Robotics and Autonomous Systems 191 (2025) 105020

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

RUMOR: Reinforcement learning for understanding a model of the real world for navigation in dynamic environments

Diego Martinez-Baselga *, Luis Riazuelo, Luis Montano

Robotics, Perception and Real Time Group, Aragon Institute of Engineering Research (I3A), University of Zaragoza, Zaragoza, 50018, Spain

ARTICLE INFO

Keywords:
Mobile robots
Motion planning
Collision avoidance
Reinforcement learning
Differential-drive robots
Dynamic environments

ABSTRACT

Autonomous navigation in dynamic environments is a complex but essential task for autonomous robots, with recent deep reinforcement learning approaches showing promising results. However, the complexity of the real world makes it infeasible to train agents in every possible scenario configuration. Moreover, existing methods typically overlook factors such as robot kinodynamic constraints, or assume perfect knowledge of the environment. In this work, we present RUMOR, a novel planner for differential-drive robots that uses deep reinforcement learning to navigate in highly dynamic environments. Unlike other end-to-end DRL planners, it uses a descriptive robocentric velocity space model to extract the dynamic environment information, enhancing training effectiveness and scenario interpretation. Additionally, we propose an action space that inherently considers robot kinodynamics and train it in a simulator that reproduces the real world problematic aspects, reducing the gap between the reality and simulation. We extensively compare RUMOR with other state-of-the-art approaches, demonstrating a better performance, and provide a detailed analysis of the results. Finally, we validate RUMOR's performance in real-world settings by deploying it on a ground robot. Our experiments, conducted in crowded scenarios and unseen environments, confirm the algorithm's robustness and transferability.

1. Introduction

Motion planning and navigation in dynamic scenarios is essential for autonomous robots, for applications as delivery or assistance. Nevertheless, traditional planners fail in environments where the map is mutable or obstacles are dynamic, leading to suboptimal trajectories or collisions. Those planners typically consider only the current obstacles' position measured by the sensors, without considering the future trajectories they may have.

New approaches that try to solve this issue include promising end-to-end Deep Reinforcement Learning (DRL) based methods. The robot learns a policy that selects the best action for each of the situations, directly represented with the sensed information. They present results that outperform model-based approaches in terms of success (reaching the goal without collisions) and time to reach the goal. However, the complexity of the real world and the huge variety of different possible situations the robot may encounter, with different number of obstacles, different shapes and different behaviors, poses a huge training challenge for these end-to-end approaches. Moreover, they typically lack in sim2real transfer capabilities, not considering robot kinodynamic constraints, or partial observability issues.

In this paper, we propose **RUMOR** (**R**einforcement learning for **U**nderstanding a **MO**del of the **R**eal world), a motion planner that combines model-based and DRL benefits. We use the deep abstraction of the environment provided by the Dynamic Object Velocity Space (DOVS) [1] to understand the surrounding scenario. The DOVS model represents the dynamism and the future of the environment in the velocity space of the robot. The planner applies this information as an input of the DRL algorithm, learning to interpret the future dynamic scenario knowledge, taking advantage over other approaches that use raw obstacle information and are not able to generalize in scenarios that are different from those previously experienced. The perception is decoupled from the learning algorithm and used to construct the DOVS, making it able to work with any sensor as a LiDAR or a camera. Fig. 1 provides a representation of RUMOR pipeline. In addition, unlike other methods that simply choose a velocity ranging from the maximum and minimum robot velocities, we propose an action space that inherently considers differential-drive kinodynamic restrictions, improving the applicability of RUMOR in real robots. The algorithm is trained and tested in a realistic simulator, where all information is extracted from the sensor measurements, even the own robot localization. Training

* Corresponding author.
E-mail addresses: diegomartinez@unizar.es (D. Martinez-Baselga), riazuelo@unizar.es (L. Riazuelo), montano@unizar.es (L. Montano).

https://doi.org/10.1016/j.robot.2025.105020
Received 30 March 2024; Received in revised form 1 March 2025; Accepted 8 April 2025
Available online 23 April 2025

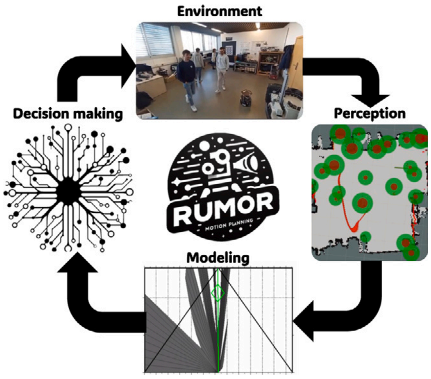


Fig. 1. Pipeline of the approach presented. It takes the information sensed from the environment to construct a model of the dynamism of the scenario. Then DRL is used to compute differential-drive velocity commands.

using the robot constraints and a simulator that includes the problems of the real world reduces the sim2real transfer complexity.

In Section 2, we analyze the background and relate our work with the state of the art and in Section 3 we state the preliminaries needed for understanding the method. Sections 4.2 and 4.3 present the approach taken. We provide comparisons of RUMOR with other method of the state-of-the-art and experiments with a real-ground robot in Section 5. Finally, the conclusions of the work are stated in Section 6.

Our contributions are summarized in:

- A motion planner that combines a very complete and descriptive information of the environment on top of DRL to merge the benefits of model-based and DRL methods.
- A way to inherently consider differential-drive robot kinodynamics in a DRL planner and deal with some real-world problems as the variety of scenario configurations or partial observability.
- Experiments that show the benefits of our contribution, comparisons with other methods and real-world demonstrations.

The code and videos of the experiments are available at https://github.com/dmartinezbaselga/RUMOR.

2. Related work

In this section, we give a brief overview of the existing solutions for autonomous navigation in dynamic environments, with special emphasis in deep reinforcement learning planners. We refer the readers to recent surveys [2–4] for a more detailed introduction to the field.

2.1. Motion planning in dynamic environments

The motion planning problem is frequently solved with the conjunction of a global planner, which computes a static path considering a continuous and static map, and a local planner that follows it accounting for the dynamics of the robot and obstacles that were not considered by the global planner. Traditional local planners like the Dynamic Window Approach (DWA) [5] do not consider that obstacles may move in time, thus leading to collisions and suboptimal trajectories, as seen in some works as the benchmark proposed in [6]. First approaches for dynamic environments use attractive forces to lead the robot to its goal and repulsive forces to drive it away from obstacles, such as artificial potential fields [7] or Social Force [8]. They were later improved with Time Elastic Bands (TEB) [9] or multiple extensions to Social Force [10,11].

A big group of works are velocity-space based. The *Velocity Obstacle (VO)*, introduced in [12], refers to the set of possible robot velocities that could lead it to a collision with an obstacle that has a certain velocity in the near future. Based on the VO concept, the *Dynamic Object Velocity Space (DOVS)* for differential-drive robots is defined in [1] as the robot velocity space that includes the safe and unsafe robot possible velocities for a time horizon derived from VO. In that work, a planner based on strategies is also defined, the S-DOVS. Another work [13] uses the DOVS to discard velocities that lead to collisions from the action space of a tables-based reinforcement learning algorithm. However, unlike our work, it does not use the information of the DOVS to interpret the environment and just filters the unsafe actions available. In addition, the results presented in their work are very similar to S-DOVS, which does not require training time. Other extensions of VO use reciprocal collision assumptions, such as Reciprocal Velocity Obstacles (RVO) [14] or Optimal Reciprocal Collision Avoidance (ORCA) [15], which leads to failures in scenarios where there are non-collaborative obstacles.

Application of optimization-based approaches in dynamic environments, such as those that are based in a Model Predictive Controller (MPC), is gaining ongoing attention in recent years. The authors of [16] present a Model Predictive Contouring Controller that considers the future obstacle trajectories to safely reach the goal. It was recently combined with a global planning approach [17] to dynamically change between possible global plans. Other approaches combine crowd motion prediction with MPC navigation [18] or introduce topological invariance to improve social navigation [19]. As our approach, these works account for kinematic and dynamic constraints of the robot, but they have the problem of being computationally costly, and their planning is reliable as long as the predictions are accurate, so they are not suitable for very dense or chaotic scenarios.

2.2. Deep reinforcement learning planners

Deep Reinforcement Learning (DRL) [20] is a method used to learn to estimate the optimal policy that optimizes the cumulative reward obtained in an episode with a deep neural network. DRL algorithms have proven state-of-the-art performance in several benchmarks, including Rainbow [21] for discrete action spaces, distributed reinforcement learning algorithms [22] or actor-critic methods [23].

Some works study the benefits of DRL in robot motion planning and the limitations of traditional planners in dynamic environments. Defining strategies for every situation that may be found in the real world or optimizing trajectories in very dynamic environments is intractable, so DRL emerges to efficiently solve the decision-making problem, which is complex and has many degrees of freedom. There are works [6,24,25] that implement and compare DRL algorithms with model-based ones, proving their performance.

There are two big groups of DRL planners: end-to-end and agent-based planners. The former use the raw sensor measurements directly as the DRL network input. These measurements may be sensor maps [26–29], LiDAR laser scans [30–33] or images [34–36] taken from robot's onboard sensors. In spite of the efforts in designing realistic simulators [25,37,38], these methods still experience distribution-shift problems when the deployment environments differ from the training one [39,40], and require the robot to have the same sensor configuration (for example, sensor position, range or resolution) both in training and in deployment, limiting the applicability of their policy. Agent-based planners extract an agent-level representation from sensor measurements and use it as the planner input. In this way, the robot does not require to have specific sensor settings and does not experience distribution shifts due to the scenario geometry and shape. Furthermore, it may learn to understand deeper agent-level behaviors by differentiating decision-making agents from other obstacles [41].

The first agent-based DRL approach proposed for motion planning [42] uses a fully connected network to select velocities regarding the position and velocity of the closest surrounding obstacles. LSTM [43] layers were included in [41,44] to account for multiple obstacles. SARL [45] uses attention to model interactions among obstacles, achieving state-of-the-art performance. Recent approaches compare their results to it, and they usually focus on improving the structure of the network. For example, [46] uses a graph neural network or [47] an attention-based interaction graph. In a recent study, [48] we showed that the performance of DRL approaches may be improved using intrinsic rewards. The use of DRL algorithms in robot navigation has many important challenges, including the need of learning from a limited subset of possible scenarios, environmental and robot constraints or partial observability [49,50]. The previous approaches do not face these problems. In our approach, instead of directly process the observations, we use the DOVS model, that represents the dynamism of the environment in the velocity space, as an input to the DRL algorithm, doing the decision-making process agnostic of the specific scenario.

Other works consider social settings, such as [51] including social stress indices, [52] a social attention mechanism or [53] social rewards to shape the robot behavior. Another approach [54] uses a social risk map as part of the input of the network. The applicability of these social approaches is limited when other obstacles are not humans, like other robots, or humans do not behave as expected.

An example of an approach that considers the kinodynamic restrictions is [55], which combines DRL with DWA [5], but only achieving a success rate of 0.54 in low dynamic scenarios. Our proposed approach solves the problem of selecting feasible velocities by considering kinematic and dynamic equations in the action space formulation, limiting the downgrade in performance.

3. Problem formulation

We consider a differential-drive robot that shares the workspace $\mathcal{W} \subseteq \mathbb{R}^2$ with static and dynamic obstacles. The robot position at time t is $\mathbf{p_t} = (x_t, y_t, \theta_t) \in \mathcal{W} \times [-\pi, \pi)$ and its velocity $\mathbf{u}_t = (\omega_t, v_t) \in \mathcal{V} \subseteq \mathbb{R}^2$ (angular and linear velocity). The robot maximum linear and angular velocities are bounded: $\mathcal{V} = [-\omega_{max}, \omega_{max}] \times [0, v_{max}]$. It has to reach a goal $\mathbf{g} = (x_g, y_g) \in \mathcal{W}$ in the minimum possible time, while avoiding static and dynamic obstacles in the environment. The robot is represented with a disk with radius r_{robot}, and the dynamic obstacles are assumed to be circular disks with radius $\mathbf{r}_{obs} \in \mathbb{R}^M$, where M is the number of dynamic obstacles. For each obstacle i, we assume that the robot may estimate its current position and velocity using the sensor information. Let $\mathcal{R}_t(\mathbf{u}) \in \mathcal{W} \times [-\pi, \pi)$ be the robot state at time t if it drives with velocity $\mathbf{u}$, $\mathcal{O}_{i,t} \in \mathcal{W}$ the set of points occupied by obstacle i at time t, $\mathcal{O}_{i,t}^{T_h} = [\mathcal{O}_{i,t}, \dots, \mathcal{O}_{i,t+T_h}]$ for the time horizon T_h and $\mathcal{O}_{\forall,t}^{T_h} = [\mathcal{O}_{1,t}^{T_h}, \dots, \mathcal{O}_{M,t}^{T_h}]$ for the whole set of static and dynamic obstacles in the environment. We consider that the robot follows the unicycle model:

$$\begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{\theta}_t \end{pmatrix} = \begin{pmatrix} \cos\theta_t & 0 \\ \sin\theta_t & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \tag{1}$$

The robot is also restricted by the low-level forward model that relates the wheel velocities to the actions $\mathbf{u}_t$ for a differential-drive robot:

$$\begin{aligned} v_t &= \frac{r_w(\omega_{r,t} + \omega_{l,t})}{2} \\ \omega_t &= \frac{r_w(\omega_{r,t} - \omega_{l,t})}{L_w} \end{aligned} \tag{2}$$

where r_w is the wheel radius, $\omega_{r,t}$ and $\omega_{l,t}$ the angular velocity of the right and left wheels, respectively, and L_w the distance between wheels. The relationship $\frac{v_t}{\omega_t} = \frac{2}{L_w}$ imposes two linear constraints regarding the maximum linear velocity a robot may take in a particular moment with respect to its angular velocity:

$$\begin{aligned} l_1 &: v_t \leq \frac{v_{max}}{\omega_{max}}\omega_t + v_{max}, \quad \omega_t \leq 0 \\ l_2 &: v_t \leq -\frac{v_{max}}{\omega_{max}}\omega_t + v_{max}, \quad \omega_t > 0 \end{aligned} \tag{3}$$

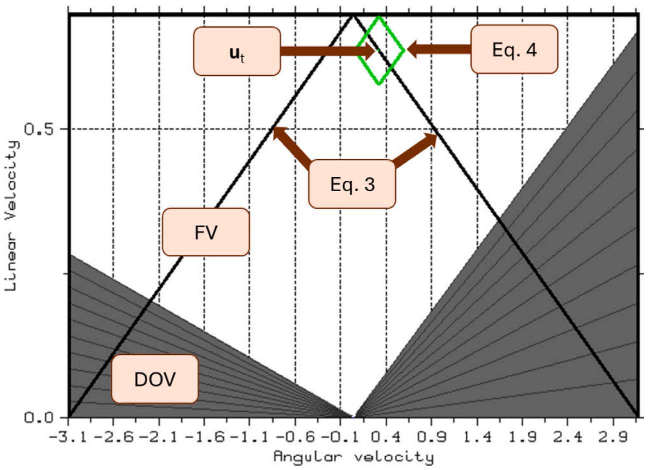


Fig. 2. Graphical representation of the DOVS model and the differential-drive robot restrictions. Constraints of Eq. (3) are represented with two black lines and restrict maximum linear velocities regarding the angular velocity. Eq. (4) are plotted as a green rhombus around $\boldsymbol{u}_t$, representing acceleration limits regarding a differential-drive robot. In this example, the robot velocity limits are $v_{max} = 0.7$ m/s, $\omega_{max} = \pi$ rad/s and $a_{max} = 0.3$ m/s^2. The dark (DOV) and white (FV) areas include unsafe and safe velocities derived using VO for a time horizon. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We consider that the robot has a maximum linear acceleration of a_{max}. The change in the linear velocity of the robot is limited by its previous linear velocity and, due to Eq. (2), its velocity limits:

$$\begin{aligned} v_{t+1} &\leq \begin{cases} \dfrac{v_{max}}{\omega_{max}}\omega_t + v_t + a_{max}\Delta t, & \omega_t \leq 0 \\ -\dfrac{v_{max}}{\omega_{max}}\omega_t + v_t + a_{max}\Delta t, & \omega_t > 0 \end{cases} \\ v_{t+1} &\geq \begin{cases} -\dfrac{v_{max}}{\omega_{max}}\omega_t + v_t - a_{max}\Delta t, & \omega_t \leq 0 \\ \dfrac{v_{max}}{\omega_{max}}\omega_t + v_t - a_{max}\Delta t, & \omega_t > 0 \end{cases} \end{aligned} \tag{4}$$

The restrictions of Eq. (3) and Eq. (4) are plotted in a graphical representation of the robot velocity space in Fig. 2. The velocity in the next control period $\boldsymbol{u}_{t+1}$ can never be above the two big black lines that relate linear and angular velocity constraints, and can never be outside the green dynamic window that surround $\boldsymbol{u}_t$, which constraints linear and angular acceleration and their relationship.

4. Methodology

In this section, we explain the concepts of the DOVS and its adaptation used in RUMOR, as well as the deep reinforcement learning solution proposed to solve the problem.

4.1. Dynamic Object Velocity Space (DOVS)

In this work, we use the DOVS [1] to model the information of the environment. It translates scenario information from the workspace $\mathcal{W}$ to the robot velocity space $\mathcal{V}$, by computing the velocities in $\mathcal{V}$ lead to a collision within a temporal horizon. This provides a robocentric, bounded, scalable and flexible model with the original information of the workspace.

The DOVS is modeled using possible robot velocities. As we consider a differential-drive robot, its velocities result in circular trajectories if they are kept constant. We denote $\tau_j \equiv \tau_j(\omega_j, v_j)$ the robot trajectory τ_j produced by $\boldsymbol{u}_j = (\omega_j, v_j) \in \mathcal{V}$ and $\mathcal{T}$ the collection of different circular trajectories that collectively cover $\mathcal{V}$. The set of colliding velocities with the obstacle i are the trajectories that intersect with i within a time horizon. They define Dynamic Object Velocity of i:

Definition 1. The Dynamic Object Velocity (DOV) of obstacle i, for the set of trajectories $\mathcal{T}$ and the time horizon T_h is defined as:

$$\begin{aligned} DOV(\mathcal{O}_{i,t}^{T_h},\mathcal{T}) = \big\{ & \mathbf{u}_j = (\omega_j, v_j) \in \mathcal{V} \mid \exists \tau_j \in \mathcal{T}, \\ & \exists t' \in [t, t+T_h],\ \mathcal{R}_{t'}(\mathbf{u}_j) \cap \mathcal{O}_{i,t'} \neq \emptyset \big\} \end{aligned} \tag{5}$$

The DOV of all obstacles in the environment is combined for the general definition:

Definition 2. The Dynamic Object Velocity (DOV) of M obstacles of the set of trajectories $\mathcal{T}$ and the time horizon T_h is defined as:

$$DOV(\mathcal{O}_{\forall,t}^{T_h},\mathcal{T}) = \bigcup_{i=1}^{M} DOV(\mathcal{O}_{i,t}^{T_h},\mathcal{T}) \tag{6}$$

The Free Velocity (FV) is constructed as the complementary set of the DOV with the collision-free velocities:

Definition 3. The Free Velocity (FV), for the set of trajectories $\mathcal{T}$ and the time horizon T_h is defined as:

$$FV(\mathcal{O}_{\forall,t}^{T_h},\mathcal{T}) = \big\{ \mathbf{u}_j = (\omega_j, v_j) \in \mathcal{V} \mid \mathbf{u}_j \notin DOV(\mathcal{O}_{\forall,t},\mathcal{T}) \big\} \tag{7}$$

Finally, the DOVS is modeled as the union of the whole set of velocities that comprise the DOV and the FV, categorizing every velocity in $\mathcal{V}$:

Definition 4. The Dynamic Object Velocity Space (DOVS), for the set of trajectories $\mathcal{T}$ and the time horizon T_h is defined as:

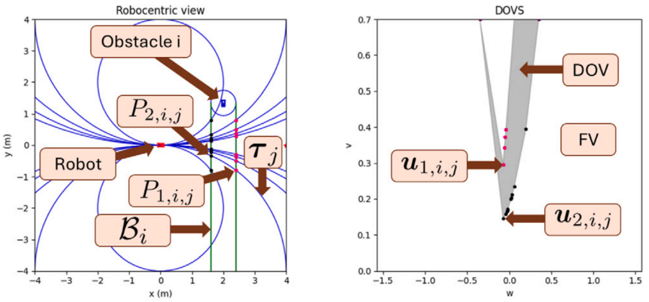
$$DOVS(\mathcal{O}_{\forall,t}^{T_h},\mathcal{T}) = \left\{ DOV(\mathcal{O}_{\forall,t}^{T_h},\mathcal{T}) \cup FV(\mathcal{O}_{\forall,t}^{T_h},\mathcal{T}) \right\} \tag{8}$$

The DOVS is computed with the concept of the collision band, $\mathcal{B}_i$. The robot is reduced to a point and the obstacle is enlarged with the robot radius, to simplify computations while keeping the same collision area. The collision band, $\mathcal{B}_i$, is the area that obstacle i will occupy by following its predicted trajectory. In this work, we assume the robot only knows obstacle i current velocity, so its trajectory is predicted by considering that its velocity is kept constant. Thus, obstacle trajectories are straight lines or circular trajectories, depending on whether the obstacle has angular velocity or not. The DOVS is computed with the intersections of trajectories and $\mathcal{B}_i$. For each robot circular trajectory $\tau_j \in \mathcal{T}$, the intersection points between the two lines limiting $\mathcal{B}_i$ and τ_j are $P_{1,i,j}(x_{1,i,j}, y_{1,i,j})$ and $P_{2,i,j}(x_{2,i,j}, y_{2,i,j})$. They are graphically represented in Fig. 3(a) in a robocentric view for a few example trajectories.

Different combinations of linear and angular velocities lead to the same trajectory, as long as the trajectory radius, $\frac{v}{\omega}$, is the same. Depending on the velocity, the robot will eventually be at the same time as obstacle i in $\mathcal{B}_i$, colliding, or will not. The time $t_{2,i,j}$, which is the time any point of the obstacle i takes to reach $P_{2,i,j}$, and $t_{1,i,j}$, which is the time it takes to stop being in contact to $P_{1,i,j}$, may be efficiently computed considering the velocity of i [1]. We denote $\boldsymbol{u}_{k,i,j}$ the velocity that makes the robot follow trajectory $\boldsymbol{\tau}_j$ and reach $P_{k,i,j}$ at time $t_{k,i,j}$, for $k = 1,2$. Colliding velocities are velocities $\boldsymbol{u}_j$ that follow $\boldsymbol{\tau}_j$ and $v_{2,i,j} \leq v_j \leq v_{1,i,j}$. Therefore, $v_{1,i,j}$ is the robot minimum linear velocity to pass before the obstacle passes without collision, and $v_{2,i,j}$ the maximum linear velocity to pass after it. They are represented in Fig. 3(b), and they are computed as:

$$\begin{aligned} \omega_{k,i,j} &= \frac{\theta_{k,j}}{t_{k,i,j}} = \frac{\operatorname{atan2}\left(2x_{k,i,j}y_{k,i,j}, x^2_{k,i,j} - y^2_{k,i,j}\right)}{t_{k,i,j}}, \quad k = 1,2 \\ v_{k,i,j} &= r_j \omega_{k,i,j}, \quad k = 1,2 \end{aligned} \tag{9}$$

The computation of $\boldsymbol{u}_{k,i,j} = (\omega_{k,i,j}, v_{k,i,j})$ for $\mathcal{T}$ and all the obstacles sets the limits of the DOV. The union of the DOV and the free robot velocity space is the DOVS, represented in Fig. 3.



(a) (b)

Fig. 3. A robocentric view of $\mathcal{W}$ (a) and a graphical representation of the DOVS (b) of a scenario where a robot faces the obstacle i that follows a linear trajectory. In (a), the robot center is represented in red, the trajectories $\tau_j \in \mathcal{T}$ sampled in blue lines, the obstacle augmented radius with a blue circle and the collision band $\mathcal{B}_i$ with green lines. The intersection points between τ_j and $\mathcal{B}_i$ are $P_{1,i,j}$ (right) and $P_{2,i,j}$ (left). In (b), the maximum velocities to pass after the obstacle are represented with black dots ($\boldsymbol{u}_{2,i,j}$) and the minimum velocities to pass before it with purple dots ($\boldsymbol{u}_{1,i,j}$). The DOV is represented in gray and the FV in white. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Remark 1. The DOVS is used for representing information of the scenario accessible by the robot in $\mathcal{V}$. This representation is therefore valid even though the future trajectories of obstacles are unknown, and the obstacles change their velocities in time. The DOVS is recomputed every control period with the updated information, which is then used for the decision making.

The differential-drive velocity constrains of Eqs. (3) and (4) are directly constructed in the DOVS, as shown in Fig. 2. This combination is used to learn how to navigate safely, as explained in the following sections.

4.2. Reinforcement learning setup

The problem of robot autonomous navigation in dynamic scenarios may be considered as a sequential decision-making problem to find an efficient policy that leads the robot to the goal while reactively adapting to the changes in the environment to avoid collisions. It may be modeled as a Partially Observable Markov Decision Process (POMDP) with a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, $T(\mathbf{s}_{t+1} \mid \mathbf{a}_t, \mathbf{s}_t)$ the transition model, $O(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1}, \mathbf{a}_t)$ the observation model and $R(\mathbf{s}_t, \mathbf{a}_t)$ the reward function. At every time step t, the robot takes an action $\mathbf{a}_t \in \mathcal{A}$ in an environment whose state is $\mathbf{s}_t \in \mathcal{S}$ by following a policy $\pi(\mathbf{s}_t)$. The agent uses information observed from the environment, $\mathbf{o}_t \in \mathcal{O}$, and the action taken modifies the environment given the transition model, resulting the next state $\mathbf{s}_{t+1} \in \mathcal{S}$. The goal of the learning process is estimating the optimal policy π^* that maximizes the expected cumulative reward:

$$\begin{aligned} \pi^*(\mathbf{s}_t) = \arg\max_{\mathbf{a}_t \in A} \Bigg[& R(\mathbf{s}_t, \mathbf{a}_t) + \gamma \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} T(\mathbf{s}_{t+1} \mid \mathbf{a}_t, \mathbf{s}_t) \\ & \sum_{\mathbf{o}_{t+1} \in \mathcal{O}} O(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1}, \mathbf{a}_t) V(\mathbf{s}_{t+1}) \Bigg] \end{aligned} \tag{10}$$

where γ is a discount factor to balance the present value of future rewards, and $V(\mathbf{s}_t)$ is the value function, which represents the expected cumulative reward starting from state $\mathbf{s}_t$ and following the policy $\pi(\mathbf{s}_t)$:

$$\begin{aligned} V(\mathbf{s}_t) = & R(\mathbf{s}_t, \pi(\mathbf{s}_t)) + \gamma \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} T(\mathbf{s}_{t+1} \mid \pi(\mathbf{s}_t), \mathbf{s}_t) \\ & \sum_{\mathbf{o}_{t+1} \in \mathcal{O}} O(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1}, \pi(\mathbf{s}_t)) V(\mathbf{s}_{t+1}) \end{aligned} \tag{11}$$

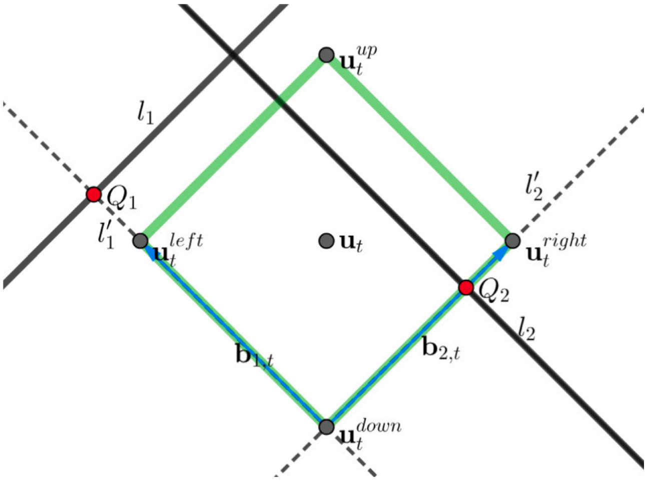


Fig. 4. Graphical representation of the notation used in the action space configuration, attaining for restrictions previously represented in Fig. 2. $\mathbf{u}_t$, $\mathbf{u}_t^{up}$, $\mathbf{u}_t^{down}$, $\mathbf{u}_t^{left}$ and $\mathbf{u}_t^{right}$ are represented with gray points, l_1 and l_2 with black lines, l_1' and l_2' with dotted lines, $\mathbf{b}_{1,t}$ and $\mathbf{b}_{2,t}$ with blue arrows, and Q_1 and Q_2 with red points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2.1. State space

The state $\mathbf{s}_t \in \mathcal{S}$ is the unobservable true condition of the system that drives the dynamics of the environment. In our problem, we consider that there is a robot with radius r_{robot} position $\mathbf{p}_t$ and velocity $\mathbf{u}_t$ at time t that drives towards a goal $\mathbf{g}$. The rest of the state is the set of points occupied by the set of static and dynamic obstacles $\mathcal{O}_{\forall,t}$, the radius $\mathbf{r}_{obs} \in \mathbb{R}^M$ and velocity $\mathcal{V}_t = \{(v_i, \omega_i)\forall i \in [0 \dots M]\}$ of the M obstacles that are dynamic:

$$\mathbf{s}_t = \{r_{robot}, \mathbf{p}_t, \mathbf{u}_t, \mathbf{g}, \mathcal{O}_{\forall,t}, \mathbf{r}_{obs}, \mathcal{V}_t\} \tag{12}$$

4.2.2. Action space

The action $\mathbf{a}_t \in \mathcal{A}$ should account for the kinematics and dynamics of the robot stated in Eq. (3) and Eq. (4). To do so, we geometrically design an action space that ensures the constraints are not violated by construction. This construction is represented in Fig. 4. Having that the robot has the velocity $\boldsymbol{u}_t$ in the current control period, $\boldsymbol{u}_{t+1}$ must not be outside the green rhombus (the dynamic window of Eq. (4)) and must not surpass l_1 nor l_2 (Eq. (3)). The idea of the action space is solving this problem by selecting velocities that respect the restrictions by definition. First, we construct an action space that make velocities be always in the dynamic window. Second, we constraint the dynamic window if it intersects with constraints in Eq. (3).

We define a continuous action space of two values $\mathbf{a}_t = (a_{1,t}, a_{2,t}) \in [0,1]^2$ that linearly combine $\mathbf{b}_{1,t}$ and $\mathbf{b}_{2,t}$, to compute velocities that always remain inside the dynamic window:

$$\mathbf{u}_{t+1} = \mathbf{u}_t^{down} + a_{1,t}\mathbf{b}_{1,t} + a_{2,t}\mathbf{b}_{2,t} \tag{13}$$

To do so, the four corners of the dynamic window are calculated:

$$\begin{aligned}
\mathbf{u}_t^{up} &= (\omega_t, v_t + a_{max}\Delta t)\\
\mathbf{u}_t^{down} &= (\omega_t, v_t - a_{max}\Delta t)\\
\mathbf{u}_t^{left} &= (\omega_t - \frac{\omega_{max} a_{max} \Delta t}{v_{max}}, v_t)\\
\mathbf{u}_t^{right} &= (\omega_t + \frac{\omega_{max} a_{max} \Delta t}{v_{max}}, v_t)
\end{aligned} \tag{14}$$

Using them, the two vectors to form a basis in the velocity space are defined:

$$\begin{aligned}
\mathbf{b}_{1,t} &= \mathbf{u}_t^{left} - \mathbf{u}_t^{down}\\
\mathbf{b}_{2,t} &= \mathbf{u}_t^{right} - \mathbf{u}_t^{down}
\end{aligned} \tag{15}$$

The kynematic restriction of Eq. (3) (l_1 and l_2 in Fig. 4) are considered by limiting $\boldsymbol{a}_t$ maximum value: $a_{1,t} \in [0, a_{1,t}^{max}]$ and $a_{2,t} \in [0, a_{2,t}^{max}]$. This limit is only needed if the dynamic window intersects with l_1 or l_2. In the velocity space, the lines that pass through $\mathbf{u}_t^{down} = (\omega_t^{down}, v_t^{down})$ and have the direction of $\mathbf{b}_{1,t}$ and $\mathbf{b}_{2,t}$ are l_1' and l_2', respectively:

$$\begin{aligned}
l_1' : v &= -\frac{v_{max}\omega}{\omega_{max}} + \frac{v_{max}\omega_t^{down}}{\omega_{max}} + v_t^{down}\\
l_2' : v &= \frac{v_{max}\omega}{\omega_{max}} - \frac{v_{max}\omega_t^{down}}{\omega_{max}} + v_t^{down}
\end{aligned} \tag{16}$$

And the intersection points between l_1 and l_1', and l_2 and l_2' are $Q_1(\omega_{1,q}, v_{1,q})$ and $Q_2(\omega_{2,q}, v_{2,q})$, with:

$$\begin{aligned}
\omega_{1,q} &= \frac{1}{2}(\frac{\omega_{max} v_t^{down}}{v_{max}} + \omega_t^{down} - \omega_{max})\\
v_{1,q} &= \frac{1}{2}(\frac{v_{max}\omega_t^{down}}{\omega_{max}} + v_t^{down} - v_{max}) + v_{max}\\
\omega_{2,q} &= \frac{1}{2}(-\frac{\omega_{max} v_t^{down}}{v_{max}} + \omega_t^{down} + \omega_{max})\\
v_{2,q} &= \frac{1}{2}(-\frac{v_{max}\omega_t^{down}}{\omega_{max}} + v_t^{down} - v_{max}) + v_{max}
\end{aligned} \tag{17}$$

The action $\mathbf{a}_t$ will be bounded only if the distance between $\mathbf{u}_t^{down}$ and Q_1 and Q_2 is smaller than $\mathbf{b}_{1,t}$ or $\mathbf{b}_{2,t}$, since the dynamic window will be in collision with l_1 or l_2. Therefore:

$$\begin{aligned}
a_{1,t}^{max} &= \min\left(1, \frac{\| P_1 - \mathbf{u}_t^{down} \|}{\|\mathbf{b}_{1,t}\|}\right)\\
a_{2,t}^{max} &= \min\left(1, \frac{\| P_2 - \mathbf{u}_t^{down} \|}{\|\mathbf{b}_{2,t}\|}\right)
\end{aligned} \tag{18}$$

4.2.3. Observation space, observation model and transition model

The observation $\mathbf{o}_t \in \mathcal{O}$ states what the robot perceives from the environment. The robot estimates from the sensor data the values of the state, but its goal and radius, which are already known:

$$\mathbf{o}_t = \{r_{robot}, \hat{\mathbf{p}}_t, \hat{\mathbf{u}}_t, \mathbf{g}, \hat{\mathcal{O}}_{\forall,t}, \hat{\mathbf{r}}_{obs}, \hat{\mathcal{V}}_t\}, \tag{19}$$

where $\hat{\mathbf{p}}_t$ is the estimated robot position, $\hat{\mathbf{u}}_t$ the estimated robot velocity, $\hat{\mathcal{O}}_{\forall,t}$ the estimated position of obstacles, and $\hat{\mathbf{r}}_{obs}$ and $\hat{\mathcal{V}}_t$ the estimated radius and velocity of the moving obstacles.

The robot is modeled to have a 2-D LiDAR laser sensor. It estimates $\hat{\mathbf{p}}_t$ and $\hat{\mathbf{u}}_t$ with the laser data and the odometry, using Adaptive Monte Carlo Localization (AMCL) [56]. An extended version of the work proposed by [57] is used to detect static and dynamic obstacles and estimate $\hat{\mathcal{O}}_{\forall,t}$, $\hat{\mathbf{r}}_{obs}$ and $\hat{\mathcal{V}}_t$ from the 2-D LiDAR measurements. It detects circular and linear obstacles from the scans, by using a grouping and splitting algorithm, associating obstacles of any shape to circles or a set of segments. The circular obstacles are tracked in time with a Kalman filter, wich also estimates its velocity assuming constant velocity in the update step of the Kalman filter. We refer the readers to [57] for more details about the tracking. This kind of conditions makes the robot face simulated occlusions and estimation errors.

We assume that the robot follows the deterministic transition model of Eq. (1), subject to the restrictions stated in Eq. (3) and Eq. (4). We model the dynamic obstacles using circular motion with constant random linear and angular velocity. The obstacles avoid each other using ORCA [15], resulting in motion that is different from the constant velocities previously defined. The robot is invisible for the obstacles, making the scenario more challenging. Otherwise, the robot could not learn to avoid non-cooperative obstacles.

4.2.4. Reward function

The goal of the agent is reaching the goal while avoiding collisions in the shortest time possible. To achieve this behavior, the reward functions proposed in [58,59] are used as inspiration. It is defined with a simple equation that discriminates between terminal and non-terminal states at time t:

$$R(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} r_{goal}, & d_{t,goal} < 0.15 \\ r_{collision}, & \text{collision detected} \\ -r_{dist}\Delta d_{t,goal} + r_{t,safedist}, & \text{otherwise} \end{cases}, \tag{20}$$

where $d_{t,goal} = \|\mathbf{g} - (x_t, y_t)\|$ and $\Delta d_{t,goal} = d_{t,goal} - d_{t-1,goal}$. The robot receives a reward of $r_{goal} = 15$, when it reaches the goal within a certain threshold; and a negative reward $r_{collision} = -15$ when it collides. Reward shaping is used to accelerate training in non-terminal states by encouraging the agent to get closer to the goal ($-r_{dist}\Delta d_{t,goal}$ with $r_{dist} = 2.5$), and by penalizing the agent if it is too close to an obstacle with:

$$r_{t,safedist} = \begin{cases} -0.1|0.2 - d_{t,obs}|, & d_{t,obs} < 0.2 \\ 0, & \text{otherwise} \end{cases}, \tag{21}$$

where $d_{t,obs}$ is the distance to the closest obstacle.

4.3. Network

We propose using a Soft Actor Critic (SAC) [23] algorithm to solve the problem, and use the DOVS model as part of the input of the network to extract the dynamism of the environment. The implementation of Stable Baselines 3 [60] is used for the SAC. SAC is an off-policy DRL algorithm that uses an actor to determine the policy and a critic to estimate the state–action values, and introduces entropy maximization to help exploration; it is known for its stability and sample efficiency.

Using raw velocity information of obstacles would require training with obstacles of every possible shape, velocity or radius the robot could face, which is impossible; so using the DOVS to model the scenario improves the adaptation and generalization of the network. The complete structure of the network is represented in Fig. 5. We process the observation of the environment with two different streams. The first stream uses a discrete set of trajectories formed with velocities ranging from the maximum and minimum velocities of the robot and equally spaced:

$$\begin{aligned} \mathcal{T}_{RUMOR} = \Big\{ & \tau_j(\omega_j, v_j) \in \mathcal{T} \mid \omega_j \in (\omega_0, \dots, \omega_{N_\omega}), \\ & v_j \in (v_0, \dots, v_{N_v}),\ \omega_0 = \omega_{min},\ \omega_{N_\omega} = \omega_{max}, \\ & v_0 = v_{min},\ v_{N_v} = v_{max} \Big\} \end{aligned} \tag{22}$$

where N_ω is set to 40 and N_v to 20. The DOVS of this set is represented with a grid of $\{-1, 1\}$ discrete values indexed by ω_j and v_j representing whether $\tau(\omega_j, v_j) \in DOV(\mathcal{O}_{\forall,t}, \mathcal{T})$, the unsafe velocities, or $\tau(\omega_j, v_j) \in FV(\mathcal{O}_{\forall,t}, \mathcal{T})$, the safe velocities, respectively. It is processed with a convolutional network of three convolutional layers and a fully connected layer all of them followed by ReLU activation functions, ψ_{DOVS}, to preserve the relationships among velocities in the velocity space. Additionally, the following other observation variables $\mathbf{o}_{t,state}$ are separately processed with a fully connected layer with ReLU activation, ψ_{obs}:

$$\mathbf{o}_{t,state} = \left\{ \hat{\mathbf{u}}_t, \hat{d}_{t,goal}, \hat{\phi}_{t,goal}, \hat{\mathcal{O}}_{t,dist}, \hat{\mathcal{O}}_{t,\theta}, \hat{\mathcal{O}}_{v,t}, \hat{\mathcal{O}}_{t,dir} \right\}, \tag{23}$$

where $\hat{d}_{t,goal}$ and $\hat{\phi}_{t,goal}$ are the distance and angle to the goal with respect to the estimated heading angle of the robot (θ_t), $\hat{\mathcal{O}}_{t,dist}$ the estimated distance of the robot to the closest obstacle, $\hat{\mathcal{O}}_{t,\theta}$ the estimated angle to it, $\hat{\mathcal{O}}_{v,t}$ its estimated velocity and $\hat{\mathcal{O}}_{t,dir}$ its estimated heading angle. Even though the information of these four last variables is already encoded in DOVS, they are included to give more information to the robot in case there is an imminent collision.

The output of both streams is concatenated and fed to an LSTM layer [43], ψ_{LSTM}, to save information of previous observation and keep temporal dependencies. The future of the environment is already considered by DOVS, but considering the past could help the network understand changes in the obstacles behavior and make it more robust to estimation errors or occlusions. Finally, a fully connected layer with ReLU activation function, ψ_{FC}, produces the final encoded observation $\mathbf{o}_{t,enc}$:

$$\begin{aligned} \mathbf{o}_{t,enc} = \psi_{FC}\Big(\psi_{LSTM}\Big(\psi_{DOVS}\left(DOVS\left(\mathcal{O}_{\forall,t}, \mathcal{T}_{RUMOR}\right)\right), \\ \psi_{obs}\left(\mathbf{o}_{t,state}\right)\Big)\Big) \end{aligned} \tag{24}$$

An overview of the actor-critic architecture is shown in Fig. 6. It may be seen that the encoder network is duplicated for the actor and the critic, so that they can separately optimize the weights for their objective.

5. Experiments

This section details RUMOR evaluation, both with simulation experiments that compare it with existing motion planners (Section 5.2 and hardware experiments that prove its performance in a ground robot (Section 5.3.

5.1. Experimental setup

Simulation setup. The planners are tested in a modified version of the Stage simulator [61]. It is also used for training RUMOR and the DRL baselines. This simulator is conveniently integrated in ROS to use AMCL for localization and realistically simulates a LiDAR sensor. The scenario used is an 6x6 m open space with dynamic obstacles the robot has to avoid to reach the goal. The scenario has walls far from this space to let AMCL localize the robot. The obstacles try to avoid each other using the ORCA algorithm, but cannot see the robot to make it perform the whole obstacle avoidance maneuvers, as stated in Section 4.2.3. Random positions are sampled for the robot, its goal and the other obstacles. The obstacles have a predefined random linear and angular velocity that try to follow. The linear velocity of obstacles is sampled between $\frac{v_{max}}{5}$ and v_{max} m/s, where v_{max} is the robot's maximum velocity. This range ensures variability in the obstacle behavior. The robot velocity limits are set to $v_{max} = 0.7$ m/s and $\omega_{max} = \pi$ rad/s and the maximum acceleration to $a_{max} = 0.3$ m/s^2, similar to those of a Turtlebot 2 platform. Although typical human walking speeds exceed the obstacles velocity range, setting it to match the robot's maximum speed allows for effective training in obstacle avoidance, despite the TurtleBot 2 being relatively slow. Moreover, the speed of people walking in crowds and when facing other people in open spaces is usually lower than the typical values.

We conducted the simulation experiments with two different perception settings: using the ground truth position of the dynamic obstacles given by the simulator (absolute perception) and using the Kalman Filter-based obstacle detector [57] (obstacle tracker) previously introduced in Section 4.2.3, thus facing occlusions and estimation errors.

Training setup. During training, the robot faces scenarios with increasing number of dynamic obstacles, up to 14, and increasing starting distance to the goal during a first training stage of 1000 episodes. In this way, it learns to reach the goal progressively by using curriculum learning, from simple to more complex tasks. After that stage, it faces 9000 scenarios with a random number of obstacles up to 14. The minimum distance between the robot and the goal is set to 6 m.

If the simulation takes more than 500 time steps, the episode is finished due to timeout. The episode time step is set to 0.2 s. The network converges in about 10 h in a computer with a Ryzen 7 5800x processor, a NVIDIA GeForce RTX 3060 graphics card and 64 GB of RAM. The key parameters used are a learning rate of 0.0003 with Adam optimizer [62], discount factor of 0.99 and soft update coefficient of 0.005. Other parameters can be consulted in the code.

All DRL policies (both RUMOR and those used as baselines) have been trained with this setup. In addition, they have been trained in both perception settings to properly train them, i.e., there are two versions

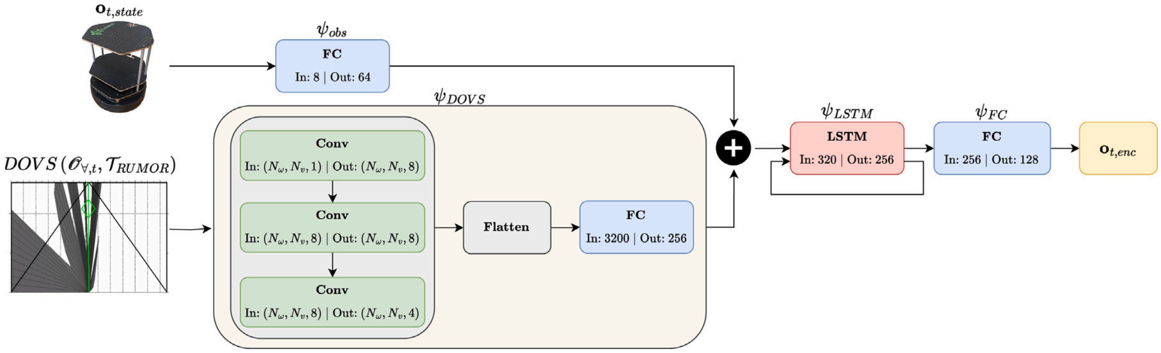

$\mathbf{o}_{t,state}$
ψ_{obs}
FC
In: 8 | Out: 64
ψ_{DOVS}
$DOVS\,(\mathcal{O}_{\forall,t}, \mathcal{T}_{RUMOR})$
Conv
In: $(N_\omega, N_v, 1)$ | Out: $(N_\omega, N_v, 8)$
Conv
In: $(N_\omega, N_v, 8)$ | Out: $(N_\omega, N_v, 8)$
Conv
In: $(N_\omega, N_v, 8)$ | Out: $(N_\omega, N_v, 4)$
Flatten
FC
In: 3200 | Out: 256
ψ_{LSTM}
LSTM
In: 320 | Out: 256
ψ_{FC}
FC
In: 256 | Out: 128
$\mathbf{o}_{t,enc}$

Fig. 5. Structure of the encoding network. The DOVS model and the robot state are processed in two different streams, joint later by a memory layer that accounts for previous observations.

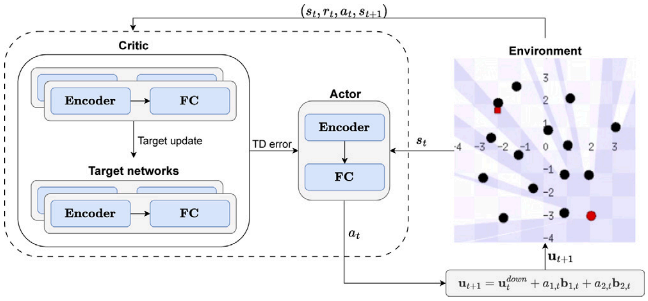

(s_t, r_t, a_t, s_{t+1})
Critic
Encoder
FC
Target update
Target networks
Encoder
FC
TD error
Actor
Encoder
FC
s_t
Environment
a_t
$\mathbf{u}_{t+1}$
$\mathbf{u}_{t+1} = \mathbf{u}_t^{down} + a_{1,t}\mathbf{b}_{1,t} + a_{2,t}\mathbf{b}_{2,t}$

Fig. 6. Structure of the SAC architecture used. The critic uses the whole transition for estimating the TD error, used to train the actor. The actor uses the environment observation to select an action, which is translated to a velocity. The environment included is the Stage simulator, used in training.

of each of the DRL planners, one trained using absolute perception and the other using the obstacle tracker.

Hardware setup. RUMOR was integrated in a Turtlebot 2 platform with a NUC with Intel Core i5-6260U CPU and 8 GB of RAM. The sensor used is a 180° Hokuyo 2D-LIDAR. Due to the real-world design of the system, the same network weights and ROS nodes used in simulation were used in the ground robot, with the same Kalman Filter-based obstacle detector and AMCL for localization. Therefore, the RUMOR version used in the simulation experiments with the obstacle tracker is the same one as the used as in the hardware ones, with the same network weights. Obstacles like walls are detected as segments by the obstacle detector. They are included in the DOVS as colliding velocities by simply computing the velocities that intersect with the segment within a time horizon. This allows the robot to simply detect them without modifying the design.

To help the robot navigate along long-paths, through rooms and avoid other non-convex obstacles, we use a way-point generator introduced in [63] to feed the robot with intermediate goals rather than simply sending the final goal to it. The idea behind it is first computing a global plan with A* regarding the static map of the environment and filter the plan to get sparse intermediate points that are corners of sharp turns. A new goal from this set of intermediate goals is sequentially set as the RUMOR goal when the robot is close to approaching the previous one, letting RUMOR compute the motion commands by itself but with intermediate guidance that helps it to avoid convex obstacles.

5.2. Simulation experiments

We conducted quantitative experiments to evaluate the performance of our proposed model. We compared our method, RUMOR, with DWA [5], TEB [9], ORCA [15], SGAN-MPC [64], S-DOVS [1], SARL [45] , RGL [46], RE3-RL [48] and SG-D3QN [52].

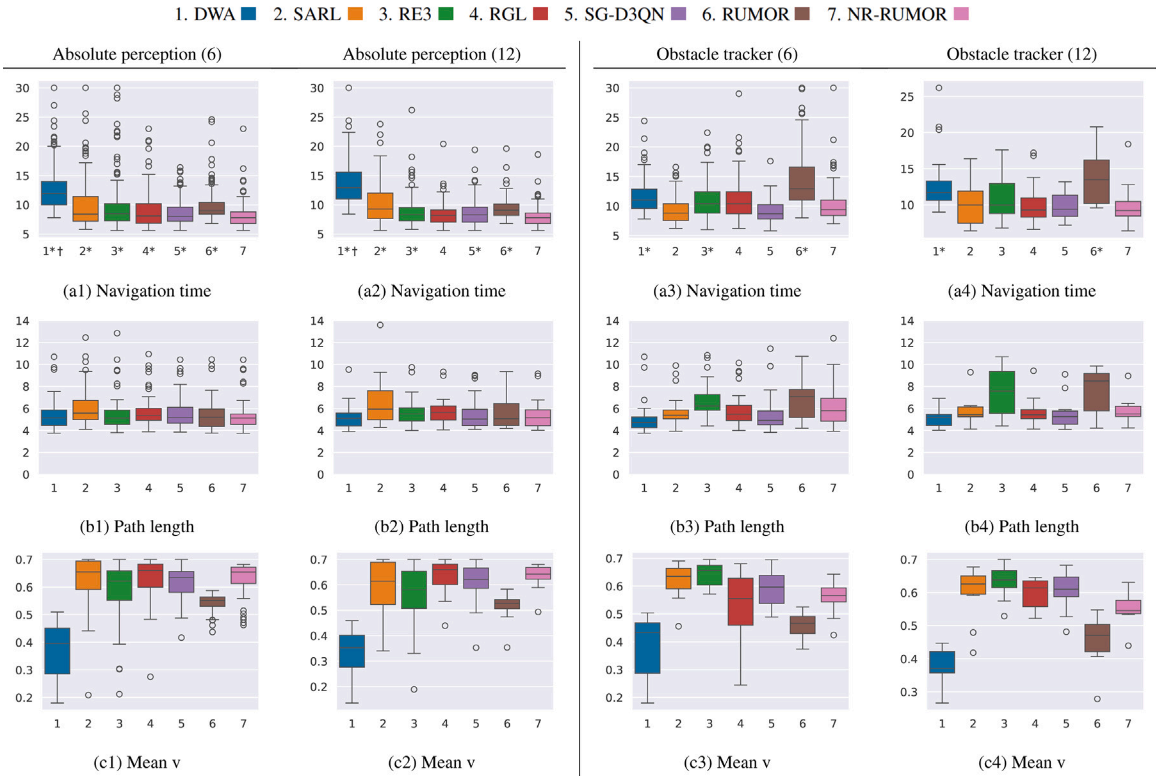
Remark 2. The comparison of RUMOR with the baselines planners is unfair, as they do not consider the same velocity constraints. ORCA, SGAN-MPC, SARL, RGL, RE3-RL and SG-D3QN only consider maximum velocity limits, overlooking constraints of Equations 3 and 4. For a more fair comparison, we consider a different version of RUMOR with no restrictions (NR-RUMOR). NR-RUMOR has a continuous action space that comprises the whole robot velocity range: $\mathbf{a}_t = (a_{1,t}, a_{2,t}) \in [0, v_{max}] \times [-\omega_{max}, \omega_{max}]$.

Remark 3. We used ROS navigation stack implementations of DWA and TEB, which considers acceleration restrictions in v and ω separately. As shown before, the maximum linear acceleration/velocity of a differential-drive robot is directly related to its angular acceleration/velocity. RUMOR accounts for these restrictions, but DWA and TEB do not. The result of this difference may be understood with Fig. 2. First, the restrictions implemented in DWA and TEB do not consider kinematic constraints of Eq. (3), so their velocities may surpass the two black lines. Second, the acceleration constraints would be equivalent to, instead of using the rhombus of Eq. (4) as the dynamic window around the current robot velocity, using a bounding box that surrounds the rhombus. DWA and TEB therefore have an acceleration and velocity space that doubles the size of RUMOR's, which accounts for every differential-drive constraint.

Remark 4. We selected as agent-based DRL baselines SARL, RGL, RE3-RL and SG-D3QN. Even though we may adapt RUMOR to their restrictions (NR-RUMOR), these baselines may not be adapted to consider the differential-drive constraints. First, they use a discrete action space, so RUMOR action space is not available for them. Second, it is not clear how to restrict a discrete action space avoiding issues in Remark 3. Third, modifying those methods to use a continuous action space would imply changing their algorithm, which could alter training time or stability.

We tested the planners in the same 500 random scenarios with low and high occupancy, with 6 and 12 dynamic obstacles, respectively. The 85% of the obstacles are set to be dynamic and the same set of scenarios is used for every method.

We mainly evaluate the planners using the two navigation metrics directly encoded in the reward function, which are the success rate (number of times the robot have reached the goal without collisions or timeouts) and the navigation time. There is a trade-off between both metrics. Trying to achieve a low navigation time leads to risky collision avoidance maneuvers, which leads to lower success rates. Collisions are sometimes unavoidable, particularly in dense environments or when estimation errors occur. In such cases, the robot may encounter trapping situations from which it cannot escape or face obstacles that move unpredictably towards it. Thus, achieving consistently high success rates in complex scenarios is unrealistic, so these results should be interpreted as a benchmark. In real-world settings, where obstacles



(a1) Navigation time

(a2) Navigation time

(a3) Navigation time

(a4) Navigation time

(b1) Path length

(b2) Path length

(b3) Path length

(b4) Path length

(c1) Mean v

(c2) Mean v

(c3) Mean v

(c4) Mean v

Fig. 7. Navigation metrics for dynamic scenarios of 6 (left) and 12 (right) obstacles using absolute perception and the obstacle tracker. In (a), * denotes statistical significant difference between NR-RUMOR and a method, and *† between both NR-RUMOR and RUMOR and a method, with p-value < 0.05.

Table 1
Success rates for the different methods for the simulation experiments, using absolute perception and the obstacle tracker. The method with the best score in each setup is in bold.

Method	Abs. Per.		Obs. Track.	
	6 Obs.	12 Obs.	6 Obs.	12 Obs.
TEB	0.47**	0.10**	0.47**	0.10**
DWA	0.69**	0.44**	0.70*	0.46*
S-DOVS	0.70**	0.36**	0.58**	0.37**
SGAN-MPC	0.66**	0.51**	0.56**	0.36**
ORCA	0.66**	0.51**	0.60**	0.35**
SARL	0.81**	0.64**	0.74	0.46*
RE3	0.87**	0.71*	0.64**	0.35**
RGL	0.88*	0.68*	0.65*	0.39**
SG-D3QN	0.91	0.78	0.72*	0.50*
RUMOR	0.91	0.72*	0.70*	0.46*
NR-RUMOR	**0.94**	**0.82**	**0.78**	**0.58**

* denotes statistical significant difference between NR-RUMOR and a method.
** between both NR-RUMOR and RUMOR and a method, with a p-value < 0.05.

typically cooperate in collision avoidance or at least do not actively move towards the robot, the success rate is likely to be higher.

The success rates obtained in the experiments are presented in Table 1. Scenarios in which all methods failed were excluded from the evaluation metrics. To assess statistical significance, we performed a χ^2 test to compare the success rate of NR-RUMOR and RUMOR against the baselines. In addition, we recorded the time to reach the goal, mean velocity and path length metrics for each testing episode where all methods succeeded. The results are visualized as box-plots in Fig. 7 for DWA and the DRL planners (the rest are not included to simplify visualization). Furthermore, we conducted Mann–Whitney U rank tests with a one-sided ("less") alternative hypothesis to statistically determine if RUMOR and NR-RUMOR achieved significantly lower navigation times compared to the other methods.

Overall, NR-RUMOR has the highest success rate in all settings, while having lower or at least comparable navigation times. Moreover, the difference with the rest of the planners is statistically significant in most of cases. The only case where the difference is not significant in both low and high density scenarios is when comparing NR-RUMOR with SG-D3QN in absolute perception settings. However, the mean time taken by NR-RUMOR planner to reach the goal is significantly lower in those cases.

The success rates show that the agent-model based planners (S-DOVS, SGAN-MPC and ORCA) performance is worse than DRL-based ones regarding their success rates. These planners use the same agent information DRL planners as input. They model the environment in every time step, but they are not robust when errors or deep changes in the surroundings occur.

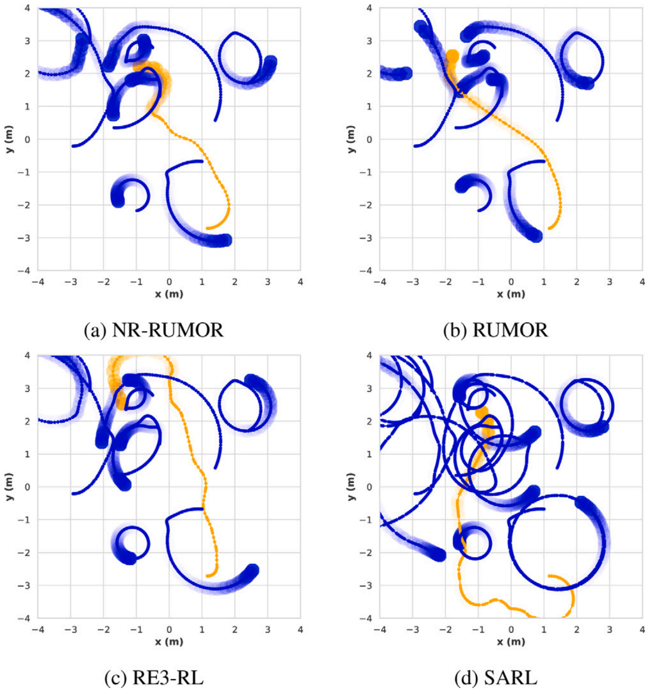
Apart from TEB and DWA, which do not use the agent information as input, every planner shows higher success rates and lower navigation times with absolute perception, due to the absence of partial observability and estimation errors. It is way more complex to predict the future of the environment when facing perception problems. Therefore, their relative performance within the same settings should be compared. It is interesting to see that there is an inconsistency in the performance of SARL and RE3-RL or RGL when changing the perception settings. When comparing them using absolute perception, RE3-RL and RGL outperform SARL in both success rate and navigation time, while when using the obstacle tracker is the other way around. This is probably due to the difference in the complexity of their networks and the exploration process, degrading the performance of complex models in the presence of erroneous observations. In contrast, both NR-RUMOR show a very consistent relative performance no matter the perception settings. SG-D3QN shows a consistent performance too, but, as stated before, with weaker results than NR-RUMOR.

RUMOR underperforms NR-RUMOR as expected, as kinematic and dynamic restrictions limits the capacity of maneuvering and sudden reactions. Nonetheless, it displays a better performance, or at least comparable, than the rest of the planners. The difference in the behavior is seen in the velocity and path length graphs. While RUMOR tries to reach the maximum linear velocity as the rest of the planners, the acceleration limits prevents it from achieving it as much time as the rest. In addition, it has to keep a lower velocity in dangerous zones, as it must stop gradually. The path length and navigation time plots show that RUMOR achieve high success rates with longer paths, meaning that it must respect the restrictions to navigate. It follows safer paths where no sudden velocity changes are needed.

TEB and DWA performance is the same in both perception settings, as their input, which is the sensor scan and not the individual agent information, is the same in both cases. TEB suffers from the same issues as the other model-based planners. DWA performance is comparable or better than other methods when they suffer from perception errors. It is similar to the restricted RUMOR in those cases. Nevertheless, in absolute perception settings, RUMOR outperforms DWA in both success rate and time to reach the goal with statistical significance, even considering Remark 3. Thus, it seems like using more accurate perception algorithms (e.g. deep learning) would increase the difference between both methods.

Two key components of the system explain NR-RUMOR results. First, the input of RUMOR network is the DOVS model of the environment. With 12 surrounding dynamic obstacles, there are unlimited number of possible scenario configurations, and the agent cannot be trained to have a robust performance in all of them. On the contrary, DOVS represents the environment in similar terms no matter the scenario, leading to a deeper understanding of the velocity space. When the robot learns to interpret DOVS, it knows how to avoid collisions regardless the amount, positions or velocities of the obstacles. Second, the memory layer of RUMOR network allows it to keep track of previous observations. This is important to overcome partial observability challenges and those derived from sudden changes in the obstacles behavior. Overall, considering every combination in the perception and occupancy settings, NR-RUMOR consistently shows the best performance when compared with the other methods.

An example of the behavior of the different DRL planners is shown in Fig. 8, in absolute perception settings. All the planners reach the goal successfully, but both RUMOR versions do it in the shortest time, the shortest path lengths and with the most simple trajectories. RUMOR presents the smoothest trajectory, due to the acceleration restrictions, whereas all the other planners present a higher path irregularity. Its action space is the only one that does not allow the robot to reach maximum or minimum velocities at any time step, so it must be consistent with the navigation decisions it makes. In addition, as expected from the metrics results (Fig. 7 (b1), (c1), (b3) and (c3)), RE3-RL performs better than SARL.



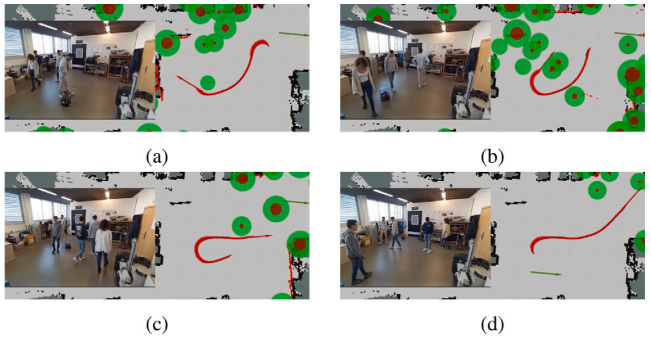
(a) NR-RUMOR

(b) RUMOR

(c) RE3-RL

(d) SARL

Fig. 8. A robot (orange) navigating with different planners in a scenario with 9 dynamic obstacles (blue). The evolution in time is represented with increasing opacity, being completely solid at the end of the episode. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a)

(b)

(c)

(d)

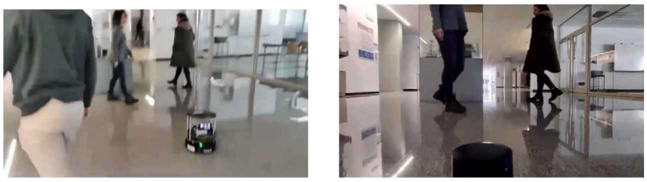
Fig. 9. Photos superposed to the visualization of four consecutive moments of a real-world experiments. The robot position is represented with a red arrow, its previous trajectory with small red arrows, the LiDAR scans with red points, the obstacle positions and radius extracted with the obstacles tracker with red and green circles and the goal of the robot with a green arrow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.3. Real-world experiments

Hardware experiments were conducted to validate and evaluate RUMOR performance in real conditions. Two different types of real-world experiments were designed, which may be seen in the supplementary video.

A set of experiments were carried out to test navigation in very dense dynamic environments. The experimental setup was a scenario with dimensions of about 6x5 m. The robot was tasked to navigate towards dynamically assigned goals in the corners of the room while encountering dynamic obstacles. Pedestrians randomly wandering within the area were used as obstacles. To ensure the validity of the avoidance maneuvers, the individuals were instructed not to visually attend to the robot's movements. Fig. 9 shows images of the robot smoothly and safely navigating between two goals. In that specific situation, the robot must reach the goal avoiding collisions with five pedestrians and the limits of the room. The robot first waits for the first person encountered to pass (a). Then, it adapts its trajectory to pass between two people (b) and keeps the collision avoidance maneuver to avoid another one (c). Finally, it continues its way to the goal (d). RUMOR is able to use the DOVS representation of the dynamism of the environment to choose natural and smooth trajectories with low path irregularities to reach the goal, instead of abruptly reacting to obstacles.

Finally, a setup different from a lab scenario was tried, in settings including a corridor with some rooms and intersections, obstacles with different shapes and people with heterogeneous behaviors, as in real life, to prove that the system works in a completely unseen environment that has not been specifically used in simulation. This may be seen in Fig. 10. The results show that the robot is able to navigate very smoothly through the dynamic obstacles that have behaviors different from those seen in training, avoids static obstacles that had not been seen before, and works with the global planner to be able to reach goals that are very far from the initial position. The use of DOVS, instead of being a complete end-to-end planner, limits out-of-distribution observations, as apparently different scenarios could result in similar DOVS representations.



(a) Third person point of view (b) Robot's point of view

Fig. 10. Images taken from the corridor experiment.

6. Conclusion

This work presents a novel motion planner for dynamic environments. It uses the DOVS model to extract the dynamism of the environment, abstracting from specific sensor data directly used by end-to-end approaches, and DRL to select motion commands that efficiently leads the robot to a goal while avoiding collisions with moving obstacles. Instead of directly use the raw obstacle information, which may lead to out-of-distribution observations in scenarios different from the ones seen in training, the system uses DOVS, which encodes it in similar terms regardless the scenario. In addition, we propose a training framework that puts the agent in real-world setting, an advanced DRL algorithm that uses memory to mitigate partial observability issues and a novel action space that intrinsically considers differential-drive kinodynamics, in order to mitigate sim2real transition. The proposed system is tested in random scenarios with absolute perception (ground truth from the simulator) and partial observability conditions, outperforming existing methods and showing the benefits of combining a model-based approach with a DRL controller. The algorithm is also tested in a ground robot in dense, dynamic and heterogeneous scenarios.

Future work could include increasing the robustness of the method in scenarios where the obstacle motion is very irregular and unnatural, extending the model to collaborative collision avoidance with multi-robot navigation or adapting the model for 3-D navigation and UAVs. In addition, we believe that studying the impact of the perception errors in both training and deployment separately would be of great interest; considering a deeper study of different perception algorithms, the effect in neural networks with different complexity, the inclusion of explicit uncertainty estimation and the evolution of the DOVS in presence of noise.

CRediT authorship contribution statement

Diego Martinez-Baselga: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Luis Riazuelo:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Luis Montano:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by MICIU/AEI/10.13039/501100011033 and ERDF/EU under grant PID2022-139615OB-I00 and grant PRE2020-094415, and Government of Aragón under grant DGA T45-23R.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2025.105020.

Data availability

Data will be made available on request.

References

[1] M.-T. Lorente, E. Owen, L. Montano, Model-based robocentric planning and navigation for dynamic environments, Int. J. Robot. Res. 37 (8) (2018) 867–889.

[2] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, J. Oh, Core challenges of social robot navigation: A survey, ACM Trans. Human- Robot. Interact. 12 (3) (2023) 1–39.

[3] A. Francis, C. Pérez-d'Arpino, C. Li, F. Xia, A. Alahi, R. Alami, A. Bera, A. Biswas, J. Biswas, R. Chandra, et al., Principles and guidelines for evaluating social robot navigation algorithms, 2023, arXiv preprint arXiv:2306.16740.

[4] P.T. Singamaneni, P. Bachiller-Burgos, L.J. Manso, A. Garrell, A. Sanfeliu, A. Spalanzani, R. Alami, A survey on socially aware robot navigation: Taxonomy and future challenges, Int. J. Robot. Res. 43 (10) (2024) 1533–1572.

[5] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. 4 (1) (1997) 23–33.

[6] L. Kästner, T. Bhuiyan, T.A. Le, E. Treis, J. Cox, B. Meinardus, J. Kmiecik, R. Carstens, D. Pichel, B. Fatloun, et al., Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments, IEEE Robot. Autom. Lett. 7 (4) (2022) 9477–9484.

[7] C. Qixin, H. Yanwen, Z. Jingliang, An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 3331–3336.

[8] D. Helbing, P. Molnar, Social force model for pedestrian dynamics, Phys. Rev. E 51 (5) (1995) 4282.

[9] C. Rösmann, F. Hoffmann, T. Bertram, Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control, in: 2015 European Control Conference, ECC, IEEE, 2015, pp. 3352–3357.

[10] G. Ferrer, A. Garrell, A. Sanfeliu, Robot companion: A social-force based approach with human awareness-navigation in crowded environments, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 1688–1694.

[11] Y.-Q. Jiang, B.-K. Chen, B.-H. Wang, W.-F. Wong, B.-Y. Cao, Extended social force model with a dynamic navigation field for bidirectional pedestrian flow, Front. Phys. 12 (2017) 1–9.

[12] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using velocity obstacles, Int. J. Robot. Res. 17 (7) (1998) 760–772.

[13] A.K. Mackay, L. Riazuelo, L. Montano, RL-DOVS: Reinforcement learning for autonomous robot navigation in dynamic environments, Sensors 22 (10) (2022) 3847.

[14] J. Van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: 2008 IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 1928–1935.

[15] J. Van Den Berg, S.J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, in: Robotics Research: The 14th International Symposium ISRR, Springer, 2011, pp. 3–19.

[16] B. Brito, B. Floor, L. Ferranti, J. Alonso-Mora, Model predictive contouring control for collision avoidance in unstructured dynamic environments, IEEE Robot. Autom. Lett. 4 (4) (2019) 4459–4466.

[17] O. de Groot, L. Ferranti, D. Gavrila, J. Alonso–Mora, Globally guided trajectory planning in dynamic environments, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 10118–10124.

[18] S. Poddar, C. Mavrogiannis, S.S. Srinivasa, From crowd motion prediction to robot navigation in crowds, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2023, pp. 6765–6772.

[19] C. Mavrogiannis, K. Balasubramanian, S. Poddar, A. Gandra, S.S. Srinivasa, Winding through: Crowd navigation via topological invariance, IEEE Robot. Autom. Lett. 8 (1) (2022) 121–128.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[21] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: Combining improvements in deep reinforcement learning, Proc. AAAI Conf. Artif. Intell. 32 (1) (2018).

[22] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, D. Silver, Distributed prioritized experience replay, in: International Conference on Learning Representations, 2018.

[23] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.

[24] L. Kästner, R. Carstens, H. Zeng, J. Kmiecik, T. Bhuiyan, N. Khorsandhi, V. Shcherbyna, J. Lambrecht, Arena-rosnav 2.0: A development and benchmarking platform for robot navigation in highly dynamic environments, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 11257–11264.

[25] L. Kästner, V. Shcherbyna, H. Zeng, T.A. Le, M.H.-K. Schreff, H. Osmaev, N.T. Tran, D. Diaz, J. Golebiowski, H. Soh, J. Lambrecht, Demonstrating arena 3.0: Advancing social navigation in collaborative and highly dynamic environments, in: Proceedings of Robotics: Science and Systems, 2024.

[26] S. Yao, G. Chen, Q. Qiu, J. Ma, X. Chen, J. Ji, Crowd-aware robot navigation for pedestrians with multiple collision avoidance strategies via map-based deep reinforcement learning, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2021, pp. 8144–8150.

[27] Q. Qiu, S. Yao, J. Wang, J. Ma, G. Chen, J. Ji, Learning to socially navigate in pedestrian-rich environments with interaction capacity, in: 2022 International Conference on Robotics and Automation, ICRA, IEEE, 2022, pp. 279–285.

[28] W. Yu, J. Peng, Q. Qiu, H. Wang, L. Zhang, J. Ji, Pathrl: An end-to-end path generation method for collision avoidance via deep reinforcement learning, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2024, pp. 9278–9284.

[29] W. Yu, J. Peng, H. Yang, J. Zhang, Y. Duan, J. Ji, Y. Zhang, Ldp: A local diffusion planner for efficient robot navigation and collision avoidance, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2024, pp. 5466–5472.

[30] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, C. Cadena, From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots, in: 2017 Ieee International Conference on Robotics and Automation (Icra), IEEE, 2017, pp. 1527–1533.

[31] R. Guldenring, M. Görner, N. Hendrich, N.J. Jacobsen, J. Zhang, Learning local planners for human-aware navigation in indoor environments, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 6053–6060.

[32] Z. Xie, P. Dames, Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles, IEEE Trans. Robot. 39 (4) (2023) 2700–2719.

[33] J.J. Damanik, J.-W. Jung, C.A. Deresa, H.-L. Choi, Lics: Navigation using learned-imitation on cluttered space, IEEE Robot. Autom. Lett. (2024).

[34] D. Dugas, O. Andersson, R. Siegwart, J.J. Chung, Navdreams: Towards camera-only rl navigation among humans, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2022, pp. 2504–2511.

[35] J. Zhao, Y. Wang, Z. Cai, N. Liu, K. Wu, Y. Wang, Learning visual representation for autonomous drone navigation via a contrastive world model, IEEE Trans. Artif. Intell. 5 (3) (2023) 1263–1276.

[36] Y. Song, K. Shi, R. Penicka, D. Scaramuzza, Learning perception-aware agile flight in cluttered environments, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 1989–1995.

[37] N. Tsoi, A. Xiang, P. Yu, S.S. Sohn, G. Schwartz, S. Ramesh, M. Hussein, A.W. Gupta, M. Kapadia, M. Vázquez, Sean 2.0: Formalizing and generating social situations for robot navigation, IEEE Robot. Autom. Lett. 7 (4) (2022) 11047–11054.

[38] X. Puig, E. Undersander, A. Szot, M.D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. Clegg, M. Hlavac, S.Y. Min, et al., Habitat 3.0: A co-habitat for humans, avatars, and robots, in: The Twelfth International Conference on Learning Representations, ICLR, 2024.

[39] L.T. Triess, M. Dreissig, C.B. Rist, J.M. Zöllner, A survey on deep domain adaptation for LiDAR perception, in: 2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops), 2021, pp. 350–357.

[40] A.H. Raj, Z. Hu, H. Karnan, R. Chandra, A. Payandeh, L. Mao, P. Stone, J. Biswas, X. Xiao, Rethinking social robot navigation: Leveraging the best of two worlds, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, 2024, pp. 16330–16337.

[41] M. Everett, Y.F. Chen, J.P. How, Motion planning among dynamic, decision-making agents with deep reinforcement learning, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 3052–3059.

[42] Y.F. Chen, M. Liu, M. Everett, J.P. How, Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 285–292.

[43] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[44] M. Everett, Y.F. Chen, J.P. How, Collision avoidance in pedestrian-rich environments with deep reinforcement learning, IEEE Access 9 (2021) 10357–10377.

[45] C. Chen, Y. Liu, S. Kreiss, A. Alahi, Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 6015–6022.

[46] C. Chen, S. Hu, P. Nikdel, G. Mori, M. Savva, Relational graph learning for crowd navigation, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 10007–10013.

[47] S. Liu, P. Chang, Z. Huang, N. Chakraborty, K. Hong, W. Liang, D.L. McPherson, J. Geng, K. Driggs-Campbell, Intention aware robot crowd navigation with attention-based interaction graph, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 12015–12021.

[48] D. Martinez-Baselga, L. Riazuelo, L. Montano, Improving robot navigation in crowded environments using intrinsic rewards, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, 2023, pp. 9428–9434.

[49] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, S. Levine, How to train your robot with deep reinforcement learning: lessons we have learned, Int. J. Robot. Res. 40 (4–5) (2021) 698–721.

[50] G. Dulac-Arnold, N. Levine, D.J. Mankowitz, J. Li, C. Paduraru, S. Gowal, T. Hester, Challenges of real-world reinforcement learning: definitions, benchmarks and analysis, Mach. Learn. 110 (9) (2021) 2419–2468.

[51] Z. Hu, Y. Zhao, S. Zhang, L. Zhou, J. Liu, Crowd-comfort robot navigation among dynamic environment based on social-stressed deep reinforcement learning, Int. J. Soc. Robot. 14 (4) (2022) 913–929.

[52] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, Z. Zhou, Robot navigation in a crowd by integrating deep reinforcement learning and online planning, Appl. Intell. (2022) 1–17.

[53] B. Xue, M. Gao, C. Wang, Y. Cheng, F. Zhou, Crowd-aware socially compliant robot navigation via deep reinforcement learning, Int. J. Soc. Robot. (2023) 1–13.

[54] H. Yang, C. Yao, C. Liu, Q. Chen, RMRL: Robot navigation in crowd environments with risk map-based deep reinforcement learning, IEEE Robot. Autom. Lett. (2023).

[55] U. Patel, N.K.S. Kumar, A.J. Sathyamoorthy, D. Manocha, DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 6057–6063.

[56] D. Fox, W. Burgard, S. Thrun, Markov localization for mobile robots in dynamic environments, J. Artificial Intelligence Res. 11 (1999) 391–427.

[57] M. Przybyła, Detection and tracking of 2D geometric obstacles from LRF data, in: 2017 11th International Workshop on Robot Motion and Control, RoMoCo, IEEE, 2017, pp. 135–141.

[58] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, J. Pan, Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 6252–6259.

[59] A.J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, D. Manocha, Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 11345–11352.

[60] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-Baselines3: Reliable reinforcement learning implementations, J. Mach. Learn. Res. 22 (268) (2021) 1–8.

[61] B. Gerkey, R.T. Vaughan, A. Howard, et al., The player/stage project: Tools for multi-robot and distributed sensor systems, in: Proceedings of the 11th International Conference on Advanced Robotics, vol. 1, Citeseer, 2003, pp. 317–323.

[62] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[63] D. Martinez-Baselga, L. Riazuelo, L. Montano, Long-range navigation in complex and dynamic environments with full-stack S-DOVS, Appl. Sci. 13 (15) (2023) 8925.

[64] S. Poddar, C. Mavrogiannis, S.S. Srinivasa, From crowd motion prediction to robot navigation in crowds, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 6765–6772.

Diego Martinez-Baselga received the B.Eng. degree in computer science and the M.Sc. degree in Robotics, Graphics and Computer Vision from the University of Zaragoza, Spain, in 2020 and 2022, respectively, where he is working toward the Ph.D. degree in computer science and systems engineering. He was a visiting researcher at TU Delft, Netherlands, in 2023, in the Autonomous Multi-Robots Lab, and at KTH, Stockholm, Sweden, in 2025, in the Planiacs group. His research interests include motion planning and collision avoidance using learning and control methods, and multi-robot systems.

Luis Riazuelo obtained his MsC in Computer Science in 2006 and his MEng in Biomedical Engineering in 2008 from the University of Zaragoza. In 2018 he obtained his PhD in Computer Science at the University of Zaragoza. He is currently an associate profesor at the Computer Science and Systems Engineering department of the University of Zaragoza. Since 2007 he is researcher of the Robotics, Perception and Real Time group at the Aragòn Engineering Research. His current research topics include scene understanding for mobile robotics in underground environments and topological semantic mapping and localization in intracorporeal medical scenes.

Luis Montano received his degree in industrial engineering in 1981 and his doctorate in 1987 from the University of Zaragoza, Spain. He is Full Professor at the University of Systems Engineering and Automation at the University of Zaragoza. He was Director of the Department of Computer Science and Systems Engineering and Deputy Director of the Aragon Engineering Research Institute of the University of Zaragoza. He is Principal Researcher of the Robotics, Computer Vision and Artificial Intelligence research Group of the Institute. His main research interests in robotics are: planning and navigation in dynamic environments, multi-robot systems.