Hybrid Decision Making for Autonomous Driving in Complex Urban Scenarios

Rodrigo Gutiérrez-Moreno[1], Rafael Barea[1], Elena López-Guillén[1], Felipe Arango[1], Navil Abdeselam[1] and Luis M. Bergasa[1]

2023 IEEE Intelligent Vehicles Symposium (IV) | 979-8-3503-4691-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IV55152.2023.10186666

***Abstract*— Autonomous driving presents significant challenges due to the variability of behaviours exhibited by surrounding vehicles and the diversity of scenarios encountered. To address these challenges, we propose a hybrid architecture that combines traditional and deep learning techniques. Our architecture includes strategy, tactical and execution modules. Specifically, the strategy module defines the trajectory to be followed. Then, the tactical decision module employs a proximal policy optimization algorithm and deep reinforcement learning. Finally, the maneuver execution module uses a linear-quadratic regulator controller for trajectory tracking and a predictive model controller for lane change execution. This hybrid architecture and the comparison with other classical approaches are the main contributions of this research. Experimental results demonstrate that the proposed framework solves concatenated complex urban scenarios optimally.**

I. Introduction

Driving in urban scenarios requires an intelligent decision-making system, capable of solving multiple situations, processing the information of the environment, and executing safe actions.

Decision-making for autonomous vehicles is typically divided into two categories: classical methods and learning-based methods. In complex dynamic environments where autonomous vehicles interact with other traffic participants, classical methods may not be effective due to lack of flexibility. Therefore, learning-based methods are often used to improve decision-making because they use real actions in the learning process to manage uncertainty. Additionally, with the advance of powerful computational technologies, learning-based approaches have become increasingly popular and developed in the field of autonomous vehicles [1]. In particular, Reinforcement Learning (RL) allows the vehicle to interact with the environment to learn the optimal action for each situation [2]. RL does not require human labeling or supervision. Instead, during its learning process it needs a reward function defined to reach the optimal policy.

This paper introduces a hybrid decision-making system for urban scenarios, like the shown in Fig. 1, based on a Proximal Policy Optimization (PPO) [3] algorithm. We combine Deep Reinforcement Learning (DRL), High Definition (HD) map information, and classic low-level control techniques in the decision-making process.

*This work has been funded in part from the Spanish MICINN/FEDER through the Artificial Intelligence based modular Architecture Implementation and Validation for Autonomous Driving (AIVATAR) project (PID2021-126623OB-I00), Electric Automated Vehicle for Aging Drivers (AVAD) project (PDC2022-133470-I00) and from the RoboCity2030-DIH-CM project (P2018/NMT- 4331), funded by Programas de actividades I+D (CAM), cofunded by EU Structural Funds and Scholarship for Introduction to Research activity by University of Alcalá.

[1]R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, Felipe Arango, Navil Abdeselam and L.M. Bergasa are with the Electronics Departament, University of Alcalá (UAH), Spain.{rodrigo.gutierrez, rafael.barea, elena.lopezg, juanfelipe.arango, navil.abdeselam, luism.bergasa}@uah.es

A. Related Works

Decision-making systems for autonomous vehicles have been the subject of extensive research in recent years. A number of different approaches have been proposed to address the challenges of creating a safe and efficient system. In this section, we present a brief description of classical and learning-based methods.

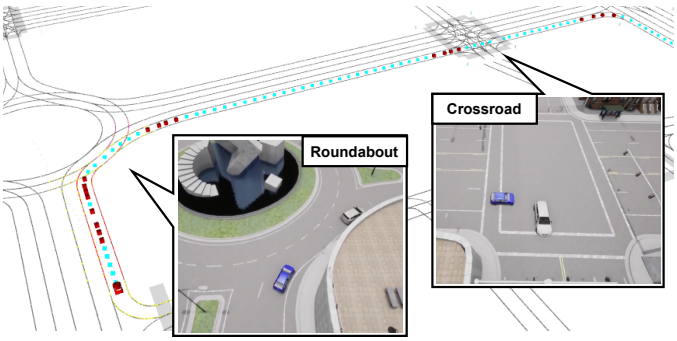


Fig. 1. Concatenated urban scenarios. Map and trajectory visualization. Top view of a roundabout and a crossroad use cases. Ego vehicle in blue and adversary in white.

One classical approach is the use of rule-based systems, where a set of predefined rules are used to make decisions. Finite-State-Machine (FSM) is the most representative of these methods [4]. Apart from FSM, some research is carried out using specific ruled-based approaches with a hybrid low diagram [5] or a hierarchical framework [6].

Another approach is the use of optimization methods, which use a utility function to generate decisions. Some works, such as [7], use Model Predictive Control (MPC) for controlling traffic situations. Others apply Game-Theory methods to adopt the optimal strategy to solve driving scenarios [8].

Probabilistic methods utilize principles of probability theory to generate behaviour results. To determine behaviour, a probabilistic model must be established. In [9], a probabilistic graphical model (PGM) was used to generate motion commands by estimating the intentions of adversaries in a merging scenario. In [10], a lane change scenario is solved using a Two-Sequential Level Bayesian Decision Network.

Learning-based methods refer to the use of artificial intelligence in the decision-making process. The use of these techniques has increased in recent years. In [11], support-vector machines (SVM) were trained to execute lane changes using the position and velocities of surrounding vehicles. Besides, deep learning-based methods are used for end-to-end approaches, using raw sensor data as input and low-level control [12].

Regarding RL-based approaches, a commonly utilized technique is the Deep Q-Network (DQN) algorithm for high-level decisions, as previously reported in literature [13]. In [14], the input vector used in the algorithm includes information about the ego vehicle, the existing lanes, and the surrounding vehicles. Two algorithms were trained, one specifically for lane change actions, and the other for acceleration purposes. The authors also introduced a novel method applying a convolutional neural network to high-level inputs. In [15], a minimal state representation and fast learning rates were presented, resulting in improved performance in terms of collisions in highway environments when compared to rule-based algorithms. An algorithm trained for decision-making at occluded intersections, utilizing a risk-based reward function, was proposed in [16]. A partially observable Markov decision process (POMDP) based low-level planner was defined in [17], resulting in safe planning outcomes with high-commute efficiency at unsignalized intersections in real-time. The work reported in [18] employs a novel simulation environment for training the algorithm, utilizing a variety of traffic scenarios to simulate a realistic environment, resulting in robust and reliable performance under noisy observations. In [19], an algorithm using a proximal policy optimization (PPO) based on automated lane change was proposed, utilizing the SUMO simulation environment.

B. Contribution

This work focuses on achieving a realistic approach to driving under different concatenated complex urban scenarios. While most works in the literature focus on individual use cases, we develop a whole autonomous driving system for multiple use cases. With our novel hybrid approach, we use RL to manage the uncertainty of the surrounding elements; and we use classic planning, mapping, and control techniques to ensure safe and robust behaviour. The results show our proposal works better than other classical approaches.

II. Background

In this section, we introduce the three decision-making approaches evaluated in this work: Time-to-Collision (TTC), which is a standard decision-making method; our previous classical decision-making module [20] based on Petri nets (PN); and a partially observable Markov decision process (POMDP), developed in this research.

A TTC decision-making module calculates the time until a potential collision based on the relative positions and velocities of the ego vehicle and the detected objects. The module then uses this information to make decisions about how to safely navigate the vehicle, such as by braking or steering to avoid a collision.

Our Petri nets decision-making module is a graphical representation of the vehicle's behaviour, where the nodes represent states and the edges represent transitions between states. This module analyzes and optimizes decision-making processes.

This work defines urban scenarios as POMDPs. We solve these POMDPs using a DRL approach, in which the ego-vehicle is defined as an agent. This agent receives information from the environment as observations and performs actions regarding a policy. This policy is updated using a reward in a process known as training. A graphic representation of this methodology is described in Fig. 2.

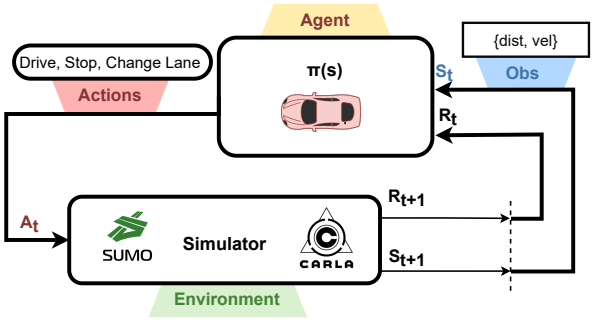


Fig. 2. An observation matrix is extracted from the simulator. The agent selects an action considering this matrix and the reward for this action.

A POMDP is defined as a tuple (S, A, Ω, T, O, R), where S is a set of states, A is a set of actions, Ω is a set of observations, T is a transition function, O is an observation function, and R is a reward function. The agent receives an observation $o \in \Omega$, rather than observing the true state s' directly. The agent's internal knowledge of the state is represented by the belief $b(s)$, which is the probability of being in a state s. The optimal policy $\pi^*(b)$ maps beliefs to actions.

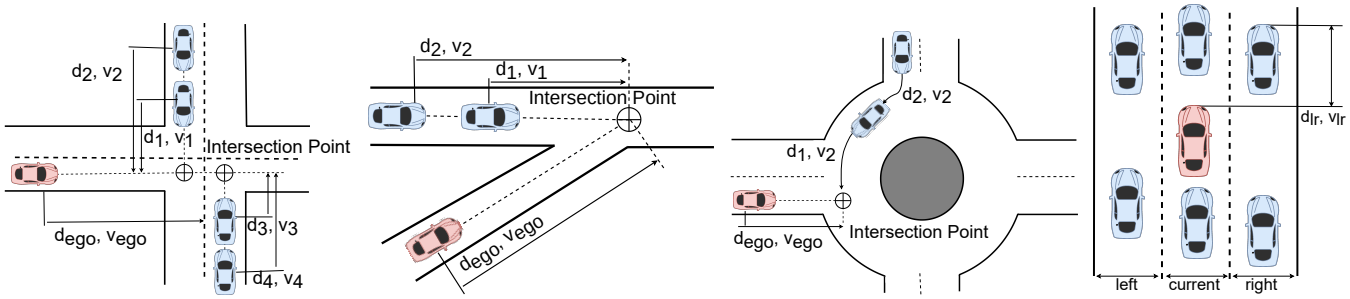
We propose using a policy-based method, which learns the policy function directly. This policy maps states to actions. The method we use is the Proximal Policy Optimization (PPO) algorithm [3]. The loss function used in PPO is given by Eq. 1, where L_t^{CLIP} is a clipping function, L_t^{VF} is the value function loss, $S[\pi_\theta]$ represents the entropy of the policy at timestep t, and $c1, c2$ are coefficients.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{1}$$

In each iteration, the algorithm runs the policy for Ts timesteps within a given length-Ts trajectory segment and uses the collected samples for an update.

$$\begin{aligned} \hat{A}_t &= \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1} \\ &where\ \delta_t = r_t + \gamma V(s_t + 1) - V(s_t) \end{aligned} \tag{2}$$

A truncated version of generalized advantage estimation is used, given by Eq. 2, where δ_t is the temporal-difference



(a) (b) (c) (d)

Fig. 3. Definition of the observation matrix for each scenario. The ego vehicle (red) and the adversaries (blue) are described for each use case. (**a**)Intersection. (**b**) Merge. (**c**) Roundabout. (**d**) Lane Change.

error and $\hat{A}_t$ is the estimated advantage at timestep t. N parallel actors collect T timesteps of data and the surrogate loss function is constructed on these NT timesteps of data. This function is optimized using minibatch stochastic gradient descent (SGD) for K epochs.

III. Modeling Urban Scenarios

For developing our decision-making framework we need to represent the urban scenarios as POMDPs. We consider intersections and multiple lanes roads for this research. Our goal is to use an RL algorithm to execute high-level decisions in these scenarios, dealing with a continuous observation matrix and a set of discrete actions. We define our vehicle as the ego vehicle and the surrounding vehicles as adversaries.

A. Intersections

A large variety of intersections can be found in an urban environment. In this work, we focus on three types of intersections: crossroads, merge, and roundabouts. For simplicity, we use the same representation as POMDPs for the three scenarios. However, each of them is solved using a different policy.

1) State: The state of a vehicle, as shown in Fig. 3, is defined by its distance to the intersection point, its longitudinal velocity, and its driving intention: $s_i = (d_i, v_i, i_i)$, these physical values are normalized between $[0, 1]$ and the driving intention of the adversarial vehicles are described by $i \in \{0, 1, 2\}$. In the crossroads, the trajectory to be followed by the adversarial vehicles is defined by their intention to turn left $i = 1$, keep straight $i = 0$, or turn right $i = 2$. In the merge and the roundabout, a vehicle with $i = 0$ may yield, while a vehicle with $i = 2$ does not yield. The state of the environment is defined as the collection of the individual states of all the vehicles: $s = (s_e, s_1, s_2, ..., s_n)$, where s_e is the ego vehicle state and $s_1, s_2, ..., s_n$ are the states of the n adversaries.

2) Observation: To create realistic experiments, we assume that the detection, by a theoretical perception module, can obtain vehicle information within 50 metres in 360º. We define the observation vector without prior information about the intersection type. The ego vehicle can observe the longitudinal velocity and position of the surrounding vehicles, being the observation vector defined as $o = (d_e, v_e, d_1, v_1, ..., d_4, v_4)$, only considering the ego vehicle and the four closest adversaries.

3) Action: We propose a discrete action space formed by just two high-level actions. The low-level controller is in charge of performing smooth driving based on these actions. These actions are focused on when the vehicle has to cross the intersection and when it has to stop. The action space is defined as $a = (stop, drive)$.

4) Reward: The objective of an RL algorithm is to optimize the expected value of the discounted future reward. Different reward functions will modify the resultant policy. The purpose of the reward function, in these use cases, is to perform the rapid and safe navigation of the ego vehicle through an intersection, avoiding collisions with adversarial vehicles. Collisions, result in a negative reward, while successful navigation results in a positive reward. To further encourage the vehicle's forward progression, a cumulative reward based on longitudinal velocity is proposed. Additionally, at the end of each episode, a negative reward is assigned proportionally to its duration, where t_e represents the episode duration and t_{out} represents the maximun episode time. The reward function is defined in the following components:

- Reward based on the velocity: $k_v \cdot v_{ego}$;
- Reward for crossing the intersection: $+1$;
- Penalty for collisions: -2;
- Penalty relative to the episode duration: $-0.2t_e/t_{out}$.

B. Roads

Roads in urban scenarios may have one or multiple lanes. We propose a lane change module to drive in these situations, where a slow or stopped vehicle can hinder ego vehicle driving. In this section, we work with a three lanes road which can be extrapolated to any number of lanes.

1) State: The state of a vehicle is defined by its distance to the ego vehicle, its longitudinal velocity, and its driving behaviour: $s_i = (d_i, v_i, i_i)$, these physical values are normalized between $[0, 1]$ and driving intention of the adversarial vehicles are described by $i \in \{0, 1, 2\}$. In this scenario, a vehicle may change left $i = 1$, keep driving in its lane $i = 0$, or change right $i = 2$. We define the state of the environment as the collection of the individual states of the adversarial vehicles: $s = (s_1, s_2, ..., s_n)$, being n the number of adversaries.

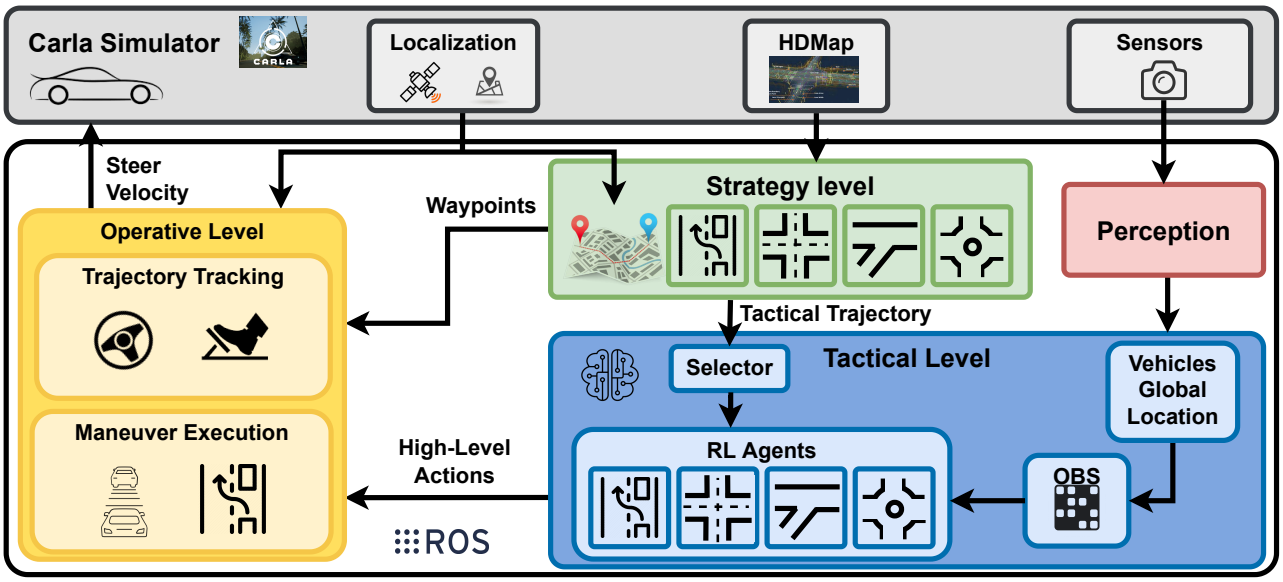


Fig. 4. The proposed hybrid architecture. The strategy level defines a tactical trajectory with the map information and the ego vehicle location. The tactical level executes high-level actions in correlation with the perception information. The operative level combines the trajectory and the actions, calculating the driving commands.

2) Observation: The observation matrix is defined by the nearest vehicles in the current and contiguous lanes of the ego vehicle. As shown in Fig. 3d we consider the information of six vehicles, three leading vehicles, and three following vehicles. The observation space is composed of the relative normalized distances to these vehicles: $o = (d_{ll}, d_{lc}, d_{lr}, d_{fl}, d_{fc}, d_{fr})$, where the indices indicate the lane of the vehicle and its relative position with the ego vehicle.

3) Action: The RL algorithm only controls the lane-changing decisions while the velocity is automatically controlled by the operative layer. This is done by an Adaptive Cruise Control (ACC) module, which softly adapts the ego vehicle velocity to the leading vehicle as it gets closer. We propose a discrete set of actions: change to the left lane, continue in the current lane, and change to the right lane: $a = (change\ left, idle, change\ right)$.

4) Reward: We aim to drive at a desired velocity without colliding with adversarial vehicles. A positive reward is given when the vehicle reaches the end of the road and a negative reward is given when it collides. We propose a cumulative reward based on its longitudinal velocity. In addition, a positive reward is given when the vehicle is in the right lane:

- Reward based on the velocity: $k_v \cdot v_{ego}$;
- Reward based on the right lane: k_i;
- Reward for reaching the end of the road: $+1$;
- Penalty for collisions: -2;

IV. Hybrid Architecture

This paper aims to address a completely autonomous driving system, including high-level decision-making and maneuver control in a hybrid architecture. It is divided into four levels as it is described in Fig. 4. The perception level process the camera data, the strategy level defines a tactical trajectory, the tactical level is in charge of the high-level decisions and the operative level executes the control maneuvers.

A. Perception Level

The perception data are obtained directly from the ground truth of the simulator. In practical implementations, these data may be acquired through the integration of cameras and lidars.

B. Strategy Level

The path planning module [21] uses the HD map input to generate a graph of lanes and roads. Then it applies the Dijkstra algorithm to calculate a topological route between the ego vehicle position and the goal location. Finally, a list of waypoints is generated. The route is calculated every time a new goal location is set. The trajectory also describes the scenarios that will be found in the route. This route, containing the waypoints and the environment information is defined as the tactical trajectory.

C. Tactical Level

Once we have introduced how the scenarios are represented as POMDPs we define our decision-making module. The selector module receives the tactical trajectory information and the ego vehicle location. With this information, an RL agent is selected when the ego vehicle approaches a scenario. The location and velocity of each adversary are provided by the perception level and processed into an observation matrix. This matrix is the input for the RL agent, which executes a high-level action.

We propose the neural network structure for the RL agents, consisting of a features extractor module and an actor-critic approach. The features of the ego vehicle and the adversaries are the input to the extractor. This features extractor is trained

in a supervised way, being the weights shared between the actor/critic network. The extracted features are concatenated in a single vector, which is the PPO input. This PPO algorithm has two models: the Actor, which is in charge of choosing an action representing the policy; and the Critic, which corresponds with the value function. Both models are defined with a simple multi-layer perceptron (MLP) formed by two hidden layers of 128 neurons each. The net structure is shown in Fig. 5.

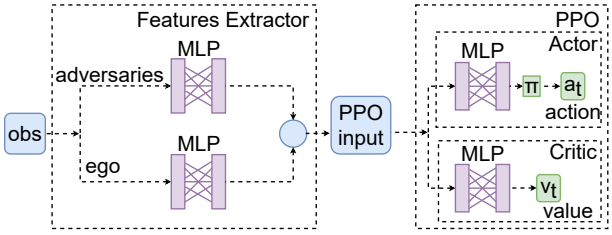


Fig. 5. Neural network: actor-critic structure with two layers of 128 neurons each.

D. Operative level

The operative level has two modules. The trajectory tracking module and the maneuver module.

The trajectory tracking is performed by our classic controller [22]. It performs a smooth interpolation of the waypoints generated by the planner. The steering command to follow this trajectory is calculated using LQR techniques. A velocity profile is defined using the trajectory section's curvature radius and the velocity limit before navigation begins. The velocity command is adjusted using this profile.

The maneuver module is in charge of executing the actions selected by the tactical level. The lane change action is performed using our MPC controller [23]. The stop-and-drive actions are performed using a ramp signal, ensuring comfort and safety. An ACC module is continuously running in the background to maintain a safe distance from the leading vehicle.

V. Experiments

Our architecture is designed for concatenated complex urban scenarios. We develop rich and complex use cases for training and validation.

A. Environmental Setup

We propose four driving scenarios according to our definitions in section III. We divide them into intersections (crossroads, merge, and roundabout) and roads (lane change).

1) Intersection: Traffic generation is one of the essentials to obtain a realistic simulation. The traffic density and vehicle features are similar for the three scenarios. A new vehicle is generated in the simulation every $sec \in [5, 10]$ seconds, these vehicles are 4 meters long and follow a standard Intelligent Driver Model (IDM) technique [24] with a commanded velocity of $v \in [4, 6]$ m/s. The adversarial vehicles present different behaviours. At crossroads, they can turn in any direction, they will yield if another vehicle is already inside the intersection and they can accelerate or decelerate. At the merge, the ego-vehicle must learn to yield and incorporate in a safe way. A similar behaviour must be learned in the roundabout, where these vehicles do not usually yield.

2) Roads: We have designed a lane change environment, consisting of three lanes forming a 400 metres road. Three types of adversarial vehicles are generated in a random lane with a probability of 0.15 every second. These vehicles have a maximum velocity of 5, 10, and 15 m/s. They may perform a lane change as defined in section III-B.1.

3) Training: Following the principles of curriculum learning, we start training with low vehicle density and then increment the scenario complexity by adding more vehicles. As presented in our previous work [25], we achieve fast convergence in our training using these techniques and a two-stage training methodology. We propose a first training stage, with a large number of episodes in a simple simulator, and a second training stage with the previous prior, in charge of adapting the model to a more complex and realistic simulator. We solve the domain adaptation problem by keeping the same inputs and outputs. In the first training stage, vehicle dynamics, velocity commands and maneuvers are defined by the SUMO simulator, while in this second stage, this is done by CARLA simulator.

B. Evaluation Metrics

In our proposed methodology, the metrics of success and collision rates and average passing time are utilized as general indicators for the assessment of performance. Specifically, the success rate serves as a direct measure of the effectiveness of the RL agent in accomplishing the designated task. These metrics are defined as:

- $success\ [\%] = end\ reached/n_e$
- $collision\ [\%] = n_{collisions}/n_e$
- $t_{avg} = \sum t_n\ /\ n_e$

where the number of episodes n_e is defined for each evaluation and simulation time is measured in seconds. The

TABLE I. Results of the previously described decision-making architectures for single use cases. The success rate S[%], collision rate C[%], timeout rate T[%] and episode time t_e are presented.

Scenario	Metric	Time-To-Collision	Petri Nets	Hybrid RL
Crossroad	S [%]	73	80	91
	C [%]	1	3	1
	T [%]	26	17	8
	t_e (s)	19	18	8
Merge	S [%]	90	95	95
	C [%]	0	0	0
	T [%]	10	5	5
	t_e (s)	10	10	7
Roundabout	S [%]	78	87	94
	C [%]	2	1	1
	T [%]	20	12	5
	t_e (s)	17	12	8
Lane Change	S [%]	96	98	98
	C [%]	3	1	0
	T [%]	1	1	2
	t_e (s)	53	51	25

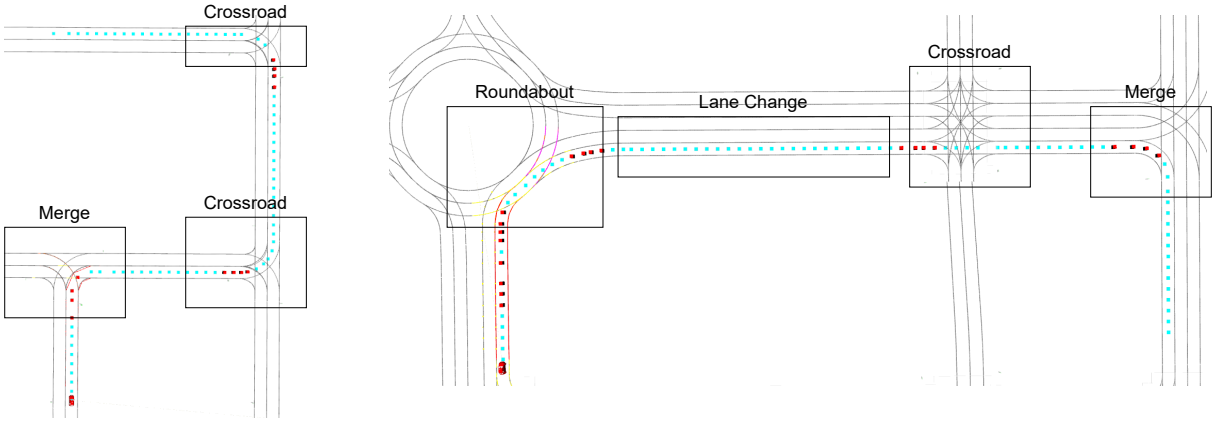


Fig. 6. A representation of the *Town 01* (left) and *Town 03* (right) concatenated scenarios.

episode time is limited. The ego vehicle does not succeed if this time is reached.

C. Results for individual use cases

In this section, we present the results of 100 test episodes for each of the previously described algorithms and scenarios. After a determined time for each scenario, the episode ends. If the agent is not able to solve the use case this episode finishes with a timeout.

The results obtained for each architecture in each scenario described in the previous section are presented in Table I. As expected, all three architectures obtain worse results in the crossroad scenario, which involves more vehicles and situations. Our Hybrid RL agent obtains the best results with an average success rate of 0.945. Besides, our proposal solves the scenario significantly faster than the other approaches. The percentage of collisions of the three approaches is similar, being the main difference the number of tests that end with a timeout.

D. Concatenated scenarios.

One of the main contributions of this work is the concatenation of use cases. We define adversarial vehicles in determined positions. These vehicles start moving at a certain moment so they present a difficult situation to the ego vehicle. In order to add complexity to these scenarios, the initial position of the adversaries and their velocities will vary from one simulation to another.

We develop two scenarios with multiple use cases using the OpenScenario tool [26]. These scenarios, described in Fig. 6, are simulated in two towns in CARLA:

- *Town 03* scenario presents the following concatenated use cases: a roundabout with one adversary, a lane change with a stopped vehicle in front, a crossroad with some adversaries approaching for both lanes and a merge with two adversaries on the main road.
- *Town 01* scenario presents the following concatenated scenarios: merge with two adversaries on the main road and two crossroads where the ego vehicle have to turn left with some adversaries.

The behavior of our framework in the *Town 03* scenario is described in Fig 7, where a temporal diagram is shown. In this diagram, both the velocity and steer commands of the ego vehicle are represented. Each colour refers to a specific use case. These use cases are shown in the images under the chart, where their trajectories are represented.

The RL agents are selected using the HD map information. These agents select the actions and the low-level execute these actions. These high-level actions are also described in this temporal diagram.

Finally, we run 100 tests with our architecture in the concatenated scenarios and present the results obtained in Table II.

TABLE II. Results of Hybrid RL architecture for concatenated use cases. The success rate S[%] and episode time t_e are presented.

	Town 01		***Town 03***	
	$S[\%]$	t_e	$S[\%]$	t_e
Hybrid RL	98	33	97	46

These scenarios are available for the scientific community through our GitHub: https://github.com/RobeSafe-UAH/HybridDecisionMaking. Besides, a video of the agent performing some simulations can be found.

VI. Conclusions and Future Works

The results of this work show that a hybrid architecture can solve driving decisions in concatenated complex urban scenarios. With the use of HD map information and Reinforcement Learning algorithms, high-level decisions are executed by a low-level controller. Besides, the proposed framework obtains better results in terms of success and execution time than the classical approaches.

In the near future, we plan to design more of these scenarios and study the influence of simulated sensors in these complex scenarios. This will lead us to transfer these models to our autonomous vehicle in controlled real scenarios.

References

[1] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep learning sensor fusion for autonomous vehicle perception and localization: A review," *Sensors*, vol. 20, no. 15, 2020.

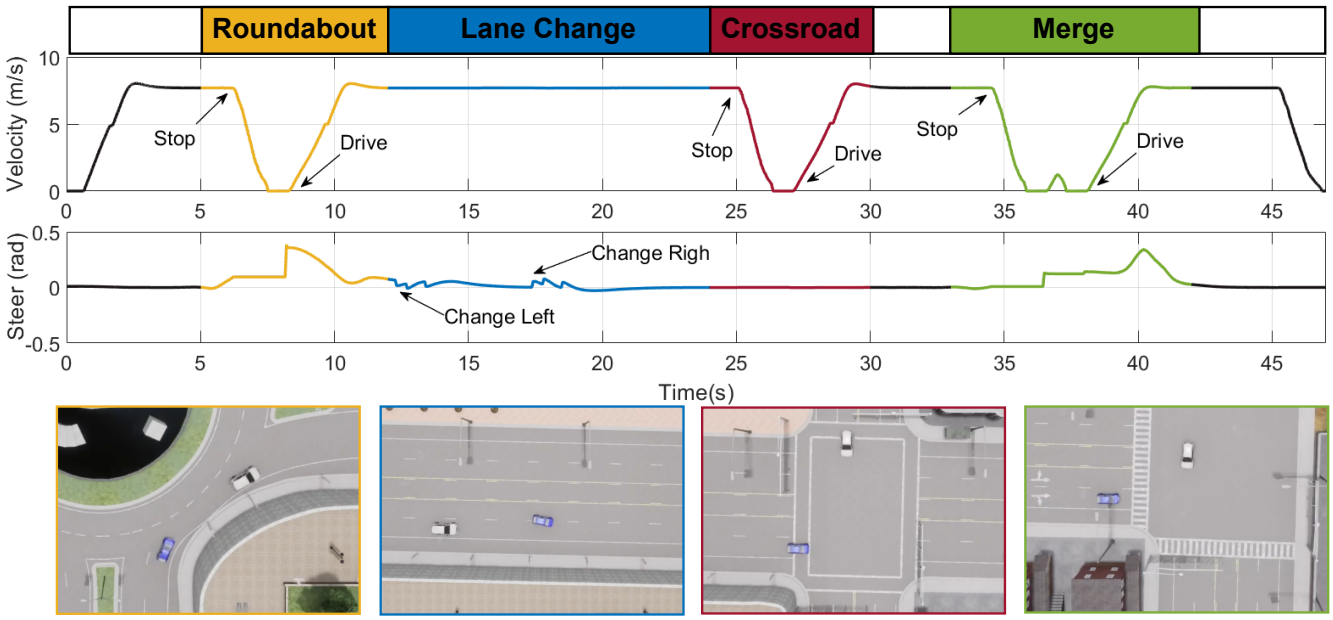


Fig. 7. A concatenated scenario temporal diagram. The behaviour of our agent is shown in the sequence of images in the CARLA simulator. The relevant events are described. The evolution of the velocity and steer are presented in the temporal chart. Ego vehicle in blue and adversary in white.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. K. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *CoRR*, vol. abs/2002.00444, 2020.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[4] Q. Liu, X. Li, S. Yuan, and Z. Li, "Decision-making technology for autonomous vehicles learning-based methods, applications and future outlook," 2021.

[5] A. Artuñedo, J. Godoy, and J. Villagra, "A decision-making architecture for automated driving without detailed prior maps," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1645–1652, 2019.

[6] P. F. Orzechowski, C. Burger, and M. Lauer, "Decision-making for automated vehicles using a hierarchical behavior-based arbitration scheme," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, oct 2020.

[7] D. Yang, K. Redmill, and U. Ozguner, "A multi-state social force based framework for vehicle-pedestrian interaction in uncontrolled pedestrian crossing scenarios," 2020.

[8] D. Isele, "Interactive decision making for autonomous vehicles in dense traffic," 2019.

[9] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1584–1589, 2017.

[10] D. Iberraken, L. Adouane, and D. Denis, "Safe autonomous overtaking maneuver based on inter-vehicular distance prediction and multi-level bayesian decision-making," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3259–3265, 2018.

[11] C. Vallon, Z. Ercan, A. Carvalho, and F. Borrelli, "A machine learning approach for personalized autonomous lane change initiation and control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1590–1595, 2017.

[12] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," *CoRR*, vol. abs/1807.00412, 2018.

[13] T. Tram, I. Batkovic, M. Ali, and J. Sjöberg, "Learning when to drive in intersections by combining reinforcement learning and model predictive control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3263–3268, 2019.

[14] C. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," *CoRR*, vol. abs/1803.10056, 2018.

[15] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2156–2162, 2018.

[16] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," *CoRR*, vol. abs/2004.04450, 2020.

[17] K. Shu, H. Yu, X. Chen, L. Chen, Q. Wang, L. Li, and D. Cao, "Autonomous driving at intersections: A critical-turning-point approach for left turns," *CoRR*, vol. abs/2003.02409, 2020.

[18] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, M. U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," *CoRR*, vol. abs/1909.11538, 2019.

[19] F. Ye, X. Cheng, P. Wang, and C. Chan, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," *CoRR*, vol. abs/2002.02667, 2020.

[20] C. Gómez-Huélamo, J. Del Egido, L. M. Bergasa, R. Barea, E. López-Guillén, F. Arango, J. Araluce, and J. López, "Train here, drive there: Ros based end-to-end autonomous-driving pipeline validation in carla simulator using the nhtsa typology," *Multimedia Tools and Applications*, pp. 1–28, 2021.

[21] A. Diaz-Diaz, M. Ocaña, A. Llamazares, C. Gómez-Huélamo, P. Revenga, and L. M. Bergasa, "Hd maps: Exploiting opendrive potential for path planning and map monitoring," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022.

[22] R. Gutiérrez, E. López-Guillén, L. M. Bergasa, R. Barea, Pérez, C. Gómez Huélamo, J. F. Arango, J. del Egido, and J. López, "A waypoint tracking controller for autonomous road vehicles using ros framework," *Sensors*, vol. 20, p. 4062, 07 2020.

[23] N. Abdeselam Abdel-Lah *et al.*, "Controlador híbrido basado en mpc y "splines" para maniobras de cambio de carril en vehículos autónomos," 2022.

[24] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, pp. 1805–1824, aug 2000.

[25] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, J. Araluce, and L. M. Bergasa, "Reinforcement learning-based autonomous driving at intersections in carla simulator," *Sensors*, vol. 22, no. 21, 2022.

[26] J.-M. Jullien, C. Martel, L. Vignollet, and M. Wentland, "Openscenario: A flexible integrated environment to develop educational activities based on pedagogical scenarios," in *2009 Ninth IEEE International Conference on Advanced Learning Technologies*, pp. 509–513, 2009.