# Heterogeneous Coordination for Persistent Monitoring using Particle Filters and Reinforcement Learning

†Md Ishat-E-Rabban, †Jingxi Chen, †Vishnu Dutt Sharma, ‡Kulbir Singh Ahluwalia, †Pratap Tokekar

***Abstract*— Persistent Monitoring using multiple ground vehicles has several applications including surveillance, patrolling, and precision agriculture. Existing works on the persistent monitoring problem use a graph theoretic formulation to repeatedly visit vertices of a given graph. In this work, we present a novel formulation of the persistent monitoring problem which is based on repeated visual coverage of an obstructed 2D environment. We propose a distributed algorithm based on particle filters and receding horizon strategy to plan the motion of multiple UGVs performing the visibility based persistent monitoring task. We also explore the scenario where a UAV flying over the environment is assisting the UGVs to plan better, and propose a reinforcement learning-based architecture to plan the motion of the UAV. We demonstrate the effectiveness of our proposed algorithms using simulation based experiments.**

## I. Introduction

The persistent monitoring (PM) problem is a variant of the coverage problem that resembles the idea of patrolling in real life. Persistent monitoring involves repeatedly covering a given area, e.g. patrolling, security surveillance, target searching, etc. In addition to the coverage maximization objective of the well-studied exploration problem, the PM problem also requires agents to revisit previously explored regions in order to achieve repeated monitoring of the space.

Existing works on the PM problem focus on graph-theoretic formulations where one or more UGVs traverse the edges of a prespecified graph to repeatedly visit vertices of the graph or given landmark objects [10]–[12]. In this work, we formulate a novel visibility based PM problem where a team of UGVs equipped with 360° visual sensors repeatedly patrols an obstructed 2D region. We call the problem visibility-based persistent monitoring (VPM) problem. We propose novel algorithms to plan the path of multiple UGVs to solve the VPM problem.

The design of an algorithm that plans the motion of the UGVs performing the VPM task depends on whether each UGV knows the position of all the other UGVs. In a practical scenario, it might be unrealistic to assume that each UGV always knows the positions of all other UGVs exactly, which requires uninterrupted communication among all the UGVs. In this work, we make a more realistic assumption that the UGVs might not always maintain communication among one another, hence have incomplete information about the position of the other UGVs.

†Rabban, Chen, Sharma, and Tokekar are with the Department of Computer Science, University of Maryland at College Park ier@umd.edu, ianchen@terpmail.umd.edu, vishnuds@umd.edu, tokekar@umd.edu

‡Ahluwalia is with the Maryland Robotics Center, University of Maryland at College Park kulbir@umd.edu

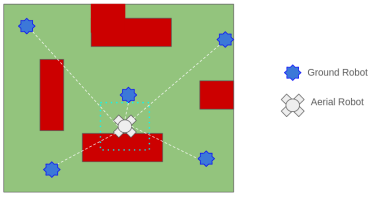


Fig. 1. The setup of our UAV-UGV collaboration

Another way of reducing the uncertainty in the position belief of the UGVs is to use a UAV as shown in Figure 1. The UAV moves faster over the obstacles, locates UGVs within the environment, and communicates the acquired location information to all UGVs within its communication range. Thus each UGV has a better estimation of the positions of the other UGVs in comparison with the case where there is no UAV. Consequently, the UGVs can plan more knowledgeably while performing the VPM task.

In this work, we make the following contributions:

- We propose a new visibility based formulation of the PM problem to repeatedly monitor an obstructed 2D region.
- We propose a distributed algorithm based on particle filters and receding horizon strategy to plan the motion of multiple UGVs performing the VPM task by maintaining position belief about other UGVs.
- We propose an algorithm based on the Actor-Critic version of Proximal Policy Optimization to plan the motion of the UAV such that it helps reduce the position uncertainty of the UGVs performing the VPM task.
- We conduct simulation-based experiments to demonstrate the effectiveness of our proposed algorithms.

## II. Related Work

The problem of persistent monitoring using multiple robots has been approached in multiple ways. A variety of factors like dynamic obstacles, the priority of monitoring locations, range of communication, and the number of robots affect the approach taken. Smith et al. [10] consider monitoring stationary locations of interest which change with time. The setup is such that locations are visited with a frequency proportional to their rate of change. The objective is to minimize the rate of change of locations between visits.

In our approach using sweep coverage [13], we have divided the environment into cells with equal latency that increases linearly with time. Other works have used graphs with vertex weights and edge lengths to represent the environment as in Alamdari et al. [11] and used *weighted latency*;

defined as the maximum time between visits to that vertex, weighted by the importance of that vertex (vertex weight). Our work is similar to the d-sweeper problem in which each point in the environment should be visible from a point on the route followed by robots. Ntafos [14] used the Watchman Route Problem (WRP) approach for the d-sweeper problem by superimposing a simple grid on the environment and using an approximate algorithm for finding a Travelling Salesman Problem (TSP) route.

Our formulation uses a visibility based approach in which the robot is not required to reach a node physically and can observe it from a range. Rezazadeh et al. [12] used a graph theoretic approach where the robot needs to visit a vertex physically. They have also considered the time taken for sensor measurement after reaching a node whereas we have assumed that the sensors take observations instantly.

While executing the persistent monitoring mission, one of the major concerns is to ensure that the UAV doesn't run out of power. Our work does not consider power constraints, but Maini et al. [1] have UAVs and UGVs move independently to their assigned goal locations with the option to recharge at refueling depots. However, this is a physical-symbiosis setting where the UGV acts only as a recharge station and does not help UAV with information sharing.

Blumenkamp et al. [2] use the idea of a heterogeneous robot system, however, their major focus is on two competing types of robots, one is cooperative, one is adversarial. The task is competing for the reward against each other. Sasaki et al. [4] use the heterogeneous robots in a cooperative setting however it uses only one UGV and one UAV communicating with each other. Also, this is a non-dynamic setting i.e. the UAV surveys the area first, and then the UGV navigates using this information for exploration. This is possible because the orienteering problem itself can be done in an offline fashion between UAV and UGV coordination.

Dille et al. [3] use a UAV as a communication relay between UGVs for non-line-of-sight operation. UGVs can provide high-resolution images and accurate data but have a limited speed and are bound to avoid obstacles. UAVs on the other hand have the advantage of flying over most obstacles but provide low-resolution images.

## III. Problem Formulation

In this section, we will go over our problem formulation including assumptions and the UAV, UGV planning problems. In this work, we solve the two planning problems independently.

### A. Assumptions

The key assumptions are listed below:

- The environment is bounded and it consists of static obstacles. The map of the environment (i.e. the boundary, and the obstacles) is known to all the robots (UGVs and UAV). The environment is represented as a 2D occupancy grid.
- All the robots are equipped with GPS, which gives each robot the capability to localize itself perfectly within the environment.
- All robots have a limited communication range. When any pair of robots come within the communication range of each other, they can exchange information about their belief of the environment.
- The UAV flies at an altitude above the obstacles. Using a downward-facing camera, the UAV can view a region on the 2D grid. The UAV can detect and uniquely identify UGVs within its viewing region.
- The UGVs are equipped with cameras with 360° field-of-view (FoV) and a limited visibility range. The views of the UGVs are occluded by the obstacles.
- The UAV can move faster than the UGVs.

### B. Multi-UGV VPM Problem

In this section, we introduce the VPM problem for multiple UGVs. First, we define *visibility* between a grid cell and a UGV, and introduce the idea of *latency*. A grid cell $c$ is defined to be visible from a UGV $r$ if the line segment $l$ joining the location of $r$ and the midpoint of $c$ is not obstructed by the obstacles and the length of $l$, is at most the visibility range of the UGV. Let $C$ denote the set of all free cells (i.e., cells not occupied by obstacles) in the 2D occupancy map of the environment. Each cell $c \in C$ has a latency value (denoted by $l_c$) in the range $[0, l_{max}]$. The latency of a cell varies according to the last time the cell was visible from some UGV. If a cell $c$ is visible from some UGV in the current time step, $l_c$ is set to 0. Otherwise, if $c$ is visible from no UGVs currently, $l_c$ increases linearly at each time step, until it reaches $l_{max}$.

In the multi-UGV VPM problem, we plan the motion of the UGVs over a given number of time-steps, $T$, which we call the *time horizon*. In each time-step, each UGV can move in any direction unless it collides with an obstacle. Note that, when a free cell $c$ becomes visible from a UGV, the latency value of $c$ drops to 0. Consequently, the goal of the VPM problem is to plan the movement of each robot over the time horizon, such that, the average latency value of all the cells in $C$ over the time horizon is minimized.

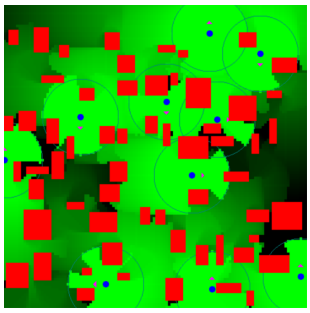

Fig. 2. An instance of the multi-UGV VPM problem.

Now we describe the VPM problem with an illustrative example. Figure 2 shows a $120 \times 120$ 2D grid environment, with 10 UGVs (shown using blue circles), and 60 obstacles

(shown using red rectangles). Shades between green and black represent latency of the cells, where pure green and pure black stands for 0 and $l_{max}$ respectively. The visibility range of the UGVs is 15 times the length of a cell (shown using thin black circles surrounding the UGVs). In each time step, each UGV can move either 1m forward, backward, left, or right. Purple arrows show the direction of the selected trajectories of the UGVs.

### C. UGV Planning Problem

We model the uncertainty in position belief and latency belief as follows. Each UGV $r$ maintains its individual local belief about the latency values of the cells in $C$, which we call *Latency Belief* of $r$. Also, $r$ maintains it's individual local belief about the positions of all UGVs, which we call *Position Belief* of $r$. The latency belief can be represented as a heat map over all cells in $C$, and the position belief can be represented as a set of particles as in the case of a particle filter. Note that, both the beliefs get updated when $r$ comes within the communication range of some other robot and exchanges belief information. Maintaining a belief about the latency of the free grid cells and the position of the other UGVs may help each UGV to plan better while performing the VPM task.

In the UGV planning problem, we divide the time horizon of $T$ time-steps into separate planning rounds each of $T_p$ time-steps. At each planning round, we plan the path of each UGV $r$ for $T_p$ time-steps in a distributed manner according to $r$'s latency and position belief.

Given the occupancy grid, the current latency belief, and the current position belief of a UGV, the UGV planning problem is to plan the motion of the UGV for the next $T_p$ time-steps such that the VPM objective is minimized.

Note that, the UGVs have different local latency and position beliefs, but all UGVs execute the same planner to select their trajectories.

### D. UAV Planning Problem

In our formulation of UAV-UGV coordination, since a UGV does know the exact position of other UGVs, we use the UAV to sample and update the belief about all UGVs' positions and use this shared information to assist the path planning of all UGVs. Thus UAV needs to plan a path that minimizes its uncertainty about UGVs' positions and share this information with them.

## IV. UGV Planning Algorithm

### A. Preliminaries

*1) Rapidly exploring Random Tree (RRT):* We use RRTs [5] to generate trajectories from a given UGV position. The UGV position is considered to be the root. We draw uniformly random sample points from the free space. For each sample, a connection is attempted between the sample and the nearest node in the tree. If the connection is unobstructed and within a pre-specified length, the sample is added to the tree as a new node. As samples are drawn uniformly from the whole search space, RRT is inherently biased to grow towards unsearched space. An example is shown in Figure 3(a).

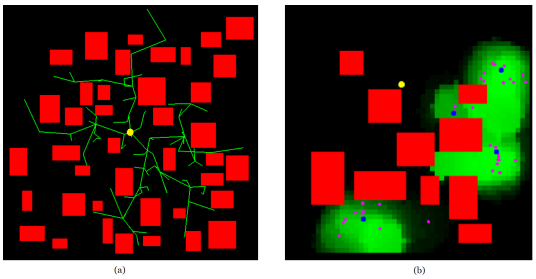

Fig. 3. (a) RRT rooted at the yellow node. (b) Position belief of the yellow node represented using particle filter. Particles are shown in magenta.

*2) Receding Horizon Planning:* Receding horizon strategy [6], [7] is commonly used in motion planning. In this strategy, an agent looks $T_q$ time-steps into the future to select the best candidate path. But the agent executes the found path for $T_p$ time-steps, where $T_p < T_q$. After $T_p$ time-steps, the planner runs again and works in the same fashion. Thus, the execution window is shorter than the planning horizon, which gives good results in practice.

*3) Particle Filter:* Particle filter [8], [9] is a Monte-Carlo technique to solve the filtering problem. In our problem, we use a particle filter to represent the position belief of a UGV. Each UGV $r_i$ maintains a set of particles for each of the other UGVs $r_j$ which represents $r_i$'s belief about the potential location of $r_j$. If the particles are close together, $r_i$ has a good estimation of where $r_j$ is located and vice versa. The position of the particles is updated at the completion of each planning round. If at that time, $r_i$ and $r_j$ are within the communication range of each other, all particles are updated to be at the true location of $r_j$. Otherwise, $r_i$ plans on behalf of $r_j$ to find the updated positions of the particles.

### B. The Algorithm

The UGV planning algorithm is presented below (Algorithm 1). The algorithm is from the perspective of robot $r$. It takes as input $r$'s latency belief $L_r$, $r$'s position belief $P_r$, the number of simulations $I$, the planning horizon $T_p$, and the extended horizon $T_q$. It returns the position of $r$ for the next $T_p$ time-steps. Here, $R$ denotes the number of robots.

The algorithm runs $I$ simulations based on the current latency and position belief of $r$. In each simulation, R particles, one particle for each robot, are sampled from $r$'s position belief $P_r$ (Line 2). The set of $R$ sampled particles is denoted by $X$, which stores the positions of the $R$ robots. In the *simulate* routine in Line 3, we construct RRTs rooted at each particle in $X$, consider a random ordering of the robots, then greedily assign paths of length $T_q$ to all the robots one after another according to the VPM objective. The positions of all the robots in all the time-steps during the extended horizon returned by the $i^{th}$ simulation are stored in the variable $S^i_{1:T_q,\,1:R}$.

Next, we update the latency belief such that it correctly estimates the latency values of the environment after $T_p$ time-

**Algorithm 1** UGV Planning Algorithm (for Robot $r$)

**Input:** $L_r$, $P_r$, $I$, $T_p$, $T_q$
**Output:** Position of $r$ for next $T_p$ time-steps

1: **for** $i \leftarrow 1$ to $I$ **do**
2: $X \leftarrow$ sample$(P_r)$
3: $S^i_{1:T_q,\,1:R} \leftarrow$ simulate$(X, L_r, T_q)$
4: **end for**
5: $L_r \leftarrow$ updateLatency$(\cup_{1\le i\le I} S^i_{1:T_p,\,1:R}, L_r)$
6: $P_r \leftarrow$ updatePosition$(\cup_{1\le i\le I} S^i_{T_p,\,1:R})$
7: $Y_{1:T_q} \leftarrow$ findBestPath$(\cup_{1\le i\le I} S^i_{1:T_q,\,1:R})$
8: **return** $Y_{1:T_p}$

steps using the *updateLatency* routine (Line 5). Starting from latency $L_r$, we simulate $T_p$ time-steps using the position values in $S$. We repeat this process for $I$ iterations and take the average, which gives the updated latency value after $T_p$ time-steps.

Next, we update the position belief such that it correctly estimates the position of the other robots after $T_p$ time-steps using the *updatePosition* routine (Line 6). For each robot other than $r$, we randomly sample the particles from $S$ at time-step $T_p$. Note that, at the beginning of the planning round, if some robot is within the communication range of $r$, there is no uncertainty in its position belief. All the particles corresponding to such robots are located at the true location.

Finally, we find the best path of robot $r$ in a receding horizon fashion using the *findBestPath* routine (Line 7). We construct an RRT rooted at $r$ and select the path of length $T_q$ that minimizes the VPM objective. We use the simulation results stored in $S$ to compute the best path over the extended horizon and return the initial segment of length $T_p$ of the selected best path (Line 8).

## V. UAV Planning Algorithm

### A. Overview

Our planning algorithms for UAV can be divided into two categories:

1) Heuristics-based Planning: We use an information greedy algorithm which moves the UAV in the direction with the highest uncertainty
2) Learning-based planning: In this case, we try reinforcement learning (RL) algorithms to learning to move efficiently for reducing uncertainty

Both of these algorithms use a particle-filter based approach described in the next section.

### B. Latency-based Particle Filter (L-PF):

L-PF is the common input for all of our UAV planning algorithms. The process of our L-PF is as shown in Figure 4. The algorithm for L-PF is shown in Algorithm 2. The $P^t_r$ is the belief (set of particles) of UGV $r$'s position. $W^t_r$ is the latency weight for each UGV, note that all particles in $P^t_r$ for UGV $r$ share this weight $W^t_r$.We use latency weight $W^t_r$ to model the time elapsed since UAV's last detection of UGV $r$, thus UAV can infer the correct ordering of future visits of UGVs. The $pos^t$ is the position of the UAV at the time $t$.

The subroutines in Algorithm 2 are as following:

1) motion: Taking in $P^t_r$ (particles about UGV $r$), and applying the motion model (path planning algorithm) for UGV $r$ to update the position of all particles.
2) measurement: Taking in UAV's position and get a measurement for evaluating the quality of the current belief/particles
3) resample: Based on the measurement $m^t$ and belief after motion $P^{t'}_r$, we can resample particles to get a better estimation/belief about UGVs' positions.
4) updateLatency: This will update the latency $W^t_r$ of all UGV/their particles. $W^{t+1}_r = 0$ if r is in UAV's FOV, otherwise $W^{t+1}_r = W^t_r + D$, where D is a user-defined latency accumulating rate to control how often should UAV revisits (re-localizes) a UGV $r$.
5) generateHeatmap: In this subroutine, we discretize the environment into the cells with equal size. the cell value $cell(i,j)$: $cell(i,j) = -100$, if UAV is in this cell, otherwise $cell(i,j)$ is the summation of all particles' latency value in this cell. As we can see in the heatmap as shown in Figure 4. The darkest cell is the UAV, and bright regions corresponding to the summation of latency value of particles in that cell.

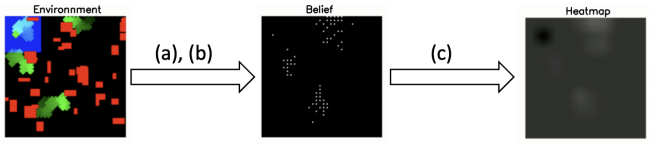


Fig. 4. The process of our L-PF algorithm. In the environment, the blue box are is the FOV of UAV, the green paths are the trajectories of UGVs. (a) is particle update as in general particle filter. (b) is latency update of UGVs (partciles) visting latency. (c) is heatmap generation.

**Algorithm 2** L-PF Algorithm

**Input:** $P^t_r$, $W^t_r$, $pos^t$ ,
**Output:** heatmap (about $P^{t+1}_r$, $W^{t+1}_r$)
$N_r \leftarrow$ Number of UGVs

1: **for** $r \leftarrow 1$ to $N_r$ **do**
2: $P^{t'}_r \leftarrow$ motion $(P^t_r)$
3: $m^t \leftarrow$ measurement $(pos^t)$
4: $P^{t+1}_r \leftarrow$ resample $(P^{t'}_r, m^t)$
5: $W^{t+1}_r \leftarrow$ updateLatency $(W^t_r, m^t)$
6: **end for**
heatmap $\leftarrow$ generateHeatmap($P^{t+1}_r$, $W^{t+1}_r$)
7: **return** heatmap

### C. UAV Planning Structure:

Our UAV planning algorithms utilize the heatmap generated by our L-PF algorithm. The structure of our planning is shown in Figure 5. The problem here can be defined as an MDP, where the state $s_t$ is the heatmap generated by the L-PF Algorithm at time $t$, action $a_t$ is the direction of motion for UAV in one of the four direction ($a_t \in A : \{North, East, South, West\}$), transition $T$ is deterministic and certain (i.e. $T(s_t, a_t, s'_t) = 1$, for all $s_t, a_t, s_{t+1}$). The reward collected by a UAV at time t and state $s_t$ is defined as as following:

$$R_\pi(s_t) = \sum_{r=1}^{N_r} -W_r^t \tag{1}$$

The reward corresponds to the summation of negative latency weight on all UGVs.

Our path planning part can be divided into two categories.

- Heuristic-based path planning: Using the beliefs from particle filter, we generate the next step for the UAV to move using an information greedy algorithm. In this method, the UAV moves in the direction of the maximum total latency calculated over state $S_t$ as following:

$$a_{t+1} = \underset{a_t \in A}{\operatorname{argmax}} \; exploreHeatmap(s_t, a_t),$$

  where $exploreHeatmap(dir)$ routine returns the sum of heatmap values, if the UAV were to explore all of its cells in the direction given by $a_t$.
- Learning-based path planning: We also try to investigate the performance of Reinforcement Learning (RL) approach on this path planning problem. We train an Actor-Critic version of Proximal Policy Optimization (PPO) [15] to see how the learning-based approach will perform on this path planning problem. The PPO takes input of heatmap and output the action for UAV.

### D. Objective function for UAV:

In this section, we define the objective for our learning-based approach. For a policy $\pi$, The objective for UAV is finding a policy $\pi$ such that:

$$\max_{\pi} \mathbb{E}[\sum_{t} R(s_t)] \tag{2}$$

Since we want to UAV to localize all UGVs as efficient as possible. (minimizing the latency weight of all UGVs)

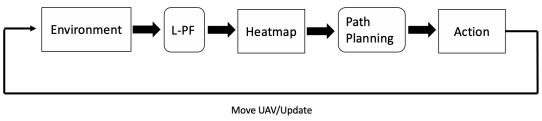


Fig. 5. The structure of UAV planning

## VI. Results and Evaluation

### A. Evaluation of UGV Planner

*1) Compared Algorithms:* In this section, we evaluate the performance of the UGV planning algorithm (Algorithm 1). We call this VPM algorithm. We compare the performance of the VPM algorithm with two algorithms. (1) The VPM-T algorithm, in which each UGV knows the true location of all other UGVs. (2) The VPM-O algorithm, in which the UGVs neither know the position of the other UGVs, nor they maintain a position belief. The evaluation metric is the VPM objective in Section III-C.

*2) Experimental Setup:* The environment is a $50 \times 50$ 2D region discretized into cells of size $1 \times 1$. There are 5 UGVs and 10 obstacles. The UGVs move at a speed of 1 unit per time-step. The communication and visibility range of the UGVs are 5 units. The time horizon is $T = 500$ time-steps and the planning horizon of each planning round is $T_p = 10$ time-steps. To generate trajectories, we construct RRTs with 50 nodes. The position belief regarding a UGV is represented using 10 particles. $l_{max} = 1.0$ and the latency value of a non-visible cell increases by 0.02 unit per second. Each experiment is performed 10 times and the average is reported. The algorithm is implemented using C++. The experiments are conducted on a core-i7 2GHz PC with 8GB RAM, running Microsoft Windows 10. The visualization tool used is OpenGL.

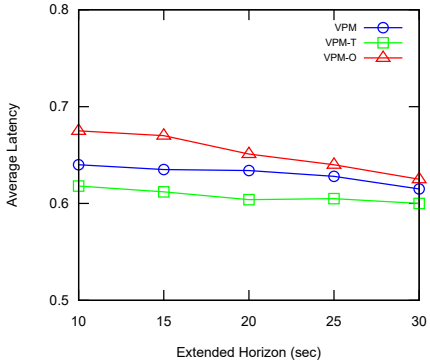


Fig. 6. Evaluation of VPM algorithm. $T_p$ = 10 time-steps.

*3) Results:* To compare the performance of the discussed algorithms, we vary the extended horizon $T_q$ from 10 time-steps to 30 time-steps in increments of 5. The experimental results are presented in Figure 6. Here lower value of average latency represents better performance and vice versa. The results show that the proposed VPM algorithm performs better and worse than the VPM-O and VPM-T respectively, because VPM-O and VPM-T has no and perfect location estimation of other UGVs respectively. Also, in the case of all the algorithms, the performance improves when the extended horizon is increased, which is a consequence of receding horizon planning strategy.

### B. Evaluation of UAV Planner

For evaluation of UAV-Planner algorithms, we developed our own environment in Python to support learning-based algorithms. We used a comparatively smaller grid ($100cells \times 100cells$) to reduce the computational cost of the particle filter and train the learning-based method faster. As the metric, we use episodic reward $R^e_\pi = \sum_{t=1}^{T} R_\pi(s_t)$, where $T$ is the length of the episode. As we were not able to integrate a UGV planner into our setup, the UGVs were moved randomly. The UGVs have a field-of-view with diameter 6 cells and move 3 cells in one step. The drone has a square filed-of-view of 30 cells and moved 6 cells in one step. The L-PF Algorithms uses 10 particles for each UGV.

As baseline we used two algorithms: (a) Random Motion Algorithm, which randomly select one of the four direction to move, and (b) Lawnmower Algorithm, which make the UAV move on a pre-defined path ($Boustrophedon\ path$) [16] scanning the grid from in a repeated manner. These algorithms do not use any information perceived by the UAV.

### C. UAV-Planner

In this section, we present our results about both learning-based and heuristic-based approach on UAV's path planning.

For the learning based approach, since our environment size is large ($100 \times 100$ grids), we do not have enough computational power to finish the training. We only train for 15000 episodes and with 500 steps in each episodes, using a GeForce RTX 2080 Ti GPU. This partial learning result is as shown in Figure 7.

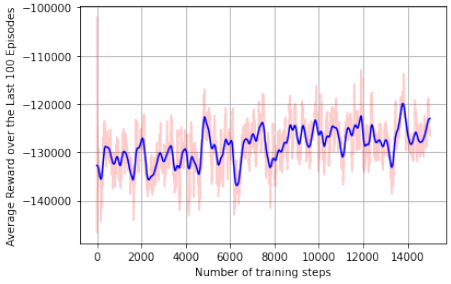


Fig. 7. The learning graph for PPO approach

We can see that our PPO-based approach learns to increase the reward collected UAV through episodes. However, this increase in reward is slow. The reason for this can be the large size of our map. Since the size of map is large, UAV can rarely get the experience of localizing any of UGVs, this lack of experience will cause difficulty in learning for our RL approach. We project that as we run the training longer, the performance of our model will still increase.

For comparison, we report the average episodic reward of 50 iteration. Each episode is of length $T$=1000, and consists of a randomly located UAV, UGVs and the obstacles. As we can see in Figure 8 and 9, the heuristic-based approach performs better than other approaches and scales comparatively well for different number of UGVs. The episodic reward decreases with increase in the number of robots as more number of robots will result in higher total uncertainty. We do not observe any significant change in reward with change in the number of obstacles. The learning-based approach doesn't follow a specific trend either. Similar to the previous setting heuristic-based algorithm performs the best followed by Lawnmower and Random motion algorithms. Learning-based algorithm does not perform well here either. The reason that the performance of learning-based approach is lowest may attribute to the insufficient training of our model as we discuss at the beginning.

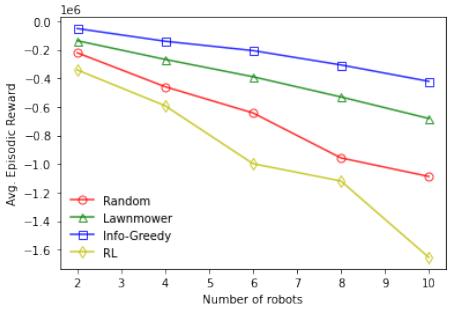


Fig. 8. Average episodic reward v/s Number of robots (4 UGVs)

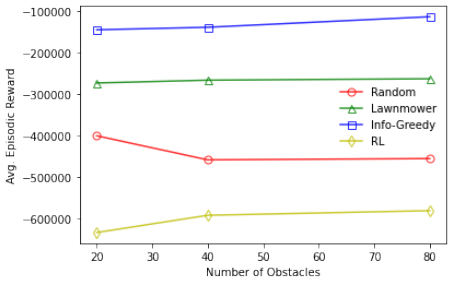


Fig. 9. Average episodic reward v/s Number of robots (40 Obstacles)

## VII. Conclusion

We empirically verify that maintaining position belief about the other UGVs improves the performance of the VPM task in comparison with the case where the position belief is not maintained. For UAV planning, the information greedy achieved good performance on localizing UGVs, however the motion model assumed here is sames as actual UGV motion. We would like to explore its performance when the assumed motion model differs from the reality. We also expect the learning-based algorithm to perform better with more training, as suggested by the learning graph.

We were not able to combine our work on UGV and UAV together and thus were not able to observe the efficacy of the proposed algorithms. The differences in the programming languages (C++ and Python) was also a contributing factor here. As next step, we plan to combine the two parts where the UAV shares its belief with the UGVs in sight and the UGVs then plan a path by utilizing this additional information.

## REFERENCES


[1] P. Maini, K. Yu, P. B. Sujit and P. Tokekar,"Persistent Monitoring with Refueling on a Terrain Using a Team of Aerial and Ground Robots," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 8493-8498.

[2] Blumenkamp, Jan, and Amanda Prorok. "The Emergence of Adversarial Communication in Multi-Agent Reinforcement Learning." arXiv preprint arXiv:2008.02616 (2020).

[3] Dille, Michael, Ben Grochosky, Stephen Nuske, Mark Moseley, and Sanjiv Singh. "Air-ground collaborative surveillance with human-portable hardware." (2011).

[4] Sasaki, Takahiro, Kyohei Otsu, Rohan Thakker, Sofie Haesaert, and Ali-akbar Agha-mohammadi. "Where to Map? Iterative Rover-Copter Path Planning for Mars Exploration." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2123-2130.

[5] S. Lavalle,"Rapidly-Exploring Random Trees: A New Tool for Path Planning," The Annual Research Report, 1998.

[6] Schouwenaars, Tom and How, Jonathan and Feron, Eric. "Receding horizon path planning with implicit safety guarantees." Proceedings of the 2004 American control conference, (2004), 5576-5581.

[7] Bircher, Andreas and Kamel, Mina and Alexis, Kostas and Oleynikova, Helen and Siegwart, Roland. "Receding horizon path planning for 3D exploration and surface inspection." Autonomous Robots, (2018), 291-306.

[8] Del Moral, Pierre. "Nonlinear filtering: Interacting particle resolution." Comptes Rendus de l'Académie des Sciences-Series I-Mathematics, (1997), 653-658.

[9] Y, Xiao and S. Katt and A. t. Pas and S. Chen and C. Amato. "Online Planning for Target Object Search in Clutter under Partial Observability." 2019 International Conference on Robotics and Automation (ICRA), (2019), 8241-8247.

[10] Smith, Stephen L and Rus, Daniela. "Multi-robot monitoring in dynamic environments with guaranteed currency of observations." 49th IEEE conference on decision and control (CDC), (2010), 514-521.

[11] Soroush Alamdari and Elaheh Fata and Stephen L. Smith. "Persistent Monitoring in Discrete Environments: Minimizing the Maximum Weighted Latency Between Observations." International Journal of Robotics Research, (2012), 138-154.

[12] Rezazadeh, Navid and Kia, Solmaz. "A sub-modular receding horizon solution for mobile multi-agent persistent monitoring." (2019).

[13] Choset, Howie. "Coverage for robotics–a survey of recent results." Annals of mathematics and artificial intelligence 31, no. 1-4 (2001): 113-126.

[14] Ntafos, Simeon. "Watchman routes under limited visibility." Computational Geometry 1, no. 3 (1992): 149-170.

[15] Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[16] Choset, Howie. "Coverage of known spaces: The boustrophedon cellular decomposition." Autonomous Robots 9, no. 3 (2000): 247-253.