2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Adaptive Leader-Follower Formation Control and Obstacle Avoidance via Deep Reinforcement Learning

Yanlin Zhou[†,1], Fan Lu[†,1], George Pu[†,1], Xiyao Ma[1],
Runhan Sun[2], Hsi-Yuan Chen[2], and Xiaolin Li[1]

***Abstract*— We propose a deep reinforcement learning (DRL) methodology for the tracking, obstacle avoidance, and formation control of nonholonomic robots. By separating vision-based control into a perception module and a controller module, we can train a DRL agent without sophisticated physics or 3D modeling. In addition, the modular framework averts daunting retrains of an image-to-action end-to-end neural network, and provides flexibility in transferring the controller to different robots. First, we train a convolutional neural network (CNN) to accurately localize in an indoor setting with dynamic foreground/background. Then, we design a new DRL algorithm named Momentum Policy Gradient (MPG) for continuous control tasks and prove its convergence. We also show that MPG is robust at tracking varying leader movements and can naturally be extended to problems of formation control. Leveraging reward shaping, features such as collision and obstacle avoidance can be easily integrated into a DRL controller.**

I. Introduction

The traditional control problem of dynamical systems with nonholonomic constraints is a heavily researched area because of its challenging theoretical nature and its practical use. A wheeled mobile robot (WMR) is a typical example of a nonholonomic system. Researchers in the control community have targeted problems in WMR including setpoint regulation, tracking [1], and formation control [2]. Due to the nature of these problems, the control law design involves sophisticated mathematical derivations and assumptions [3].

One of these problems is image-based localization, which involves an autonomous WMR trying to locate its camera pose with respect to the world frame [4]. It is an important problem in robotics since many other tasks including navigation, SLAM, and obstacle avoidance require accurate knowledge of a WMR's pose [4]. PoseNet adopts a convolutional neural network (CNN) for indoor and outdoor localization [5]. Using end-to-end camera pose estimation, the authors sidestep the need for feature engineering. This method was later extended by introducing uncertainty modeling for factors such as noisy environments, motion blur, and silhouette lighting [6]. Recently, there is an emerging trend of localization in dynamic indoor environments [7], [8].

Given accurate localization methods, various vision-based control tasks such as leader following can be accomplished. The leader-following problem is defined as an autonomous vehicle trying to follow the movement of a leader object [9]. The form of a leader is not limited to an actual robot, but can also include virtual markers [10], which can serve as a new method for controlling real robots.

However, a virtual leader is able to pass through a territory that a real follower cannot, which require the follower to perform obstacle avoidance [11]. The major challenge of applying classical control methods to the obstacle avoidance problem is having controllers correctly respond to different obstacles or unreachable areas. [12], [13]. A common approach is to design an adaptive or hybrid controller by considering all the cases, which is time-consuming [11].

Formation control is a leader-follower problem but with multiple leader/followers. It includes the additional challenge of collision avoidance among group members [14]. Two major approaches include graph-based solutions with interactive topology and optimization-based solutions [15]. However, the nonlinearity of nonholonomic robots adds to the challenges of modeling robots and multi-agent consensus. Deep RL solves these problems, but training an agent requires complex environmental simulation and physics modeling of robots [16], [17].

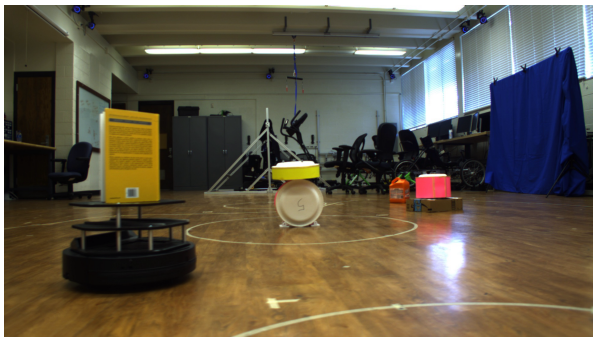


Fig. 1. A screenshot from our custom dataset.

In this paper, modular localization circumvents the challenge of not having an actual WMR's physics model or time-consuming environmental simulation. While end-to-end training from images to control has become popular [18], [19], partitioning the problem into two steps—localization and control—enables the flexibility of retraining the con-

† Equal contribution.

This work was supported in part by National Science Foundation (CNS-1624782, CNS-1747783) and Industrial Members of NSF Center for Big Learning (CBL).

[1] National Science Foundation Center for Big Learning, Large-scale Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, 32611-6250, USA zhou.y@ufl.edu; fan.lu@ufl.edu; pu.george@ufl.edu; maxiy@ufl.edu; andyli@ece.ufl.edu

[2] Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611-6250 USA runhansun@ufl.edu; hychen@ufl.edu

troller while reusing the localization module. The flowchart of our modular design is shown in Figure 2. The first phase of this work is to apply a CNN model called residual networks (ResNets) [20] to vision-based localization. We focus on indoor localization which includes challenges such as dynamic foreground, dynamic background, motion blur, and various lighting. The dynamic foreground and background are realized by giving intermittent views of landmarks. A picture of our training and testing environment is shown in Figure 1. We show that our model accurately predicts the position of robots, which enables DRL without need for a detailed 3D simulation.

To replace traditional controllers that usually involve complex mathematical modeling and control law design, we leverage DRL for several control problems including leader tracking, formation control, and obstacle avoidance. We propose a new actor-critic algorithm called Momentum Policy Gradient (MPG), an improved framework of TD3 that reduces under/overestimation [21]. We theoretically and experimentally prove MPG's convergence and stability. Furthermore, the proposed algorithm is efficient at solving leader-following problems with regular and irregular leader trajectories. MPG can be extended to solve the collision avoidance and formation control problems by simply modifying the reward function.

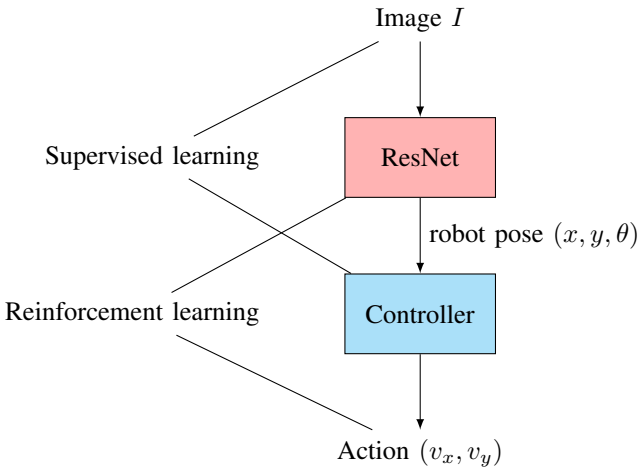


Fig. 2. Information flow between modules. Blocks are neural networks.

In summary, our contribution is four-fold.

- We propose Momentum Policy Gradient for continuous control tasks that combats under/overestimation.
- A modular approach that circumvents the need for complex modelling work or control law design.
- Robust image-based localization achieved using CNNs.
- Natural extensions from the single leader-follower tracking problem to collision/obstacle avoidance and formation control with reward shaping.

II. Mobility of Wheeled Robots

We consider the problem of forward kinematics for WMRs that has been extensively studied over the decades. A WMR collects two inputs: angular and linear velocities. The velocity inputs are then fed into on-board encoders to generate torque for each wheel. Due to the different sizes of robots, number of wheels, and moving mechanisms, robots can be classified into holonomic agents (*i.e.,* omni-directional Mecanum wheel robots) [22] and nonholonomic agents (*i.e.,* real vehicles with constrained moving direction and speed) [23]. In this paper, we consider nonholonomic agents.

Ideally, angular and linear velocities are the action components of a DRL control agent. However, these two action spaces have very different scales and usually cause size-asymmetric competition [24]. We found out that training a DRL agent with angular and linear velocities as actions converges slower than our methods presented below. Since there is no loss in degree of freedom, it is sufficient to use scalar velocities in x and y axes similar to the work done in [2].

Let (x_i, y_i) be the Cartesian position, θ_i as orientation, and (v_i, w_i) denote linear and angular velocities of the WMR agent i. The dynamics of each agent is then

$$\dot{x}_i = v_i \cos(\theta_i), \quad \dot{y}_i = v_i \sin(\theta_i), \quad \dot{\theta}_i = w_i \tag{1}$$

The nonholonomic simplified kinematic Equation (2) can then be derived by linearizing (1) with respect to a fixed reference point distance d off the center of the wheel axis (x_i', y_i') of the robot, where $x_i' = x_i + d_i \cos\theta, y_i' = y_i + d_i \sin\theta$.

Following the original setting $d = 0.15$ meters in [2], it is trivial to transfer velocity signals as actions used in nonholonomic system control.

$$\begin{bmatrix} v_i \\ w_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -(1/d)\sin\theta_i & -(1/d)\cos\theta_i \end{bmatrix} \begin{bmatrix} a_{x_i'} \\ a_{y_i'} \end{bmatrix} \tag{2}$$

where $a_{x_i'}$ and $a_{y_i'}$ are input control signals to each robot i.

In addition, other differential drive methods such as Instantaneous Center of Curvature (ICC) can be used for nonholonomic robots. However, compared to (2), ICC requires more physical details such as distance between the centers of the two wheels and eventually only works for two-wheeled robots [25]. Meanwhile, decomposing velocity dynamics to velocities on x and y axes can be reasonably applied to any WMRs.

III. Localization

The localization module focuses on the problem of estimating position directly from images in a noisy environment with both dynamic background and foreground. Several landmarks (*i.e.,* books, other robots) were placed so as to be visible in the foreground. Using landmarks as reference for indoor localization tasks has proven to be successful for learning-based methods [26]. As the WMR moves around the environment, a camera captures only intermittent view of landmarks and their position as the frame changes.

Overall, data was gathered from 3 types of robot trajectories (*i.e.,* regular, random, whirlpool) with multiple trials taking place at different times of day when the background

changes greatly and lighting conditions vary from morning, afternoon, and evening. The image dataset and ground truth pose of the agent was collected using a HD camera and a motion capture system, respectively. As the WMR moved along a closed path, images were sampled at rate of 30 Hz, and the ground truth pose of the camera and agent at a rate of 360 Hz. Data collection was performed at Nonlinear Controls and Robotics Lab. The camera used for recording images is placed on a TurtleBot at a fixed viewing angle. An example from our dataset is displayed in Figure 1.

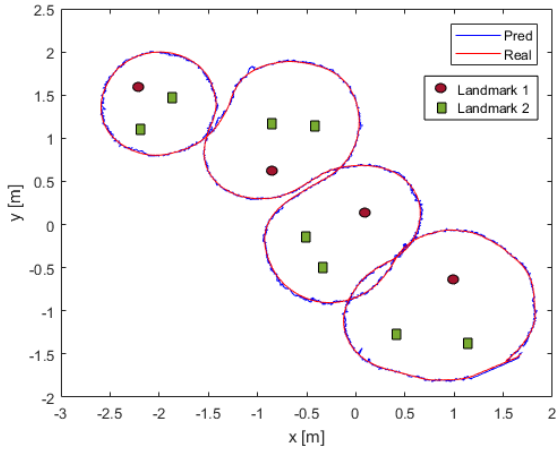


Fig. 3. Example snail trajectory with predicted pose.

A ResNet-50 model [20] with a resized output layer is trained from scratch on the dataset. Residual networks contain connections between layers of different depth to improve backpropagation. This eliminates the vanishing gradient problems encountered when training very deep CNNs. The ResNet predicts the robots current 2D position, orientation, and the distance to nearby landmarks. Distance to landmarks was not used in later modules, but helps focus attention onto the landmarks, which is a more reliable indicator of current pose, instead of any background changes between images.

We claim that our approach is robust enough to accurately predict a robot's pose for even very complex trajectories. These paths can have multiple points of self-intersection. Furthermore, ResNet based localization works well even with a dynamic foreground (multiple landmarks) and a dynamic background. Different lighting conditions and changes in background objects between trials do not affect the accuracy. The error rates are given in Table I for the robot's 2D coordinates and 4D quaternion orientation. They are all less than 1%. Figure 3 shows an example predicted and motion captured poses as well as landmarks; there is almost no difference between the two.

TABLE I
RESNET52 POSE PREDITION ERROR

x (%)	y (%)	q1 (%)	q2 (%)	q3 (%)	q4 (%)
0.439	0.1191	0.7462	0.3199	0.1673	0.152

IV. MOMENTUM POLICY GRADIENT

Algorithm 1 Momentum Policy Gradient

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $Q_{\hat{\theta}_1}, Q_{\hat{\theta}_2}$ with $\hat{\theta}_1 = \theta_1, \hat{\theta}_2 = \theta_2$
Create empty experience replay buffer E
for $t = 1$ **to** T **do**
  Take action $a = \pi_\phi(s) + \mathcal{N}(0, v_{explore})$
  $v_{explore} = \min(\lambda \cdot v_{explore}, v_{min})$
  Get next state s', reward r from environment
  Push (s, a, s', r) into E
  Sample mini-batch of N transitions from E
  Set $\Delta_{last} = 0$
  for $i = 1$ **to** I **do**
    $a' \leftarrow \pi_\phi(a) + \mathcal{N}(0, v_{train})$
    $\Delta_{adj} \leftarrow \frac{1}{2}(\Delta_{last} + \left|Q_{\hat{\theta}_1}(s', a') - Q_{\hat{\theta}_2}(s', a')\right|)$
    $\Delta_{last} \leftarrow \left|Q_{\hat{\theta}_1}(s', a') - Q_{\hat{\theta}_2}(s', a')\right|$
    $q \leftarrow \max(Q_{\hat{\theta}_1}(s', a'), Q_{\hat{\theta}_2}(s', a')) - \Delta_{adj}$
    $y \leftarrow r + \gamma q$
    $L^C \leftarrow \frac{1}{N} \sum_{batch} \sum_{i=1,2} (Q_{\theta_i}(s', a') - y)^2$
    Minimize L^c
    if $i \bmod F = 0$ **then**
      $L^A \leftarrow$ average of $-Q_{\theta_1}(s, \pi_\phi(a))$
      Minimize L^a
      $\theta_1 \leftarrow \tau\theta_1 + (1-\tau)\hat{\theta}_1$
      $\theta_2 \leftarrow \tau\theta_2 + (1-\tau)\hat{\theta}_2$
    end if
  end for
end for

Since nonholonomic controllers have a continuous action space, we design our algorithm based on the framework established by DDPG [27]. There are two neural networks: a policy network predicts the action $a = \pi_\phi(s)$ given the state s, a Q-network Q_θ estimates the expected cumulative reward for each state-action pair.

$$Q_\theta(s, a) \approx \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right] \tag{3}$$

The Q-network is part of the loss function for the policy network. For this reason, the policy network is called the actor and the Q-network is called the critic.

$$L^A = Q_\theta(s, \pi_\phi(s)) \tag{4}$$

The critic itself is trained using a Bellman equation derived loss function.

$$L^C = (Q_\theta(s, a) - [r + \gamma Q_{\hat{\theta}}(s', \pi_\phi(s'))])^2 \tag{5}$$

However, this type of loss leads to overestimation of the true total return [28]. TD3 fixes this by using two Q-value estimators $Q_{\theta_1}, Q_{\theta_2}$ and taking the lesser of the two [21].

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i}(s', \pi_{\phi_1}(s')) \tag{6}$$

Note that this is equivalent to taking the maximum, and then subtracting by the absolute difference. However, always choosing the lower value brings underestimation and higher variance [21].

To lower the variance in the estimate, inspired by the momentum used for optimization in [29], we propose Momentum Policy Gradient illustrated in Algorithm 1 which averages the current difference with the previous difference Δ_{last}.

$$q = \max(Q_{\hat{\theta}_1}(s', a'), Q_{\hat{\theta}_2}(s', a')) - \Delta_{adj} \tag{7}$$

$$\Delta_{adj} = \frac{1}{2}\left(|Q_{\theta_1}(s', a') - Q_{\theta_2}(s', a')| + \Delta_{last}\right) \tag{8}$$

This combats overestimation bias more aggressively than just taking the minimum of $Q_{\theta_1}, Q_{\theta_2}$. Moreover, this counters any over-tendency TD3 might have towards underestimation. Because neural networks are randomly initialized, by pure chance $|Q_{\theta_1}(s', a') - Q_{\theta_2}(s', a')|$ could be large. However, it is unlikely that Δ_{last} and $|Q_{\theta_1}(s', a') - Q_{\theta_2}(s', a')|$ are both large as they are computed using different batches of data. Thus Δ_{adj} has a lower variance than $|Q_{\theta_1}(s', a') - Q_{\theta_2}(s', a')|$.

In the case of negative rewards, the minimum takes the larger $Q_{\theta_i}(s', a')$ in magnitude. This will actually encourage overestimation (here the estimates trend toward $-\infty$).

Before proving the convergence of our algorithm, we first require a lemma proved in [30].

Lemma 1: Consider a stochastic process $(\alpha_t, \Delta_t, F_t), t \in \mathbb{N}$ where $\alpha_t, \Delta_t, F_t \colon X \to \mathbb{R}$ such that

$$\Delta_{t+1}(x) = [1 - \alpha_t(x)]\Delta_t(x) + \alpha_t(x)F_t(x)$$

for all $x \in X, t \in \mathbb{N}$. Let (P_t) be a sequence of increasing σ-algebras such that $\alpha_t, \Delta_t, F_{t-1}$ are P_t-measurable. If

1) the set X is finite
2) $0 \leq \alpha_t(x_t) \leq 1$, $\sum_t \alpha_t(x_t) = \infty$, but $\sum_t \alpha_t^2(x_t) < \infty$ with probability 1
3) $\|\mathbb{E}[F_t|P_t]\| \leq \kappa \|\Delta_t\| + c_t$ where $\kappa \in [0, 1)$ and c_t converges to 0 with probability 1
4) $\mathrm{Var}[F_t(x)|P_t] \leq K(1 + \|\Delta_t\|)^2$ for some constant K,

Then Δ_t converges to 0 with probability 1.

The theorem and proof of MPG's convergence is borrowed heavily from those for Clipped Double Q-learning [21].

Theorem 1 (Convergence of MPG update rule): Consider a finite MDP with $0 \leq \gamma < 1$ and suppose

1) each state-action pair is sampled an infinite number of times
2) Q-values are stored in a lookup table
3) Q, Q' receive an infinite number of updates
4) the learning rates satisfy $0 < \alpha_t(s_t, a_t) < 1$, $\sum_t \alpha_t(s_t, a_t) = \infty$, but $\sum_t \alpha_t^2(s_t, a_t) < \infty$ with probability 1
5) $\mathrm{Var}[r(s, a)] < \infty$ for all state-action pairs.

Then Momentum converges to the optimal value function Q^*.

Proof: Let $X = \mathcal{S} \times \mathcal{A}$, $\Delta_t = Q_t - Q^*$, and $P_t = \{Q_k, Q'_k, s_k, a_k, r_k, \alpha_k\}_{k=1}^t$. Conditions 1 and 2 of Lemma 1 are satisfied. By the definition of Momentum Policy Gradient,

$$\begin{aligned}
\Delta_t^{adj} &= \frac{1}{2}(|Q_t(s_t, a_t) - Q'_t(s_t, a_t)| + \\
&\qquad |Q_{t-1}(s_{t-1}, a_{t-1}) - Q'_{t-1}(s_{t-1}, a_{t-1})|) \\
y_t &= r_t + \gamma(m_t - \Delta_t^{adj}) \\
Q_{t+1}(s_t, a_t) &= [1 - \alpha_t(s_t, a_t)]Q_t(s_t, a_t) + \alpha_t(s_t, a_t)y_t
\end{aligned}$$

where $m_t = \max\{Q_t(s_t, a_t), Q'_t(s_t, a_t)\}$. Then

$$\begin{aligned}
\Delta_{t+1}(s_t, a_t) &= [1 - \alpha_t(s_t, a_t)][Q_t(s_t, a_t) - Q^*(s_t, a_t)] \\
&\quad + \alpha_t(s_t, a_t)[y_t - Q^*(s_t, a_t)] \\
&= [1 - \alpha_t(s_t, a_t)][Q_t(s_t, a_t) - Q^*(s_t, a_t)] \\
&\quad + \alpha_t(s_t, a_t)F_t(s_t, a_t)
\end{aligned}$$

where

$$\begin{aligned}
F_t(s_t, s_t) &= y_t - Q^*(s_t, a_t) \\
&= r_t + \gamma(m_t - \Delta_t^{adj}) - Q^*(s_t, a_t) \\
&= r_t + \gamma(m_t - \Delta_t^{adj}) - Q^*(s_t, a_t) \\
&\quad + \gamma Q_t(s_t, a_t) - \gamma Q_t(s_t, a_t) \\
&= F_t^Q(s_t, a_t) + \gamma b_t
\end{aligned}$$

We have split F_t into two parts: a term from standard Q-learning, and γ times another expression.

$$\begin{aligned}
F_t^Q(s_t, a_t) &= r_t + \gamma Q_t(s_t, a_t) - Q^*(s_t, a_t) \\
b_t &= m_t - \Delta_t^{adj} - Q_t(s_t, a_t)
\end{aligned}$$

As it is well known that $\mathbb{E}\left[F_t^Q|P_t\right] \leq \gamma \|\Delta_t\|$, condition 3) of Lemma 1 holds if we can show b_t converges to 0 with probability 1. Let $\Delta'_t = Q_t - Q'_t$. If $\Delta'_t(s_t, a_t) \to 0$ with probability 1, then

$$m_t \to Q_t(s_t, a_t), \qquad \Delta_t^{adj} \to 0$$

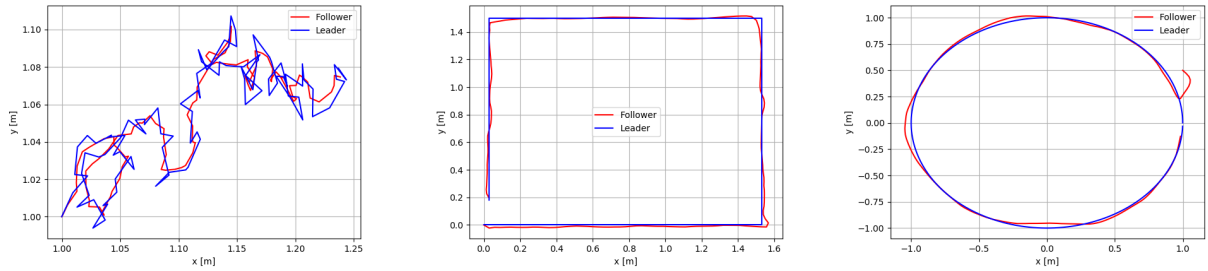
so $b_t \to 0$. Therefore showing $\Delta'_t(s_t, a_t) \to 0$ proves that b_t converges to 0.

$$\begin{aligned}
\Delta'_{t+1}(s_t, a_t) &= Q_{t+1}(s_t, a_t) - Q'_{t+1}(s_t, a_t) \\
&= [1 - \alpha_t(s_t, a_t)]Q_t(s_t, a_t) + \alpha_t(s_t, a_t)y_t - \\
&\quad ([1 - \alpha_t(s_t, a_t)]Q'_t(s_t, a_t) + \alpha_t(s_t, a_t)y_t) \\
&= [1 - \alpha_t(s_t, a_t)]\Delta'_t(s_t, a_t)
\end{aligned}$$

This clearly converges to 0. Hence Q_t converges to Q^* as $\Delta_t(s_t, a_t)$ converges to 0 with probability 1 by Lemma 1. The convergence of Q'_t follows from a similar argument, with the roles of Q and Q' reversed. ■

V. Continuous Control

We demonstrate MPG on a variety of leader-follower continuous control tasks. In all simulations, neural networks have two hidden layers of 400 and 300 units respectively. Output and input sizes vary depending on the environment and purpose of the model. Constraints on the motion of robots (Table II) are enforced by ending episodes once they are breached. A penalty of $-k_b$ is also added to the reward



(a) Test on random (b) Test on square (c) Test on circle

Fig. 4. Performance of a follower trained at 100 episodes with random leader on (a) random trajectory, (b) square leader, (c) circle leader.

TABLE II

KINEMATIC CONSTRAINTS OF AGENTS

	x_{min}	x_{max}	y_{min}	y_{max}	v_{min}	v_{max}
Leader	-1	1	-1	1	-0.7	0.7
Follower	-2	2	-2	2	-0.7	0.7

TABLE III

MQN HYPERPARAMETERS

Hyper-parameter	**Symbol**	**Value**
Actor Learning Rate	α	10^{-3}
Critic Learning Rate	α_C	10^{-2}
Batch Size	–	16
Discount factor	γ	0.99
Number of steps in each episode	–	200
Training noise variance	v_{train}	0.2
Initial exploration noise variance	$v_{explore}$	2
Minimum exploration noise variance	v_{min}	0.01
Exploration noise variance decay rate	λ	0.99

for that time step. The hyperparameters of MPG are given in Table III.

Suppose there are N agents whose kinematics are governed by Equations 1 and 2. Let $z_i = [\dot{p}_i \ \ddot{p}_i]^\mathsf{T}$ denote the state of agent i and the desired formation control for agents follow the constraint [31]:

$$F(z) = F(z^*) \tag{9}$$

Definition 1: From (9), we consider displacement-based formation control with the updated constraint given as:

$$F(z) := [...(z_j - z_i)^\mathsf{T}...]^\mathsf{T} = F(z^*) \tag{10}$$

Each agent measures the position of other agents with respect to a global coordinate system. However, absolute state measurements with respect to the global coordinate system is not needed. A general assumption made for formation control communication is that all agent's position, trajectory or dynamics should be partially or fully observable [3], [32].

The leader-follower tracking problem can be viewed as formation control with only $N = 2$ agents.

A. Tracking

We first test MPG by training a follower agent to track a leader whose position is known. The leader constantly moves without waiting the follower. The follower agent is punished based on its distance to the leader and rewarded for being very close. Let p_l be the 2D position of the leader and p_f be the corresponding position for the follower. The reward function for discrete leader movement is defined as

$$r = -\left\|p_l - p_f\right\| \tag{11}$$

where $\|\cdot\|$ is the L2 norm.

Then, We train a follower to track a leader moving in a circular pattern. Remarkably, this follower: can generalize to a scaled-up or scaled-down trajectory, is robust to perturbations in the initial starting location, and even tracks a leader with an elliptical motion pattern that it has never encountered before. However, the follower fails to track a square leader which is caused by under exploration of the entire environment. The under exploration issue is further highlighted when training a follower to track a leader with square motion. The model learns to go straight along an edge in less than 10 episodes, but fails to learn to turn a 90° angle for around 50 episodes.

Agents trained against a random moving leader generalize well to regular patterns.

$$v_x^l, v_y^l \sim \mathcal{N}(0, 1) \tag{12}$$

This is very similar to the discrete courier task in [33]. The random leader provides the follower with a richer set of possible movements compare to previous settings. As shown in Figure 4, this allows the follower to track even more regular trajectories. The follower is robust to changes in the initial position, as seen in Figure 4(c) where the follower starts outside the circle. Average distance between the follower and the leader and the average reward are given in Table IV.

TABLE IV

RANDOM LEADER, SINGLE FOLLOWER RESULTS.

	Average Distance	Average Reward
Random	0.0047	-0.5027
Square	0.0218	-2.4007
Circle	0.0394	-4.3689

For comparison, we also trained several followers using TD3. An example reward is displayed in Figure 5. The values are smoothed using a 1D box filter of size 200. For the circle

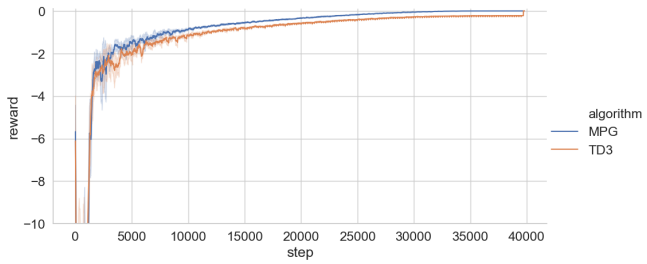
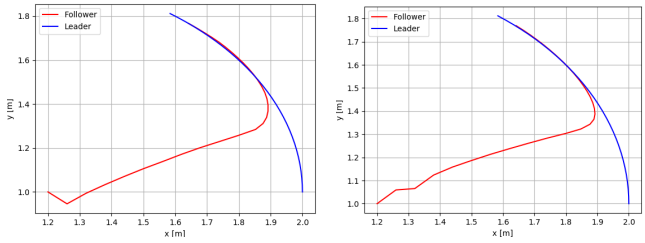


Fig. 5. MPG vs. TD3 rewards during training for the circle leader-follower task.



(a) TD3 (b) MPG

Fig. 6. Performance of TD3 and MPG followers trained with identical hyperparameters at 200 episodes.

leader task, TD3 which struggles to close the loop, slowly drifting away from the leader as time progresses. The MPG trained agent does not suffer from this problem.

We also noticed that some TD3 trained followers do not move smoothly. This is demonstrated in Figure 6. When trying to track a circle, these agents first dip down from $(1, 1.2)$ despite the leader moving counter-clockwise, starting at $(2, 1)$.

B. Formation Control

We naturally extend from the tracking problem to displacement-based formation control by adding additional trained followers to track the sole leader. However, additional work is needed to achieve collision avoidance among the followers. Based on displacement-base formation control, we design and conduct two simulations similar to [34]:

- **Unison formation control**: There is a pre-defined formation F with respect to global frame. All agents maintain a rigid formation throughout the entire movement process. The orientation of the formation does not change.
- **Consensus formation control**: Agents are given freedom to adapt rapidly while keeping the formation. The orientation of the formation can be changed.

In all simulations, a single neural network takes as input the positions of all the agents with respect to a global coordinate system and issues movement commands to all agents in the same coordinate system.

For unison formation control, groups of agents have a rigid formation that should not rotate. Given the position of the leader, whose index is 1, each follower i should try to minimize its distance to an intended relative location $\mathcal{F}_i$ with respect to the leader while avoiding collisions. As the leader moves, the expected positions $\mathcal{F}_i$ move in unison. Let C be the minimum safety distance, above which collision can be avoided. The reward is

$$r = -k_c n_c - \sum_{i \geq 1} \|p_i - \mathcal{F}_i\| \tag{13}$$

where k_c is collision coefficient and n_c is the number of collisions and p_i is the position of agent i. Upon any collisions, we reset the environment and start a new episode.

Then we explore unison formation control for a square formation with curved and random leader movements. As seen in Figure 7, three followers move in unison equally well when tracking (a) a leader with smooth trajectory starting from lower left corner and (b) a random leader. During training, we observed that adding more agents results in longer training time. This is because adding an agent increases the state space and action space, and thus our input and output dimensions, by 2. Training time grows approximately linearly in the number of agents. This makes it intractable to train and deploy large formations in the hundreds or thousands of robots. The average reward and distances are reported in Table V.

TABLE V

UNISON AVERAGE REWARD AND DISTANCES

Pattern	Reward	Dist. to $\mathcal{F}_2$	Dist. to $\mathcal{F}_3$	Dist. to $\mathcal{F}_4$
Random	-0.7214	0.0089	0.0116	-0.0134
Regular	-0.5239	0.0011	0.0123	-0.0026

TABLE VI

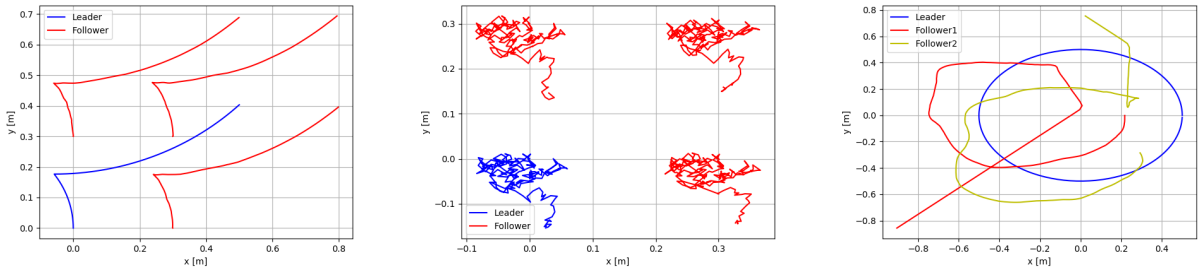
CONSENSUS AVERAGE REWARD AND DISTANCES

Reward	$\lvert d_{1,2} - \mathcal{D}_{1,2} \rvert$	$\lvert d_{1,3} - \mathcal{D}_{1,3}) \rvert$	$\lvert d_{2,3} - \mathcal{D}_{2,3} \rvert$
-27.9942	0.0219	0.0203	0.0275

Unlike unison formation control which dictates the individual motion of all agents, multi-agent consensus keeps a formation with respect to the local frame. The formation can rotate with respect to the global frame. The problem definition allows for switching and expansion within a given topology, as long as there are no collisions. This can be beneficial. For example, the agents may need to move further apart to avoid an obstacle or tighten their formation to fit through some passageway. In general, agents i, j should maintain a constant distance $\mathcal{D}_{i,j}$ to each other. Letting $d_{i,j} = \|p_i - p_j\|$, the reward function is given according to the following equation.

$$r = -k_c n_c - \sum_{i,j} |d_{i,j} - \mathcal{D}_{i,j}\ | \tag{14}$$

In our simulations, we trained 2 follower agents to maintain triangle formation with a leader undergoing random motion. As shown in Figure 7 (c), we test the performance of the multi-agent consensus while having the leader traverse a counter-clockwise circular trajectory. The leader starts at $(0.5, 0.5)$, follower 1 starts at a random lower left position, and follower 2 starts at a random upper right position.



(a) Rigid Unison with regular leader (b) Rigid Unison with random leader (c) Multi-agent Consensus

Fig. 7. Formation control: (a) and (b) use the unison definition while (c) allows the followers to swap positions.

Initially, the three agents are very far away from each other but quickly formed a triangle when the leader reaches around $(0.1, 0.5)$. We observe that the followers swapped their local positions within the formation when the leader arrives $(-0.5, 0.0)$. This is because the reward function is only interested in their relative distances, and the agents can maintain the formation with less total movement by switching their relative positions in the group. Results are reported in Table VI.

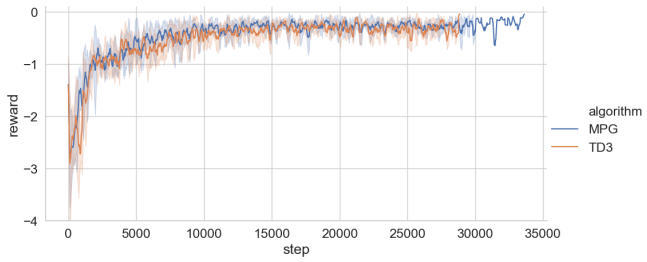


Fig. 8. MPG vs. TD3 rewards during training for multi-agent consensus.

For the purpose of comparison, an agent was also trained using TD3. The rewards per time step are shown in Figure 8. These were collected over 5 training runs of 200 episodes each. The MPG curves are longer than the TD3 curves, because the MPG networks avoid episode ending collisions for longer. Hence, MPG trained agents achieve the desired behavior sooner than the TD3 agents.

C. Obstacle avoidance

Based on the collision penalty $-k_c n_c$ embedded in Equation (14) of formation control, fixed or moving obstacle avoidance can naturally be integrated into the leader following or formation control problems. Instead of control law redesign, obstacle avoidance can easily be achieved by adding an additional term in the reward function.

The simulation setup is illustrated in Figure 9. The leader linearly travels from $(-1, -1)$ to $(1, 1)$. We have 2 fixed obstacles on the path of the leader that only stop the follower, and a moving obstacle travelling linearly from $(-1, 1)$ to $(1, -1)$. The follower, starting from a random location, must track the leader without hitting any obstacles. The reward

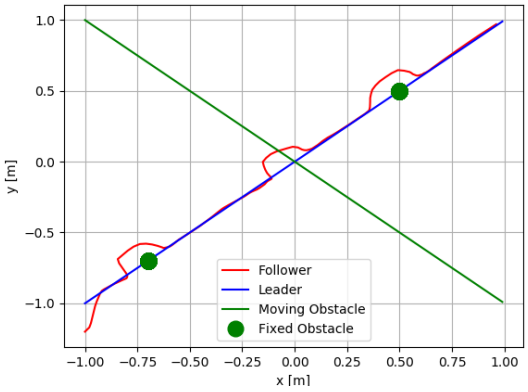


Fig. 9. Follower avoiding the moving and fixed obstacles.

function is

$$r = -\|p_l - p_f\| + k_o \sum_i \|p_f - o_i\| \tag{15}$$

where o_i is the position of the obstacles and k_o is the relative importance of avoiding the obstacles.

We use this reward, instead of a single penalty for colliding with an obstacle, because Equation (15) gives constant feedback. Training with sparse rewards is a big challenge in DRL [35]. In particular, because the obstacle is encountered later on, the follower learns to strictly copy the leader's movements without regard for the obstacles. This is a local optimum that the agent fails to escape. But for our purpose, it is not so important that the agent learns with even poorly shaped rewards. The realism of the training settings is irrelevant as the agent is already in a artificial and simplified environment.

VI. Conclusion and Future Work

In this paper, we propose a new DRL algorithm Momentum Policy Gradient to solve leader-follower tracking, formation control, and obstacle/collision avoidance problems, which are difficult to solve using traditional control methods. These controllers can be trained in a simple toy environment, and then plugged into a larger modular framework. The results show that MPG performs well in training agents to tackle a variety of continuous control tasks. Image-based localization is achieved with a ResNet trained on a

custom dataset. Analysis demonstrates that the model can reliably predict a WMR's pose even when there are dynamic foreground/background, lighting, and view.

These methods are computationally inexpensive. On a M40 Nvidia GPU and Intel Xeon E5 processor, MPG only takes, at most, 2 hours to fully converge. For localization, the CNN model took about 30 hours to finish training on a dataset of over 9 GB of images. This is because the ResNet-50 model has 50 residual blocks, compared to the 2 layers in our DRL agents.

In the future, we would like to apply MPG to a wider variety of robotics problems. In particular, we believe our current framework can generalize to larger indoor and outdoor settings. Some work has already been done in this field [16], [36], but there are still major challenges that have not been solved. One of these is data collection. Currently, we require many samples to train our localization module. However, using the techniques of few/one-shot learning [36] and data augmentation [16], it will no longer be necessary to collect so many samples at different times of day. This opens up the possibility of online learning as a robot explores a new area of the environment.

REFERENCES

[1] W. Dixon, D. Dawson, E. Zergeroglu, and F. Zhang, "Robust tracking and regulation control for mobile robots," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 10, no. 4, pp. 199–216, 2000.

[2] W. Ren, "Consensus tracking under directed interaction topologies: Algorithms and experiments," in *2008 American Control Conference*, pp. 742–747, IEEE, 2008.

[3] Z. Han, K. Guo, L. Xie, and Z. Lin, "Integrated relative localization and leader–follower formation control," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 20–34, 2019.

[4] N. Piasco, D. Sidibé, C. Demonceaux, and V. Gouet-Brunet, "A survey on visual-based localization: On the benefit of heterogeneous data," *Pattern Recognition*, vol. 74, pp. 90–109, 2018.

[5] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946, 2015.

[6] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *2016 IEEE international conference on Robotics and Automation (ICRA)*, pp. 4762–4769, IEEE, 2016.

[7] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu, "Indoor relocalization in challenging environments with dual-stream convolutional neural networks," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 651–662, 2018.

[8] Y. Lin, Z. Liu, J. Huang, C. Wang, G. Du, J. Bai, S. Lian, and B. Huang, "Deep global-relative networks for end-to-end 6-dof visual localization and odometry," *arXiv preprint arXiv:1812.07869*, 2018.

[9] F. Mutz, V. Cardoso, T. Teixeira, L. F. Jesus, M. A. Golçalves, R. Guidolini, J. Oliveira, C. Badue, and A. F. De Souza, "Following the leader using a tracking system based on pre-trained deep neural networks," in *Neural Networks (IJCNN), 2017 International Joint Conference on*, pp. 4332–4339, IEEE, 2017.

[10] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," tech. rep., CALIFORNIA INST OF TECH PASADENA CONTROL AND DYNAMICAL SYSTEMS, 2004.

[11] J.-H. Chin and H.-C. Tsai, "A path algorithm for robotic machining," *Robotics and computer-integrated manufacturing*, vol. 10, no. 3, pp. 185–198, 1993.

[12] W. E. Dixon, M. S. de Queiroz, D. M. Dawson, and T. J. Flynn, "Adaptive tracking and regulation of a wheeled mobile robot with controller/update law modularity," *IEEE Transactions on control systems technology*, vol. 12, no. 1, pp. 138–147, 2004.

[13] M. S. Fibla, U. Bernardet, and P. F. Verschure, "Allostatic control for robot behaviour regulation: An extension to path planning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1935–1942, IEEE, 2010.

[14] Y. Hong, G. Chen, and L. Bushnell, "Distributed observers design for leader-following control of multi-agent networks," *Automatica*, vol. 44, no. 3, pp. 846–850, 2008.

[15] X. Ge, Q.-L. Han, D. Ding, X.-M. Zhang, and B. Ning, "A survey on recent advances in distributed sampled-data cooperative control of multi-agent systems," *Neurocomputing*, vol. 275, pp. 1684–1701, 2018.

[16] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "Learning deployable navigation policies at kilometer scale from a single traversal," *arXiv preprint arXiv:1807.05211*, 2018.

[17] Q. Liu and Q. Hui, "The formation control of mobile autonomous multi-agent systems using deep reinforcement learning," *13th Annual IEEE International Systems Conference*, 2018.

[18] T. Krajník, F. Majer, L. Halodová, and T. Vintr, "Navigation without localisation: reliable teach and repeat based on the convergence theorem," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1657–1664, IEEE, 2018.

[19] H. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end to end," *CoRR*, vol. abs/1809.10124, 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[21] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[22] B. E. Ilon, "Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base," Apr. 8 1975. US Patent 3,876,255.

[23] R. L. Bryant, "Geometry of manifolds with special holonomy," *150 Years of Mathematics at Washington University in St. Louis: Sesquicentennial of Mathematics at Washington University, October 3-5, 2003, Washington University, St. Louis, Missouri*, vol. 395, p. 29, 2006.

[24] J. Weiner, "Asymmetric competition in plant populations," *Trends in ecology & evolution*, vol. 5, no. 11, pp. 360–364, 1990.

[25] G. Dudek and M. Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2010.

[26] N. Lee, S. Ahn, and D. Han, "Amid: Accurate magnetic indoor localization using deep learning," *Sensors*, vol. 18, no. 5, p. 1598, 2018.

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[28] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.

[29] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.

[30] S. Singh, T. Jaakkola, M. Littleman, and C. Szepesvári, "Convergence results for single-step on-policyreinforcement-learning algorithms," *Machine Learning*, vol. 38, pp. 287–308, 2000.

[31] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[32] F. Zegers, H.-Y. Chen, P. Deptula, and W. E. Dixon, "A switched systems approach to consensus of a distributed multi-agent system with intermittent communication," in *Proc. Am. Control Conf.*, 2019.

[33] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell, *et al.*, "Learning to navigate in cities without a map," in *Advances in Neural Information Processing Systems*, pp. 2424–2435, 2018.

[34] S. He, M. Wang, S.-L. Dai, and F. Luo, "Leader–follower formation control of usvs with prescribed performance and collision avoidance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 572–581, 2019.

[35] T. Salimans and R. Chen, "Learning montezuma's revenge from a single demonstration," *arXiv preprint arXiv:1812.03381*, 2018.

[36] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-shot reinforcement learning for robot navigation with interactive replay," *arXiv preprint arXiv:1711.10137*, 2017.