2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Decentralized Multi-Robot Navigation Coupled with Spatial-Temporal RetNet Based on Deep Reinforcement Learning

Lin Chen[1], Yaonan Wang[1], Zhiqiang Miao[1], Mingtao Feng[2], Yuanzhe Wang[4], Yang Mo[1], Zhen Zhou[1], Hesheng Wang[3], *IEEE Senior Member*, and Danwei Wang[4], *IEEE Life Fellow*

***Abstract*— Navigating robots through dynamic multi-robot environments, avoiding collisions with both other robots and obstacles, has emerged as a central challenge in robotics. The existing approaches fall short in allowing the policy network to effectively capture spatial-temporal reciprocal collision avoidance in multi-robot environments, comprising both static and dynamic obstacles, resulting in inadequate safety and efficiency in directing robot movement. In this study, we introduce a novel policy neural network called Spatial-Temporal RetNet (STR), designed to encode reciprocal collision avoidance states between robots in spatial and temporal dimensions. The goal is to improve the safety and efficacy of the policy neural network in directing robots to complete assigned tasks. The spatial state encoder module is built upon a parallel RetNet structure, which strengthens the neural network's capacity in extracting reciprocal collision avoidance states between robots in spatial dimensions. This module addresses the limitations of position encoding in transformer-based multi-robot navigation policy neural networks. We design a temporal state encoder utilizing a recurrent RetNet structure. This innovation bolsters the multi-robot navigation policy neural network's capability to capture features in the temporal dimension of multi-robot movements. It addresses the limitations of transformer-based multi-robot navigation policy neural networks, particularly in recurrently inferring information across time dimensions. Simulation experiments were conducted to showcase the superior safety and effectiveness of our proposed method compared to previous state-of-the-art approaches in guiding robots to accomplish tasks.**

I. INTRODUCTION

Multi-robot collision avoidance has garnered attention from researchers in recent years due to its broad applicability in various scenarios, including autonomous driving, intelligent warehouse robotics, and crowd simulation [1]–[3]. At its core, the robot avoids collisions in environments with multiple robots, including both static and dynamic obstacles [3]–[6]. Current multi-robot navigation methods are typically categorized into centralized and decentralized approaches, depending on their reliance on a central server for coordination and decision-making [5]. In a centralized approach, a central server is utilized to coordinate the movement of each robot, leveraging information gathered from all robots to plan their trajectories. In this approach, the robot's path is initially planned without considering collisions. Subsequently, a scheduling scheme is employed to address potential collision instances at conflict locations, ensuring collision avoidance during navigation [7]. However, the growing number of robots escalates computational demands, resulting in increased resource requirements, longer computation times, and delays in the exchange of control signals between each robot and the central server.

In a decentralized framework, each robot within a multi-robot system can autonomously plan collision-free paths and execute tasks independently, relying on the environmental information it perceives [8]–[14]. While these methods address the limitations of centralized approaches, they encounter the challenge of being unable to determine optimal speeds under limited sensing information conditions. This involves ensuring safe and effective collision avoidance with both dynamic and static obstacles, as well as preventing collisions with other robots.

In recent years, machine learning methods have played a crucial role in improving the safety and effectiveness of policies guiding robots within decentralized frameworks as they complete specified tasks [14]–[21]. A decentralized multi-agent collision avoidance deep reinforcement learning algorithm (CADRL), as described in [1], was developed. This algorithm employs a value network to generate collision-free velocity vectors by considering the positions and velocities of the robot itself and surrounding robots. In GA3C-CADRL (GPU/CPU asynchronous advantage actor-critic CADRL) as presented in [8], the introduction of the long short-term memory (LSTM) structure, as outlined in [22], empowers the policy neural network to effectively encode the positions and velocities of an arbitrary number of surrounding robots. The policy neural network utilizes an attentive pooling mechanism as proposed in SARL [9], which indirectly enhances the robot's anticipation capability by extracting the relative importance of surrounding neighboring robots. In RL-RVO (reinforcement learning - RVO) as described in [4], the reciprocal velocity obstacle (RVO) and reinforcement learning methods are integrated to enable the policy to

This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFB3903804) and the National Natural Science Foundation of China (Grant No. 62293515, 62293510, 62027810, 62273138, 62133005, 62373293, 62103141).

[1]Lin Chen, Yaonan Wang, Zhiqiang Miao, Yang Mo, and Zhen Zhou are with the School of Electrical and Information Engineering, Hunan University, Changsha, 410082, China, and also with the National Engineering Research Center for Robot Visual Perception and Control Technology, Changsha 410082, China. (email: chenlin21@hnu.edu.cn; yaonan@hnu.edu.cn; miaozhiqiang@hnu.edu.cn; moyanghnu@hnu.edu.cn; zhenzhou@hnu.edu.cn) (Corresponding author: Yaonan Wang)

[2]Mingtao Feng is with the School of Artificial Intelligence, Xidian University, Xian, 710126, China.(email: mintfeng@hnu.edu.cn)

[3]Hesheng Wang is with the Department of Automation, Shanghai Jiao Tong University, Shanghai, 200030, China. (email: wanghesheng@sjtu.edu.cn)

[4]Yuanzhe Wang and Danwei Wang are with School of Electrical and Electrical Engineering, Nanyang Technological University, Nanyang Avenue, 639798, Singapore. (email: wang0951@e.ntu.edu.sg; EDWWANG@ntu.edu.sg)

guide the robot in avoiding collisions with other robots and obstacles, even in situations with limited information. Simultaneously, RL-RVO incorporates bidirectional gated recurrent units (BiGRUs) into the policy neural network, providing the capability to map sequential inputs and corresponding action outputs. These methods solely focus on the spatial dimension of interaction among robots, which results in insufficient safety and effectiveness of policies to guide robots to complete tasks, as they overlook the interaction among robots along the temporal dimension [10]. The ST^2 (Spatial-Temporal State Transformer) was devised utilizing the Transformer network architecture, consisting of both a global spatial state encoder and a temporal state encoder, as detailed in [10]. It concurrently considers environmental features across both temporal and spatial dimensions. However, the ST^2 policy neural network exhibits weak position encoding capability for spatial environmental state data and fails to consider reciprocal collision avoidance between robots. The task of the policy network to capture a spatial-temporal representation of reciprocal collision avoidance in intricate and dynamic multi-robot environments remains an challenge.

To overcome the above challenges, we introduce a novel Spatial-Temporal RetNet (STR) designed to capture the spatial-temporal state of reciprocal collision avoidance among robots in dynamic multi-robot environments, even in situations with limited information. The proximal policy optimization (PPO) algorithm is employed to train a policy neural network, empowering it to guide robots through a multi-robot environment featuring both static and dynamic obstacles. Our main contributions can be summarized as follows:

(1) We introduce a novel Spatial-Temporal RetNet (STR) aimed at encoding reciprocal collision avoidance states between robots in both spatial and temporal dimensions. This approach utilizes reinforcement learning to acquire the navigation policy, resulting in enhanced safety and effectiveness of multi-robot navigation in complex environments.

(2) The parallel RetNet architecture is proposed to develop a spatial state encoder, augmenting the neural network's capacity for extracting reciprocal collision avoidance states among robots within spatial dimensions. This approach addresses the limitations of transformer-based multi-robot navigation policy neural networks in position encoding.

(3) The recurrent RetNet structure is employed to design a temporal state encoder, enhancing the multi-robot navigation policy neural network's ability to encode features in the temporal dimension of multi-robot movements. It overcomes the transformer-based multi-robot navigation policy neural network's inability to recurrently infer information in the time dimension.

(4) The simulation results demonstrate that our method outperforms state-of-the-art approaches in terms of both effectiveness and safety in guiding robots to complete tasks.

II. Preliminaries

The issue of enabling robots to avoid collisions with other robots, dynamic obstacles, and static obstacles in a multi-robot environment can be framed as a constraint problem, effectively addressed using deep reinforcement learning (DRL). The typical DRL framework comprises four key elements: the state space of the agent, the reward function design, the policy neural network design, and the agent's action space. This framework aims to train the policy neural network to guide the agent in avoiding collisions and successfully completing tasks. The details of the state space of robots, the reward function design, and the action space of robots within the multi-robot collision avoidance reinforcement learning framework are elaborated in this section. At time t, the spatial state $\mathbf{o}_t$ of reciprocal collision avoidance between the robot and surrounding neighbor robots includes the state $\mathbf{o}_t^{\text{self}}$ of the robot itself and the state $\mathbf{o}_t^{\text{sur}}$ of neighboring robots, mathematically expressed as $\mathbf{o}_t = (\mathbf{o}_t^{\text{self}}, \mathbf{o}_t^{\text{sur}})$. For the temporal dimension, three consecutive moments, t, $t-1$, and $t-2$, of reciprocal collision avoidance spatial states are considered. Consequently, the spatial-temporal reciprocal collision avoidance state $\mathbf{O}_t$ is mathematically expressed as $\mathbf{O}_t = [\mathbf{o}_t, \mathbf{o}_{t-1}, \mathbf{o}_{t-2}]$. The robot's own state includes its current velocity, direction, desired velocity, and safety radius. The spatial state of the i-th robot at time t is mathematically represented as $\mathbf{o}_{it}^{\text{self}} = [\mathbf{v}_{it}, \mathbf{o}_{itr}, \mathbf{v}_{it}^{\text{des}}, R_{ic}]$. The reciprocal collision avoidance state encompasses the interaction between the robot and surrounding neighbor robots, including relative distance information and the time to collision at the current velocity. The relative distance between the center positions of the i-th and j-th robots at time t is mathematically represented as d_{ijt}. Similarly, the required time duration for the i-th and j-th robots to collide at their current velocities at time t is expressed as t_{eij}. However, if these two robots will never collide at their current velocities, t_{eij} becomes infinite. Since infinity cannot be computed, in reference [4], t_{eij} is replaced with the reciprocal of t_{eij} plus a constant C. This approach is adopted in this work, and during experimentation, the constant C is set to 0.2. It is mathematically expressed as follows,

$$r_{eij} = 1/\left(t_{eij} + C\right). \tag{1}$$

The RVO method is employed to determine the reciprocal collision avoidance state between robots, highlighting the mutual interaction involved. The reciprocal vector between the i-th and j-th robots is denoted as $\mathbf{c}_{ij}$. This reciprocal vector $\mathbf{c}$, which comprises six elements derived from the three components $\mathbf{v}_p$, $\mathbf{vl}$, and $\mathbf{vr}$, can be mathematically defined as $\mathbf{c} = [\mathbf{v}_p, \mathbf{vl}, \mathbf{vr}]$. The calculation of $\mathbf{v}_p$, $\mathbf{vl}$, and $\mathbf{vr}$ is depicted in Fig. 1. The spatial reciprocal collision avoidance states surrounding the i-th robot are represented as $\mathbf{o}_{ij}^{\text{sur}} = [\mathbf{c}_{ij}, d_{ij}, r_{eij}]$, where $j = 0, 1, \ldots, m$, and m denotes the number of nearby robots and obstacles.

The robot's velocity in the plane at time t is denoted as $\mathbf{v}_t$. This velocity is constrained within the range defined by the maximum velocity $\mathbf{v}_{\max}$ and the minimum velocity $\mathbf{v}_{\min}$, represented as $[\mathbf{v}_{\min}, \mathbf{v}_{\max}]$. The output of the policy neural network, which determines the agent's actions within the reinforcement learning framework, is defined as the velocity adjustment $\mathbf{a}$, expressed as $\mathbf{a} = [\Delta v_x, \Delta v_y]$. Consequently,

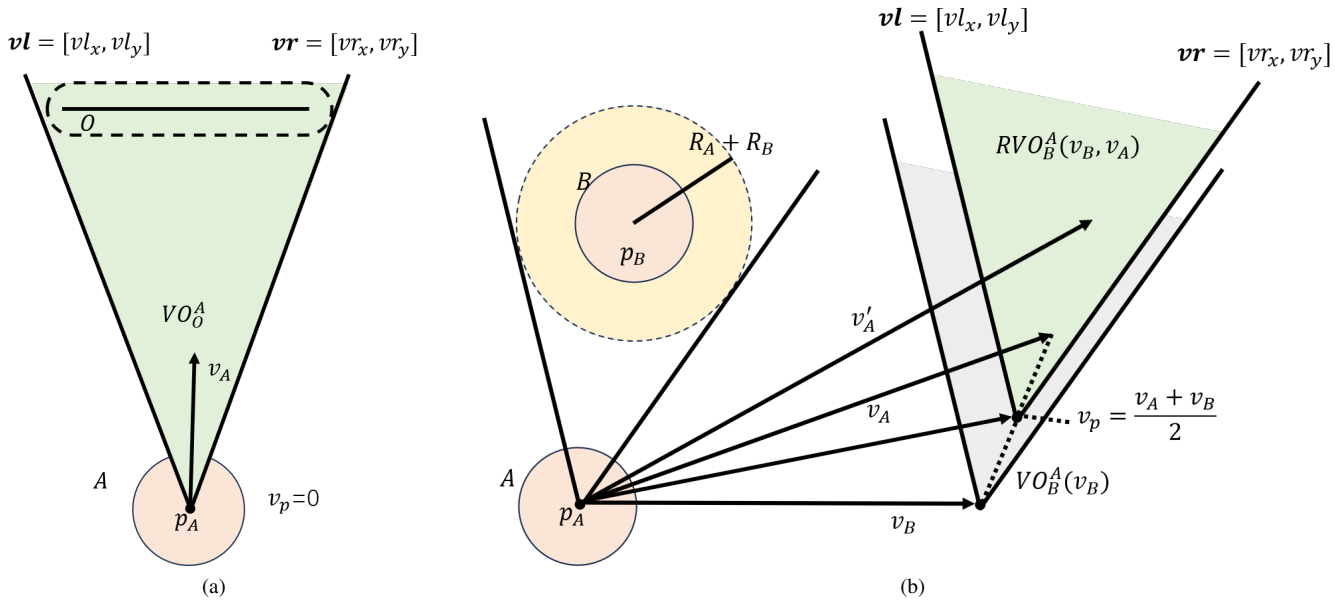


Fig. 1: The diagram illustrates the reciprocal velocity obstacles for both static line obstacles and dynamic agents.

the velocity $\mathbf{v}_{t+1}$ at time $t+1$ is computed as the current velocity $\mathbf{v}_t$ plus the adjustment $\mathbf{a}_t$, formulated as $\mathbf{v}_{t+1} = \mathbf{v}_t + \mu \cdot \mathbf{a}_t$. The reward function's design is a crucial aspect of the reinforcement learning framework, aimed at encouraging the robot to efficiently complete tasks while avoiding collisions with other robots, static obstacles, and dynamic obstacles. The reward function utilized in reference [4] is employed in this work, thus describing the reward value at time t as follows,

$$r^t = \begin{cases} -e * (\xi + f)^{-1} & \text{if } \xi \leq 0.1 \\ c - d * (\xi + f)^{-1} & \text{if } \mathbf{v}_t \in rvo \text{ and } \xi > 0.1 \\ a - b * des_v^t & \text{if } \mathbf{v}_t \notin rvo \text{ or } \xi > 5 \end{cases} \quad (2)$$

where ξ denotes the minimum time at which a collision will occur between the robot and its neighboring robots, given their current velocities. At time t, the discrepancy between the agent's current velocity and its desired velocity is mathematically expressed as des_v^t. In our experimental setup, the constants a, b, c, d, e, and f are assigned the values 0.3, 1.0, 0.3, 1.2, and 0.2, respectively. The criteria for determining whether the velocity $\mathbf{v}_t$ lies within the region rvo is defined as follows:

$$\begin{cases} \mathbf{v}_t \notin rvo := (\mathbf{v}_t - \mathbf{v}_p) \times \mathbf{vl} < 0 \vee (\mathbf{v}_t - \mathbf{v}_p) \times \mathbf{vr} > 0 \\ \mathbf{v}_t \in rvo := (\mathbf{v}_t - \mathbf{v}_p) \times \mathbf{vl} \geq 0 \wedge (\mathbf{v}_t - \mathbf{v}_p) \times \mathbf{vr} \leq 0 \end{cases} \quad (3)$$

where $\vee$ represents the logical OR operation, $\wedge$ signifies logical AND, and $\times$ denotes the vector product. The definitions and computations of the vectors $\mathbf{v}_p$, $\mathbf{vl}$, and $\mathbf{vr}$ are depicted in fig.1.

III. Spatial-Temporal RetNet

In this section, we present a novel Spatial-Temporal RetNet developed in this study. This network is designed to map environmental states perceived by robots and the corresponding actions they undertake. By training the network parameters using the reinforcement learning framework detailed in Section II, the RetNet enables robots to effectively avoid collisions with nearby robots, dynamic obstacles, and static obstacles.

As illustrated in Fig. 2, the Spatial-Temporal RetNet comprises the spatial state encoder module, the temporal state encoder module, and a fully connected neural network. At time steps t, $t-1$, and $t-2$, the environmental states perceived by the robot are encoded by the spatial state encoder module. In this work, the spatial state encoder module processes data from these three time steps using consistent network parameters and structure. These encoded states are subsequently input into the temporal state encoder module, yielding a spatial-temporal state. A two-layer fully connected network is employed to map the mean $\mathbf{v}_{\text{mean}}$ and the features STS_{LN}, derived from applying Layer Normalization to the spatial-temporal state. Similarly, an additional two-layer fully connected neural network serves as the critic network, where STS_{LN} is processed to obtain $V_\psi(\mathbf{o})$. The action executed by each robot is sampled from a Gaussian distribution defined by the mean $\mathbf{v}_{\text{mean}}$ and log standard deviation $\mathbf{v}_{\log\ \text{std}}$. Detailed descriptions of the spatial state encoder module and the temporal state encoder module can be found in Subsections III-A and III-B, respectively.

A. Spatial State Encoder Module

The spatial state encoder module is designed to capture the spatial characteristics of reciprocal velocity obstacle sequences by integrating a Bidirectional Gated Recurrent Unit (BiGRU) with a parallel Residual Network (RetNet) architecture. The fundamental component of the BiGRU is the GRU cell, which serves to encode sequences of varying lengths, corresponding to different numbers of neighboring

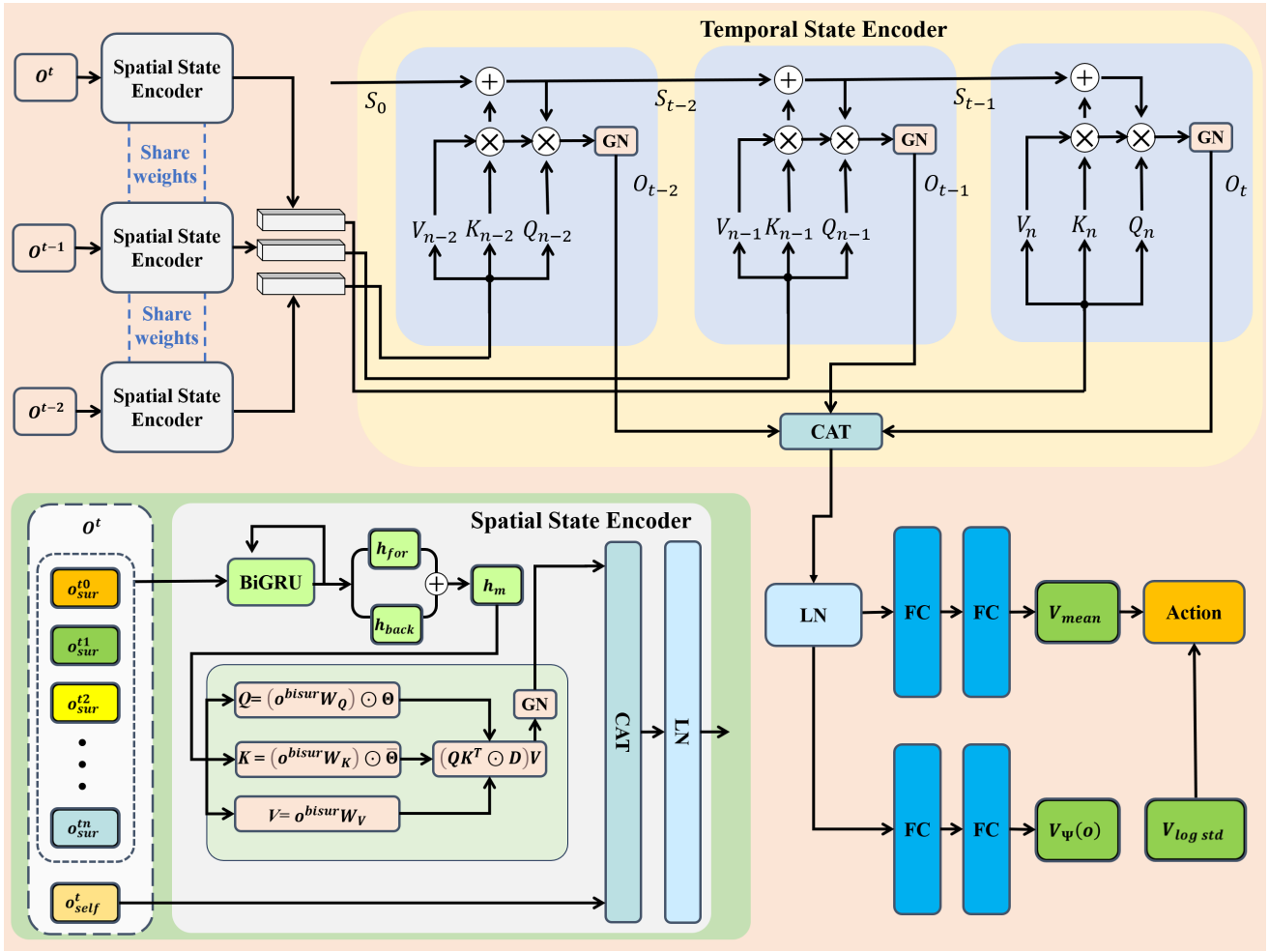


Fig. 2: Overview of the Spatial-Temporal RetNet architecture. The architecture comprises a spatial state encoder module, a temporal state encoder module, and a fully connected neural network. The notation CAT refers to the concatenation operation, LN stands for Layer Normalization, and GN denotes Group Normalization.

robots. The calculation of GRU is as follows,

$$\begin{aligned} z_t &= \sigma\left(\mathbf{W}_{\mathrm{z}}\left[h_{t-1}, x_t\right]\right) \\ r_t &= \sigma\left(\mathbf{W}_{\mathrm{r}}\left[h_{t-1}, x_t\right]\right) \\ \tilde{h}_t &= \tanh\left(\mathbf{W}\left[r_t \odot h_{t-1}, x_t\right]\right) \\ h_t &= \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \tilde{h}_t, \end{aligned} \tag{4}$$

where $\mathbf{W}_{\mathrm{z}}$, $\mathbf{W}_{\mathrm{r}}$, and $\mathbf{W}$ represent the learnable parameters of the network. The symbol $\odot$ signifies the Hadamard Product, which entails the element-wise multiplication of corresponding entries in matrices. The activation function, denoted by σ, is implemented as a sigmoid function in this study. The Bidirectional Gated Recurrent Unit (BiGRU) is constructed with two independent GRU units: one processes the sequence in the forward direction, while the other processes it in the reverse direction. The calculations for the forward and backward passes are as follows:

$$\begin{aligned} \overrightarrow{h_t} &= \mathrm{GRU}\left(\overrightarrow{h_{t-1}}, x_t\right) \\ \overleftarrow{h_t} &= \mathrm{GRU}\left(\overleftarrow{h_{t+1}}, x_t\right), \end{aligned} \tag{5}$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ denote the outcomes of the forward and backward computations, respectively. Upon encoding the neighboring states o_t^{sur} using a BiGRU, the resultant outputs are designated as h_{for} and h_{back}. The combined effect of h_{for} and h_{back} yields h_m, as illustrated by the following equation:

$$\begin{aligned} h_{for}, h_{back} &= BiGRU(o_t^{sur}), \\ h_m &= h_{for} + h_{back}. \end{aligned} \tag{6}$$

The encoding of h_m via the parallel RetNet results in the extraction of spatial features from the surrounding neighbor states, as illustrated in Fig. 2. The computation process of the parallel RetNet is detailed as follows:

$$\begin{aligned} Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \bar{\Theta}, \quad V = XW_V \\ \Theta_s = e^{is\theta}, \quad D_{sd} = \begin{cases} \gamma^{s-d}, & s \geq d \\ 0, & s < d \end{cases} \\ \mathrm{Retention}(X) = GN\left(\left(QK^{\top} \odot D\right) V\right), \end{aligned} \tag{7}$$

where W_Q, W_K, and W_V denote trainable network parameters, while $\bar{\Theta}$signifies the complex conjugate of Θ. The

term GN refers to Group Normalization. The matrix $D \in \mathbb{R}^{|x|\times|x|}$ is constructed by integrating exponential decay over relative distances with causal masking. Ultimately, the spatial dimension of the environmental state features is formed by concatenating the spatial features of the surrounding neighbor states with the robot's own state o_t^{self}, as mathematically represented below:

$$SSE\left(o_t\right) = LN\left(Concat\left(\text{Retention}(h_m), o_t^{self}\right)\right), \quad (8)$$

where $Concat$ denotes the concatenation operation, while LN stands for Layer Normalization.

B. Temporal State Encoder Module

The temporal state encoder module employs three recurrent RetNet modules to effectively capture the temporal characteristics of environmental states. The computational process is as follows:

$$\begin{aligned} S_{t-2}, O_{t-2} &= TSE_0\left(SSE\left(o_{t-2}\right), S_0\right) \\ S_{t-1}, O_{t-1} &= TSE_1\left(SSE\left(o_{t-1}\right), S_{t-2}\right) \\ O_t &= TSE_2\left(SSE\left(o_t\right), S_{t-1}\right) \end{aligned} \quad (9)$$

where SSE denotes the spatial state encoder module. The components TSE_0, TSE_1, and TSE_2 correspond to segments of the temporal state encoder module responsible for encoding spatial states at time steps $t-2$, $t-1$, and t, respectively. The computation of the initial segment TSE_0 of the temporal state encoder module is conducted as follows:

$$\begin{aligned} S_{t-2} &= \gamma S_0 + K_{n-2}^{\top} V_{n-2} \\ O_{t-2} &= GN\left(Q_{n-2} S_{t-2}\right). \end{aligned} \quad (10)$$

The computation method for Q, K, and V follows the approach outlined in equation 7, with γ denoting a scalar value. Initially, all entries in S_0 are set to zero. The procedures for calculating TSE_1 and TSE_2 are detailed as follows:

$$\begin{aligned} S_{t-1} &= \gamma S_{t-2} + K_{n-1}^{\top} V_{n-1}, \\ O_{t-1} &= GN\left(Q_{n-1} S_{t-1}\right), \end{aligned} \quad (11)$$

$$O_t = GN\left(Q_n\left(\gamma S_{t-1} + K_n^{\top} V_n\right)\right), \quad (12)$$

where γ is defined as in Equation 10. Similarly, the computation methods for Q, K, and V follow the same procedures outlined in Equation 7. The spatial-temporal state is formed by concatenating O_t, O_{t-1}, and O_{t-2}. The detailed calculation process is as follows:

$$STS_t = Concat\left(O_t, O_{t-1}, O_{t-2}\right), \quad (13)$$

where STS_t represents the robot's spatio-temporal state at time t.

IV. Experiments and Results

In this section, we begin with a comprehensive overview of the simulation experiment setup and the evaluation metrics employed. Following this, we present a comparative analysis between our proposed policy and other state-of-the-art methods to assess its efficacy.

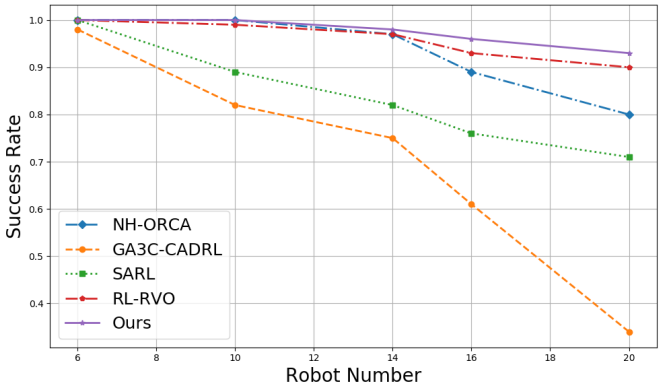


Fig. 3: Line Chart of Success Rates for Different Methods in the Circle Scenario

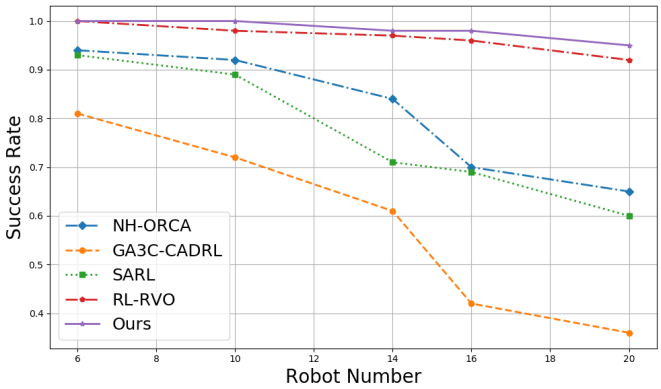


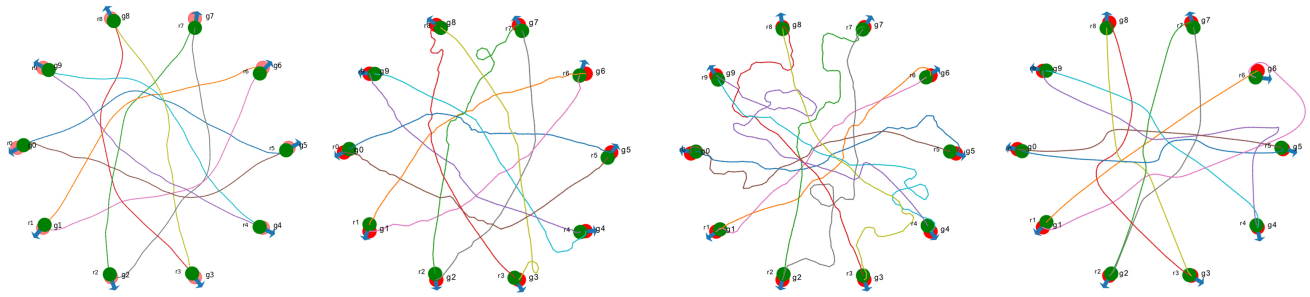
Fig. 4: Line Chart of Success Rates for Different Methods in the Random Scenario

A. Simulation setup

The simulation environment from RL-RVO [4], which is based on OpenAI Gym, is employed in this study. Training and testing of the policy neural network are conducted using two scenarios: the circle scenario and the random scenario. The proximal policy optimization (PPO) algorithm [23] is used to update the parameters of the policy neural network during the training process.In the random scenario, initial and target positions for all robots are generated randomly within a 5×5 meter area. Conversely, in the circle scenario, robots are evenly distributed along a circular path with a radius of 4.5 meters, with target positions set as the

TABLE I: The results of the ablation studies conducted within the Circle scenario.

Model	Success Rate	Travel Time /std	Average Speed (m/s) / std
Ours	0.99	100.72/4.67	0.90/0.03
STR_{T+R}	0.97	113.36/8.27	0.81/0.05
STR_{P+T}	0.98	104.12/7.37	0.86/0.05

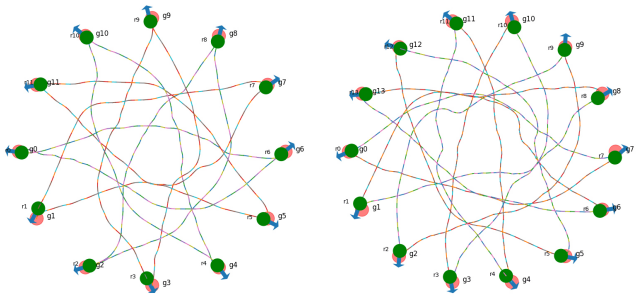


(a) Ours (b) SARL (c) GA3C-CADRL (d) NH-ORCA

Fig. 5: The trajectory plots of 10 robots being guided by our policy, SARL, GA3C-CADRL, and NH-ORCA in the circle scenario.

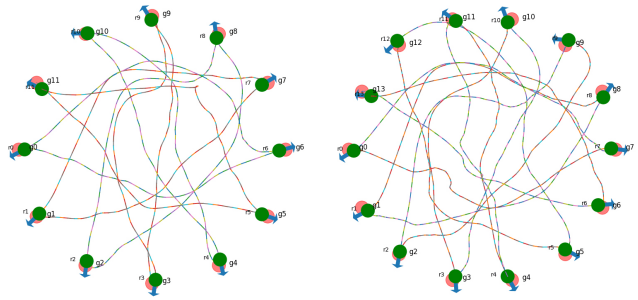
TABLE II: The results of the ablation studies conducted within the random scenario

Model	Success Rate	Travel Time /std	Average Speed (m/s) / std
Ours	0.99	85.57/13.28	0.65/0.07
STR_{T+R}	0.98	94.18/8.27	0.59/0.13
STR_{P+T}	0.97	94.39/19.25	0.57/0.10



(a) iteration step is 83 (b) iteration step is 89

Fig. 6: The trajectory plots and iteration steps of 12 and 14 robots guided by our policy in the circle scenario.



(a) iteration step is 98 (b) iteration step is 101

Fig. 7: The trajectory plots and iteration steps of 12 and 14 robots guided by RL-RVO in the circle scenario.

center-symmetric points relative to their starting positions.We evaluate the policy's safety and effectiveness in guiding robot motion using metrics such as success rate, travel time, and average speed. A task is considered successful if the robots complete it without collisions under policy guidance. We conduct 100 test trials, defining the success rate as the ratio of successful attempts to total attempts. Travel time is measured as the iteration step at which all robots reach their respective targets. Average speed is calculated as the mean velocity of all robots during their movement.A total of 10 test scenarios are set up, including both circle and random scenarios, with configurations involving 6, 10, 14, 16, and 20 robots, respectively.

The policy neural network initially undergoes supervised learning to calibrate certain parameters prior to further refinement through reinforcement learning. In the reinforcement learning phase, an interactive scenario is established, featuring a circular environment with 10 robots. The network parameters are updated using the Adam optimizer [24], with the policy learning rate set at 4e-6 and the value function learning rate at 5e-5. Simulations and training were conducted on a system equipped with an Intel i7-9700K CPU and an Nvidia Titan XP GPU.

B. Simulation results

We compare the proposed method with RL-RVO [4], SARL [9], GA3C-CADRL [8], and NH-ORCA [25] in terms of safety and effectiveness in guiding robots to accomplish designated tasks. The success rates for different methods in circular and random scenarios, with varying numbers of robots, are depicted in Fig. 3 and Fig. 4, respectively. The data clearly demonstrate that as the number of robots per unit area increases, the success rate for task completion diminishes. This trend highlights the increased likelihood of collisions with a higher robot density. Our method exhibits a superior success rate compared to the other four methods, underscoring its enhanced safety in robot navigation.

As illustrated in Fig. 5, our proposed policy, SARL [9], GA3C-CADRL [8], and NH-ORCA [25] successfully guide the trajectories of robots in a circular formation with 10 robots. Figs. 6 and 7 respectively show the paths and iteration

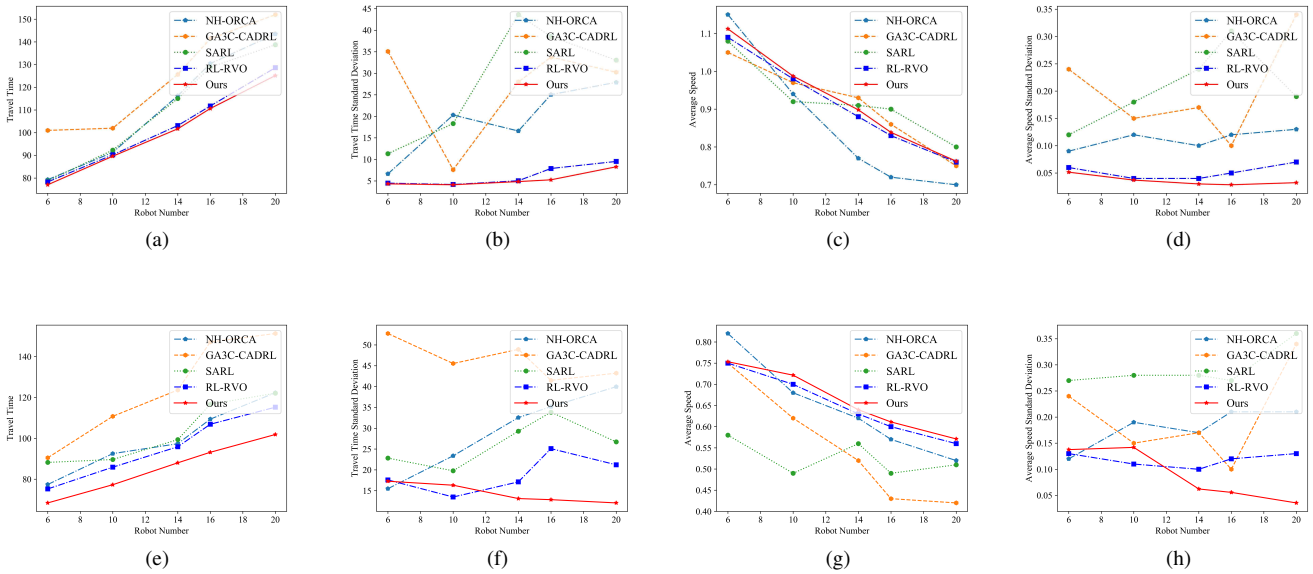


Fig. 8: The line plots of travel time and average speed, along with their variance, where (a)-(d) represent the results of guiding robot motion with different methods in the circle scenario, and (e)-(h) represent those in the random scenario.

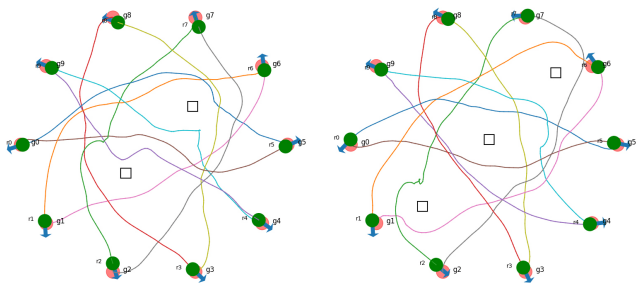


Fig. 9: The trajectory plots of 10 robots being guided by our policy in environments containing varying numbers of static obstacles.

steps of our policy compared to the state-of-the-art RL-RVO method, guiding robot motion in circular scenarios with 12 and 14 robots. Figs. 5, 6, and 7 indicate that our policy provides superior guidance for robot motion relative to the other advanced methods. To evaluate the efficiency of robot task completion, we employed travel time and average speed as metrics. The experimental results in both circular and random scenarios with varying numbers of robots are depicted in Fig. 8. Specifically, Fig. 8(a) and Fig. 8(e) show that the average travel time for task completion under our policy is shorter than that of the other four methods, demonstrating improved effectiveness. Furthermore, Fig. 8(b) and Fig. 8(f) reveal a lower variance in travel time, indicating greater stability of our policy. The average speeds of robot motion, as guided by our policy and the other four methods, in circular and random scenarios are presented in Fig. 8(c) and Fig. 8(g), respectively. Our policy achieves a higher average speed compared to the RL-RVO method, as shown in these figures. Additionally, the low variance in average speed, illustrated in Figs. 8(d) and 8(h), suggests that the stability of our policy is on par with the other methods. GA3C-CADRL and SARL do not adequately represent reciprocal interaction information among robots. RL-RVO employs a BiGRUs module to represent the spatial states of robots, resulting in insufficient representational capacity of the policy network. Moreover, these three methods consider only spatial information, neglecting the temporal characteristics inherent in robot motion.

The ablation experiments aim to determine whether the spatial-temporal RetNet addresses the limitations observed in the transformer-based multi-robot navigation policy neural network. To this end, we developed STR$_{P+T}$ by integrating a transformer block into the spatial state encoder, replacing the parallel RetNet. Similarly, STR$_{T+R}$ was created by substituting a transformer block for the recurrent RetNet in the temporal state encoder module. Both STR$_{P+T}$ and STR$_{T+R}$ policy networks were trained using the same scenarios and methodologies as the spatial-temporal RetNet.We assessed the performance of STR$_{P+T}$, STR$_{T+R}$, and the spatial-temporal RetNet in circle scenarios and random scenarios, each involving 13 robots. The results, detailed in Tables I and II, encompass success rate, travel time, and average speed metrics for both scenarios. The spatial-temporal RetNet consistently outperformed STR$_{T+R}$ in terms of success rate, travel time, and average speed, demonstrating that the spatial state encoder enhances the ability to guide robots more effectively than the transformer-based policy neural network, primarily by improving position encoding capabilities. Additionally, STR$_{P+T}$ was outperformed by the

spatial-temporal RetNet, indicating that the temporal state encoder significantly enhances the safety and efficiency of robot navigation by improving the recurrent processing of temporal information. The efficacy of our policy was further evaluated in simulated environments with both static and dynamic obstacles. As illustrated in Fig. 9, the trajectories of 10 robots navigating through scenarios with varying numbers of static obstacles showcase the robustness of our policy.

V. Conclusion

In this study, we introduce the Spatial-Temporal RetNet (STR), aimed at guiding multiple robots through environments featuring static and dynamic obstacles. The goal is to navigate these robots, ensuring collision avoidance and successful completion of navigation tasks, achieved through reinforcement learning training. We employ the parallel RetNet structure to develop a spatial state encoder, augmenting the neural network's capability in multi-robot navigation policies to extract reciprocal collision avoidance states between robots in spatial dimensions. To improve the multi-robot navigation policy neural network's capacity to encode features in the temporal dimension of multi-robot movements, we introduce the recurrent RetNet structure to design a temporal state encoder. Simulation experiment results demonstrate that the Spatial-Temporal RetNet enhances the safety and effectiveness of multi-robot navigation policies in guiding robots through multi-robot environments, surpassing the state-of-the-art method RL-RVO. Ablation experiments reveal that the spatial state encoder enhances task completion performance compared to the transformer-based multi-robot navigation policy neural network by strengthening position encoding capability. Additionally, the temporal state encoder improves the safety and effectiveness of robot motion by enhancing recurrent inference of information in the time dimension. In future research, our focus will be on exploring how robots can navigate efficiently in denser and more intricate environments, prioritizing safe motion.

References

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

[2] L. Chen, Y. Wang, Z. Miao, Y. Mo, M. Feng, Z. Zhou, and H. Wang, "Transformer-based imitative reinforcement learning for multirobot path planning," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 10, pp. 10 233–10 243, Oct 2023.

[3] L. Chen, Y. Wang, Z. Miao, M. Feng, Z. Zhou, H. Wang, and D. Wang, "Toward safe distributed multi-robot navigation coupled with variational bayesian model," *IEEE Transactions on Automation Science and Engineering*, pp. 1–16, 2023.

[4] R. Han, S. Chen, S. Wang, Z. Zhang, R. Gao, Q. Hao, and J. Pan, "Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5896–5903, 2022.

[5] L. Chen, Y. Wang, Y. Mo, Z. Miao, H. Wang, M. Feng, and S. Wang, "Multiagent path finding using deep reinforcement learning coupled with hot supervision contrastive loss," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 7, pp. 7032–7040, 2023.

[6] J. Lv, Y. Wang, S. Wang, X. Bai, R. Wang, and M. Tan, "A collision-free planning and control framework for a biomimetic underwater vehicle in dynamic environments," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 3, pp. 1415–1424, 2023.

[7] Z. Zhang, R. Han, and J. Pan, "An efficient centralized planner for multiple automated guided vehicles at the crossroad of polynomial curves," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 398–405, 2022.

[8] 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)*, 2018, pp. 3052–3059.

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

[10] Y. Yang, J. Jiang, J. Zhang, J. Huang, and M. Gao, "St^2: Spatial-temporal state transformer for crowd-aware autonomous navigation," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 912–919, 2023.

[11] K.-T. Song, S.-Y. Jiang, and S.-Y. Wu, "Safe guidance for a walking-assistant robot using gait estimation and obstacle avoidance," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 5, pp. 2070–2078, 2017.

[12] J. Xu, C. Qian, J.-W. Park, and K.-S. Park, "Adaptive sampling-based moving obstacle avoidance for cable-driven parallel robots," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 4983–4993, 2022.

[13] I. Mas and C. Kitts, "Obstacle avoidance policies for cluster space control of nonholonomic multirobot systems," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 6, pp. 1068–1079, 2012.

[14] 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)*. IEEE, 2018, pp. 6252–6259.

[15] M. Feng, H. Hou, L. Zhang, Y. Guo, H. Yu, Y. Wang, and A. Mian, "Exploring hierarchical spatial layout cues for 3d point cloud based scene graph prediction," *IEEE Transactions on Multimedia*, pp. 1–13, 2023.

[16] M. Feng, Z. Li, Q. Li, L. Zhang, X. Zhang, G. Zhu, H. Zhang, Y. Wang, and A. Mian, "Free-form description guided 3d visual graph network for object grounding in point cloud," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3722–3731.

[17] L. Chen, Y. Wang, Z. Miao, M. Feng, Z. Zhou, H. Wang, and D. Wang, "Reciprocal velocity obstacle spatial-temporal network for distributed multirobot navigation," *IEEE Transactions on Industrial Electronics*, pp. 1–11, 2024.

[18] Y. Zhai, B. Ding, X. Liu, H. Jia, Y. Zhao, and J. Luo, "Decentralized multi-robot collision avoidance in complex scenarios with selective communication," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8379–8386, 2021.

[19] H. Wang, A. H. Tan, and G. Nejat, "Navformer: A transformer architecture for robot target-driven navigation in unknown and dynamic environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 6808–6815, 2024.

[20] M. Feng, S. Z. Gilani, Y. Wang, L. Zhang, and A. Mian, "Relation graph network for 3d object detection in point clouds," *IEEE Transactions on Image Processing*, vol. 30, pp. 92–107, 2020.

[21] M. Feng, K. Liu, L. Zhang, H. Yu, Y. Wang, and A. Mian, "Learning from pixel-level noisy label: A new perspective for light field saliency detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1756–1766.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

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

[25] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, *Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 203–216. [Online]. Available: https://doi.org/10.1007/978-3-642-32723-0_15