Mitigating Imminent Collision for Multi-robot Navigation: A TTC-force Reward Shaping Approach

Jinlin Chen
The Hong Kong Polytechnic University
Hong Kong, China
csjlchen@comp.polyu.edu.hk

Jiannong Cao, Fellow, IEEE
The Hong Kong Polytechnic University
Hong Kong, China
csjcao@comp.polyu.edu.hk

Zhiqin Cheng†
The Hong Kong Polytechnic University
Hong Kong, China
aaronworry128@gmail.com

Wei Li†
Jiangnan University
Jiang Su, China
cs_weili@jiangnan.edu.cn

ABSTRACT

We study the distributed multi-robot navigation problem, which refers to a group of mobile robots avoiding collision with each other while navigating from their start positions to the goal positions. Existing works still suffer from two limitations: 1) accurately quantify the risk of collisions for heterogeneous robots and 2) effectively capture the state representation under dynamic environments. These limitations make the heterogeneous robots prone to collisions in high-density and dynamic environments. This work proposes a new time-to-collision force (TTC-force) reward shaping approach, termed *Tfresh*, incorporating reinforcement learning to learn a policy that adaptively chooses the optimal actions to mitigate the imminent collision. Specifically, we use TTC-force to quantify the risk of each robot exerted by its neighbors and shape the reward signal with TTC-force in applying the reinforcement learning scheme. Meanwhile, we design the spatial attention mechanism involving the dynamic adjacent matrix to capture the state representation effectively. We evaluate the learned policy in numerous simulated scenarios in which groups of mobile robots perform navigation tasks. The experimental results demonstrate that our approach outperforms the state-of-the-art methods regarding success rate, travel distance, and travel time.

KEYWORDS

Collision Avoidance; Multi-robot Navigation; Reward Shaping; Reinforcement Learning

ACM Reference Format:
Jinlin Chen, Jiannong Cao, Fellow, IEEE, Zhiqin Cheng†, and Wei Li†. 2023. Mitigating Imminent Collision for Multi-robot Navigation: A TTC-force Reward Shaping Approach. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

With the rapid growth in logistics automation and robotic technologies, many companies have designed autonomous delivery systems in warehouses [24] to reduce labor costs, and schools have designed mobile food delivery robots on campus sidewalks to reduce human-to-human contact during the COVID-19 pandemic [40]. Recently, Amazon has launched a fully autonomous warehouse robot, deploying more than 520,000 robot-driven units in its fulfillment and sorting centers [15, 39]. The robots are built to be automatically directed to perform their work and move around, meaning that they can operate autonomously by using perception and navigation techniques.

Although large-scale mobile robotic systems have a high potential to revolutionize warehouse and last-mile delivery applications, they pose a formidable challenge to the control of heterogeneous robots when performing tasks autonomously. The situation becomes challenging when it comes to scenarios where the density of robots is high, and the robots need to cooperate and interact with each other in dynamic environments. This is because the robots should move safely by avoiding imminent collisions when they come into contact with their neighbors in navigating from their start positions to the goal positions.

A high-efficient and collision-free multi-robot navigation system can facilitate task completion performance to accelerate productivity. Three mainstream approaches are prevalent in the field of multi-robot navigation, including velocity obstacle-based, force-based, and learning-based approaches. Specifically, velocity obstacle-based approaches and their variants [1, 8, 26, 37] first compute the velocity obstacle (VO) region and then select the collision-free velocity outside of the VO regions. The conservative feasible velocities selection often causes the robot to discard too many admissible velocities, leading to robots getting stuck. Force-based approaches [18, 27] are inspired by artificial potential field [3], which use the relative position and velocity to compute the repulsive force to avoid collision and the attractive force to reach the goal position. However, force-based approaches cannot work competently when the obstacle is near the goal position. This is because the repulsive and attractive forces cancel out, making the robot stuck in place.

As an alternative, reinforcement learning-based approaches [13, 25, 30, 33, 34] have been widely used to train the navigation policy by mapping the observed state into continuous control commands to steer the robots to achieving the goal position. Although reinforcement learning-based methods have made significant progress by benefiting from converting rich trial-and-error experience into knowledge [35], which enables robots to make more aggressive

†: Corresponding author.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

movement decisions, the existing models still suffer from two limitations in designing reinforcement learning-based multi-robot navigation policy. First, how to accurately quantify the risk of collision for the heterogeneous robots to give an immediate reward signal. Second, how to effectively extract the state representation from the observed state in dynamic environments.

In this work, we propose a new collision avoidance approach, termed *Tfresh*, to overcome the limitations mentioned above in mitigating the imminent collision of multi-robot navigation. First, we quantify the risk of collision with TTC-force inspired by the TTC concept [19]. The TTC-force is obtained by following a power-law relationship concerning the TTC. If the TTC force between the two robots is approaching zero, the two robots will not collide in the future. We use TTC-force to shape the reward signal in each timestep, guiding the robot to learn a high-performance navigation policy because the dense reward signal can provide immediate feedback [21]. Second, we use the state encoder to encode the robots' state and learn the state embedding given the observed sequential states of a time horizon. We then feed the state embedding into a spatial attention mechanism to capture the effective state representations from the state embedding. Due to its flexibility, a dynamic adjacent matrix obtained from the relative distance between robots is involved in the spatial attention mechanism to model the spatial relations between the robots and their neighbors. Subsequently, the spatial attention mechanism enables robots to pay more attention to the state of the most-related neighboring robots. We implement and deploy *Tfresh* with a group of heterogeneous mobile robots in terms of different size and motion constraints by performing numerous navigation tasks. The experimental results regarding navigation efficiency show that our method can provide a satisfactory collision avoidance solution for heterogeneous mobile robots navigating in a limited workspace. The core contributions of this work are:

- We quantify the risk of collisions for each robot exerted by their surrounding robots with TTC-force, which effectively measures the severity of potential collisions for heterogeneous mobile robots.
- We develop a spatial attention mechanism to enable robots to pay attention to the relatively important interactions of surrounding robots, which helps to capture the most significant state representations in navigation policy learning.
- We design a novel TTC-force reward shaping approach, which provides a dense reward signal to encourage the robot to learn a high-performance navigation policy with reinforcement learning.

The rest of the paper is organized as follows. We first examine the related works in detail and their differences from our work in Section 2. We formally introduce the preliminary background and the problem formulation in Section 3. We then present the proposed multi-robot navigation approaches in Section 4. This is followed by the experiment design and results in Section 5. In Section 6, we present the conclusion and future work.

2 RELATED WORK

In this section, we divide the related work about multi-robot navigation into three categories, i.e., velocity obstacle-based, force-based, and learning-based approaches, from the implementation point of view.

Velocity Obstacle-based Approaches - The methods in this category directly plan the new velocity for each robot. Many existing collision avoidance methods are based on the concept of velocity obstacle (VO) proposed by Paolo Fiorini and Zvi Shiller [11]. The VO stands for the set of robot velocities that will cause a potential collision with a given obstacle moving at a given velocity. The velocity obstacle-based methods can generate avoidance maneuvers by selecting the velocities outside the velocity obstacle. In light of this, robots can safely navigate by choosing new velocities outside any VOs caused by their neighbors.

Many extensions based on VO have been proposed to address different challenges and consider different kinematic-constrained robots. Van den Berg et al. [5] have proposed the ORCA framework, which provides an efficient way of computing the collision-free velocity outside the union of all VOs imposed by neighbors. A feasible collision-free solution is obtained by conservatively approximating each VO as a half-plane and then using linear programming based on all half-planes. To extend ORCA to non-holonomic robots, NH-ORCA [1] has been proposed to deal with the kinematics of non-holonomic robots, which controls the robots staying within a maximum tracking error ϵ of an ideal holonomic trajectory. To consider uncertainty in the motion and sensing of the robots, Jamie Snape et al. [36] have proposed HRVO, in which they use a Kalman filter to accurately estimate the uncertainties in terms of radius, position, and velocity. All the aforementioned algorithms have the limitation that they are only guaranteed to provide safe motion for robots with the linear equation of motion. Thus, Dman Bareiss and Jur Van den Berg [4] proposed a generalized reciprocal collision approach that unifies a variety of dynamic systems ranging from single integrator to car-like, differential-drive, and arbitrary, linear equations of motion into a single, generalized representation using control obstacle. This allows general mobile robots to select new control inputs while avoiding collision with others.

Although the VO-based method and its ORCA variants have gained applicability in robotic applications. The problem of conservatively selecting feasible velocities causes the robot to discard too many admissible velocities because ORCA approximates the VO cone with a line [12]. In dense multi-robot navigation scenarios, the robots may stop moving because of the infeasible approaches, leading to unfinished tasks.

Force-based Approaches - The approaches in this category model the behavior of each robot as a collection of forces, which is inspired by the potential energy in Physics. In the work of Helbing et al., [16], the interaction forces have been proposed to model the interaction between humans, and the force between humans and their surrounding obstacles.

The traditional force-based models resolve collisions by using artificial potential fields [3], in which the repulsive potential is used to avoid obstacles and the attractive potential allows the robots to move to the goal positions. But such a model only reacts when the robots get too close to each other and overlook the velocities of neighbors. To address this issue, Karamouzas et al. [19] have proposed a predictive force-based model. They found that the estimated time-to-collision is the function of the interaction force between two pedestrians, which follows an inverse power-law relationship. The resulting time-to-collision model allows robots to emulate better how humans resolve collisions in practice, leading to more efficient behavior than VO-based methods. Later, Karamouzas et al. [12] proposed a UTTC model to solve the uncertainty issues taken by the robot sensor. The uncertainties regarding radius, position, and velocity are accounted for in the future trajectories of interacting robots. To extend to the team with a large number of agents, Samaneh Hosseini et al. [18] have proposed a force-based motion planning based on the flocking algorithm, which allows each robot to calculate the repulsive force and navigation force to avoid collisions.

Although the force-based approaches are real-time, distributed, and more flexible than VO-based approaches, they still have limitations in providing a time-optimal navigation path when getting into local optimum.

Learning-based Approaches - In recent years, learning-based multi-robot navigation approaches have become popular. Most methods aim to learn a decision-making policy with reinforcement learning by inputting local or global information and outputting the actions for robots. Pinxin Long et al. [25] proposed a distributed sensor-level collision avoidance policy trained with multi-robot proximal policy optimization, which maps raw sensor measurements to an agent's steering commands in terms of movement velocity. Afterward, this method is extended with hybrid control architecture [10] which switches between different policies based on the obstacle density in the environment. Compared with sensor-level methods, agent-level reinforcement learning methods [6, 9] use environmental models to achieve high computational efficiency and flexibility for sensor modalities. To tackle the issue of changing number of neighboring robots in distributed environmental settings, GA3C-CADRL [9] uses RNNs to tackle a variable number of moving obstacles. Similarly, the socially aware RL [6] has been proposed to infer the importance with the self-attention mechanism because of the surrounding dynamic obstacles. Qingbiao Li et al.[22, 23] proposed a message-aware graph attention network to determine the relative importance of features in the messages received from neighboring robots. Recently, RL-RVO [13] combines the concept of reciprocal velocity obstacle and the scheme of deep reinforcement learning (DRL) to solve the reciprocal collision avoidance problem under limited information. The area of ORCA is regarded as the reward function to guide the actor-critic based reinforcement learning to learn a multi-robot navigation policy.

To summarize, the approaches mentioned above still have their limitations, which lie in accurately quantifying the risk of collisions and effectively capturing the state representation of robots. To overcome these limitations, we proposed a new TTC-force reward shaping approach to quantify the risk and design a spatial attention mechanism that allows the robots to pay attention to the most-related neighbors. It enables reinforcement learning to learn a high-performance navigation policy that can make collision-free movements and steer the robots to their goal positions faster.

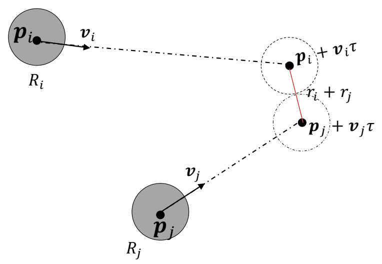


Figure 1: Illustration of the calculation of time to collision τ between robots R_i and R_j.

3 PRELIMINARIES AND PROBLEM FORMULATION

In this section, we first introduce the model of time-to-collision, reward shaping of reinforcement learning, and then define the multi-robot navigation problem formally.

3.1 Time to Collision

To avoid collision during the navigation, the robot should be able to predict whether and when it will collide with its nearby robots so that it can adapt its velocity accordingly. We adopt the concept of time-to-collision to reason about the potential collisions, which refers to the time required for two robots to collide if they continue moving with their present velocity and direction [14, 38]. Considering two robots R_i and R_j as shown in Fig. 1, the position, velocity, and radius of them are denoted as $\boldsymbol{p} = [p_x, p_y]$, $\boldsymbol{v} = [v_x, v_y]$, and r. A collision between two robots occurs at the timeslot $\tau \geq 0$ if the corresponding discs intersect. In other words, a collision occurs when the distance between two robots is equal to the sum of their radius,

$$||(\boldsymbol{p}_j + \boldsymbol{v}_j\tau) - (\boldsymbol{p}_i + \boldsymbol{v}_i\tau)|| = r_i + r_j \tag{1}$$

where $||\cdot||$ represents an Euclidean norm. After taking the square of both sides, we obtain the following quadratic equation with respect to τ,

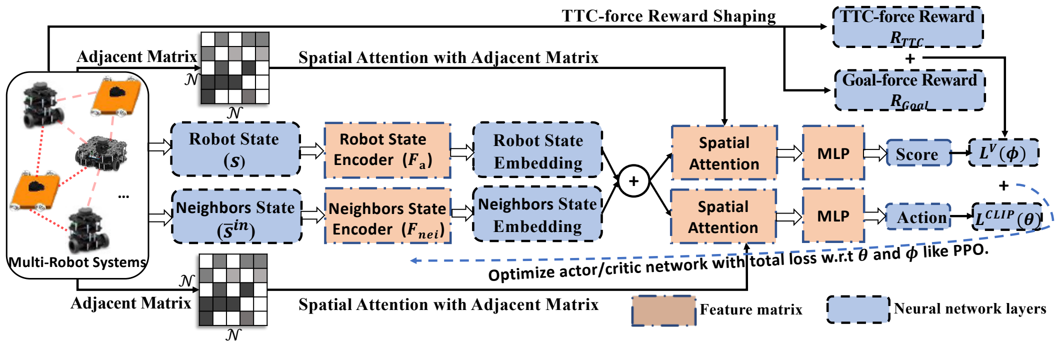
$$\underbrace{(\Delta\boldsymbol{v}\cdot\Delta\boldsymbol{v})}_{a}\tau^2 + 2\cdot\underbrace{(\Delta\boldsymbol{p}\cdot\Delta\boldsymbol{v})}_{b}\tau + \underbrace{\Delta\boldsymbol{p}\cdot\Delta\boldsymbol{p} - r_{ij}^2}_{c} = 0 \tag{2}$$

where the relative position $\Delta\boldsymbol{p} = \boldsymbol{p}_i - \boldsymbol{p}_j$, the relative velocity $\Delta\boldsymbol{v} = \boldsymbol{v}_i - \boldsymbol{v}_j$, and the combined radius $r_{ij} = r_i + r_j$. Moreover, a, b, c denotes the coefficient terms of the quadratic equation with τ as the variable. After simplification and solving the quadratic equation, we can obtain the solution of τ with

$$\tau = \frac{c}{-b + \sqrt{\mathcal{D}}}, \tag{3}$$

$$\text{where} \quad \begin{cases} a = \Delta\boldsymbol{v}\cdot\Delta\boldsymbol{v}, \\ b = \Delta\boldsymbol{p}\cdot\Delta\boldsymbol{v}, \\ c = \Delta\boldsymbol{p}\cdot\Delta\boldsymbol{p} - r_{ij}^2, \\ \mathcal{D} = b^2 - a\cdot c \end{cases} \tag{4}$$

It is evident that there are no collisions if τ is negative or if τ is undefined when $\mathcal{D} \leq 0$.



TTC-force Reward Shaping
TTC-force Reward R_{TTC}
Goal-force Reward R_{Goal}
Adjacent Matrix
Spatial Attention with Adjacent Matrix
Robot State (s)
Robot State Encoder (F_a)
Robot State Embedding
Neighbors State ($\bar{s}^{in}$)
Neighbors State Encoder (F_{nei})
Neighbors State Embedding
Spatial Attention
MLP
Score
$L^V(\phi)$
Action
$L^{CLIP}(\theta)$
Multi-Robot Systems
Optimize actor/critic network with total loss w.r.t θ and ϕ like PPO.
Feature matrix
Neural network layers

Figure 2: Framework overview of *Tfresh*. The proposed framework consists of a state feature extraction module, an actor-network, and a critic network. First, the state feature module learns the state embedding of the robot's state and its neighboring robots' state. Then, the concatenated (⊕) state embedding and adjacent matrix ($\mathcal{N}$ by $\mathcal{N}$) are fed into the spatial attention. Finally, the output state representation is fed into the MLP to output the action and critic score for the action and critic network, respectively. Besides, the loss of policy $\mathcal{L}^{CLIP}(\theta)$ and loss of critic $\mathcal{L}^{\mathcal{V}}(\phi)$ are used to optimize the neural network parameters.

3.2 Reward Shaping

Reward shaping is a technique that provides immediate feedback based on prior knowledge, which can effectively boost RL agents by converting the domain knowledge into additional rewards [21]. This advantage makes the technique widely used to guide reinforcement learning processes.

Specifically, reward shaping is equivalent to learning in a more complementary environment. The new environment is the evolution of the native Markov decision process (MDP) to a shaped MDP with augmented rewards. Using the notation for a native MDP, $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$, where $\mathcal{S}$ is the finite state; $\mathcal{A}$ is a set of actions; $\mathcal{P}$ is the next-state transition probabilities; $\mathcal{R}$ specifies the reward distributions, and γ is the discounted factor. $\mathcal{M}$ is converted to the shaped MDP, $\mathcal{M}' = \{\mathcal{S}, \mathcal{A}, \mathcal{R} + \mathcal{F}, \mathcal{P}, \gamma\}$. Any conventional RL algorithm can be used on the shaped process $\mathcal{M}'$, and it can improve the convergence rate of reinforcement learning agents. Successful shaping transforms the native process $\mathcal{M}$ such that the shaped process is easier to learn and still preserves the optimal policy of $\mathcal{M}$.

Taking advantage of the benefits that reward shaping brings to RL, we shape a new reward function (see Sec. 4.3) to guide the RL converging at an optimal point in policy learning.

3.3 Problem Formulation

The multi-robot navigation problem can be formulated as a sequential decision-making problem in a reinforcement learning framework [13, 32]. We consider N mobile robot navigating in a 2D environment, and each robot has its goal position $\boldsymbol{p}_g$. At each time t, each robot R_i has the state $\boldsymbol{s}_t$, the action $\boldsymbol{u}_t$, and its neighboring robots' state $\bar{s}_t = [s_{t,1}, \ldots s_{t,N_i}]$ where $\mathcal{N}_i$ is the number of neighbors. The state consists of two parts, i.e., internal and external state $\boldsymbol{s}_t = [\boldsymbol{s}_t^{in}, \boldsymbol{s}_t^{ex}]$. The internal states are the robot's position, velocity, and radius, $\boldsymbol{s}_t^{in} = [p_x, p_y, v_x, v_y, r] \in \mathbb{R}^5$, which can be obtained by the robots. The external states are the goal position, preferred velocity, and orientation, $\boldsymbol{s}_t^{ex} = [p_{gx}, p_{gy}, v_x^{pref}, v_y^{pref}, \psi] \in \mathbb{R}^5$, which can be specified by the global scheduling systems. The action is the velocity of robots $\boldsymbol{u}_t = [v_{xt}, v_{yt}] \in \mathbb{R}^2$. We aim to develop a policy $\pi_\theta : \left(\boldsymbol{s}_t, \bar{\boldsymbol{s}}_t^{in}\right) \mapsto \boldsymbol{u}_t$, with the objective of minimizing expected traveling time $\mathbb{E}[\mathcal{T}]$ to goal positions while avoiding collision with other robots,

$$\underset{\pi_\theta\left(s_t, \bar{s}_t^{in}\right)}{\text{argmin}} \quad \mathbb{E}\left[\mathcal{T} | s_0, \bar{s}_0^{in}, \pi_\theta\right] \tag{5}$$

$$\text{subject to}: \quad \forall t \in [1, \mathcal{T}],\ \tau < 0 \text{ or } \mathcal{D} < 0 \tag{6}$$

$$\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \pi_\theta(\boldsymbol{s}_{t-1}, \bar{\boldsymbol{s}}_{t-1}^{in}) \cdot \Delta t \tag{7}$$

$$\boldsymbol{p}_{\mathcal{T}} = \boldsymbol{p}_g \tag{8}$$

where Eq. (5) is the expected traveling time given the robots' state and the learned policy. Eq. (6) is the collision avoidance constraint, meaning that the distance between robot R_i and its neighbors is larger than the sum of their radius. Eq. (7) is the motion equation of robots, and Eq. (8) refers to the goal reaching constraint.

We use RL framework to learn the policy, by considering the robot's state with its neighbors' state, $\boldsymbol{s}_t = \left[\boldsymbol{s}_t, \bar{\boldsymbol{s}}_t^{in}\right]$. The policy is learned by mapping the robots' states to actions, then the goal of the robot is to find a policy that achieves maximal expected discounted total reward by learning from the experience of outcomes in the process. That is,

$$\mathcal{J}(\theta) = \mathbb{E}_{s \sim p^\pi, \boldsymbol{u} \sim \pi_\theta} \sum_{t=0}^{\mathcal{T}} \gamma_t \mathcal{R}_t \tag{9}$$

where p^π is the state distribution under policy π_θ, $\sum_{t=0}^{\mathcal{T}} \gamma_t \mathcal{R}_t$ is the cumulative rewards in a time $\mathcal{T}$. For each step in the learning process, the agent chooses an action and transitions to a new state by the dynamics of the domain model in the transition distributions. Each transition provides some reward as described by a reward function, in which the reward is shaped by the TTC-force and will be presented in Sec. 4.3.

4 OUR APPROACH

In this section, we first describe the overall architecture of our approach. Then, we introduce the reinforcement learning framework regarding observation and action space, and how to shape the reward with the TTC-force according to the time to collision. Finally, we exhibit how TTC-force guides reinforcement learning to learn a multi-robot navigation policy.

4.1 Framework Overview

The framework overview of the proposed *Tfresh* is shown in Fig. 2. We design three major components, i.e., the state encoders, an actor network, and a critic network. Specifically, we feed the robots' state into the state encoders, and the state encoders output the corresponding state embedding, the concatenated state embedding and adjacent matrix are fed into the spatial attention of the actor and critic network to capture the relatively important state representation. The state representation is fed into the multi-layer perceptron (MLP) feature decoder of the actor and critic network to output the action and critic score. Finally, the sum of actor and critic loss is used to update the neural network parameters with the Adam optimizer.

To compute the adjacent matrix, we first obtain the relative distance $\mathcal{M}_d^{ij}$ between robots R_i and R_j according to their positions. Then, the adjacent matrix $\mathcal{A}_s^{ij}$ is obtained by a softmax function over distance matrix $\mathcal{M}_d^{ij}$ with a scale parameter η.

$$\mathcal{A}_s^{ij} = \mathtt{softmax}(\eta \frac{1}{\mathcal{M}_d^{ij}}) \tag{10}$$

where $\mathtt{softmax}$ function is applied to the rows, the diagonal values are set as one. During the training stage, the adjacent matrix changes in each iteration. Thus we design a learned matrix $\mathcal{W}_d^{ij}$, which combines with the adjacent matrix to allow the model to capture the pattern of the adjacent relations. We named it as dynamic adjacent matrix $\mathcal{A}_d^{ij}$,

$$\mathcal{A}_d^{ij} = \mathtt{softmax}(\mathcal{W}_d^{ij} \odot \mathcal{A}_s^{ij}) \tag{11}$$

where $\odot$ is an element-wise product, $\mathcal{W}_d^{ij}$ is a learned parameter matrix. The dynamic adjacent matrix is employed on the state embedding, which can capture the relatively important state representation from neighboring robots. Finally, the resulting state representation is fed into the MLP decoders to get the action and critic scores.

4.2 Observation and Action Space

The observation represents the environmental information obtained by the robots. Given the observation $\boldsymbol{s}_t$ at time step t, the robots decide its optimal moving velocity decision, denoted as action $\boldsymbol{u}_t$. We consider that each robot R_i can observe its own state $\boldsymbol{s}_t^i$ and the states that are sent out by the number of N_i neighboring robots $\bar{\mathbf{s}}_t^{in}$. For simplicity, we omit the time t in the notations:

$$\boldsymbol{s} \;= \left[p_x,\, p_y,\, v_x,\, v_y,\, r,\, p_{gx},\, p_{gy},\, v_x^{pref}, v_y^{pref},\, \psi\right] \tag{12}$$

$$\bar{\mathbf{s}}^{in} = \left[\bar{p}_x^j,\, \bar{p}_y^j,\, \bar{v}_x^j,\, \bar{v}_y^j,\, r^j\right], j \in \{0, 1, \cdots, \mathcal{N}_i\} \tag{13}$$

The action of the robot is the velocity in the x and y plane, $\boldsymbol{u} = [v_x, v_y]$, which is bounded by the robot's motion constraint in the range of minimum and maximum velocity, $\boldsymbol{u} \in [v_{min}, v_{max}]$.

4.3 TTC-force Reward Shaping

The reward value is associated with each transition from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$ after taking action $\boldsymbol{u}_t$ to evaluate the quality of this action. The design of the reward function plays a vital role in reinforcement learning-based policy learning. This is because the proper reward function can obtain a satisfactory reward value, making learning faster and converging to an optimal point. Therefore, to enforce the multi-robot avoiding collision and reaching their goal positions, we shape the reward to enable the robot to plan a collision-free action which is termed as TTC-force reward, $\mathcal{R}_{TTC}$, and the reward to allow robots reach their goal as quickly as possible which is termed as goal-force reward, $\mathcal{R}_{Goal}$. Thus the total reward function of the robot R_i is represented as,

$$\mathcal{R} = \sum_{i \neq j}^{\mathcal{N}_i} \mathcal{R}_{\text{TTC}}\left(\Delta \boldsymbol{p}, \Delta \boldsymbol{v}, r_{ij}\right) + \mathcal{R}_{\text{Goal}}(\boldsymbol{v}_i) \tag{14}$$

where $\Delta \boldsymbol{p}_{ij}$, $\Delta \boldsymbol{v}_{ij}$ are the relative distance and velocity between robot R_i and R_j, respectively. $\mathcal{N}_i$ is the number of neighbors of R_i.

In particular, we first adopt the TTC model presented in Sec. 3.1, in which the time of potential collision is quantified as τ in Eq. (3). The TTC-force is inspired by the study of Karamouzas Ioannis et al. [20], in which the interaction behavior of two agents follows a power-law relationship [19] concerning the TTC τ. Thus the collision avoidance behavior can be quantified as the TTC-force between robots. Here, we define the interaction energy function as $\mathcal{U}(\Delta \boldsymbol{p}, \Delta \boldsymbol{v}, r_{ij})$ which is the function of τ and follows the power-law relationship:

$$\mathcal{U} = \mathcal{F}(\tau) = k\tau^{-m} e^{-\tau/\tau_{\mathcal{H}}} \tag{15}$$

where k is a scaling constant, and m denotes the exponent of the power-law. The time horizon $\tau_{\mathcal{H}}$ models the time duration that tends to ignore collisions. $\Delta \boldsymbol{p}$ and $\Delta \boldsymbol{v}$ are the relative position and velocity between two robots. The intuition behind the power-law relationship is that when τ is small, a collision is imminent, and a large avoiding force should be used to prevent the collision. When τ is large, the collision occurs far in the future, and the avoidance force should have a small magnitude and vanish to zero at the time horizon τ_H.

Then, the TTC-force between the robot R_i and R_j follows the negative gradient of the interaction energy according to their position $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. That is,

$$\mathcal{F}_{ij} = -\frac{\partial \mathcal{U}}{\partial \boldsymbol{p}_i} = -\mathcal{F}'(\tau)\frac{\partial \tau}{\partial \boldsymbol{p}_i} \tag{16}$$

We can obtain $\mathcal{F}'(\tau)$ from the derivative of Eq. (15),

$$\mathcal{F}'(\tau) = -\frac{ke^{-\tau/\tau_H}}{\tau^{m+1}}\left(m + \frac{\tau}{\tau_{\mathcal{H}}}\right) \tag{17}$$

As τ depends on $\Delta \boldsymbol{p} = \boldsymbol{p}_i - \boldsymbol{p}_j$, thus

$$\frac{\partial \tau}{\partial \boldsymbol{p}_i} = \frac{\partial \tau}{\partial \Delta \boldsymbol{p}} = \frac{\Delta \boldsymbol{p} + \Delta \boldsymbol{v}\tau}{\sqrt{\mathcal{D}}} \tag{18}$$

Combining Eq. (3) and Eq. (15), we can obtain the TTC-force of the robot R_i exerted by R_j,

$$\mathcal{F}_{ij} = \frac{ke^{-\tau/\tau_{\mathcal{H}}}}{\tau^{m+1}}\left(m + \frac{\tau}{\tau_{\mathcal{H}}}\right)\left(\frac{\Delta \boldsymbol{p} + \Delta \boldsymbol{v}\tau}{\sqrt{\mathcal{D}}}\right) \tag{19}$$

Additionally, we construct the goal-force to attract the robots to reach their goals quickly. Typically, a robot's goal-force is inferred from its goal velocity. No attraction is needed if the robot is currently moving in its desired direction and velocity. On the contrary, if the robot is moving too fast, too slow, or in the wrong direction, the goal-force can gradually drive the robot to its goal velocity with the following:

$$\mathcal{F}_{\text{Goal}} = k_{\text{g}} \left(v_i^{pref} - v_i \right) \tag{20}$$

where k_g is a tunable parameter controlling the strength of the goal-force, and v^{pref} denotes the preferred goal velocity.

To shape the reward, we cannot directly use the TTC-force $\mathcal{F}_{ij}$ and the goal-force F_{Goal}. This is because the TTC-force has the direction, which is selected to push the robot's predicted position away from its neighbor's predicted position. Thus, the negative absolute value of $\mathcal{F}_{ij}$ is regarded as the penalty if the robot R_i has a potential collision with R_j in time of τ. Similarly, to shape the reward with the goal-force F_{Goal}, we obtain the numeric reward value by obtaining the negative absolute value of F_{Goal}. Finally, we shape the **reward function** Eq. (14) for each robot R_i with N_i neighbors as

$$\begin{aligned} \mathcal{R} &= \sum_{i \neq j}^{\mathcal{N}_i} \mathcal{R}_{\text{TTC}} \left(\Delta \boldsymbol{p}, \Delta \boldsymbol{v}, r_{ij} \right) + \mathcal{R}_{\text{Goal}}(\boldsymbol{v}_i) \\ &= -\alpha | \sum_{i \neq j}^{\mathcal{N}_i} \mathcal{F}_{ij}(\Delta \boldsymbol{p}, \Delta \boldsymbol{v}, r_{ij}) | - \beta | \mathcal{F}_{\text{Goal}} \left(\boldsymbol{v}_i \right) | \end{aligned} \tag{21}$$

where $| \cdot |$ denotes the absolute value, α and β are constant values that can be tuned to adjust the policy performance.

4.4 Training Algorithm

To train the multi-robot navigating in a 2D workspace, we use proximal policy optimization (PPO) [31] to learn the navigation policy. This is because PPO is a robust and popular optimization method for multi-robot reinforcement learning. In this work, the detail of the training algorithm is shown in Algorithm 1. The algorithm follows the centralized training distributed execution framework with parameter sharing, where all robots share the same policy π_θ and centralized value function $\mathcal{V}_\phi$. Specifically, we begin with initializing the multi-robot training environment. In each episode, each of $\mathcal{N}$ robots collect $\mathcal{T}$ timesteps of data by executing the policy model $\pi_{\theta\text{old}}$ (line 3). Meanwhile, the trajectories data in terms of robot's state $\boldsymbol{s}_t^i$, action $\boldsymbol{u}_t^i$, and reward r_t^i are collected and stored into the experience buffer $\mathcal{B}$ (lines 6 ~ 8). We then construct the surrogate loss $\mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t[\min(\mathcal{R}_t(\theta)\hat{A}_t, \text{clip}(\mathcal{R}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$ (line 13) and mean square error of critic score $\mathcal{L}^{\mathcal{V}}(\phi) = \left(V_\phi(s_t) - \hat{\mathcal{R}}_t \right)^2$ (line 14). The navigation policy network parameters π_θ are optimized with Adam optimizer based on the total loss (lines 15 – 16) in the epoch number of $\mathcal{K}_{epoch}$ learning. Finally, we update the policy and value network model with the updated parameters (line 17).

Algorithm 1: Training Algorithm of Navigation Policy.

```
Input: Number of robots $\mathcal{N}$; Reward weights $\alpha, \beta \in [0, 1]$;
       Initialize the policy actor $\pi_\theta$ and critic $\mathcal{V}_\phi$;
       Experience buffer $\mathcal{B}$ for storing trajectories.
Output: A trained navigation policy model $\pi_\theta$.
1  Initialize the multi-robot training environment ;
2  for episode ← 1 to $\mathcal{N}_{eps}$ do
3      // Collect trajectory data with $\pi_{\theta\text{old}}$.
4      for robot $R_i$ ← 1 to $\mathcal{N}$ do
5          Run policy $\pi_{\theta\text{old}}$ for $T$ timesteps ;
6          Collect trajectories $(\boldsymbol{s}_t^i, \boldsymbol{u}_t^i, r_t^i(Eq. 21))$ for $T$ timesteps ;
7          Compute advantage estimates $\hat{A}_1^i, \cdots, \hat{A}_T^i$ ;
8          Store the data $\mathcal{B} \leftarrow \mathcal{B} \cup (\boldsymbol{s}_t, \boldsymbol{u}_t, r_t)$ ;
9      end
10     // Update policy and value network models.
11     for batch ← $\mathcal{B}$ do
12         for $k$ ← 1 to $\mathcal{K}_{epoch}$ do
13             Compute clipped surrogate objective $\mathcal{L}^{CLIP}(\theta)$ ;
14             Compute the mean square error $\mathcal{L}^{\mathcal{V}}(\phi)$ ;
15             Total loss = $\mathcal{L}^{CLIP}(\theta) + \mathcal{L}^{\mathcal{V}}(\phi)$ ;
16             Update parameters $\theta$ and $\phi$ with Adam optimizer ;
17             $\pi_{\theta\text{old}} \leftarrow \theta, \phi_{old} \leftarrow \phi$ ;
18         end
19     end
20 end
```

5 EXPERIMENTS AND RESULTS

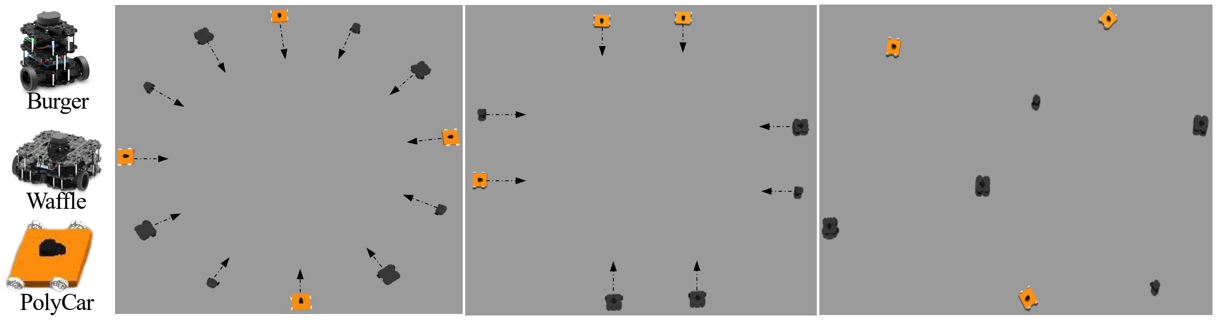
In this section, we conduct extensive experiments to evaluate our approach in the Gazebo simulator [7] and real-world environments, comparing our proposed approach with three different methods in terms of evaluation metrics. Specifically, we establish a simulator to train the navigation policy, then deploy and test the learned policy model in the simulator and the real-world multi-robot environment. Finally, we discuss the experimental results of *Tfresh* against baselines in different scenarios to show the benefits of our approach.

5.1 Experimental Settings

We design a multi-robot system consisting of a group of heterogeneous robots, in which different types of robots have different sizes and motion constraints in terms of maximum velocity. Specifically, we use three types of robots, i.e., Burger, Waffle, and PolyCar [2], as shown in Fig. 3 (a), in which the size and motion constraints of Burger and Waffle robots are different. The Burger and Waffle are differential drive [17] robots with 2 independently driven motors and 1 caster wheel for stability, and PolyCar is an Ackerman wheel robot with four independently driven motors. The size of Burger, Waffle, PolyCar in terms of radius are 0.15 m, 0.22 m, and 0.24 m, respectively. The maximum linear velocity is 0.5 m/s, and the maximum angular velocity is between −1.0 rad/s and 1.0 rad/s, respectively.

5.2 Implementation Details

The proposed *Tfresh* has state encoders, one actor network, and one critic network. The additional details of them are provided within the supplementary. We train the policy in simulation environments, and the well-trained model is tested in the simulator



(a) Robots (b) Circle Scenario (c) Crossing Scenario (d) Random Scenario

Figure 3: (a) Three types of robots, i.e., Burger, Waffle, and PolyCar. (b) 12 **robots navigate in a circle scenario, and all the robots exchange antipodal locations. (c)** 8 **robots navigate a crossing scenario in which the start and goal positions are selected randomly. (d)** 8 **robots navigate in a random scenario, where the start and goal positions are randomly generated in an** 8 **by** 8 **square region. The black arrow indicates the forward direction of the robots at the start position.**

Table 1: Comparison of navigation performance (*mean* ± *std*) of different methods, i.e., NH-ORCA [1], UTTC [12], RL-RVO [13], *Tfresh*, in the circle, crossing, and random scenarios, with varied scene sizes and different numbers of mobile robots.

		Circle Scenarios		*Crossing Scenario*	*Random Scenario*
Metrics	Methods	8 robots (8 m)	12 robots (8 m)	8 robots (6 m)	8 robots
Success Rate	NH-ORCA	1	1	0.8	0.76
	UTTC	1	1	0.9	0.8
	RL-RVO	1	0.98	0.9	0.92
	Tfresh	**1**	**1**	**1**	**1**
Travel Distance (m)	NH-ORCA	10.202 ± 0.028	11.508 ± 0.712	6.190 ± 0.912	5.378 ± 1.924
	UTTC	9.045 ± 0.6995	10.264 ± 2.609	9.269 ± 1.393	7.676 ± 3.241
	RL-RVO	8.610 ± 0.452	8.863 ± 0.708	7.726 ± 0.62	6.283 ± 2.094
	Tfresh	**8.536 ± 0.052**	**8.237 ± 0.013**	**6.963 ± 0.117**	**6.105 ± 2.521**
Travel Time (s)	NH-ORCA	19.588 ± 1.402	34.258 ± 18.293	21.984 ± 4.941	21.476 ± 7.757
	UTTC	17.487 ± 0.294	22.758 ± 2.145	30.878 ± 10.297	16.810 ± 5.629
	RL-RVO	18.413 ± 0.512	20.328 ± 1.854	16.548 ± 0.792	14.783 ± 4.836
	Tfresh	**17.375 ± 0.136**	**17.837 ± 0.249**	**16.232 ± 0.545**	**14.122 ± 5.231**

and the real-world environment. *Tfresh* is implemented with Pytorch [28] framework using Python language and runs on an Intel (R) Xeon(R) CPU E5-2680 V2@2.8GHz and 32G RAM.

5.3 Compared Methods

In particular, we compare our method with three baselines in the same environments. The technical details of them are introduced in the following.

- NH-ORCA [1], which is the velocity obstacle-based method that extends the method of ORCA [37] to non-holonomic robots. We use the open-source NH-ORCA implementation from [8] in the evaluation.
- UTTC [12], which is a TTC-based collision avoidance model considering uncertainty in the future trajectories of interacting robots. We implement the method according to the details presented in the paper.
- RL-RVO [13], the current state-of-the-art learning-based collision avoidance method, in which the reward function is designed based on the area of reciprocal velocity obstacle.

5.4 Evaluation Metrics

We evaluate the navigation performance with three metrics regarding success rate, travel distance, travel time, and computational cost in the experiments.

- Success rate - The success rate is a ratio of the successful cases without collision or being stuck during navigation, which measures the policy's collision avoidance ability.
- Travel distance - The travel distance refers to the average trajectory distance of robots from the start position to the goal position, reflecting the navigation policy's efficiency.
- Travel time - The travel time refers to the amount of average time taken by the robot from the start to the goal positions, reflecting the navigation policy's efficiency.
- Computational cost - The computation time of the collision avoidance method takes when executing on target platforms.

5.5 Results and Discussion

In this section, we present the experimental details and discuss the experimental results by deploying heterogeneous robots in both simulators and real-world environments.

5.5.1 *Performance in the Simulator.*

In the simulator, we evaluate the performance of different methods in the circle, crossing, and random scenarios. We conduct 50 times of test for each scenario by performing the navigation task, and the experimental results are shown in Table. 1.

Circle Scenario. To evaluate the performance of *Tfresh* in handling the imminent head-on collision, we design circle scenarios with the numbers of 8 and 12 robots, including Burger, Waffle, and PolyCar robots. As shown in Fig. 3 (b), there are a total number of 12 robots (4 for each type) that form a circle at their start position. The goal of each robot is the position of the robot opposite to them. The joint line distance (ground truth distance) from the start location to each robot's goal location is 8 m. We can observe that the head-on direction of the robots may hit each other when they travel close to the center of the circle in opposite directions. Thus, all robots should plan optimal actions to avoid imminent collisions and reach their goal positions.

As shown in Table. 1, compared to NH-ORCA, UTTC, and RL-RVO methods, *Tfresh* is superior to them in terms of success rate, average distance, and average time. The trajectories of robots executing *Tfresh* and the baselines are shown in the supplementary. The results of *Tfresh* are also superior to the baselines regarding the travel trajectories' smoothness and distance, indicating that *Tfresh* can efficiently control the robots.

Crossing Scenario. To evaluate the performance of *Tfresh* in handling the imminent collision from three perpendicular directions. We design the crossing scenario with 8 robots, including 2 Burger, 3 Waffle, and 3 PolyCar robots. As shown in Fig. 3 (c), crossing scenarios require each robot to move from the start position and reach the robot's position opposite to them. For each opposite pair, the initial directions are mutually parallel. Each robot must take action to avoid imminent collisions from the front-end half plane and cross the surrounding robots to reach its goal position.

As shown in Table. 1, the *Tfresh* has a 100% success rate which outperformed all the baselines. Compared to the trajectories with baselines, as shown in the supplementary, *Tfresh* still achieves a shorter trajectory distance and the highest success rate without getting stuck. The results indicate that *Tfresh* is more efficient when controlling the heterogeneous mobile robots' in the navigation.

Random Scenario. To evaluate the performance of *Tfresh* in complex environments with dense robots. We define robot density as the number of agents divided by the area of the workspace. Here, the density is $8/16 = 0.5$, meaning 0.5 agent per m^2. This density is high in automated warehouses and fulfillment centers because of the large size of the robots. We experiment with random scenarios with heterogeneous robots as shown in Fig. 3 (d), in which 2 Burger, 3 Waffle, and 3 PolyCar robots are deployed. We conduct 50 times of test by randomly setting each robot's non-overlapping start and goal positions. From the experimental results, we can observe that our proposed *Tfresh* can achieve a high success rate and efficient navigation in terms of average travel distance and travel time. This is because the proposed approach can enable the robots to catch important state information in the dynamic environment.

We have taken video demos for different scenarios which are available at https://youtu.be/H4cEQ4Fhc0k.

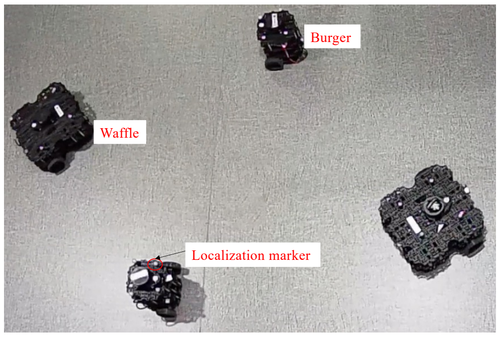


Figure 4: A real-world environment that deploys four robots includes two Burgers and two Waffles.

Table 2: The comparison of the computational cost of different approaches in milliseconds (ms).

NH-ORCA	UTTC	RL-RVO	*Tfresh*
10.05	11.20	103.76	90.51

5.5.2 *Performance in the Real-world.*

The real-world environment is shown in Fig. 4, we use the OptiTrack localization system [29] which consists of 6 cameras. Four physical robots including 2 Burger and 2 Waffle robots are used to perform the navigation tasks. Each mobile robot holds four markers to locate itself, and the localization error is around ±0.4 mm in the environment. Each robot is armed with a Raspberry Pi 3 Model B, an embedded computing platform consisting of a 32-bit ARM Cortex-M7 core. We evaluate the computational cost as shown in Table. 2. The results indicate that our proposed *Tfresh* has a lower computational cost than RL-RVO. This is because of the lightweight design of the policy network in our method.

6 CONCLUSIONS AND FUTURE WORK

This work presents a new navigation approach mitigating imminent collision in heterogeneous multi-robot systems, allowing the robots to move safely with each other and reach their goal positions efficiently. The proposed *Tfresh* overcomes the limitations in quantifying the collision risk of heterogeneous robots when adopting a reinforcement learning-based framework and the ineffective state representation under dynamic environments. We use the TTC-force to quantify the collision risk and shape it as the reward signal, which provides dense reward feedback to facilitate the convergence of robot learning. We use the dynamic adjacent matrix to allow the robots to pay more attention to the most important state representations, which helps to capture more effective state representations even with dynamically changing neighbors. The learned policy is deployed in numerous comprehensive experiments in both simulation and real-world environments. The experimental results indicate that *Tfresh* outperforms the baselines regarding navigation efficiency and lower computational cost. Our work currently considers the reward shaping between mobile robots. In the future, we will consider extending the TTC-force reward shaping approach for robots avoiding static obstacles or surrounding humans.

ACKNOWLEDGMENTS

This work was supported by the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University, HK RGC Research Impact Fund No. R5009-21 and R5034-18, Innovation and Technology Fund-Mainland-Hong Kong Joint Funding Scheme No. MHP/013/21, and the National Key Research and Development Program of China No. 2022YFF1101100.

REFERENCES

[1] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. 2013. Optimal reciprocal collision avoidance for multiple non-holonomic robots. (2013), 203–216. https://doi.org/10.1007/978-3-642-32723-0_15

[2] Robin Amsters and Peter Slaets. 2019. Turtlebot 3 as a robotics education platform. (01 2019), 170–181. https://doi.org/10.1007/978-3-030-26945-6_16

[3] T. Balch and M. Hybinette. 2000. Social potentials for scalable multi-robot formations. 1 (2000), 73–80. https://doi.org/10.1109/ROBOT.2000.844042

[4] Daman Bareiss and Jur Van den Berg. 2015. Generalized reciprocal collision avoidance. *The International Journal of Robotics Research* 34, 12 (2015), 1501–1514. https://doi.org/10.1177/0278364915576

[5] Jur van den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. In *Robotics research*. Springer, 3–19.

[6] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. 2019. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. (2019), 6015–6022. https://doi.org/10.1109/ICRA.2019.8794134

[7] Jinlin Chen, Jiannong Cao, Zhixuan Liang, Zhiqin Cheng, and Jia Wang. 2022. GraphWare: A graph-based middleware enabling multi-robot cooperation. *Concurrency and Computation: Practice and Experience* 34, 17 (2022), e6995. https://doi.org/10.1002/cpe.6995

[8] Daniel Claes, Daniel Hennes, Karl Tuyls, and Wim Meeussen. 2012. Collision avoidance under bounded localization uncertainty. (2012), 1192–1198. https://doi.org/10.1109/IROS.2012.6386125

[9] Michael Everett, Yu Fan Chen, and Jonathan P How. 2018. Motion planning among dynamic, decision-making agents with deep reinforcement learning. (2018), 3052–3059. https://doi.org/10.1109/IROS.2018.8593871

[10] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. 2020. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research* 39, 7 (2020), 856–892. https://doi.org/10.1177/0278364920916

[11] Paolo Fiorini and Zvi Shiller. 1998. Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research* 17, 7 (1998), 760–772. https://doi.org/10.1177/027836499801700706

[12] Zahra Forootaninia, Ioannis Karamouzas, and Rahul Narain. 2017. Uncertainty models for TTC-based collision-avoidance. 7 (2017). https://doi.org/10.15607/RSS.2017.XIII.002

[13] Ruihua Han, Shengduo Chen, Shuaijun Wang, Zeqing Zhang, Rui Gao, Qi Hao, and Jia Pan. 2022. Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards. *IEEE Robotics and Automation Letters* 7, 3 (2022), 5896–5903. https://doi.org/10.1109/LRA.2022.3161699

[14] John C Hayward. 1972. Near miss determination through use of a scale of danger. (1972).

[15] Brian Heater. 2022. *Amazon debuts a fully autonomous warehouse robot*. https://techcrunch.com/2022/06/22/amazon-debuts-a-fully-autonomous-warehouse-robot/

[16] Dirk Helbing, Illés Farkas, and Tamas Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803 (2000), 487–490. https://doi.org/10.1038/35035023

[17] Daniel Hennes, Daniel Claes, Wim Meeussen, and Karl Tuyls. 2012. Multi-robot collision avoidance with localization uncertainty. (2012), 147–154.

[18] Samaneh Hosseini Semnani, Anton H. J. de Ruiter, and Hugh H. T. Liu. 2022. Force-based algorithm for motion planning of large agent. *IEEE Transactions on Cybernetics* 52, 1 (2022), 654–665. https://doi.org/10.1109/TCYB.2020.2994122

[19] Ioannis Karamouzas, Brian Skinner, and Stephen J Guy. 2014. Universal power law governing pedestrian interactions. *Physical review letters* 113, 23 (2014), 238701. https://doi.org/10.1103/PhysRevLett.113.238701

[20] Ioannis Karamouzas, Nick Sohre, Rahul Narain, and Stephen J Guy. 2017. Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13. https://doi.org/10.1145/3072959.3073705

[21] Adam Daniel Laud. 2004. *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign.

[22] Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. 2020. Graph neural networks for decentralized path planning. (2020), 1901–1903.

[23] Qingbiao Li, Weizhe Lin, Zhe Liu, and Amanda Prorok. 2021. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters* 6, 3 (2021), 5533–5540. https://doi.org/10.1109/LRA.2021.3077863

[24] Minghua Liu, Hang Ma, Jiaoyang Li, and Sven Koenig. 2019. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS).*

[25] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. 2018. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*. 6252–6259. https://doi.org/10.1109/ICRA.2018.8461113

[26] Yuexin Ma, Dinesh Manocha, and Wenping Wang. 2018. Efficient reciprocal collision avoidance between heterogeneous agents using CTMAT. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1044–1052.

[27] Nima Moshtagh, Nathan Michael, Ali Jadbabaie, and Kostas Daniilidis. 2009. Vision-based, distributed control laws for motion coordination of nonholonomic robots. *IEEE Transactions on Robotics* 25, 4 (2009), 851–860. https://doi.org/10.1109/TRO.2009.2022439

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc.

[29] Tong Qin, Peiliang Li, and Shaojie Shen. 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics* 34, 4 (2018), 1004–1020. https://doi.org/10.1109/TRO.2018.2853729

[30] Alexander Schperberg, Stephanie Tsuei, Stefano Soatto, and Dennis Hong. 2021. SABER: Data-driven motion planner for autonomously navigating heterogeneous robots. *IEEE Robotics and Automation Letters* 6, 4 (2021), 8086–8093. https://doi.org/10.1109/LRA.2021.3103054

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[32] Samaneh Hosseini Semnani, Hugh Liu, Michael Everett, Anton De Ruiter, and Jonathan P How. 2020. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3221–3226. https://doi.org/10.1109/LRA.2020.2974695

[33] Naman Shah and Siddharth Srivastava. 2022. Using deep learning to bootstrap abstractions for hierarchical Robot planning. International Foundation for Autonomous Agents and Multiagent Systems, 1183–1191. https://doi.org/10.5555/3535850.3535982

[34] Weixian Shi, Yanying Zhou, Xiangyu Zeng, Shijie Li, and Maren Bennewitz. 2022. Enhanced spatial attention graph for motion planning in crowded, partially observable environments. In *International Conference on Robotics and Automation*. IEEE, 4750–4756. https://doi.org/10.1109/ICRA46639.2022.9812322

[35] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. 2021. Reward is enough. *Artificial Intelligence* 299 (2021), 103535.

[36] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. 2011. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics* 27, 4 (2011), 696–706.

[37] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. In *Robotics research*. Springer, 3–19.

[38] Richard Van Der Horst and Jeroen Hogema. 1993. Time-to-collision and collision avoidance systems. (1993).

[39] Wei Wang, Prosanta Gope, and Yongqiang Cheng. 2022. An AI-driven secure and intelligent robotic delivery system. *IEEE Transactions on Engineering Management* (2022), 1–16. https://doi.org/10.1109/TEM.2022.3142282

[40] Xi Vincent Wang and Lihui Wang. 2021. A literature survey of the robotic technologies during the COVID-19 pandemic. *Journal of Manufacturing Systems* 60 (2021), 823–836. https://doi.org/10.1016/j.jmsy.2021.02.005