Multi-Agent Reinforcement Learning for Non-Convex Obstacle Navigation in Collective Transport

Joshua Bloom
Robotic Engineering
Worcester Polytechnic Institute
Worcester, MA, United States
jdbloom@wpi.edu

Steven Brewer
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, United States
scbrewer@wpi.edu

***Abstract*—Navigating environments populated with non-convex obstacles has traditionally proved difficult for multi-agent systems without an oracle. These obstacles prove difficult for reinforcement learning to overcome due to the local minima at the vertex of the obstacle. We aim to overcome this type of obstacle in a multi-robot collective transport task via local emergent communication.**

***Index Terms*—Reinforcement Learning, Collective-Transport, Multi-Agent Systems**

I. Introduction

Can a cooperative-competitive multiple robot system transporting an object escape from local optima through communication? Non-convex obstacles create local optima that historically multi-robot systems struggled to overcome. Through deep reinforcement learning with emergent communications it can be shown that the multi-robot system can outperform previous methods for escaping the local minima.

Reinforcement learning involves an agent that must learn about a dynamic environment through trial and error and choose the best outcome to maximize reward. In the standard reinforcement learning model the agent has the choice in action based on the state received from the environment. The goal of the agent is to determine the optimal behavior. The agent is trained to complete a specific task with the goal of generalizing to similar tasks.

The tasks are modeled as Markov decision processes (MDP). An MDP is memoryless, meaning given the current state the future is independent of the past states [7]. The environment and agent states S consist of $s_1, s_2, s_3, \dots s_t$ where $s \in S$. The set of all possible actions the agent can make A consists of $a_1, a_2, a_3, \dots a_t$ where $a \in A$. r is the immediate reward after transition from state s to s' where $s' = s_{t+1}$ after taking action a. Reinforcement learning is stochastic in nature and therefore the probability of transitioning from s to s' given action a, $P(s'|s, a)$, is calculated. The goal is to learn a policy π that maximizes the cumulative reward. This is done by selecting the action a with the greatest probability of transitioning the agent to the goal state.

Utilizing multiple robots in a system, makes the network fault tolerant. The system of robots excel at transporting large cumbersome objects, surveillance of large areas, and robotic tasks that can be distributed and solved in parallel [8]. Using multiple robots to manipulate an object, known as collective transport, has broad applications in logistics, infrastructure, agriculture, disaster recovery, or any scenario around moving objects where human safety is of extreme concern.

Using multiple robots to solve complex tasks traditionally was solved with hand-built algorithms for each specific transport tasks. Reinforcement Learning and collective transport merge into an interesting machine learning problem which can solve trajectory planning of the multi-robot system. Applying reinforcement learning to minimal agents in a collective transport task allows for the study reinforcement learning in a cooperative-competitive setting.

The action space is comprised of the two wheel speeds from 0 to 10. The observation space consists of the distance and angle of the robot to the goal, distance and angle of the robot to the object, distance and angle of the object to the goal, wheel speeds, and 24 proximity sensor values.

For this work we are working in the ARGoS simulator [16]. We are using TCP to link between ARGoS and python where ARGoS is acting as the environment and python houses all the code for learning. We have implemented several reinforcement learning deep neural networks through pytorch.

II. Literature Review

A. Collective Transport

Collective robotics is any task requiring coordination between multiple robots. Collective transport is simply collective robotics applied to an object transport task. Such tasks are common in practice when objects are too heavy or too awkward for a single robot to successfully maneuver on their own. There are three traditional methods implemented as outlined in [8]. Pushing only is a method whereby robots can only apply force in the direction they are currently moving, thereby pushing the intended object. Caging is a process where robots surround the object and guide the object with some robots pushing and others searing. Finally, there is grasping where the robots are allowed to rigidly attach themselves to the object they are transporting. Here, we implement a combination of caging and grasping where the robots are evenly spaced around the object and are rigidly attached via a gripper.

B. Reinforcement Learning

Reinforcement Learning has been of great interest in the robotics community and the machine learning world recently. Mnih et al [4] proved that neural networks were capable of human level control of atari games using a Deep Q-Network (DQN). The DQN was a pivotal moment because it opened the door to using neural networks in reinforcement learning to accomplish tasks previously thought non-commutable. Following the DQN, van Hasselt [3] showed improved performance and learning stability with a Double Deep Q-Network (DDQN). DDQN uses two networks, an evaluation network and a target network, to compute the temporal difference error used in the learning step in reinforcement learning. DDQN also introduced the idea of using batch learning which allows for the algorithm to become temporally independent and learn in a pseudo off-line fashion. Both DQN and DDQN are suited to environments where discrete actions can be taken, Lillicrap et al [2] addressed this by extending the Deterministic Policy Gradient (DPG) of Silver et al. [12] with Deep Deterministic Policy Gradient, thus allowing for an Actor-Critic reinforcement learning regime. DDPG suffers from stability issues in the weight biases within the network and thus Fujimoto et al. [1] introduced the Twin Delayed Deep Deterministic Policy Gradient (TD3). TD3 introduced the idea of having two critics and taking the minimum value between the two when completing a learning step. This forces the network to undervalue state-action pairs which does not get backpropagated, whereas, DDPG was over-valuing the state-action pairs leading to tremendous compounded error.

C. Non-Convex Obstacles

Non-Covex obstacles have been an issue in the robotics community due to the local minima problem. This is a problem whereby an algorithm will become stuck in a policy that garners the maximum reward in the local policy space and due to the nature of the obstacle the algorithm is not able to branch the gap to a better policy. In the case of Non-Convex objects, the algorithm will become stuck in the crook of the obstacle because going either way will result in a worse reward than where it currently is. However, if the algorithm could see just a little bit further then it would see a better policy. Getting the algorithm to do this is the main focus of this paper. Previous work has been done on this subject [11], [13], [14], [15], however none of these have explore a multi-robot setting where robots are connected into a single moving system. This is not only a multi-robot navigation problem, it is a multi-robot cooperative navigation problem. We chose to use reinforcement learning here because of its ability to scale to larger systems and different environment sizes and shapes.

III. Problem Description

A. Problem Statement

Non-convex objects cause local minima in training which traditional multi-robot systems struggle to overcome. An obstacle is non-convex if one of more of the interior angles are greater than 180 degrees. to overcome the local minima of non-convex obstacles, emergent communications will be employed. The primitive communication between robots in the system allows the system to overcome the obstacle.

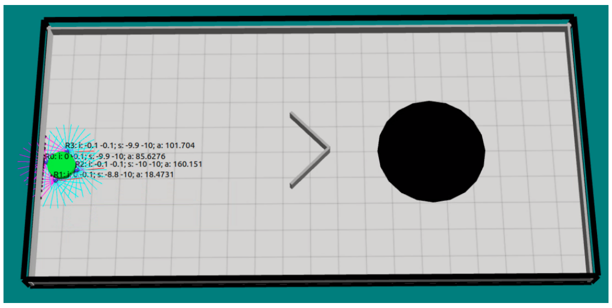

R3: i: -0.1 -0.1; s: -9.9 -10; a: 101.704
R0: i: 0 -0.1; s: -9.9 -10; a: 85.6276
R2: i: -0.1 -0.1; s: -10 -10; a: 160.151
R1: i: 0 -0.1; s: -8.8 -10; a: 18.4731

Fig. 1. Non-Convex Obstacle is the 'V' shape shown in dark grey. The Black circle is the goal location. The green cylinder is the object to be moved. The four blue cylinders around the green one are the robots. The lines are the proximity sensor where the line color is how close the sensor is to an object

B. Goals

The primary goal is to explore non-convex obstacle avoidance. This will be done using minimal assumptions. Variability in reinforcement learning algorithms will be explored in a collective transport setting, this includes partial observability in minimal robot systems. The stability of learning in various conditions will also be explored. This will lead to a determination of which algorithms best avoids local minima in non-convex objects.

IV. Methodology

There are several control systems that could be implemented for the task on collective transport in with multiple robot systems. The first being hand tuned features. Hand tailored control is good for repeatable environments or repeatable tasks. They have historically been used to allow robots to complete tasks however they do not generalize well to complex environments. The system is overly specialized and if the environment changed slightly the system fails.

Q-Learning adds the ability for the system to learn about the environment. The state action pair is stored inn a Q-table that is used to look up the probability of reward. Q-Learning generalize better than hand tailored control and might work, however, the complexity of the state-action space is such that no known computer could handle a policy table for all combinations.

Deep Q-Learning (DQN) trains a neural network to learn and approximate the policy function. Because the policy is approximate, the model generalizes to different obstacles. DQN suffers form a problem with over-fitting. This leads to unstable learning as show in Figure 4.

To overcome the issues with DQN Hasselt et. al [3] introduced Double Deep Q-Learning (DDQN). DDQN uses two neural networks, one that is adjusted every learning step and one that is adjusted at a much slower frequency, for instance, every 1000 learning step. The second network, called the target network, is copied from the current learning network, called the evaluation network, and used in the learning algorithm to provide a stable target policy approximation. This secondary network disassociates the learning steps from time and provides a stable target for the network to use when calculating the value of state-action pairs.

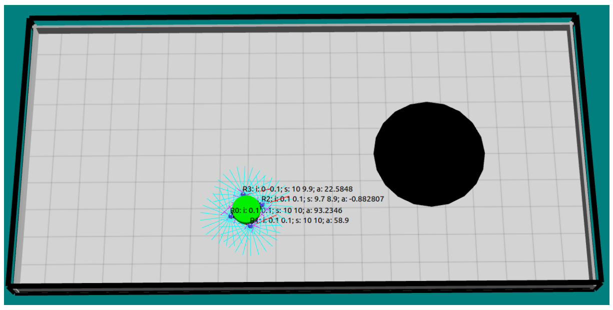
A separate, but equally important mechanism to decouple our learning from time is the replay buffer. Using the replay buffer allows us to store $[State, Action, Reward, NewState, Termination]$ sequences and then randomly sample from these sequences so the network will learn from time invariant samples. This can be implemented in both DQN and DDQN.

Due to the Makrov Assumption, the state we are currently in is not dependent on the previous state or and previous state. This assumption limits our ability in sampling the replay buffer. This leads on sampling single independent samples. However, there is a trick which is referred to as Horizon Learning. This trick allows us to look at a string of sequential samples as a single sample. When we sample multiple of these sequences we are enabling the network to link specific state-action pairs to other state-action pairs while maintaining the Markov Assumption.

There is one more learning scheme that is worth noting which is Actor-Critic. Actor-Critic methods have proven to learn continuous action spaces with tremendous results. We briefly explored two types of actor critic in this work but did not pursue them due to time limitations. We looked at Deep Deterministic Policy Gradient [2] which is able to learn but suffers from overestimation biases and we looked at Twin Delayed Deep Deterministic Policy Gradient [1] which introduces a second Critic, takes the minimum between the two critic values and uses that to learn. This minimization prevents overestimation because under-estimation is not propagated during back propagation where as overestimation is.

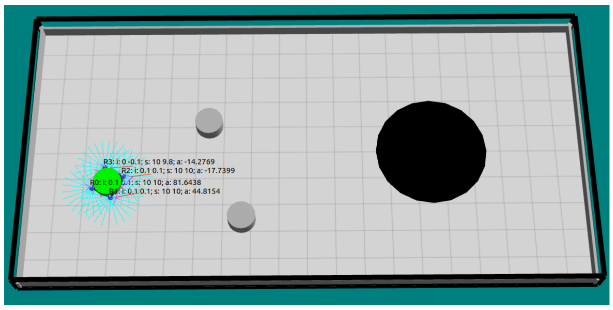
In order to reach collective transport with non-convex obstacles we set up 3 environments to test our network on. The first is shown in Figure 2 and is an environment with no obstacles. This is referred to as the Baseline and is used to benchmark all networks and control schemes. The second is shown in Figure 3 and is an environment with 2 randomly placed obstacles. This environment is called Obstacles. Finally, we have the Non-Convex environment, shown in Figure 1. The Obstacle environment allows us to recognise if the network is capable of navigating simple obstacles and the Non-Convex environment introduces a non-convex obstacle.

The final mechanism studied here is that of communication. Rather than set a fixed variable or message type to send we introduce another DDQN which looks at the observation space and outputs a message of length 1, rather, a character c, where $c \in A$, and A is a discrete alphabet. The chosen message is then sent in two ways, directed or undirected. In directed communication, the message is passed to either the robot directly clockwise or counterclockwise from the agent. All agents share the same communication scheme. So, for instance, in directed clockwise communication all agents are passing their messages to the agent clockwise from themselves. In



R3: i: 0 -0.1; s: 10 9.9; a: 22.5848
R2: i: 0.1 0.1; s: 9.7 8.9; a: -0.882807
R0: i: 0.1 0.1; s: 10 10; a: 93.2346
R1: i: 0.1 0.1; s: 10 10; a: 58.9

Fig. 2. Environment with No Obstacles. The black circle is the goal location. The green cylinder is the object to be moved and the blue cylinders are the robots where the line around them are the proximity sensors.



R3: i: 0 -0.1; s: 10 9.8; a: -14.2769
R2: i: 0.1 0.1; s: 10 10; a: -17.7399
R0: i: 0.1 0.1; s: 10 10; a: 81.6438
R1: i: 0.1 0.1; s: 10 10; a: 44.8154

Fig. 3. Environment with Obstacles, Green cylinder is the object to be moved. The blue cylinders are the robots where the lines around them are the proximity sensors. The grey cylinder are the obstacles and the black circle is the goal location.

undirected communication the agents pass messages both clockwise and counterclockwise. The intuition is that one agent will engage the obstacle and pass along a message saying so. The other robots will receive this message and change their actions accordingly. When an agent receives a message it is treated the same as the rest of the observation space and is appended to the end of the observation. Here we note that the messages are being represented as one-hot encoding of the actual message value.

V. Results and Discussion

Figures 4 and 5 show the reward curves over time for the DQN and the DDQN respectively. We can see that the DDQN is much more stable in its learning life than the DQN. However, interestingly, when we take the best models learned from each in the Baseline environment

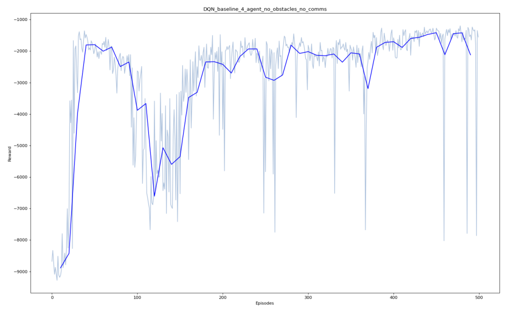


Fig. 4. DQN Baseline

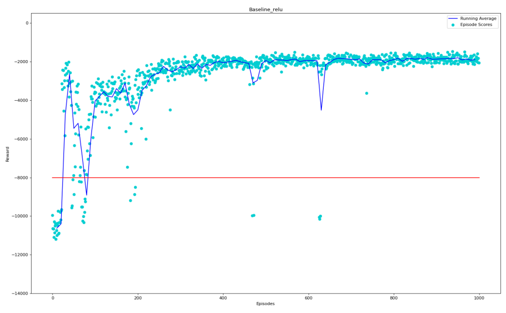


Fig. 5. DDQN Baseline. Red line indicated the robots have found the goal

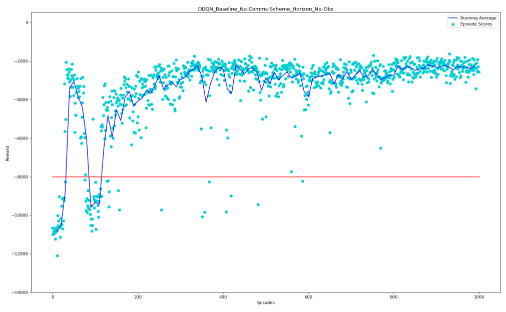


Fig. 6. DDQN Baseline with Horizon learning. Red line indicated the robots have found the goal

we see the same performance. Namely, both reach 100% success over 1000 episodes. The difference comes when we explore the standard deviation between the scores for the test. The DQN achieves a lower standard deviation than the DDQN which is the opposite of what we expect. However, when these are trained with obstacles and with the non-convex obstacle this result flips and the DDQN is much better. The DQN fails to learn in the simple obstacle environment and is from here on out excluded in our analysis.

When we look at the Obstacles environment we see that the DDQN is able to efficiently learn the environment, testing with a 3.2% failure rate. When we add in Horizon Learning, we actually see the performance decrease slightly, achieving a failure rate of 4.5%. We also see the model achieve a lower score and a worse standard deviation. This could be due to the model overfitting

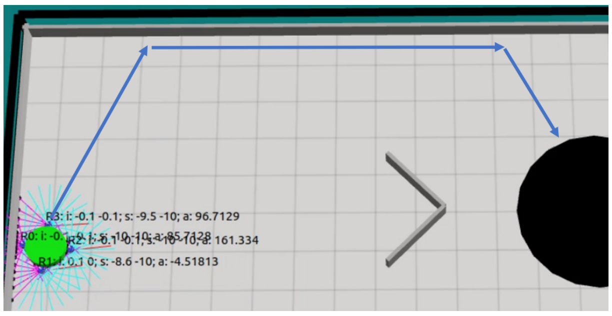

R3: i: -0.1 -0.1; s: -9.5 -10; a: 96.7129
R0: i: -0.1 -0.1; s: -10 -10; a: 85.7128
R2: i: -0.1 -0.1; s: -10 -10; a: 161.334
R1: i: 0.1 0; s: -8.6 -10; a: -4.51813

Fig. 7. DDQN Baseline Non-Convex Obstacle Behavior

TABLE I
NONLINEAR MODEL RESULTS

Test	Testing Scheme	Average Reward	Score STD	Percent Failures
1	Baseline 4 Agents DQN No Horizon Learning No Obstacles No Communication	-1787.7	158.1	0.0%
2	Baseline 4 Agents DDQN No Horizon Learning No Obstacles No Communication	-2219	404.0	0.0%
3	Baseline 4 Agents DDQN Horizon Learning No Obstacles No Communication	-2162	163.9	0.0%
4	4 Agents DDQN No Horizon Learning 2 Obstacles No Communication	-2187.8	1522.3	3.2%
5	4 Agents DDQN Horizon Learning 2 Obstacles No Communication	-2563.1	1890.9	4.5%
6	4 Agents DDQN Horizon Learning 2 Obstacles Right Communication	-2496.3	2665.9	3.3%
7	4 Agents DDQN Horizon Learning 2 Obstacles Neighbor Communication	-3786.1	1842.5	5.4%
8	4 Agents DDQN No Horizon Learning Non-Convex Obstacles No Communication	-4472.8	1180.0	2.7%
9	4 Agents DDQN Horizon Learning Non-Convex Obstacles No Communication	-3995.0	2731.0	55.9%
10	4 Agents DDQN Horizon Learning Non-Convex Obstacles Right Communication	-10233.0	1587.7	95.6%
11	4 Agents DDQN Horizon Learning Non-Convex Obstacles Neighbor Communication	-8856.2	3795.6	74.0%

on the sequential data and causing a failure in specific scenarios when it comes in contact with an obstacle. We then look at directed and undirected communication. Directed communication does the worst with 3.3% failure where as undirected only fails 5.4% of the time. This is an interesting result because it shows that communication does not improve performance but rather hampers it. This can be caused due to the fact that there is no signalling informing the communication adopted. We are allowing the communication to be emergent however we are not telling the algorithm to emphasize specific signals in certain scenarios rather than others. We are also not influencing the action network in any way thus we are not telling either network to preface communication. So what we end up seeing is a network with communication data that it doesn't know what to do with and thus treats it as noise.

Unfortunately, we see much the same in the Non-Convex environment, only now it is much more pronounced. The baseline network is able to solve the environment with a 2.7% failure. It is worth taking a moment here to discuss the behavior of the robots. Figure 7 shows the policy learned by the algorithm. The network learns to ride the walls in the northern direction and then in the eastern direction. Once the robots get to a specific point on the wall, probably a specific angle to the goal, they rotate and move south towards the goal. This is a very interesting phenomenon and one that was only achieved once out of 10 training rounds. Adding horizon learning and communications only makes the performance worse, although, this is mostly due to the difficulty of the environment and the fact that we were able to produce a model capable of solving this is an anomaly. The issue with the model that is capable of solving the environment is that it does not interact with the obstacle. It has learned a specific trajectory to take rather than an interaction. This model is incapable of solving the non-convex obstacle.

There are a couple tricks to forcing the robots to interact with the obstacle. The first is to make the environment narrower in the north-south directions and increase the probability of collision/ interaction between the robots and the obstacle. The other option is to move the obstacle around the environment and place it in the direct path of the robots and the goal. This randomization would force the robots to interact and learn the solution to the obstacle rather than the environment. If the network is able to solve the obstacle then is doubtful with the current network setup. Communication for this type of obstacle is a necessity and as mentioned above, the current implementation does not reward good, bad, or any other type of communication. What is needed is either a reward structure or a loss tied to the communication. The problem with a reward structure is that the network needs to know information that is not readily available without an oracle. The best way to go is to use a loss function, however, the loss is complex as it needs to tie communication with future actions. Due to the action space being ± 0.1m/s and our max velocity per wheel is 10m/s which induces a momentum. So, any action chosen at the current time step could take a few time steps to make a difference in the movement of the robots. This work is ongoing and will be reported on in a future paper.

VI. Conclusion

In this work we explore environments ranging from basic collective transport to collective transport in the presence of obstacles to navigating a non-convex obstacle. Each environment is successively more difficult to solve and as such we introduce several learning strategies with the overall goal of solving an environment with a non-convex obstacle. The first strategy we employ is horizon based learning where we learn from a series of experiences rather than randomly selecting a series of experiences. The second is communication where we have directed and undirected channels. We show that the basic DDQN is able to solve all environments, although we note that the non-convex obstacle was not solved, rather the robots learned to navigate around it.

Future work will involve exploring communication deeper and developing a new loss function that influences meaningful communication. We will also be exploring other types of obstacles including non-stationary obstacles and high-valued obstacles, such as a person.

References

[1] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing Function Approximation Error in Actor-Critic Methods,” arXiv:1802.09477 [cs, stat], Oct. 2018, Accessed: Dec. 05, 2020. [Online]. Available: http://arxiv.org/abs/1802.09477.

[2] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” arXiv:1509.02971 [cs, stat], Jul. 2019, Accessed: Jul. 06, 2020. [Online]. Available: http://arxiv.org/abs/1509.02971.

[3] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” p. 7.

[4] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[5] Eccles, Tom, et al. “Biases for Emergent Communication in Multi-Agent Reinforcement Learning.” ArXiv:1912.05676 [Cs], Dec. 2019. arXiv.org, http://arxiv.org/abs/1912.05676.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A Brief Survey of Deep Reinforcement Learning,” IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[7] M. Wiering and M. van Otterlo, Eds., Reinforcement Learning, vol. 12. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[8] E. Tuci, M. H. M. Alkilabi, and O. Akanyeti, “Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art,” Front. Robot. AI, vol. 5, p. 59, May 2018, doi: 10.3389/frobt.2018.00059.

[9] O. Shorinwa and M. Schwager, “Scalable Collaborative Manipulation with Distributed Trajectory Planning,” p. 8.

[10] Patel, J., et al. “Mixed-Granularity Human-Swarm Interaction.” 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 1059–65. IEEE Xplore, doi:10.1109/ICRA.2019.8793261.

[11] Alam, M. S., and M. U. Rafique. “Mobile Robot Path Planning in Environments Cluttered with Non-Convex Obstacles Using Particle Swarm Optimization.” 2015 International Conference on Control, Automation and Robotics, 2015, pp. 32–36. IEEE Xplore, doi:10.1109/ICCAR.2015.7165997.

[12] Silver, David, et al. “Deterministic Policy Gradient Algorithms.” International Conference on Machine Learning, PMLR, 2014, pp. 387–95. proceedings.mlr.press,http://proceedings.mlr.press/v32/silver14.html.

[13] Vasilopoulos, Vasileios, and Daniel E. Koditschek. “Reactive Navigation in Partially Known Non-Convex Environments.” Algorithmic Foundations of Robotics XIII, edited by Marco Morales et al., Springer International Publishing, 2020, pp. 406–21. Springer Link, doi:10.1007/978-3-030-44051-0_24.

[14] Adibi, M., et al. “Adaptive Coverage Control in Non-Convex Environments with Unknown Obstacles.” 2013 21st Iranian Conference on Electrical Engineering (ICEE), 2013, pp. 1–6. IEEE Xplore, doi:10.1109/IranianCEE.2013.6599856.

[15] Kowalczyk, W., and K. Kozlowski. Control of the Differentially-Driven Mobile Robot in the Environment with a Non-Convex Star-Shape Obstacle: Simulation and Experiments. 2016. Semantic Scholar, doi:10.12700/APH.13.1.2016.1.9.

[16] Pinciroli, Carlo, et al. “ARGoS: A Modular, Multi-Engine Simulator for Heterogeneous Swarm Robotics.” 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 5027–34. IEEE Xplore, doi:10.1109/IROS.2011.6094829.