Navigation In Urban Environments Amongst Pedestrians Using Multi-Objective Deep Reinforcement Learning

Niranjan Deshpande, Dominique Vaufreydaz, Anne Spalanzani

NAVIGATION IN URBAN ENVIRONMENTS AMONGST PEDESTRIANS USING MULTI-OBJECTIVE DEEP REINFORCEMENT LEARNING

AUTHOR VERSION

Niranjan Deshpande[1], **Dominique Vaufreydaz**[2] **and Anne Spalanzani**[1†]

Abstract

Urban autonomous driving in the presence of pedestrians as vulnerable road users is still a challenging and less examined research problem. This work formulates navigation in urban environments as a multi objective reinforcement learning problem. A deep learning variant of thresholded lexicographic Q-learning is presented for autonomous navigation amongst pedestrians. The multi objective DQN agent is trained on a custom urban environment developed in CARLA simulator. The proposed method is evaluated by comparing it with a single objective DQN variant on known and unknown environments. Evaluation results show that the proposed method outperforms the single objective DQN variant with respect to all aspects.

1 INTRODUCTION

One of the most important aspects of autonomous driving is navigation. A typical navigation system of an autonomous vehicle employs a sequential pipeline which is widely used by the industry and is considered the conventional approach [1], [2]. It consists of a *Route Planner* at the highest level, generating a list of sparse way-points from start to destination. This is followed by a *Behavioral Decision Making* (BDM) layer, responsible for deciding on a driving maneuver based on the information of other traffic participants' behavior, road conditions and traffic signals. The minimal goal of BDM is to ensure safety, while the other goals are speed, comfort, fuel efficiency. Based on the selected behaviour, *Motion Planning* and *Control* modules are used for continuous path generation and execution respectively.

A natural approach to automating the BDM is to model each of its behavior as states in a finite state machine (FSM). In fact, FSMs coupled with different heuristics were adopted as a mechanism for behavior planning by most teams in the DARPA challenges, given their simplicity and transparency. However, their major drawback is the difficulty to model the uncertainties and complex urban traffic scenarios.

In recent years, Deep Reinforcement Learning (deep-RL), has been successfully applied to several sequential decision-making problems. This has let to a surge in the use of deep-RL based decision making approaches for autonomous driving as well. However, these approaches are applied for highway driving scenarios in the presence of vehicles as other road users.

Furthermore, existing deep-RL approaches for autonomous driving consider navigation as a simplified single objective RL problem and employ a scalar reward function for different objectives like safety and speed. This simplification works for highway driving, however, in the context of urban environments, an autonomous vehicle often has to satisfy several conflicting objectives: avoiding collision with vulnerable road users such as pedestrians, maintaining speed limits, following traffic rules and ensuring passenger comfort.

This work formulates autonomous navigation in urban environments as a multi-objective RL problem where each objective is learned by a separate agent. A common policy is then formulated by these agents by taking into consideration all these objectives. The motivation behind this rationale is the difficulty in designing a scalar reward that could properly weigh the importance of each of the above mentioned objectives.

This work presents the following contributions:

- A multi-objective RL method called thresholded lexicographic Deep Q-learning is presented for navigation in urban environments in the presence of pedestrians.
- A single-objective RL method presented in our previous work [3, 4], is improved with a new reward function and used for comparison purposes.
- For training, a custom urban environment is developed and the proposed methods are compared and evaluated to quantify their performance.

[*1] Univ. Grenoble Alpes, Inria, 38000 Grenoble, France, email: FirstName.LastName@inria.fr

[†2]Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

2 BACKGROUND

2.1 Single Objective Reinforcement Learning

Reinforcement learning (RL) is often formulated as a Markov Decision Process (MDP), a tuple *(S, A, T, R)*, characterized by a set of states *S*, a set of actions *A* and a scalar reward function *R* and a transition function *T*. In the single-objective reinforcement learning (SORL), at each time *t*, an agent interacts with the environment by observing the state s_t and performing an action a_t, according to some policy π. Performing an action generates a reward r_t and takes the agent to next state s_{t+1} and the process repeats until a termination condition. The agents goal is to maximize the expected discounted reward $R_t = \sum_{t=0}^{T} \gamma \cdot r_t$, where $\gamma \in [0, 1]$ is a discount factor deciding the importance of future rewards.

In Q-learning algorithm, actions are selected using a Q-function *Q(s,a)*, which constitutes the expected discounted reward for executing an action *a* in state *s*. The optimal action for a state *s* is defined as:

$$a' = \arg\max_{a \in A} Q(s, a) \tag{1}$$

An agent is said to be following an optimal policy π^*, given it always selects the optimal action. The function $Q(s, a)$ helps in forming an optimal policy by simply selecting actions with highest Q-values.

In Deep Q-Networks, the $Q(s, a)$ values are approximated using deep neural networks making Q-learning feasible for many real-world complex applications. A variant of DQN, called Double Deep Q-Network (DDQN) uses a separate target network to make the learning process more stable. Its parameters are periodically updated using actual network.

2.2 Multi-Objective Reinforcement Learning

In several complex applications, there exists more than one objective to be considered. If these objectives are independent or directly related, they could be merged to form a single objective. However, often the objectives are in conflict, that is, maximizing one objective causes minimization of another. For such applications, multi-objective reinforcement learning (MORL) is used to characterize two or more objectives simultaneously. MORL is often modelled as Multi-objective Markov Decision Process (MOMDP) instead of single objective MDP. An MORL agent, instead of a scalar reward, receives a vector of rewards at each time step, corresponding to each objective. Given *O* as a set of *n* objectives, the reward vector is defined as: $R = [r_1, r_2, ..., r_n]$, where r_i represents scalar reward for objective $o_i \in O$. The expected discounted reward for an objective o_i, at time *t*, is defined as:

$$R_{i,t} = \sum_{t=0}^{T} \gamma \cdot r_{i,t} \tag{2}$$

Algorithm 1: Superior

Input : $TQ(s, a)$, $TQ(s, a^{'})$, i
Output: if $TQ(s, a)$ superior to $TQ(s, a^{'})$
if $TQ_i(s, a) > TQ_i(s, a^{'})$ **then**
  return true;
else if $TQ_i(s, a) = TQ_i(s, a^{'})$ **then**
  if *i = n* **then**
    return true
  else
    return Superior($TQ(s, a)$, $TQ(s, a^{'})$, $i + 1$)
  end
else
  return false
end

Usually, a separate Q-function $Q_i(s, a)$ is considered for each objective such that:

$$Q_i(s, a) = E[R_{i,t} | s_t = s, a_t = a] \tag{3}$$

resulting into a vector of Q-functions. At each time step *t*, the agent uses the function $Q_i(s, a)$ to find the optimal action satisfying objective o_i. Since the agent can perform a single action at a time, a truncation method is required.

2.3 Thresholded Lexicographic Q-learning

Threshold lexicographic Q-Learning (TLQ-Learning), introduced by *Gabor et al.* [5] is an algorithm which uses lexicographic ordering on objectives based on their priorities. A threshold is used for each objective to define the minimum or maximum acceptable Q-value (depending on whether the objective is to be minimised or maximised). Thresholds are set on the first $i - 1$ objectives and the last objective is left unconstrained.

Selection of action is done by using a combination of thresholding and lexicographic ordering to the objective's Q-values. If $Q_i(s, a)$ is the value for i^{th} objective corresponding to action *a* in state *s* and T_i is threshold, then:

$$TQ_i(s, a) = min(Q_i(s, a), T_i) \tag{4}$$

Given a state *s*, a greedy action a' is selected, satisfying the recursive function as defined in Algorithm 1 [6].

3 RELATED WORK

3.1 Navigation in the presence of vehicles

In recent years, deep-RL for navigation on highway environments has been explored extensively. Several approaches use Deep Q-Network (DQN) and its variants for lane change decision making. A detailed discussion of relevant approaches could be found in our previous work [3, 4] and in survey papers [1, 2].

Overtaking maneuver is formulated as a multi-objective reinforcement learning problem by *Ngai and Yung* in [7]. Values from a tabular Q-function of each objective are

scalarized by weighted sum into a single policy. Another approach is proposed by *C. Li and K. Czarnecki* in [8] for urban intersection handling. The trained multi-objective DQN agent learns to drive on multi-lane roads and intersections, yielding and changing lanes according to traffic rules. It presents a deep learning variant of thresholded lexicographic Q-learning for the task of autonomous driving.

3.2 Navigation in the presence of pedestrians

There exists some prior work on navigation of small mobile robots around pedestrians in outdoor environments. *Bai et al.* in [9] propose an intention aware method for navigation in environments with pedestrians. A hybrid A^* planner is used for global path planning, while an online POMDP (Partially observable Markov decision process) is used for velocity control. An intention awake decision maker is proposed in [10] where pedestrian behavior is modelled in terms of potential fields. An POMDP approach in [11] proposes a pedestrian motion predictor called PORCA to model their intentions and interactions. *Barbier et al.* in [12] model autonomous vehicle and pedestrian interaction at intersections using POMDP. However, solving a POMDP exactly is an intractable problem. Safety of POMDP solutions in the context of autonomous driving is still a topic of research [15]. Furthermore, online POMDP planners are often computationally complex and require much time, limiting their use for autonomous vehicle platforms.

Some approaches also explore the use of game theory. In [13], *Fox et al.* propose a game-theoretical mathematical model called *sequential chicken*, for two agents meeting at an unsigned interaction and negotiating for priority. The *sequential chicken* model is further extended in [14] to negotiate collision with a pedestrian at an unsignalized intersection. However, game-theoretic approaches are too simplistic to model complex autonomous vehicle's interactions with pedestrians [15].

In our previous work [3, 4], a unique 3-D grid representation, suitable for urban environments, is used by a DQN agent to learn navigation in environment densily populated with pedestrians. This 3-D representation is also evaluated by *K. Mokhtari* and *A. Wagner* with different variants of DQN [15]. They further propose a safe deep-RL approach in [16] where an LSTM based future collision algorithm is used to mask unsafe actions.

4 PROBLEM STATEMENT

Autonomous navigation is usually formulated as a single-objective RL problem, especially for highways. However, in the context of urban environments, navigation is more challenging, since along with other vehicles, there are vulnerable road users such as pedestrians and cyclists. Pedestrian motions are less constrained and even slightest collision is likely to be fatal. Furthermore, an autonomous vehicle navigating in urban setups often needs to satisfy multiple conflicting objectives simultaneously, such as: collision avoidance, speed control, traffic rules and passenger comfort.

This work formulates autonomous navigation in urban environments as a multi-objective RL problem, where each objective is learned separately and a combined policy is formed that takes into account all the objectives. These objectives have different priorities associated with them. For instance, an autonomous vehicle needs to avoid any collision situations before considering speed control and traffic rules. Since thresholded lexicographic learning follows a similar procedure, this work extends the deep learning variant of thresholded lexicographic Q-learning proposed in [8] for the task of urban autonomous driving amongst pedestrians.

5 METHODOLOGY

This work considers an autonomous vehicle as an RL agent with multiple objectives, receiving rewards with respect to these objectives and having a separate Q-function for each of its objectives.

5.1 Objectives

The agent should avoid collisions with surrounding pedestrians and simultaneously drive at the desired speed. For this, two objectives in lexicographic order are considered:

- safety (o_{safety}): To avoid any collision situations.
- speed (o_{speed}): To ensure the agent drives as far as possible, within the speed limits.

5.2 MOMDP formulation

5.2.1 State Space

The state space contains information about the surrounding environment and the ego vehicle itself.

To store environment information (*environment state*), a multi-layered 3-D grid representation [3] is employed by defining a region of interest (ROI) around the ego vehicle with length L and width W. This ROI is further discretized into multiple 2-D grids, with each grid storing a particular type of information: occupancy, relative speed (in $\mathrm{m/s}$), relative heading (in degrees) and semantic information (road type).

For ego vehicle state information (*ego vehicle state*), its current velocity (in $\mathrm{m/s}$) is available.

5.2.2 Reward Vector

In MORL paradigm, the reward for the agent is in terms of a vector corresponding to each objective. In this work the following reward vector is considered:

$$R = [r_{safety}, r_{speed}] \tag{5}$$

where r_{safety} is the reward associated with the o_{safety} objective and r_{speed} is for o_{speed}.

Safety reward With respect to safety, an autonomous vehicle should be able to avoid any collisions while driving. After testing several reward combinations, the following function is used to formulate the reward:

$$r_{safety} = \begin{cases} r_c & \textit{for collision} \\ r_{nc} & \textit{for near collision} \end{cases} \tag{6}$$

where $r_c = -4$, is a large collision penalty and r_{nc} is a penalty for near collision situations. To compute r_{nc}, a dynamic distance (d_r) range is defined as a function of ego-vehicle's current speed (v_{ev}). If there exists a pedestrian crossing the road within (d_r), a penalty for the nearest front pedestrian is calculated using the following equations [17]:

$$\begin{aligned} d_r &= max\left(\frac{v_{ev}^2}{2a_{max}}, d_0\right) \\ r_{nc} &= exp\left(\frac{d_p - d_r}{d_r}\right) \end{aligned} \tag{7}$$

where, a_{max} denotes the ego-vehicle's maximum deceleration, d_0 is a minimum safety distance to the front pedestrian and d_p is the distance to the pedestrian.

Speed reward The reward for speed is computed using the following function:

$$r_{speed} = \begin{cases} \lambda \cdot (v_{ref} - v_{ev}), & \textit{if} \quad 0.0 < v_{ev} \leq v_{ref} \\ -1.0, & \textit{if} \quad v_{ev} \leq 0.0 \\ -0.5, & \textit{if} \quad v_{ev} > v_{ref} \end{cases} \tag{8}$$

where $\lambda = 1/v_{ref}$, v_{ev} is ego vehicle current speed and v_{ref} is the target desired speed.

5.2.3 Action Space

The action space consists of four high level actions for each DQN agent.

Actions	Description	Values
accelerate	Gradual speed-up	+1 $\mathrm{m/s^2}$
decelerate	Gradual slow-down	-1 $\mathrm{m/s^2}$
brake	Braking	-5 $\mathrm{m/s^2}$
steer	Maintain current speed	0 $\mathrm{m/s^2}$

5.3 Thresholded Lexicographic DQN

A separate DQN is used as an approximator for the Q-function of each objective: DQN_{safety} and DQN_{speed}. Each DQN receives a different state and reward with respect to their objectives, however, both have same action space (described in 5.2.3). The DQN_{safety} uses *environment state* as its state space input and r_{safety} is the reward generated. For DQN_{speed}, the state is *ego vehicle speed* and reward generated is r_{speed}.

In single objective RL, given a state *s*, the algorithm selects the greedy action a' as optimal action, such that:

$$a' = \arg\max_{a \in A} Q(s, a) \tag{9}$$

Whereas, Thresholded Lexicographic Q-learning (TLQ) selects an action for each objective based on a threshold that specifies the minimum admissible value for each objective. However, this work uses a slightly different meaning of the threshold, specifying how much worse than the optimal action is considered acceptable [8]. At every time step, a list of acceptable actions $(A_i \in A)$ is generated for each objective, however, the allowable actions for *i*th objective are restricted to those allowed by objective *i-1*.

The threshold is defined in terms of percentage:

$$T = [\tau_{safety}, \tau_{speed}] \tag{10}$$

For example, at time t, the optimal action by DQN_{safety} is a', then all other actions whose Q-value is greater than or equal to $\tau_{safety} * Q(s, a')$ are considered acceptable for objective o_{safety}. Then out of these acceptable actions, DQN_{speed} selects the optimal action for objective o_{speed} based on its state input and learned policy. The action selection procedure is described in Algorithm 2 [8]. The input to the algorithm is a list of learned Q-functions for each objective, a list of states for each objective and a boolean list indicating which objective is to be explored.

Algorithm 2: TLO Action Selection

Input : vector Q, vector S, vector E
Output: final action
$A_0 := A$
foreach *i* $\in$ *n* **do**

$$A_i(s_i) := \left\{ a \in A_{i-1}(s_{i-1}) \middle| Q_i(s_i, a) \geq \max_{a' \in A_{i-1}} Q_i(s_i, a') * \tau_i \right.$$
$$\left. \textbf{or} \quad a = \arg\max_{a' \in A_{i-1}} Q_i(s_i, a') \right\}$$

```
/* if objective i chosen to be explored */
```
if *E[i]* == *True* **then**
  return random action from $A_{i-1}(s_{i-1})$
end
end
return greedy action from $A_n(s_n)$

6 EXPERIMENTS

6.1 Neural Network Models

Two different MORL agents are investigated in this work, labelled as *Agent1* and *Agent2* respectively. The difference lies in the neural network (NN) architecture of their DQN_{safety}, while both agents have the same architecture for their DQN_{speed}.

Architecture for DQN$_{safety}$ The input to DQN_{safety} is the multi-layered 3-D grid of size $80 \times 60 \times 4$. The input is passed through three convolutional layers. The first layer has n_{conv1} filters set to 32, second layer has n_{conv2} filters

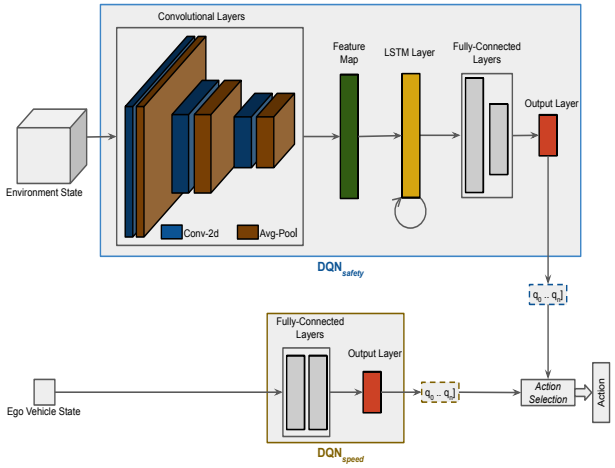

Figure 1: Separate DQNs used for o_{safety} and o_{speed} objectives. Thresholding and lexicographic ordering is performed on the Q-value outputs from these DQNs to generate final action

set to 64, third layer has n_{conv3} filters set to 64. Each layer uses a filter size of 5, stride of 3, has a ReLU activation function and is also followed by an AveragePooling layer for dimentionality reduction.

For *Agent1*, the output from these convolutional layers is fed into two fully connected layers with n_{full1} and n_{full2} units, set to 128 and 64 respectively, followed by ReLu activation functions. Finally, the output layer has 4 neurons corresponding to the action set.

For *Agent2*, the output from the convolutional layers is flattened and fed into an LSTM layer as shown in Figure 1 with n_{lstm} hidden units, set to 128. The output of LSTM is then fed into fully connected layers followed by the output layer.

Architecture for DQN$_{\text{speed}}$ Both the agents use same NN design for their DQN_{speed}. The input is the *ego vehicle state* (its current speed) which is passed through two fully connected layers with n_{full1} and n_{full2} units, set to 32 and 32, followed by ReLu activation. The output layer is of size 4.

6.2 Training Details

For the purpose of training, CARLA simulator [18] is used. Both the agents are trained for a total of 500000 steps. However, an episode ends if the agent reaches the goal or encounters a collision situation.

Both the DQNs maintain a separate experience replay memory to store up to 10,000 transitions. A mini-batch of size 32 is selected uniformly from the replay memory buffer by each DQN and the network weights are then updated accordingly using RMSProp algorithm [19]. Learning rate for DQN_{safety} is set to 0.00025, while for DQN_{speed} its 0.0025 respectively. Both the DQNs are trained using the Double-DQN algorithm with target networks which are updated every 1000 steps. However, since DQN_{safety} of *Agent2* has an LSTM layer, it uses a mini-batch of 32 experiences, each experience consisting of 4 timesteps.

At each time step, one of the objectives is randomly selected for exploration. If objective o_{safety} is chosen for exploration. then objective o_{speed} is no longer considered for action selection (described in Algorithm 2). Each DQN uses an ϵ greedy approach for action selection, where the action is selected randomly with a probability of ϵ, else an action with the highest Q-value is selected. For DQN_{safety}, the ϵ is initialized to 0.9 and it linearly decreases to a minimum value of 0.3 over 4,00,000 steps. For DQN$_{\text{speed}}$, ϵ decreases from 0.8 to 0.1 linearly over 4,00,000 steps.

6.3 Simulation Details

The proposed method is trained and evaluated in CARLA simulator [18], an open source simulator for autonomous driving. A custom urban environment consisting of two-way roads, four-way unsignalized intersections with crosswalks is developed in the simulator. The simulator provides the environment and ego vehicle information and receives desired speed values for the ego vehicle as input. The simulator time step is set at $t = 0.1sec$.

At the beginning of each episode, the ego vehicle is spawned at the start position and follows a list of waypoints generated by the simulator to reach its destination. The desired speed limit for the ego vehicle is set at $8\,\mathrm{m/s}$.

At each time step, a total of up to 30 pedestrians exists within a vicinity of 35 meters around the ego vehicle (in front) with random goals and velocities between 0.4-$1.2\,\mathrm{m/s}$. A pedestrian crossing factor (set to 0.8) is used, indicating how many pedestrians can cross the road (80%). Pedestrians moving farther away (40 meters) from the ego vehicle are removed and an equal number of new pedestrians are spawned to make sure busy pedestrian traffic exists around the ego vehicle.

6.4 Results

As described in Section 6.1, two MORL agents with different neural network architectures are investigated in this work. *Agent1* uses convolutional layers followed by fully connected layers for DQN_{safety}. However, *Agent2* adds a LSTM layer after the convolutional layers adding memory to the DQN_{safety} network. Both agents use same network structure for their DQN_{speed}.

Furthermore, the proposed MORL agents are compared with two SORL agents as well, which are based on our previous work in [4] but with an improved reward function. Both these SORL agents use a single DQN to satisfy both the objectives and learn the desired policy. *Agent1* of SORL has a similar neural network architecture as that of MORL *Agent1*, where the *environment state* (3-D multi-layered grid) information is passed through convolutional layers followed by fully connected layers. However, the

ego vehicle state (its current speed) information is concatenated with the first fully connected layer and passed to the output layer. *Agent2* of SORL, uses two separate LSTMS: one for the *environment state* and other for *ego vehicle state*. The *environment state* is passed through convolutional layers to the first LSTM and the *ego vehicle state* is directly passed to the second LSTM. Outputs of both LSTMs are then concatenated and passed to fully connected layers followed by the output layer. Since a SORL agent expects reward as a scalar number, the safety (r_{safety}) and speed (r_{speed}) rewards described in Section 5.2.2 are added up and passed as a single reward value to the SORL agents. In summery, four variants are compared, two SORL agents ($Agent1_{CNN}$, $Agent2_{LSTM}$) and two proposed MORL agents ($Agent1_{CNN}$, $Agent2_{LSTM}$).

All the agents are tested for 100 episodes in the known environment (training environment) and on unseen new environments. *Town01* and *Town04* of CARLA simulator are used as unseen environments. *Town01* is similar to the custom training environment but consists of three-way intersections. While *Town04* is an urban environment with four-way roads. The agents are compared and evaluated using the following metrics:

- **Collision free episodes:** Percentage of episodes executed without collision.
- **Success rate:** Percentage of episodes where agent reached the goal in time.
- **Distance travelled:** Average distance travelled during each episode.
- **Average Steps:** Number of steps executed in each episode.
- **Average speed:** Average speed of the agent during each episode.
- **Speed violation:** Percentage of episodes where the target speed limit was violated by the agent.
- **Average crossing duration:** This indicates how long it takes for the agent to handle an intersection, calculated in terms of percentage of steps at the intersection with respect to total steps.
- **Average Stops:** Number of times the agent brakes to stop in every episode.
- **Distance to closet pedestrian:** Average distance to closest ($< 2m$) front crossing pedestrian in the same lane.

6.5 Discussion

The test results for all the agents on seen and unseen environments are presented in Table 1. The proposed MORL agents clearly outperforms the SORL agents with respect to each measure. As mentioned in the above sections, collision avoidance is the foremost objective of an autonomous vehicle driving in urban environments amongst pedestrians. The SORL agents already perform better with 97% and 98% collision free episodes, however, the MORL method shows further improvements with 100% collision free episodes for both the agents in known as well as unknown environments. Another interesting aspect is the number of times braking is performed. The SORL performs more braking to stop compared to MORL. This is because SORL agents use a scalar reward and are always trying to balance the penalties/rewards received for near collisions and ego vehicle speed. However, the MORL agents perform less stopping actions resulting in more natural driving. The MORL agents show 100% success rates in known environments, but in unknown environments their success rate is 98%. All the agents avoid any speed violations indicating that the ego vehicle speed is always within the speed limits. However, agents with a LSTM layer maintain higher speeds since they have memory and can anticipate the intentions of surrounding pedestrians better. The advantage of having a memory is also reflected in terms of lesser waiting times at the intersections and higher pedestrian distance maintained by LSTM based agents.

7 CONCLUSION

This work formulates autonomous navigation in urban environments amonsgt pedestrians as a multi objective reinforcement learning problem. Two conflicting objectives are considered in this work, namely, *safety* and *speed*. Two variants of the proposed method are trained in a custom urban environment and are compared with similar single objective reinforcement learning agents in known and unknown environments. All the agents are thoroughly compared and evaluated and the results confirm the better performance of the proposed method.

For future work, we plan to extend the method to incorporate more objectives such as considering traffic rules and passenger comfort. We also plan to extend the approach for a truly urban environment consisting of other vehicles along with pedestrians.

8 ACKNOWLEDGMENT

This work was funded under project CAMPUS (Connected Automated Mobilty Platform for Urban Sustainability) sponsored by Programme d'Investissements d'Avenir (PIA) of french Agence de l'Environnement et de la Maîtrise de l'Énergie (ADEME).

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles", In *IEEE Transactions on Intelligent Vehicles*, pp. 33-55, 2016.

[2] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman and N. Muhammad, "A Survey of End-to-End Driving: Architectures and Training Methods", *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-21, 2020.

Environment	Known Environment				Unknown Environments			
Agent	Single Objective		Multi Objective		Single Objective		Multi Objective	
Architecture	$Agent1_{CNN}$	$Agent2_{LSTM}$	$Agent1_{CNN}$	$Agent2_{LSTM}$	$Agent1_{CNN}$	$Agent2_{LSTM}$	$Agent1_{CNN}$	$Agent2_{LSTM}$
Collision Free Episodes (%)	97%	98%	**100%**	**100%**	96%	98%	**100%**	**100%**
Success Rate (%)	95%	98%	**100%**	**100%**	95%	97%	**98%**	**98%**
Distance Travelled (m)	142.48	145.1	**152.1**	**152.1**	118.1	118.96	**122.01**	**123.0**
Average Steps	779.62	603.4	**721.1**	**598**	642	641.2	**702**	**711**
Average Speed (m/s)	3.46	5.07	**5.6**	**6.01**	3.43	5.12	**5.52**	**5.98**
Speed Violation (%)	0%	0%	**0%**	**0%**	0%	0%	**0%**	**0%**
Crossing Duration (%)	77.19%	59.3%	**70%**	**48.89%**	61%	52.22%	**60.2%**	**48.88%**
Average Stops	11.51	7.01	**6.11**	**6.11**	9.89	6.55	**6.48**	**5.99**
Closet Pedestrian Distance (m)	0.46	0.61	**0.69**	**0.82**	0.41	0.6	**0.67**	**0.75**

Table 1: Performance comparison of all agents in known and unknown environments

[3] N. Deshpande and A. Spalanzani, "Deep Reinforcement Learning based Vehicle Navigation amongst pedestrians using a Grid-based state representation", In *Intelligent Transportation Systems Conference (ITSC)*, pp. 2081-2086, 2019.

[4] N. Deshpande, D. Vaufreydaz and A. Spalanzani, "Behavioral decision-making for urban autonomous driving in the presence of pedestrians using Deep Recurrent Q-Network", In *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 428-433, 2020.

[5] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria Reinforcement Learning", In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pp. 197–205, 1998.

[6] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms", *Machine Learning*, vol. 84, pp. 51-80, 2010.

[7] D. C. K. Ngai and N. H. C. Yung, "Automated Vehicle Overtaking based on a Multiple-Goal Reinforcement Learning Framework", In *Intelligent Transportation Systems Conference (ITSC)*, pp. 818-823, 2007.

[8] C. Li and K. Czarnecki, "Urban Driving with Multi-Objective Deep Reinforcement Learning", In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[9] H. Bai, S. Cai, N. Ye, D. Hsu and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd", In *International Conference on Robotics and Automation (ICRA)*, pp. 454-460, 2015.

[10] T. Bandyopadhyay, K. S. Won, E. Frazzoli, and D. Hsu, "Intention Aware Motion Planning", In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pp. 475–491, 2012.

[11] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee and D. Manocha, "PORCA: Modeling and Planning for Autonomous Driving Among Many Pedestrians", *IEEE Robotics and Automation Letters*, vol. 3, pp. 3418-3425, 2018.

[12] M. Barbier, C. Laugier, O. Simonin and J. Ibañez-Guzmán, "Probabilistic Decision-Making at Road Intersections: Formulation and Quantitative Evaluation", In *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 795-802, 2018.

[13] F. Camara, R. Romano, G. Markkula, R. Madigan, N. Merat and C. W. Fox, "When Should the Chicken Cross the Road? - Game Theory for Autonomous Vehicle - Human Interactions", In *International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS)*, 2018.

[14] C. W. Fox, F. Camara, G. Markkula, R. A. Romano, R. Madigan and N. Merat, "Empirical game theory of pedestrian interaction for autonomous vehicles", In *International Conference on Methods and Techniques in Behavioral Research*, 2018.

[15] K. Mokhtari and A. R. Wagner, "Pedestrian Collision Avoidance for Autonomous Vehicles at Unsignalized Intersection Using Deep Q-Network.", *arXiv preprint arXiv:2105.00153*, 2021.

[16] K. Mokhtari and A. R. Wagner, "Safe Deep Q-Network for Autonomous Vehicles at Unsignalized Intersection.", *arXiv preprint arXiv:2106.04561*, 2021.

[17] Z. Qiao, Z. Tyree, P. Mudalige, J. Schneider and J. M. Dolan, "Hierarchical Reinforcement Learning Method for Autonomous Vehicle Behavior Planning", In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 6084-6089, 2020.

[18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, "CARLA: An Open Urban Driving Simulator", In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, pp. 1-16, 2017.

[19] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude", *OURSERA: Neural Networks for Machine Learning*, 4, 26-31.