Deep Reinforcement Learning and Object-Tracking for SmallSat/UAV Coverage Path-Planning

Jonathan Ponniah, Varsha Thirumalai, Milind Patil
Department of Electrical Engineering
San Jose State University

B. G. Lewis, A. K. Wang, T. Chang, M. Enoch, J. Pandya
Lockheed Martin
Sunnyvale, CA

***Abstract*—The effectiveness of UAVs deployed in commercial, civil, and tactical missions has driven significant research in autonomous agent decision making. As swarm size and beyond line-of-sight distance grows, decentralized motion planning algorithms are critically needed to control and coordinate the position and velocity of each individual agent in the swarm. In this paper, we apply a deep reinforcement learning (DRL) path planning algorithm [1] and integrate multi-object tracking (MOT) processing. Publicly available images of first-responder/natural disaster scenes are used as an exemplar to demonstrate the performance of the Deep Affinity Network (DAN) MOT front-end [2]. The automated point-of-interest target detection, identification, and classification results from DAN are used to drive the input observation state and target mapping required by the convolutional neural network. Monte Carlo simulation and analytical lower bounds are developed to evaluate algorithm performance for coverage path planning. Applicability of the proposed approach to enable communications and delivery of physical goods and services *at scale* in both civilian scenarios and contested theaters is described.**

***Index Terms*—Deep Reinforcement Learning, Autonomy, UAV, UUV, Satellite Communications**

I. Introduction

SmallSat, UAV, and UUV swarms have disrupted cyber-physical boundaries, enabling higher resolution sensing and last-mile wireless communications and networking in contested theaters [3]. They are also used in applications that deliver physical goods and services. Examples of such applications include clearing resident space objects (RSOs), mine sweeping, battlefield/underwater countermeasures, providing humanitarian aid/supplies to first responders and military personnel, and applying fire retardant to contain wildfires [4]. These applications share a common objective of tracking/covering "interest-points" or targets over some operating theatre or spatial grid. Interest-points correspond to RSOs, red/blue soldier platoons, first responders, civilian war-displaced camps, explosive mines, or burning trees. In all cases, the goal of the swarm is to visit (or cover) the maximum number of interest points in the shortest possible time.

We focus on two important issues in swarm tracking/coverage: enabling drones to identify interest-points from their observations (i.e., images/data from onboard cameras and sensors) and computing trajectories for each drone that collectively cover the interest-points on a two-dimensional grid (i.e., motion-planning). We propose an integrated solution that combines a multi-object tracking (MOT) front-end processor for target/interest-point detection, with a control policy derived via deep reinforcement learning (DRL) for planning the trajectories of each agent/drone in the swarm.

The solution we propose for MOT involves training a Deep Affinity Neural Network (DAN) with an image dataset of damaged buildings taken in the aftermath of earthquakes. We write python scripts to pool these images from the internet, and use labeling tools to process the data and obtain exemplars of healthy and damaged buildings at consistent resolutions. This approach avoids the difficulty of creating a dataset from images taken single-handedly at each scene. The DAN scores an accuracy of 95% on the test data, confirming that it is effective at detecting multiple objects within single frames in real-time.

We develop a multi-agent motion-planning control policy using DRL to send agents/drones to the interest-points identified by the MOT front-end (i.e., the DAN). One challenge in multi-agent tracking/coverage is that the state-space (i.e., the positions of all the drones and interest-points) grows exponentially in the number of agents, interest-points, and grid dimension. This makes it difficult to train the control policy or compute theoretical performance limits to compare with the actual system performance. To reduce the complexity of the state-space we decompose the observation space into local and global maps centered at each agent. The global map is downsampled, which reduces the number of inputs into the control policy and also the communication bandwidth needed for agents to arrive at a common view of the state-space. This way of decomposing the observation space relies on the hypothesis that "locally-interactive" structure exists in the state-space of multi-agent tracking/coverage systems; each agent only requires precise views of the local map and downsampled views of the global map for the system as a whole to collectively achieve (approximately) optimal performance.

Another challenge in the multi-agent setting is that the action-space grows exponentially in the number of agents, which also makes the training process computationally infeasible. To avoid this problem, we train the control policy at the level of an individual agent where the action-space is fixed in size, but allow each agent to observe the (downsampled) global map of interest-points and agents. The idea is to endue the control policy with an internalized awareness of how

other agents in the swarm behave. We show the control policy applied to each agent in the swarm induces "emergent" coordination in simulation, where agents leave interest-points in their vicinity, in favor of remote interest-points not easily accessible to other agents; behavior that is locally sub-optimal but globally optimal.

Multi-agent coverage/tracking is a more general version of the Traveling Salesman Problem (TSP), with one agent and a sequence of interest-points fixed on a two-dimensional grid. Solving the TSP requires an exponential number of computations in the number of interest-points. The general multi-agent coverage problem inherits this complexity on a more severe scale, which makes it difficult to compare actual performance of a control policy with its fundamental limits. We introduce a computationally feasible lower-bound on the shortest paths, the max-min completion-time and show that the control policy derived via DRL, tracks this bound over a variety of grid sizes.

A. Use Cases

The UAV first responder/natural disaster mission is the specific use case demonstrated in this paper but our approach can be also applied to UAV tactical missions. For example, in a UAV tip-and-cue operation, the blue-force swarm UAVs must search for and identify an adversary point-of-interest (such as a weapons cache) and provide live detailed visual evidence. Optimizing geographical coverage, time-of-arrival, and resource allocation across UAV agents the system is needed to achieve the mission. Effective object detection and coordinated agent path planning across agencies and allied nations becomes increasingly important for complex Joint All Domain Command and Control (JADC2) missions.

Our approach can also be applied to space-based autonomous maneuvering swarm missions for earth sensing, debris mitigation, and the exploration and industrialization of space. An example use case is the Hyper-Angular Rainbow Polarimeter (HARP) cubesat recently launched by NASA [5]. The planned NASA cubesat swarm will use artificial intelligence to self-determine position to optimally observe important weather markers. Swarm AI autonomously changes cubesat flight paths and altitudes, moving relative to each other as needed, to study weather events around the globe. A longer-term objective is the use of spacecraft swarms for asteroid exploration for mineral mining and earth defense [6]. Swarms of cubesats use spectrometers to measure the presence of minerals such as uranium or ice/water or hydrocarbons for potential use as resources. Decentralized coordinated coverage and obstacle avoidance is needed to complete the operation without incurring the delay of earth-originated command and control.

B. Outline

Sections II and III discuss the DAN architecture used in the MOT front-end and its performance. Section IV describes the DRL approach used in multi-agent motion-planning and introduces a computationally-feasible lower-bound on the completion-time as a performance metric for the system. Section V discusses the simulation results of the control policy derived via DRL. Section VI concludes the paper and summarizes our findings.

II. Methodology: Multi-Object Tracking

The Deep Affinity Network (DAN) Multi-Object Tracking (MOT) [2][9] is a computer vision algorithm which classifies and detects points-of-interest in every frame of the camera and then tracks the interest-points by associating the images across multiple frames of the video. DAN uses a convolutional neural network to extract the features of objects of interest in the frame and the applies the Hungarian algorithm to perform object tracking across frames. Objects that leave and then return to a scene are tracked by searching forward and backward through frame memory. DAN simultaneously models the objects appearance and computes the inter-frame object affinities. Standard metrics to evaluate object tracking performance include the alignment of the predicted bounding box and the ground truth, the number of false positives/negatives, the number of identity switches, and the number of tracked/lost targets.

We use DAN MOT image tracking software as the automated visual front-end to drive the input observation state required by our proposed SmallSat/UAV coverage motion planning algorithm. In a fielded mission, DAN uses the images collected from downward facing cameras mounted on the aerial vehicle to perform automated interest-point target detection, identification, classification, and tracking. The results are mapped on a square grid world and provided to the path planning algorithm convolutional neural network.

A. Generating a Labeled Data Set

A good quality dataset has sufficiently many images at the same resolution of the drone cameras/sensors, with labeled exemplars of healthy and damaged buildings. It is generally difficult to obtain every image from a single source; images tend to come from multiple scattered sources since disasters strike at random times and places. To create the dataset, we developed a python script to search the web for images of buildings damaged by earthquakes (other languages like "R" are equally capable of collecting such data). We then used the open source VGG Image Annotator (VIA) tool [10] to label healthy and damaged structures, manually selecting images at consistent resolutions. An example of the image labeling is depicted in Figure 1.

III. Simulations and Results: Multi-Object Tracking

Our objective is to collect and label 500 independent and representative images of intact (undamaged) structures and the associated post-natural disaster damaged structure image of equivalent image resolution. DAN MOT training is performed on 300 pairs of images which are painstakingly labelled with bounding boxes allowing VIA to generate the corresponding pixel metadata needed to establish learning reinforcement truth.

(a) (b)

Fig. 1: Generating labeled datasets of damaged and healthy buildings using VIA software. (a) An image of a healthy building prior to a disaster [7]. (b) The same building after an earthquake [8].

The in-progress results to date shown in this draft submitted paper use 80 labelled intact and earthquake damaged modern low to mid-rise buildings as a proof-of-concept, with full acknowledgement of interim overfitting. The final camera-ready paper, if accepted, will show DAN MOT results for a properly fitted sufficient number of images, where we expect a correctly tracked rate of 95%.

IV. Methodology: Deep Reinforcement Learning

We describe the DRL approach to path-planning, and the architecture of the neural net used to train the control policy. In addition, we introduce a lower-bound on the performance of this policy since it is infeasible to compute the optimal performance analytically.

A. The Markov Decision Process

The basic reinforcement learning setup is defined by the tuple $(\mathcal{S}, \mathcal{A}, R, O, \gamma)$ where $\mathcal{S}$ represents the system state-space, $\mathcal{A}$ the action space, $R : \mathcal{S} \to \mathbb{R}$ a reward function that assigns value or preference to each state, O an observation function $O : \mathcal{S} \to \mathcal{O}$ that determines the information available to each agent from the global state $\mathcal{S}$, and γ a discount parameter used to compute the present value of expected future rewards. Let D denote the set of all drones (i.e., agents) and I, the set of all interest-points. The global state $\mathcal{S}$ is defined below:

$$\mathcal{S} := \mathbb{N}^{2\times|D|} \times \mathbb{N}^{2\times|I|}. \tag{1}$$

At each time t, the state $s(t) \in \mathcal{S}$ is defined by $s(t) := (\mathbf{D}, \mathbf{I})$ where $\mathbf{D} := \{(x_d, y_d) : d \in D\}$ is the set of all agent positions, and $\mathbf{I} := \{(x_i, y_i) : i \in I\}$ is the set of all interest-point positions. Each set of coordinates (x_d, y_d) and (x_i, y_i) represent pairs of natural numbers corresponding to a grid cell. The actions of each agent are defined as $\mathcal{A} := \{\text{up}, \text{down}, \text{left}, \text{right}\}$. The action space of the system is then $\mathcal{A}^{|D|}$ where each $a(t) \in \mathcal{A}^{|D|}$ is defined as $a(t) := \{a_d \in \mathcal{A} : d \in D\}$. The reward function assigns a positive reward r_p for each agent in the same cell as a previously unvisited interest-point and a negative reward r_n for each agent in a cell without a previously unvisited interest-point.

B. Restructuring the Observation Space

Since the action space and state space grow exponentially in the number of agents, the conventional single-agent DRL approach does not extend directly to the multi-agent scenario. A different approach is required that is both scaleable and preserves the ability of agents to coordinate their activity. To address this problem, we propose two strategies.

1) *Decomposing the Observation Space into Local/Global Components.* Even though multi-agent tracking in principle, is far more complicated than the TSP, the simplifying premise is that each agent only requires precise knowledge of the interest-points in its immediate vicinity and coarse/aggregate knowledge of remote interest-points.

We decompose the observation space of each agent into two components as described in [1]. The first component is a local sub-grid centered at the current position/cell of the agent (and appropriately zero-padded in case the agent is close to the grid boundary). The local grid includes the positions of the agents and interest-points in the vicinity of the agent. The second component is the global grid, also centered and zero-padded at the position of the agent, that includes the positions of all agents and interest-points. Each map is passed into separate convolutional layers and then concatenated into a fully connected network.

2) *Inducing Scaleable Multi-Agent Coordination.* To enable coordination between agents while avoiding exponentially increasing action spaces, each training "experience" generated during a simulation step is defined by the following set of tuples:

$$\{(s_d(t), a_d(t), r(t), s_d(t+1)) : d \in D\}, \tag{2}$$

where $s_d(t)$ is the decomposed observation space (described previously) for agent d at time t, $a_d(t)$ is the action taken

by agent d at time t, $r(t)$ is the system reward computed at time t, and $s_d(t+1)$ is the decomposed observation space in time $t+1$ after action $a_d(t)$ has been taken. The tuple set in (2) is stored in the experience replay buffer and randomly sampled during training.

The resulting control policy is common to all agents and thus induces an internalized awareness of the behavior of other agents in response to the observed state. However, the actions produced by this policy apply to individual agents in sequence so the size of the action space is constant in the number of agents.

C. The Double Deep Reinforcement Learning Architecture

The basic DRL architecture proposed in [1] involves two neural networks: an online network parameterized by $\bar{\theta}$ and a target network parameterized by θ. The target network evaluates the policy whereas the online network determines the next action of the policy. The online network is updated more frequently than the target network to make the training process less sensitive to "noisy" experiences. This interleaving of networks is designed (along with the experience replay mechanism) to improve convergence.

As a general rule-of-thumb, reinforcement learning falls into one of two categories: q-learning or actor-critic/policy-optimization (some approaches straddle the divide and attempt to incorporate elements from each category). We adopt a q-learning approach that finds the following so-called state-action value function of a policy π:

$$Q^{\pi}(s(t), a(t)) = \mathbb{E}_{\pi}\left[\sum_{k=t}^{\infty} \gamma^{k-t} R_k\right], \tag{3}$$

where R_k is the reward accrued in time-slot k. The loss function used to train the online network is given by:

$$L(\theta) = \mathbb{E}_B\left[(Q_{\theta}(s,a) - \hat{Q}(s,a))^2\right], \tag{4}$$

where the expectation in (4) is taken over a random batch of experiences B sampled from a replay buffer and the experience (s, a, r, s') is the state-action-reward-state tuple defined in (2). The target $\hat{Q}(s,a)$ is defined:

$$\hat{Q}(s,a) = r + \gamma Q_{\bar{\theta}}(s', \arg\max_{a'} Q_{\theta}(s', a')), \tag{5}$$

where (s, a, r, s') is the experience defined in (2). After a parameterized interval of steps, the online network is cloned into the target network: $\theta \rightarrow \bar{\theta}$. Figure 2 depicts screenshots from a grid-world simulation with trajectories traced for certain agents. We will revisit Figure 2 in Section V.

D. Performance Metrics

In military applications, knowing how far a system performs from the theoretical optimal, can aid in risk management, tactical deployment, and anticipating worst case scenarios with respect to adversarial capabilities. Unfortunately, multi-agent coverage problems are too difficult to solve analytically.

We introduce a computationally feasible lower-bound on the shortest time required for the drones to visit all interest-points. This lower-bound is the "max-min" completion time; the minimum time needed for some drone to visit a fixed interest-point, maximized over all interest-points. The max-min completion time, as a performance metric, gives some indication of how far the system is from the optimal. Let $p(d) := (x_d, y_d)$ and $p(i) := (x_i, y_i)$ denote the positions of drone $d \in D$ and interest-point $i \in I$ respectively on the grid. Let $t_i(d) := |x_d - x_i| + |y_d - y_i|$ denote the number of steps required for drone d to visit interest-point i. The max-min completion time t_{opt} is:

$$t_{opt} := \max_{i \in I} \min_{d \in D} t_i(d). \tag{6}$$

Figure 3 depicts a series of plots comparing the actual completion-times achieved in simulation with the max-min completion-times defined in (6). We will discuss Figure 3 in Section V. The accuracy or "tightness" of the max-min bound in (6) depends on factors such as the ratio of agents to interest-points, the size of the grid, and whether or not the interest-points are mobile. Intuitively, (6) works well when the number of drones and interest-points are approximately the same and the interest-points are fixed. This bound is looser when there are many more interest-points than drones. To the best of our knowledge, (6) is the first proposed low-complexity lower-bound for the multi-agent coverage problem, so more investigation into other possible lower-bounds is needed.

V. Simulations and Results: Deep Reinforcement Learning

The DRL approach in Section IV was used to train a control policy over random configurations of drones and interest-points. We will discuss the general criteria used to evaluate the overall effectiveness of our approach and explain what the simulations reveal. The max-min performance metric in (6) provides concrete benchmarks that will assist our analysis.

1) Evidence of Coordination. To reduce the complexity of the action-space, we train each drone on the same control policy (i.e., a local control policy) but provide each drone with global knowledge of the state space (i.e., the positions of the other drones and interest-points). This strategy fixes the size of the action space but gives each drone some awareness of how other drones behave and react. It is not obvious a priori if this "internalized awareness" actually leads to genuine coordination between drones. Examples of such coordination are spontaneous "divisions of labor" where a drone chooses to visit a remote interest-point over a local interest-point when other drones are in the vicinity. This prevents two drones from converging on the same interest-point and requires drones to make locally sub-optimal decisions in favor of globally optimal ones.

Figure 2 shows a series of time-step snapshots taken from a simulation of five drones and interest-points on a 10x10 grid. These snapshots illustrate the division-of-labor expected in coordinated multi-agent systems. In this example, a drone chooses to bypass a closer interest-point (to the left) in favor of a more remote collection of interest-points (underneath). Coordination usually requires higher-

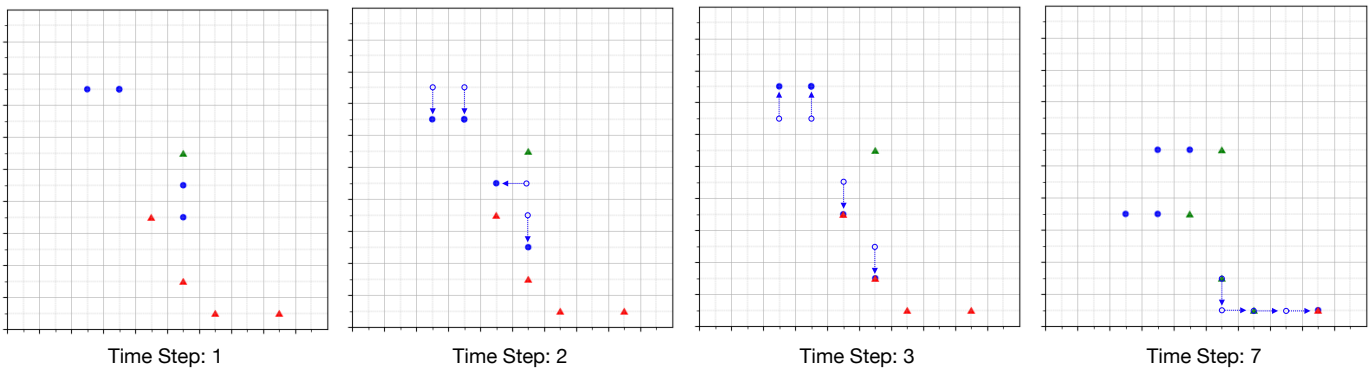

Fig. 2: A sequence of time-steps taken from a grid-world simulation of five agents (circles) and five interest-points (triangles). The un-visited and visited interest-points are red and green respectively. Time-steps 1-3 show an agent bypassing an interest-point one step to the left in favor of an interest-point two steps down. Although locally sub-optimal, this decision leads to faster completion, because another advancing agent is in the vicinity.

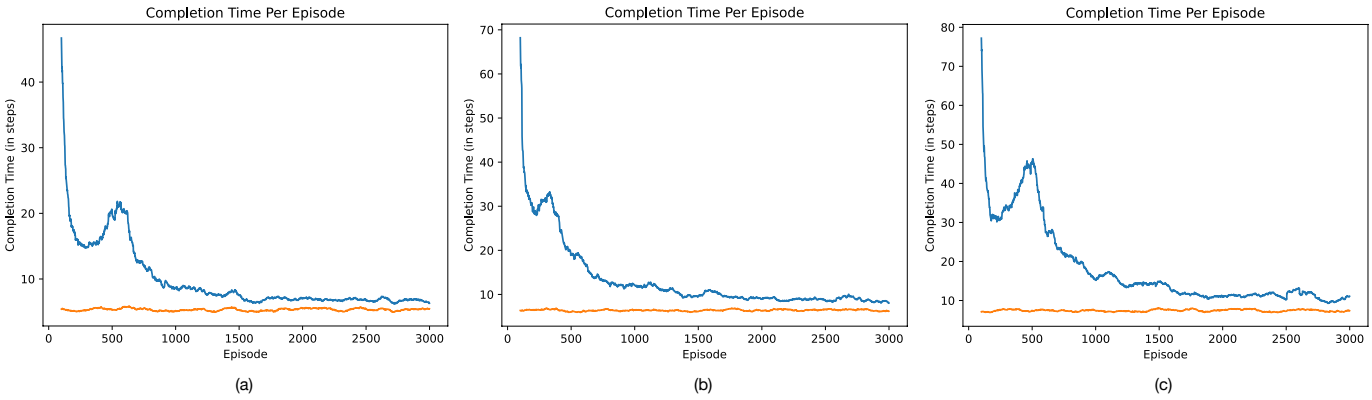

Fig. 3: Comparing optimal and actual completion-times for different grid sizes with 5 agents and 5 interest-points. (a) A 10x10 grid. (b) A 12x12 grid. (c) A 14x14 grid. Orange curves represent the max-min completion-time. Blue curves represent actual completion-times achieved during simulation. The actual and optimal completion-times converge as the number of training episodes grows large.

dimensional action-spaces that define the actions of clusters or groups of agents. Figure 2 is an example of coordination emerging from the control policy (to an extent), when the agents are trained on the same policy and have knowledge of the global state. This training strategy could be further enhanced in future work by including agent identifiers in the input to the policy with explicit inter-agent communication.

2) The Performance vs Complexity Trade-off. Our running hypothesis is that multi-agent tracking problems have locally-interactive structure: drones only require precise local state information but downsampled (i.e., compressed or aggregated) global information for the system as a whole to achieve approximately optimal performance. We exploit this structure (to reduce the complexity of the state-space), by slicing the grid into global and local maps. Figure 3 shows a series of plots comparing the max-min and actual completion-times curves with respect to the number of training episodes for different sized grids.

The curves converge as the number of episodes grows large. In the 10x10 grid, the actual completion-times stay approximately within 50% of the max-min lower-bound. This performance is encouraging, since the total number of possible initial states in this setup is 10^{20} as given in (1). A small fraction of these states are sampled in the training process (the axis spans 3000 randomly initialized training episodes). Yet the actual completion-times closely track the max-min upper-bound. This supports the hypothesis that multi-agent coverage problems have locally-interactive structure.

The gap between the max-min lower-bound and actual completion-time appears to increase for larger grid sizes. A possible contributing factor is that random initial policies get significantly worse with large grids; exponentially more states must be explored in each episode. One way to mitigate this decline (in future work), is via hard-coded heuristic policies that guide the initial exploration and serve as starting points for further optimization.

3) The Effectiveness of Down-sampling Techniques. The global map is down-sampled to reduce both the training complexity and communication bandwidth needed for disseminating local-state information. Averaging and max-pooling

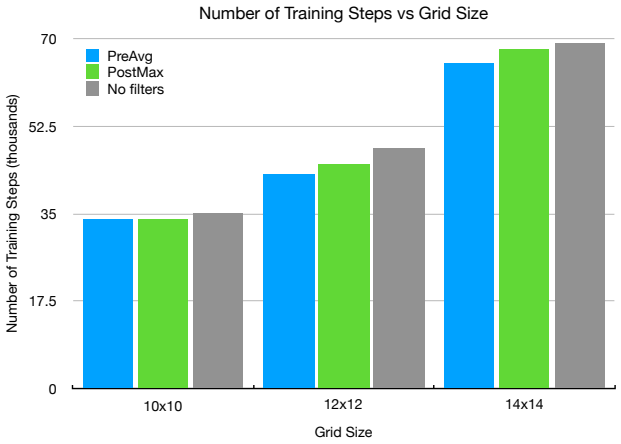

Fig. 4: The number of training steps required for different grid sizes and downsampling techniques in a system with 5 agents and 5 interest-points. Blue bars indicate an averaging filter applied to the global map, green bars indicate a max filter applied to the output of the convolutional layers, and grey bars indicate no downsampling.

are common ways of downsampling in convolutional neural networks. The averaging filter is a natural choice for multi-agent tracking, since the average number of drones and interest-points in any given region is clearly relevant to the control policy. However, max-pooling is often preferred in image-processing, because it identifies features more decisively. The choice, size and stride of these filters are design parameters that can only be tuned and refined through simulation. The goal is to have a "smooth transition" between local/precise and remote/down-sampled observations, so that the performance and complexity of the system does not precipitously drop for large state-spaces.

We performed three sets of simulations over 10x10, 12x12, and 14x14 grids with five agents and five interest-points. Each set involved three simulations with different filtering techniques: an average filter applied to the global map, a max-pool filter applied to the output of the convolutional layers, and a neutral architecture with no downsampling. The average and max-pool filters were 2x2 sized with a stride length of 2. Figure 4 depicts the outcome of the simulations with respect to the number of training steps. The simulations show a noticeable but modest reduction in the number of training steps (i.e., complexity) after downsampling. No appreciable change in performance was observed with respect to the completion-times. Although downsampling reduces the training complexity, more investigation is needed to understand the extent of this reduction and the performance trade-offs involved.

VI. Conclusion

We developed a multi-agent coverage motion-planning policy to collectively send agents to all interest-points (targets) in the shortest possible time. A double deep reinforcement learning (DDRL) architecture was used to implement a tractable and scalable solution. The two-dimensional observation grid is decomposed into global and local maps centered at each agent enabling a common operating view for decentralized decision making across all agents. To realize large-scale implementation, global maps are downsampled to reduce the communication bandwidth between agents. Interest-point (target) detection using publicly available imagery of earthquake-damaged infrastructure was performed using the Deep Affinity Network (DAN) algorithm. Results on a limited sample of image datasets shows promising object detection accuracy. Extension of DAN feature learning using additional earthquake damaged image data sets is in progress. DAN is equally suited for detection of damaged infrastructure due to other types of natural disaster or military/terrorism effects.

In this paper, we proposed a computationally feasible lower-bound on the shortest coverage completion-time (the max-min completion-time) to evaluate the performance of the DDRL policy in scenarios with multiple agents and interest-points. To the best of our knowledge, this is the first instance of such bounds being used to quantify fundamental performance limits in multi-agent coverage/tracking problems. We showed that the performance of the DDRL control policy approaches the max-min completion time, and that downsampling the global maps reduces the number of training steps.

Future directions of research were identified, including the integration of heterogenous and heuristic hard-coded control policies, to improve coverage motion planning performance as agent, interest-point, and grid dimension increases.

References

[1] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning," in *2021 20th International Conference on Advanced Robotics (ICAR)*, 2021, pp. 539–546.

[2] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep Affinity Network for Multiple Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 104–119, 2021.

[3] "UAS-UAV," https://www.3gpp.org/uas-uav, Accessed: May 17, 2023.

[4] R. N. Haksar and M. Schwager, "Distributed Deep Reinforcement Learning for Fighting Forest Fires with a Network of Aerial Robots," in *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1067–1074.

[5] "NASA Works to Give Satellite Swarms a Hive Mind," https://www.nasa.gov/feature/goddard/2021/nasa-works-to-give-satellite-swarms-a-hive-mind., Accessed: Sept. 1, 2021.

[6] G. Viavattene and M. Ceriotti, "Artificial Neural Networks for Multiple NEA Rendezvous Missions with Continuous Thrust," *Journal of Spacecraft and Rockets*, vol. 59, no. 2, pp. 574–586, 2022.

[7] Google Maps, "12 Nail Bilen Cd., Gaziantep, Turkey," [Accessed May 23, 2023], https://www.google.com/maps/@37.0705606,37.3692316,3a,75y,158h,100.98t/data=!3m6!1e1!3m4!1sY1QJ4wEcjyyYDUkhbiChuw!2e0!7i16384!8i8192?entry=ttu.

[8] Mustafa Karali, "Turkey Earthquake, Image ID: 23037769217462," Associated Press General License Agreement, Order No. 106781225, May 23, 2023.

[9] "3D Multi-Object Tracking Using LIDAR Data," https://www.crcv.ucf.edu/wp-content/uploads/2018/11/Report_Farhat.pdf., accessed: 2023-05-17.

[10] "Visual Geometry Group (VGG)," University of Oxford, n.d., https://www.robots.ox.ac.uk/ vgg/software/via/.