A Low-Cost, Open Platform for End-to-End Autonomous Driving on a Miniature Ackermann Vehicle

Gustavo Claudio Karl Couto[1], **Eric Aislan Antonelo**[1], **Gabriel George Zipperer**[1]

[1]Department of Automation and Systems Engineering
Federal University of Santa Catarina (UFSC) – Florianópolis – SC – Brazil

gustavo.karl.couto@posgrad.ufsc.br, eric.antonelo@ufsc.br,
gabriel.zipperer@grad.ufsc.br

Abstract. *This paper presents a low-cost, open experimental platform for research in end-to-end autonomous driving with miniature Ackermann vehicles. The platform combines a physical vehicle, a printed urban track, data collection tools, trajectory registration, and a Webots digital twin, enabling controlled experiments that connect simulation-based autonomous-driving methods to real-world execution. As a first baseline, we implement command-conditioned behavior cloning, in which a neural policy receives an on-board camera image and a high-level navigation command and outputs steering and speed. The system is evaluated both on the physical vehicle and in simulation. In real closed-loop experiments, the learned policy follows lanes and executes commanded turns, reaching a mean cross-track error of* $6.1\,\mathrm{cm}$ *with respect to the reference route, close to the* $4.7\,\mathrm{cm}$ *observed in human demonstrations. In the digital twin, camera field of view has a strong effect on performance, reducing the mean cross-track error from* 35.6 *to* $3.3\,\mathrm{cm}$ *when widened from* 58° *to* 120°*. Using the digital twin to generate synthetic driving data and a learned sim-to-real image translator to reduce the appearance gap, we further show that a higher-capacity policy trained on this synthetic data combined with real demonstrations is the only configuration that completes all four track routes in closed loop, whereas the compact baseline and the same network trained on real data alone complete fewer. These results establish the open platform as a practical testbed for sim-to-real studies and provide an initial command-conditioned imitation-learning baseline; we release it to support reproducible research.*

1. Introduction

Autonomous driving systems have traditionally been organized as modular pipelines, in which perception, localization, planning, and control are designed as separate components [Paden et al. 2016]. In parallel, learning-based approaches have explored end-to-end policies that map sensor observations directly to driving actions, reducing the amount of hand-crafted intermediate structure required by the system [Pomerleau 1991]. Among these methods, imitation learning is particularly attractive because it can train driving policies from demonstrations collected by a human or privileged controller.

Large-scale simulators have played an important role in this research direction. Environments such as CARLA provide controllable urban scenarios, repeatable evaluation protocols, and access to privileged information that is difficult to obtain in the real world [Dosovitskiy et al. 2017]. They have enabled the development and comparison of end-to-end driving architectures, including policies conditioned on high-level

navigation commands [Codevilla et al. 2018], methods that learn from privileged teachers [Chen et al. 2019], and approaches based on intermediate scene representations. However, simulation alone does not expose several practical constraints present in physical systems, such as camera latency, limited field of view, actuator imperfections, calibration errors, odometry drift, and imperfect synchronization between sensing and control.

Low-cost model-scale vehicles offer a useful intermediate step between simulation and full-size autonomous cars. They make it possible to evaluate autonomous-driving methods on physical hardware while preserving safety, repeatability, and accessibility [Bechtel et al. 2018, O'Kelly et al. 2020, Hyldmar et al. 2019]. Such platforms are especially valuable when they are paired with a digital twin, since the same track, vehicle geometry, and evaluation metrics can be used in both simulated and real experiments. This creates a practical setting for sim-to-real studies and for re-evaluating architectures first developed in high-fidelity simulators under real sensing and control constraints.

This paper presents a low-cost, open experimental platform for end-to-end autonomous driving with a miniature Ackermann vehicle. The platform includes a physical vehicle, an urban printed track, a data-collection procedure, a map-registration tool for quantitative trajectory evaluation, and a Webots digital twin. The goal is not only to demonstrate autonomous lane following on a small vehicle, but also to provide a controlled experimental setup in which simulation-based driving methods can be transferred, adapted, and compared in the real world.

As an initial learning baseline, we consider command-conditioned behavior cloning. A purely reactive image-to-action policy is ambiguous at intersections: the same visual observation may be compatible with turning left, turning right, or continuing straight. Conditional imitation learning addresses this ambiguity by providing a high-level command to the policy [Codevilla et al. 2018]. In our setting, the policy receives an on-board camera image and a discrete command, and outputs steering and speed for the Ackermann vehicle. This baseline is intentionally simple, serving as a reference point for future experiments with more advanced architectures and learning objectives.

The main contributions of this work are:

- a low-cost, open experimental platform for autonomous-driving research with a physical Ackermann mini-vehicle, an urban printed track, data-collection tools, trajectory registration, and a Webots digital twin, released to support reproducible sim-to-real research;
- an experimental setup that enables sim-to-real studies and the re-evaluation, on physical hardware, of end-to-end driving architectures originally developed in simulation;
- a command-conditioned behavior-cloning baseline for lane following and commanded turning from an on-board camera image;
- a sim-to-real data pipeline that records synthetic driving data in the digital twin and maps it to the camera domain with a learned image translator, together with evidence that this synthetic augmentation is what lets a higher-capacity policy complete every track route;
- an initial set of ablations comparing design choices such as camera field of view, command conditioning, network capacity, and the use of sim-to-real-translated synthetic data.

Experimental results show that the platform supports closed-loop autonomous driving on the physical track and provides quantitative route-following metrics through map registration. The command-conditioned baseline follows lanes and executes operator-issued turns, reaching a mean cross-track error close to that of the human demonstrations. Comparing policies, a higher-capacity network trained with sim-to-real-translated synthetic data added to the real demonstrations completes all four routes in closed loop, while the compact baseline and the real-data-only network complete fewer. In the digital twin, the camera field of view is shown to be a critical factor for route-following performance.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 describes the command-conditioned behavior-cloning baseline. Section 4 presents the policy architectures. Section 5 describes the physical platform, digital twin, dataset, and experiments. Section 6 concludes the paper and discusses future work.

2. Related Work

Several works exploit simulation to train and evaluate end-to-end policies for urban driving. Conditional imitation learning introduced high-level navigation commands to disambiguate visuomotor policies at intersections [Codevilla et al. 2018], while later studies analyzed the limitations of behavior cloning under distribution shift and dataset bias [Codevilla et al. 2019]. Other approaches use intermediate or privileged information to improve learning, for example privileged-teacher training [Chen et al. 2019]. Related studies have also explored implicit behavior cloning for multimodal action distributions [Florence et al. 2022] and diffusion-based offline behavior cloning. These works motivate physical testbeds where architectures first studied in simulation can be re-evaluated under real sensing, latency, calibration, and actuation constraints.

Low-cost model-scale vehicles provide an intermediate step between simulation-only research and full-size autonomous cars. DeepPicar demonstrated that a small vehicle with an embedded computer can run a CNN-based end-to-end controller in real time [Bechtel et al. 2018]. F1TENTH established a widely used 1/10-scale autonomous racing platform with both hardware and virtual environments for safe and repeatable experimentation [O'Kelly et al. 2020]. Other miniature-car platforms have also been proposed for cooperative-driving and multi-vehicle experiments [Hyldmar et al. 2019]. Compared with these platforms, the present work focuses on command-conditioned urban navigation on a printed road network, with a low-cost Ackermann vehicle, a data-collection pipeline, map-referenced trajectory evaluation, and a Webots digital twin. The goal is not to replace larger platforms, but to provide an accessible setup for studying sim-to-real transfer and for testing end-to-end driving architectures previously evaluated mainly in simulation.

3. Command-Conditioned Behavior Cloning Baseline

Given an RGB image I_i from the on-board camera and a high-level command c_i, the policy $\pi_\theta(I_i, c_i)$ predicts a continuous steering command $\hat{\delta}_i$ and a signed speed command $\hat{v}_i$. The expert label for each frame is $a_i = (\delta_i, v_i)$, collected during teleoperation. Both outputs are bounded to $[-1, 1]$ by a $\tanh$ activation and then denormalized to the physical actuator limits.

The policy is trained by weighted mean squared error,

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big[w_\delta \big(\delta_i - \hat{\delta}_i\big)^2 + w_v \big(v_i - \hat{v}_i\big)^2 \Big], \tag{1}$$

where $w_\delta = 1$ and $w_v \leq 0.25$ weight the steering and speed terms; speed is down-weighted because route-following accuracy is dominated by steering and the closed-loop speed is capped. We use deterministic bounded regression rather than a probabilistic action model to keep the baseline simple.

3.1. Command conditioning

A purely reactive policy cannot choose among the valid maneuvers at an intersection [Codevilla et al. 2018]. We therefore expose the intended maneuver as a discrete command c and condition the policy on it, $\pi_\theta(I, c)$. The command takes one of three values, `follow_lane`, `turn_left`, and `turn_right`, encoded as a one-hot vector. During data collection, the command for each frame is derived from labelled intersection spans: a turn command is active only while the vehicle is inside the labelled intersection interval, and the command is `follow_lane` everywhere else. At test time, the operator selects it from a direction pad, holding the turn command across the intersection, so the same visual policy can be steered through different routes.

4. Policy Architectures

4.1. Input and output representation

The policy input is an RGB image from the vehicle's on-board camera, resized to 160×120 pixels, together with the one-hot navigation command. Frames are kept in RGB channel order and scaled to $[0, 1]$, with no lens undistortion applied; the higher-capacity variant instead takes 320×240 frames with the top 35% cropped to remove the region above the road. The action space is $a \in [-1, 1]^2$: steering is denormalized to $\pm 45^\circ$ and speed to the configured maximum (up to $0.7\,\mathrm{m/s}$, capped at $0.15\,\mathrm{m/s}$ in the physical experiments for safety).

4.2. Compact CNN baseline

The main baseline policy (Fig. 1a) is a compact CNN that maps the input image into an embedding and combines it with the command to produce the two actions. The 160×120 image passes through four strided convolutional blocks (16, 32, 64, 96 channels; 5×5 then 3×3 kernels; batch normalization and ReLU) followed by global average pooling, producing an embedding vector of size 96. This embedding is concatenated with the one-hot command and processed by a two-layer MLP that maps to the desired actions through a $\tanh$ output. The whole network has $94{,}882$ parameters.

4.3. Architecture variants for ablation

For ablation we compare it with a larger command-conditioned variant (Fig. 1b, about 1.0M parameters) that keeps the same input-output interface and task; besides isolating the effect of capacity, the larger variant lets us compare training on real data alone against the mixed synthetic-plus-real data (Section 5).

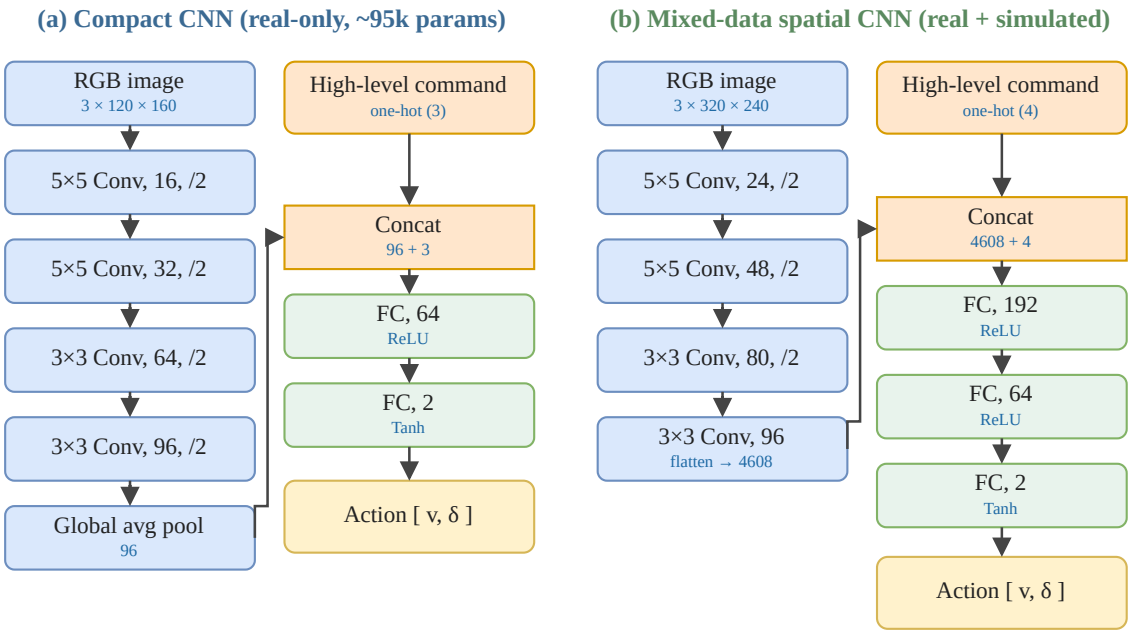


Figure 1. Policy architectures. (a) The compact command-conditioned CNN baseline (94,882 parameters): a convolutional encoder with global average pooling whose embedding is concatenated with the one-hot command and decoded by a two-layer MLP. (b) The larger command-conditioned variant: a spatial encoder whose flattened features are concatenated with the command and decoded by a three-layer MLP.

5. Experiments

5.1. Vehicle platform and track

The vehicle is an off-the-shelf Ackermann chassis (Yahboom ROSMASTER R2L class) with two rear drive motors, a servo-steered front axle, and a forward camera on a short mast (Fig. 2). Its wheelbase is $0.2354\,\mathrm{m}$, its track width $0.1911\,\mathrm{m}$, and steering is limited to $\pm 45°$. An ESP32-S3 runs an ESP-IDF firmware with a micro-ROS client [Belsare et al. 2023, Macenski et al. 2022]. It closes a per-wheel incremental-PID speed loop at $100\,\mathrm{Hz}$ (gains $0.75/0.12/0.04$, 836-count/rev encoders), with left/right setpoints derived from the commanded speed and steering angle by the Ackermann geometry (an electronic differential), drives the steering servo, and enforces a battery cut-off and a $500\,\mathrm{ms}$ command watchdog. The firmware publishes a raw odometry twist and the IMU at $20\,\mathrm{Hz}$, which a host-side extended Kalman filter [Moore and Stouch 2016] fuses into the pose used for trajectory plots. A separate ESP32 camera streams QVGA images over a low-latency UDP link whose measured transport delay has a median of $27\,\mathrm{ms}$ and a 95th percentile of $31\,\mathrm{ms}$; the end-to-end capture-to-actuation latency (including $2.5\,\mathrm{ms}$ of policy inference) has a median of $59.5\,\mathrm{ms}$ and a 95th percentile of $118.7\,\mathrm{ms}$, with the policy running at $10\,\mathrm{Hz}$. The camera is calibrated at QVGA from a ChArUco board's chessboard corners [Zhang 2000, Garrido-Jurado et al. 2014].[1]

[1]Firmware, host software (operator node, policy inference, camera bridge, EKF and calibration files) and the deployed checkpoint are released under the MIT license at `https://github.com/gustavokcouto/teledriving-in-miniature`.

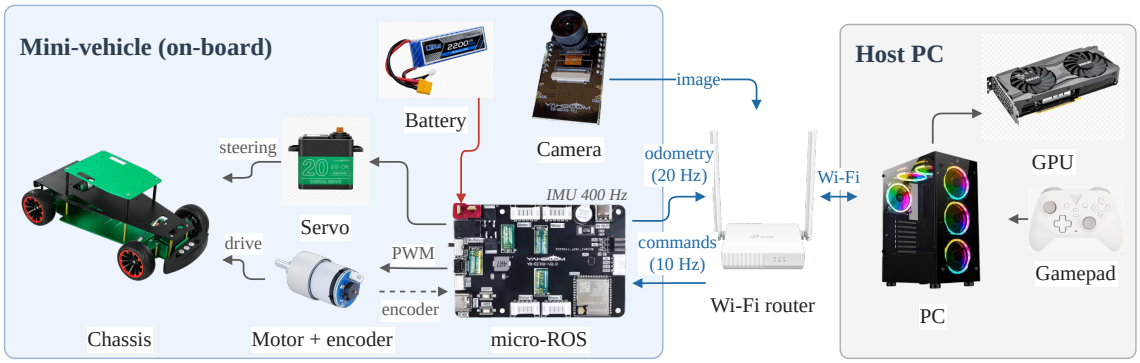


Figure 2. Experimental setup, split into the on-board mini-vehicle and the host PC: the ESP32-S3 micro-ROS board closes the 100 Hz speed PID loop and drives the servo, while a separate ESP32 camera streams images to the host PC, which runs the policy and the extended Kalman filter and returns steering and speed commands.

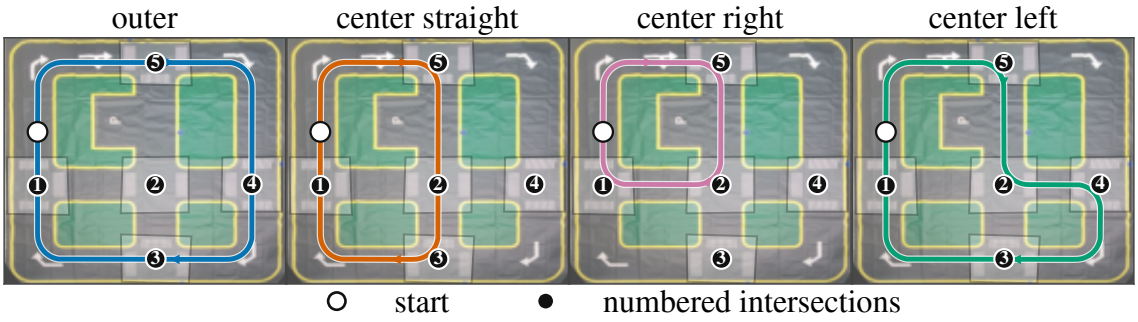


Figure 3. The four ideal routes on the rectified mat used as the Webots floor texture. Each panel shows one route, driven in a single direction from the shared start marker (white circle). Numbered markers indicate the labelled intersections used for command annotation.

The experiments are run on a printed road mat of $3.2 \times 2.8\,\mathrm{m}$. All streets are driven in a single direction, which yields four selectable loops (`outer`, `center_left`, `center_right`, and `center_straight`) that also define the high-level commands (Fig. 3). To obtain a metric reference frame, the mat is photographed and perspective-rectified into a flat top-down texture. Road intersections are marked manually with a companion interface in which the operator draws, for each intersection, the four straight road edges that bound it (shown demarcated in Fig. 3); the region they enclose defines the intersection span later used to assign the high-level command. The rectified map is reused as the Webots floor texture and as the common reference frame for labelled intersections, ideal routes, and cross-track error.

5.2. Dataset generation

The expert dataset is generated by a human driver who teleoperates the vehicle with a gamepad while holding a deadman trigger that gates the recording; Fig. 4 summarizes the resulting end-to-end signal flow, from this manual collection to autonomous deployment. Each session is stored as a folder of 320×240 frames recorded at $10\,\mathrm{Hz}$ together with labels for steering, speed, fused odometry pose, IMU, encoder signals, and com-

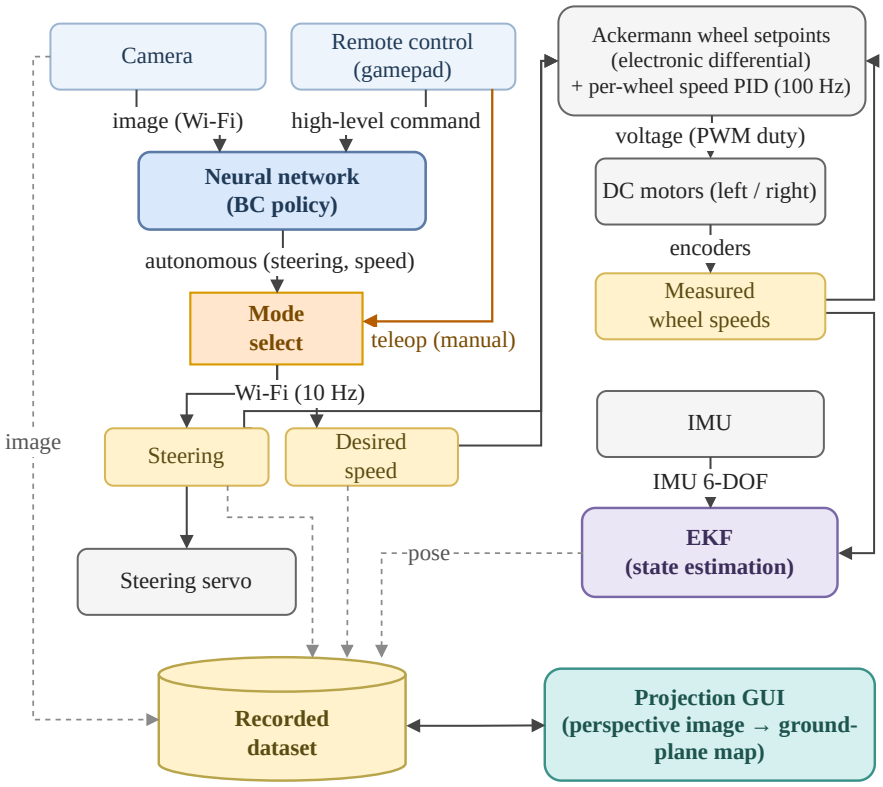


Figure 4. End-to-end signal flow. During data collection a human teleoperates the vehicle and the mode-select logs the commanded steering/speed—with the camera image and the EKF pose—into the recorded dataset; at deployment the behavior-cloning policy replaces the teleoperator, producing steering and a speed setpoint; on board, steering drives the servo and, with the speed setpoint, defines per-wheel setpoints (Ackermann electronic differential) for the 100 Hz PID loops, while the projection GUI maps recorded images to the ground plane.

mand gate. The expert dataset comprises four route families with 41 sessions and 19,206 frames, of which 18,467 are active driving frames (5,137 for `center_left`, 3,544 for `center_right`, 4,512 for `center_straight`, and 5,274 for `outer`). High-level commands are assigned from labelled intersection spans (75 right-turn, 10 left-turn, and 78 straight segments), remapping the recorded straight action to `follow_lane`.

5.3. Synthetic data and sim-to-real translation

The higher-capacity policy is additionally trained with synthetic data from a Webots digital twin of the same track. A scripted controller follows the ideal routes of the road network and records clean expert steering and speed labels at 10 Hz, while actuator-noise and off-route recovery perturbations enrich the visited states without altering those expert labels. This produces 96 laps (24 for each of the four routes) and 43,581 active frames.

To narrow the appearance gap between the simulator and the on-board camera, each synthetic frame is converted offline by a sim-to-real image translator before training; only the image is changed, while the expert labels are kept. The translator is a four-level U-Net [Ronneberger et al. 2015] trained with a paired L_1 objective on 2,559 registered simulator–real image pairs (from 16 real sessions), applying pho-

tometric domain randomization [Tobin et al. 2017] to the synthetic input. Among the methods we compared—this domain-randomized U-Net, pix2pix [Isola et al. 2017], and SimGAN [Shrivastava et al. 2017]—it produced the lowest closed-loop route error in the twin, reducing the mean route error of a real-trained policy from 28.0 to $13.1\,\mathrm{cm}$. The translated synthetic laps are then concatenated with the 41 real sessions, giving a mixed dataset of 137 sessions and $62{,}048$ active frames.

5.4. Training protocol

Policies are trained by behavior cloning with the weighted loss of Equation (1) on the active frames, using AdamW (learning rate 10^{-3}, weight decay 10^{-4}) and batch size 32, holding out whole sessions for validation to avoid leakage between adjacent frames. The compact baseline is trained on the real dataset, whereas the higher-capacity policy is trained either on the real dataset alone or on the mixed synthetic-plus-real dataset, with speed- and steering-loss weights of 0.25 and 1.0. For the mixed dataset, the policy is trained for 200 epochs and the checkpoint is selected by closed-loop rollout in the digital twin rather than by lowest validation loss, since the two criteria do not coincide.

5.5. Map registration and route-deviation metric

The dataset images are local camera observations, whereas the route is defined on the global map. To obtain a map-referenced accuracy metric, each run is registered to the rectified track with a calibrated tool that projects the camera image onto the ground plane using intrinsics and extrinsics (the planar homography of Fig. 5), and an operator aligns the projected trajectory to the map. This yields a globally consistent trajectory $p_i = (x_i, y_i)$ in metres (Fig. 6). The ideal route centre lines are built from the track loops and the marked intersections. For a registered point p_i and the ideal polyline Γ_r of route r, the cross-track error is

$$e_i = \min_{q \in \Gamma_r} \|p_i - q\|_2. \tag{2}$$

We report per-run mean, median, and 95th-percentile errors, scoring only fully localized runs—those with at least ten manual anchors spanning $\geq 85\%$ of their frames—namely 10 human sessions and the three neural-control runs.

5.6. Physical closed-loop evaluation

We evaluate three fully autonomous runs of the compact baseline (about $1{,}300$ frames each), in which the agent drives continuously while the operator only issues high-level commands and the speed is capped at $0.15\,\mathrm{m/s}$. Each run repeats the outer, center-straight, and center-right loops; the more demanding center-left (left-turn) loop is completed only by the larger mixed-data policy (Fig. 7). Steering stays centred under `follow_lane` and consistently negative under `turn_right` (median around $-22°$), with no operator interventions. Registering each run to the map (Eq. (2)) and matching every lap to its ideal loop, the policy drives with a $6.1 \pm 1.2\,\mathrm{cm}$ mean cross-track error, close to the $4.7 \pm 1.8\,\mathrm{cm}$ of the fully localized human demonstrations—a reference for the available demonstrations rather than a strict upper bound, since teleoperation itself introduces some tracking error.

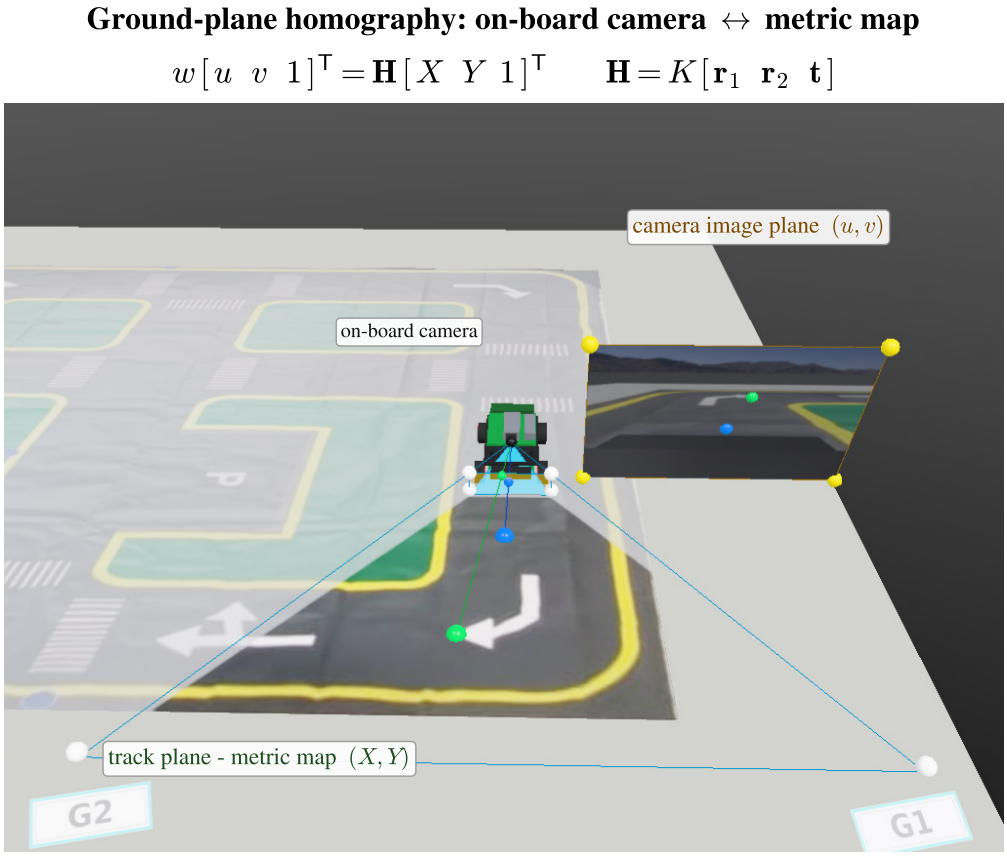


Figure 5. Ground-plane homography. Because the track is planar, camera intrinsics and pose define a single 3×3 homography between metric map coordinates (X, Y) and image pixels (u, v). The projection tool uses this mapping, or its inverse, to relate on-board camera observations to the rectified map; the figure illustrates the geometry in the Webots digital twin. The blue and green markers are the same physical points on the track and in the camera image.

5.7. Digital twin and camera field-of-view ablation

The physical metric conflates many factors. To isolate camera field of view with exact ground truth, we evaluate the same policy in the Webots [Michel 2004] digital twin (the same track and the vehicle's Ackermann geometry—wheelbase, track width, and wheel sizes) while varying the simulated lens. The calibrated narrow lens ($58°$) follows the lane only loosely, whereas widening it to $120°$ cuts the mean cross-track error from $35.6 \pm 12.1\,\mathrm{cm}$ (per-route 25–53 cm, with peaks above $1\,\mathrm{m}$) to $3.3 \pm 0.2\,\mathrm{cm}$ (per-route 3.1–3.5 cm), identifying field of view as a major limiting factor.

5.8. Architecture and command-conditioning ablations

Keeping the command input fixed, we compare the compact baseline (CNN-small) with the larger policy (CNN-large) under two training sets, real-only and mixed synthetic-plus-real. The clearest difference is how many of the four routes each policy completes in closed loop (Fig. 7). The compact baseline completes the outer, center-straight, and center-right loops but fails the center-left turn; the larger policy trained on real data alone completes only the outer and center-straight loops; and the larger policy trained on the

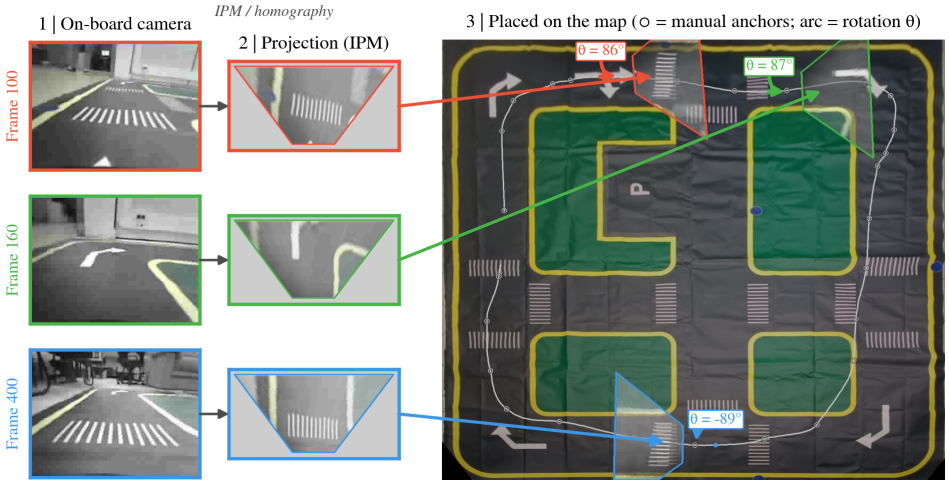


Figure 6. Map-registered localization of an outer-loop run: per-frame camera images are projected to the ground plane and the reconstructed trajectory is overlaid on the rectified map, illustrating the registration behind the cross-track-error metric.

mixed dataset is the only one to complete all four routes. On the routes it completes, the mixed-data policy stays close to the ideal path, with per-route mean cross-track errors of 3.9–6.1 cm (outer), 6.0–6.5 cm (center-straight), 5.1 cm (center-right) and 4.2 cm (center-left), for an overall mean of about 5.1 cm. The added capacity thus generalizes to every maneuver only when the sim-to-real-translated synthetic data broadens the training distribution.

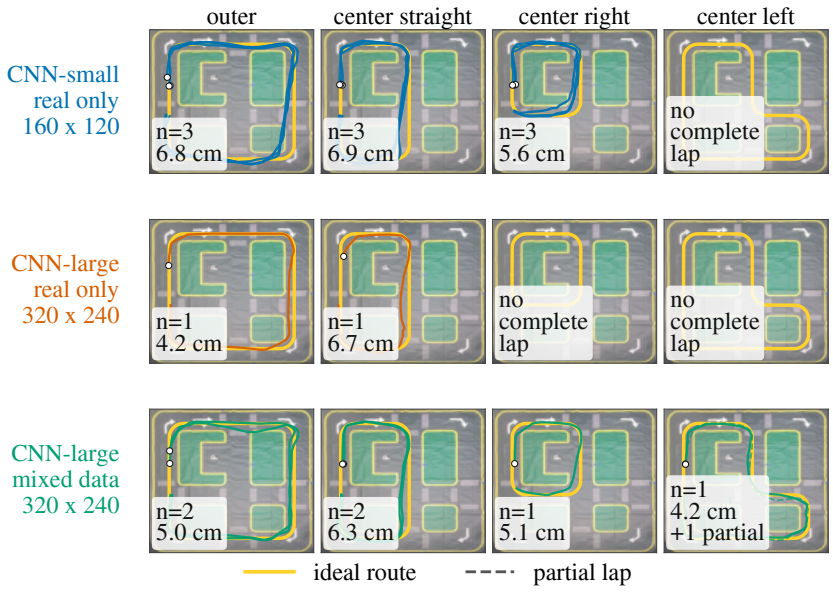


Figure 7. Comparison of map-registered neural-control laps by policy variant. Each row groups the autonomous runs from one policy; columns show the four route loops. Colored lines are registered real-robot laps, dashed lines are partial laps, yellow lines are ideal routes, and each panel reports the mean cross-track error over complete laps.

6. Conclusion

This work presented a low-cost, open sim-to-real platform for end-to-end autonomous driving with an Ackermann mini-vehicle, together with a command-conditioned behavior-cloning baseline. On the physical track, the compact policy follows lanes and executes operator-issued turns at a $6.1\,\mathrm{cm}$ mean cross-track error, close to the $4.7\,\mathrm{cm}$ of the human demonstrations; a higher-capacity network completes all four routes only when its real data is augmented with sim-to-real-translated synthetic laps; and in the digital twin, widening the camera from 58° to 120° cuts cross-track error from 35.6 to $3.3\,\mathrm{cm}$.

The main limitations are the small number of physical runs, the manual component of the trajectory registration, and the preliminary ablations, which lack a compact-network/mixed-data condition and confound capacity with input resolution. Future work includes automatic image-based registration, a wider-field-of-view camera, and stronger baselines (BEV-based, multimodal, diffusion or transformer policies).

Acknowledgment

This work was partially funded by the National Council for Scientific and Technological Development – CNPq, Brazil (Grant No. 420148/2025-6), and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. G. G. Z. also acknowledges CNPq for the undergraduate research scholarship granted through PIBIC.

References

Bechtel, M. G., McEllhiney, E., Kim, M., and Yun, H. (2018). Deeppicar: A low-cost deep neural network-based autonomous car. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*.

Belsare, K., Rodriguez, A. C., Sanchez, P. G., Hierro, J., Kołcon, T., Lange, R., Lütkebohle, I., Malki, A., Losa, J. M., Melendez, F., et al. (2023). Micro-ROS. In *Robot Operating System (ROS): The Complete Reference (Volume 7)*, pages 3–55. Springer.

Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2019). Learning by cheating. In *Conference on Robot Learning (CoRL)*.

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Codevilla, F., Santana, E., López, A. M., and Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9329–9338.

Dosovitskiy, A., Ros, G., Codevilla, F., López, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, pages 1–16.

Florence, P., Lynch, C., Zeng, A., Ramirez, O., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2022). Implicit behavioral cloning. In *Conference on Robot Learning (CoRL)*, pages 158–168.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292.

Hyldmar, N., He, Y., and Prorok, A. (2019). A fleet of miniature cars for experiments in cooperative driving. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66).

Michel, O. (2004). Cyberbotics ltd. webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42.

Moore, T. and Stouch, D. (2016). A generalized extended kalman filter implementation for the robot operating system. In *Intelligent Autonomous Systems 13 (IAS-13)*, volume 302 of *Advances in Intelligent Systems and Computing*, pages 335–348. Springer.

O'Kelly, M., Zheng, H., Karthik, D., and Mangharam, R. (2020). F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. In *NeurIPS 2019 Competition and Demonstration Track*.

Paden, B., Čáp, M., Yong, S. Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55.

Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2107–2116.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334.