Modular Platooning and Formation Control

Yanlin Zhou [*1] **George Pu** [*2] **Fan Lu** [*1] **Xiyao Ma** [1] **Xiaolin Li** [1]

Abstract

Moving vehicles in formation, or platooning, can dramatically increase road capacity. While traditional control methods can manage fleets of vehicles, they do not address the issues of dynamic road conditions and scalability (*i.e.*, sophisticated control law redesign and physics modeling). We propose a modular framework that averts daunting retrains of an image-to-action neural network, provides flexibility in transferring to different robots/cars, while also being more transparent than previous approaches. First, a convolutional neural network was trained to localize in an indoor setting with dynamic foreground/background. Then, we design a new deep reinforcement learning algorithm named Momentum Policy Gradient (MPG) for continuous control tasks and prove its convergence. MPG is sucessfully applied to the platooning problem with obstacle avoidance and intra-group collision avoidance.

1. Introduction

Platooning, a major application of self-driving cars, has been extensively studied in the control community but has yet to draw much attention in the deep reinforcement learning (DRL) community. Platooning is when cars on a road move in a tight formation, such as a line, to increase road capacity. Previous traditional control methods apply constrained convex optimization to control a fleet of cars (Koshal et al., 2009; Gong et al., 2016; Koshal et al., 2011). Platooning can be divided into a formation control problem, where an autonomous vehicle trying to follow the movement of a leader object (Mutz et al., 2017), and obstacle avoidance problem, where a controller is expected to path around different obstacles or unreachable areas.

A common traditional control approach is to design an adaptive or hybrid controller by considering all the cases, which is time-consuming (Chin & Tsai, 1993; Dixon et al., 2004; Fibla et al., 2010). However, the non-linearity of non-holonomic robots adds to the challenges of modeling robots and multi-agent consensus. For instance, a wheeled mobile robot (WMR) is a nonholonomic system due to its constrained moving direction and speed. Controlling nonholonomic models usually involves differential drive derivation such as Instantaneous Center of Curvature (ICC) or physical modeling such as a kinematic bicycle model (Dudek & Jenkin, 2010; Rajamani, 2011; Ong & Gerdes, 2015; Kong et al., 2015).

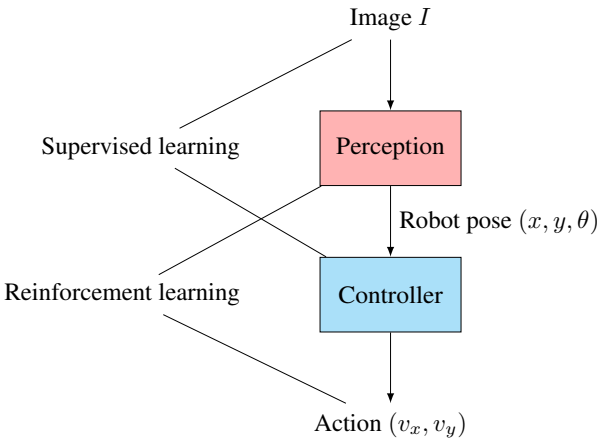


Figure 1. Information flow between modules. Blocks are neural networks.

Model-free DRL solves these problems, but training an agent requires complex environmental simulation and physics modeling of robots (Bruce et al., 2018; Liu & Hui, 2018). In this paper, we circumvent this challenge by splitting training with two neural networks: a perception module that estimates a WMR's location from images and a controller module that predicts the action needed to complete a task. While end-to-end training from images to action is popular (Krajník et al., 2018; Chiang et al., 2018), partition-

[*]Equal contribution [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 [2]Department of Computer and Information Science and Engineering, University of Florida, Gainesville, Florida, 32611-6250, USA. Correspondence to: Yanlin Zhou <zhou.y@ufl.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

ing the problem into two steps has a number of advantages. The need for complex simulation is avoided. There is the flexibility of retraining the controller while reusing the localization module, Module input/output is interpretable. A flowchart of our modular design is shown in Figure 1.

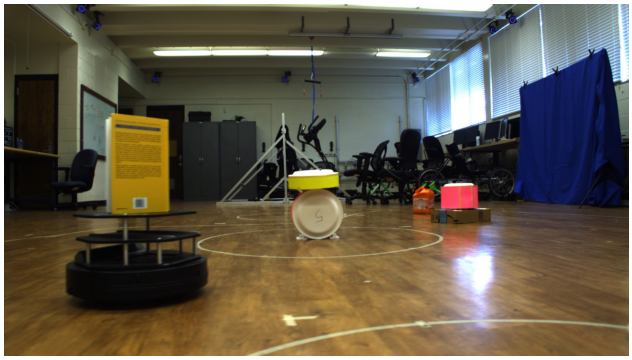


Figure 2. A screenshot from our custom dataset. Three lankmarks can be seen.

The first phase of this work was to train a residual network (ResNet) (He et al., 2015) to perform vision-based localization. We focus on indoor localization which includes many challenges such as a dynamic foreground/background, motion blur, and changing lighting. Various landmarks were placed in the environment, which could be used to determine a WMR's pose. 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 2. The model accurately predicts the position of robots, which enables DRL without need for a detailed 3D simulation. Instead, the controller module "sees" only the positions of itself and other WMRs.

To replace traditional controllers, we use a new actor-critic algorithm called Momentum Policy Gradient (MPG) to train a policy network. MPG is an improvement TD3 that reduces under/overestimation of the value function (Fujimoto et al., 2018). MPG's convergence is theoretically proven and stability shown experimentally. The proposed algorithm is efficient at solving the platooning problem.

Our contributions can be summarized as follows.

1. A modular approach designed to circumvent the need for extensive simulation of the real world.

2. The Momentum Policy Gradient DRL algorithm for continuous control tasks that combats value under/overestimation.

3. An application to a the real world problem of platooning and formation control with obstacle avoidance.

2. Mobility of Wheeled Robots

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 (*i.e.,* omni-directional Mecanum wheel robots) (Ilon, 1975) and nonholonomic agents (*i.e.,* real vehicles with constrained moving direction and speed) (Bryant, 2006). In this paper, we consider the case of nonholonomic agents.

Ideally, angular and linear velocities are the outputs of a DRL trained policy network. However, these two action spaces have very different scales and usually cause size-asymmetric competition (Weiner, 1990). We discovered that training a DRL agent with angular and linear velocities as actions converges slower than the methods presented in later pages. 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 (Ren, 2008).

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

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

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 $x_i' = x_i + d_i \cos\theta, y_i' = y_i + d_i \sin\theta$.

Using $d = 0.15$ meters in (Ren, 2008), it is trivial to transfer x and y direction velocity signals to the actions used in nonholonomic system control.

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

Here $a_{x_i'}$ and $a_{y_i'}$ are the input control signals for 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 (Dudek & Jenkin, 2010). Meanwhile, decomposing velocity dynamics to velocities on the x and y axes can be reasonably applied to any WMR.

3. Modular Design

By splitting a 3D locomotion task into a localization and control problem, which are solved by separate neural networks, is the crux of this paper. Though image-to-action

networks are still commonplace, there has been a recent shift towards multi-part DRL networks (Banino et al., 2018; Wayne et al., 2018). Our approach has similarities to (Ha & Schmidhuber, 2018), except we require that inputs/outputs from models be human interpretable. This gives greater flexibility in replacing or adding new modules.

This modular solution offers a number of advantages over an end-to-end system trained completely using reinforcement learning.

1. **Avoids 3D simulations**: Solving real-world problems using reinforcement learning requires environments that mimic the real world. However, complex 3D simulations are expensive to create and run. Despite the efforts of organizations like OpenAI (Brockman et al., 2016), researchers looking to explore more specialized domains have no choice but to create their own environments. However, because of our modular approach, we avoid this problem. Modules can be trained using simpler simulations, without need for realistic graphics.

2. **Re-training**: If and end-to-end system is found defective or needs upgrades, the entire model must be retrained. Depending on the scenario, this retraining can take weeks of compute time on very powerful GPUs. This prohibits regular but minor changes to deep neural networks, a practice which is commonplace in software engineering.

   However, using a modular design means that, if one module needs retraining, the others do not need to be discarded. Because modules tends to be small, retraining is easy and the new model can be sent to edge devices as a small software update.

3. **Transparency**: A big problem with deep neural networks is their opacity. They are envisioned as black boxes. With our design, the inputs and outputs between modules is interpretable by humans. Moreover, each unit can be independently tested and verified, allowing flaws to be traced back to a particular module.

   This transparency also allows traditional control methods to be integrated with deep neural networks. Trained models are a piece of software, and should be interoperable with other software.

The emphasis on human readability may lead to unnecessary encoding and decoding of inputs/outputs. Feature reuse would likely save on computation time and model size. But we believe the benefits are significant enough to merit this choice.

4. Localization

The perception module focuses on the problem of estimating position directly from images in a noisy environment. Several landmarks (*i.e.,* books, other robots) were placed in the environment 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 (Lee et al., 2018). As the WMR moves around an environment, a mounted camera observes only a subset of the landmarks. Depending on the position and orientation of the robot, the placement of the landmarks within the frame changes.

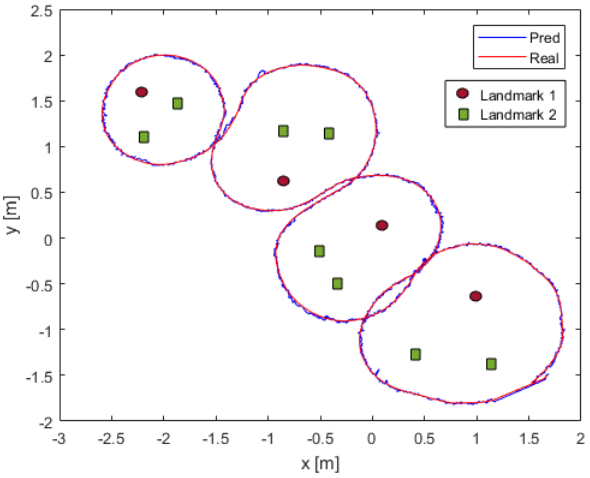
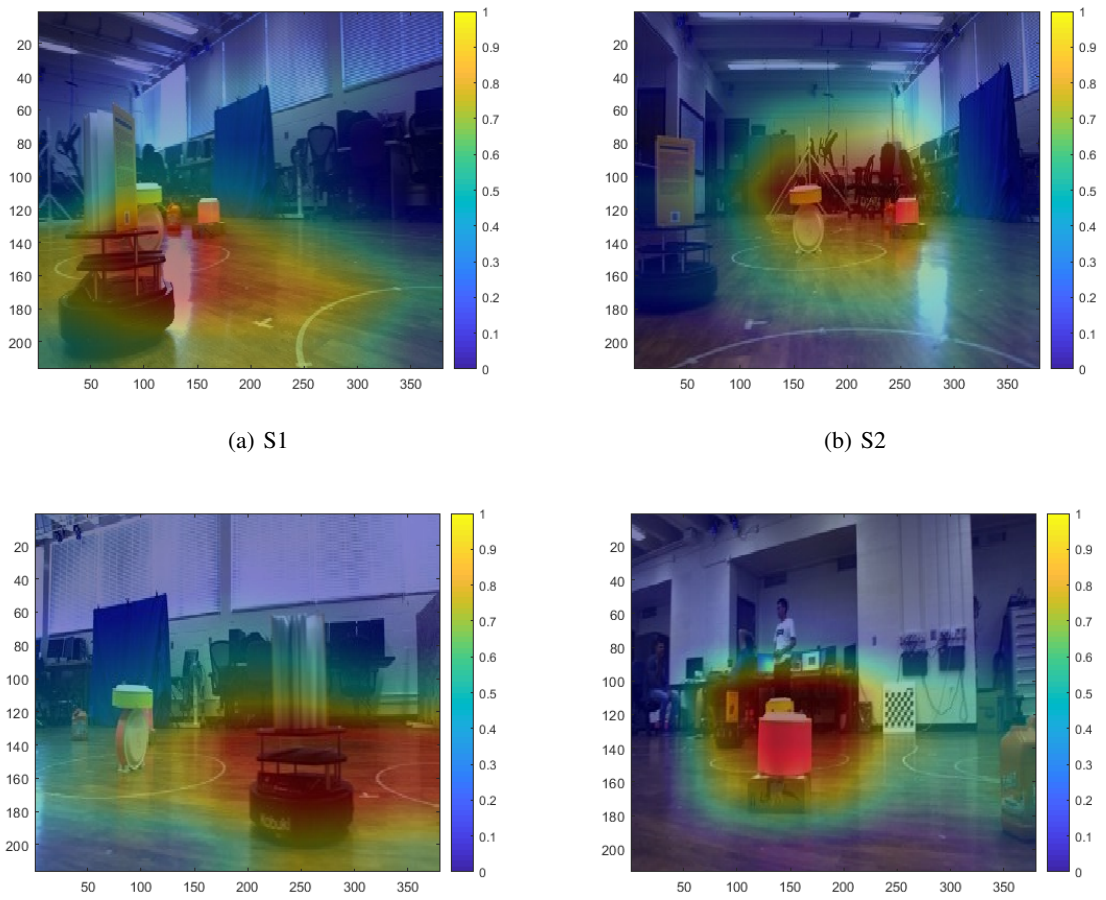


Figure 3. Example snail trajectory with predicted pose.

Overall, data was gathered from 3 types of robot trajectories (regular, random, whirlpool) with multiple collections at different times of day (morning, afternoon, evening). Data collection was performed at Nonlinear Controls and Robotics Lab. Not only do the lighting conditions change between examples, but the background varies from human activity throughout the day. The vision module must learn to ignore this noise and rely on the landmarks for localization.

The images and ground truth pose of the agent were 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. 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 2.

A ResNet-52 model (He et al., 2015) with a resized output layer is trained from scratch on the dataset. Residual networks contain connections between layers of different depth to improve gradient flow during backpropagation. This eliminates the vanishing gradient problem encountered when training very deep convolutional neural networks. The



(a) S1 (b) S2

(c) S3 (d) S4

Figure 4. Saliency maps showing the effect of each pixel on the prediction accuracy.

ResNet predicts the robots current the 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.

Our approach is robust enough to accurately predict a robot's pose for even very complex trajectories, even ones with 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. Error rates for the robot's 2D coordinates and 4D quaternion orientation are given in Table 1. 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.

To visualize what our ResNet has learned from raw images, we create saliency maps of four sample images as shown in Figure 4. Since the ResNet directly outputs the (x, y) coordinates of the WMR rather than a discrete set of probabilities, the saliency maps are thus created by measuring the accuracy when sliding a 3x3 occluding window on the raw image. As illustrated in Figure 4, landmarks are highlighted while other parts of the images are occluded, such as people walking around, indicating that the ResNet learns to focus on the landmarks which give the TurtleBot reliable position information, while ignoring background noise.

Table 1. ResNet52 Pose Prediction Error.

X (%)	Y (%)	Q1 (%)	Q2 (%)	Q3 (%)	Q4 (%)
0.439	0.1191	0.7462	0.3199	0.1673	0.152

5. Momentum Policy Gradient

Since nonholonomic controllers have a continuous action space, we design our algorithm using the framework established by DDPG (Lillicrap et al., 2015). 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 (François-Lavet et al., 2018). TD3 fixes this by using two Q-value estimators $Q_{\theta_1}, Q_{\theta_2}$ and taking the lesser of the two (Fujimoto et al., 2018).

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

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 (Fujimoto et al., 2018).

To lower the variance in the estimate, inspired by the momentum used for optimization in (Zhang et al., 2015), 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 $-\infty$). The convergence of MPG update rule is provided in Theorem 1. The proof can be found in the Appendix.

Algorithm 1 Momentum Policy Gradient

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, 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 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

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 Policy Gradient converges to the optimal value function Q^*.

6. Continuous Control

We conducted a variety of experiments designed to replicate real-world indoor and outdoor robotics tasks. In all

experiments, 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 position Table 2 are enforced by ending episodes once they are breached. A penalty of $-k_b$ is also added to the reward for that time step. The hyperparameters of MPG are given in Table 5.

Table 2. 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

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 (Oh et al., 2015):

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

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

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

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 (Zegers et al., 2019; Han et al., 2019).

6.1. Formation Control

The most basic type of formation control considered is the leader-follower task: one agent—the follower—must move to the location of another agent—the leader. The leader constantly moves without waiting for the follower. We can extend the single leader-follower problem by adding additional trained followers to track the sole leader.

We first design and conduct an experiment similar to (He et al., 2019). There is a pre-defined formation $\mathcal{F}$ with respect to the global frame. All agents maintain a rigid formation throughout the entire movement process. The orientation of the formation does not change. A neural network learns to move the followers in unison with the leader, hence the name *Unison Formation Control*.

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. The reward is

$$r = -k_c n_c - \sum_{i \geq 1} \|p_i - \mathcal{F}_i\| \tag{11}$$

where k_c is collision coefficient and n_c is the number of collisions and p_i is the position of agent i. A collision occurs once the distance between two agents drops below a certain threshold C. Upon any collisions, we reset the environment and start a new episode.

We explored Unison Formation Control for a square formation with curved and random leader movements. As seen in Figure 5, three followers move in unison equally well when tracking a leader with smooth trajectory starting from lower left corner (a) and a random leader (b). The average reward and distances are reported in Table 3.

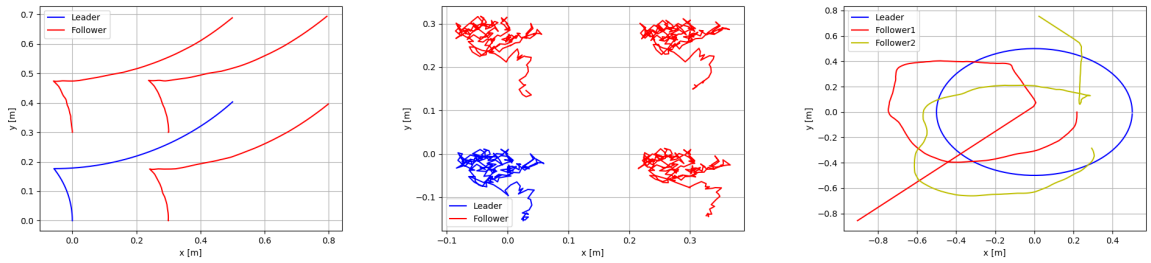
Table 3. Unison Average Reward and Distances.

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

However, for many applications, Unison Formation Control is overly restrictive as it dictates the individual motion of all agents. *Consensus Formation Control* requires that agents keep 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. The extra flexibility 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 $\mathcal{D}_{i,j}$ between each other. Letting $d_{i,j} = \|p_i - p_j\|$, the reward function is given according to the following equation.

$$r = -k_c n_c - \sum_{i,j} |d_{i,j} - \mathcal{D}_{i,j}| \tag{12}$$

In our experiment, we trained two follower agents to maintain triangle formation with a leader undergoing random motion. As shown in Figure 5 (c), we test the performance of the controller network at multi-agent consensus while the leader traverses 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. Initially, the three agents start far away from each other but quickly formed a triangle when the leader reaches around $(0.1, 0.5)$. We observed that the followers swapped their local positions within the formation when the leader arrives $(-0.5, 0.0)$. This is because, as the reward function is only interested in relative distances, the policy network learns that the agents can maintain the formation with less



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

Figure 5. Formation control: (a) and (b) use the unison definition while (c) allows the followers to swap positions.

total movement by switching their relative positions in the group. Results are reported in Table 4.

Table 4. 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

6.2. Platooning

While platooning of a truck fleet can be considered a special case of Consensus Formation Control, where the formation is a straight line, due to the presence of obstacles such as pedestrians, cars, and even trash on the roadway, agents must be able to adjust their formation on the fly to evade potentially dangerous objects. Instead of control law redesign, fixed or moving obstacle avoidance can naturally be integrated into the formation control problem by adding an additional term in the reward function.

Depending on the number of vehicles in a fleet, there are two options for platooning control. In *Distributed Platooning*, each vehicle has its own neural network as a control module. In *Centralized Platooning*, one DRL actor controls all vehicles.

The modular implementation of perception and control functionality makes it possible to equip each vehicle in a large fleet with customized DRL controllers. For instance, the perception module can be used for localization among all cars. Based on the physical characteristics of each vehicle, neural networks can be trained accordingly to control each car. On the other hand, using a single DRL controller allows for better overall coordination and saves computation resources for small fleets. For this reason, only Centralized Platooning is investigated in this work.

The experiment setup is shown in Figure 6. The leader linearly travels from $(-1, -1)$ to $(1, 1)$. There are 2 fixed obstacles on the path of the leader that only stop the followers, and a moving obstacle travelling linearly from $(-1, 1)$ to $(1, -1)$. There are two followers: one must main a certain distance from the leader without hitting any obstacles, the other must track the first follower without hitting any obstacles or the first follower.

We adopt the reward from Equation 12, but add additional terms that punish closeness between the followers and the obstacles. Agents i, j must maintain a distance $\mathcal{D}_{i,j}$ from each other, with $i = 1$ being the index of the leader.

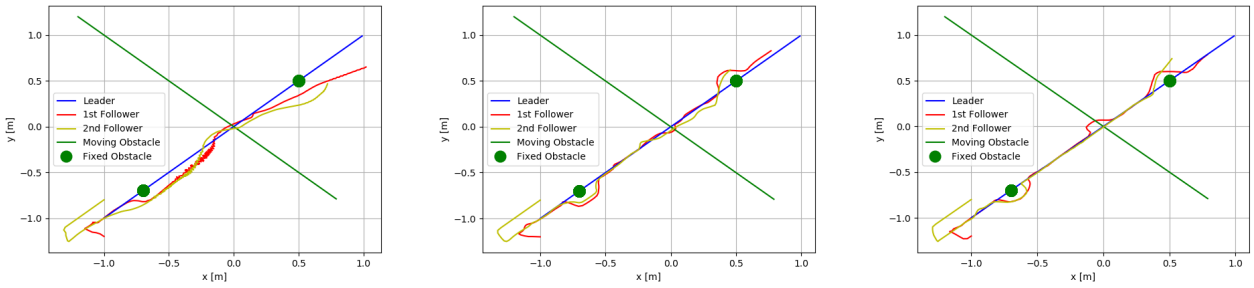
$$r = -k_c n_c - \sum_{i,j} |d_{i,j} - \mathcal{D}_{i,j}| + k_o \sum_{i>1,j} \|p_i - o_j\| \quad (13)$$

Here 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 one-time penalty for colliding with an obstacle, because Equation (13) gives constant feedback. Training on sparse rewards is an unsolved challenge in DRL (Salimans & Chen, 2018). In particular, because the obstacle is encountered later on, the follower learns to strictly copy the leader's movements without regard for obstacles. Once the obstacles are encountered, the follower maintains its previously learned behavior. But because we are only training a single module, it is not so important that the agent learns with even poorly shaped rewards. The realism of the training settings is irrelevant as long as the agent learns the desired behavior.

The result is shown in Figure 6 (a). Although both followers roughly form a platoon with leader after 200 episodes of training, the formation is rather loose which may not be useful for a real road environment. Note that both followers display a clear zig-zag movement around $(-0.25, -0.3)$, when the first follower is closer to first fixed obstacle and the moving obstacle. We conjecture that this issue arises due to various terms in the reward function fighting with each other that causes the oscillation in action output. As a result, the followers do not move smoothly.

One solution to the zig-zaging is to add an additional term in



(a) Platooning (b) Platooning with velocity constraint. (c) Platooning with reward shaping

Figure 6. Platooning with different methods

reward function that discourages large changes in velocity. The updated reward function is then

$$r = -\|a_t - a_{t-1}\| - k_c n_c - \sum_{i,j} |d_{i,j} - \mathcal{D}_{i,j}| + k_o \sum_{i>1,j} \|p_i - o_j\| \tag{14}$$

where a_t is the current action taken and a_{t-1} is the previous action. As shown in Figure 6 (b), the followers manage to stay on the path of the leader while still avoiding the obstacles. However, this requires that the agent see its last action, which increases the state dimension by 2.

Alternatively, we can adjusted the coefficients k_c, k_o in Equation (13) so that the followers are rewarded greater for keeping the platoon moving along a straight line. The simple weight change worked surprisingly well, with the results shown in Figure 6 (c). Through reward shaping, the controller learns the desired behavior.

7. Conclusion and Future Work

In conclusion, we propose Momentum Policy Gradient—a new DRL algorithm—to solve formation control and obstacle/collision avoidance problems central to platooning, which are difficult to solve using traditional control methods. The experimental results show that MPG performs well in training agents to tackle a variety of continuous control tasks. These controllers can be trained in a simple toy environment and then plugged into a larger modular framework. We also achieve image-based localization with a ResNet-52 network trained on a custom dataset. Analysis demonstrates that the model can reliably predict a WMR's pose in spite of a dynamic background, varied lighting, and changing view.

There are still many hurdles that this paper was not able to resolve. Our DRL controllers are trained without regard for friction or acceleration. Even using on-board computers and control algorithms to convert x, y-velocity into the appropriate motor signals, the current controllers would struggle with complications such as slipping, delays in executing actions, and vehicle performance constraints. One solution is to improve the virtual environment with better physics modeling, such as using acceleration as the actions instead of velocity. However, it is impossible to perfectly mimic the wide variety of real-world conditions. More work is needed in developing DRL algorithms which can train models that generalize to unfamiliar environments. However, we believe the proposed modular framework can eventually be expanded to solve these challenges.

There are many other directions that merit future work. The perception module can be extended to outdoor localization by integrating multiple sources of sensory information like LiDAR and GPS. Each sensor could have its own neural network or one neural network that combines different streams of information. Orthogonal to the goal of sensor fusion is Distributed Group Platooning which is faster but requires more computation power compared to Centralized Group Platooning. Importantly, Distributed Platooning scales better to large fleets as it does not require global communication between vehicles. A platoon can be controlled by multiple DRL policies, each of which collects position information of corresponding vehicles takes control actions.

References

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705): 429, 2018.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Bruce, J., Sünderhauf, N., Mirowski, P., Hadsell, R., and Milford, M. Learning deployable navigation policies at

kilometer scale from a single traversal. *arXiv preprint arXiv:1807.05211*, 2018.

Bryant, R. L. 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*, 395:29, 2006.

Chiang, H. L., Faust, A., Fiser, M., and Francis, A. Learning navigation behaviors end to end. *CoRR*, abs/1809.10124, 2018. URL http://arxiv.org/abs/1809.10124.

Chin, J.-H. and Tsai, H.-C. A path algorithm for robotic machining. *Robotics and computer-integrated manufacturing*, 10(3):185–198, 1993.

Dixon, W. E., de Queiroz, M. S., Dawson, D. M., and Flynn, T. J. Adaptive tracking and regulation of a wheeled mobile robot with controller/update law modularity. *IEEE Transactions on control systems technology*, 12(1):138–147, 2004.

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

Fibla, M. S., Bernardet, U., and Verschure, P. F. 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.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

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

Gong, S., Shen, J., and Du, L. Constrained optimization and distributed computation based car following control of a connected and autonomous vehicle platoon. *Transportation Research Part B: Methodological*, 94:314–334, 2016.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Han, Z., Guo, K., Xie, L., and Lin, Z. Integrated relative localization and leader–follower formation control. *IEEE Transactions on Automatic Control*, 64(1):20–34, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

He, S., Wang, M., Dai, S.-L., and Luo, F. Leader–follower formation control of usvs with prescribed performance and collision avoidance. *IEEE Transactions on Industrial Informatics*, 15(1):572–581, 2019.

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

Kong, J., Pfeiffer, M., Schildbach, G., and Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099. IEEE, 2015.

Koshal, J., Nedić, A., and Shanbhag, U. V. Distributed multiuser optimization: Algorithms and error analysis. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 4372–4377. IEEE, 2009.

Koshal, J., Nedić, A., and Shanbhag, U. V. Multiuser optimization: Distributed algorithms and error analysis. *SIAM Journal on Optimization*, 21(3):1046–1081, 2011.

Krajník, T., Majer, F., Halodová, L., and Vintr, T. 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.

Lee, N., Ahn, S., and Han, D. Amid: Accurate magnetic indoor localization using deep learning. *Sensors*, 18(5): 1598, 2018.

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

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

Mutz, F., Cardoso, V., Teixeira, T., Jesus, L. F., Golçalves, M. A., Guidolini, R., Oliveira, J., Badue, C., and De Souza, A. F. 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.

Oh, K.-K., Park, M.-C., and Ahn, H.-S. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

Ong, H. Y. and Gerdes, J. C. Cooperative collision avoidance via proximal message passing. In *2015 American Control Conference (ACC)*, pp. 4124–4130. IEEE, 2015.

Rajamani, R. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

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

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

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

Wayne, G., Hung, C.-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

Weiner, J. Asymmetric competition in plant populations. *Trends in ecology & evolution*, 5(11):360–364, 1990.

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

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

A. Proof of Theorem 1

The theorem and proof of MPG's convergence borrows heavily from those for Clipped Double Q-learning (Fujimoto et al., 2018). Before proving the convergence of our algorithm, we first require a lemma proved in (Singh et al., 2000).

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* $\sum_t \alpha_t^2(x_t)$ *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 proof is based on the proof of Theorem 1 found in (Fujimoto et al., 2018).

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

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

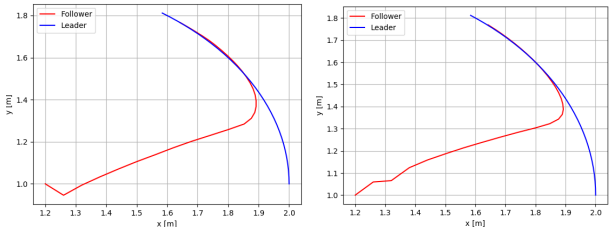
so $b_t \to 0$. Therefore showing $\Delta'_t(s_t, a_t) \to 0$ proves that b_t converges to 0.

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

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. □

B. MPG vs. TD3 Comparison

B.1. Leader-Follower



(a) TD3 (b) MPG

Figure 7. Performance of TD3 and MPG followers trained with identical hyperparameters at 200 episodes.

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

$$r = -\|p_l - p_f\| \tag{15}$$

where $\|\cdot\|$ is the L2 norm. We experimented with several trajectory types: circle, square, and random.

For the sake of comparison, we trained several followers using TD3 and MPG. An example reward is displayed in Figure 8. The values are smoothed using a 1D box filter of size 200. For the circle leader task, TD3 which struggles to close the loop, slowly drifiting away from the leader as time progresses. The MPG trained agent does not suffer from this problem.

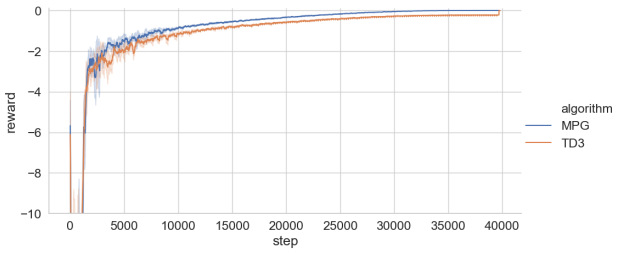


Figure 8. MPG vs. TD3 rewards during training for the circle leader-follower task.

We also noticed that some TD3 trained followers do not move smoothly. This is demonstrated in Figure 7. 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.2. Consensus Formation Control

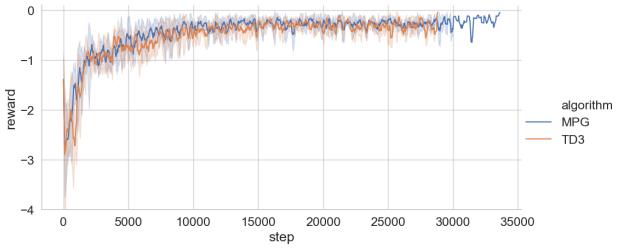


Figure 9. MPG vs. TD3 rewards during training for multi-agent consensus.

For the purpose of comparison, TD3 was used to train a neural network to perform consensus formation control. The rewards per time step are shown in Figure 9. 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. Hyperparamters

Table 5. MPG 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