End-to-End Visual Autonomous Navigation with Twin Delayed DDPG in the CARLA and ROS 2 Ecosystem

Daniel Terra Gomes[1]

***Abstract*—The development of autonomous vehicle (AV) systems is a central challenge in robotics and artificial intelligence. While Deep Reinforcement Learning (DRL) is a powerful paradigm for learning continuous control, foundational actor-critic algorithms like the Deep Deterministic Policy Gradient (DDPG) suffer from function approximation errors that lead to overestimated Q-values and suboptimal policies. This work presents an end-to-end AV that utilizes the Twin Delayed DDPG (TD3) algorithm to mitigate this bias. Using primarily camera data within the CARLA simulator and the ROS 2 ecosystem, we aim to demonstrate the superiority of TD3 over the DDPG and baselines quantitatively for vehicle control. Therefore, the proposed research intends to establish a modular and reproducible framework for visual navigation, with expected outcomes showing significant improvements in vehicle control.**

***Index Terms*—Deep Reinforcement Learning, Autonomous Vehicles, Twin Delayed DDPG (TD3), CARLA, ROS 2, Visual Navigation, End-to-End Control.**

I. Introduction

The pursuit of fully autonomous vehicles (AV) has created significant research into robust decision-making and control systems [1]. While classical control methods (e.g., potential fields, geometric planners, rule-based systems) have proven effective, their reliance on hand-engineered rules limits their adaptability in complex, dynamic environments [2]. Machine Learning, particularly Deep Reinforcement Learning (DRL), offers a compelling alternative by enabling agents to learn optimal policies through direct interaction with the environment, reducing the need for exhaustive expert-labeled datasets and manual feature engineering [3].

For continuous control tasks inherent to driving, actor-critic algorithms like the Deep Deterministic Policy Gradient (DDPG) are a natural fit [2]. However, a well-documented flaw in value-based Reinforcement Learning (RL) is the overestimation bias, where function approximation errors lead to inflated Q-value estimates [3]. This problem persists in the actor-critic setting, often causing the agent to converge to suboptimal or unstable policies [4].

The Twin Delayed DDPG (TD3) algorithm was specifically proposed to address this critical issue [4]. It introduces three key mechanisms: clipped double Q-learning to curb value overestimation, delayed policy updates to promote learning stability, and target policy smoothing to regularize the learned policy. These enhancements have demonstrated superior performance and stability over DDPG in various continuous control benchmarks [4], [5].

While many state-of-the-art AV systems rely on sensor fusion (e.g., LiDAR, RADAR, and cameras), camera-only systems present an important and cost-effective research direction [2].

Therefore, this work proposes the development and evaluation of an end-to-end AV based on TD3, primarily using camera data. Our objective is to establish a strong visual navigation baseline and quantitatively demonstrate the benefits of TD3's architectural improvements over DDPG in a realistic simulation environment. The system will be built within the CARLA [6] and ROS 2 ecosystem [7], promoting a modular, reusable, and reproducible research framework consistent with modern robotics development practices [8].

II. Related Work

The application of DRL to AV in CARLA is an expanding field. Early work often compared the discrete-action Deep Q-Network (DQN) with the continuous-action DDPG, generally concluding that DDPG's continuous nature is better suited for the nuances of vehicle control [1], [2]. As DDPG's limitations became apparent, subsequent research has explored more advanced algorithms. Elallid et al. [5] successfully applied TD3 to the complex task of intersection navigation using only image sequences, highlighting its stability and safety benefits. The original TD3 paper by Fujimoto et al. [4] remains the foundational work, detailing the theoretical and empirical justification for the algorithm's design.

To improve architectural robustness and real-world applicability, researchers have integrated DRL agents with robotics middleware. Perez-Gil et al. [8] demonstrated a portable DRL framework using ROS and Docker, enabling a seamless transition from simulation to physical hardware. Others have focused on enhancing DRL with external knowledge, such as using real-world human driving data to refine a simulation-trained policy [9]. More recently, there is a strong focus on safety, with novel algorithms like RECPO being developed to ensure agents adhere to strict safety constraints, achieving zero-collision rates in highway scenarios [10]. A summary of this literature is presented in Table I.

The authors are with the Department of Computer Science, Federal University of Minas Gerais (UFMG), Belo Horizonte, MG, Brazil.

TABLE I
SUMMARY OF RELATED LITERATURE ON DRL FOR AV

Reference	Algorithm(s)	Main Contribution
Fujimoto et al. [4]	DDPG, TD3	Foundational paper proposing TD3 to address DDPG's overestimation bias through twin critics, delayed policy updates, and target policy smoothing regularization.
Ragheb & Mahmoud [1]	DQN, DDPG	Compares DQN and DDPG in CARLA using waypoints, finding DDPG achieves a higher average reward but DQN has a lower collision rate, highlighting the trade-offs between continuous and discrete action spaces.
Perez-Gil et al. (2022) [2]	DQN, DDPG	Implements and compares multiple DRL agents, concluding DDPG is superior to DQN for navigation due to its continuous nature. Demonstrates that agents relying only on CNNs perform worse than those with direct waypoint data.
Perez-Gil et al. (2021) [8]	DDPG	Proposes a distributed and portable software architecture using ROS, Docker, and CARLA to robustly train and validate DRL-based control algorithms for AV.
Elallid et al. [5]	TD3	Successfully applies TD3 for navigating complex T-intersections in dense traffic using a sequence of front-camera images as the state, demonstrating stable convergence and improved safety.
Li & Okhrin [9]	DDPG	Proposes a two-stage method that first trains a DDPG agent in simulation (CARLA with ROS) and then refines the policy by leveraging a real-world human driving dataset to achieve more human-like behavior.
Zhao et al. [10]	RECPO (Safe RL)	Introduces a safe RL algorithm with an experience replay pool using importance sampling to mitigate catastrophic forgetting and long-tail issues, achieving a zero-collision rate in highway driving scenarios.

III. METHODOLOGY

The proposed methodology is grounded in a modular system architecture, a mathematical problem formulation, and a description of the learning algorithm.

A. System Architecture

To ensure modularity and reusability, the system will be built within the ROS 2 ecosystem, which will serve as the middleware for communication between different software components [8]. The architecture comprises three primary nodes:

- **Simulation Node (CARLA):** runs the CARLA server, which is responsible for rendering the 3D environment, simulating vehicle physics, controlling non-player character (NPC) traffic, and publishing sensor data;
- **CARLA-ROS Bridge Node:** standard utility node [11] acts as a bidirectional interface. It subscribes to data streams from the CARLA server (e.g., camera images, vehicle status) and publishes them as ROS 2 topics. It also subscribes to ROS 2 control topics and translates them into commands for the ego vehicle in CARLA;
- **DRL Agent Node (TD3):** core of our system where the decision-making logic resides. This Python-based ROS 2 node will:
    1) Subscribe to the processed sensor and vehicle status topics;
    2) Construct the state vector at each time step;
    3) Perform inference using the trained TD3 actor network to generate an action;
    4) Publish the action to a control command topic, which is then read by the bridge and sent to CARLA.

B. Problem Formulation as an MDP

We model the AV task as a Markov Decision Process (MDP), formally defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ [3]. The transition dynamics $P(s_{t+1}|s_t, a_t)$ are unknown to the agent, necessitating a model-free approach like TD3:

- **State Space ($\mathcal{S}$):** the state $s_t \in \mathcal{S}$ at time t will be a concatenation of visual, kinematic, and goal-oriented information. This design is depicted in Fig. 1;

  Visual Input Processing (CNN Feature Extractor): to capture temporal dynamics, we adopt the approach from [5], using a stack of four consecutive, pre-processed front-camera images as input to a Convolutional Neural Network (CNN). The CNN acts as a feature extractor, converting high-dimensional pixel data into a compact latent vector. The specific architecture will be based on a simplified ResNet or MobileNet, leveraging Transfer Learning to initialize weights from a model pre-trained on a large-scale vision dataset used in [2].

  Kinematic and Goal-Oriented Features: this visual feature vector will then be concatenated with essential kinematic data (e.g., current vehicle velocity v_t, lateral deviation d_t, and heading error ϕ_t) and goal-oriented information (e.g., coordinates of the next few waypoints relative to the vehicle's local frame). The inclusion of waypoints is crucial for providing the agent with navigational intent beyond simple lane-following [2]. The final concatenated vector constitutes the complete state s_t fed to the TD3 actor and critic networks;
- **Action Space ($\mathcal{A}$):** $a_t \in \mathcal{A}$ will be a two-dimensional vector, $a_t = [\text{steering}, \text{throttle/brake}]$, where:
    - *steering*: A value in $[-1, 1]$, representing full left to full right.
    - *throttle/brake*: A value in $[-1, 1]$, where positive values $[0, 1]$ map to throttle and negative values $[-1, 0)$ map to braking intensity.

  This unified output for longitudinal control is more straightforward than having separate outputs for throttle and brake;
- **Reward Function (R):** $R(s_t, a_t)$ is engineered to promote driving behavior that is safe, efficient, and comfortable. It is structured as a weighted sum of several

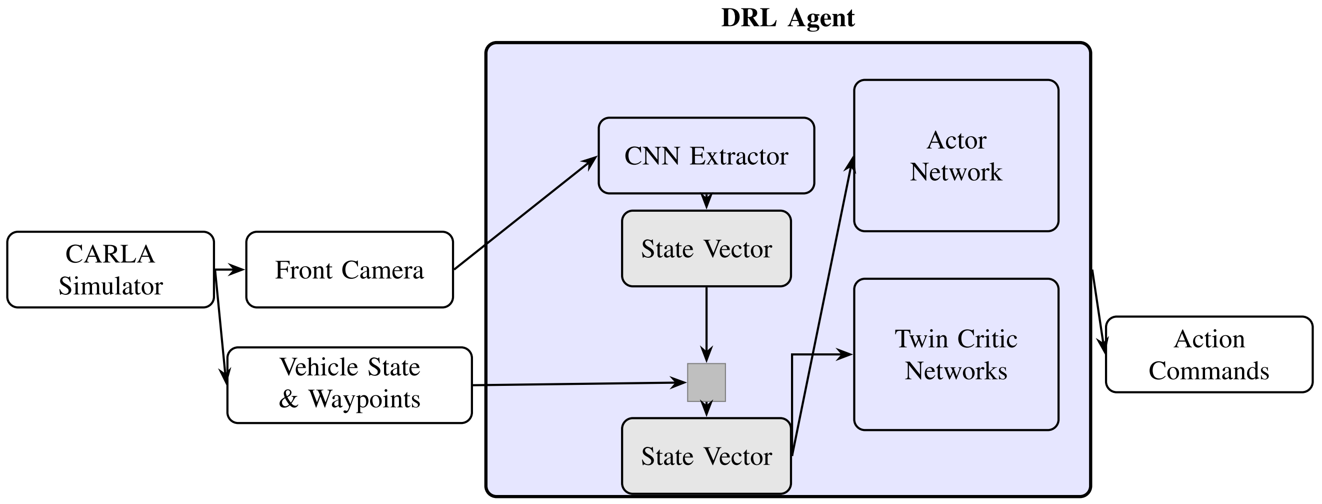

Fig. 1. Detailed DRL Agent architecture for end-to-end visual navigation. The CNN extracts features from camera images, which are concatenated with vehicle state and waypoint data to form the complete state vector for the TD3 actor and critic networks.

components, drawing inspiration from the safety-focused design in [10]:

- **Efficiency Reward:** positive reward for maintaining a velocity close to a target speed limit, while penalizing excessive speeding;
- **Lane Keeping Reward:** reward for minimizing the lateral distance to the center of the lane and the heading error;
- **Comfort Penalty:** penalty proportional to the magnitude of longitudinal jerk to discourage abrupt acceleration and braking;
- **Safety Penalty:** large, negative penalty for terminal events such as collisions with other objects.

- **Discount Factor (γ):** $\gamma = 0.99$. This high value encourages the agent to adopt a farsighted strategy, prioritizing long-term goals such as safely reaching the destination over myopic, immediate rewards [3].

C. *Twin Delayed DDPG (TD3) Algorithm*

Our implementation will be based on the original TD3 algorithm proposed by Fujimoto et al. [4]. As seen, the TD3 is a model-free, off-policy actor-critic algorithm that builds upon DDPG by introducing several key mechanisms to address function approximation errors and improve training stability. The agent learns a policy (actor) network π_ϕ and a pair of Q-value (critic) networks $Q_{\theta_1}, Q_{\theta_2}$. The core improvements are [4]:

1) **Clipped Double Q-Learning:** combat the overestimation bias inherent in actor-critic methods, the learning target for the critic networks is calculated using the minimum value from the two target critic networks $(Q_{\theta'_1}, Q_{\theta'_2})$. The target value y is computed as:

$$y = r + \gamma(1-d)\min_{i=1,2} Q_{\theta'_i}(s', \tilde{a}) \tag{1}$$

where r is the reward, γ is the discount factor, d is a flag for terminal states, and $\tilde{a}$ is the smoothed target action.

2) **Delayed Policy Updates:** the actor network and the target networks are updated less frequently than the critic networks (e.g., every two critic updates). This allows the critic's value estimates to converge toward a more stable target before being used to update the actor, which prevents propagating errors from a noisy value estimate into the policy.

3) **Target Policy Smoothing:** prevent the actor from overfitting to narrow peaks in the value function, a smoothed target action $\tilde{a}$ is used when calculating the target value y. This is achieved by adding clipped noise to the action selected by the target actor:

$$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) \tag{2}$$

This regularization encourages a smoother value landscape, making the learned policy more robust.

IV. Experimental Plan

To ensure a rigorous and fair evaluation, the experimental plan is designed to compare the proposed agent against relevant baselines across a standardized set of scenarios and metrics, following the practices outlined in [10].

A. *Test Scenarios*

Testing will be conducted in CARLA's 'Town01'. To assess generalization and robustness, experiments will be run, initially on a pre-defined route with a turn. Furthermore, tests will be repeated under different traffic densities (e.g., 20, 50, and 100 NPC vehicles) and routes to evaluate the agent's performance in both sparse and congested environments.

B. *Agents for Comparison*

A multi-faceted comparison to properly contextualize the performance of the proposed agent:

- **Proposed Agent (TD3):** the end-to-end visual TD3 agent as described in the methodology;

- **Primary Baseline (DDPG):** a standard DDPG implementation with an identical network architecture and state/action space. This will directly measure the impact of TD3's specific improvements;
- **Classical Baseline (IDM + MOBIL):** to benchmark the DRL agents against a well-established, non-learning approach, we will include a classical controller, and we will use the Intelligent Driver Model (IDM) for longitudinal control and the MOBIL model for lateral maneuvers. This provides a valuable reference for assessing whether the learned policies achieve performance comparable to or better than traditional, rule-based traffic models.

C. Evaluation Metrics

To provide a holistic assessment, performance will be evaluated across three key domains. Each metric will be averaged over a minimum of 20 test runs per scenario, and reported with mean and standard deviation.

- **Safety (Primary Priority):**
  - *Success Rate (%):* percentage of episodes completed without a safety violation (collision or off-road event). This is the most critical metric;
  - *Average Number of Collisions:* total number of collisions per kilometer driven;
  - *Time-to-Collision (TTC):* analysis of the minimum TTC values encountered to quantify near-miss events [9].
- **Efficiency:**
  - *Average Speed (km/h):* agent's average speed during successful episodes;
  - *Route Completion Time (s):* time taken to successfully complete the defined route.
- **Comfort:**
  - *Average Longitudinal Jerk (m/s^3):* measure of the smoothness of acceleration and braking;
  - *Average Lateral Acceleration (m/s^2):* measure of the smoothness of steering maneuvers.

The results will be compiled and analyzed to conclude the relative strengths and weaknesses of each algorithm.

References

[1] N. Ragheb and M. M. Mahmoud, "Implementing deep reinforcement learning in autonomous control systems," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 41, no. 1, pp. 168–178, 2024.

[2] Ó. Pérez-Gil, R. Barea, E. López-Guillén, L. M. Bergasa, C. Gómez-Huélamo, R. Gutiérrez, and A. Díaz-Díaz, "Deep reinforcement learning based control for autonomous vehicles in carla," *Multimedia Tools and Applications*, vol. 81, no. 3, pp. 3553–3576, 2022.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2018.

[4] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[5] B. B. Elallid, H. El Alaoui, and N. Benamar, "Deep reinforcement learning for autonomous vehicle intersection navigation," *arXiv preprint arXiv:2310.08595*, 2023.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the Conference on Robot Learning*. PMLR, nov 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[7] Robot Operating System. (2025) Ros 2 documentation. [Online]. Available: https://www.ros.org/blog/getting-started/

[8] Ó. Pérez-Gil, R. Barea, E. López-Guillén, L. M. Bergasa, C. Gómez-Huélamo, R. Gutiérrez, and A. Díaz, "Deep reinforcement learning based control algorithms: Training and validation using the ros framework in carla simulator for self-driving applications," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 1268–1273.

[9] D. Li and O. Okhrin, "Modified ddpg car-following model with a real-world human driving experience with carla simulator," *arXiv preprint arXiv:2112.14602*, 2022.

[10] R. Zhao, Z. Chen, Y. Fan, Y. Li, and F. Gao, "Towards robust decision-making for autonomous highway driving based on safe reinforcement learning," *Sensors*, vol. 24, no. 13, p. 4140, 2024.

[11] "CARLA ROS Bridge," https://carla.readthedocs.io/en/latest/ros_bridge/, 2025, accessed: Oct. 2025.