Robot Guided Emergency Evacuation with Multi-Agent Reinforcement Learning

Nathaniel Belles
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA
nbelles3@gatech.edu

Alan Hesu
School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA
ahesu6@gatech.edu

Kin Man Lee
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
klee863@gatech.edu

***Abstract*—In this paper, we explore the topic of robot guidance of people in fire evacuation. A custom environment is introduced to model the interaction between robots and human actors with simplified human behavior. Using multi-agent reinforcement learning (MARL), we show that it is possible to train robots to become leaders to guide the actors to an exit in a simulation environment. Emphasis is placed on the shaping of our reward function, and the impact of the weighted parameters. We test our policy across diverse environments and present our findings. Our results show that there is potential in using MARL for robot guidance in fire escape, however, there is significant future work required to teach the robots complex behaviors and to generalize across varying environments in this setting.**

I. Introduction

Indoor fire emergencies present challenging environments for occupants to navigate, as a quickly spreading fire offers little time for people to locate nearby fire exits and navigate to them. Evacuation efficiency can be further hampered by a number of factors, including a person's unfamiliarity with the environment, congestion near hallways or exits, and decreased visibility due to smoke [1]. As building construction continues to grow, the need for effective fire evacuation procedures remains a critically relevant safety concern. One potential aid is the use of evacuation guidance robots that can efficiently guide people towards nearby fire exits.

The effectiveness of leaders and groups in evacuation scenarios has been well modeled and studied [2]. These leaders may be specially trained building staff or fire marshals, or they may simply be occupants who are more familiar with the building layout and are able to help other people navigate to the exit. Furthermore, people strongly tend to favor familiarity in exit scenarios rather than adopt a general "panic" behavior. This applies both to familiar pathways in buildings as well as an individual's social relationships with others [3]. These factors suggest that evacuation guidance robots can provide similar aid by acting as leaders and overcoming an individuals unfamiliarity with the environment by guiding them to safer nearby exits that they might not otherwise take.

These evacuation robots may be placed throughout the building in critical areas or may be multi-function robots, such as mobile healthcare robots that adopt an evacuation guidance behavior in the event of an emergency. In such a scenario, the robot would then move towards the best exit option, guiding nearby occupants along the way to reach the exit. In addition to overcoming potential unfamiliarity with the exit locations, the robots may prove critical when the human visibility is impaired by smoke. With onboard sensors or even networked communication to give them global information about the building and its occupants, the robots would be far less hampered by the smoke, so they can still function as evacuation leaders at or near full capacity.

This study focuses on the simulation of a multi-robot team of evacuation robots using a multi-agent reinforcement learning (MARL) approach. To provide adequate coverage in large building spaces, multiple robots will be needed, and they will need to act cooperatively in order to minimize the effects of congestion while maximizing the rate at which humans can exit the space. Thus, we propose a simulated evacuation environment involving multiple robots and human followers. The design of this reinforcement learning environment is outlined, and the results from multiple variations in environment and training parameters is then discussed. Finally, limitations of our approach and potential future work are considered.

II. Related Works

A. Evacuation Modeling

The efficacy of evacuation guidance robots has been demonstrated before. Robinette et al. [4] performed a study where human participants placed a high degree of trust in guide robots in high risk emergency scenarios. The results suggest that humans may be likely to follow robots to more unfamiliar locations, as they believe the robot is guiding them towards the optimal exit.

Other studies, on the other hand, model the behavior of multiple guide robots in a crowd setting rather than the single-robot single-follower scenario above. Zhang and Guo [5] developed a cooperative gradient estimation algorithm to perform trajectory planning. They divide the robots into leaders and shepherds that form a formation where the leaders move towards the exit, and the shepherds guide nearby people as well. Tang et al. [6] simulate the use of an optimal exit selection strategy that involves a human crowd model and an estimation of the best path to the exit. These, however, use explicitly defined algorithms for robot behavior rather than ones learned through reinforcement learning techniques.

These studies also incorporate models of human behavior in evacuation scenarios, for which there exist multiple approaches that typically fall under macroscopic and microscopic models. A macroscopic approach models the crowd behavior as a whole, while a microscopic approach examines behavior on an individual level and how each robot and actor interacts with each other and its environment, which may provide a more accurate simulation [7]. Among microscopic models, the cellular automata, social force, and steering model are examples of different approaches to modeling movement [8]. Additional research has focused on capturing the complexities of human behavior in the event of an evacuation and the factors that affect them, such as an individual's nervousness, familiarity with the environment, and willingness or ability to aid others [9].

B. Multi-Agent Reinforcement Learning

The recent advancements in computing technology and data collection has led to an explosion in machine learning research. Among the learning methods, reinforcement learning is used to determine how an agent will act in a given environment to achieve the best long-term reward. This can be expanded into a case where there are multiple agents in an environment, commonly known as multi-agent reinforcement learning (MARL). Recent developments in MARL have led to the development of models that have the capability to simulate a large number of agents in increasingly complex environments. Combined with deep neural networks, these simulation models have been deployed with great success when modeling multi-robot collaborative teams.

Many flavors of deep reinforcement learning algorithms have been used to model evacuations. By selecting a random environment for training, the generated policy is able to generalize across many scenarios. There has also been work to explore the difference in behavior of agents for multiple exits with a learning approach. One method utilizes ORCA for collision detection of the agents and the Rainbow DQN to simulate agent behavior [10]. Another approach selects several agents as leaders that will learn the optimal path while the other agents follow a similar path as the leader [7].

III. Methodology

A. Assumptions

We make several assumptions in the setup of the environment in our experiments. First, we assume that robots in the environment have full visibility of the state space (i.e., their observation space is equivalent to the full state space). For simulated humans, we make the assumption that they generally follow the closest robot within their range of vision. For both humans and robots, we assume that they can be modeled where their movement is limited to a set of discrete states and actions. We acknowledge that human behavior is complex, and often times unpredictable, especially in high stress situations involving emergencies. We do not claim that our model of humans in the environment accurately depicts real human behavior in fire scenarios; rather, the simulated humans primarily act as simple followers and potentially cause congestion that the robots need to navigate around. The setup we have created is largely a low-fidelity simulator designed to experiment with MARL robot guidance methods when given simplified human behavior.

B. Environment Definition

Our environment is represented by a two-dimensional square array where each element of the array can either be empty or hold one of the following objects; a robot, or a human, an exit, or a wall. The environment is surrounded by walls with the exception of exits, each of which are two elements wide. Robot and actor locations are randomly generated at the beginning of each simulation and placed in the environment. During each time step each robot and actor gets the chance to act by moving a single element in one of the eight ordinal directions or stay still, all while ensuring no two objects occupy the same space at the same time. When a robot or actor chooses to act such that they occupy the same space as an exit, they are considered to have exited the environment and are removed from the rest of the simulation and do not occupy any space.

An example of the visualization of the environment with three robots and ten humans is shown below in Figure 1. Green indicates a robot, blue indicates a human, red indicates an exit and grey indicates a wall. The black line on each robot and human indicates the direction that it most recently traveled (a black dot if they stayed stilled) to help show the movement of each robot or actor.

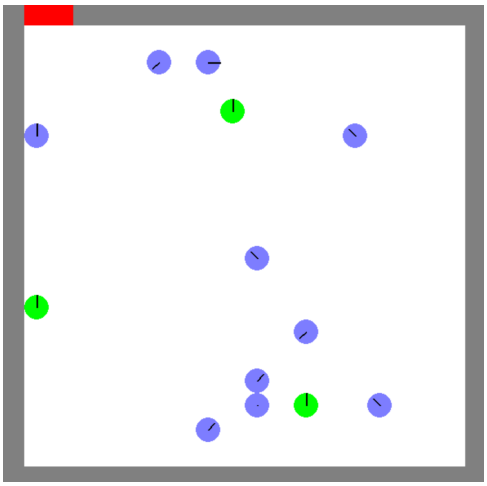

Fig. 1. Visualization of an Example Environment

C. Human Actors

Human actors in the environment are modeled using a simple set of rules. Within its vision range and line-of-sight, each human finds the nearest robot or exit, prioritizing exits at a greater distance over robots. Then, it calculates an optimal action that maximizes its change in distance to the target. It will take this action with 0.75 probability or a random available action with 0.25 probability. This models

the humans' decision to generally follow the guidance robots while adding some degree of stochasticity.

D. Multi-Agent Reinforcement Learning

We leverage MARL to train the robots to learn a policy that guides humans to the exit. The problem is formulated as an Agent Environment Cycle (AEC) as defined in [11]. We experiment with three reinforcement learning algorithms from Stable-Baselines3 [12]. The algorithms are PPO [13], DQN [14], and A2C [15]. We experiment with different values of the hyperparameters for each corresponding algorithm to tune for performance.

The state space is an $n \times n$ 2D array of equal height and width representing the simulation environment. Each index in the array can take on a single value representing either an empty space, wall, exit, robot, or human. The action space is composed of the 8 possible ordinal directions to move into each adjacent square as well as an additional 9th action for standing still. Finally, the observation space is a vectorized 2D array that is defined to be the same as the entire state.

The reward equation is defined in 1 as a sum of multiple terms.

$$\begin{aligned} R = w_{exit}R_{exit} + w_{collide}R_{collide} + w_{wall}R_{wall} \\ + w_{penalty}R_{penalty} + w_{goal}R_{goal} \\ + w_{count\_exited}R_{count\_exited} \end{aligned} \tag{1}$$

where the weights are normalized such that $\sum_{i=1}^{N} w = 1$, and the constants $R_{exit} = 1$, $R_{collide} = -1$, $R_{wall} = -1$, and $R_{penalty} = -1$. Several weights are defined based on the resulting state s' as follows

$$w_{exit} = \begin{cases} 1, & s' = exit \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$w_{collide} = w_{wall} = \begin{cases} 1, & s' = \text{blocked} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The reward for the change in distance to the goal R_{goal} is defined as

$$R_{goal} = \frac{d^{t-1} - d^{t}}{\sqrt{2}} \tag{4}$$

where d is the euclidean distance to the closest exit, and $\sqrt{2}$ is the maximum possible change in distance corresponding to a diagonal move.

The reward for exiting humans R_{count_exited} is defined as the number of people that have exited in the previous timestep.

$$R_{count\_exited} = n_{exit}^{t-1} \tag{5}$$

Additionally, the total reward is clipped in the range $[-1, 1]$, which prevents large changes in the gradient given large magnitudes in the reward value.

E. Environmental Changes

To examine the effects of differing environments on the performance of robot guidance, we introduce different variations of the base environment shown in Figure 1. The changes we make to the environment include random exit locations, and a variable number of exits, robots, and humans. We also create five additional layouts and randomly select one of these environments to train on in each episode for generalization. Layouts of the environments can be seen in Figure 2.

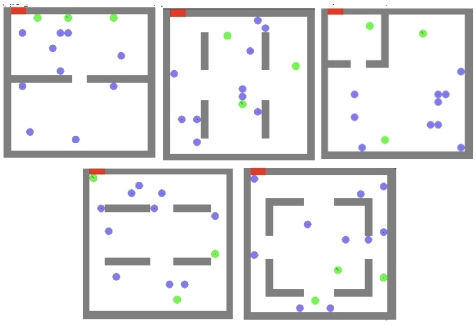

Fig. 2. Layouts of the five varying environments. (Ordered 1-5 from top-left horizontally to bottom right)

F. Performance Metrics

The performance of the trained robots in the simulation environment was evaluated primarily based on two metrics: total evacuation time and percent exited. The total evacuation time was measured using the number of simulation steps taken for every robot pass through the exit. As a caveat, the AEC environment definition takes steps cyclically per individual agent, so the total steps taken is a scaled measure based on the number of robots. Measuring the total human evacuation speed may be a more appropriate measure, but these simulations often involved humans that would never find and pass through an exit. As such, the second metric of percent exited measured how many of the humans could leave through the exit by the time all of the robots left.

IV. Results

Initially, a baseline policy was tuned by varying the weights in the reward function (1), the results of which are summarized in Table I. A number of weights were kept constant, such as those that reward reaching the exit or penalize collisions. While each reward function resulted in very similar performance, a combination of weights $w_{penalty} = 0, w_{goal} = 1, w_{count\_exited} = 1$ ultimately generated the best performance by equally rewarding moving towards the exit and guiding other humans to the exit. This baseline policy equation can be written as

$$\begin{aligned} R_{baseline} = w_{exit}R_{exit} + w_{collide}R_{collide} + w_{wall}R_{wall} \\ + w_{goal}R_{goal} + w_{count\_exited}R_{count\_exited} \end{aligned} \tag{6}$$

TABLE I
RESULTS FROM VARYING REWARD FUNCTION WEIGHTS

$w_{penalty}$	w_{goal}	w_{count_exited}	# steps	% Humans Exited
1	0	0	59.00	58.60
1	1	1	50.93	62.00
0	1	1	59.54	65.20

This baseline was then modified in pieces to show different aspects of the way the environment performs. The parameters of this baseline policy are:

- RL Algorithm: DQN
- Weights: $w_{exit} = 1$, $w_{collide} = 1$, $w_{wall} = 1$, $w_{penalty} = 0$, $w_{collect} = 0$, $w_{num\_follow} = 0$, $w_{goal} = 1$, $w_{count\_exited} = 1$, $w_{hum\_dist} = 0$
- Map size: $20 * 20$
- Number of Robots: 3
- Number of Exits: 1
- Number of Humans: 10
- Human Vision Distance: 7
- Obstacles: No
- Random Exits: No
- Learning Rate: $1e-4$
- Batch Size: 4096
- Time Steps: $8e5$
- Exploration Fraction: 0.3
- Exploration Final Episodes: 0.1

The first thing we wanted to compare is how well it performs when varying the number of humans in the environment relative to the baseline. All parameters, besides the number of humans, are the same as what was used in the baseline policy. The results of this test can be seen below in Table II.

TABLE II
RESULTS FROM VARYING NUMBER OF HUMANS

# Humans	# Steps	% Humans Exited
1	116.09	65.00
2	193.37	75.50
3	117.22	73.00
4	68.49	65.40
5	127.25	72.50
6	84.56	73.28
7	87.73	68.50
8	92.35	66.11
9	80.32	67.50
10	90.93	65.92
15	81.41	67.30
20	126.18	69.24
25	165.82	61.98
50	109.26	57.76

Next, a number of other environment parameters were modified, as shown in Table III. The baseline refers to the environment and reward function as defined above. Next, an additional exit was added, as shown in Figure 3. As expected, this additional exit offered greater throughput and thus resulted in a lower total evacuation time of 55.47 steps and a higher percentage exited of 71%.

TABLE III
RESULTS FROM VARYING ENVIRONMENT PARAMETERS

Environment Variant	# steps	% Humans Exited
Baseline	59.54	65.20
Two exits	55.47	71.00
Switching policy	87.40	68.67
Randomized exit	99.63	77.00
Increased vision	64.97	82.00

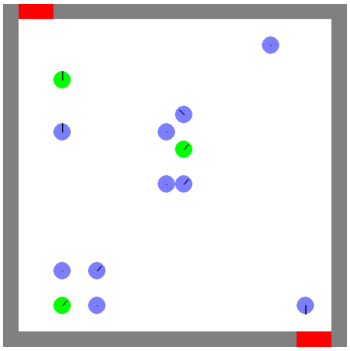

Fig. 3. Environment with two exits

The next variation involved simulating the robots using two separate policies. As an additional new policy, a different reward function was used that differs from the baseline reward function (6) by removing the R_{goal} term.

$$R_{collect} = w_{exit}R_{exit} + w_{collide}R_{collide} + w_{wall}R_{wall} + w_{count\_exited}R_{count\_exited} \quad (7)$$

This was formulated with the goal of increasing the likelihood that robots would remain in the environment and potentially guide additional humans to the exit rather than moving directly to the exit themselves. The new policy was then combined with the baseline policy by running the new policy on the robots for the first 50 steps in the simulation before switching to the baseline policy. The results are demonstrated in the "Switching policy" entry of Table III, where the evacuation time of 87.40 steps sees a significant increase due to the initial exploration phase, while the percent exited of 68.67% reflects a marginally improved performance as more humans are guided to the exit.

The next environment variation, "Randomized exit" in Table III, refers to a randomization of the exit position along the upper wall in the environment. To enable the model to converge given the additional stochasticy in the environment, this policy was trained for far longer, using $3e6$ timesteps compared to the baseline's $8e5$ timesteps. Furthermore, the R_{goal} term was modified by subtracting a constant offset $R_{goalnew} = R_{goal} - 0.25$. Without this offset, it was found that robot would oscillate in front of the exit instead of leaving, as moving in either direction of an exit two cells wide would not change the robot's distance to the goal. The resulting convergence behavior is shown in Figure 4. The resulting evacuation time of 99.63 steps and 77% exited suggests that

while the robots did not move towards the exits as optimally, the increased time spent allowed potentially more humans to follow and exit.

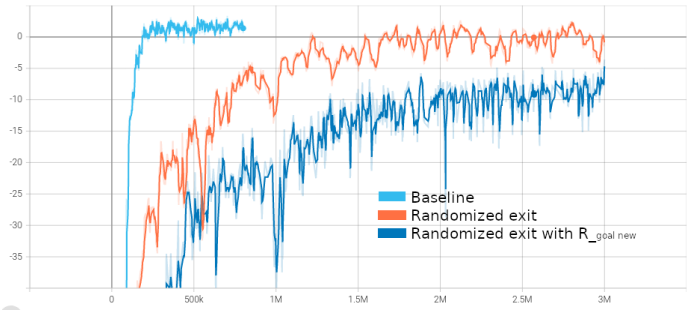

Fig. 4. Mean episodic reward for the baseline policy, randomized exit, and randomized exit with $R_{goalnew}$

The final variation, "Increased vision" in Table III, increases the human vision range, allowing them to see exits and robots from farther away. This potentially reflects a lower amount of smoke and better visibility conditions. Expectedly, the evacuation time of 64.97 steps and 82% shows a marked improvement in performance.

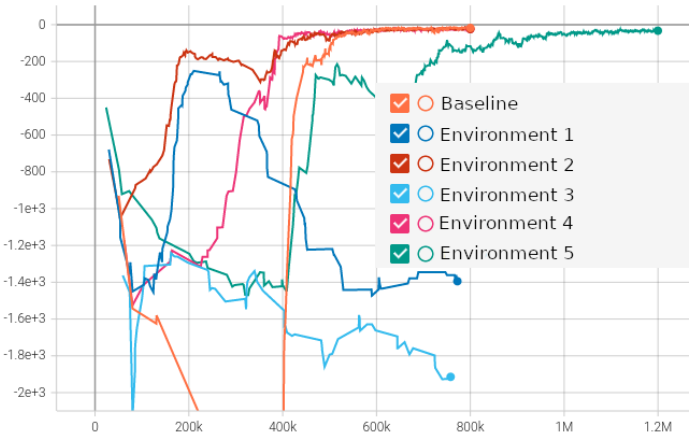

Fig. 5. Mean episodic reward of layouts

TABLE IV
RESULTS FROM VARYING LAYOUTS OF ENVIRONMENT

Environment	# Steps	% Humans Exited
1	DNF	DNF
2	62.40	36.33
3	DNF	DNF
4	73.10	37.66
5	114.20	35.66

Table IV shows the average evacuation time and percent of humans exited over an average of 100 runs across the five environment layouts. Entries with DNF indicates that we were unable to achieve completion of 100 runs within a reasonable amount of time without manual termination. This is caused by robots getting stuck on obstacles in the environment, as termination of an episode is based upon the exit of all robots. Figure 5 shows the mean episodic reward over the step count for each layout. All environments beside 1 and 3 converge to a stable reward.

V. DISCUSSION

A. Reward Function

Each term in the reward function was designed to promote specific behavior such that the combination would produce the desired outcome. R_{exit} represents the terminal reward for reaching the exit, whereas $R_{collide}$ and R_{wall} penalize collisions with other robots, humans, and walls. This in effect reduces the chance of severe congestion events occurring, as robots will attempt to avoid other obstacles and seek more open areas to maneuver. The constant negative $R_{penalty}$ term promotes reaching the goal along an optimal path and prevents oscillatory behavior that may occur as a result of training if the model were to discover a way to continuously accrue positive reward instead of leaving through the exit. The R_{goal} term promotes taking steps to move towards the exit and is formulated in a similar manner as found in [16]; although here, it is also normalized to return a value in the range [-1,1]. This term partially acts in opposition to the R_{count_exited}, which attempts to improve behavior that will pick up more humans rather than directly escaping to the nearest exit. As shown in the resulting baseline weight values, including both of these two parameters generated the best performance.

The inability to completely evacuate every single human and achieve a percent exited score at or near 100% is due in large part to the robots prioritizing movement towards the exit rather than ensuring every human is guided along the way. Instead of training a single policy to achieve both effective exploration and escape behaviors, a combination of multiple policies may be more effective. The result of such an example can be seen in the "Switching policy" in Table III. Realistically, this simulates a behavior that involves exploration and finding human stragglers initially before guiding the entire group to the best exit.

B. Reinforcement Learning Algorithms

Out of the three algorithms we tested, only PPO and DQN provided reasonable performance for our scenarios. Despite training on different sets of hyperparameters, robots trained on A2C consistently get stuck on walls even on our simplest environment. As such, we focused our efforts on PPO and DQN. In general, DQN and PPO offers similar performance on our evaluated metrics. While PPO offered a faster convergence rate based on the number of steps taken, the wall time convergence of DQN was observed to be faster in our experiments. Consequently, we chose DQN as our default RL method for most of our experiments and all results shown are based on DQN unless otherwise mentioned.

C. Environmental Changes

From the data we gathered based on the five layouts that were tested, we observed that our trained policy was unable to

generalize well across the differences. This is especially true in cases where there are narrow passages, such as environments 1 and 3 in Figure 2. In those environments, robots struggle to navigate to the gaps to continue to the exit and frequently get stuck in the corners.

We expected that increasing the number of robots would increase the percentage of humans that exit the environment, but that was not precisely a direct correlation. When we initially increased the number of robots, it increased the percentage of humans able to escape. But, as the number of robots got very large, performance declined, because the environment got very crowded, and actors were not able to move as much as they would have desired. This is also the case for increasing the number of humans while keeping the number of robots the same. We expected that increasing the number of humans would increase the percentage of humans that exit. This was also not true for varying the humans because at a certain point, continuing to increase the number of humans just creates more congestion near the exit and humans are less likely to exit.

D. Hyperparameters

To get the most out of each episode, multiple environments were trained in parallel. This means that the learning that each environment made are all applied back to the same shared policy. This helped a little in speeding up the training because multiple areas can be learned at once and we were able to train for less overall time-steps to get similar training results. This also helped in generalizing to new environments because it trained on a wider set of stochastic data.

Tuning the number of steps taken required manually observing when the mean episodic reward plateaus. We terminate training once the reward stabilizes as the policy does not improve much beyond that point. We also experimented with different batch sizes and found 4096 to work well for both PPO and DQN. For DQN, we modified the Exploration Fraction and Exploration Final Episodes from default values of 0.1 and 0.05 to 0.3 and 0.1 respectively to allow for longer duration of exploration to encourage robots to search for humans in the space. However, we did not observe any noticeable improvements with this change.

One thing that would have made training our policy a lot easier would have been having auto-generated checkpoints that would save the current policy at specified points throughout the training. We spent a lot of time trying to ensure we are training for long enough that our trained policy is as optimal as possible given the reward functions and the environment, but we also wanted to ensure we were not over-fitting to the reward function we had created. This meant that we lost a lot of trained policies because they had over-fit or under-fit. We attempted to get the builtin checkpoint features to work but there were issues with the checkpoints that it created such that they would not evaluate when loaded in to simulate or visualize.

E. Limitations

One major limitation of this environment is that it does not accurately model the behavior of humans. This is because human behavior is very complex and is hard to model especially in emergency scenarios. Because of this, we do not try to model accurate human behavior but instead try to model a scenario where simulated humans simply try to follow the intelligent robots. This would be an ideal scenario where the humans completely trust the robots to take them to the exit.

Another major limitation of this training environment is that it does not generalize well across different environments. Variations may include something as little as changing the number of other humans, number of other robots, or something big like adding obstacles, changing the number or location of exits. All of these things cause the robot policies to behave sporadically. The robots act as though they don't know where the exits are and that they don't know that there are walls in front of them. This may be due to the representation of the observation as a 2D grid of the entire environment. In such a representation, the spatial information may become more difficult to discern. For example, an obstacle at position [2,3] is distinct from an obstacle at position [3,3], even though they are spatially very close to each other.

We believe greater performance could result from changing the observation space to a "vision" based model, which could consist of objects and the distance to them at each degree interval around the robot, similar to a Lidar measurement. This representation better captures relative spatial information and may improve generalization; a robot that sees an exit to its right, for example, does not care where it is in the grid. Furthermore, limiting the observation space will promote exploration and more realistically model actual robots, as they will have to move to see changes in the environment.

Another limitation that we found from testing is that often the robot will get stuck in corners. This comes from how the humans have been programmed to follow the robots and how the robots have learned to gather humans. This causes issues because once the robot gets stuck in a corner surrounded by humans it has almost no way to get unstuck besides waiting for the randomness of the humans to create an opening long enough for the robot to escape and thus the humans can follow.

The consistency of the robot behavior seems to be another limitation of our environment. The robots do not always act the same in similar scenarios. For example, if all the humans have exited the environment, sometimes the robot chooses to exit but other times it chooses to continue maneuvering the environment and never exiting. This is likely due to how each simulation is determined to be complete. Because our environment currently decides that the simulation is complete when all robots have exited and not when all humans have exited, it ends up teaching the robot to wait near the exit for different amounts of time because it does not know how long the simulation will take and does not consider how many humans are left.

VI. Conclusion

Given the need for improving fire safety and the potential for robots to act as leaders in an evacuation, developing safe cooperative behaviors is critical. This study investigated the use of a MARL environment to model and train emergency evacuation guide robots to fulfill such a role. Through shaping the reward function, a preferred behavior is achieved that minimizes the evacuation time and maximizes the number of people that can quickly escape. Then, modifying the environment parameters demonstrates the extent to which the models can generalize.

The performance of the baseline reward function demonstrates the effect that tuning the weights for each term can have on the overall performance. Because the robots needed to a achieve a complex and sometimes competing set of goals involving finding humans, guiding them to the exit, and avoiding congestion or other obstacles, a well designed reward function was necessary to enable fast convergence of the RL model. Given greater computing resources and more training time, a sparser reward function may achieve the same performance.

While some modifications to the environment yielded satisfactory results, such as an increase in the number of exits, others did not, such as the addition of concave obstacles in the form of walls. The ability to generalize using RL is still an ongoing area of research, and the difficulty in doing so is exhibited in our results as well. The limited performance of our simulated robots thus cautions against relying on behaviors trained using RL in safety critical situations. Other approaches with stronger theoretical guarantees may prove easier to implement and more reliable with known failure modes.

VII. Future Work

One place for future improvement would be modifying the reward function such that rewards are not given for humans that initially started in a location that could see the exit and immediately start traveling towards it. This would help the robot to realize from the reward function that we only care about it getting people who cannot see the exit to a location where they can see the exit.

Another place for future improvement would be properly simulating human behavior so that our simulation is more real to life and could better help the humans when implemented in a real world scenario. This could also improve the overall capabilities of our environment. If a real human were to follow a robot but the robot moved too far out of the view of the human, the human would know to try to keep walking in the direction they last saw the robot to either find the robot again or to find the exit that the robot used. This is not currently modeled in our environment but could help the overall effectiveness if actually deployed.

While a very simple combined two policy simulation was demonstrated, more complex combinations of behaviors could be imagined. These may include behaviors that block off unsafe areas or backtrack to find humans that have gotten lost. These behaviors, each with an individually trained policy, could in turn be controlled using a high level decision maker. Because a robot's decision to move towards the exit versus find more humans is based on how many it has found before, utilizing historical information via a recurrent neural network may also improve performance.

References

[1] A. R. Chaturvedi, A. Mellema, S. A. Filatyev, and J. P. Gore, "Dddas for fire and agent evacuation modeling of the rhode island nightclub fire," in *International Conference on Computational Science*, 2006.

[2] N. Pelechano and N. I. Badler, "Modeling crowd and trained leader behavior during building evacuation," *IEEE Computer Graphics and Applications*, vol. 26, no. 6, pp. 80–86, 2006.

[3] J. D. Sime, "Affiliative behaviour during escape to building exits," *Journal of Environmental Psychology*, vol. 3, no. 1, pp. 21–41, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S027249448380019X

[4] P. Robinette, W. Li, R. Allen, A. M. Howard, and A. R. Wagner, "Overtrust of robots in emergency evacuation scenarios," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016, pp. 101–108.

[5] S. Zhang and Y. Guo, "Distributed multi-robot evacuation incorporating human behavior," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, 2013, pp. 864–869.

[6] B. Tang, C. Jiang, H. He, and Y. Guo, "Human mobility modeling for robot-assisted evacuation in complex indoor environments," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 5, pp. 694–707, 2016.

[7] Q. Wang, H. Liu, K. Gao, and L. Zhang, "Improved multi-agent reinforcement learning for path planning-based crowd simulation," *IEEE Access*, vol. 7, pp. 73 841–73 855, 2019.

[8] E. Ronchi, "Developing and validating evacuation models for fire safety engineering," *Fire Safety Journal*, vol. 120, p. 103020, 2021, fire Safety Science: Proceedings of the 13th International Symposium. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0379711220300679

[9] J. E. Almeida, R. J. F. Rossetti, and A. L. Coelho, "Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems," *ArXiv*, vol. abs/1303.4692, 2013.

[10] J. Lee, J. Won, and J. Lee, "Crowd simulation by deep reinforcement learning," *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, 2018.

[11] J. K. Terry, B. Black, A. Hari, L. Santos, C. Dieffendahl, N. L. Williams, Y. Lokesh, C. Horsch, and P. Ravi, "Pettingzoo: Gym for multi-agent reinforcement learning," *CoRR*, vol. abs/2009.14471, 2020. [Online]. Available: https://arxiv.org/abs/2009.14471

[12] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *Journal*, 2016. [Online]. Available: https://arxiv.org/abs/1602.01783

[16] D. Xu, X. Huang, J. Mango, X. Li, and Z. Li, "Simulating multi-exit evacuation using deep reinforcement learning," *Transactions in GIS*, 01 2021.