research platform for autonomous cars. In: IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society. vol. 1, pp. 6871–6876 (2019)
24. Nguyen, T., Yoo, M.: Fusing lidar sensor and rgb camera for object detection in autonomous vehicle with fuzzy logic approach. In: 2021 International Conference on Information Networking (ICOIN). pp. 788–791 (2021)
25. Orsolic, I.: Building a Self-Driving RC Car. Master's thesis, University of Zagreb, Croatia (2019)
26. Pahlavan, M., Papatriantafilou, M., Schiller, E.M.: Gulliver: A test-bed for developing, demonstrating and prototyping vehicular systems. In: 2012 IEEE 75th Vehicular Technology Conference (VTC Spring). pp. 1–2 (2012)
27. Percinski, M., Marcinkiewicz, M.: Architecture of the system of 1:10 scale autonomous car — requirements-based design and implementation. In: 2018 International Interdisciplinary PhD Workshop (IIPhDW). pp. 263–268 (2018)
28. Prabhu, N., Min, S., Nam, H., Tewolde, G., Kwon, J.: Integrated framework of autonomous vehicle with traffic sign recognition in simulation environment. In: 2020 IEEE International Conference on Electro Information Technology (EIT). pp. 514–521 (2020)
29. Raffone, E., Rei, C., Rossi, M.: Optimal look-ahead vehicle lane centering control design and application for mid-high speed and curved roads. In: 2019 18th European Control Conference (ECC). pp. 2024–2029 (2019)
30. Rohmer, E., Singh, S.P.N., Freese, M.: V-rep: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1321–1326 (2013)
31. SAE On-Road Automated Driving (Orad) Committee: Sae j3016-taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. SAE-Society of Automotive Engineers (2018)
32. Snider, J.: Automatic steering methods for autonomous automobile path tracking. Tech. rep., Robotics Institute, Carnegie Mellon University (2009)
33. Sumardi, Taufiqurrahman, M., Riyadi, M.: Street mark detection using raspberry pi for self-driving system. Telkomnika (Telecommunication Computing Electronics and Control) **16**, 629–634 (04 2018)
34. Sun, Z., Huang, L., Jia, R.: Coal and gangue separating robot system based on computer vision. Sensors **21**, 1349 (02 2021)
35. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Stanley: The robot that won the darpa grand challenge. Journal of Field Robotics **23**(9), 661–692 (2006)
36. Törő, O., Bécsi, T., Aradi, S.: Design of lane keeping algorithm of autonomous vehicle. Periodica Polytechnica Transportation Engineering **44**(1), 60–68 (2016)
37. Vialidad Nacional Argentina: Memoria descriptiva - corredor vial nacional c. `https://www.argentina.gob.ar/sites/default/files/sppp-cvc-anexo-ii.pdf`, [Online; accessed 25-May-2021]