Strategies for Simulating Pedestrian Navigation With Multiple Reinforcement Learning Agents

Francisco Martinez-Gil · Miguel Lozano · Fernando Fernández

Received: date / Accepted: date

Abstract In this paper, a new Multi-agent Reinforcement Learning (MARL) approach is introduced for the simulation of pedestrian groups. Unlike other solutions, where the behaviors of the pedestrians are coded in the system, in our approach the agents learn by interacting with the environment. The embodied agents must learn to control their velocity, avoiding obstacles and the other pedestrians, to reach a goal inside the scenario. The main contribution of this paper is to propose this new methodology that uses different iterative learning strategies, combining a Vector Quantization (state space generalization) with the Q-learning algorithm (VQQL). Two algorithmic schemas, Iterative VQQL (ITVQQL) and Incremental (INVQQL), which differ in the way of addressing the problems, have been designed and used with and without transfer of knowledge. These algorithms are tested and compared with the VQQL algorithm as a baseline in two scenarios where agents need to solve well-known problems in pedestrian modeling. In the first scenario, agents in a closed room need to reach the unique exit producing and solving a bottleneck. In the second, two groups of agents inside a corridor need to reach their goal that is placed in opposite sides (they need to solve the crossing). In the first scenario, we focus on scalability, use metrics from the pedestrian modeling field, and compare with the Helbing’s social force model. The emergence of collective behaviors, that is, the shell-shaped clogging in front of the exit in the first scenario, and the lane formation as a solution to the problem of the crossing, have been obtained and analyzed. The results demonstrate that the proposed schemas find policies that carry out the tasks, suggesting that they are applicable and generalizable to the simulation of pedestrians groups.

Francisco Martinez-Gil · Miguel Lozano
Departament d’Informàtica, Escola Tècnica Superior d’Enginyeria (ETSE), Universitat de València, Avda. de la Universidad s/n 46100 Burjassot, Valencia, Spain
E-mail: Francisco.Martinez-Gil@uv.es, Miguel.Lozano@uv.es

Fernando Fernández
Department of Computer Science, University Carlos III of Madrid, Avda. de la Universidad 30, 28911 Leganés, Madrid, Spain
E-mail: ffernand@inf.uc3m.es
1 Introduction and related work

Pedestrian simulation has engaged the attention of researchers over the past few decades. Different technical areas, such as architecture, civil engineering, and game development, can benefit from the simulation of groups of pedestrians, in order to check the capacities of the facilities in a building, to prevent accidents and/or disasters, or to give realism to simulated urban scenarios. Pedestrian dynamics has basically two main perspectives: microscopic and macroscopic. Microscopic models consider individual features, such as local perceptions and interactions, while macroscopic ones are described by using global functions such as flows or densities. In recent years, the attention has been focused on microscopic models, because they have desirable properties. First, the collective effects observed in the motion of pedestrian crowds are a direct consequence of the microscopic dynamics [40]. Second, only the microscopic approaches seem to allow higher-level decision-making without major modifications of the basic behavioral model [38]. In this way, several microscopic pedestrian models have been developed, such as cellular automata models [15], force-based models [22, 35], rule-based models [37], queuing models [30], models based on psychological [49] and cognitive factors [44], and models oriented towards crowd simulation [46, 57]. Other models have been designed and calibrated using empirical data collected from video sequences or from experiments with real pedestrians [39, 41, 8, 53, 5].

In this work a novel approach to addressing the problem of simulating groups of pedestrians, based on Multi-Agent Reinforcement Learning (MARL) techniques is presented. Instead of classical approaches, where the designer writes and calibrates the behavior rules of the agents (perception–action), in our case, the embodied agents must learn individually these rules to control their velocity in a wide range of collision situations. The aim is to demonstrate empirically, that MARL is a suitable framework for generating plausible pedestrian simulations (not to reproduce real crowd behaviors) using learned microscopic interactions. Plausibility means the generation of visually realistic navigation situations, analogous to those observed in real human pedestrians, although the position and velocities can be different from them. The result is a simulation that produces a believable visual appearance, adequate for representing secondary actors in videogames and civil/military simulators or computer graphic applications designed for training in specific tasks (known as serious games). For this purpose we have used two scenarios. The first represents a closed room with a single door (exit) that the agents have to reach [17]. This scenario constitutes an uni-directional flow at a bottleneck where arch-like blockings and cloggings appear. Due to the multiplicity of interactions among the agents, it is adequate to analyze in depth the learned behaviors. The second shows a crossing of two groups of agents inside a narrow corridor. In real situations (under normal conditions), the formation of lanes is expected as an emergent collective phenomena. These lanes consist of pedestrians walking in the same direction [17]. The generation of emergent collective phenomena is a quality indicator for models and simulations of pedestrian groups and has been studied in the pedestrian modeling field [18].
The MARL framework proposes several iterative learning schemas based on the VQQL algorithm [9], which combines the use of Vector Quantization [16] as the generalization method for the state space with Q-learning [58]. The different schemas try to accelerate the learning, introducing transfer learning capabilities. The framework assumes, as in the real world, that decision making is distributed, without cooperation nor communication protocols, and that there are only limited action and perception capabilities.

The expected benefits of the application of the MARL approach to the pedestrian simulation problem can be summarized as follows:

1. Low computational cost associated to the agents' behavior: the learning process and the corresponding simulation are separated in time, so once the system has learned (off-line), the simulator can easily import the behaviors (policies) using lookup tables. This is an important property for the scalability of the simulation in the number of agents.

2. The richness of the group behavior in terms of its variability, as each agent learns different rules depending on their own learning experiences. This is an important point, as the heterogeneity of individual behaviors in a group has a large effect on pedestrian flow characteristics [6].

3. Model-free design of the problem: a model of the environment (including the agent’s model of behavior) is not assumed. On the contrary, in many classical approaches, the agents’ behaviors have to be provided by the model [25].

4. Emergent collective behaviors: The MARL framework offers examples in several domains of emergent collective behaviors which are consequences of individual decisions [34, 10]. Other studies in pedestrian dynamics [20, 40] have stated the main role of learning in the emergence of these collective behaviors in real pedestrians.

Although RL is very well-known in the machine learning community, it is not so common in the field of simulation and is almost anecdotal in the pedestrian dynamics domain. An approximation to navigational problems using learning is the layered intelligence model [2] for path planning navigation to different goals. It is not a microscopic approach, so it does not involve individual agents who learn. RL has been used for learning a centralized control for a Multi-Agent navigation system but, as in the previous paper mentioned, RL has not been used to learn individual behaviors [26]. A closer approach to our paper was introduced by Torrey et al. [55], who proposed a microscopic learning framework for learning behaviors that are a mix of socialization and goal-oriented traffic. Their approach focuses on the learning process but does not address the problem of the scalability in the number of agents nor the analysis of the learned behaviors in terms of their adequacy to pedestrian dynamics and its properties. Inside the pedestrian simulation field, our work could be also related to those of the Pettre’s autonomous virtual humans [36] and the multi-agent navigation studies of the Gamma group at the University of North Carolina [3] in terms of the size of the simulated groups and the type of studied navigational problems.

Our previous work in this area began with the development of a discrete RL model oriented to study the scalability in the problem of controlling the navigation of pedestrian agents [31]. That introductory work assumed discrete state spaces, what simplified the problem, but limited its usability. In the work [33], the iterative models used in this article were introduced to tackle continuous state
spaces. In that previous work, however, knowledge transfer in the learning process was not studied, and the learning processes, together with the simulation results, were evaluated in a single scenario with preliminary models of the physics of the environment. More recently, in the authors’ paper [32], the calibration of the physics engine used in both learning and simulation environments was studied in depth. Such calibration was oriented to find the values, from actual pedestrian dynamics studies, that produce realistic local interactions (e.g., collision friction and response). To undertake this calibration we use the scenarios provided by pedestrian dynamics literature so our results could be also compared with them. This paper is not just a compendium of previous works, but a step forward in crowd simulation research which makes the following contributions: a) the application and analysis of different knowledge transfer approaches in the learning process, b) an extensive performance analysis of the different learning algorithms proposed c) a new learning+simulation scenario is studied, where two different groups of agents are placed facing each other in a corridor d) a set of new micro+macro simulation metrics to analyze the quality of the behaviors produced in both scenarios, e) a comparison with a well-known pedestrian model (Helbing model).

The rest of the present paper is structured as follows. Section 2 formalizes the pedestrian navigation problem as a MARL. Section 3 summarizes the generalization approach we have followed. Section 4 describes our MARL approach, while Sections 5 and 6 analyze the empirical results. Section 7 exposits the conclusions and proposes avenues for future research.

2 Modeling pedestrian navigation from a MARL point of view

Reinforcement learning has a solid foundation inside the Machine Learning field. It is applied for the optimization problems that can be modeled as Markov Decision Processes (MDP) [48]. An MDP is a 4-tuple constituted by an state space, $S$, an action space, $A$, a probabilistic transition function $P : S \times A \times S \rightarrow [0, 1]$ and the reward function $R : S \times A \rightarrow \mathbb{R}$. In a state, the decision process can select an action from the action space. Each decision is accompanied by an immediate reward that represents the value of the decision taken in this state. The goal is to find a policy, that is, a mapping between states and actions that provide the maximum discounted expected reward $V(s) = E\{\sum_{t=0}^{\infty} \gamma^t r_t\}$ in each state of the space state. The expected maximum reward is represented as a value function $V(s)$ (named $Q(s,a)$ in the control problems because of the additional dependence of the action) and the discount parameter $\gamma$ sets the influence of future rewards ($0 \leq \gamma \leq 1$) and $r_t$ is the immediate reward in time $t$. When the probabilistic transition function $P$ is unknown, RL algorithms are useful to calculate the optimal value function $Q^*(s,a)$. One of the most known RL algorithms is Q-learning [58]. In its tabular version, Q-learning uses a table (called Q) to represent the value function. For each entry of Q, the expected accumulated reward of being in state $s$ and doing action $a$ is stored. The process of updating the Q table with a new immediate reward $r_t$ at instant $t$ is performed using the Q-learning update equation shown in Equation 1.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a\{Q(s_{t+1},a)\} - Q(s_t,a_t)], \quad (1)$$

where $\alpha$ is the learning rate.
Although the natural extension of the single agent RL problem to Multi-Agent systems are the Markov Games [29], the use of joint actions of all learning agents makes this approach difficult to use in practice, due to the combinatorial explosion of the action space. On the other hand, if the goal of each agent is to maximize their own reward, an approach based on independent learning agents is adequate. In fact, independent multiagent Q-learning usually converges even when theoretical guaranties, such as the stationarity of the environment, are lost [54, 47]. Several problems in other domains have been addressed with this approach, such as robot navigation [34, 42], cooperative problems [7], and the Keepaway Soccer task [47, 14].

2.1 The scenarios

The first pedestrian navigation problem consists of a group of agents inside a closed room with a door. The agents have to learn how to reach the door and leave the room. The second is a narrow corridor in which two groups of four agents each have to cross to reach the opposite end. In order to resemble the model of pedestrians, the agents are constrained to move on the plane with a maximum velocity of 1.8 m/s (a typical value in pedestrian models [53]) and are placed in the centre of a bounding circumference with radius 0.3 meters that represents a middle size “body” according to pedestrian models. This representation of the body takes into account the two primary human measurements used in pedestrian models: body depth and shoulder breadth [13, 46].

The environments are modeled as a two dimensional continuous plane. In the first experiment, the virtual environment is a 225 square meters room with an aperture of 0.8 meters (which represents the door) in the centre of one of the sides. The limits of the square are defined by walls. In the second experiment, a corridor of 15 meters long by 2 meters wide is set. Each group of agents is placed at one end of the corridor and its goal is to reach the opposite end. Thus, a crossing must be produced. (see Figure 1).

2.2 State and action spaces

The definition of the states that the agents sensorize follows a deictic representation. The central premise underlying a deictic representation is that the agent only registers information about objects that are relevant to the task at hand [1, 60]. The selection of features that represent the state for the agent is critical for the success of learning. We have chosen features that provide local information about the agent’s kinematical state, the neighboring agents, and the nearest walls, modeling the real situation of a pedestrian inside a group. This deictic representation of the state space based on local information is particularly suitable for RL navigational tasks as the study of Lane et al. [27] reveals. As a result, the state for each agent is described by the features shown in Table 1. When representing the state space of a scenario, there is always a trade-off between representing accurately the kinematical situation around the agent and the burden of dimensionality. The chosen features have been used previously in pedestrian models and considered as relevant for the kinematical description of the pedestrian [39] or for characterizing
the imminence of a collision [4]. The number of sensorized neighboring agents and neighbor objects is configurable. In the first scenario, the number of maximum sensorized neighbors is seven, and can be varied in one of the algorithmic schemas (see the next section). This sets the number of different possible configurations of the state space to eight (sensorizing 0, 1, 2, 3, 4, 5, 6 and 7 neighbors). In the second scenario, the number of maximum sensorized neighbors is four. A narrow corridor is more limited in terms of movements than a wide room, therefore we have considered relevant a number of neighbors that could be in contact directly with the agent. Four neighbors homogeneously distributed in the vicinity of an agent occupy practically all his sides. The number of sensorized static objects (walls) is always fixed at two. The maximum number of features describing one state space is 28 in the first scenario and 23 in the second scenario.

Table 1 Description of the features of the agent’s state. The reference line joins the agent’s position with its goal position

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sag</td>
<td>Module of the velocity of the agent.</td>
</tr>
<tr>
<td>Av</td>
<td>Angle of the velocity vector relative to the reference line.</td>
</tr>
<tr>
<td>Dgoal</td>
<td>Distance to the goal.</td>
</tr>
<tr>
<td>Srel&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Relative scalar velocity of the i&lt;sup&gt;th&lt;/sup&gt; nearest neighbor.</td>
</tr>
<tr>
<td>Dag&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Distance to the i&lt;sup&gt;th&lt;/sup&gt; nearest neighbor.</td>
</tr>
<tr>
<td>Aag&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Angle of the position of the i&lt;sup&gt;th&lt;/sup&gt; nearest neighbor relative to the reference line.</td>
</tr>
<tr>
<td>Lag&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Label to identify the group that the neighbor belongs to.</td>
</tr>
<tr>
<td>Dob&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Distance to the j&lt;sup&gt;th&lt;/sup&gt; nearest static object (walls).</td>
</tr>
<tr>
<td>Aob&lt;sub&gt;j&lt;/sub&gt;</td>
<td>Angle of the position of the j&lt;sup&gt;th&lt;/sup&gt; nearest static object relative to the reference line.</td>
</tr>
</tbody>
</table>
The agent’s actions modify its velocity vector. The variation of this velocity vector has been used to control the trajectories in pedestrian models [4]. In an agent’s decision, the actions are taken in pairs, which modify the speed (increasing or reducing) and the orientation of the velocity vector (clockwise or counterclockwise), respectively. There are eight different ratios plus the ‘no operation’ option for both the speed and the angle, resulting in 81 possible combined actions. In the case of the speed, the possible additions or subtractions to the current velocity are the fractions $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1 of a reference value, specifically, one-half of the maximum velocity. In the case of the angle of the velocity (turn), the possible actions are the addition or subtractions of the fractions $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1 of the reference value $\frac{\pi}{4}$. The use of a constant value to add or subtract to the velocity vector makes the learning problem easier, because the actions always have the same effect regardless of the kinematical situation. The agent cannot have a negative speed: when the deceleration produces this, the final speed is always 0 m/s. Besides, the speed cannot grow to exceed the maximum speed (1.8 m/s). Therefore, each action over the modulus of the velocity has a range of values which have the same effect because the result would otherwise exceed an extreme value, as Figure 2 displays. The kinematical module of the environment moves the agents across the plane using the velocity vector of each agent. The simulation is discretized in time slots of the same size, assigned to the agent’s decisions. In the simulation of a decision, the agent’s velocity remains constant unless a crash occurs. The kinematical module actuates following a configurable clock signal so that the user can specify the number of decisions per second that the agent must take.

![Fig. 2](image.png)

Fig. 2 The range of operation for the actions that modify the speed with a reference value fixed at one-half of the maximum speed $V$. The grey rectangles show the range of the speed where the action has the same effect. Above: operation range for the actions that decelerate. Below: operation range for the actions that accelerate.

1 This feature is used in the crossing experiment only
2.3 Modeling the agents’ behavior

The behavior of the agents is modeled according to the immediate payoffs listed in Table 2. In the first scenario, the payoff function models the prevention of collisions as an important task that a navigation controller must take in account. However, crashing against another agent is punished less than crashing against a wall, to lessen the shading effect of the frequent crashes in the bottleneck over the rest of the immediate rewards. The ignorance about the effects of other interactions is modeled with an immediate reward of value 0. In the second scenario, the immediate reward is more simple. It ignores crashes against agents and walls. In this domain, empirical tests were carried out with the rewards of the first scenario giving worse performance than the test with the proposed reward.

<table>
<thead>
<tr>
<th>Table 2 Description of the values of the immediate rewards for both scenarios.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Scenario (Agents in a room)</strong></td>
</tr>
<tr>
<td>Crash against other agent</td>
</tr>
<tr>
<td>Crash against a wall</td>
</tr>
<tr>
<td>Reach the goal</td>
</tr>
<tr>
<td>Default</td>
</tr>
<tr>
<td><strong>Second Scenario (Crossing)</strong></td>
</tr>
<tr>
<td>Reach the goal</td>
</tr>
<tr>
<td>Default</td>
</tr>
</tbody>
</table>

3 State space generalization

The states are generalized using Vector Quantization (VQ), which has been demonstrated to be an accurate approach for state space generalization and transfer learning in RL frameworks [9, 14, 24]. A vector quantizer $V_Q$ of dimension $K$ and size $N$ is a mapping from a vector space (in this paper, the state space) in a $k$-dimensional Euclidean space, $\mathbb{R}^k$, to a finite set $C$ containing $N$ states. A sensorized state is aggregated to its nearest state in $C$, also called its prototype. Thus, given $C$ and a state $x \in \mathbb{R}^k$, then $V_Q(x) = \arg \min_{y \in C} \{\text{dist}(x, y)\}$. The prototypes, that is, the members of $C$, are found using the Generalized Lloyd Algorithm (GLA) [28] which selects samples of the dataset randomly for the initial configuration of the prototypes. The GLA algorithm together with the Euclidean metric, define the Voronoi regions of the state space. Vector quantization makes possible the use of a table for representing the value function, and therefore the use of classic TD algorithms, such as Q-learning or Sarsa.

Vector Quantization for Q-Learning (VQQL) [9, 14] is a learning schema that uses VQ as the generalization method for the state space and the tabular version of Q-Learning for the learning process. In VQQL, given a sensorized state, $s_t$, and a selected action, $a_t$, the Q table entry to be updated is $(V_Q(s_t), a_t)$. The exploration–exploitation trade-off is modeled using an $\varepsilon$-greedy policy, although different strategies could be implemented. The complete algorithm is described in Table 3.

The final condition is defined in our case as a fixed number of iterations determined empirically.
Table 3 Description of the VQQL algorithm.

<table>
<thead>
<tr>
<th>Single-agent VQQL Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Generate the set $T$ of samples of the state space $S$ interacting with the environment using an exploratory policy.</td>
</tr>
<tr>
<td>2. Discretize the state space:</td>
</tr>
<tr>
<td>(a) Use GLA to obtain a state space discretization $C \subseteq S$ from the sample set $T$.</td>
</tr>
<tr>
<td>(b) Let $VQ : S \rightarrow C$ be the function that, given any state in $S$, returns the discretized value in $C$.</td>
</tr>
<tr>
<td>3. Learn the Q-Table</td>
</tr>
<tr>
<td>While the final condition is not reached</td>
</tr>
<tr>
<td>(i) Get an experience tuple $&lt;s_1, a, s_2, r&gt;$ by interacting with the environment.</td>
</tr>
<tr>
<td>(ii) Map the states of the experience tuple using $VQ$. Each acquired tuple of experience $&lt;s_1, a, s_2, r&gt;$ is mapped to $&lt;VQ(s_1), a, VQ(s_2), r&gt;$.</td>
</tr>
<tr>
<td>(iii) Apply the Q-Learning update function defined in Equation 1 to learn a tabular value function $Q : C \times A \rightarrow \mathbb{R}$, using the mapped experience tuple.</td>
</tr>
<tr>
<td>4. Return $Q$ and $VQ$</td>
</tr>
</tbody>
</table>

The use of VQ requires two elements. The first one, as stated in the first step algorithm, is a dataset, $T$, from which a good model of the state space can be generalized. The VQQL algorithm assumes that the dataset is significant, that is, the dataset contains the relevant information to represent the whole state space. This is not the case with our domain, where a random walk can bias the exploration towards irrelevant states. Therefore, it is not interesting to model the whole state space uniformly, but rather the point is to get the states that are informative for solving the problem. Our methodology includes a policy-guided bias, as described in the next section, to overcome this problem. The second element to gather is the number of prototypes to use, that is, the resolution of the state space. Typically, a coarse discretization composed of a reduced number of prototypes does not have enough expressiveness to represent the optimal value function. On the other hand, too many states introduce the generalization problem, although with a finite number of states. In order to fix the resolution of the space in both scenarios, two sets of experiments were made, testing different numbers of prototypes to represent the state space. In each scenario, a set of six experiments is performed with a specific number of prototypes: $k = \{512, 1024, 2048, 4096, 8192, 16384\}$, with the same configuration. In each experiment, the agents perform a series of random walks in the environment to get unbiased sensorization data. Then, the dataset is used to build the vector quantizer with a fixed number of prototypes. The sizes of the datasets ranging from 20000 samples in the case of 512 prototypes to 70000 samples in the case of 16384 prototypes. The calculated vector quantizers are different for each agent because they have collected their own dataset. Finally, each agent uses its own vector quantizer in the VQQL learning process. Figure 3 shows the results from the set of experiments performed with the first scenario (the results for the second scenario are omitted because they are equivalent). In these experiments, the agents have to learn to leave the room in a number of episodes$^2$. In each episode, the agents are placed randomly inside the room and a maximum number of steps (the same in all the episodes and for all the agents) is given to get the goal. At each step, the agent must take a decision (action) to adjust its velocity to the current local situation in the environment. In Figure 3, the $x$ axis

$^2$ The term ‘trial’ situated in the abscissa of the graphics has the same meaning that the term ‘episode’ in the text
shows the number of episodes executed, and the y axis represents the normalized percentage of successful episodes, that is, the percentage of agents that leave the room. The quality of the learned behaviors is given by the asymptotic value of the curves at the end of the process. Note that the performance increases with the number of prototypes used in the quantizer (the resolution of the quantizer). However when duplicating the number of prototypes, the time for the assignment of the dataset to the clusters in each iteration of GLA is also duplicated. The best trade-off between performance and computational cost was set to 4096 prototypes for the first scenario (agents in closed room) and 8192 for the second scenario (crossing).

Fig. 3 Learning curves for the first scenario using the VQQL algorithm and vector quantizers with different numbers of prototypes. The curves are the means of 18 learning processes (18 agents).

4 Algorithms

The application of RL to pedestrian navigation faces three challenges. The first one arises because pedestrian simulation is a multi-agent environment. Therefore, we first need to decide how many agents should learn from scratch. It is easy to understand that if we set 100 agents to learn from scratch, rarely they will be able to learn an effective policy. This is due to the fact that the learning agents could modify their policies while learning, which makes the environment non-stationary. Besides, a study of whether they should be introduced into the environment from the beginning or gradually added is needed. However, this produces a second challenge: the agents can perceive a different number of agents in different situations, and therefore the space representation should be different. Therefore, two different situations can arise:
Situation 1 (S1). The different descriptions of the state space are inherent in the incremental setting of the agents. For example, in a learning process for five agents, the space representation should consider the description of at most four neighboring agents. But in a learning process with six agents, the space representation should consider five neighboring agents.

Situation 2 (S2). The evolution of the episode in time creates a variable perception of neighbors. When the agents are gradually reaching their goal, the rest of the agents can perceive fewer neighbors around.

The third challenge is that a random exploration of the environment does not produce a representative dataset with which to generate a correct set of prototypes for the VQ space generalization method. Therefore a process for selecting better datasets is necessary.

To address these challenges, we propose an iterative learning schema. A pipeline of \( N \) learning processes are carried out over the same problem, where the knowledge learned in an iteration is transferred to the next iteration. We make two different contributions in this research. Firstly, we propose two different iterative learning schemes for Vector-Quantization based RL in pedestrian simulation, as is described in subsection 4.1. Secondly, we propose to speed up learning transferring the value function through the different iterations of the algorithms in subsection 4.2.

4.1 The iterative schemas

Two different approaches have been defined: Iterative Vector Quantization for Q-Learning (ITVQQL) and Incremental Vector Quantization for Q-Learning (IN-VQQL). They all perform different iterations of the VQQL algorithm’s basic steps (computing the vector quantizer and learning the Q table) but in different ways, as described below. Additionally, in the two schemas, the learned policies in an iteration are used to gather a policy-biased dataset for building a new state space model that will be used in the next learning process. This process is referred to in this text as policy transfer and is carried out in the point 4(b)ii of the Table 4. However, the schemas differ in the way of placing the agents in the scenario and, hence, in the number of agents learning at the same time in each iteration and with the configuration of the vector quantizer. Specifically, regarding the number of agents per iteration:

1. The ITVQQL schema puts the same number of agents for every learning iteration. Besides, it models the state space with one set of prototypes with a fixed number of features to represent the neighboring agents.
2. The IN-VQQL schema puts the agents into the stage beginning with one agent in the first iteration and incrementing by one more agent in each following iteration to reach the total number of agents in the last iteration.

Regarding the definition of the vector quantization:

1. The ITVQQL approach represents the state space using prototypes with a constant number of features, which means that different perceptions (in terms of sensorized neighbors) must be described with the same space vector and therefore with a fixed number of features (this problem appears in situations (S1) and (S2) described above).
2. The INVQQL approach uses different vector spaces to represent the different perceptions in terms of the neighborhood \((0, 1, 2, \ldots)\) neighboring agents. Therefore, a collection of vector quantizers with their respective sets of prototypes with different numbers of features is used.

The Table 4 describes the general algorithm. The first three steps initialize the counters, the quantizer, and the policy. Step 4 is the loop. Each iteration is an entire learning process where the sub-steps (b) and (c) correspond with the execution of the VQQL algorithm described in Table 3. Note that step (c) is an entire Q-learning process. In this step, an initialization of the Q-tables as well as the parameters of the exploratory policy are carried out. Next, an iterative process with a fixed number of episodes is performed to get an approximation of the optimal policy for the problem. The number of iterations of the main loop of step 4, \(N\), should be configured differently for each schema. For the INVQQL schema, the number of iterations, \(N\), coincides with the total number of agents with which the experiment was planned. For the ITVQQL schema, the number of iterations, \(N\), are enough to reach a good learning performance (see curves for the ITVQQL schema in the Figure 5, the Figure 6 for the first scenario and the Figure 8 for the second scenario. In the experiments presented in this article, the same number of iterations is used for the both schemas to set similar experimental conditions and to compare the results.

Table 4 Description of the two schemas. The bold statements mean different options in each schema as explained in the text.

<table>
<thead>
<tr>
<th>Multi-agent ITVQQL/INVQQL schemas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry: The number of iterations (N)</td>
</tr>
<tr>
<td>Return: The sets (Q_N) and (V_N) (The value table of Q-learning and the vector quantizer respectively).</td>
</tr>
</tbody>
</table>

1. \(i \leftarrow 1\)
2. **Set \(p\) to the initial number of agents in the environment**
3. For each agent, \(k \ (1 \leq k \leq p)\) set:
   - its initial vector quantizer, \(V^k_0(s) = \emptyset\)
   - its initial policy \(\pi^k_0 = \text{random}\)
4. Repeat:
   (a) **Decide whether or not to include new agents.** Set \(p\) consequently.
   (b) For each agent, \(k \ (1 \leq k \leq p)\) do:
      i. Collect a dataset \(T^k_i\) for agent \(k\) using the policies \(\pi^k_{i-1}\) with \(V^k_{i-1}\)
      ii. **Build \(V^k_i\) using \(T^k_i\) for agent \(k\) following a transfer learning strategy**
   (c) Learn \(Q^k_i \ \forall k, 1 \leq k \leq p\) and hence the policies \(\pi^k_i\) using Q-Learning (with the option of using transfer of value functions).
   (d) \(i \leftarrow i + 1\)
   Until \(i = N\)
5. Return \(Q_N\) and \(V_N\)

In the Table 4, the decisions that differentiate the schemas have been bolded and are explained in the following enumeration:

1. **Iterative VQQL (ITVQQL).** A fixed number of learning agents is set in step (2). Therefore, no new agents are included in step (4.a). There is no specific
transfer between quantizers in step (4.b.ii). The space state representation has a fixed number of features. The problem discussed in (S2) is solved by setting unobserved features to random values.  

2. Incremental VQQL with feature selection (INVQQL). The interactions between the agents are learned gradually by increasing the number of agents from one iteration to the next. It can be understood as a kind of shaping [48], where a more complex task is achieved through the learning of easier tasks, which are oriented towards the solution of the final task of ultimate interest. In step (2), there is not any agent at the beginning ($p = 0$). In (4.a), one agent is added in each iteration. The novelty here is that this schema calculates a different VQ for each state space of different dimensionality incrementally. The calculated VQs are transferred to the next iteration, point (4.b.ii), avoiding their recalculation. For instance, in the first scenario, a learning process of eight agents has inherited the previous VQs which models the space state sensorized by 7, 6, 5, 4, 3, 2, and 1 agents. In the first scenario, the number of prototypes grows in the first eight iterations (of a total of 18 iterations), modeling the eight possible different configurations of our experimental setting. In the second scenario, the number of prototypes grows during the first four iterations (of a total of 8 iterations).

An additional problem arises in (4.b.i) when using the policies learned in one state space configuration to collect data in a space with a newly added agent, and therefore, with another state space configuration. The policy transfer between spaces with different number of features is carried out using a projection. A projection can be understood as a selection of features $\Gamma : R^m \rightarrow R^s$ where $m > s$. Therefore, the projection is from the higher dimensional state space to the lower dimensional one. The set of vector quantizer $V_s$ and the value functions $Q_s$ of the previous iteration can be used to collect biased experiences in the highest dimensional state space. The difference between two spaces of different dimensions in our scenarios is always constituted by the features that describe a new neighbor agent in the higher dimension space. In practice the projection is carried out by deleting this new set of features. Such mapping between different state spaces have been evaluated successfully in previous transfer learning problems [11].

As defined above, INVQQL goes a step further because the prototypes learned in previous iterations, where the state configurations are different in terms of the neighbors perceived, are transferred to the next iteration. In this way, the whole range of different state configurations are covered. In Table 5, the properties of each schema are summarized.

4.2 Value function transfer procedures

In this section, we propose using the transfer of the value function learned in a previous iteration as the initial values for the value function of the next one. In our problem, the source task and the target task are the same but with different state representations. Our problem can be considered a simple case of those described in

---

3 In Machine Learning, many different approaches are used to fill in unobserved features. We have studied informally some of them, specifically random imputation and mean imputation, obtaining similar performances.
Table 5 Summary of the settings of the schemas.

<table>
<thead>
<tr>
<th>Feature</th>
<th>ITVQQL</th>
<th>INVQQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of prototypes</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>Number of features per prototype</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>Number of agents per iteration</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>Inter-iteration policy transfer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Inter-iteration prototype transfer</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Inter-iteration value function transfer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[50]. In that paper, the authors focused on the problem of representation transfer, that is, how to transfer the learned knowledge between tasks that differ in the function approximator (the state space generalizer) or in the learning algorithm. Specifically, they call the transfer of an inter-task value function between different representations of the state space complexification. The transfer proposed here is a form of complexification and it is described in the Table 6. The idea is to initialize the target Q table with the learned values of the source Q table. The transfer is performed using a metric for the similarity between the prototypes of the source and the target vector quantizers. Thus for each prototype of the quantizer associated to the target Q table, the nearest prototype of the quantizer associated to the source Q table is calculated. Then the values of the source Q table are transferred to the target Q table using this association. In the experiments, the algorithm uses the Euclidean metric to determine the similarity between the sets of prototypes. In our case, the transfer requires that the two Q tables (the source and the target) have the same dimension and parametrization. Although the initial values are the same as those of the learned values of the previous iteration, the set of prototypes used in the new configuration is different than the previous ones and the epsilon-greedy policy begins the new iteration with high rates of exploration. Therefore, the new process is not in any case a mere continuation of the previous learning process.

Simple Complexification with a Q-table

1. Train with a source representation and save the learned Q-table and the vector quantizer $Q_{source}, V_{source}$
2. for each prototype $q^{i}_{target} \in C_{target}$
   - Find prototype $q^{j}_{source} \in C_{source}$ $\min_{j} ||q^{i}_{target} - q^{j}_{source}||$
   - $Q_{target}(q^{i}_{target}, action_{k}) = Q_{source}(q^{j}_{source}, action_{k}) \forall action_{k}$

Table 6 Description of the value function transfer procedure. In the implementation, the euclidean norm is used.

Several metrics to evaluate transfer learning methods have been proposed [51]. In the present paper, the following metrics are considered for testing in the performance curves:

1. Jumpstart: The improvement in the initial performance of an agent.
3. Time to Threshold: The learning steps needed by the agent to achieve a pre-specified performance level.

The results of the evaluation will be shown in the next section (see Figure 7).

5 Experimental set-up and performance results for the learning process

This section describes the learning processes in the two scenarios in terms of mean performance. First, the results for the agents in a closed room scenario are described. Secondly, the results for the crossing experiment are exposed.

5.1 Results for the first scenario

In all the learning schemas described above, the number of iterations performed is 18 ($N = 18$). Each iteration has been set empirically to 50,000 episodes. An episode ends when all the agents reach the goal or when a maximum of 150 decisions have been taken. The learning algorithms have the same parameter configuration in all processes for all agents: $\gamma = 0.9$, $\alpha = 0.4$, and an initial value of $\epsilon = 0.4$ which is decremented exponentially with the number of episodes following the expression $\exp\left(\frac{-\text{episode}}{k}\right)$, where $k$ is a constant to regulate the decay. All the agents learn simultaneously. That means that all the agents are in the same step of the same episode. This configuration favors a similar progress in learning for all the agents, palliating the intrinsic non-stationarity nature of the environment.

The dataset gathered in each iteration to learn the vector quantizer (step 4.b.i of algorithm in the Table 4) is generated using an $\epsilon$-greedy policy to avoid the overfitting of the training data to the previously learned policy: if the learned policy is used fully greedy, the state space visited may be excessively restricted by the policy, so adding a small amount of noise in the action selection can reduce this effect. Before using the GLA algorithm, the collected data are standardized (each feature has zero mean and standard deviation equal to 1). In Figure 4, the effect of the refinement of the VQs along an execution of the ITVQQL schema in the first scenario is shown. The graphics display the values of two features of the 4096 prototypes that constitute a VQ. Specifically, in the X-axis the speed of the agent is represented while in the Y-axis the distance to the goal. Because the features are standardized values, negative values for distances and speeds appear. The prototypes of the first iteration (on the left) and those of the last iteration (on the right) are displayed. In the first, the prototypes are distributed quite homogeneously in the space because the data are collected using a random policy as indicated in the point 3 of the algorithm in the Table 4. At the graphic on the right, a big density of prototypes in the region of low speed and low distance to the goal is observed, showing the bias in the collected data generated for a learned policy. The prototypes located in the high density region are probably generalizing states of agents placed near the door, where low distances are associated to low speeds due to the bottleneck.

\footnote{In the experiments, we will show that 18 iterations is a value large enough to ensure convergence in all the proposed scenarios}
Fig. 4 Visualization of the prototypes calculated for the first iteration (left) and the last iteration (right) of the ITVQQL schema in the room scenario. The speed (in the x-axis) and the distance of the agent to the goal (in the y-axis) features are displayed. The prototypes of the left graphic are calculated using data collected from a random policy. The prototypes of the right graphic are calculated using data collected with a learned policy correspondent to the penultimate iteration.

Figures 5 and 6 summarize the results of the learning processes for, respectively, the schemas without and those with transfer of value function. Each point represents the averaged final performance reached by the agents when they use a greedy policy over the learned value functions in that iteration, that is, an iteration of the loop in step 4 of the algorithm of Table 4. For each iteration, a simulation of 100 episodes without learning calculates the performance of the value functions. The performance is measured in terms of the percentage of successful episodes in the simulation. In Figure 7 the shapes of the performance curves for one learning process along all the episodes can be seen. The mean performance given by the learned value functions calculated in this process will constitute a point displayed in the graph of Figures 5 and 6. As an example, the ITVQQL averaged curves displayed in Figure 7 correspond to the iteration number three, and the averaged performance of the resulting value functions becomes the point of the abscissa number 3 in Figures 5 and 6. The VQQL algorithm is included for comparison. The VQQL is not iterative, and the graph displays a learning process of 900,000 episodes, which is the total number of episodes completed by the other schemas in the 18 iterations. Each point in the VQQL curve is calculated using the state of the value functions in the corresponding number of episodes (first iteration equals 50,000 episodes, second equals 100,000 episodes, and so on). Table 7 shows the specific configuration of the parameters described in Table 5 for the experiments.

In the ITVQQL schema, the first point in the curve shows less performance than INVQQL because the INVQQL uses one agent while the ITVQQL uses 18 agents. Note in Figure 5 that after a short number of iterations, the learned policies do not improve the performance for this domain. A similar curve appears in Figure 6, although a soft increment with fewer oscillations is seen.

The INVQQL schema works with a state space with eight possible feature configurations (from 0 to 7 sensorized neighbors), creating eight sets of 4096 prototypes for quantifying each state space configuration. The first eight points in both graphs decrease in their performance because the dimensionality of the state
specific settings for the learning experiments in the first scenario.

<table>
<thead>
<tr>
<th>Key</th>
<th>TVQQL</th>
<th>NVQQL</th>
<th>VQQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>50000</td>
<td>50000</td>
<td>90000</td>
</tr>
<tr>
<td>Iterations</td>
<td>18</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>Prototypes</td>
<td>4096</td>
<td>From 4096 to 32768</td>
<td>4096</td>
</tr>
<tr>
<td>Features per prototype</td>
<td>28</td>
<td>From 7 to 28</td>
<td>28</td>
</tr>
<tr>
<td>Agents per iteration</td>
<td>18</td>
<td>From 1 to 18</td>
<td>18</td>
</tr>
<tr>
<td>Inter-iteration prototype transfer</td>
<td>0</td>
<td>4096</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 5 The learning process performance for all the schemas without transfer of the value functions for the closed room scenario. The performance is calculated using a greedy policy over the learned value functions. A simulation with greedy policy without learning, with a total of 100 episodes with 18 agents per iteration was done to calculate each point. The points are sorted by iteration number inside the learning process.

The VQQL curve in Figures 5 and 6 represents the performance of one iteration of the algorithm with a number of episodes equal to the addition of the episodes for all the iterations in any other schema. Obviously there is no transfer of value functions. The ε parameter for exploration is adjusted to the number of episodes as well as the learning ratio parameter α. The first point of the curve (iteration 1) is lower than in the rest of the schemas because at this point this algorithm has a high rate of exploration. In comparison with the curves of Figure 3 that also correspond to a VQQL learning process (although with different configurations), the performance results are higher here because we display the mean performance of a greedy policy which exploits the value function. On the other hand, the graph of Figure 3 displays the mean performance of a process that uses exploration, especially in the early episodes, which causes a decrease in the total performance.
The same effect explains the lower values in Figure 7 of the iterative schemas compared to the values reached in Figures 5 and 6.

Figure 7 compares, for each schema, the performance curves of the learning processes with and without transfer in the last iteration. The three metrics introduced in § 4.2 (i.e., jumpstart, asymptotic performance and time to threshold) are easily checked in the graphs. The jumpstart is very clear in both schemas. This means that the transferred knowledge accelerates the new learning process. This conclusion is supported by the fact that the curves with transfer are crescent. If the transferred knowledge was not useful, the agent should unlearn it and the learning curves would have a decreasing zone. The time to threshold is also very influenced by the transfer. We have set the threshold to 0.7 (70% of success) as a limit where the good behavior begins. Note that the processes with transfer advance substantially the arrival to the threshold saving at least half the number of necessary episodes for the ITVQQL schema, being much more dramatic in the INVQQL schema. The asymptotic performance is also improved with the transfer in the ITVQQL schema and significantly in the INVQQL schema. Summarizing the transfer results, the acceleration of the learning is significant in the ITVQQL schema and especially in the INVQQL schema. In the case of the INVQQL schema, it seems connected with the fact, discussed above, that it uses, in the displayed iteration, 32,768 prototypes, while the other schemas use 4096 prototypes. Therefore, the number of states to explore and evaluate in the INVQQL schema require more episodes to get enough experience, and this problem is alleviated with the knowledge transferred.

From the previous evaluation, we can draw two main conclusions:

1. It has been empirically proved that the learning processes converge for the two proposed learning schemas, in the considered domain.
Table 8 Specific settings for the learning parameters common to all the experiments in the second scenario.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma )</td>
<td>0.9</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>from 0.3 to 0.1</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>1.0 (initial value)</td>
</tr>
<tr>
<td>( \psi )</td>
<td>1.0 (initial value)</td>
</tr>
</tbody>
</table>

2. The transfer of knowledge benefits especially the INVQQL schema in the learning task. In this schema, this high benefit reveals that the chosen number of episodes per iteration were not enough to learn properly, probably due to the size of the Q table’s being bigger than the sizes of the other schemas. The transfer of knowledge alleviates this lack of episodes.

5.2 Results for the second scenario

Eight agents \((N = 8)\) are divided in two groups that move counterflow inside the corridor. Each iteration has 50000 episodes. The INVQQL schema needs 8 iterations to insert one by one the total number of learning agents, therefore the number of iterations has been set to 8 in both schemas to be comparable in number of learning processes, as it was done in the first scenario. The dynamic of the episode has been described in the previous subsection with the first experiment. Tables 8 and 9 resume the configuration of the learning parameters.\(^5\) In this problem, a learning rate decreasing with the number of iterations was used. As indicated in Table 8 the value of \( \alpha \) begins in the first iteration to 0.3 and decreases to be 0.1 in the last iteration.

In this scenario the experiments are carried out with transfer of value functions once proved in the first scenario that it equals or increases the performance of the

\(^5\) Assuming that a soft variation in the values of the parameters produce a soft variation in the learning performance (the experiments agree with this assumption), the way of finding the values for the learning parameters consists of a coarse search inside the allowed values followed by a refinement over the candidate with better performance.
schemes and accelerates the learning process. The graphics for the performance are displayed in Figure 8 (left). The INVQQL schema begins with a 100% of performance because in the first iteration only one agent is learning. The shape is similar to that of the first experiment because the agents are introduced incrementally in the iterations. The first iteration of the VQQL schema has the lowest performance because in this stage of the learning process the exploratory policy is still dominant. As occurred in the first experiment, the ITVQQL with transfer gets higher performance in the last iteration than the rest. The asymptotic shape indicates that not all the iterations were necessary for this schema.

\[ \pi \]

\[ \psi \]

\[ \epsilon \]

\[ \text{greedy} \]

\[ \text{Policy Reuse transfer technique} \]

**Table 9** Specific settings for the learning experiments in the second scenario.

<table>
<thead>
<tr>
<th>Key</th>
<th>ITVQQL</th>
<th>INVQQL</th>
<th>VQQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>50000</td>
<td>50000</td>
<td>40000</td>
</tr>
<tr>
<td>Iterations</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Prototypes</td>
<td>8192</td>
<td>From 8192 to 40960</td>
<td></td>
</tr>
<tr>
<td>Features per prototype</td>
<td>24</td>
<td>From 8 to 24</td>
<td>24</td>
</tr>
<tr>
<td>Agents per iteration</td>
<td>8</td>
<td>From 1 to 8</td>
<td>8</td>
</tr>
<tr>
<td>Inter-iteration prototype transfer</td>
<td>0</td>
<td>8182</td>
<td>0</td>
</tr>
</tbody>
</table>

In order to accelerate the learning process we have used Policy Reuse [12] as a knowledge transfer technique, in a similar way it is used in other works [52,56]. A policy \( \pi_0 \) is used as a bias (or advice) in the exploration-exploitation trade-off. Specifically, the policy \( \pi_0 \) is used with a probability \( \psi \) that decays exponentially in the number of episodes while the \( \epsilon \)-greedy exploratory policy is used with probability \( 1 - \psi \). The probability function is shown in Equation 2 and its evolution
in time is displayed in Figure 8. In our problem, the policy \( \pi_0 \) always suggest the use of an action that drives the agent towards one side of the corridor \(^6\). It is important to note that Policy Reuse is used as a way of exploring efficiently the space of policies to find a solution for the crossing problem. If the agent does not find useful to follow the policy \( \pi_0 \) in a state, it will learn a better policy because the exploratory policy is active along all the learning process.

\[
\begin{align*}
\psi & \quad \text{choose the } \pi_0 \text{ policy} \\
(1 - \psi)\epsilon & \quad \text{choose an aleatory action} \\
(1 - \psi)(1 - \epsilon) & \quad \text{choose the greedy policy}
\end{align*}
\]

6 Pedestrian simulation

In this section, we analyze the policies learned from the point of view of the pedestrian simulation. The value function learned by each agent in the last iteration of the processes is used in the simulation tests for two reasons: i) They represent the culmination of their respective learning iterative schemas ii) In the last iteration the number of agents is the same for both iterative schemas. The analysis of the simulations focuses on different aspects in the two scenarios. In the first, the focus is on the dynamics. A macroscopic and a microscopic study of different metrics to characterize the learned dynamics is presented. In the second, we focus on the emergent collective behavior, therefore a macroscopic study is performed to determine the shape of the emergent lines and its influence on the performance. In both, a comparison at the macro-dynamic level with the Helbing social forces model is performed.

In the following subsection, several metrics for analyzing the learned behaviors will be proposed. The reader can complement the data provided in these analysis with the visualization of several videos of these simulations in the URL http://www.uv.es/agentes/RL/index.htm

6.1 Simulation metrics and methodology

To determine the quality of a pedestrian simulation, the works \([45, 8]\) perform two assessments at two different levels. At the macroscopic level, they use the fundamental diagrams \([59]\) that establish the relationship between velocity and density. At the microscopic level, they study the trajectories of each pedestrian in order to analyze their movements.

To evaluate the quality obtained in our simulations, we have used three different kind of metrics, partly inspired by those assessments:

1. Local interactions (microscopic level): Velocity vs. collision distance correlation.
2. Macro-dynamics (macroscopic level): Fundamental diagram of pedestrians (speed vs. density relationship) and density maps.
3. Performance: Path length, number of decisions per episode and number of fails (episodes where the agent did not reach the goal).

\(^6\) Specifically, the policy \( \pi_0 \) choose randomly from the set of actions that turns the agent’s velocity vector towards the right side of the corridor
Firstly, the micro-dynamic metrics are used to analyze the local interactions of the pedestrians, by measuring the speed controller reactions under specific situations. The most interesting situation to analyze here is the collision response, so we have correlated the distance to the nearest neighbor with the agent speed, in order to study this important relationship.

Secondly, macro-dynamics are also very interesting for evaluating the behavior of the simulated group, and it is also an easy way to contrast the proposed schemas. In this case, we use density maps to have an idea of the group dispersion during the simulation, and the fundamental diagram that summarizes the local interactions the agents have learned in a single diagram. The fundamental diagram is an important tool in pedestrian modeling to compare and interpreting the dynamics of pedestrians [43]. These macro-dynamics indicators are compared with the Helbing model of social forces for pedestrians.

The specific formulation for the calculation of the weighted averaged density and velocity necessary for the fundamental diagram has been described and justified in the literature [21, 20]. Specifically, the local density is obtained by averaging over a circular region of radius $R$. The local density at the place $r = (x, y)$ at time $t$ is measured by

$$\rho(r, t) = \sum_j f(r_j(t) - r), \quad (3)$$

where $r_j(t)$ are the positions of the pedestrians $j$ in the surrounding of $r$ and

$$f(r_j(t) - r) = \frac{1}{\pi R^2} \exp\left[-\frac{|r_j - r|^2}{R^2}\right] \quad (4)$$

is a Gaussian, distance-dependent, weight function. The local speeds have been defined via the weighted average

$$V(r, t) = \frac{\sum_j v_j f(r_j(t) - r)}{\sum_j f(r_j(t) - r)}. \quad (5)$$

Some scalability tests have been performed in the simulation experiments for the first scenario. To scale in the number of agents, each simulated agent uses one copy of one of the learned value functions with its corresponding vector quantizer. In simulation, there are as many different behaviors as agents in the learning process, therefore several agents in the simulation use the same learned policy. Therefore, in this section all the experiments for the first scenario with more than 18 agents use copies of the original set of value functions and vector quantizers. Before reassigning a copy of a behavior to an agent for the $n$-th time, the rest of the behaviors have had to be reassigned $n - 1$ times.

Finally, we have also included different performance-oriented metrics, such as path lengths, the number of required decisions, and the number of fails, which are also interesting for comparing the schemas.

An important feature of MARL-based navigation groups is its robustness when increasing the number of simulation agents. To evaluate this, we have focused in the first scenario (agents inside a room). We have calculated all the quality metrics introduced for the two learning schemas presented (ITVQQL, INVQQL) and for different group sizes (i.e., scaling the group size by one (18 agents), by three (54 agents), by five (90 agents)) and therefore we can compare the results. We have
run 100 episodes for each experiment. An episode represents a single simulation where a group of agents must leave the room, and an experiment is represented by the schema used and the group size. The maximum duration of the episodes (measured by the number of decisions allowed) was 700, with a rate of two decisions per second during the simulations.

6.2 Local interactions analysis for the first scenario

In this section, we analyze the micro-behavior of the agents in the first scenario and focus specifically on the agents’ learned capacity to regulate their speed when interacting with the environment. These interactions represent the normal navigation conflicts and situations produced in collision avoidance scenarios for embodied autonomous agents. For this analysis, two parameters of the agent’s state space have been observed: the speed of the agent and the distance to the nearest neighbor. In a collision scenario, it seems reasonable that both parameters would have some correlation. That is, when the distance is shortened, the agent should reduce its velocity, and the reverse. Nevertheless, this is not always to be expected, since these parameters do not give a complete description of the local situation. For example, the nearest neighbor might be approaching from behind, therefore the agent could accelerate if no other agent is in front; or, in other situations, if the distance to the neighbor is large, the proximity of a wall could force a reduction of the agent’s speed. Therefore a positive correlation is expected between these two parameters although it should not be a total correlation. In Figures 9 and 10, these parameters have been plotted for individual agents in an episode and for different learning schemas. The graphics tell the story of an agent in an episode. The schemas have used the transfer of value functions in the learning phase. The graphs show representative cases of a simulation without scaling in the left column (18 agents) and scaling by five (90 agents) in the right column. Note that each graph represents an individual episode, therefore the number of steps is different in each one, depending on the random initial positions and the different local interactions that appear in it. For the same reason, the scales of the Y axis for the velocity and the distance are different in each graph. For the purpose of comparison, we have added two more experiments, as shown in Figure 10. In that figure, the first row represents the results for the basic VQQL schema while the second row presents the results of the RANDOM experiment, where the agents take random actions in each decision step, that is, they use no learned knowledge. The VQQL and RANDOM experiments have the same number of agents and the same configuration as the other schemas.

The curves of the graphs of Figures 9 and 10 have two different zones in terms of the collision situation. The zones of minimum distance, with a value around 0.6 meters, represents a situation where the neighboring agent is practically in contact with the observed agent. In this situation, a continuous acceleration (to avoid the neighboring agent) or deceleration (due to a new approach and possible crash) creates the speed oscillations in these zones of the curves. Note that the frequencies of the oscillations are not comparable in a simple observation of the graphs because of the different number of decisions in the presented episodes. In comparison, in the RANDOM experiment, the speed oscillations are present at any value of the distance. For example, in the graphic of the left column for
Fig. 9 Local interaction results for an agent in an episode. All the schemas have used transfer in the learning phase. The order of the displayed schemas are: ITVQQL in the first row, INVQQL in the second row. The first column is without scaling (18 agents), and the second column, with scaling (90 agents). The blue curve displays the distance to the nearest neighbor and gives an idea of the imminence of a collision. The red curve represents the velocity proposed by the learned controller. Along the abscissa, the number of decisions is a measure of the time.

the RANDOM experiment between the decisions number 0 and 20 the nearest neighbor is far from the agent but the speed oscillates from 0 to a maximum value of 1.0 with high frequency. This effect does not happen with an agent with a learned policy as it is observed at the graphic of the VQQL (above the RANDOM experiment). In the left graphic of the VQQL, a similar situation as thus described for the RANDOM experiment between the decisions number 1 and 7 is observed. However, in the VQQL graphic the speed increases monotonically and decreases when the distance becomes small. On the other hand, the curves show that, in general, the speed is increased by the agent in the zones of the curves where the distance to the nearest neighbor grows, reaching the maximum allowed speed (1.8 m/s) when the distance is large enough. See for example the zones at the beginning of the curves of the left column for all the learning schemas. The horizontal sections in the region of maximum speed match the regions of the distance curve where the distance is large: this is considered as a good indication of the stability of the learned behavior (for example, see this situation in the left column graph of the ITVQQL schema around the number of decision 7, and also in the left column graph of the INVQQL schema around the number of decision 26). Also
Fig. 10 Baseline experiments for the local interaction results for an agent in an episode. The order of the displayed schemas are: VQQL in the first row and RANDOM in the second row. The first column is without scaling (18 agents), and the second column, with scaling (90 agents). The blue curve displays the distance to the nearest neighbor and gives an idea of the imminence of a collision (when the minimum distance values 0.6). The red curve represents the velocity proposed by the learned controller. Along the abscissa, the number of decision is a measure of the time.

This property appears in the VQQL baseline curves (in the right column in the decisions 3, 4 and 37, 38) although the length of the sections are shorter than in the other schemas. In contrast, the RANDOM graphs do not have this property (see for example the erratic behavior of the speed at the beginning of the episode in the left column although the nearest neighbor is far away). In the graph of the right column (for the RANDOM experiment), the same situation appears. Note the lack of correlation between the speed and the distance around the decision number 50 and in number 200. Therefore, there is a high difference at the microscopic level between a random policy and a learned policy (independently of the learning schema).

A study of the correlation between the pairs of speed and distance is also given in Table 10. The study assumes that a correlation factor greater than +0.5 means a significant correspondence between the two parameters. The scenarios are very different, varying with the number of agents, because the local interactions

---

7 In order to fit the size of the table, we have abbreviated the names of the schemas in all the tables. Thus, IT means ITVQQL and the prefix TF means “with transfer of knowledge.”
are different in situations with high or low densities (see the comments to the fundamental diagrams of Figure 11 in the next subsection). Therefore the data in the two rows of Table 10 are not directly comparable, because it is to be expected that high density interactions are more frequent with 90 agents than with only 18. In reference to the first row (18 agents), the percentage of episodes with correlation greater than +0.5 between speed and distance is higher in the experiments with learning than in the RANDOM control, with higher percentages in the IT and TF\JT schemas respect IN and TF\IN schemas and VQQL. When 90 agents are used (second row), the percentages of episodes with this characteristic are lower in all the schemas but, as occurred for 18 agents, higher percentages where found in the IT and TF\JT schemas respect to the other ones. Because the neighbors are closer to the agent than in low density situations (the distance to the nearest neighbor is often the minimum value possible), speed oscillations often occur as explained before in the graphs of Figure 9 and Figure 10. This justifies a decrease in the correlation index between the two parameters in the data of the second row (90 agents). Note that the learned behaviors have correlation percentages significantly higher than the RANDOM control in both rows.

Table 10 Percentage of episodes in which the selected speed and distance to the nearest neighbor of an agent have a correlation coefficient greater than +0.5. The data comes from the simulation of 100 episodes with 18 agents each (a total of 1800 episodes) in the first row and with 90 agents (a total of 9000 episodes) in the second row.

<table>
<thead>
<tr>
<th># Ag.</th>
<th>IT</th>
<th>IN</th>
<th>TF\JT</th>
<th>TF\IN</th>
<th>VQQL</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>77.3%</td>
<td>37.1%</td>
<td>79%</td>
<td>46.5%</td>
<td>45.7%</td>
<td>1.72%</td>
</tr>
<tr>
<td>90</td>
<td>48.8%</td>
<td>12.4%</td>
<td>48.6%</td>
<td>12.1%</td>
<td>41.8%</td>
<td>1.1%</td>
</tr>
</tbody>
</table>

6.3 Macro-dynamics

This section gives the fundamental diagram (speed vs. density) and the density maps obtained in the experiments and their comparison with the Helbing model for the same configuration. The Helbing model used is described in [19], although the panic component has been disabled to resemble our experiments. The fundamental diagram summarizes the micro-behavior recently exposited, showing the relation between the speed and the density of all the agents involved. They are important for evaluating a pedestrian model. The left column of Figure 11 shows the fundamental diagram for all the schemas, specifically the average and its standard deviation of the speed values measured at different densities. The data was measured in a position in front of the door within a circular area of radius one meter.

The fundamental diagrams show in all the experiments that the velocity decreases when the density increases, which is a fundamental property of pedestrian dynamics [40]. The range of velocities in the INVQQL and ITVQQL schemas are higher than in the VQQL schema. Specifically, with high densities the velocity decreases to 0.2 in the INVQQL schema and less than 0.4 in the ITVQQL, whereas at high densities the velocity is 0.6 approximately for the VQQL. This may indicate a different dynamics of the VQQL respect to the other two schemas at high den-
Simulating Pedestrian Navigation with MARL

Fig. 11 Fundamental diagrams (mean velocity and std deviation vs. density) and density maps with 90 agents. First row=ITVQQL, 2nd=INVQQL, 3rd=VQQL, 4th=Helbing model. The data are from 100 episodes.
sities. Furthermore, in our schemas and VQQL the standard deviation decreases with the density, indicating that the agents have learnt to choose small variations of the velocity in high densities (that is, in a crowded situation). In a low density situation, an agent has more opportunity to modify the velocity vector, because collisions are less likely; therefore the standard deviation is high. This is also considered a positive characteristic in terms of the control of the velocity. The Helbing experiment shows the same property as, with increasing the density decreases the speed.

The right column of Figure 11 shows the density maps in the selected configurations. A density map is a histogram that counts the number of agents occupying a spatial zone along the time of the experiment, and therefore reflects the use of this zone. In these graphs, the $X$ and $Z$ axes represent the spatial positions in the room. The square room has been divided by a $25 \times 25$ grid. The $Y$ axis represents the occupation of the space in terms of the number of times that a unit square of the grid has been occupied in the 100 episodes of the experiment. The shell-shaped pattern in front of the door is a typical collective effect of clogging in the kind of bottlenecks studied in pedestrian dynamics [40, 17]. Here, the bottleneck is clearly represented in the surface shape close to the door for all the experiments, including the Helbing’s experiment. Note the absence of significant heaps in other zones of the space, highlighting the absence of any wandering around the room. This is a common characteristic with the Helbing’s model.

With respect to the scalability, note that the mentioned characteristics of the results explained above are tested in an environment with 90 agents whereas the number of agents in the learning process was 18. This means that the approach is robust to scaling in the number of agents (note that the scaling is performed without additional learning, only sharing the learned value functions by the agents) and is capable of generalization in terms of the persistence of the learned dynamics. This statement is also corroborated by the results of Table 13 as discussed below.

6.4 Performance

The next tables characterize the different schemas with respect to the scalability in the number of agents. The learning tasks were performed with 18 agents.

Table 11 shows the length, in meters, of the agents’ paths. The lengths are practically constant with the scaling (read the data by columns), suggesting that similar behaviors occur independently of the scaling factor. Specifically, the agents tend to avoid detours in their trajectories. This behavior is also observed in real pedestrians and has been reported in [23]. There is little difference in the use or not of transfer for the ITVQQL and the INVQQL schemas. In general, the lengths for the schemas with transfer are greater than those without it.

The Table 12 displays the averaged number of decisions per episode taken by the agents. The number of decisions increases when scaling up the number of agents because they take longer to exit. The number of decisions also increases between schemas. The ITVQQL schema is the one with fewer decisions per episode on average both with and without transfer. The schemas with transfer have higher mean number of decisions. The high deviation of the mean in all the measures of the table is a result of two effects. First the number of decisions per episode is a sparse magnitude. In every episode, the first agents reaching to the exit use few decisions.
Table 11: Averaged lengths and standard deviation for the paths in meters. The averages are over 100 episodes and for all the agents.

<table>
<thead>
<tr>
<th>#Ag.</th>
<th>IT</th>
<th>IN</th>
<th>TF_IT</th>
<th>TF_IN</th>
<th>VQQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>14 ± 7</td>
<td>20 ± 21</td>
<td>15 ± 7</td>
<td>18 ± 10</td>
<td>15 ± 11</td>
</tr>
<tr>
<td>36</td>
<td>13 ± 5</td>
<td>18 ± 15</td>
<td>15 ± 7</td>
<td>18 ± 8</td>
<td>15 ± 17</td>
</tr>
<tr>
<td>54</td>
<td>14 ± 6</td>
<td>18 ± 13</td>
<td>16 ± 8</td>
<td>19 ± 10</td>
<td>15 ± 11</td>
</tr>
<tr>
<td>72</td>
<td>15 ± 6</td>
<td>18 ± 12</td>
<td>16 ± 8</td>
<td>19 ± 11</td>
<td>15 ± 10</td>
</tr>
<tr>
<td>90</td>
<td>15 ± 7</td>
<td>18 ± 11</td>
<td>17 ± 9</td>
<td>20 ± 11</td>
<td>17 ± 10</td>
</tr>
</tbody>
</table>

Table 12: Average number and standard deviation of decisions per episode. The figures are averaged over 100 episodes and for all the agents.

<table>
<thead>
<tr>
<th>#Ag.</th>
<th>IT</th>
<th>IN</th>
<th>TF_IT</th>
<th>TF_IN</th>
<th>VQQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>28 ± 56</td>
<td>41 ± 56</td>
<td>28 ± 56</td>
<td>63 ± 35</td>
<td>27 ± 21</td>
</tr>
<tr>
<td>36</td>
<td>32 ± 41</td>
<td>75 ± 52</td>
<td>67 ± 53</td>
<td>93 ± 58</td>
<td>43 ± 60</td>
</tr>
<tr>
<td>54</td>
<td>51 ± 55</td>
<td>118 ± 77</td>
<td>95 ± 74</td>
<td>135 ± 86</td>
<td>52 ± 60</td>
</tr>
<tr>
<td>72</td>
<td>68 ± 70</td>
<td>121 ± 78</td>
<td>120 ± 84</td>
<td>175 ± 112</td>
<td>69 ± 62</td>
</tr>
<tr>
<td>90</td>
<td>84 ± 85</td>
<td>144 ± 95</td>
<td>145 ± 96</td>
<td>211 ± 131</td>
<td>105 ± 135</td>
</tr>
</tbody>
</table>

On the contrary, when the episode evolves and the bottleneck appears, the agents involved in the clogging have to use a higher number of decisions. Second, the existence of permanent cloggings in several episodes, produces situations where the agents cannot reach the door. When this occurs, each agent reaches the maximum possible number of decisions per episode, trying to get out of the clogging.

Table 13 displays the averaged number of agents who fail to exit by the door and its correspondent median. A statistical analysis was performed using the Statgraphics software package. A Kruskal-Wallis test using the medians was conducted to determine statistical significant differences among the experiments because neither the raw data nor the transformed data adjusted to a normal distribution. As main results we can observe:

1. The means of all the schemas remain similar when a scaling with 54, 72 and 90 agents is performed. This fact suggests the possibility that the scalability results could be similar with a higher number of agents.
2. In the experiments with 18 agents (without scaling), the schemas with transfer are statistically different from the schemas without transfer and also from the VQQL. This shows that transfer has a significant impact when it is used without scaling. When the number of agents is scaled up, the IT and the TF_IT are statistically classified as a different group respect to the IN and TF_IN schemas. The VQQL is placed in a different group from the TF_IT schema in three of the five experiments.
3. The means in the VQQL are significantly higher than the others when scaling up the number of agents. Thus, it behaves the worst in all the experiments.

Reading by rows, the TF_IT schema has the best performance (lowest mean) in all the configurations. Note the positive effect of the transfer of knowledge in the TF_IT schema when the number of agents is scaled up, as opposed to the lack of such an effect in the TF_IN schema when scaling. However, if we compare the rows for 18 agents for both schemas with and without transfer, we can see that both improve the performance using transfer. This agrees with the performance results for 18 agents in the learning processes (see Figure 5 and Figure 6).

From previous evaluations, we can extract several conclusions:
Table 13 Medians and means (in parenthesis) for the agents that do not reach the door (fails) when scaling up the number of agents. The means are averaged over 100 episodes (N=100) and are considered a measure of performance. Median values separated by different letters for the same number of agents (within a row), are significantly different ($P \leq 0.05$) according to Kruskal-Wallis test.

<table>
<thead>
<tr>
<th>#Ag.</th>
<th>IT</th>
<th>IN</th>
<th>TFJT</th>
<th>TFJN</th>
<th>VQQL</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>1 b (2.4)</td>
<td>4 c (4.2)</td>
<td>0 a (1.0)</td>
<td>0 a (2.9)</td>
<td>1 b (2.8)</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>1 ab (6.5)</td>
<td>4 c (4.8)</td>
<td>0 a (2.8)</td>
<td>3.5 bc (6.2)</td>
<td>0 ab (6.8)</td>
<td>4 x 10^{-9}</td>
</tr>
<tr>
<td>54</td>
<td>1 a (6.4)</td>
<td>4 b (6.0)</td>
<td>0 a (3.6)</td>
<td>4 b (6.4)</td>
<td>10 ab (15.7)</td>
<td>7 x 10^{-10}</td>
</tr>
<tr>
<td>72</td>
<td>1 ab (9.4)</td>
<td>4 b (5.6)</td>
<td>1 a (3.9)</td>
<td>4 b (6.6)</td>
<td>4 b (22.1)</td>
<td>4 x 10^{-8}</td>
</tr>
<tr>
<td>90</td>
<td>1 ab (10.3)</td>
<td>4 b (6.0)</td>
<td>1 a (4.1)</td>
<td>3 b (6.5)</td>
<td>18.5 c (25.0)</td>
<td>4 x 10^{-10}</td>
</tr>
</tbody>
</table>

1. The analysis of the graphs of speed and distance to the nearest neighbors, which are related to the microscopic (local) behavior of the agents, shows that all the schemas provide behaviors with a correlation between the curves. The observation of individual episodes also reveals that the controller tends to remain unchanged when the distance to the nearest neighbor is large, which is considered to be a rational behavior.

2. The fundamental diagrams and density maps reveal that the main characteristics of the pedestrian dynamics and collective behavior appear in all the schemas. These tests, which are mainly related with the macroscopic (collective) behavior, do not exhibit any significant differences between the two schemas.

3. The scalability tests displayed in the tables show properties of real pedestrian behavior and demonstrate empirically that the learned behaviors are generalizable to other configurations with more agents.

4. The results showed in the Table 13 indicate that the VQQL baseline algorithm has worse performance (highest mean) than the rest of the schemas when scaling up the number of agents. Also it shows that the TFJT has the best performance (lowest mean) in all the experiments with different number of agents.

6.5 Macroscopic analysis for the second scenario

In the crossing scenario we are interested in studying the emergence of collective behaviors. Specifically the formation of lines as the emergent behavior that solves the crossing. Our scenario has two meters wide, therefore, only three agents are necessary to obstruct the corridor. Without an organization of the groups, the clogging is a difficult problem to solve.

The solution found by the agents in the learning schemas (TFJT, TFJINV) and in VQQL, has the same structure. The agents that belong to a group form one line in anticipation of the moment of crossing as can be seen in the sequence of images in Figure 12. The reader can see some videos of the results in the URL http://www.uv.es/agentes/RL/index.htm.

In this scenario, the performance is measured considering whether all members of the group have reached to the goal. Thus, only the episodes in which the crossing has been solved successfully are taken into account. This performance measure is more restrictive than that used in the first scenario, where is measured how many agents reached to the goal. The results of the Table 14 indicates that not all the
episodes solve the crossing. Although the lanes appear in all the episodes, it is necessary that the lane formation anticipates the instant of the crossing to avoid a clogging that the agents cannot solve with the limited time given to each episode (80 decisions that is 40 seconds in simulation time). The TF\_IT schema has the highest performance with significant statistical difference respect to the TF\_IN schema and the VQQL baseline.

In Figure 13, the density maps are displayed. As in the first scenario, the maps are built accumulating the positions of the agents in 100 episodes. The flat zones at the borders are the space occupied by the walls. Note the two high density zones near the walls that corresponds to the lane formations. The high values of the histogram in those zones reveals that the lane formation occurs often (actually in all the episodes) and with similar structure in all the schemas (including the VQQL algorithm). This fact together with the results showed in the Table 14 reveal that the line formation is not sufficient by itself to solve the clogging and it is also necessary that the lanes appear before the crossing. In order to emphasize the differences among the schemas, a side view of the map is included in the right column of Figure 13. In the side view, the shape of the middle region of the corridor in terms of densities can be seen. It can be observed that the VQQL experiment has the highest density in the middle zone of the corridor, meaning the existence of cloggings in this zone. The TF\_IT schema has the lowest density in the middle of the corridor, suggesting that the anticipation in the lane formation has been learned better.

From the exposed results we extract the following conclusions for the second scenario:

1. The emergence of the lanes occurs in all the learning approaches evaluated. In all the cases, the lanes have similar structure.
Fig. 13 Density maps of the schemas for the crossing scenario. The data come from 100 simulations. The right column shows a side view of the same map to compare the density of the middle region of the corridor. Note that the graphics have not the same scale.

2. The TF IT has better performance than the TF IN schema and the VQQL baseline experiment. The ANOVA test shows significant differences in the results.

7 Conclusions

In this paper, a new MARL methodology approach has been introduced for the simulation of pedestrian groups. Unlike other solutions, where the behaviors of the pedestrians are coded in the system using an existing model (see the papers
reviewed in Section 1), in our approach the agents learn their behaviors by interacting with the environment using reinforcement learning. The use of learning techniques provides several advantages for the pedestrian simulation field:

1. The agents learn independently. This implies that the learned policies are different for each agent, creating a variety of behaviors, as occurs with real pedestrians.
2. The agent’s learning is offline. Once the learning phase has been completed, the simulation of the learned policies simply relies on looking up in a table, so it has a very low computational overhead. This feature is a necessity for proper scaling in the number of agents as required.
3. In domains where the interactions are complex and not totally known, model-free techniques such as Q-learning avoid introducing hand-coded domain knowledge into the system.
4. The agent is not a closed system where its knowledge comes only from its own experience. The RL framework provides techniques for incorporating, into the agents, external knowledge (i.e., using knowledge transfer techniques).

Two different algorithmic schemas have been designed to increase the performance of the VQQL algorithm. Each one proposes a different strategy to address the problem of getting an adequate description of the state space for the task as well as how the problem of the multi-agent learning is dealt. In addition, different techniques of knowledge transfer have been used to accelerate the learning process. In both scenarios, the ITVQQL with transfer (TF.IT) has better performance than the others for the studied configurations. Specifically it gives the best performance in the first scenario when scaling up the number of agents as well as in the second scenario in the number of solved crossings. Although the studied scenarios in this paper are different enough to indicate a broad usability of the ITVQQL with transfer algorithm, it is specially useful in domains in which the random exploration of the state space does not lead to quantizers with a good enough representation of the state space to find the optimal policy.

Several goals have been addressed successfully in this paper. First, the learning experiments demonstrate that the VQQL algorithm and its derivate schemas are convergent in the different pedestrian scenarios, finding policies that carry out the proposed tasks. This suggest that the introduced MARL framework is applicable and generalizable to the simulation of pedestrians groups. Second, the simulation of these policies has confirmed the learning of the basic rules of pedestrian dynamics at different levels. From a macro-dynamics point of view, the speed-density relation in the fundamental diagram and the emergence of a basic collective behaviors (lane formation in the corridor and shell-shaped organization of the pedestrians in front of the door) result as a consequence of the policies learned. Specifically for the first scenario and from a micro-dynamics perspective, the stability of the speed controller shows the rational behavior learned (see also the videos in the URL referenced). Third, the learned behaviors in the first scenario seem robust to scalability in the number of agents. Besides, the comparison at the macroscopic level with the Helbing pedestrian model of social forces shows similarities with our results in terms of the fundamental diagrams and the density maps in the first scenario (closed room with an exit). These similarities with the Helbing model supports the idea that our agents have developed plausible behaviors of pedestrians.
The research presented here opens up several perspectives. With regard to the environment, more complex designs of the embodied agents and/or more realistic physics-based interactions (i.e., with the presence of frictional forces), could provide more realistic simulations. Also, we could consider other environments with different facilities, where different collective behaviors could appear. With regard to the agents, other generalization methods of the state space such as tile coding will be analyzed in future work.

Acknowledgments

This work has been jointly supported by the Spanish MICINN and European Commission FEDER funds under grants Consolider-Ingenio CSD2006-00046 and TIN2009-14475-C04-04. Fernando Fernández is supported by Spanish MINECO under grant TIN2012-38079-C03-02 and TRA2009-0080.

References


