Intelligent Multirobot Patrol System

Liam Prevelige Mehmet Eren Aldemir

I. Introduction

Autonomous multi-robot systems allow for safe, consistent behavior in situations that might provide challenges for a human to do the same; one such situation, commonly explored by modern robotics researchers, is with patrol systems. In scenarios where physically visible and alert guards provide deterrence, such as providing security to buildings or public utility plants, a multi-robot system can be very valuable. The most basic method for such security is the task of patrolling the area. In many scenarios, patrolling the given area with equally distributed attention may appear to be the most reasonable plan. However, every system contains some weak links that are most likely to be exploited. For this reason, adapting to these weaknesses by patrolling often exploited regions more heavily is expected to provide an advantage over a more simplistic approach. Luckily, computers - particularly autonomous robots - can be programmed to adapt to these circumstances. In this paper, we propose and experimentally test a patrol system implementation in which a security system adapts the behavior of a multi-robot system depending on the area's weaknesses.

II. Related Work

The problem of creating a multi-robot patrol system for detecting opponents has attracted significant attention, with research directed toward exploring various patrol scheme infrastructures (i.e. levels of communication, number of robots) as well as methods of patrolling, effects of diverse topological environments, and differences in levels of opponent intelligence.

Much research has gone toward analyzing the tradeoffs of patrol system infrastructure. Agmon et al. determine that uncoordinated patrol behavior, where robots are unaware of the current path of other robots to a target waypoint, are typically more efficient than systems with higher-levels of communication, while allowing for the patrolling of an arbitrary area with near-optimal behavior [7]. Portugal and Rocha provide an estimate for the minimum number of robots required to patrol a region with each target vertex being visited at a predefined frequency range [3].

There has also been significant experimentation with the patrollers' method of movement. Chevaleyre finds that, in theory, a cyclical movement around a polygon patrol region typically provides an optimal patrol strategy, as opposed to partitioning into subregions [11]. Patrolling behavior has also been subject to Bayesian reasoning, where reward-based learning drives the movement of a system that's aware of historical captures and teammates' current actions [1]. Alternative methods include a division of waypoints using a similar cost function as the former reward-based learning, with the additional implementation of a competitive auction system [6]. A more in-depth analysis of patroller waypoint assignment discovers a distributed, scalable solution for finding paths using Bayesian principles and light communication, resulting in efficient patrolling without repeated visits to target vertices [2].

The effect of dynamic topological environments on patrolling algorithms has also been explored. Basilico, Gatti, and Amigoni find an optimal strategy to explore a topologically diverse world using a leader-follower strategy based on simple, synchronized rotations around the map [4, 9].

Finally, the introduction of smart adversaries —opponents with some knowledge about the state of patrollers— introduces new complexities that disrupt the standard patrol patterns described above. Agmon, Kraus, and Kaminka note that a deterministic patrol scheme with full-knowledge opponents leads to penetration of probability 1, combatting this issue with a non-deterministic patrol scheme; movements are determined with a random component that maximizes the minimal probability of intruder detection [5, 8]. Similarly, Alam et al. note the impossibility of an effective deterministic algorithm with knowledgeable opponents, while also developing a patrol pattern using Markov chains that account for instances where the intruder is capable of hiding using obstacles in the environment [10].

III. Problem Statement

Let us assume a planar environment $E \subset \mathbb{R}^2$. Let us further assume that there are k confined robots in the confinement area $C \subset E$, who are being monitored by n overseer robots to prevent them from escaping. E is bounded and is represented as a two-dimensional occupancy grid. Furthermore, the exact layout of E is known by each of the confined robots and overseer robots. Each confined robot r_i, where $i \in [1, k]$, aims to pass through the doors and reach the safe zone $S \subset E$, where it would be considered to have escaped the confinement area C. Each r_i is considered to be 'caught' if it gets into the field of view of an overseer robot f where $j \in [1, n]$. Field of view of f is defined as having a line-of-sight with the target within a distance d and angle α After being caught, confined robot r_i returns back to C.

Each confined robot r_i prefers the first, second, or third shortest path to the closest point in S to escape C. Which shortest path r_i will use is determined randomly but with respect to the distances of each point, closer doors having a higher chance. In a given time t, each r_i's probability p_l to attempt to leave C is predefined.

Both the confined robots and overseer robots use the same robotic platform, the Turtlebot 3. Turtlebot 3 is equipped with a differential drivetrain and a laser rangefinder that can be used to avoid collisions.

The goal is to create an effective patrolling strategy for overseer robots that will in the end result in a velocity grid containing the velocities to be used in each area of the map.

IV. Proposed Method

It is assumed that the two-dimensional grid G that represents E is already known by all the robots, both confined and overseer robots. In G, a node can be painted in 5 different colors: black, indicating that the node is a wall or obstacle; red, indicating that the node is part of C; white, indicating that the node is part of the patrolling area; green, indicating that the node is the center node of a door; and yellow, indicating that the node is within S. It is assumed that the green nodes are placed once for each door and at the center of the doors, and the doors are wide enough for the robots to pass through.

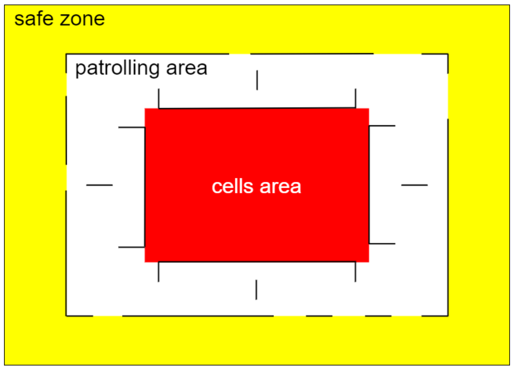

Fig. 1: an example map of the setup

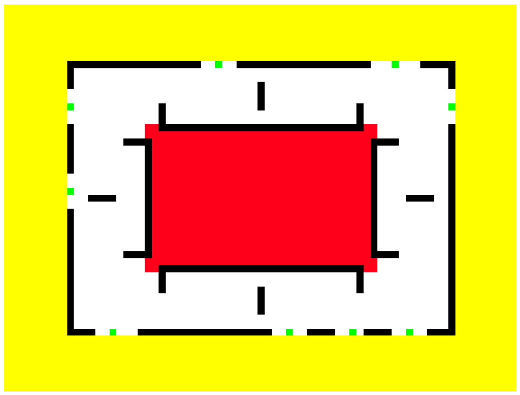

Fig. 2: the approximate two-dimensional grid that corresponds to the map above

A. Confined Robots

When a confined robot r decides to undergo an escape attempt, it will determine the closest point in the occupancy grid that is a door, i.e., a green node, by calculating the Euclidean distance between itself and every other green node. When the closest 3 green nodes are found, it randomly selects one of them with weights based on each door's distance and generates the path to that door using A* search.

Whether or not r will attempt an escape at time t, or more precisely, at the current cycle, is determined by the predefined probability p_l.

If r receives a signal from the security system indicating that it was caught or if it successfully makes it to S, it returns back to its initial spot inside C to continue the simulation.

B. Overseer Robots

i. Baseline

The first approach utilizes a non-adapting, predefined behavior in which overseer robots start at roughly equally spaced points in the patrolling area. Then, they simply follow a circular path in the same direction and at the same speed. The simplified pseudocode for this approach to movement is outlined below; note that detection of escapees and subsequent handling is not outlined. This approach will be used to determine the effectiveness of the Learned approach by comparing the relative number of escaped confined robots.

```
Function BaselineMovement(start_corner, corners[]):
current_corner = start_corner
While the simulation is not stopped:
    follow_shortest_path(corners[current_corner])
    current_corner = current_corner + 1
    current_corner = current_corner % 4
```

(1) Simplified pseudocode for the movement pattern of baseline approach

ii. Learned

In this approach, E is divided into z rectangular regions which are represented as groups of nodes in G, and the scores for each region are kept in a list $L = \{s_0, s_1, \ldots, s_z\}$. All values in L are initially zero. Whenever a confined robot r is captured in a region q, the score of that region, and regions preceding the captured coordinates (i.e. regions the bot first travels through) is incremented by one. At the same time, all regions that have yet to have a robot detected in its area will have its score decremented by one.

```
Function LearnedMovement(start_corner, corners[]):
current_corner = start_corner
While the simulation is not stopped:
    If robot_detected:
        update_velocity_grid()
    follow_shortest_path(corners[current_corner],
                         velocity_grid)
    current_corner = current_corner + 1
    current_corner = current_corner % 4
```

(2) Simplified pseudocode for the movement pattern of Learned approach

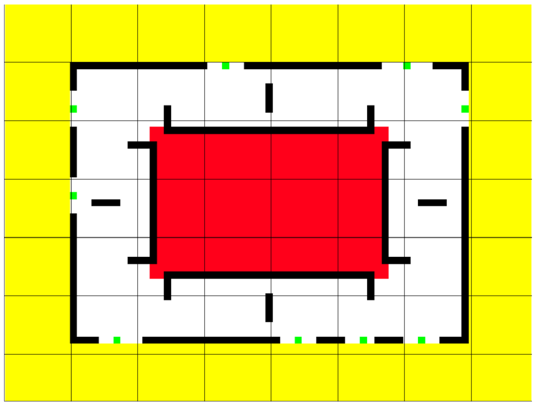

Fig. 3: map divided into subregions, each of which points to a score

Similar to the naive approach, overseer robots start at equally spaced points in the patrolling area and follow a circular path in a single direction. When they enter a region q, their default velocity V will be altered by a constant weight W times the score of the region q, s_q.

Thus, the function function for the speed of an overseer i is expressed simply as follows:

$$(3)\ v_i = V * (W * s_q)$$

The result of this change in velocity is that overseers will spend more time in regions with frequent detection of escapees, thereby improving the chances of detecting future escape attempts.

C. Security System

An overarching node maintains the system's high level of information, paralleling a security system that might exist in a patrol scheme. The responsibilities of the security system are as follows: add all of the confined and overseer robots to a registry, start the simulation behavior, track detections of confined robots by patrollers, notify intruders of detection, and, if using the Learned approach, update the grids according to the procedure outlined in *B.ii*. The security system's responsibility for tracking and notifying escapees of detection is not practical for the real world, but is used to simplify the overseer-escapee interaction in simulation.

V. Implementation

The system was implemented using ROS, and the simulations are conducted using the Stage environment. The robots utilized basic differential drivetrains, and the movement functionality of the robots was implemented using a simple proportional control system with feedback coming from ground truth data of robots. A laser rangefinder was employed to avoid collisions between robots moving within the field. Confined robots are programmed to return back to their starting position when they are caught or successfully escaped to be able to continue attempting other escapes to test the success rates without initializing too many robots that would slow down the simulation.

The circular patrolling pattern utilized by overseers is implemented in a simple way using the A* search algorithm. Coordinates of each of the four corners of the map are hard-coded in the overseer nodes, and they are made to follow the shortest path to each of them in a consecutive fashion.

Detection of confined robots by overseers is implemented within the security node to simplify the implementation. Using the ground truth information about the poses of each robot, the overarching security node constantly calculates and checks if there are any confined robots within the line-of-sight of any overseer robots. The field of view of overseers is limited to 4 meters within a 60 degrees angle toward the front. Furthermore, cases that include the obstruction of view by walls are also included in the calculations.

The actual method of calculating visibility depends on following the approximate path of a line between the patroller and each confined robot. If the patroller is too far from the confined robot, facing the wrong angle, or the line between the two intersects a wall, then visibility is blocked.

The simulation time is tracked to conduct consistent experiments. When a confined robot *r* has escaped or caught, the simulation time counter and the motion of all robots except for *r* are paused. When *r* successfully returns back to its initial position inside the confinement area, the simulation is continued from where it was paused.

The interactions between the security, overseers, and confined robots depend on the use of several subscribers, publishers, and services. The security system receives the locations of every patroller and confined robot (primarily used to determine visibility and escapee detection) using a subscriber. Along with sending their positions, all nodes register with the security system. The pausing and stopping of the simulation, sending of velocity, hearing of escape, notification of detection, and publishing of a velocity map (for the learned implementation) all depend on this registry. A number of services are set up as a result of registrations. When a confined robot is detected or escapes, custom messages are passed between the confined bot and security system to return back to starting position, while having all other robots stay idle. In addition, when a confined robot escapes, we look to reduce the number of collisions by publishing the locations of all confined robots to be temporarily designated as a 'wall' during A* search. Finally, the use of a 'velocity grid' indicates the required velocity at a given point on the map when using the Learned implementation. Every point on the map has a respective index in a velocity grid which is passed to each patroller at the start of the simulation, as well as following the detection of an escapee. The heavy use of services maintains robustness to latency and adjustments to implementation (i.e. switching between baseline and Learned).

VI. Experimental Results

A. Setup

The success of the learned algorithm relative to the baseline (non-learning) algorithm will be determined using extensive simulation tests in the Stage simulator.

Both the overseers and confined robots are simulated using the TurtleBot 3.

All simulations are conducted on the map depicted in the proposed section method. The layout of the map is given as an occupancy grid to all nodes before each round of simulation. A fixed number of confined robots are used in each round of simulation, with the number of confined robots equal to the number of exits of the confinement area *C*. In the preliminary map, there are four exits in *C*, so four confined robots are used, each positioned a small distance from one of the exits. The number of overseers ranges from 2-5 robots for each round, with the total simulation testing time fixed at 3minutes for both the baseline and learned algorithm. For the learning algorithm, training will occur before the testing period with a duration ranging from 0-5 minutes. For all units of time used in the experiment, simulation time, rather than real-world, will be used.

The expected initial linear velocity to be used in the simulation for robots with no nearby obstacles is 0.5 m/s for both confined robots and overseers. The baseline algorithm will use this velocity constantly. The learned algorithm has this velocity contained within the occupancy grid; upon the capture of a confined robot for the learned approach, the velocity values for relevant sections of the grid are either reduced or increased by 20%. A minimum and maximum velocity for sections of the grid is 0.2 m/s and 1.25 m/s respectively.

B. Simulation Testing Procedure

A number of tests are used in simulation to test the patrolling algorithm while comparing the effectiveness between the baseline and learned algorithm. First, the baseline algorithm is executed with 2, 3, and 5 overseer robots for 3 minutes of simulation time - the simulations for each count of overseers is run 3 times, with the key metrics (e.g. detected vs total attempts) averaged across the runs. Although inconsistent, the variability in increase of counts of patrollers is used to maintain a relatively consistent percentage increase between simulations without overly cluttering the patrolling area. During the simulations, when a confined robot escapes or is caught, the rest of the robots stop moving, the tracked simulation time is paused, and the confined robot moves back to their initial position at the respective corner of the confinement area. The number of times the confined robots escape and the number of times they are caught is recorded.

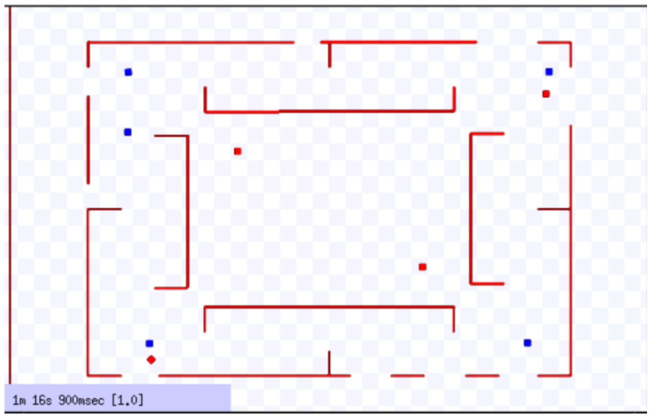

Fig. 4: Example simulation with 4 confined (red) and 5 patrollers (blue) mid simulation

An example of a simulation with 5 overseers and 4 confined robots is shown above; this simulation has been running for 50 seconds, with two confined robots attempting escape - both of which were ultimately detected.

A similar configuration is tested with the learned approach. The learned approach simulations only differ from the baseline by having a preliminary training period, during which the overseers can have their moving behavior adjusted without tracking key metrics. The learned approach is tested with the following durations of preliminary training time: 0 minutes, 2.5 minutes, and 5 minutes. When the training period ends and the testing period begins, we keep track of key metrics while preserving the trained grid with adjusted velocities. Regardless of the duration of training time, 3 minutes of testing is used to record the results.

The results are monitored throughout the simulations by making note of the ratio of caught confined robots to total escape attempts. For the baseline approach, this ratio is averaged for every set of simulations with a given number of patrollers. For the learned approach, this ratio is averaged for every set of simulations with a given number of patrollers *and* amount of training time.

C. Simulation Testing Results

Over the course of the testing process, occasional collisions were discovered between overseers and confined robots with each other or walls; although extensive measures were taken to strengthen robustness against problems with collision, they still proved to be an issue under specific circumstances due to the complexity of some movement patterns and sensor errors. When these collisions did occur, one of two courses of action were taken: (1) If the confined robot was attempting to escape the confined area, the simulation and key metrics were reset. (2) If the confined robot was attempting to return to its starting position, this action was completed manually through a 'drag and drop' in Stage.

Below, figures 5 & 6 depict the performance of the two patrol implementations both within their own set of simulations and comparing between the two. The key metrics from the simulation generally indicate the ratio of detected escapees to total attempted escapes is higher for the learned algorithm than the trained - this difference is especially noticeable given higher levels of training time.

Within the respective simulations for the baseline and learned approach, we found that the ratio of escaped robots to total attempts was relatively consistent for simulations with the same number of overseers and —for the learned approach— training time; however, slight variations were found due to the random escape behavior of confined bots. Fig. 5 suggests that while differences between the ratio of detections to escape attempts exist, the magnitude of this difference is relatively small.

For both the baseline and learned approach, a higher number of overseers was clearly correlated with a higher ratio of detections to attempts, as expected; this change can be observed through the difference in magnitude of the ratio between columns for the baseline implementation, and between rows for the learned.

For the learned approach, the success rate also increased with training time. Increased training time allowed overseers to adjust their speeds —with quality of the adjustments increasing with training time— in advance of the testing period. This change in performance is visible in Figure 6, with a clear upward trend in the ratio of detected to total attempted escapes as training time increases. Although there are maximum and minimum limits for the velocity a learning overseer robot can move, the duration of simulations made it rare to ever reach this limit. In addition, although we expected to observe decreasing returns to training time, this pattern was not apparent in our simulation tests—still, it may become visible with further testing. In addition, there is also the risk of poor performance using the learned approach since, in the worst case scenario, confined robots will randomly happen to attempt escape/be detected at one of the four corners for some initial period of time, increasing the chance that successful escapes occur in the other corners due to an increase in velocity; fortunately, this was not observed in simulation testing.

Ultimately, the learned approach tended to yield a higher success rate in detection of escapees, although the difference was not extreme. For longer simulations and increased training time, we would expect the difference in the performance of the learned vs baseline approach to increase.

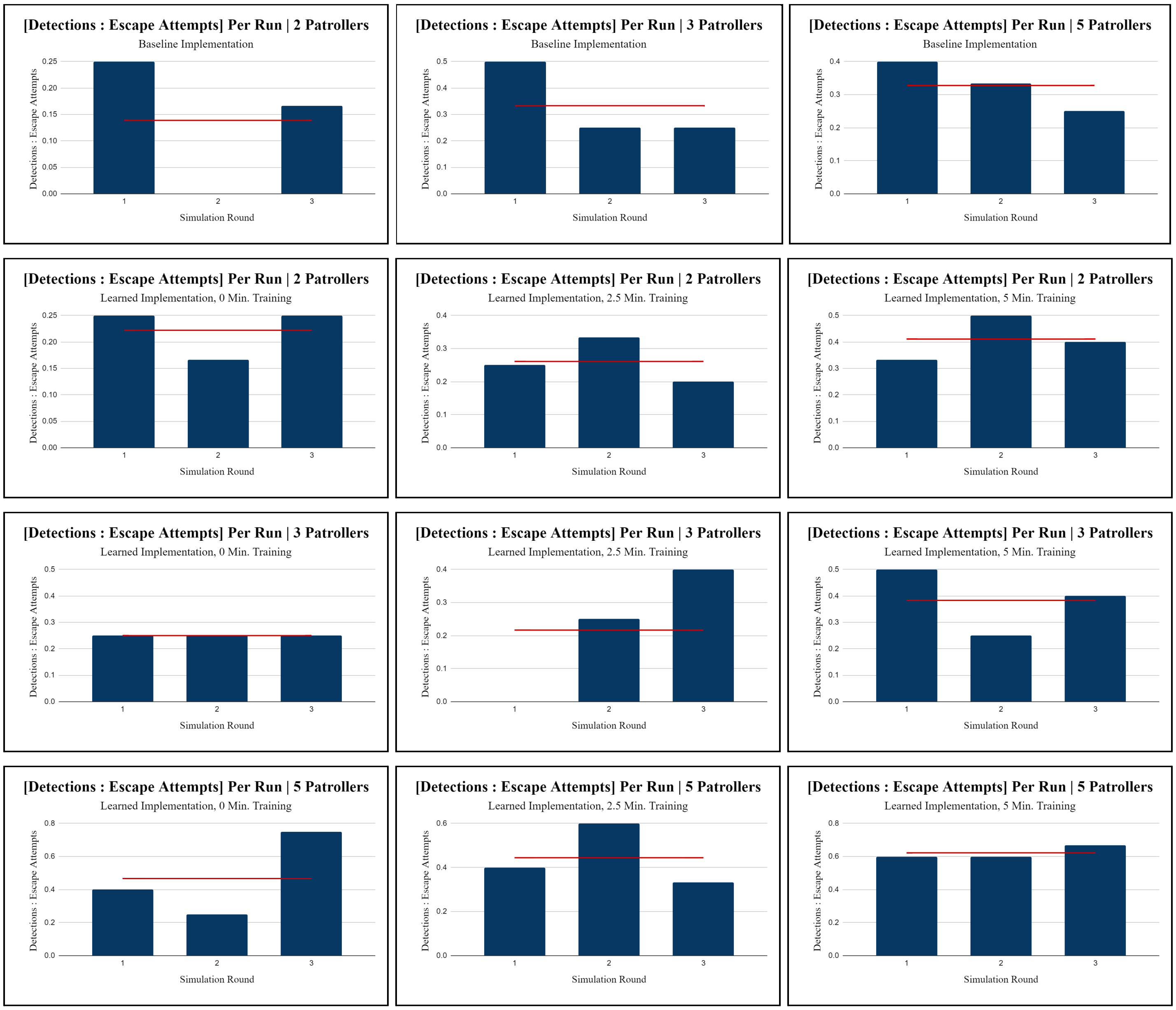

Fig. 5a-5l: The ratio of detections to total escape attempts for all combinations of simulations to be averaged; 5a-5c deal with baseline, 5d-5l show each set of simulations given training time.

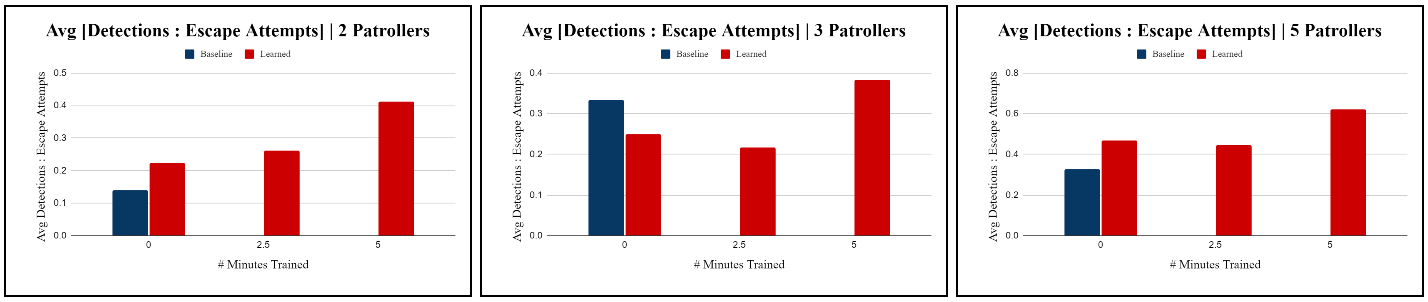

Fig. 6a-6c: The average ratio for detections to total escape attempts for the baseline vs each set of learned training.

VII. Conclusions

In this paper, we presented two methods of patrolling an area for the detection of robots trying to escape from a confined area. One approach involves a consistent rotation of patrollers around the confined area at a constant velocity. The second approach involved a similar rotation pattern, with the velocity of patrollers changing in regions where confined robots are more frequently detected; to some extent, this method of patrolling proved to be more effective in detecting escapees.

A number of follow-up research can be conducted to validate the results of this experiment, improve upon the learned patrolling pattern, and extrapolate applications of the learned approach to new scenarios.

First, further simulations can be conducted that are longer in testing and produce greater variation with starting patroller velocity, percent change & boundaries for velocity in the learned approach, detection distance, movement error margins, and PID values. Doing so can manipulate the effectiveness of patrolling behavior while optimizing movement to minimize collisions. In addition, the edge cases in which collisions did occur, resulting in the scrapping of simulation data or physical movement of the confined robots, might have biased the findings of this paper in unknown ways. Improving upon collision detection, perhaps through the assumption that ground truth information can be shared to detect robots that are nearing collision, may change findings.

Future work can also expand upon the specific implementation of patrollers and confined robots in this paper.

Significant research has been conducted on patrol approaches with escapees that are aware of the overseers' positions and patrol habits. The current implementation of the escapees can be extended to have knowledge about the patrollers while modifying the overseer movement with non-deterministic velocity and directional components.

Manipulating the obstacles, confinement exits, and overall structure of the simulation environment can yield new findings for the patrol implementation. Testing the current implementation of patrollers and confined robots on new maps can create interesting comparisons with this paper's findings.

Finally, related research outlines a number of other patrolling mechanisms that can be implemented, including changes in level of communication and deviation from a straight, cyclical path throughout the outer edge of the map.

Ultimately, this paper looks to explore a new method of smart patrolling for a multi-robot system. Creating effective methods of patrol can allow for a number of real world applications, including monitoring the effect of environmental changes (i.e. migration patterns), creating strong security systems, and creating low-infrastructure approaches to learned movement patterns. This paper also sets a strong foundation for extrapolating patrolling behavior to more complex scenarios, as earlier outlined. In the process of exploring patrol systems, we urge researchers to be mindful of ethical questions regarding such a situation, but ultimately believe an effective implementation of an intelligent patrol system would provide significant benefits in the modern world.

VIII. References

[1] D. Portugal, , R.P. Rocha "Cooperative multi-robot patrol with Bayesian learning," Autonomous Robots 40, 2016, pp. 929–953, doi: 10.1007/s10514-015-9503-7.

[2] D. Portugal and R. P. Rocha, "Decision methods for distributed multi-robot patrol," 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2012, pp. 1-6, doi: 10.1109/SSRR.2012.6523869.

[3] D. Portugal and R. P. Rocha, "Performance Estimation and Dimensioning of Team Size for Multirobot Patrol," in IEEE Intelligent Systems, vol. 32, no. 6, pp. 30-38, November/December 2017, doi: 10.1109/MIS.2017.4531222.

[4] F. Amigoni, N. Basilico and N. Gatti, "Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments," 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 819-824, doi: 10.1109/ROBOT.2009.5152497.

[5] G. Kaminka, S. Kraus, “Multi_robot Adversarial Patrolling: Facing a Full-Knowledge Opponent,” Journal of Artificial Intelligence Research 42, 2011, pp. 887-916. https://arxiv.org/ftp/arxiv/papers/1401/1401.3903.pdf

[6] K. Hwang, J. Lin and Hui-Ling Huang, "Cooperative patrol planning of multi-robot systems by a competitive auction system," 2009 ICCAS-SICE, 2009, pp. 4359-4363.

[7] N. Agmon, C. Fok, Y. Emaliah, P. Stone, C. Julien and S. Vishwanath, "On coordination in practical multi-robot patrol," 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 650-656, doi: 10.1109/ICRA.2012.6224708.

[8] N. Agmon, S. Kraus and G. A. Kaminka, "Multi-robot perimeter patrol in adversarial settings," 2008 IEEE International Conference on Robotics and Automation, 2008, pp. 2339-2345, doi: 10.1109/ROBOT.2008.4543563.

[9] N. Basilico, N. Gatti, F. Amigoni, “Leader-follower strategies for robotic patrolling in environments with arbitrary topologies,” 8th International Joint Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 57-64, doi: 10.1145/1558013.1558020.

[10] T. Alam, M.M. Rahman, P. Carrillo et al, “Stochastic Multi-Robot Patrolling with Limited Visibility,” J Intell Robot Syst 97, 2020, pp. 411–429, doi: 10.1007/s10846-019-01039-5.

[11] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004)., 2004, pp. 302-308, doi: 10.1109/IAT.2004.1342959.