Received April 29, 2022, accepted May 11, 2022, date of publication May 16, 2022, date of current version May 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3175493

Learning From Oracle Demonstrations–A New Approach to Develop Autonomous Intersection Management Control Algorithms Based on Multiagent Deep Reinforcement Learning

ANTONIO GUILLEN-PEREZ AND MARIA-DOLORES CANO, (Senior Member, IEEE)
Department of Information and Communication Technologies, Universidad Politécnica de Cartagena, 30202 Cartagena, Spain

Corresponding author: Antonio Guillen-Perez (antonio.guillen@edu.upct.es)

This work was supported in part by the MCIN/AEI/10.13039/501100011033 under Grant PID2020-116329GB-C22; and in part by the Fundación Séneca, Región de Murcia, Spain, under Grant 20740/FPI/18.

ABSTRACT Worldwide, many companies are working towards safe and innovative control systems for Autonomous Vehicles (AVs). A key component is Autonomous Intersection Management (AIM) systems, which operate at the level of traffic intersections and manage the right-of-way for AVs, thereby improving flow and safety. AIM traditionally uses control policies based on simple rules. However, Deep Reinforcement Learning (DRL) can provide advanced control policies with the advantage of proactively reacting and forecasting hazardous situations. The main drawback of DRL is the training time, which is fast in simple tasks but not negligible when addressing real-world problems with multiple agents. Learning from Demonstrations (LfD) emerged to solve this problem, significantly speeding up training, and reducing the exploration problem. The challenge is that LfD requires an expert to extract new demonstrations. Therefore, in this paper, we propose the use of an agent, previously trained by imitation learning, to act as an expert to leverage LfD. We named this new agent *Oracle*, and our new approach was called Learning from *Oracle* Demonstrations (LfOD). We implemented this novel method over the DRL TD3 algorithm, incorporating significant changes to TD3 that allowed the use of *Oracle* demonstrations. The complete version was called TD3fOD. The results obtained in the AIM training scenario showed that TD3fOD notably improves the learning process compared with TD3 and DDPGfD, speeding up learning to 5–6 times, while the policy found offered both significantly lower variance and better learning ability. The testing scenario also showed a significant improvement in multiple key performance metrics compared with other vehicle control techniques on AIM, such as reducing waiting time by more than 90% and significantly decreasing fuel or electricity consumption and emissions, highlighting the benefits of LfOD.

INDEX TERMS Autonomous intersection management, intelligent transport systems, intersection traffic management, learning from demonstrations, multi-agent deep reinforcement learning.

I. INTRODUCTION

Deep Reinforcement Learning (DRL) has demonstrated a remarkable ability to solve several complex real-world sequential decision-making problems [1]–[6]. However, this success is currently limited due to the extensive training process required by traditional DRL algorithms, which requires days to months of training and tens or hundreds of graphics cards in parallel [7], [8].

Autonomous Vehicles (AVs) control is a field where DRL has been extensively studied. This field achieves perfect symbiosis when combined with computer simulators, making them an excellent framework for new advanced control systems. The management of these AVs in cities should be performed collectively with centralized information to be as efficient as possible. This centralized control would be an intelligent system capable of controlling all AVs simultaneously, in real-time, and ensures a high degree of security.

The study of Autonomous Intersection Management (AIM) [9], [10] began a few years ago, even before the large-scale deployment of the first AVs. However, these AIM

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Tang.

based their operations on simple rules, unable to achieve advanced control policies that could obtain truly intelligent and proactive behavior. This was the scenario until the development of RAIM [11], an AIM that is based on DRL along with other advanced deep learning techniques. However, the main problem with this approach is that it requires many interactions with the environment to achieve a good performance.

Recently, researchers have been studying new alternatives that can speed up the training of DRL algorithms, either by imitating a behavior (Imitation Learning, IL [12]), imitating some observations (Imitation from Observation, IfO [13]), or using an initial training phase of supervised learning of demonstrations (tuples of {*state, action, new_state, reward*}) offered by an expert to progressively train the pre-trained policy using DRL (Learning from Demonstrations, LfD [14]). Several previous efforts have demonstrated that LfD can significantly accelerate the training of DRL algorithms in environments where there are expert demonstrations [14]–[16]. However, although we consider that LfD has the potential for improvement, there are environments where the demonstrations offered by an expert cannot be extracted (totally or partially), such as in traffic simulators used to train new AV control systems. (e.g., SUMO [17]). In these simulators, each vehicle has its own internally modeled controller, and it is not possible to extract demonstrations from each vehicle in a straightforward manner.

Owing to the above limitations, this work proposed a new approach within the field of LfD that used an IL-trained agent to model hidden expert controllers. This agent allowed us to extract the knowledge of a hidden expert (new experiences) and to ask about each state, what action the hidden expert would have taken. In other words, the IL-trained agent could ask for each state: "What would my expert say I should do in this state?" In this way, in environments where there is no expert (or one is hidden) from which to extract the demonstrations, we could train an agent that imitated the hidden expert behavior and could be considered as such to leverage LfD to further train another agent via DRL. We called this new agent *Oracle*. To enable the use of demonstrations offered by the *Oracle*, we proposed several modifications to the DRL algorithm used, TD3: *i*) a modification to the error equation that updates the TD3 actor so that the error produced by the RL action was gradually considered; *ii)* the introduction of several parameters for a smooth and progressive transition between LfD and RL (τ_1 and τ_2); and *iii)* the use of two replay buffers, one for demonstrations to train *Oracle* and the other for TD3, in addition to the use of Prioritized Experience Replay (PER) to accelerate learning. Following the nomenclature used in previous works, we called this new approach *Learning from Oracle Demonstrations* (LfOD), and the subsequent DRL algorithm was called TD3fOD. TD3fOD was used to train an AIM algorithm previously proposed in an autonomous vehicle traffic scenario [11]. The results showed a notable improvement over not using LfD, speeding up by 5–6 times, and noticeably reducing the variance in the policy obtained. The results compared with DDPGfD showed that the use of the *Oracle* allowed to triple the training speed and considerably reduce the variance in the control behavior during the training phase. Finally, in a testing scenario, the AIM algorithm trained with TD3fOD was compared with other autonomous vehicle control algorithms, and the results showed an improvement in all the studied metrics, reducing the waiting time by approximately 95%, among other factors.

Thanks to our proposal, it is possible to extract the hidden agent (learned by IL by the *Oracle*) in those simulators where it is not possible (or too complicated), to take advantage of the benefits offered by LfD (training acceleration and more robust policies) for the development of new complex control algorithms.

The primary motivation of this work lies in the development of a new algorithm to accelerate agent training using DRL and LfD, which can be applied to any problem where there is no agent from which to extract demonstrations. Furthermore, the development of advanced cooperative control systems for AVs and Multi-Agent DRL-based systems can be speeded up based on the contributions of this work.

Therefore, the main contributions of this work are:

(*i*) Propose a new LfD approach that can be used in environments where there are no experts from which to extract demonstrations, taking advantage of hidden agent demonstrations.

(ii) Demonstrate that using the *Oracle* in LfOD speeds up the training of DRL algorithms, reducing the training time and variance in trained policies.

(iii) Development of a new AIM trained with the proposed algorithm (RAIM with LfOD), capable of improving the performance of AIM algorithms in a cooperative autonomous vehicle control scenario.

The rest of the article is organized as follows. Section II provides the background of DRL, IL, and LfD. A review of previous related work on both AIM and LfD is discussed in Section III. Section IV details our proposal, TD3fOD. The experimental setup, simulation scenarios, and the explanation of the modifications made to TD3, and AIM are included in Section V. Section VI shows the results obtained in both the training and the testing scenarios. Finally, Section VII concludes the paper.

II. BACKGROUND

This section explains DRL, Multi-Agent DRL (MADRL), and the algorithm modified in this work, TD3. In addition, we detail how IL works, and, finally, we explain the basics of LfD.

A. DEEP REINFORCEMENT LEARNING

Reinforcement Learning (RL) is an area of machine learning in which an agent learns to complete a task in an environment where it can take an action and receive a reward for the action. The agent's goal is to find a policy that performs actions that maximize the rewards accumulated during the entire task, which are known as expected discounted total rewards.

The environment where the agent is located is usually modeled by a Markov Decision Process (MDP) because many RL algorithms employ dynamic programming techniques to solve these MDP. An MDP is defined by the tuple $\langle S, A, R, T, \gamma \rangle$, where S represents a set of states of an environment, A represents the set of actions that the agent can take, T is the transition function $T : S \times A \times S \rightarrow [0, 1]$ that determines the transition probability from any state $s \in S$ to any state $s' \in S$ when the action $a \in A$ is taken. R is the reward function $R : S \times A \times S \rightarrow \mathbb{R}$ and $\gamma \in [0, 1]$ represents the discount factor that adjusts the trade-off between immediate and future rewards.

Resolving an MDP generates a policy $\pi : S \rightarrow A$, which maps states $s \in S$ to actions $a \in A$. An optimal policy π^* maximizes the expected total discounted reward for all the states. This approach to finding the optimal policy can be formulated using the state-action value function (*Q-function*): $Q^{\pi}(s, a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$. This *Q-function* determines the expected reward by starting from the state s, taking the action a, and following the policy π.

If we focus on DRL, the introduction of Neural Networks (NN) in traditional RL algorithms has been a great revolution, considerably speeding up the learning of these algorithms and allowing them to be applied to tasks that seemed impossible before, because NNs can act as approximating functions of the policy to be learned.

Within DRL, there are several approaches; however, the approach that has attracted the most attention in recent years is based on actor-critics for continuous control problems. Both the actors (a policy that decides what action to take for each state) and the critic (given a state and an action, it gets what expected reward, or *Q-value*, is obtained, indicating to the actor whether the action is going to be good or not) are modeled by NNs.

B. MULTI-AGENT DEEP REINFORCEMENT LEARNING

MADRL [18] is a subset of RL problems in which multiple agents interact with each other and their environment, each of which attempts to learn a policy and learn to collaborate/compete, depending on the task.

Within MADRL, there are two different learning approaches depending on the problem to be solved: *cooperative* multi-agent learning [19], in which agents cooperate to maximize the total cumulative reward; and *competitive* multi-agent learning [20], in which agents compete with each other to obtain the highest possible reward individually (or from the group they belong to).

To train and execute these algorithms, different techniques have been developed that take advantage of the benefits offered by collective learning. One of these is centralized training and decentralized execution [21]. Training is performed centrally in an environment where each agent sends its information, and the control policies of each agent are obtained. Each agent then obtains the policy and executes it decentralized. Another approach is centralized training and execution [22]. In this case, agents are trained centrally and then executed individually. Last, we have decentralized training and decentralized execution. In this case, agents are trained in a decentralized manner and execute policies individually. The benefit of centralized training is better knowledge of the entire environment, so the policy is found more quickly and robustly. However, it is not always possible to centralize knowledge for training because of communication limitations; thus, decentralized training is required. In decentralized training, each agent only has local knowledge for training. Thus, each agent updates its policy individually while sharing its policy with other agents. Finally, decentralized execution means that agents are executed on a decentralized controller individually while considering all agents simultaneously.

C. TD3

Our method combines demonstrations extracted from an *Oracle* to exploit this knowledge and accelerate the training of a new control policy using the DRL TD3 algorithm [23]. TD3 is a DRL algorithm whose acronym stands for Twin Delayed Deep Deterministic Policy Gradient. Therefore, we can get a new way to develop AIM systems in a fast and easy way.

TD3 is one of the most powerful and advanced off-policy model-free DRL algorithms used for continuous tasks. TD3 is an evolution of the Deep Deterministic Policy Gradient (DDPG [24]) DRL algorithm where several key improvements are added specifically, Clipped Double-Q Learning, Target Policy Smoothing, and "Delayed" Policy Updates [23], [25]. As many works have shown, TD3 offers fast convergence for complex tasks that require continuous control [23], [26]. For this reason, we decided to implement our proposal on TD3 and not on other algorithms such as SAC [27], PPO [28], or A3C [29].

1) CLIPPED DOUBLE-Q LEARNING

Instead of using only one critic network, TD3 adds a second critic network that reduces the estimation bias by selecting the smallest *Q-value* of the two critic networks, encouraging underestimation of *Q-values*. This underestimation bias is not a problem since low values will not propagate through the algorithm, unlike overestimated values. Thus, it provides a more stable approximation and improves the stability of the whole algorithm.

2) TARGET POLICY SMOOTHING

To reduce the overfitting produced by high variance target values when updating the critics, TD3 adds a small noise to each selected action. In addition, it performs double clipping, first on the aggregated noise and then on the noisy action. This reduces the variance of the selected actions and results in more stable *Q-values*.

3) DELAYED POLICY UPDATED

TD3 updates the policy and target networks less frequently than *Q-functions*, providing more stable and efficient training. The original paper [23] suggested updating the policy and target networks every two updates of the *Q-functions*. In addition, the policy network π_θ is updated with a gradient ascent step by simply maximizing Q_{ϕ_1} as shown in (1).

$$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} Q_{\phi_1} (s, \pi_\theta (s)) \quad (1)$$

D. IMITATION LEARNING

IL arises from the need to train an agent more efficiently than through RL alone. With IL, an agent learns a policy by ''imitating'' an expert who knows what action to take for each state. Through Supervised Learning (SL), IL is more effective than RL when an expert demonstrates the desired behavior and thus teaches a policy [30]. Besides, the problem of reward shaping is eliminated, where it is necessary to carefully select the reward received by the agent or design a hand-coded function that changes smoothly to achieve a stable and consistent policy. Within IL, there are several alternatives:

1) BEHAVIORAL CLONING

Behavioral Cloning (BC) [12], [31], [32] is the simplest form of IL, where a policy is trained based on expert demonstrations; that is, there is an expert agent who is capable of producing pairs of state-action demonstrations. These demonstrations are used in traditional SL and can provide a policy that achieves a behavior that clones the expert.

BC can work excellently for some applications in which the entire state-action space is explored. However, BC is problematic in most cases. The main concern is that SL assumes that the samples used are i.i.d. (independent and identically distributed); however, that assumption cannot be guaranteed due to the nature of the samples' capturing process.

Furthermore, when a trained agent takes control, it can make mistakes in predicting actions that can lead to states never seen before during the expert-supervised training. In these states, the agent's behavior can lead to hazardous situations known as compounding errors [12], from which the agent can never recover. An example of compounding errors is depicted in Fig. 1. Several alternatives have been proposed to obtain more samples online while testing the trained policies and resolving the problems presented by BC.

2) DIRECT POLICY LEARNING VIA INTERACTIVE DEMONSTRATOR

Direct Policy Learning (DPL) is an enhanced version of BC. This is an iterative process in which expert feedback is collected during the training loop. The entire process starts by collecting demonstrations from the expert, who serves to train the agent. After the first training, the trained policy is rolled out and the new states that are visited are stored. Then, the expert is asked what actions it would take in those new states, collecting new demonstrations. These new demonstrations (feedback) allow us to obtain more data to train the agent again using SL. This loop continues until converging

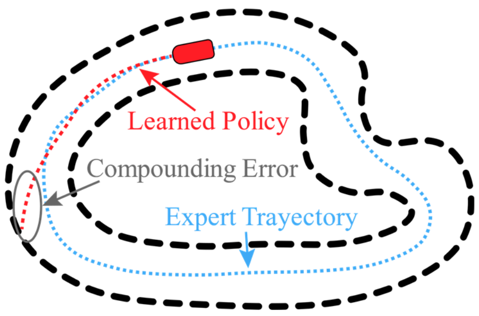


FIGURE 1. Compounding errors example.

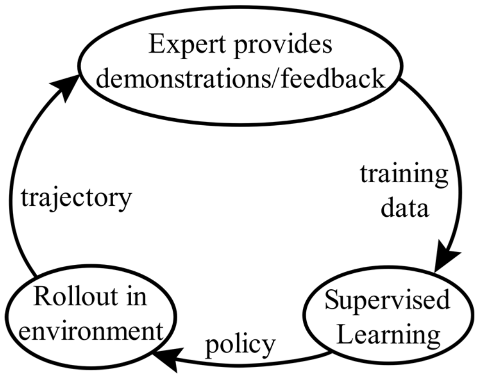


FIGURE 2. The general direct policy learning (DPL) algorithm.

(see Fig. 2). It is important to store and use all the collected demonstrations to remember the mistakes made by the agent in the past.

Within this group of IL, there are several powerful algorithms, most notably: SEARN (Search-based Structured Prediction). [33], SMILe (Stochastic Mixing Iterative Learning) [30], and DAGGer (Data AGGregation) [12].

However, these methods have a significant drawback. They only use SL to obtain behaviors similar to those of the expert and do not employ any RL technique to obtain superior behavior. In addition, they need an online expert from which to obtain feedback about the states visited during rollout.

E. LEARNING FROM DEMONSTRATION

Learning from Demonstrations (LfD) was introduced to overcome the limitations of DPL. LfD appeared in DeepMind's Deep Q-Learning from Demonstrations (DQfD) work [14], elegantly unifying IL and RL.

LfD employs an expert's experiences (demonstrations, which can be potentially suboptimal) to pre-train an agent through SL and then uses RL algorithms to improve the learned policy. However, using SL in the demonstration data and applying RL in the pre-trained policy is not ideal. LfD employs the demonstration data throughout the training process. Thus, the agent finds a policy that eventually surpasses the expert policy. In summary, LfD allows the initialization (pre-train) of an agent through expert demonstrations and then uses RL to discover a better policy by interacting with the environment and using the previous expert demonstrations to not forget the base policy. What differentiates IL from LfD is that the latter only has expert demonstrations, a sequence of (s_t, a_t, r_t, s_{t+1}) and not an online expert to obtain feedback, as well as the training stage via RL to improve the learned policy.

LfD was presented using Deep Q-Network (DQN) algorithm. DQN is used to control agents in discrete space actions, and the transition from a discrete space action to a continuous one is not trivial. A modification of DDPG that allowed the use of demonstrations for its training was introduced in [15], namely, DDPGfD.

DDPGfD stores both the demonstrations collected from the expert and the experiences collected by the agent in a replay buffer. In addition, it proposes a set of improvements such as the use of a mix of 1-step and n-step return losses, learning multiple times per environment step, and L2 regularization losses. However, DQfD and DDPGfD have the main drawback of misadjusting the internal parameters of the agent obtained during pre-training when the agent begins to take control, which can lead to forgetting everything that was prelearned.

III. RELATED WORKS

This section summarizes works from the related literature that address AIM and LfD.

A. AUTONOMOUS INTERSECTION MANAGEMENT

AIM has emerged as an alternative control method for AVs at traffic-light-regulated intersections. In [34], Dresner and Stone proposed the first AIM that regulated the crossing of AVs at intersections using a reservation-based method following a "*First Come, First Served*" (FCFS) policy and eliminating traffic lights. This policy worked as follows: when a vehicle approached the intersection, it requested to reserve the space-time the vehicle needed to cross the intersection. If the reservation did not conflict with another vehicle's reservation, the intersection accepted it, and the vehicle followed the route it had requested. Otherwise, the vehicle received a reservation denial and slowed down to request another reservation later in searching for available space-time slots.

The first results showed that FCFS could outperform traffic light control in terms of flow and delay. Later studies [35], [36] proposed alternative control protocols that included non-autonomous vehicles (FCFS-LIGHT) and emergency vehicles (FCFS-EMERG). The authors also proposed a mechanism to switch among policies (FCFS, FCFS-LIGHT, and FCFS-EMERG) depending on intersection conditions, improving performance by using the policy that best suited each situation [37]. Their results outperformed traditional traffic light control.

A more detailed study of the FCFS protocol was proposed in [38], where it was tested against an optimized traditional traffic light signal. The results showed that FCFS reduced the delay against the traditional traffic light signal by more than 90%. An improvement of FCFS also based on reservations was proposed by Huang *et al.* [39]. They suggested that when the intersection sent the denial of reservation, it also sent a recommended deceleration speed to reach the stop line when the vehicle stopped. Furthermore, this algorithm separated the vehicles into three groups according to their current and past status. The proposed algorithm was compared to a roundabout and a traffic light, but not to the original FCFS. The results showed a reduction in delay of 85%, and a reduction in fuel consumption of 50%.

Because FCFS does not consider any mechanism for grouping (batching) requests that have the same direction, several enhancements have been proposed [37], [40] where batching of requests was used to improve the flow of the intersection, either by having more requests to make smarter decisions or by allowing vehicles in the same flow to pass in batches. The results showed an improvement in both FCFS and traffic light control, doubling the flow and reducing the delay by 85%.

Other approaches to AVs control use mathematical optimization to obtain right-of-way [41]–[45]. The results achieved by these proposals were similar to the previously shown algorithms. However, due to the solving characteristics of these algorithms, the resolution complexity increases significantly when the number of vehicles increases. Consequently, these algorithms face a sizeable computational complexity problem, making them unfeasible for the real-time control of AIM. Using this approach, we can find the work of Wu *et al.* [46]. In this work, the authors allowed all movements in all lanes and developed two modules, one in charge of deciding the temporal instant at which the vehicle should enter the intersection and other in charge of deciding in which entry lane the vehicle should be placed and which exit lane it should take. This work followed a Mixed-Integer Linear Programming (MILP) problem approach to solve the proposed set of equations and constraints. The results showed the goodness offered by the AIMs; however, due to the approach followed, there are many open problems. A heuristic approach was followed in [47]. In this case, the authors used this approach to resolve spatiotemporal conflicts in AVs. They followed an approach that modeled the conflict points within the intersection as points of interest. The SUMO tool was used to simulate vehicle behavior. The results showed that the proposed system offered a shorter vehicle waiting time than other IM schemes and traffic light-based systems. Additionally, there are other novel algorithms motivated by different fields, such as those inspired by auctions [48], those that use ant colony-based optimization [49], or those that use a Monte Carlo tree search to obtain the order of priority to be assigned to vehicles [50].

Although previous studies showed promising results, Levin *et al.* [51] demonstrated that further study of the proposed algorithms is necessary because, under certain situations, FCFS may present inappropriate behaviors that can lead to inappropriate results. Control policies require a detailed and in-depth study before they become operational in real control systems. Furthermore, as can be seen, all the proposed algorithms are based on simple approaches, which do not analyze the past or future behavior of the intersection or consider the consequences of actions taken in future states. One approach that may be considered for this purpose is the use of RL for vehicle control.

If we focus on RL, very few works have applied this technique in AIM, although it has been widely studied to address traffic light control [52]–[58]. The work proposed by Wu *et al.* [59] is interesting. Their proposal calculates the priority order to be assigned to each vehicle using Multi-Agent RL (MARL). The results compared with FCFS and with a variant proposed by the same authors, namely Longest-Queue-First (LQF), showed that the use of MARL could obtain a sequence of decisions that reduced the delay by more than 60%.

An approach based on DRL can be found in [60]. In this case, the authors employed an ego-centric policy trained by RL and attention learning mechanisms to develop an intersection control system. The results showed that the policy outperformed other control systems under different traffic conditions. However, this approach leaves the control to each vehicle individually, and therefore, it cannot exploit all the advantages that AIMs can offer, i.e., the benefits of centralizing the knowledge in a centralized agent of all AVs. Another work that applied DRL can be found in [61]. In this case, the proposal models different types of AVs with different behaviors and through a game model based on cognitive hierarchy, allows the AVs to adapt to the reactions of the other AVs. Although these results are promising, it is necessary to study the performance of the proposed solution in more complex environments with more lanes and a higher vehicular flow.

In our humble opinion, it makes perfect sense to use RL to control AVs at intersections. Using RL, the system can learn and gain in-depth knowledge of AV control through trial-and-error. In addition, we expect that RL will provide a safer and faster solution that helps overcome the limitations of existing AIM algorithms.

B. LEARNING FROM DEMONSTRATIONS

RL allows solving complex problems and provides advanced control policies. Although there are many techniques for agent optimization, one technique called Learning from Demonstrations (LfD) has generated significant interest in recent years. This technique allows the pre-training of a policy quickly utilizing demonstrations of an expert and later applies RL to find another policy that improves the expert's policy, as described in [14]. In that work, the DQN algorithm was adapted to incorporate expert demonstrations. The results showed a notable boost in speed up training, allowing to fulfill the tasks of Atari Games much earlier and finding a better policy than those offered by human demonstrations.

DDPG from Demonstrations (DDPGfD) was proposed in [15] to control agents in continuous action spaces by incorporating demonstrations. The demonstrations and actions performed by the agent were stored in a replay buffer for an unlimited time. The results showed the benefits of LfD: the obtained policy performed the tasks more efficiently than demonstrations and solved the tasks between 2-4 × steps less than DDPG.

Another approach proposed over DDPG was presented by Nair *et al.* [16], where they used the Hindsight Experience Replay (HER) as a replay buffer sampling method. They obtained an order of magnitude improvement in speed up over RL on simulated robotic tasks. Finally, Jing *et al.* [62] investigated the problem of learning with imperfect expert data. The results of their proposal revealed the rapid learning of a control policy that improved on other proposed methods.

All of these learning techniques rely on the existence of an expert agent that can be used to obtain demonstrations (states and actions). For the design of new AVs control systems, there are numerous simulators, such as SUMO [17], CARLA [63], or VISSIM [64]. However, due to the inherent design of these simulators, there is no trained expert agent from which to obtain demonstrations. Therefore, we decided to investigate the opportunity to train an agent that we can ask (*Oracle*) while training a controller (through TD3) to control AVs. As a result, we propose in this paper a new method called TD3 from *Oracle* Demonstrations (TD3fOD).

IV. TD3 FROM ORACLE DEMONSTRATIONS–TD3FOD

Our method combined TD3 with demonstrations extracted from an expert (*Oracle*). The *Oracle* is trained by BC to continuously obtain demonstrations, optimizing the extracted knowledge and improving and speeding up the learning process. In this section, we describe our method and evaluate these insights through our experiments.

Our algorithm presents as a novelty the *Oracle* from which to obtain new demonstrations. This *Oracle* is trained using the collected experiences of the expert and modifies the parameters of π_θ (TD3 actor) using *soft_update* (soft-copy of the parameters). This *soft_update* is inspired by that used by Mnih *et al.* [2]. In this case, the weights of π_θ network (θ^π_{actor}) are updated as depicted in (2).

$$\theta^\pi_{actor} = \tau_1 \cdot \theta^\pi_{oracle} + (1-\tau_1) \cdot \theta^\pi_{actor} \quad with\; 0 < \tau_1 < 1 \tag{2}$$

By employing *soft_update*, we force the actor to learn more slowly than the *Oracle*, thereby increasing the stability of the training. To adjust the importance that the *Oracle* has on the actor, we assume that the parameter τ_1 decreases smoothly along the simulations following (3).

$$\tau_1 = sigmoid\left(-\frac{sim - th}{th/5}\right) \tag{3}$$

The *th* parameter adjusts the smoothness and the number of simulations from which we consider that the learning through RL has more importance than the learning done by the *soft_update* of the *Oracle*.

As can be seen in (3), at the beginning of the training ($sim \in 1, 2, \ldots, N_{simulations}$), the parameters of π_θ will be very similar to *Oracle* (being τ_1 practically 1). However,

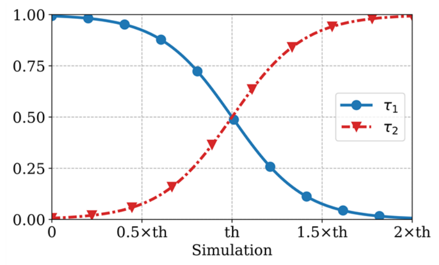


FIGURE 3. Evolution of τ_1 and τ_2 in function of *th* parameter.

the importance that *Oracle* has in the changes to π_θ is progressively reduced until the number of simulations (*sim*) $\gg$ *th*, where τ_1 is practically equivalent to 0 and cancels the first term of (3), canceling the changes in π_θ due to *Oracle soft_update*. This evolution is illustrated in Fig. 3.

In addition, our algorithm improves TD3 in several aspects:

1) The error equation is modified so that the importance of the error produced by the RL actions increases progressively. More specifically, we modified (1) (which was used to update π_θ), and the new form is shown in (4), where the factor τ_2 controls the relevance of Q_{ϕ_1} (*Q-values* of critic Q1) over the update of π_θ. τ_2 is given by (5). The evolution of both τ_1 and τ_2 throughout the simulations are shown in Fig. 3.

$$\nabla_\theta \frac{1}{\mathcal{B}} \sum_{s \in \mathcal{B}} \tau_2 Q_{\phi_1}(s, \pi_\theta(s)) \tag{4}$$

$$\tau_2 = 1 - \tau_1 \tag{5}$$

2) Incorporation of two replay buffers: one for Imitation (*Oracle* training) and one for RL. Moreover, these buffers use PER to speed up training by using the experiences from which each network can learn more information. Although PER is a technique used for RL, the original PER paper [65] suggests that it can also be employed in supervised learning.
3) The *Oracle* has been added to provide an expert agent to obtain new demonstrations because the simulator does not have this feature. This *Oracle* was trained using BC from the experiences extracted from the simulator. These experiences are stored in the Imitation replay buffer (with a fixed size). New experiences replace older ones when the buffer is full.
4) Finally, we add an exponential increase factor (β) that allows π_θ to control the vehicles spontaneously and in an incremental way; meaning that for each timestep, there is a probability that π_θ carries out the control of the vehicles instead of the simulator (expert). This probability increases smoothly over time until a simulation where π_θ always controls all vehicles. This operation offers more stability at the beginning of training and a gradual and smooth transition from BC to RL. Furthermore, because there is a small probability that the action will be taken by π_θ and not by the expert (being able to consider a kind of "*sticky actions*" or noise actions), the proposed procedure allows the *Oracle* to explore a large set of states at the start of the training, with all of the benefits that this can offer and reduce the compounding errors.

Algorithm 1 : *TD3fOD*

```
Input: env Environment; $\theta^{\pi}_{actor}$ initial actor policy parameters; $\theta^{\pi'}_{actor}$ initial actor policy target parameters.
Input: $\phi^{Q}_{1}$ initial Q1-function parameters; $\phi^{Q'}_{1}$ initial Q1-function target parameters.
Input: $\phi^{Q}_{2}$ initial Q2-function parameters; $\phi^{Q'}_{2}$ initial Q2-function target parameters.
Input: $\theta^{\pi}_{oracle}$ initial Oracle policy parameters; $\mathcal{D}^{imitation}$ Imitation replay buffer; $\mathcal{D}^{reinforcement}$ Reinforcement replay buffer; p coeff to expert decay;
  warmup_simulations number of simulations to pre-train actor, Q-functions, and Oracle; imitation_learn_every every each timestep run imitation_module;
  reinforcement_learn_every each timestep run reinforcement_module;
1  for simulation $sim \in \{1, \dots, N_{simulations}\}$ do:
2    # run simulation(sim):
3     $\beta = p^{sim-warmup\_simulations}$
4     for timestep $t \in \{1, \dots, Max_{episodie}\}$ do:
5       $expert\_control = True$ if $random(0,1) < \beta$ else $False$
6       obtain state $s_t$ $\forall$ agent
7       if expert_control then:
8         # Expert[1] selects the actions $a_t$ $\forall$ agent
9       else:
10        # Actor selects the actions $a_t$ $\forall$ agent
11        $a_t = \pi_\theta(s_t)$
12        $env.actions(a_t)$
13      end if
14      # Get next state and reward $\forall$ agent
15      $r_t, d_t, s_{t+1} \leftarrow env.step()$
16      if expert_control then:
17        # Obtain $a_t$ comparing $s_t$ and $s_{t+1}$ $\forall$ agent
18        # Store $(s_t, a_t)$ in $\mathcal{D}^{imitation}$
19      end if
20      # Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ in $\mathcal{D}^{reinforcement}$
21      if t mod $imitation\_learn\_every = 0$ then:
22        run imitation_module(sim)
23      end if
24      if t mod $reinforcement\_learn\_every = 0$ then:
25        run reinforcement_module(sim)
26      end if
27    end for
28  end for
```

The complete TD3fOD algorithm is divided into Algorithms 1, 2, and 3.

V. EXPERIMENTAL SETUP

Our algorithm focuses on simulators or environments in which there is no expert to ask or obtain feedback about past experiences (although it is possible to obtain new demonstrations).

[1]We can't ask to an expert to obtain the actions.

Algorithm 2 : Imitation Module

```
   # Run imitation module (sim):
1  for epoch $e \in \{1, \ldots, N_{imitation\_epochs}\}$ do:
2    for $mini\_batches$ $\mathcal{B}$ in $\mathcal{D}^{imitation}$ do:
3        $actions, states = unzip(\mathcal{B})$
4        $actions\_pred = \pi_{oracle}(states)$
5        # Update Oracle params ($\theta^{\pi}_{oracle}$) with one step of
         gradient descent.
6    end for
7  end for
8  # $soft\_update(\pi_{oracle}, \pi_{\theta}, \tau_1)$:
9  $\theta^{\pi}_{actor} = \tau_1 \cdot \theta^{\pi}_{actor} + (1 - \tau_1) \cdot \theta^{\pi}_{oracle}$
```

Algorithm 3 : *TD3* Reinforcement Module

```
    # Run reinforcement module (sim):
1   for epoch $e \in \{1, \ldots, N_{reinforcement\_epochs}\}$ do:
2     # Sample a $mini\_batch$ $\mathcal{B}$ from in $\mathcal{D}^{reinforcement}$
3     $s_t, a_t, r_t, s_{t+1}, d_t = unzip(\mathcal{B})$
4     $a'_t = \left(\pi'_{\theta}(s_t) + \epsilon\right).clamp(a_{min}, a_{max})$
5     # Update Q-functions parameters $\phi^{Q}_{1}$ and $\phi^{Q}_{2}$ as in TD3
6     if $e$ mod $update\_actor\_every = 0$ then:
7        $\tau_2 = 1 - \tau_1$
8        # Update actor policy with one step of gradient ascent
         with equation 4 and using $\tau_2$
9        # Update target networks as in TD3 and using $\tau_3$
10       # $soft\_update(\phi^{Q}_{1}, \phi^{Q'}_{1}, \tau_3)$
11       # $soft\_update(\phi^{Q}_{2}, \phi^{Q'}_{2}, \tau_3)$
12       # $soft\_update(\pi_{\theta}, \pi'_{\theta}, \tau_2)$
13    end if
14  end for
```

However, because the controller is internally modeled and cannot be accessed due to the nature of the simulator (lack of an API and/or closed-source software licensing), our algorithm exploits new demonstrations to build an expert to ask (*Oracle*) and then train a new control system by RL via LfD. By taking advantage of the benefits offered by LfD, training can be greatly accelerated and improved. For example, a simulator with these characteristics is SUMO [17]. SUMO is a microscopic traffic simulator where each vehicle is explicitly simulated and is widely used by the scientific community and urban planners to obtain better traffic controllers or optimize existing ones. Therefore, we decided to use this simulator in this study. TD3fOD was programmed in Python 3.7 and Pytorch 1.5.0. A 16-core processor was used as the CPU, together with an NVIDIA 2080TI GPU.

A. RAIM OVER TD3FOD (RAIMFOD)

TD3fOD was used to train RAIM [11], an algorithm developed for AIM systems. RAIM can control the speed of AVs in the surroundings of intersections so that the flow and safety of these vehicles can be notably increased, significantly reducing waiting time, pollutant emissions, and consumption of both fuel and power electricity. RAIM leverages the advantages provided by MADRL, to find a policy to control vehicles intelligently, collectively, and collaboratively. RAIM belongs to MADRL's centralized trained and

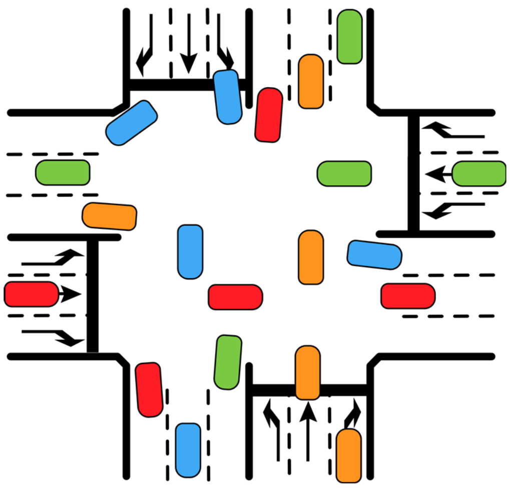


FIGURE 4. Representation of simulated intersection with 4 approaches and 3 lanes/approach, where the movements go straight, turn right, and turn left are allowed.

centralized execution cooperative approach, where vehicles send their states to the AIM. The AIM obtains the action for each AV (centralized training and execution in the AIM) with a common goal [19].

In the original study, RAIM was trained through curriculum-based learning, increasing the simulated vehicle flow when some stability in the results was achieved. The proposed solution optimized the system but required a large number of simulations and a considerable training time. Using TD3fOD, we aim to accelerate learning and reduce the number of simulations without using curriculum-based learning. This new approach is called RAIM from *Oracle* Demonstrations (RAIMfOD). The network architecture for both actors and critics consisted of four fully connected layers (448, 128, 50, and 1 neuron for each layer). The input of the network includes the characteristics of the vehicles to be controlled (e.g., position, speed, route, lane, etc.), and the output indicates the speed at which the vehicle should drive during the next time interval. These features, as well as the inner workings of RAIM, can be seen in more detail in the original RAIM article [66].

B. TRAINING SCENARIO

The training scenario was used to optimize TD3fOD/RAIM fOD. This scenario consisted of an intersection of four branches and three lanes for each branch, where it was allowed to turn left, turn right, and go straight with one movement for each lane. A representation of the simulated intersection is shown in Fig. 4.

As a reward signal, the following sparse reward was designed. Each agent (vehicle) received each timestep: *+10* (strong positive reward) when the vehicle crossed the intersection, *−10* (strong negative reward) when the vehicle collided with another vehicle, and *-timestep* otherwise to promote crossing the intersection as quickly as possible. Table 1 includes the hyperparameter values. These

TABLE 1. Parameters used.

Simulator	Simulation step	0.25 segs
	Flow	1200 veh/hour
	Train duration	5 mins
	Train scenario	4 branches, 2 lanes/way, and all ways.
	Control distance	100m
TD3fOD	Batch size	64
	TD3 Gamma	0.99
	τ_3	4×10^{-3}
	Learning rate actor	1×10^{-5}
	Learning rate *Oracle*	1×10^{-4}
	Learning rate critics	1×10^{-4}
	Weigh decay	1×10^{-8}
	Policy Noise	0.15
	Policy Noise Clip	0.1
	$N_{reinforcement\_epochs}$	200
	TD3 update actor every	2
	$\mathcal{D}^{reinforcement}$ $size$	$2^{20} \approx 1\times10^{6}$
	$reinforcement\_learn\_every$	15
	$N_{imitation\_epochs}$	5
	$\mathcal{D}^{imitation}$ $size$	$2^{17} \approx 1\times10^{5}$
	$imitation\_learn\_every$	15
	$warmup\_simulations$	100
	p	0.995
	th	250

hyperparameters and the reward values were selected empirically, offering notable performance and a stable training process.

C. TESTING SCENARIO

A testing scenario was incorporated to test the ability of our algorithm to address never-before-seen scenarios. For this purpose, a scenario with a traffic distribution that presents multiple variations was proposed, with low (500 veh/h), medium (1000 veh/h), and high (2000 veh/h) flows. In addition, it presented asymmetric and symmetric traffic regarding the branches of origin, North/South (N/S) and West/East (W/E). The intersection was the same as in the previous scenario, with three lanes, where left, right, and straight turns were allowed. Fig. 5 shows the time distribution of the simulated flow. The algorithms used to compare the performance of RAIMfOD were: no control, a stop signal, a fixed-time traffic light algorithm with total cycles of 30, 60, and 90 s, iREDVD [67] (a traffic light algorithm based on queueing theory), the previous RAIM algorithm and the direct implementation of RAIM with DDPGfD [15].

The following key performance metrics were used to compare the different algorithms: travel time, waiting time, time loss due to congestion, and pollution and consumption metrics (CO, CO2, HC, PMx, NOx, and fuel and electricity). The vehicle distribution used was 35% diesel cars, 35% gasoline cars, and 30% electric cars, with zero emissions.

VI. RESULTS

This section shows the results obtained in both the training and testing scenarios.

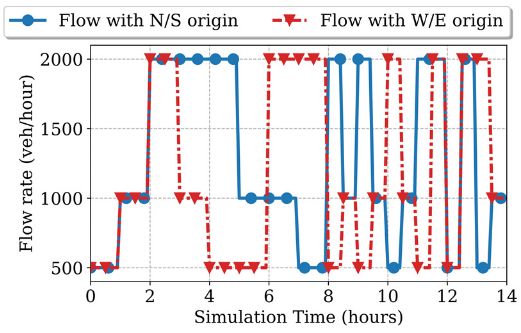
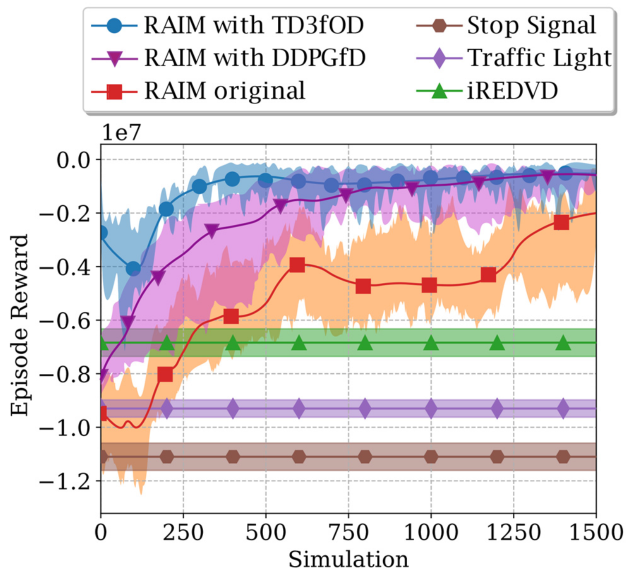
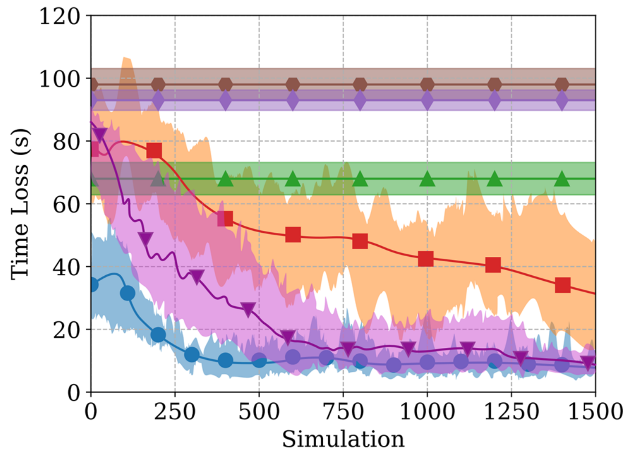


FIGURE 5. Flow test scenario distribution.



(a) Episode Reward evolution.



(b) Time loss evolution.

FIGURE 6. Training results. (a) Episode reward evolution (b) Time Loss evolution. RAIM with TD3fOD was able to learn a robust policy faster than the original RAIM. We plot the smoothed mean with an exponential moving average and a 90% confidence interval across 3 seeds.

A. TRAINING SCENARIO

Fig. 6 depicts the reward of each episode throughout the simulations and the average time loss for each vehicle in the training scenario. The time loss is due to driving below the

TABLE 2. Results in the testing scenario.

	Algorithm	Travel Time (s)	Waiting Time (s)	Time loss (s)	CO emiss. (g)	CO2 emiss. (g)	HC emiss. (mg)	PMx emiss. (mg)	NOx emiss. (mg)	Fuel cons. (ml)	Elect. cons. (W)
Traditional	*No control*	121.42 ± 11.41	83.32 ± 17.53	91.44 ± 23.96	2.47 ± 1.65	111.78 ± 68.32	13.97 ± 7.65	5.21 ± 1.11	375.87 ± 51.54	41.24 ± 10.32	33.97 ± 3,21
	Stop Signal	107.53 ± 8.94	72.65 ± 12.42	78.15 ± 19.85	1.74 ± 0.99	81.70 ± 32.44	11.65 ± 6.01	3.52 ± 0.99	272.65 ± 37.95	35.33 ± 8.99	35.44 ± 2,88
	TL30	86.85 ± 4.43	50.07 ± 15.21	55.46 ± 17.83	1.08 ± 0.64	69.45 ± 23.64	7.32 ± 3.23	2.61 ± 0.86	194.99 ± 31.43	30.55 ± 7.66	32.39 ± 2,97
	TL60	81.69 ± 4.87	45.72 ± 12.65	50.07 ± 12.32	1.40 ± 0.75	82.82 ± 33.55	8.62 ± 2.31	2.85 ± 0.89	211.53 ± 32.32	33.96 ± 7.32	32.03 ± 3,02
	TL90	91.52 ± 9.87	52.69 ± 16.98	61.46 ± 19.26	1.33 ± 0.83	79.15 ± 38.52	8.76 ± 2.78	3.36 ± 0.92	231.53 ± 37.45	31.42 ± 8.31	36,64 ± 3,04
	iREDVD	72,06 ± 6,75	27.12 ± 9.98	32.25 ± 15.23	1.49 ± 0.43	81.34 ± 23.42	9.00 ± 3.05	3.18 ± 0.67	221.94 ± 33.95	35.60 ± 9.85	36.50 ± 2.87
	RAIM	36,91 ± 9,86	2.31 ± 2.05	6.21 ± 3.22	1.10 ± 0.88	65.52 ± 33.88 ±	7.55 ± 2.03	2.74 ± 0.55	124.48 ± 24.92	29.69 ± 4.32	26.40 ± 2.66
	RAIM over DDPGfD	34,11 ± 3,73	1.92 ± 0.91	4.22 ± 1.18	1.08 ± 0.34	53.38 ± 10.85	6.81 ± 1.32	2.66 ± 0.24	138.29 ± 15.13	29.02 ± 1.41	24.46 ± 1.16
	RAIM over TD3fOD	**33,24 ± 3,21**	**1.86 ± 0.99**	**4.21 ± 1.21**	**1.02 ± 0.23**	**52.54 ± 10.21**	**6.79 ± 1.68**	**2.47 ± 0.21**	**136.85 ± 14.33**	**28.88 ± 1.27**	**24.03 ± 1.09**
Improvement of RAIM over TD3fOD											
	Abs	*[-88.19, -38.82]*	*[-81.46, -25.26]*	*[-87.23, -28.04]*	*[-1.45, -0.06]*	*[-58.24, -16.91]*	*[-7.18, -0.53]*	*[-2.74, -0.14]*	*[-239.02, -58.14]*	*[-12.36, -1.67]*	*[-12.61, -8.01]*
	%	*[72.62, 53.87]*	*[97.77, 93.14]*	*[95.40, 86.95]*	*[58.70, 5.56]*	*[53.01, 24.35]*	*[51.40, 7.24]*	*[52.59, 5.36]*	*[63.59, 29.82]*	*[29.97, 5.47]*	*[34.42, 24.98]*

The improvement shows the range between [best-case, worst-case]. It is shown as an absolute value (abs) in the corresponding units and percentage value (%).

TLXX means Traffic Light with a fixed total cycle duration of XX segs.

ideal speed and is defined as follows: *total_duration* ∗ *(1 - speed/ideal_speed)*. The results showed significant improvements in several aspects.

The main improvement is the increase in training speed, which reduced by 5 to 6 times the number of simulations required to achieve even better performance than the original DRL TD3 (RAIM), with a reduced variance of the results and a much more stable and robust policy. Compared to DDPGfD, it can be seen that the use of the *Oracle* to extract feedback and the improvements proposed in TD3 allow a threefold increase in speedup and considerably reduce the variance in the control behavior of the trained RAIM policy. As shown in Fig. 6b, RAIM with TD3fOD (RAIMfOD) can reduce the time loss to less than 20 s over 200 simulations. Comparatively, the original RAIM must train more than 1500 simulations to obtain a policy that reduces time loss by 20 s. Fig. 6b depicts the reward metric. As it can be seen, it reduces the number of simulations by more than x6, accelerating the training of new advanced control systems. Finally, a comparison of the training results with those obtained using traditional control techniques shows that the metrics are much better with RAIM and RAIMfOD.

In Fig. 6a and 6b, there are three different phases of RAIMfOD training. The pre-training began between simulations 0 and 100, filling the Imitation and RL replay buffers. The pre-training ended from simulation 100 to simulation 250, and RAIMfOD started. This situation can be seen in the shift in the metrics' trend over 100 iterations. In this range of simulations, the transition between learning by "*soft-copying*" the *Oracle* and the RL of TD3 begins, with the simulator taking more actions and the actor in TD3 acting as a "*sticky action.*" From simulation 250 onwards, the τ_1 and τ_2 curves cross each other, and most of the actions are carried out by the TD3 actor, allowing us to find a better control policy to further optimize the results. This highlights the notable performance of LfOD, allowing the achievement of a policy that outperforms that offered by the expert through a smoothed step from a pre-trained policy made by the *Oracle* to the policy learned by RL.

B. TESTING SCENARIO

We demonstrate the ability of our algorithm to generalize and adapt to new situations in the testing scenario. This illustrates the benefits offered by MADRL and LfOD. The results are presented in Table 2. In this table, it can be seen that RAIM with TD3fOD and original RAIM obtain very similar results, demonstrating that both algorithms find solutions with very notable performance, but RAIMfOD finds a policy much earlier and with a much lower variance because of LfOD, as can be seen in Fig. 6.

From the results included in Table 2, we confirm a reduction in the time loss between 95% and 86%, and a reduction in waiting time between 97% and 93%. This represents a reduction in travel time of between 72% and 53%. Regarding the emission of pollutant gases, we see that a significant improvement has been achieved in all metrics, decreasing all studied variables by over 50%. Finally, in terms of vehicle fuel and electricity consumption, a reduction of between 29% and 5% is achieved for combustion vehicles, and between 34% and 24% for electric vehicles. These results indicate the potential of LfOD and MADRL systems for the control of AVs using a centralized approach.

C. DISCUSSION

The results obtained in this work demonstrate the benefits of using MADRL with TD3fOD to solve different real problems. This solution has the potential to solve these problems in real-time, decreasing the time needed to train new systems, and improving the performance of existing ones. The TD3fOD method can find a policy that significantly improves the results obtained by traditional training techniques, resulting in more stable and robust policies.

When comparing the performance of RAIM with TD3fOD (RAIMfOD) with the performance of the original RAIM, we observe that the learning performance of RAIMfOD is between $\times 5$ and $\times 6$ faster, and the variance of the results is substantially smaller. Moreover, when compared to RAIM with DDPGfD, we can see that thanks to the advantages offered by TD3, as well as the use of the *Oracle*, we can reduce the number of simulations by up to $\times 3$, also significantly reducing the variance of the policy during training. Finally, if we look at the testing scenario, we see that the analyzed metrics outperform the original RAIM, showing the robustness of the proposed algorithm in scenarios never seen before, obtaining an improvement of between 72% and 53% in travel time and a reduction in waiting time of between 97% and 93%. In addition, in most cases, emissions of polluting gases are reduced by more than 50% and energy or fuel consumption by almost 30%.

VII. CONCLUSION

The success of AVs depends on advances in various driving and control system components, and the understanding and handling of unpredictable situations that can arise in complex driving environments. The application of MADRL allows the development of dynamic systems capable of adapting to many conditions, acting collectively and proactively, anticipating dangerous situations and ultimately preventing accidents and increasing flow. LfD can provide a simple way to find adaptive control policies capable of solving highly complex tasks in different fields of work, such as teaching robots how to walk, navigate through dangerous traffic, manage the presence of obstacles, avoid collisions with other road users, and perform safe and efficient maneuvers at intersections.

To enable the use of an expert's demonstrations in environments where the expert is not accessible, we propose in this work the use of an *Oracle* in LfD, obtaining LfOD. The *Oracle* is trained by Imitation Learning; thus, it can be employed to teach from demonstration an agent using RL. This original approach facilitates the use of LfD in environments where no expert is available to obtain feedback. In this way, an agent can be trained much more quickly, achieving a better policy than the expert can offer and presenting a lower variance in the results.

TD3 was the algorithm modified to use the demonstrations offered by *Oracle*. Following the nomenclature used in the algorithms developed for LfD, we called the new proposal TD3 from *Oracle* Demonstrations (TD3fOD). The modifications made to TD3 were: *i*) incorporation of an *Oracle* trained by Imitation Learning from the states extracted from the simulator; *ii)* inclusion of several parameters for a smooth and progressive transition between LfOD and RL; and *iii)* use of two replay buffers, one for demonstrations to train *Oracle* and the other for RL, in addition to the use of PER to speed up learning. TD3fOD was applied to the SUMO traffic simulator to speed up the learning of an AIM system. The only AIM to date that used RL was RAIM, and for this reason, this algorithm was used above TD3fOD (RAIM over TD3fOD). The results obtained in the training scenario demonstrated that TD3fOD achieved more efficient learning than TD3, finding a faster control policy and speeding up the training by 5–6 times. In addition, the policy found offers significantly lower variance, providing more robust results. Furthermore, it outperforms the policy shown by the expert. TD3fOD also achieved good results in the testing scenario, improving those obtained by RAIM in all studied metrics and with a lower variance. These results highlight the benefits of using LfOD. RAIM over TD3fOD reduces the waiting time between 97% and 93%, resulting in a reduction of up to 50% in the emission of contaminating gases compared to other traditional vehicle control techniques, such as traffic lights, and other advanced techniques, such as iREDVD. Finally, in terms of consumption, combustion vehicles reduce their fuel consumption by up to 29% and electric vehicles by 34%. Moreover, if we compare TD3fOD with another LfD algorithm, such as DDPGfD, we show that the use of the proposed LfOD approach allows to speed up the training, reducing the number of interactions with the simulator by up to $\times 3$.

Based on our proposal, it is possible to extract the hidden agent (learned by imitation with an Oracle) in those simulators where it is not possible (or too complicated), to take advantage of the benefits offered by LfD (training acceleration and more robust policies) for the development of new complex control algorithms. The proposed approach for LfOD is applicable to different RL algorithms. One of the main contributions of this work is the development of an expert agent (Oracle) in environments where it does not exist to take advantage of LfD, as well as the incorporation of this approach in the TD3 DRL algorithm, making severe modifications in TD3 to adapt the training process to the presence of *Oracle*.

As future work, we plan to include multiple hierarchical systems that enable level-based control of different actors in a complete network of traffic intersections and explore the development of new algorithms that can use LfOD in other domains.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, ''Playing Atari with deep reinforcement learning,'' in *Proc. NIPS Deep Learn. Workshop*, 2013, p. 9.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, ''Human-level control through deep reinforcement learning,'' *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, ''Mastering the game of go with deep neural networks and tree search,'' *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[4] J. Gauci, E. Conti, Y. Liang, K. Virochsiri, Y. He, Z. Kaden, V. Narayanan, X. Ye, Z. Chen, and S. Fujimoto, ''Horizon: Facebook's open source applied reinforcement learning platform,'' Nov. 2018, *arXiv:1811.00260*.

[5] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, ''Learning dexterous in-hand manipulation,'' *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, Jan. 2020.

[6] W.-X. Liu, J. Cai, Q. C. Chen, and Y. Wang, ''DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks,'' *J. Netw. Comput. Appl.*, vol. 177, pp. 1–10, Mar. 2021.

[7] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, ''Mastering the game of go without human knowledge,'' *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[8] J. S. Obando-Ceron and P. S. Castro, ''Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research,'' in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1373–1383.

[9] K. Dresner and P. Stone, ''Multiagent traffic management: A reservation-based intersection control mechanism,'' in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, vol. 2, 2004, pp. 530–537.

[10] G.-P. Antonio and C. Maria-Dolores, ''AIM5LA: A latency-aware deep reinforcement learning-based autonomous intersection management system for 5G communication networks,'' *Sensors*, vol. 22, no. 6, p. 2217, Mar. 2022.

[11] A. Guillen-Perez and M. Cano, ''Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrows intersections,'' *IEEE Trans. Veh. Technol.*, early access, Apr. 25, 2022, doi: 10.1109/TVT.2022.3169907.

[12] S. Ross, G. J. Gordon, and J. A. Bagnell, ''A Reduction of imitation learning and structured prediction to no-regret online learning,'' in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 15, 2010, pp. 627–635.

[13] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, ''Imitation from observation: Learning to imitate behaviors from raw video via context translation,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1118–1125.

[14] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Leibo, and A. Gruslys, ''Deep *Q*-learning from demonstrations,'' in *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI)*, 2018, pp. 3223–3230.

[15] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, ''Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards,'' 2017, *arXiv:1707.08817*.

[16] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, ''Overcoming exploration in reinforcement learning with demonstrations,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6292–6299.

[17] P. A. Lopez, E. Wiessner, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, and P. Wagner, ''Microscopic traffic simulation using SUMO,'' in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.

[18] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, ''Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications,'' *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[19] Y. J. Park, Y. J. Lee, and S. B. Kim, ''Cooperative multi-agent reinforcement learning with approximate model learning,'' *IEEE Access*, vol. 8, pp. 125389–125400, 2020.

[20] P. J. Hoen, K. Tuyls, L. Panait, S. Luke, and J. A. La Poutré, ''An overview of cooperative and competitive multiagent learning,'' in *Proc. Int. Workshop Learn. Adaption Multi-Agent Syst.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2006, pp. 1–46.

[21] G. Chen, ''A new framework for multi-agent reinforcement learning—Centralized training and exploration with decentralized execution via policy distillation,'' in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2020, pp. 1–10.

[22] P. K. Sharma, E. G. Zaroukian, R. Fernandez, A. Basak, and D. E. Asher, ''Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training,'' *Proc. SPIE*, vol. 11746, p. 84, Apr. 2021.

[23] S. Fujimoto, H. van Hoof, and D. Meger, ''Addressing function approximation error in actor-critic methods,'' in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 4, 2018, pp. 2587–2601.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, ''Continuous control with deep reinforcement learning,'' in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

[25] A. Raffin, J. Kober, and F. Stulp, ''Smooth exploration for robotic reinforcement learning,'' in *Proc. 5th Conf. Robot Learn. (CoRL)*, 2020, pp. 1–15.

[26] A. Stooke and P. Abbeel, ''rlpyt: A research code base for deep reinforcement learning in PyTorch,'' 2019, *arXiv:1909.01500*.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, ''Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,'' in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1861–1870.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, ''Proximal policy optimization algorithms,'' 2017, *arXiv:1707.06347*.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, ''Asynchronous methods for deep reinforcement learning,'' in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1928–1937.

[30] S. Ross and D. Bagnell, ''Efficient reductions for imitation learning,'' in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, in Proceedings of Machine Learning Research, vol. 9. Sardinia, Italy, May 2010, pp. 661–668. [Online]. Available: http://proceedings.mlr.press/v9/ross10a.html

[31] M. Bain and C. Sammut, ''A framework for behavioural cloning,'' *Mach. Intell.*, vol. 15, no. 1, pp. 103–129, 2001.

[32] S. Daftry, J. A. Bagnell, and M. Hebert, ''Learning transferable policies for monocular reactive MAV control,'' in *Proc. Int. Symp. Exp. Robot.*, in Springer Proceedings in Advanced Robotics, vol. 1, 2017, pp. 3–11.

[33] H. Daumé, J. Langford, and D. Marcu, ''Search-based structured prediction,'' *Mach. Learn.*, vol. 75, no. 3, pp. 297–325, Jun. 2009.

[34] K. Dresner and P. Stone, ''A multiagent approach to autonomous intersection management,'' *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Mar. 2008.

[35] P. Stone and K. Dresner, ''Human-usable and emergency vehicle-aware control policies for autonomous intersection management,'' in *Proc. 4th Int. Workshop Agents Traffic Transp. (ATT)*, Hakodate, Japan, May 2016, pp. 17–25.

[36] K. Dresner and P. Stone, ''Sharing the road: Autonomous vehicles meet human drivers,'' in *Proc. 20th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 1, Jan. 2007, pp. 1263–1268.

[37] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti, ''Revisiting street intersections using slot-based systems,'' *PLoS ONE*, vol. 11, no. 3, Mar. 2016, Art. no. e0149607.

[38] D. Fajardo, T.-C. Au, S. T. Waller, P. Stone, and D. Yang, ''Automated intersection control: Performance of future innovation versus current traffic signal control,'' *Transp. Res. Rec. J., Transp. Res. Board*, vol. 2259, no. 1, pp. 223–232, Jan. 2011.

[39] S. Huang, A. W. Sadek, and Y. Zhao, "Assessing the mobility and environmental benefits of reservation-based intelligent intersections using an integrated simulator," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1201–1214, Sep. 2012.

[40] X. Wei, G. Tan, and N. Ding, "Batch-light: An adaptive intelligent intersection control policy for autonomous vehicles," in *Proc. IEEE Int. Conf. Prog. Informat. Comput.*, May 2014, pp. 98–103.

[41] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 81–90, Mar. 2012.

[42] F. Zhu and S. V. Ukkusuri, "A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment," *Transp. Res. C, Emerg. Technol.*, vol. 55, pp. 363–378, Jun. 2015.

[43] M. W. Levin, H. Fritz, and S. D. Boyles, "On optimizing reservation-based intersection controls," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 505–515, Mar. 2017.

[44] W. Sun, J. Zheng, and H. X. Liu, "A capacity maximization scheme for intersection management with automated vehicles," *Transp. Res. Proc.*, vol. 23, pp. 121–136, Jul. 2017.

[45] A. Mirheli, L. Hajibabai, and A. Hajbabaie, "Development of a signal-head-free intersection control logic in a fully connected and autonomous vehicle environment," *Transp. Res. C, Emerg. Technol.*, vol. 92, pp. 412–425, Jul. 2018.

[46] W. Wu, Y. Liu, W. Liu, F. Zhang, V. Dixit, and S. T. Waller, "Autonomous intersection management for connected and automated vehicles: A lane-based method," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 31, 2022, doi: 10.1109/TITS.2021.3136910.

[47] A. P. Chouhan and G. Banda, "Autonomous intersection management: A heuristic approach," *IEEE Access*, vol. 6, pp. 53287–53295, 2018.

[48] S. R. K. Branavan, D. Silver, and R. Barzilay, "Learning to win by reading manuals in a Monte–Carlo framework," *J. Artif. Intell. Res.*, vol. 43, pp. 661–704, Apr. 2012.

[49] J. Wu, A. Abbas-Turki, and A. E. Moudni, "Cooperative driving: An ant colony system for autonomous intersection management," *Appl. Intell.*, vol. 37, no. 2, pp. 207–222, Sep. 2012.

[50] H. Xu, Y. Zhang, L. Li, and W. Li, "Cooperative driving at unsignalized intersections using tree search," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4563–4571, Nov. 2020.

[51] M. W. Levin, S. D. Boyles, and R. Patel, "Paradoxes of reservation-based intersection controls in traffic networks," *Transp. Res. A, Policy Pract.*, vol. 90, pp. 14–25, Aug. 2016.

[52] E. van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learning for traffic light control," in *Proc. 30th Conf. Neural Inf. Process. Syst. (NIPS)*, no. 1, 2016, p. 8.

[53] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA J. Automat. Sinica*, vol. 3, no. 3, pp. 247–254, Jul. 2016.

[54] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2496–2505.

[55] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[56] T. Tan, F. Bao, Y. Deng, A. Jin, Q. Dai, and J. Wang, "Cooperative deep reinforcement learning for large-scale traffic grid signal control," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2687–2700, Jun. 2020.

[57] A. Fontoura, D. Haddad, and E. Bezerra, "A deep reinforcement learning approach to asset-liability management," in *Proc. 8th Brazilian Conf. Intell. Syst. (BRACIS)*, Oct. 2019, pp. 216–221.

[58] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 3414–3421.

[59] Y. Wu, H. Chen, and F. Zhu, "DCL-AIM: Decentralized coordination learning of autonomous intersection management for connected and automated vehicles," *Transp. Res. C, Emerg. Technol.*, vol. 103, pp. 246–260, Jun. 2019.

[60] H. Seong, C. Jung, S. Lee, and D. H. Shim, "Learning to drive at unsignalized intersections using attention-based deep reinforcement learning," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 559–566.

[61] M. Yuan, J. Shan, and K. Mi, "Deep reinforcement learning based game-theoretic decision-making for autonomous vehicles," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 818–825, Apr. 2022.

[62] M. Jing, X. Ma, W. Huang, F. Sun, C. Yang, B. Fang, and H. Liu, "Reinforcement learning from imperfect demonstrations under soft expert guidance," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5109–5116.

[63] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn. (CoRL)*, 2017, pp. 1–16.

[64] S. Panwai and H. Dia, "Comparative evaluation of microscopic car-following behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 3, pp. 314–325, Sep. 2005.

[65] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–21.

[66] A. Guillen-Perez and M.-D. Cano, "RAIM: Reinforced autonomous intersection management—AIM based on MADRL," in *Proc. NeurIPS Workshop Challenges Real-World RL*, 2020, pp. 1–12.

[67] A. Guillen-Perez and M. Cano, "Intelligent IoT systems for traffic management: A practical application," *IET Intell. Transp. Syst.*, vol. 15, no. 2, pp. 273–285, Feb. 2021.

ANTONIO GUILLEN-PEREZ received the B.Sc. degree in telecommunication engineering and the M.Sc. degree in information and communication technologies from the Universidad Politécnica de Cartagena, Murcia, Spain, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Department of Information and Communication Technologies.

He has published several papers in national and international conferences and journals. His main research interests include intelligent transport systems (ITS), artificial intelligence (AI), deep learning (DL), multi-agent systems, multiagent deep reinforcement learning, autonomous cars, the IoT, and smart cities. He is a member of many IEEE technical committees. He has collaborated as a reviewer for international conferences.

MARIA-DOLORES CANO (Senior Member, IEEE) received the telecommunications engineering degree from the Universidad Politécnica de Valencia, Spain, in 2000, and the Ph.D. degree from the Universidad Politécnica de Cartagena (UPCT), Spain, in 2004.

She joined UPCT, in 2000, where she is currently an Associate Professor with the Department of Technologies and Communications. She has published numerous research works in international journals and conferences in the areas of quality of service (QoS)/quality of user experience (QoE), intelligent transportation systems, the IoT, and security provisioning. She was awarded a Fulbright Grant as a Postdoctoral Researcher at Columbia University, New York, NY, USA, in 2006. She has also been collaborating with several universities in South America, since 2007. She has received the Best Paper Award at the Tenth IEEE International Symposium on Computers and Communications, in 2005, the Exemplary Reviewer Recognition by IEEE Communications Letters, in 2010, and among other recognitions.