LEARNING LOCOMOTION FOR QUADRUPED ROBOT BY APPLYING IMITATION AND CURRICULUM ON REINFORCEMENT LEARNING

XIATAO SUN [SXT@SEAS], ZHONGCHUAN XU [ZCXU@SEAS]

ABSTRACT. We implement a reinforcement learning model for quadruped robot locomotion in this project. Learning-based controllers with some traditional approaches, such as MPC and LQR, are compared in this project. The implementation only uses imitation learning with PPO in the beginning. As the initial trained model faces reward sparsity, we introduce a 3-stage curriculum to address this issue. The trained model is then tested on four different unseen terrains. The reward data generated when testing is recorded and analyzed.

1. INTRODUCTION

Legged robots have better performance overall on challenging terrains than tracked and wheeled robots, thanks to their structural advantages. Among legged robots, the optimal choice considering stability, mobility and ease of control is the quadruped robot [16]. Trajectory optimization and traditional model-based controllers have been widely applied for controlling legged robots. Improvements in numerical methods and models keep making progress on handling the optimization[7]. However, with limited computational resources, the traditional methods are brutal to respond to large perturbations or handle high dynamic motions [24].

This project aims to create a robust quadruped robot controller using a data-driven, physics-based approach by utilizing reinforcement learning (RL) techniques. Unlike the traditional method, our controller was trained in a physics-based dynamic environment with demonstrations generated by reference gait cycles in multiple learning cycles. The controller is trained based on a 3-stage curriculum. It uses observations given by the simulation environment, high-level user input and estimations of the robot's attitude through reward functions to learn the optimal control policy in response to the feedback. Through Deep Reinforcement Learning (DRL), the controller gains the ability to adapt and recover from unseen environments and scenarios.

1.1. Contributions.

The purpose of the project is for us to explore the techniques of developing a robust controller for quadruped robots through RL. In this project, we implemented

- a physics-based controller for quadruped robot that is trained by applying imitation and curriculum on DRL that can adapt to unseen terrains with various surface properties
- while being able to respond to high-level navigation inputs for easy high-level navigation module integration

2. BACKGROUND

A quadruped robot refers to a mobile robot with four legs. It has a unique advantage compared to wheeled and tracked robots due to its excellent exploration ability on various terrains, which is from the biological imitation of structures of quadruped animals and insects. While keeping the stability, the quadruped is also easy to be developed and controlled compared with robots that have more legs [16]. Many commercial quadruped robots are available, such as Spot from Boston Dynamics, which let us be interested in implementing the controller for quadruped agents.

The environment of the quadruped robot was built using Unity, which is a game and physics simulation engine. This project primarily used Unity ML-Agents with PyTorch for training the intelligent agent[1]. The specific quadruped robot chosen for the training is Spot for its good real-world performance and availability [6].

The implementation of this project primarily uses Proximal Policy Optimization (PPO), a policy-based RL algorithm where policy is updated explicitly in the policy gradient method. Compared to the vanilla policy gradient method, PPO addresses the convergence problem and the scalability problem of natural policy gradient [10]. PPO performs well while being simple to implement and tune. Imitation learning and curriculum learning are also implemented on top of PPO to alleviate the problem of reward sparsity.

3. Related Work

Motions are initially computed by trajectory optimization, which utilizes numerical optimization. This method aims at minimizing defined objectives and considering nonlinear dynamics. However, the intense computation required by this method makes online implementation hard to achieve[17].

Model Predictive Control (MPC) can be used as an online motion generation scheme mostly for relative motion between the contact point and the center of mass. MPC relates to the artificial synergy synthesis approach, which manages the dynamic constraints by assignment of degrees of freedom, allowing the relative Independence of the remaining. Many modern robot control can be traced to MPC. It can be robust and versatile [17], but still requires considerable computation power, and it relies on modeling the dynamics of the environment, which often requires human expertise.

Linear Quadratic Regulators (LQR) are similar to MPC as they are both expressions of optimal control, but they are different from schemes of setting up optimal cost. The length of the time window differentiates MPC and LQR[12]. MPC uses a moving time window that is limited and relatively small to mitigate the effect of changing environment on the optimality of the solution. In contrast, LQR considers the entire time window[19]. LQR presents some advantages to MPC but suffers from the assumption of linearity and many drawbacks of model-based controllers as MPC.

In a computation resource-constrained environment, Central Pattern Generators (CPG) are often used to take elementary control inputs, including angle and speed, and create synchronized quasi-cyclic motion patterns. Hopf oscillators have been used to introduce CPG in legged robots. However, essentially the CPG belongs to open-loop motion generators. Although a feedback loop is often implemented, it is weak in response to large perturbation[17]. Although it has drawbacks, it is helpful to generate a demonstration sequence for the imitation learning phase. We used a simple harmonic oscillator to generate the gait cycle for the robot.

Physics-based, model-free methods with DRL mitigated many of the drawbacks above. It has been shown that DRL performs well with good versatility and robustness in various problems with a continuous action space, including robot control problems in the real world. However, RL suffers from the problem of reward sparsity[23]. This project implemented curriculum learning and imitation learning using behavior cloning to mitigate this issue.

4. Approach

Although our final approach is applying imitation learning and curriculum learning on RL, initially, the approach is only to directly train the model using behavior cloning and the PPO algorithm. The demonstration for behavior cloning is generated using a simple harmonic oscillator[18]. The differential equation for the harmonic motion applied on each leg is shown as the equation below, where A is the amplitude, ω is the angular velocity of the joint, and t is the time:

$$x(t) = Acos(\omega t)$$

When recording the demonstration, the lower and upper joints on each leg are fixed, and only the shoulder joint is allowed to move. Although plenty of literature uses the Hopf oscillator to generate demonstration, the harmonic oscillator is adequate in our scenario since the motion is simplified to one degree of freedom on each leg.

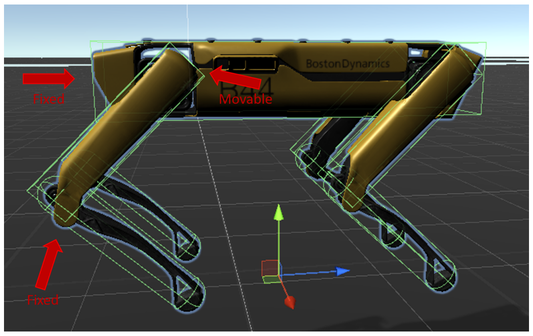


FIGURE 1. Joints

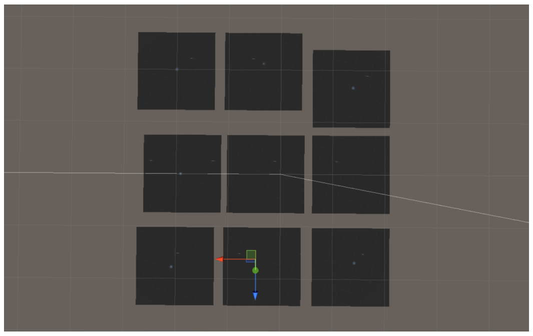


FIGURE 2. Training Scene

The rewards are set up with inspiration from [26], where s is the tunable strength parameter and p is the tunable exponential parameter:

- Inclination Reward, where h_0 is the robot height, and h is the current height of the robot:
$$R_i = s_i(h_0 - 0.5h)^{p_i}$$
- Height Reward, where h is the current robot height, and $\bar{h}$ is the average robot height:
$$R_h = s_h(1 - \left|\bar{h} - h\right|)^{p_h}$$
- Walking Direction Reward, where θ_w is the walking direction of the robot:
$$R_w = s_w(1 - (\theta_w - \theta_{target}))^{p_w}$$
- Looking Direction Reward, where θ_w is the looking direction of the robot, we want to robot to look ahead:
$$R_l = s_l(1 - \theta_l)^{p_l}$$
- Speed Punishment, where v is the velocity, return the normalized velocity:
$$R_v = -s_v||v - v_{target}||$$

The environment is a flat terrain for the agent. Speed, walk and look directions are the three inputs to the agent. During training, the agent is given a set of random inputs every 10 seconds. As Figure 2 shows, nine agents are placed and trained on separate terrain in the environment simultaneously to speed up the training process.

However, the trained model from this approach is unusable on the quadruped robot in the simulation environment. The unusable model might result from reward sparsity, which is an inherent problem in RL. Although imitation learning is already a mitigation to this problem, additional countermeasures need to be implemented.

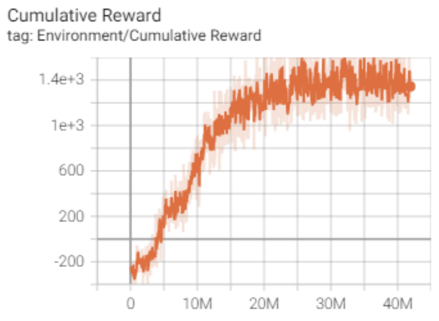
Therefore, we adjust the approach and divide it into a 3-stage curriculum to guide the training process inspired by [11]. The rewards are the same as the previous approach for all three stages. The purpose of the first stage is to let the agent learn from the demonstration. In the first stage, the environment is the flat terrain that is the same as the previous approach. The agent is controlled by fixed speed, walk and look directions. Behavior cloning is applied using the demonstration generated from the oscillator with 0.5 strength. The strength of the extrinsic reward signal is set to 0.1. The first stage is executed for 10 million steps to learn from the demonstration while avoiding overfitting. The purpose of the second stage is to adapt to dynamic speed. In the second stage, the environment is still flat terrain. The demonstration is removed, and the strength of the extrinsic reward is set to 1. The agent is trained for 25 million steps in this stage. The purpose of the third stage is to adapt to different terrain structures and dynamic directions. In the third stage, the terrain is now more challenging and unstable instead of the previous flat terrain. Moreover, the input speed, walk and look direction are random variables that refresh every 10 seconds. The agent is trained for 30 million steps in this stage.

In this project, all the training sessions are performed on PC specs of i7 and RTX 3080 with 16GB VRAM.

5. Experimental Results

As we mentioned in the previous section, initially, the model is trained using demonstration, rewards, and other conditions altogether. Although training the model using the initial approach looks good from the Tensorboard plots, such as the plot of cumulative reward during training in Figure 3, when testing it in the environment after training, the quadruped robot is not even able to move. As the plot of testing rewards at each step in Figure 3 shows, the agent receives a nearly constant negative reward on speed, indicating that it does not move.

(A) Cumulative Reward During Training



(B) Testing Rewards in Each Step

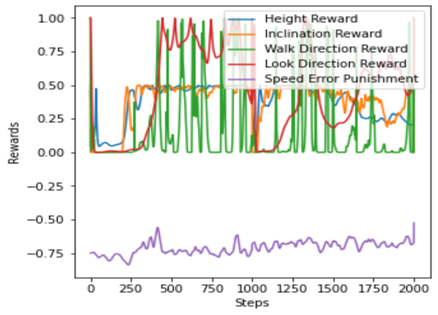
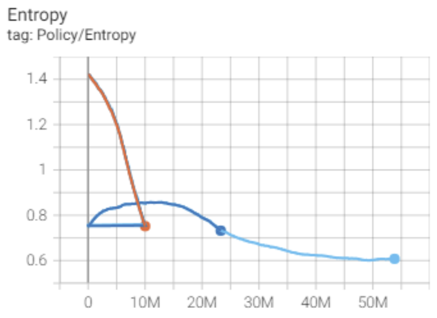


FIGURE 3. The Result of Initial Approach

Therefore, we modify our approach, and Figure 4 shows plots from Tensorboard indicating that the training is overall successful.

(A) Entropy of the Policy During Training



(B) Cumulative Reward During Training

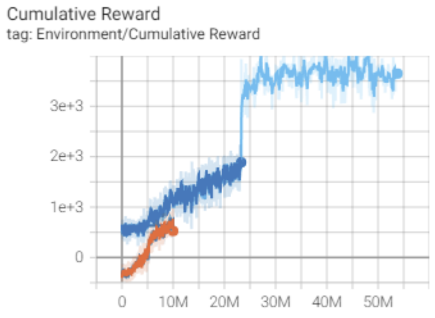
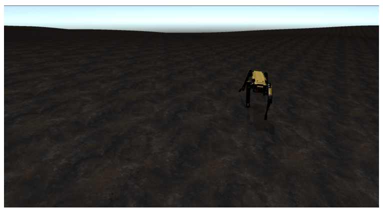


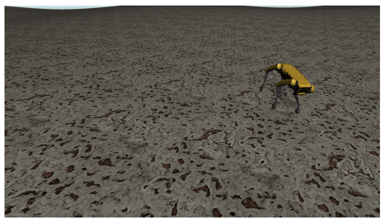
FIGURE 4. Showing the 3 - Stages Training Result

The model was trained for a total of 65 million steps. Notice the plot has an error as the starting point of the second stage should be at 10 million. The first 10 million steps were trained with behaviour cloning on the demonstration sequence generated by the harmonic oscillator. The following 25 million steps were trained by removing the behavior cloning and adding the high-level input of dynamic speed. The last 30 million steps were trained by adding randomized directions and more challenging terrain. The cumulative reward defined by the aforementioned reward functions converges at around 3.5e+3. Much higher than the cumulative reward of a single-stage training using behavior cloning and PPO algorithm at 1.4e+3. The entropy of the policy also decreases drastically during imitation learning with a bit of exploration. Entropy increases a bit during the second stage due to the removal of the imitation and the addition of more exploration, and it drops further at the end of the second and third stages. Both metrics indicate that the training is successful. However, in the cumulative reward, the reward increases a lot during the first two stages, but it plateaus in the third stage. It could indicate that the terrain is not challenging enough in terms of surface friction and roughness or the randomized high-level direction input is not difficult for the controller.

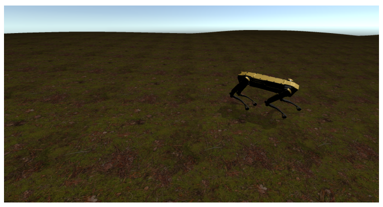
(A) Rock



(B) Sand



(C) Moss



(D) Snow

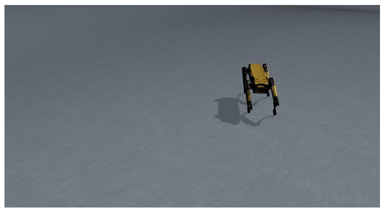


FIGURE 5. Screenshots of the Robot Using Trained Controller on Different Terrains

After the training, the model was tested in four terrains with different combinations of surface friction and roughness. The testing environments are rock, moss, snow, and sand, representing high friction challenging terrain, low friction challenging terrain, low friction flat terrain, and high friction flat terrain. The reward data from all testing environments are recorded and analyzed using Pandas. The robot can walk with a similar level of versatility as in the training environment, showing a suitable generalization of the model in an unseen environment. By comparison, the robot performs best on rock, which is high friction and challenging terrain.

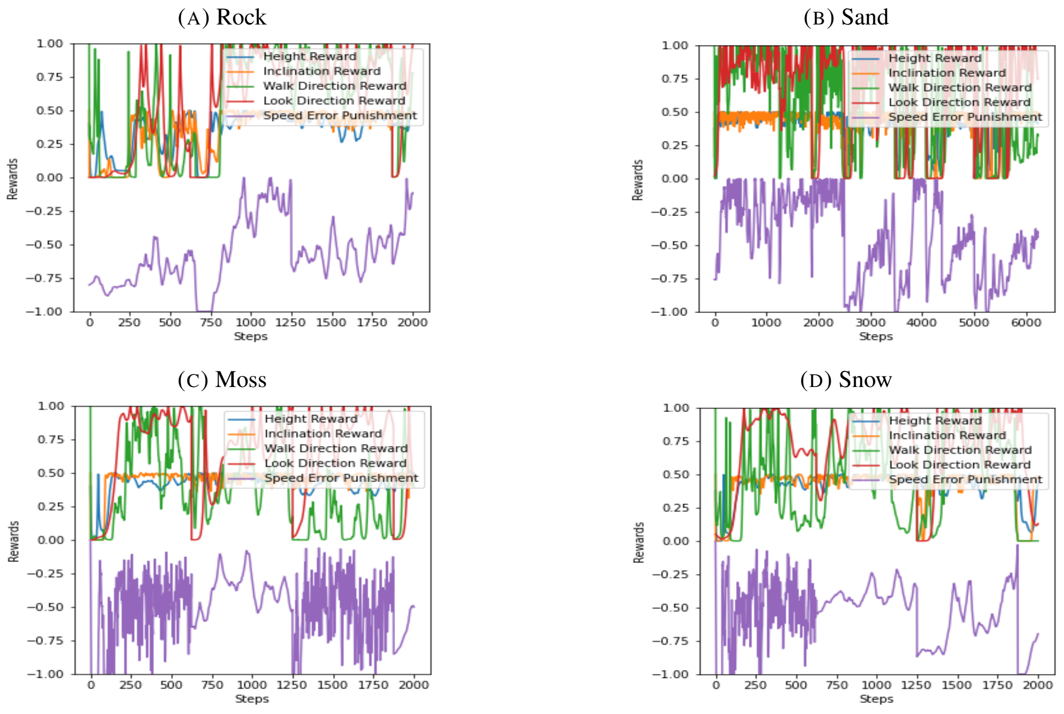


FIGURE 6. Rewards at Each Steps On Different Testing Terrains

6. DISCUSSION

The initial attempt is to directly train the model using behavior cloning and the PPO algorithm without multi-stage training. The constant negative reward on speed indicates that the trained model using the initial approach cannot move. The poor performance is likely the cause of the sparse reward of the DRL. That is made worse by mixing up high-level control inputs and many reward functions simultaneously for a single-stage learning process. Separating the learning into different stages and introducing high-level control one by one mitigates the issue and eventually gives a working model. By guiding the training process using the 3-stage curriculum, the agent can learn strategically and reduce the chance of falling into a local minimum. In the first stage of the curriculum, the vast decrease in entropy indicates the incorporation of behavior cloning, and demonstration serves as a great starting point for the entire training process, which allows the agent to quickly on the track to discovering the optimal policy. The entropy decreases to a plateau in the third stage, indicating that the robot is not learning effectively. The plateau might be because the terrain is not challenging enough to differentiate from the flat terrain. The trained model using the adjusted approach also performs well on unseen terrains, especially when the friction is high and the terrain is challenging. The comparably good performance on rock terrain might be because combining high friction with the bumping surface provides more gripping to the robot, making it walk easier on this type of terrain.

Given more time, there are several things we would like to implement. (1) We can replace the simple harmonic oscillator with a Hopf oscillator or a more biologically inspired neural oscillator for more accurate biological gait cycles. Hopefully, with a better demonstration sequence, we can free up the locked second joint implemented and have a lower entropy even in the first stage and eventually a better performing model. (2) We can implement a high-level controller for navigation and speed input to the locomotion controller. This way, we can utilize the training for the high-level input with the locomotion controller. We can use the high-level navigation controller to continue improving the locomotion controller in navigating challenging terrains, obstacle avoidance, and motion in constrained spaces. (3) We saw that the robot's performance does not improve much in challenging terrains, possibly due to the terrain being not complex enough. This way, we can try to make the terrain more challenging by incorporating maps such as more rocks, stairs, or complex real-life environments. The model can be hopefully generalized even more in unseen situations.

REFERENCES

[1] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, ”Unity: A general platform for intelligent agents,” arXiv.org, 06-May-2020. [Online]. Available: https://arxiv.org/abs/1809.02627v2. [Accessed: 01-May-2022].

[2] D. Jain, A. Isen, and K. Caluwaerts, ”Hierarchical reinforcement learning for quadruped locomotion,” 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[3] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, ”Real-time motion planning of Legged Robots: A model predictive control approach,” 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017.

[4] F. Raza, D. Owaki, and M. Hayashibe, ”Modeling and control of a hybrid wheeled legged robot: Disturbance analysis,” 2020.

[5] Honglak Lee, Yirong Shen, Chih-Han Yu, G. Singh, and A. Y. Ng, ”Quadruped Robot Obstacle Negotiation via reinforcement learning,” Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.

[6] G. McKechnieFollow, “Spot Mini (rigged) - download free 3D model by Greg McKechnie (@mckechniegreg6) [5dcbee7],” Sketchfab, 01-Jan-1968. [Online]. Available: https://sketchfab.com/3d-models/old-spot-mini-rigged-5dcbee77730640269cef5bd2587e328a. [Accessed: 01-May-2022].

[7] J. Carpentier and P.-B. Wieber, ”Recent progress in Legged Robots Locomotion Control,” Current Robotics Reports, vol. 2, no. 3, pp. 231–238, 2021.

[8] J. Chen, H. San, and X. Wu, ”Gait regulation of a bionic quadruped robot with Antiparallelogram Leg based on CPG oscillator,” Complexity, vol. 2019, pp. 1–11, 2019.

[9] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, ”Learning agile and dynamic motor skills for Legged Robots,” Science Robotics, vol. 4, no. 26, 2019.

[10] J. Hui, “RL - Proximal Policy Optimization (PPO) explained,” Medium, 29-Dec-2018. [Online]. Available: https://jonathan-hui.medium.com/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12. [Accessed: 01-May-2022].

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, ”Learning quadrupedal locomotion over challenging terrain,” Science Robotics, vol. 5, no. 47, 2020.

[12] L. Wang, in Model predictive control system design and implementation using MATLAB, London: Springer, 2010, p. xii.

[13] M. M. Antique, M. R. Sarker, and M. A. Ahad, ”Development of an 8DOF quadruped robot and implementation of inverse kinematics using denavit-hartenberg convention,” Heliyon, vol. 4, no. 12, 2018.

[14] Baske, M (2021) Angry AI [Source code]. https://github.com/mbaske/angry-ai.

[15] N. Sugimoto and J. Morimoto, ”Phase-dependent trajectory optimization for CPG-based biped walking using path integral reinforcement learning,” 2011 11th IEEE-RAS International Conference on Humanoid Robots, 2011.

[16] P. Biswal and P. K. Mohanty, ”Development of quadruped walking robots: A Review,” Ain Shams Engineering Journal, vol. 12, no. 2, pp. 2017–2031, Jun. 2021.

[17] P.-B. Wieber, R. Tedrake, and S. Kuindersma, “Modeling and Control of Legged Robots,” in Springer Handbook of Robotics, Springer, 2016, pp. 1203–1234.

[18] R. P. Feynman, The Feynman lectures on physics. San Francisco, CA: Pearson Addison Wesley, 2006.

[19] ”Tutorial overview of Model Predictive Control,” IEEE Control Systems, vol. 20, no. 3, pp. 38–52, 2000.

[20] V. Klemm, A. Morra, L. Gulich, D. Mannhart, D. Rohr, M. Kamel, Y. de Viragh, and R. Siegwart, ”LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3745–3752, 2020.

[21] V. Matos and C. P. Santos, ”Omnidirectional locomotion in a quadruped robot: A CPG-based approach,” 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

[22] W. Yang, N. Chong, and B. You, ”Entrainment-enhanced neural oscillator for imitation learning,” 2006 IEEE International Conference on Information Acquisition, 2006.

[23] X. Chen, A. Ghadirzadeh, J. Folkesson, M. Bjorkman, and P. Jensfelt, ”Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments,” 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

[24] X. Da, R. Hartley, and J. W. Grizzle, ”Supervised learning for stabilizing underactuated Bipedal Robot Locomotion, with outdoor experiments on the Wave Field,” 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017.

[25] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, and W. Yang, ”Learning free gait transition for quadruped robots via Phase-Guided Controller,” IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 1230–1237, 2022.

[26] Y.-S. Luo, J. H. Soeseno, T. P.-C. Chen, and W.-C. Chen, ”Carl,” ACM Transactions on Graphics, vol. 39, no. 4, 2020.

[27] Zheng Liu, M. H. Ang, and W. K. G. Seah, ”Reinforcement learning of cooperative behaviors for multi-robot tracking of multiple moving targets,” 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.