2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561462

DWA-RL: Dynamically Feasible Deep Reinforcement Learning Policy for Robot Navigation among Mobile Obstacles

Utsav Patel[1], Nithish K Sanjeev Kumar[1], Adarsh Jagan Sathyamoorthy[2] and Dinesh Manocha[1].

***Abstract*— We present a novel Deep Reinforcement Learning (DRL) based policy to compute dynamically feasible and spatially aware velocities for a robot navigating among mobile obstacles. Our approach combines the benefits of the Dynamic Window Approach (DWA) in terms of satisfying the robot's dynamics constraints with state-of-the-art DRL-based navigation methods that can handle moving obstacles and pedestrians well. Our formulation achieves these goals by embedding the environmental obstacles' motions in a novel low-dimensional observation space. It also uses a novel reward function to positively reinforce velocities that move the robot away from the obstacle's heading direction leading to significantly lower number of collisions. We evaluate our method in realistic 3-D simulated environments and on a real differential drive robot in challenging dense indoor scenarios with several walking pedestrians. We compare our method with state-of-the-art collision avoidance methods and observe significant improvements in terms of success rate (up to 33% increase), number of dynamics constraint violations (up to 61% decrease), and smoothness. We also conduct ablation studies to highlight the advantages of our observation space formulation, and reward structure.**

I. Introduction

There has been considerable interest in using Deep Reinforcement Learning (DRL)-based local planners [1], [2], [3], [4] to navigate a non-holonomic/differential drive robot through environments with moving obstacles and pedestrians. They are effective in capturing and reacting to the obstacles' motion over time, resulting in excellent mobile obstacle avoidance capabilities. In addition, these methods employ inexpensive perception sensors such as RGB-D cameras or simple 2-D lidars and do not require accurate sensing of the obstacles. However, it is not guaranteed that the instantaneous robot velocities computed by DRL-based methods will be *dynamically feasible* [5], [6]. That is, the computed velocities may not obey the acceleration and non-holonomic constraints of the robot, becoming impossible for the robot to move using them. This leads to highly non-smooth and jerky trajectories.

Desirable behaviors such as computing dynamically feasible velocities are developed using a DRL method's reward function, where they are positively rewarded and undesirable behaviors such as collisions are penalized. However, a fully trained policy could over-prioritize the collision avoidance behavior over dynamic feasibility, if the penalty for collision is not appropriately balanced with the reward for computing feasible velocities [7]. Therefore, acceleration limits and the non-holonomic constraints of the robot may not be satisfied. It is crucial that the policy account for such fundamental constraints especially when the robot navigates among pedestrians and other mobile obstacles.

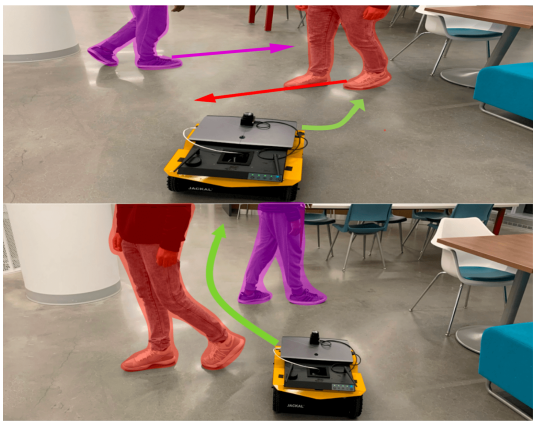

Fig. 1: Our robot avoiding mobile obstacles using dynamically feasible, smooth, spatially-aware velocities. The red and violet arrows indicate the obstacles' motion and green arrows shows the robot's trajectory in two time instances for different obstacle positions. Our hybrid approach, DWA-RL, considers the motion of the moving obstacles over time in its low-dimensional observation space which is used to compute the robot velocities. This results in fewer collisions than DWA [6], and DRL-based methods [4]. Since our method computes the robot velocities based on DWA's feasible velocity space, the computed robot velocities are guaranteed to obey the acceleration and non-holonomic constraints of the robot.

Another issue with such methods [1], [2] is that they use high-dimensional data such as RGB or depth images as inputs during training to detect and observe obstacles. This greatly increases the overall training time and makes it harder for the policy to generalize the behaviors learnt in one environment to another.

On the other hand, the Dynamic Window Approach (DWA) [6], is a classic navigation algorithm that accounts for the robot's dynamics constraints and guarantees that the velocities in a space known as the *dynamic window* are collision-free and feasible/achievable for the robot within a time horizon Δt. However, DWA's formulation only considers the robot's sensor data at the current time instant to make decisions. As a result, avoiding mobile obstacles becomes challenging, leading to higher number of collisions [8].

Main Results: We present a hybrid approach, DWA-RL, that combines the benefits of DWA and DRL-based methods for navigation in the presence of mobile obstacles. We present a DRL-based collision avoidance policy that utilizes a novel observation space formulation and a novel reward function to generate spatially aware, collision-free, dynamically feasible velocities for navigation. We show that our approach has a superior performance compared to DWA and a DRL-based method [4] in terms of success rate, number of dynamics constraints violations, and smoothness. The main contributions of our work include:

This work was supported in part by ARO Grants W911NF1910069 and W911NF1910315, and Intel.

[1] Authors are with Dept. of Computer Science, University of Maryland, College Park, MD, USA. upatel22@umd.edu, nithish@terpmail.umd.edu, dm@cs.umd.edu

[2] Author is with the Dept. of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA. asathyam@umd.edu

- A novel formulation for the observation space, based on the concept of dynamic window, is used to train our DRL-based navigation policy. The observation space is constructed by calculating the robot's feasible velocity set at a time instant and the costs corresponding to using those velocities in the past *n* time instants. This formulation embeds the time evolution of the environment's state and preserves the dynamic feasibility guarantees of DWA (Section IV). This leads to a significantly lower dimensional observation space unlike other DRL methods [1], [2]. This also results in significantly lower training times, and easier sim-to-real transfer of the fully trained policy.
- A novel reward function that is shaped such that the robot's navigation is more spatially aware of the obstacles' motion. That is, the robot is rewarded for navigating in the direction opposite to the heading direction of obstacles. This leads to the robot taking maneuvers around moving obstacles. This is different from DWA, which might navigate directly into the path of a mobile obstacle or collide with it. Overall, our approach reduces the collision rate by 33% in dynamic environments as compared to DWA.

We evaluate our method and highlight its benefits over prior methods in four high-fidelity 3-D simulated environments that correspond to indoor and outdoor scenes with many static and moving obstacles. To demonstrate the sim-to-real capabilities of our method, we use DWA-RL to navigate a real differential drive robot using a simple 2-D lidar in indoor scenes with randomly walking pedestrians.

II. Related Work

A. Collision Avoidance in Dynamic Scenes

Global collision avoidance methods [9], [10], [11], [12] compute an optimal trajectory for the entire route, but they generally work offline which is not suitable for dynamic obstacles. On the other hand, vector-based local approaches such as DWA[6] use limited sensory information and are computationally efficient when avoiding static obstacles.

Several works have extended DWA's capabilities to avoid mobile obstacles by using techniques such as D* graph search [13], look-ahead to detect non-convex obstacles [14], or by extending beyond the local dynamic window to compute possible future paths using a tree [15]. The Curvature-Velocity method [16] is another method similar to DWA which formulates collision avoidance as a constrained optimization problem incorporating goal and vehicle dynamics.

B. DRL-based Collision Avoidance

There have been numerous works on DRL-based collision avoidance in recent years. Methods such as [17] use a deep double-Q network for depth prediction from RGB images for collision avoidance in static environments, whereas more advanced methods [18] use Convolutional Neural Networks to model end-to-end visuomotor navigation capabilities.

An end-to-end obstacle avoidance policy for previously unseen scenarios filled with static obstacles a few pedestrians is demonstrated in [19]. A decentralized, scalable, sensor-level collision avoidance method was proposed in [4], whose performance was improved using a new hybrid architecture between DRL and Proportional-Integral-Derivative (PID) control in [20]. Assuming that pedestrians aid in collision avoidance, a cooperative model between a robot and pedestrians was proposed in [21] for sparse crowds. An extension to this work using LSTMs [22] to capture temporal information enabled it to operate among a larger number of pedestrians.

A few deep learning-based works have also focused on training policies that make the robot behave in a socially acceptable manner [23], [24] and mitigate the freezing robot problem [2], [25]. However, such policies do not provide any guarantees on generating dynamically feasible robot velocities.

Symbols	Definitions
k	Number of linear or angular velocities robot can execute at a time instant
n	Number of past time instants used in observation space formulation
v, ω	Robot's linear and angular velocities
$\dot{v}_l, \dot{\omega}_l$	Robot's linear and angular acceleration limits
$distobs^t(v, \omega)$	Function gives the distance to the nearest obstacle at time *t* from the trajectory generated by velocity vector(v, ω)
$dist(\mathbf{p}_1, \mathbf{p}_2)$	Euclidean distance between $\mathbf{p}_1$ and $\mathbf{p}_2$ (two 2-D vectors).
v_a, ω_a	Robot's current linear and angular velocity
d_t	Distance between the pedestrian and robot
r	Reward for a specific robot action during training
R^{rob}	Robot's radius
$\mathbf{p}^t_{rob}$	Position vector of the robot in the odometry frame at any time instant t
$\mathbf{g}$	Goal location vector
$EndPoint(v_i, \omega_i)$	Function to compute the end point of an trajectory generated by a (v_i, ω_i) vector

TABLE I: List of symbols used in our work and their definitions.

III. Background

In this section we provide an overview of the different concepts and components used in our work.

A. Symbols and Notations

A list of symbols frequently used in this work is shown in Table I. Rarely used symbols are defined where they are used.

B. Dynamic Window Approach

The Dynamic Window Approach (DWA) [6] mainly uses the following steps to search for a collision-free, dynamically feasible [v, ω] velocity vector in a 2-dimensional velocity space known as the *dynamic window*. The non-holonomic constraints for a differential drive robot are inherently addressed since the linear and angular velocities of the robot are controlled separately. The dynamic window is a discrete space with k^2 [v, ω] vectors, which is constructed as follows.

First, a set V of k^2 [v, ω] vectors is formed based on the robot's maximum and minimum linear and angular velocities. Second, the distance to the nearest obstacle from the robot while using a $[v, \omega] \in V$ is computed. A $[v, \omega]$ vector is considered *admissible* only if the robot is able to stop before it collides with the nearest obstacle.

Third, every admissible $[v, \omega]$ vector that is not feasible/achievable within a Δt duration given the robot's linear and angular acceleration limits, are removed to form a set called the *dynamic window* which is formulated as,

$$V_d = \{v, \omega | v \in [v_a - \dot{v}_l \cdot \Delta t, v_a + \dot{v}_l \cdot \Delta t], \\ \omega \in [\omega_a - \dot{\omega}_l \cdot \Delta t, \omega_a + \dot{\omega}_l \cdot \Delta t]\}. \quad (1)$$

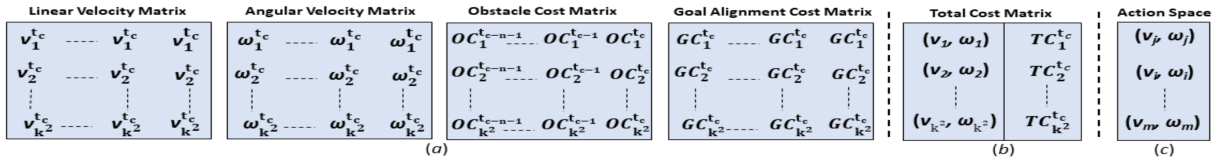

Fig. 2: **(a)** The initial construction of our observation space. The linear and angular velocity matrices ($[v, \omega] \in V_f$) along with their obstacle and goal alignment cost matrices for *n* time instants are constructed. **(b)** The total costs TC for each velocity vector belonging to set V_f. **(c)** The action space which is a reordered set of feasible velocities for the robot at the current time instant t_c.

Once the dynamic window V_d is formed, the $[v, \omega]$, which maximizes the objective function defined in equation 2, is searched for in V_d.

$$G(v,\omega) = \sigma(\alpha.heading(v,\omega) + \beta.distobs^{t_c}(v,\omega) + \gamma.vel(v,\omega)). \quad (2)$$

For a [v, ω] vector used by the robot, *heading()* measures the robot's progress towards the goal (more progress $\implies$ higher value) and the *vel()* function checks that $v \neq 0$. α, β and γ denote weighing constants. Refer to [26] for more details.

C. DRL Policy Training

DRL-based collision avoidance policies are usually trained in simulated environments (similar to Fig.7) using a robot that uses the said policy to perform certain *actions* based on environmental *observations* to earn some *rewards*. The robot's observation consists of information regarding its environment (such as the positions of obstacles), and the set of all observations that the robot's sensors can make is called its *observation space* ($\mathbf{o}^t$). The robot's actions are represented by the velocities that it can execute, and the set of all the robot's velocities is called its *action space* ($\mathbf{a}^t$).

The policy's objective during training is to maximize a *reward function* by performing the actions which are rewarded and avoiding actions that are penalized. This proceeds until the robot continuously achieves the maximum reward for several consequent training iterations. Collision-free velocities can then be computed from the fully trained policy π as,

$$[v, \omega] \sim \pi(\mathbf{a}^t | \mathbf{o}^t). \quad (3)$$

IV. Our Approach

In this section, we explain the construction of our novel observation space, the reward function, and our network architecture.

A. Observation Space Generation

The steps used in the observation space construction are detailed below.

1) Dynamically Feasible Velocity Vectors: Unlike DWA, we do not first generate an admissible velocity set that contains collision-free robot velocities. Instead, we first compute sets of feasible/reachable linear and angular velocities (*lin* = $[v_a - \dot{v} \cdot \Delta t, v_a + \dot{v} \cdot \Delta t]$ and *ang* = $[\omega_a - \dot{\omega} \cdot \Delta t, \omega_a + \dot{\omega} \cdot \Delta t]$) using equation 1. We discretize these sets *lin* and *ang* into k intervals such that the total number of [v, ω] vectors obtained from the intervals is k^2. We then form the set of feasible velocities V_f from these discretized sets as,

$$V_f = \{(v,\omega) | v \in lin_k, \omega \in ang_k\}. \quad (4)$$

The velocity vectors in V_f do not account for the locations of the obstacles in the current time instant t_c or the past *n*-1 time instants. Therefore, some velocities in V_f could lead to collisions. The *k* linear and angular velocities in V_f are appended *n*-1 times as column vectors in two matrices each of size ($k^2 \times n$) and the generated linear and angular velocity matrices are shown in the Fig. 2(a).

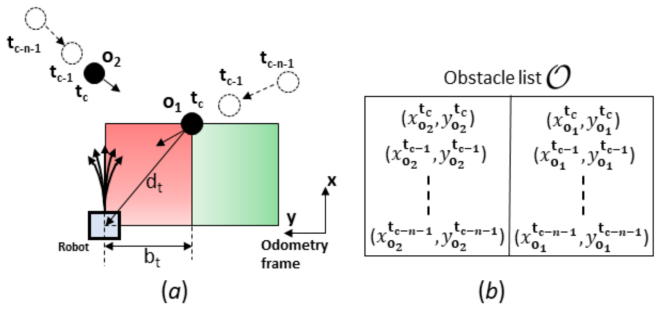

Fig. 3: Obstacle set construction. **(a)** The change in the location of the obstacle in the past n-1 time instances with respect to the location of the robot in the current time instance. The red region ($\mathcal{R}$) and the green region ($\mathcal{G}$) denote the regions of high risk and low risk of collisions respectively. They depend on the relative positions and motions of the robot and the obstacle. **(b)** The list of the obstacle sets obtained at various time instances each column corresponds to the location of the obstacles at particular time instance

2) Obstacle sets: We use a 2-D lidar scan to sense the location of the obstacles around the robot. For each time instant, the obstacle locations are obtained relative to a fixed odometry coordinate frame and stored in a set. The odometry frame is attached to the ground at the location from where the robot started. In Fig. 3(a), the locations of two obstacles in the current as well as in the past *n*-1 time steps are shown. We add the set of obstacle locations in a list $\mathcal{O}$ of length *n* (see Fig. 3(b)), where each row shows the set of obstacle locations for a specific time instant. We use $\mathcal{O}$ to incorporate information regarding the motion of various obstacles in the environment.

3) Obstacle cost calculation: Next, we calculate the *obstacle cost* for every velocity vector in V_f using the $distobs^t()$ function. Each vector in V_f is forward simulated for a time duration Δt to check if it leads to a collision, given the obstacle positions in $\mathcal{O}$. The costs are calculated as,

$$OC_i^{t_j} = \begin{cases} c_{col} & \text{if } distobs^{t_j}(v_i,\omega_i) < R^{rob}, \\ \dfrac{1}{distobs^{t_j}(v_i,\omega_i)} & \text{otherwise.} \end{cases} \quad (5)$$

Where, c_{col}= 40. The Fig.2 (a) shows the obtained ($k^2 \times n$) obstacle cost matrix.

4) Goal alignment cost calculation: Each $[v, \omega]$ in V_f is forward simulated for a time Δt and the distance from the endpoint of the trajectory to the robot's goal is measured (equation 6). The velocity vectors that reduce the distance between the robot and the goal location are given a low cost.

$$GC_i^{t_c} = dist(EndPoint(v_i,\omega_i), \mathbf{g}) \ * \ c_{ga} \quad (6)$$

The goal alignment cost is independent of the location of the obstacles around the robot, therefore the same cost for each pair is appended *n* times to obtain a goal alignment cost matrix of shape $(k^2 \times n)$ as seen in Fig. 2(a), and in the equation 7.

$$GC_i^{t_c} = GC_i^{t_{c-1}} = ... = GC_i^{t_{c-n-1}} \tag{7}$$

Where, c_{ga}= 2.5.

5) Total cost calculation: The total cost for the robot using a vector $[v_i, \omega_i]$ for the current time instant t_c is calculated as,

$$TC_i^{t_c} = OC_i^{t_c} + GC_i^{t_c} \tag{8}$$

and is shown in Fig.2(b).

6) Sorting the Velocity Vectors: The linear, angular, obstacle cost and goal alignment cost matrices obtained in Section IV-A are now reordered to better represent which velocities in V_f have the lowest costs given the obstacle positions for the past *n* time instants. The velocity vectors are sorted in ascending order according to the total cost of the velocity vectors at the current time instant. The elements in the velocity and cost matrices are then reordered in same order.

7) Observation Space and Action Space: Finally, our observation space is constructed using the reordered linear, angular matrices along with the obstacle and goal alignment cost matrices and stacking them to get a matrix of size $(k^2 \times n \times 4)$. Our action space is the reordered set of feasible velocities for the robot at the current time instant (see Fig.2c). The observation space is then passed to the policy network (see Fig.4).

B. DRL Navigation Framework

In this section, we detail the other components of our DRL policy's training, and run-time architecture.

1) Reward Function Shaping: Rewards for the basic navigation task of reaching the goal and avoiding collisions with obstacles are provided with high positive or negative values respectively. In order to make the training faster, the difference between distance from the goal in the previous and the current time instant is utilized in the reward function. This incentivizes the policy to move the robot closer to the goal each time step, or otherwise be penalized as,

$$(r_g)^t = \begin{cases} r_{goal} & \text{if } dist(\mathbf{p}_{rob}^t, \mathbf{g}) < 0.3\text{m}, \\ -2.5(dist(dist(\mathbf{p}_{rob}^t, \mathbf{g})) - \mathbf{p}_{rob}^{t-1}, \mathbf{g}) & \text{otherwise.} \end{cases} \tag{9}$$

We define the penalty for collision as,

$$(r_c)^t = \begin{cases} r_{collision} & \text{if } dist(\mathbf{p}_{rob}^t, \mathbf{p}_{obs}^t) < 0.5\text{m}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

When the distance between a dynamic obstacle (located at $\mathbf{p}_{obs}^t = [x_{obs}^t, y_{obs}^t]$) and the robot (located at $\mathbf{p}_{rob}^t = [x_{rob}^t, y_{rob}^t]$) is less than a certain threshold, the robot receives the steering reward/penalty (equation 12). The parameters d_t and b_t which influence this reward are depicted in Fig.3a, and defined as follows,

$$d_t = dist(\mathbf{p}_{rob}^t, \mathbf{p}_{obs}^t) \quad | \quad b_t = y_{rob}^t - y_{obs}^t. \tag{11}$$

$$(r_{steer})^t = \begin{cases} -|b_t| * r_{spatial} - \dfrac{r_{proximity}}{d_t} & \text{if } p_{rob}^t \in \mathcal{R} \\ +|b_t| * r_{spatial} & \text{if } p_{rob}^t \in \mathcal{G}. \end{cases} \tag{12}$$

From equation 12, it can be seen that the robot is rewarded positively when it is in the green region $\mathcal{G}$ (behind the obstacle) shown in Fig.3a and penalized when it is in the red region $\mathcal{R}$ (along the obstacle's heading direction). This reinforces *spatially aware* velocities when handling dynamic obstacles i.e., velocities which move the robot away from an obstacle's heading direction, thereby reducing the risk of collision.

Proposition IV.1. *Region $\mathcal{R}$ has a high risk of collision.*

Proof. Please refer to [26] for a proof which shows that the distance between the obstacle and the robot in region $\mathcal{R}$ is always a decreasing function. ■

In the case of an obstacle moving head-on the total steering reward is zero. In the presence of multiple dynamic obstacles around the robot, the union of the red zones is to be constructed for the total negative rewards.

This is also supplemented by providing negative rewards *inversely proportional to the distance* from all the obstacles in the sensor range of the robot. This reduces the danger of collision as negative reward is accumulated as the robot approaches the obstacle.

$$(r_{dangerOfCollison})^t = -\frac{r_{dCollision}}{d_t} \tag{13}$$

We set $r_{goal} = 2000, r_{collison}$ =-2000, $r_{proximity} = 10, r_{spatial} = 25, r_{dCollison} = 30$.

2) Network Architecture: The policy network architecture that we use is shown in Fig.5. Five 2-D convolutional layers, followed by 3 fully-connected layers are used for processing the observation space. ReLU activation is applied between the hidden layers. This architecture is much simpler and requires fewer layers for handling our observation space.

3) Policy Training: We simulate multiple Turtlebot2 robots each with an attached lidar to train the models. The Turtlebots are deployed in different scenarios in the same simulation environment (refer to [26]), to ensure that the model does not overfit to any one scenario. Our policy finishes training in less than 20 hours, which is significantly less than the 6 days it takes to train methods such as [1], [2], which use similar training environments.

4) Run-time Architecture: The output of a fully trained policy network is the index *i* that corresponds to a velocity pair in the action space. The $[v, \omega]$ vector at the i^{th} location in the action space is then used by the robot for navigation at the current time instant t_c.

Proposition IV.2. *The velocity chosen by our fully trained policy will always obey the dynamics constraints of the robot.*

Proof. The proof follows trivially from the fact that our action space is a subset of our observation space (Fig.2c), which in turn is constructed using the dynamic feasibility equations of DWA. Thus, our policy preserves the dynamic feasibility guarantees of DWA. ■

Our full run-time architecture is shown in Fig.4.

V. Results, Comparisons and Evaluations

A. Implementation

We use ROS Melodic and Gazebo 9 to create the simulation environments for training and evaluating on a workstation with an Intel Xeon 3.6GHz processor and an Nvidia GeForce RTX 2080TiGPU. We implement the policy

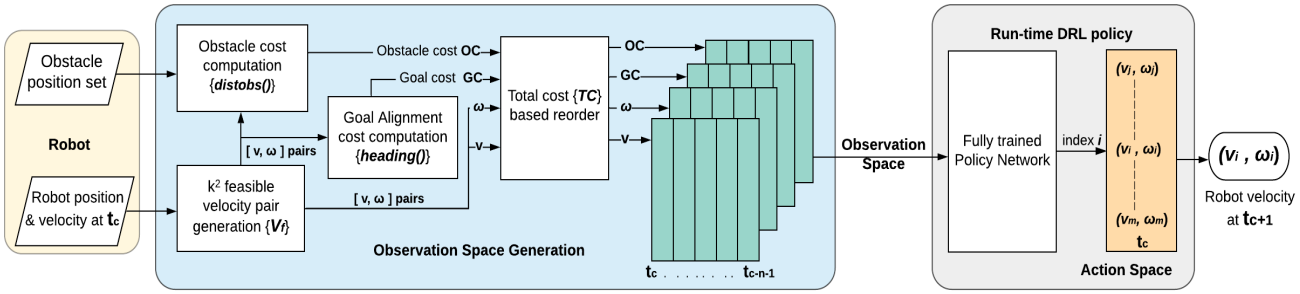

Fig. 4: Our method's run-time architecture. The observations such as obstacle positions measured by the robot's sensors (lidar in our case) and the robot's position and velocity at time t_c, along with the obstacle and goal-alignment costs are reordered (Section IV-A.6) to generate a $(k^2 \times n \times 4)$ dimensional observation space (Section IV-A.7) shown in green corresponding to time instant t_c. The fully trained DRL policy network (shown in Fig. 5) uses the observation space to compute the index of the output velocity in the action space.

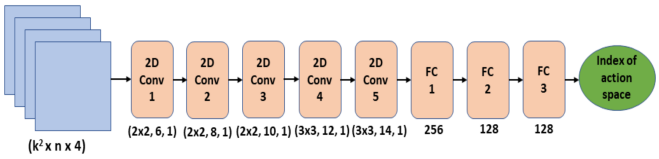

Fig. 5: Architecture of the policy network used for training. The input observation space is marked in blue, the network layers are marked in orange. The initial 5 layers are the convolutional layers and the remaining 3 layers are the fully connected layers. The output of the network is marked in green.

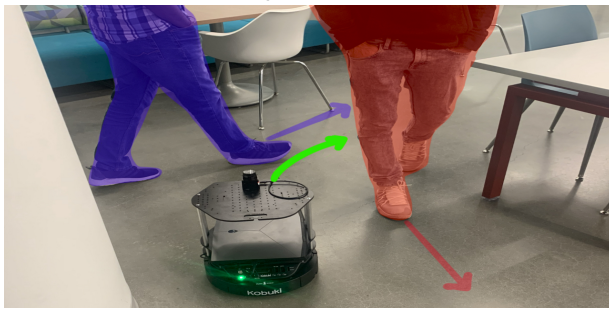

Fig. 6: DWA-RL tested on a Turtlebot in a dense scenario. This shows that DWA-RL can be easily transferred from simulations to different real-world robot platforms. The trajectories of the obstacles is shown in blue and red. The robot's trajectory is shown in green.

network using TensorFlow and use the PPO2 implementation provided by stable baselines to train our policy. In the DWA implementation the cycle runs at 20 Hz, other DWA related parameters can be found in the code [1].

To test the policy's sim-to-real transfer and generalization capabilities, we use it to navigate a Turtlebot 2 and a Jackal robot in challenging indoor scenes with randomly moving pedestrians (see attached video). DWA-RL does not require accurate sensing of the obstacles' positions in real-world scenes.

B. Testing Scenarios

We evaluate DWA-RL and compare with prior methods in the following scenarios (see Fig.7).

Zigzag-Static: This scenario contains several sharp turns with a number of static obstacles to resemble a cluttered indoor environment.

[1]https://github.com/NithishkumarS/DWA-RL

Metrics	Method	Zigzag Static	Occluded Ped	Sparse Dynamic	Dense Dynamic
Success Rate	DWA-RL	1.0	1.0	0.54	0.4
	DRL	1.0	0.76	0.33	0.31
	DWA	1.0	0.9	0.42	0.3
Avg Traj. Length (m)	DWA-RL	28.85	27.26	11.95	12.23
	DRL	26.89	25.63	12.10	11.57
	DWA	26.62	25.07	11.99	12.81
Avg Velocity (m/s)	DWA-RL	0.38	0.46	0.42	0.37
	DRL	0.42	0.51	0.37	0.50
	DWA	0.44	0.57	0.6	0.64

TABLE II: Relative performance of DWA-RL versus DWA [6] and Long et. al's method [4].

Occluded-Ped: This scenario contains several sharp turns and two pedestrians who could be occluded by the walls.

Sparse-Dynamic: This scenario contains 4 walking pedestrians in a corridor-like setting moving at 45° or 90° angles with the line connecting the robot's start and goal locations.

Dense-Dynamic This scenario contains 17 pedestrians in an area of $13 \times 8m^2$ who could be static or moving and resembles dense dynamic outdoor environments.

C. Evaluation Metrics

We compare our approach with: (i) Dynamic Window Approach [6] (ii) Long et al.'s method [4]. We also provide ablation studies to demonstrate the effects of our various design choices while formulating the observation space and reward function. We use the following metrics to compare the methods and the ablation study models.

- **Success Rate** - The number of times the robot reached its goal without colliding with obstacles over 50 trials. The obstacles' initial positions are randomly assigned in each trial.
- **Average Trajectory Length** - The total distance traversed by the robot, until the goal is reached, averaged over the number of successful trials.
- **Average Velocity** - It is the trajectory length over the time taken to reach the goal in a successful trial.

D. Analysis and Comparison

The results of our comparisons and ablation studies are shown in tables II, III and IV.

From table II, we observe that in terms of success rate all approaches perform well in the Zigzag-Static scenario. However, in the environments with mobile obstacles, DWA-RL collides significantly less number of times. This is because DWA-RL considers obstacles' motion over time (in the observation space) and computes velocities that avoid the

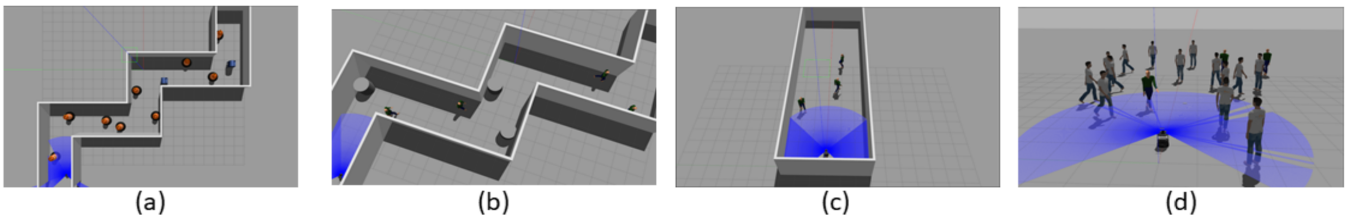

Fig. 7: Different testing scenarios used to evaluate the collision avoidance approaches **(a)**: Zigzag-Static Scenario; **(b)**: Occluded-Ped scenario where the dynamic obstacles are suddenly introduced; **(c)**: Sparse-Dynamic scenario; **(d)** Dense-Dynamic environment contains the combination of static and dynamic obstacles.

Metrics	Method	Zigzag Static	Occluded Ped	Sparse Dynamic	Dense Dynamic
Success Rate	With PR	1.0	1.0	0.54	0.42
	Without PR	0.86	0.64	0.44	0.4
Avg Traj. Length (m)	With PR	28.85	27.26	11.95	12.46
	Without PR	28.12	27.86	11.6	12.78
Avg Velocity (m/s)	With PR	0.37	0.47	0.42	0.38
	Without PR	0.34	0.41	0.41	0.34

TABLE III: Ablation study showing relative performance of the models trained with positive reinforcement (PR) and without positive reinforcement.

Metrics	Method	Zigzag Static	Occluded Ped	Sparse Dynamic	Dense Dynamic
Success Rate	3-matrix	0.82	0.94	0.36	0.36
	4-matrix	1.0	1.0	0.54	0.42
Avg Traj. Length (m)	3-matrix	27.93	27.04	11.56	12.34
	4-matrix	28.85	27.26	11.95	12.46
Avg Velocity (m/s)	3-matrix	0.38	0.44	0.46	0.37
	4-matrix	0.37	0.47	0.42	0.38

TABLE IV: Ablation study showing relative performance of the models trained with 3-matrix and 4-matrix observation space formulation.

region in front of the obstacle (reinforced in reward function). DWA and Long et al.'s method try to avoid the obstacles from in-front and collide, especially in the Occluded-Ped scenario, where obstacles are introduced suddenly. Even with limited temporal information, DWA-RL always guides the robot in the direction opposite to the obstacle's motion, thereby reducing the chances of a collision. DWA-RL achieves this while maintaining a comparable average trajectory lengths and velocities for the robot.

Ablation Study for the Positive Reinforcement: We compare two policies trained with and without the positive reinforcement (PR) ($|b_t| * r_{spatial}$) term in equation 12 in different test environments. From Table III, we observe that the policy trained with PR outperforms the model trained without it in all the test environments. The policy trained without PR mostly tries to avoid an obstacle by navigating in-front of it, predominantly resulting in collisions.

Ablation Study for the Observation Space: Our observation space uses four matrices stacked together as show in Fig. 4 which include velocities and the obstacle and goal-alignment costs. We compare this formulation with one which uses three matrices; the linear and angular velocity matrices and a total cost matrix stacked together. The total cost matrix is the sum of the obstacle and goal-alignment cost matrices. The results for both the policies are shown in Table IV. We observe that the 4-matrix formulation outperforms the 3-matrix formulation in all the scenarios. This is because, the information about environmental obstacles is better imparted into the policy when the obstacle cost is provided separately.

Dynamics Constraints Violation The Fig. 8 shows the graph of angular velocities generated by the Long et. al's method [4] in the Dense Dynamic environment. We observe

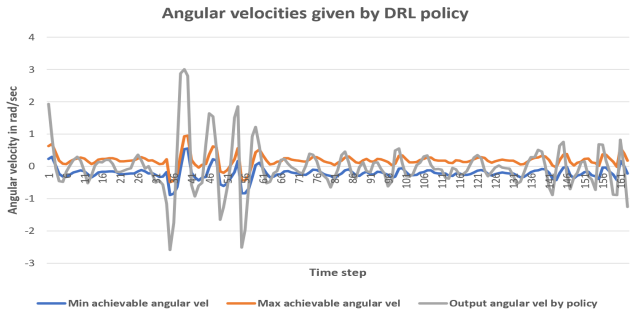

Fig. 8: Graph showing the change in the angular velocity generated by the DRL approach along with the maximum and the minimum achievable velocity at that time instant. For this experiment, we use Turtlebot 2 with max angular velocity, min angular velocity and max angular acceleration limit of 3.14 rad/s, -3.14 rad/s and 2 rad/s^2 respectively.

that the output angular velocities lie outside the maximum and minimum attainable angular velocities of the robot 61% of the times, leading to oscillatory/jerky motion. DWA-RL on the other hand, produces velocities that always lie within the attainable velocity range. This results in considerably smoother robot trajectories. We point the reader to [26] for a more results regarding DWA-RL.

VI. Conclusions, Limitations and Future Work

We present a novel formulation of a Deep Reinforcement Learning policy that generates dynamically feasible and spatially aware smooth velocities. Our method addresses the issues associated with learning-based approaches (dynamic infeasible velocities) and the classical Dynamic Window Approach (sub-optimal mobile obstacle avoidance). We validate our approach in simulation and on real-world robots, and compare it with the other collision avoidance techniques in terms of collision rate, average trajectory length and velocity, and dynamics constraints violations.

Our work has a few limitations which we wish to address in the future. For instance, the model needs at least few observations to compute a velocity that is spatially aware. If the obstacles are suddenly introduced in the field of view of the robot, the robot might freeze. Efficiency of this approach with an integrated global planner is yet to be studied. Also, the current model uses Convolutional Neural Network as layers in the policy network, but the use of LSTM [27] could improve the processing of the temporal data from the observation space.

References

[1] J. Liang, U. Patel, A. Sathyamoorthy, and D. Manocha, "Crowd-steer: Realtime smooth and collision-free robot navigation in densely

crowded scenarios trained using high-fidelity simulation," 07 2020, pp. 4193–4200.
[2] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," 2020.
[3] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," 2019.
[4] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6252–6259.
[5] B. d'Andrea Novel, G. Bastin, and G. Campion, "Modelling and control of non-holonomic wheeled mobile robots," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1130–1131.
[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.
[7] C. R. Shelton, "Balancing multiple sources of reward in reinforcement learning," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. NIPS'00. Cambridge, MA, USA: MIT Press, 2000, p. 1038–1044.
[8] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8620–8626.
[9] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[10] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136
[11] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.
[12] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré, "Parameter estimation and comparative evaluation of crowd simulations," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 303–312.
[13] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," 05 2007, pp. 1986 – 1991.
[14] C.-C. Chou, F.-L. Lian, and C.-C. Wang, "Characterizing indoor environment for robot navigation using velocity space approach with region analysis and look-ahead verification," *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, pp. 442 – 451, 03 2011.
[15] E. J. Molinos, Ángel Llamazares, and M. Ocaña, "Dynamic window based approaches for avoiding obstacles in moving," *Robotics and Autonomous Systems*, vol. 118, pp. 112 – 130, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889018309746
[16] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, April 1996, pp. 3375–3382 vol.4.
[17] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *CoRR*, vol. abs/1706.09829, 2017. [Online]. Available: http://arxiv.org/abs/1706.09829
[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *CoRR*, vol. abs/1504.00702, 2015. [Online]. Available: http://arxiv.org/abs/1504.00702
[19] Y. Kim, J. Jang, and S. Yun, "End-to-end deep learning for autonomous navigation of mobile robot," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2018, pp. 1–6.
[20] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," *CoRR*, vol. abs/1808.03841, 2018. [Online]. Available: http://arxiv.org/abs/1808.03841
[21] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *ICRA*. IEEE, 2017, pp. 285–292.
[22] M. Everett, Y. F. Chen, and 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)*, Oct 2018, pp. 3052–3059.
[23] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *ICRA*, May 2018, pp. 1111–1117.
[24] Y. F. Chen, M. Everett, M. Liu, and J. How, "Socially aware motion planning with deep reinforcement learning," 09 2017, pp. 1343–1350.
[25] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.
[26] U. Patel, N. Kumar, A. Jagan Sathyamoorthy, and D. Manocha, "Dynamically Feasible Deep Reinforcement Learning Policy for Robot Navigation in Dense Mobile Crowds," *arXiv e-prints*, p. arXiv:2010.14838, Oct. 2020.
[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735