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

Course: Aprendizado por Reforço

Student: Daniel Terra Gomes

Professor: Luiz Chaimowicz

Belo Horizonte, Novembr 27, 2025

VERLab Computer Vision and Robotics | PROGRAMA DE PÓS-GRADUAÇÃO CIÊNCIA DA COMPUTAÇÃO UFMG | UFMG

Index

VERLab Computer Vision and Robotics
PROGRAMA DE PÓS-GRADUAÇÃO CIÊNCIA DA COMPUTAÇÃO UFMG
UFMG

The Autonomous Vehicle Challenge

Why Autonomous Vehicles?

- Safety: 94% of accidents caused by human error
- Efficiency: reduced traffic congestion
- Accessibility: mobility for all
- Because it is interesting!

Traditional Approaches:

- Rule-based systems (potential fields, geometric planners)
- Hand-engineered features
- Limited adaptability in complex scenarios

$\Rightarrow$ **Need for learning-based approaches!**

Deep Reinforcement Learning for AV

DRL Advantages: (SUTTON; BARTO, 2018; PÉREZ-GIL et al., 2022)

- Learn optimal policies through interaction;
- No need for labeled datasets;
- Adaptable to dynamic environments;
- End-to-end learning from sensors to control.

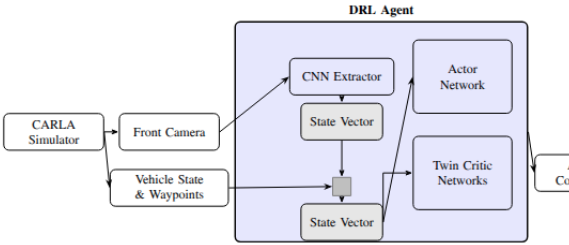


Figure: RL agent-environment interaction loop

The DDPG Problem: Value Overestimation

Deep Deterministic Policy Gradient (DDPG):

- Natural fit for continuous control (steering, throttle)
- Actor-critic architecture
- BUT: Suffers from Q-value overestimation bias!

The Problem

Function approximation errors $\Rightarrow$ Inflated Q-estimates
$\Rightarrow$ Suboptimal/unstable policies $\Rightarrow$ Training collapse

Solution: Twin Delayed DDPG (TD3) (FUJIMOTO; HOOF; MEGER, 2018)

Research Objective

Main Goal

Develop and evaluate an **end-to-end visual autonomous navigation system** using TD3 in a realistic simulation environment

Key Components:

1. Camera-only perception (cost-effective approach)
2. TD3 algorithm to mitigate overestimation bias
3. CARLA 0.9.16 simulator + ROS 2 Humble integration
4. Modular, reproducible framework
5. Quantitative comparison with classical baseline

Twin Delayed DDPG (TD3) - Key Innovations

Three mechanisms to address DDPG's overestimation:

1. Clipped Double Q-Learning

Use **minimum** of two Q-networks for target:

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

Prevents optimistic bias from bootstrapping

2. Delayed Policy Updates

Update actor & targets less frequently than critics (e.g., every 2 steps)

$\Rightarrow$ More stable value estimates before policy update

VERLab Computer Vision and Robotics | Programa de Pós-Graduação Ciência da Computação UFMG | UFMG

Twin Delayed DDPG (TD3) - Key Innovations

Three mechanisms to address DDPG's overestimation:

3. Target Policy Smoothing

Add clipped noise to target action:

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

Regularizes value function, prevents overfitting

VERLab Computer Vision and Robotics | Programa de Pós-Graduação Ciência da Computação UFMG | UFMG

MDP Formulation for Autonomous Driving

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$

State Space $\mathcal{S}$:

- **Visual:** 4 stacked front-camera frames (84×84 grayscale) $\rightarrow$ CNN features
- **Kinematic:** velocity v_t, lateral deviation d_t, heading error ϕ_t
- **Goal:** next waypoints coordinates (relative to vehicle frame)

Action Space $\mathcal{A} \in \mathbb{R}^2$:

$$a_t = [\text{steering}, \text{throttle/brake}] \in [-1, 1]^2$$

Discount Factor: $\gamma = 0.99$ (long-term planning)

Reward Function Engineering

Multi-component reward: $R(s_t, a_t) = \sum w_i \cdot r_i(s_t, a_t)$

Component	Weight	Key Parameters
Efficiency	2.0	Target: 30 km/h (8.33 m/s)
Lane Keeping	2.0	Tolerance: 0.5m lateral, 5.7° heading
Comfort	1.0	Jerk threshold: 5.0 m/s^3
Safety	1.0	Collision: -100, Offroad: -50
Progress	3.0	Waypoint bonus: +1.0

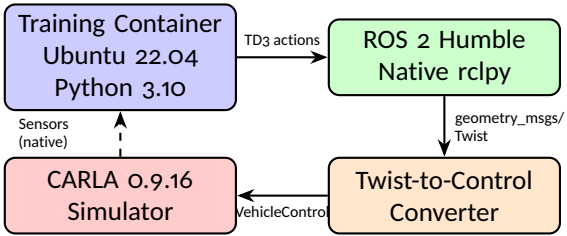
Design Principles:

- Balanced weights prevent reward domination
- Strong progress incentive (weight: 3.0) encourages goal-directed behavior
- Safety penalties severe but recoverable (TD3's pessimism provides inherent caution)

VERLab Computer Vision and Robotics | Programa de Pós-Graduação Ciência da Computação UFMG | UFMG

ROS 2 + CARLA Integration Architecture

Challenge: CARLA 0.9.16 has native ROS 2 for *sensors only*, not vehicle control



Solution: Standard ROS 2 messages + custom converter

- **geometry_msgs/Twist:** Standard ROS 2 (no custom packages!)
- **630× faster** than docker-exec approach
- **2.3 GB smaller** image (no carla_msgs build)

CNN Feature Extractor Architecture

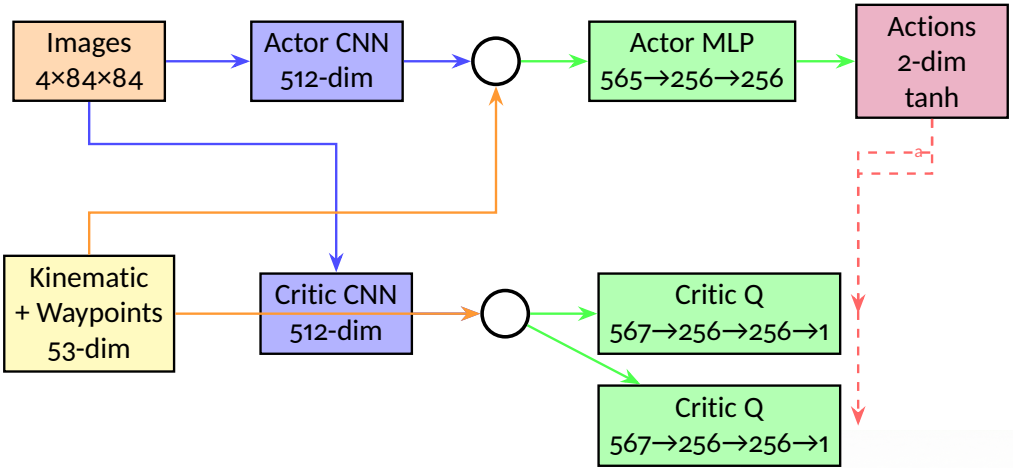
Visual Processing Pipeline:

1. Input: 4 stacked grayscale frames (84×84) $\rightarrow$ temporal dynamics
2. NatureCNN architecture (from DQN paper):
   - Conv1: 32 filters, 8×8 kernel, stride 4
   - Conv2: 64 filters, 4×4 kernel, stride 2
   - Conv3: 64 filters, 3×3 kernel, stride 1
   - Fully connected: 512 units
3. Output: 512-dim feature vector
4. Concatenate with kinematic features (53-dim)

Why this design?

- Proven in Atari DQN (MNIH et al., 2015)
- Captures spatial hierarchies and motion patterns
- Used by (ELALLID; ALAOUI; BENAMAR, 2023) for intersection navigation

TD3 Network Architecture



TD3 Network Architecture

Data Flow: Images → Separate CNNs (512) Kinematic (53) → State (565) → Actor/Critics

Key Innovation: Separate CNNs prevent gradient interference (FUJIMOTO; HOOF; MEGER, 2018)

Training: LR=1e-3, batch=256, =0.99, =0.005, policy delay=2

TD3 Innovation: Clipped Double Q-Learning

Problem: Single-critic methods (DDPG) suffer from *overestimation bias*

- Function approximation errors cause Q-values to be overestimated
- Actor exploits these overestimates $\rightarrow$ learns suboptimal policies

TD3 Solution: Twin Critics with minimum operator

DDPG (Single Critic):

$$y = r + \gamma Q_{\theta'}(s', \mu_{\phi'}(s'))$$

Overestimates Q-values

Unstable training

TD3 (Twin Critics):

$$y = r + \gamma \min(Q_{\theta'_1}(s', \tilde{a}), Q_{\theta'_2}(s', \tilde{a}))$$

Conservative estimate

Stable convergence

TD3 Innovation: Clipped Double Q-Learning

How it works:

1. Two independent Q-networks estimate $Q_1(s, a)$ and $Q_2(s, a)$
2. Target computation uses **minimum**: $\min(Q_1', Q_2')$ (upper bound)
3. Both critics train toward the **same conservative target**
4. Actor maximizes $Q_1(s, \mu(s))$ for policy gradient

Result: Prevents value overestimation, improves policy quality (FUJIMOTO; HOOF; MEGER, 2018)

CARLA Environment Configuration

Simulation Setup:

- CARLA 0.9.16, Town01 with a 222m pre-defined route waypoint path with turn and 20 NPC vehicles

Figure: CARLA Town01 testing environment

Episode Termination: Collision, off-road, waypoint completion, or 1000 steps

Baseline Controller Performance

Classical PID + Pure Pursuit (6 episodes, 20 NPCs):

Metric	Mean	Std Dev
Safety		
Success Rate (%)	50.0	–
Collisions/km	2.787	2.850
Min TTC (s)	0.25	–
Efficiency		
Avg Speed (km/h)	24.62	4.92
Completion Time (s)	16.5	4.9
Comfort		
Avg Jerk (m/s^3)	17.779	1.111
Avg Lateral Accel (m/s^2)	0.105	0.105

Key Findings: High collision rate, abrupt acceleration (high jerk)

Baseline Trajectory Visualization

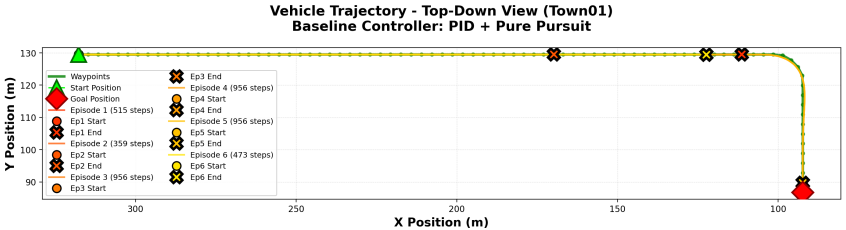


Figure: PID + Pure Pursuit trajectory: reactive behavior, collision in dense traffic

Observations:

- Follows waypoints accurately in sparse traffic
- Reactive (not predictive) $\Rightarrow$ collisions in congested areas
- Lacks adaptation to dynamic NPC behavior

TD3 Training Progress (100K Timesteps)

Training Phases:

Metric	Exploration	Learning	Late Learn.
Episodes	10	501	(final)
Avg Length (steps)	60.4	180.0 ±317.5	656.0 ±468.8
Collision Rate (%)	0.0	0.4	1.0
Lane Invasion (%)	0.0	1.0	1.0
Mean Reward	74.10 ±39.62	-3.58 ±197.05	-322 ±230

Key Insights:

- Successful system validation - environment wrapper, reward function, TD3 implementation all working
- Episode lengths increased 10× (60 → 656 steps)
- High variance due to stochastic NPC traffic
- Partial convergence - requires recommended 1M timesteps (????)

Computational Constraints & Achievements

Challenge

Limited access to HPC infrastructure prevented full training (1M timesteps)
Completed only 100K timesteps (10% of recommended duration)

Despite constraints, we achieved:

- **Complete system validation** - TD3 + CARLA + ROS 2 integration
- **Baseline benchmarks established** - quantitative comparison metrics
- **ROS 2 innovation** - geometry_msgs/Twist integration (2.3 GB smaller Docker image)
- **Reproducible framework** - open-source code, config management
- **Initial policy learning** - episode lengths improved from 60 to 656 steps

Next Steps: Complete 1M training on available infrastructure

ROS 2 CARLA Bridge Innovation

Challenge: Traditional CARLA ROS 2 integration requires building custom carla_msgs package

Traditional Approach:

- Custom carla_msgs package
- CarlaEgoVehicleControl
- Build dependencies (colcon, etc.)
- $\sim$2.8 GB larger image
- Simulation-specific code

Our Approach:

- geometry_msgs/Twist
- Standard ROS 2 message
- No compilation needed
- Minimal dependencies
- Platform-agnostic

Architecture:

Training Container → Twist msgs → CARLA Twist-to-Control Node → CARLA API

Main Contributions

1. Reproducible TD3 Framework for Autonomous Driving

- Complete end-to-end visual navigation system
- Open-source code with comprehensive documentation
- Configuration management for hyperparameters

2. Innovative ROS 2 + CARLA Integration

- Standard geometry_msgs/Twist (no custom packages)
- 33% smaller images with native rclpy integration
- Platform-agnostic design for sim-to-real transfer

3. Quantitative Baseline Benchmarks

- PID + Pure Pursuit: 50% success, 2.787 collisions/km
- Establishes comparison metrics for DRL agents
- Highlights limitations of reactive control

Technical Achievements & Validation

Validated System Components:

- TD3 algorithm implementation (clipped double-Q, delayed updates, target smoothing)
- CNN feature extractor (NatureCNN with 512-dim output)
- Multi-component reward function (safety/efficiency/comfort)
- CARLA environment wrapper (Gymnasium v0.26+ API)
- ROS 2 Humble native integration

Initial TD3 Training Results (100K timesteps):

- Episode lengths: 60 $\rightarrow$ 656 steps (10× improvement)
- Collision rate: 0.4% during learning phase
- Demonstrates policy convergence beginning
- High variance due to stochastic traffic dynamics

Challenges Encountered

1. **Environment Wrapper Development**
   - CARLA Python API synchronous mode management
   - Sensor callback handling and episode termination logic
   - Gymnasium v0.26+ API migration (terminated/truncated separation)

2. **ROS 2 + CARLA Compatibility**
   - Ubuntu 20.04 (Python 3.8) vs 22.04 (Python 3.10)
   - CARLA 0.9.16 + PyTorch + ROS 2 Humble dependencies
   - Solved with geometry_msgs/Twist approach

3. **DRL Algorithm Debugging**
   - Distinguishing bugs from expected stochastic behavior
   - Reward function engineering (balancing safety/progress/comfort)
   - Validation against reference implementations (Stable-Baselines3)

4. **Computational Constraints**

Future Work Directions

Immediate Next Steps:

- Complete 1M timestep training with HPC access
- Evaluate TD3 vs baseline across full metrics
- Analyze learned policy behavior and failure modes

Multi-Scenario Generalization:

- Test on Town02-07 (transfer learning)
- Weather variations (rain, fog, night)
- Traffic densities (50-100 NPCs)

Advanced Algorithms:

- Soft Actor-Critic (SAC) with entropy tuning
- Safe RL methods (CPO, RECPO)
- Comparative study: TD3 vs SAC vs PPO

Future Work (continued)

Sim-to-Real Transfer:

- Domain randomization in CARLA
- Domain adaptation techniques
- Fine-tuning with real-world data
- Validation on physical vehicle platforms

Multi-Sensor Fusion:

- Integrate LiDAR point clouds
- Semantic segmentation masks
- Enhanced perception robustness

Hierarchical Control:

- High-level planning (route selection, lane changes)
- Low-level control (steering, throttle)
- Improved sample efficiency and interpretability (CHEN; LI; TOMIZUKA, 2020)

Key Takeaways

Research Question

Can TD3 address DDPG's overestimation bias for end-to-end visual autonomous navigation?

Answer:

1. System Validated - Complete TD3 + CARLA + ROS 2 framework functional
2. Innovation Demonstrated - geometry_msgs/Twist integration reduces complexity
3. · · · Training Incomplete - 100K/1M timesteps (10%) due to HPC constraints
4. Foundation Established - Baseline benchmarks + modular architecture for future research

Impact:

- Reproducible framework for DRL-based AV research
- Standard ROS 2 integration pattern for CARLA
- Quantitative metrics for classical vs learning-based approaches

Thank You!

Questions?

Daniel Terra Gomes
Department of Computer Science
Federal University of Minas Gerais (UFMG)

danielterragomes@dcc.ufmg.br

Code Repository: github.com/ARRETdaniel/av_td3_system
Advisor: Prof. Luiz Chaimowicz

References I

CHEN, J.; LI, S. E.; TOMIZUKA, M. **Interpretable End-to-end Urban Autonomous Driving with Latent Deep Reinforcement Learning**. 2020. Disponível em: <https://arxiv.org/abs/2001.08726>. 27

ELALLID, B. B.; ALAOUI, H. E.; BENAMAR, N. Deep reinforcement learning for autonomous vehicle intersection navigation. **arXiv preprint arXiv:2310.08595**, 2023. 12

FUJIMOTO, S.; HOOF, H. van; MEGER, D. **Addressing Function Approximation Error in Actor-Critic Methods**. 2018. Disponível em: <https://arxiv.org/abs/1802.09477>. 5, 14, 16

References II


MNIH, V. et al. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

12


PÉREZ-GIL, Ó. et al. Deep reinforcement learning based control algorithms: Training and validation using the ros framework in carla simulator for self-driving applications. In: IEEE. **2021 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.], 2021. p. 1268–1273.


PÉREZ-GIL, Ó. et al. Deep reinforcement learning based control for autonomous vehicles in carla. **Multimedia Tools and Applications**, Springer, v. 81, n. 3, p. 3553–3576, 2022.

4

References III


SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. 2. ed. Cambridge, MA: MIT Press, 2018. (Adaptive Computation and Machine Learning). ISBN 978-0262039246.

4