Receding Horizon Multi-Robot Coverage
[Submitted to ICCPS 2018]

Sankar Narayan Das
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Uttar Pradesh 208016, India
Email: sdas@cse.iitk.ac.in

Indranil Saha
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Uttar Pradesh 208016, India
Email: isaha@cse.iitk.ac.in

***Abstract*—Coverage of a partially known workspace for information gathering is the core problem for several applications, such as search and rescue, precision agriculture and monitoring of critical infrastructures. We propose a planning framework for the coverage of a partially known environment employing multiple robots. To cope with the limitation of having incomplete information, our planner adopts a receding horizon planning strategy where the safe trajectories of the robots are generated optimally for a short duration based on the currently available information about the workspace. Moreover, as multi-robot motion planning for coverage is a computationally complex problem, our framework clusters the robots into small groups to increase the planning efficiency dynamically. In each time horizon, the robots follow the motion plans provided by the planner, gather information about the workspace while executing their plan and update the global knowledge base about the workspace. The planning algorithm manages the activities of the robots in such a way the energy consumption by the robots and the total time required for the complete coverage of the workspace get minimized. Simulation results show that the proposed hierarchical framework efficiently ensures the coverage quality of a partially known workspace, as well as scales up effectively with the number of robots and the size of the workspace.**

***Index Terms*—Mobile-Robot, Multi-Robot System, Coverage, Planning, Receding Horizon, Clustering**

I. Introduction

Applications such as search and rescue, precision agriculture, and monitoring of critical infrastructure heavily rely on the quality and timeliness of the acquired information from an area of interest (AoI) for successful planning, resource allocation, and utilization. For example, precision agriculture and monitoring of a nuclear reactor demand micro-management at various levels to cope with different types of inter and intra-field variability. The micro-management helps to maximize the crops yield in case of precision agriculture [14] and can be useful in generating a map of radiation level of a nuclear reactor and its surroundings in a timely manner [21]. One important aspect of the above-mentioned micro-management is gathering detailed data in a timely fashion for optimizing resource utilization.

The use of static sensor networks is widespread for efficient data gathering in different applications. Several algorithms and systems have been proposed for that purpose in the past (e.g. [18, 31, 13, 30]). However, deployment and management of a static sensor network can be challenging due to the lack of proper knowledge of the workspace where the nodes have to be deployed and due to the lack of mobility of the sensor nodes. Recently, mobile robots have been identified as an excellent medium for carrying out sensing in known or unknown environments (e.g. [27, 8, 28, 1, 17]). However, the scalability of the proposed algorithms have been limited to small number of robots. Recently the revolution in the creation of several miniature robots (for example, Kilobot [24], Jasmine [15]) motivates us to consider deploying mobile robots for sensing and data gathering applications in a large scale. In this paper, our goal is to demonstrate that the data gathering for appropriate resource utilization and monitoring can be efficiently accomplished by a large team of multi-robot systems.

Data gathering from an AoI using a group of robots requires the robots to cover the area efficiently. For example, in case of a disaster response application, the robots have to cover every location in the workspace to detect if some survivors are present. Similarly, for the precision agriculture applications, the robots should visit every part of the field to detect the crops that can be harvested or the locations in the field affected by pests. Coverage of an unknown or partially known AoI with a multi-robot system is a challenging problem. At the beginning, the robots have limited or no information about the AoI. Consequently, offline planning is impossible; the robots require to coordinate effectively among themselves and perform planning online for substantial coverage.

We propose a planning framework for the coverage of a partially known AoI or workspace by a set of robots. Our framework comprises a computational facility called the *global planner* and multiple robots equipped with sensing and limited computational capability. To deal with the unavailability of the complete information about the workspace, the global planner adopts a receding horizon planning mechanism in which the planner partitions the total planning into multiple short planning-horizons of varying lengths. This strategy is motivated by the similar techniques that have been successful in other domains, for example, model predictive control [4] and receding horizon reactive motion planning [33]. At the beginning of each planning horizon, the global planner generates a global view of the workspace based on the local views of the robots. Next, the global planner divides the robots

into clusters based on their proximity and estimates the length of the current horizon. The robots are clustered in such a way that in the current horizon there is no possibility that the robots in one cluster will collide with the robots in another cluster. Moreover, partitioning the robots into smaller clusters helps in scaling up the multi-robot motion planning. The global planner generates a suitable and safe motion plan for the robots in each cluster. Being inspired by the recent success of SMT solvers in solving task and motion planning problems for robotic systems [12, 22, 25, 32, 26, 6, 9], we formulate the coverage planning problem for each cluster in each horizon as an SMT solving problem [2] and use an off-the-self SMT solver Z3 [20] to generate the trajectories for the robots.

The global planner communicates the plans to the robots, and the robots execute their respective plans. While executing the plans provided by the global planner, a robot performs sensing in the locations it visits. The robots also gather information about the workspace by detecting the locations of the obstacles in the surrounding regions, and communicates its updated local view to the global planner at the end of each horizon.

The reason for introducing a centralized global planner instead of designing a distributed protocol for robots is twofold. First, the end users can easily interact with the robots through the global planner, and can specify their requirements in an *online* manner. Second, the global planner can help the robots to use the concepts of Cloud Robotics [10, 19], and apply resource demanding methodologies, such as VSLAM [3, 7] and SMT solver [26], during various phases of a planning process. Moreover, the global planner can also facilitate the resource sharing, coordination, and usage of low-level robots effectively.

We have implemented our coverage planning framework using C++. Our software uses SMT solver Z3 [20] for solving the planning problems for the clusters in each horizon. Moreover, to utilize the availability of several processing cores in the computing systems, we use OpenMP framework [5] and solve the planning problems for the clusters in a horizon concurrently as much as possible. Our experimental results show that we can achieve complete coverage of a $128\mathrm{m}\times128\mathrm{m}$ workspace by 64 robots in less than $30\mathrm{min}$, demonstrating that our framework can be an efficient tool for utilizing large-scale multi-robot systems for sensing and data gathering operations for various applications.

In summary, we make the following contributions in this paper.

- We provide an SMT-based framework for multi-robot coverage of a partially known environment. The framework employs a receding horizon planning strategy to deal with partial observability of a workspace.
- We propose a clustering based approach and several heuristics to reduce the planning overhead and scale up the multi-robot coverage planning with respect to number of robots and size of the workspace.
- We provide a C++ implementation of our coverage planning framework and demonstrate the efficacy of our proposed framework through extensive simulation.

II. Problem

A. Terminology and Definitions

In this section, we formalize the definitions needed for the rest of the paper.

1) Workspace representation: We assume that a set of robots, denoted by $R = \{r_1, \ldots, r_{|R|}\}$, is operating in the AoI or workspace for coverage. The workspace for an application is represented as a 2-D *rectangular* occupancy grid map. The grid decomposes the workspace into square-shaped cells. If a workspace contains x_L and y_B ($x_L, y_B \in \mathbb{N}$) cells along the x and y dimension, respectively, the size of the workspace is represented as $[x_L \times y_B]$. Each cell is assigned a unique identifier which represents the *location* of that cell in the workspace. The identifier of the block with co-ordinate (i,j), $i \in \{0, \ldots x_L\}$ and $j \in \{0, \ldots x_L\}$, is denoted by w_{ij}. The set of all locations in the workspace is denoted by the set $\mathcal{W}$.

Some parts of the workspace can be occupied by static obstacles. If a grid cell is partially occupied by an obstacle, we mark the entire grid cell to be covered by obstacle. The set of locations covered by obstacles is denoted by Ω. The set of free locations in the workspace is denoted by $\mathcal{F}$, where $\mathcal{F} = \mathcal{W} \setminus \Omega$.

2) Motion model of the robot: To define the motion model of the robot, we first define the state of the robot. We assume that the robot has a finite set of *velocity configurations*, denoted by $\mathbb{V}$. A velocity configuration represents a velocity with a constant magnitude and a direction.

Definition 1 (State of a robot)**.** *The state of a robot ϕ is a pair $\langle l, v\rangle$, where $l \in \mathcal{W}$ denotes its position in the 2-D workspace, and $v \in \mathbb{V}$ is the velocity configuration of the robot.*

Motion primitives: The dynamic model of a robot is captured using a set of motion primitives. Motion primitives are a set of short closed-loop trajectories of a robot under the action of a set of precomputed control laws [16]. The set of motion primitives form the basis of the motion for a robot. For example, in the most simple case a ground robot may have five motion primitives: $\{\mathtt{H}, \mathtt{L}, \mathtt{R}, \mathtt{U}, \mathtt{D}\}$, where the primitive `H` keeps the robot in the same grid block and the primitives `L`, `R`, `U` and `D` move the robot to the adjacent left, right, upper, and lower grid block, respectively. The special primitive `H` is called the *rest primitive*. A desired trajectory for a robot can be generated by concatenating a sequence of motion primitives appropriately.

The set of motion primitives for robot $r \in R$ is denoted by Γ_r. Each motion primitive $\gamma \in \Gamma_r$ is associated with a *pre-condition* $pre(\gamma)$, which is a formula over the states specifying under which conditions a motion can be executed. For a robot in state ϕ and with a motion primitive γ, we denote by $\mathtt{post}(\phi, \gamma)$ the state ϕ' where the robot moves when the motion primitive γ is applied to the robot in state ϕ. We use $\mathtt{intermediate}(\phi, \gamma)$ to denote the set of grid locations through which the robot may traverse after applying γ in state ϕ (including $\phi.l$ and $\phi'.l$ where, $\phi' = \mathtt{post}(\phi, \gamma)$).

For a motion primitive γ, we denote by $\texttt{cost}(\gamma)$ the cost (e.g., energy expenditure) to execute the motion primitive.

We assume that for all robots in the system, each motion primitive requires δ unit time for execution. This assumption may not hold for heterogeneous systems and extending our approach for such systems is left as a future work.

3) Trajectory and motion plan: We now define the trajectory and motion plan for a multi-robot system. We start with defining the state of the multi-robot system.

Definition 2 (State of a Multi-Robot System)**.** *The state of a multi-robot system with the set of robots given in R is denoted by $\Phi = [\phi_1, \ldots, \phi_{|R|}]$, where ϕ_i denotes the state of robot $r_i \in R$.*

We define the runtime behavior of a multi-robot system using a discrete-time transition system $\mathcal{T}$. We define a state transition as follows.

Definition 3 (Transition)**.** *Let $\Phi_1 = [\phi_{11}, \ldots, \phi_{1|R|}]$ and $\Phi_2 = [\phi_{21}, \ldots, \phi_{2|R|}]$ be two states of the multi-robot system, and $\gamma = [\gamma_1, \ldots, \gamma_{|R|}]$, where $\gamma_i \in \Gamma_{r_i}$, be a vector that contains as elements the primitives applied to individual robots in state Φ_1 to move them to state Φ_2. The transition from Φ_1 to Φ_2 is governed by the following rule:*

$$\Phi_1 \xrightarrow{\gamma} \Phi_2$$

iff

- $\forall r_i \in R$: $\Phi_{1i} \models pre(\gamma_i)$ *and* $\Phi_{2i} = post(\Phi_{1i}, \gamma_i)$.
- *The trajectory of $r_i \in R$ between the states Φ_1 and Φ_2 does not pass through a block occupied by an obstacle, that is*
  $\forall r_i \in R : intermediate(\Phi_{1i}, \gamma_i) \cap \Omega = \emptyset$.
- *The robots do not collide with each other while doing an (atomic) move from state Φ_1 to state Φ_2, that is*
  $\forall r_i \in R, \forall r_j \in R \setminus r_i :$
  $intermediate(\Phi_{1i}, \gamma_i) \cap intermediate(\Phi_{1j}, \gamma_j) = \emptyset$.

Note that the complexity of collision avoidance grows quadratically with the number of robots.

Definition 4 (Trajectory and Motion Plan)**.** *A trajectory of length L for a multi-robot system R is defined as a sequence of multi-robot states $\Phi = (\Phi(0), \Phi(1), \ldots, \Phi(L))$ such that the states are related by the transitions in the following way:*

$$\Phi(0) \xrightarrow{\gamma(1)} \Phi(1) \xrightarrow{\gamma(2)} \Phi(2) \ldots \Phi(L-1) \xrightarrow{\gamma(L)} \Phi(L).$$

A motion plan for the trajectory Φ is defined as a sequence of vectors of motion primitives $(\gamma(1)\gamma(2)\ldots\gamma(L))$ to be applied to the robots in R to generate the trajectory. The trajectory of a robot r is given by $\Phi_r = (\Phi_r(0), \Phi_r(1), \ldots, \Phi_r(L))$ and the corresponding motion plan as $\gamma_r = (\gamma_r(1)\gamma_r(2)\ldots\gamma_r(L))$.

Definition 5 (Length of a Trajectory)**.** *The length of a trajectory Φ, denoted as $Length(\Phi)$, of a multi-robot system is the number of transitions in the trajectory. If the trajectory is Φ = $(\Phi(0), \Phi(1), \ldots, \Phi(L))$, $Length(\Phi) = L$.*

The total duration to move all robots from their initial state to final state is given by $L \times \delta$.

Definition 6 (Cost of a Trajectory)**.** *The cost of a trajectory of a multi-robot system is equal to the cumulative cost of all the primitives used in the trajectory. For a trajectory Φ:*

$$\Phi(0) \xrightarrow{\gamma(1)} \Phi(1) \xrightarrow{\gamma(2)} \Phi(2) \ldots \Phi(L-1) \xrightarrow{\gamma(L)} \Phi(L),$$

where $\gamma(i) = [\gamma_{r_1}(i), \ldots, \gamma_{r_{|R|}}(i)]$, $\gamma_{r_j}(i) \in \Gamma_{r_j}$, the cost of the trajectory Φ, denoted by $\texttt{cost}(\Phi)$, is given by

$$\texttt{cost}(\Phi) = \sum_{i=1}^{L} \sum_{j=1}^{|R|} \texttt{cost}(\gamma_{r_j}(i)) \tag{1}$$

Definition 7 (Cost-optimal Trajectory)**.** *A multi-robot trajectory Φ is cost optimal if there does not exist another trajectory Φ' that can be synthesized using the motion primitives in Γ such that $\texttt{cost}(\Phi') < \texttt{cost}(\Phi)$.*

B. Robot Equipment

We assume that each robot is equipped with two sensors: a *scalar sensor* for an application specific sensing and a *range finder* to identify the obstacles. For example, the robot may use the scalar sensor to measure some quantity like temperature, wireless signal strength, radiation strength etc. The scalar sensor may be used to take picture of a cell in the workspace to identify some object. In this work, we assume that the robot is required to visit a cell to cover it. However, in many applications, the robot would require to visit only a subset of cells to achieve full coverage in terms of information gathering.

The robot is also equipped with a range finder that helps the robot to detect nearby obstacles. Depending on how many range sensor the robot carries and the way they are placed on the robot, the robot is able to detect some obstacles surrounding it. If a robot detects a workspace cell to be partially covered by an obstacle, the robot can mark the cell to be covered by obstacle. However, if the robot can observe a cell partially, and find the cell to be partially free, the robot cannot decide whether the cell is obstacle-free or occupied by obstacle.

The robots are equipped with a communication module to communicate their sensed values to a central server. We assume that the robots are equipped with localization sensors that enable them to follow a given trajectory precisely.

C. Problem Statement

The input *coverage planning problem for multi-robot system* can be expressed by a tuple $P_{cov} = \langle \mathcal{W}, \Omega_0, R, I, \Gamma \rangle$, where $\Omega_0 \subseteq \Omega$ is the set of initially know obstacle-covered cells, $I : R \rightarrow \mathcal{W} \setminus \Omega_0$ represents the initial locations of the robots, and $\Gamma = [\Gamma_{r_1}, \ldots, \Gamma_{r_{|R|}}]$ denotes the the motion primitives available to the robots. Given a coverage planning problem P_{cov}, our overall objective is to generate safe (obstacle-free and collision-free) trajectories Φ for the robots in R so that the robots can meet the following objectives:

- Minimize the total exploration time, i.e.,

$$\text{Minimize} \quad Length(\Phi). \tag{2}$$

- Minimize the total cost of the trajectories, i.e.,

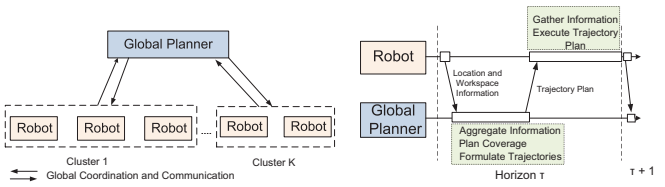
$$\text{Minimize} \quad \texttt{cost}(\Phi). \tag{3}$$

- Satisfy the coverage constraint. In this work, we assume that the robots have to visit all obstacle-free cell in the workspace for the full coverage. Thus, the robots have to satisfy the following constraint:

$$\forall w \in \mathcal{F}.\ \exists r \in R. \bigvee_{i \in \{0,1,\ldots,Length(\Phi)\}} \Phi_r(i).l = w \tag{4}$$

III. A Receding Horizon Framework for Multi-Robot Coverage Planning (RHOCOP)

In this section, we present our coverage planning framework for a multi-robot system, named RHOCOP, which enables the robots to explore a partially known workspace efficiently. As the workspace is partially observable, meaning that the locations of the obstacles are not known a priori, an offline optimal planning for the robots is not feasible. Our framework relies on a central server that communicates with the robots to gather their local knowledge about the workspace and builds a partial global view about the workspace at a regular interval. Based on this partial global view of the workspace, the central server invokes a planner to generate a task plan (which cells in the workspace a robot should visit) and motion plans (what paths the robots should follow to visit those cells) for the robots. More concretely, the central server is called the global planner (GP). Figure 1(a) illustrates the overall architecture of the system.



(a) An Architecture of Receding Horizon Multi Robot Planning Framework

(b) Interaction between Global Planner and a robot within a planning horizon τ

Fig. 1: Schematic Representation of Hierarchical Planning Framework

More concretely, at the beginning of the τ^{th} horizon, the GP builds a global view of $\mathcal{W}$ by collecting information from the robots, makes a feasible plan of length $L^{(\tau)}$ for each robot, and communicates the plans to the robots. The robots now execute the plans and gather information about the workspace by their scalar sensor and the obstacle detecting sensor while following the trajectories governed by the motion plans provided by the GP. At the end of the τ^{th} horizon, the robots communicate their states and corresponding views of $\mathcal{W}$ to the GP. After receiving the responses from the robots, the GP updates its view of $\mathcal{W}$, and generates a plan for the next $(\tau+1)^{th}$ horizon based on the updated information. A schematic representation of the interaction between the GP and a robot is illustrated in Figure 1(b).

A. Information Gathering by a Robot

At the beginning of the τ^{th} horizon, a robot r receives a motion plan $\gamma_r^{(\tau)}$, by executing which the robot follows a trajectory $\Phi_r^{(\tau)}$ of length $L^{(\tau)} = Length(\Phi_r^{(\tau)})$. While executing the trajectory $\Phi_r^{(\tau)}$, the robot uses its scalar sensor in each cell $w \in \mathcal{F}$ on its trajectory for gathering required data. Moreover, the robot uses its obstacle detection sensor in each cell on its trajectory to detect any surrounding cell to be occupied by obstacle. In the horizon, the robot assigns to each cell $w \in \mathcal{W}$ a probability $Pr_w(OBS)$ that indicates the probability of the cell w to be occupied by an obstacle. If the probability $Pr_w(OBS) < \eta_{min}$ ($Pr_w(OBS) > \eta_{max}$), the robot decides the cell to be obstacle-free (occupied by an obstacle), where $0 \leq \eta_{min} < \eta_{max} \leq 1$, and η_{min} and η_{max} are two threshold for deciding a cell to be obstacle-free or occupied by an obstacle. If $\eta_{min} \leq Pr_w(OBS) \leq \eta_{max}$, then the robot r does not have a definite information of the cell w in that horizon. Thus the robot declare the cell w to be *unexplored*.

The status of each cell $w \in \mathcal{W}$ as in the view of robot r in the horizon τ is given below.

$$\begin{aligned} Status_r^{(\tau)}(w) = \textit{covered} \quad & \text{if} \bigvee_{i \in \{0,1,\ldots,L^\tau\}} \Phi_r^{(\tau)}(i).l = w \\ = \textit{visible} \quad & \text{if } Status_r^{(\tau)}(w) \neq \textit{covered} \wedge \\ & (Pr_w(OBS) < \eta_{min} \vee Pr_w(OBS) > \eta_{max}) \\ = \textit{unexplored} \quad & \text{if } Status_r^{(\tau)}(w) \neq \textit{covered} \wedge \\ & \eta_{min} \leq Pr_w(OBS) \leq \eta_{max} \end{aligned} \tag{5}$$

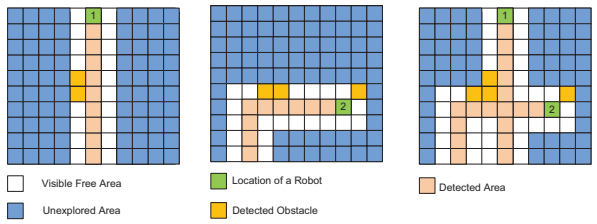
The local view of r, represented as $\mathcal{W}_r^{(\tau)}$, is expressed as follows:

$$\begin{aligned} \mathcal{W}_r^{(\tau)} &= \langle C_r^{(\tau)}, V_r^{(\tau)}, U_r^{(\tau)} \rangle \\ C_r^{(\tau)} &= \{w \in \mathcal{W} \mid Status_r^{(\tau)}(w) = \textit{covered}\} \\ V_r^{(\tau)} &= \{w \in \mathcal{W} \mid Status_r^{(\tau)}(w) = \textit{visible}\} \\ \mathsf{U}_r^{(\tau)} &= \{w \in \mathcal{W} \mid Status_r^{(\tau)}(w) = \textit{unexplored}\}, \end{aligned} \tag{6}$$

where $C_r^{(\tau)}$, $V_r^{(\tau)}$, and $\mathsf{U}_r^{(\tau)}$ represent the covered, the visible, and the unexplored cells from the perspective of robot r in horizon τ, respectively. Figure 2(a) and 2(b) depict an example of local views of robot r_1 and r_2, respectively.

B. Aggregation of Local Views

The GP, at beginning of any planning horizon τ, collects the local views of the robots created in time horizon $\tau - 1$, and combines all the collected local views to build a global



(a) Local view of Robot r_1 (b) Local view of Robot r_2 (c) Aggregated Global view

Fig. 2: An example of multiple local views and the global view, built from the local views

view of the workspace. The global view, $\mathcal{W}_G^{(\tau)}$, at beginning of any planning horizon τ can be expressed as follows:

$$\begin{aligned}
\mathcal{W}_G^{(\tau)} &= \langle C_G^{(\tau)}, V_G^{(\tau)}, U_G^{(\tau)} \rangle \\
C_G^{(\tau)} &= C_G^{(\tau-1)} \cup (\underset{r \in R}{\cup} C_r^{(\tau-1)}) \\
V_G^{(\tau)} &= (V_G^{(\tau-1)} \cup (\underset{r \in R}{\cup} V_r^{(\tau-1)})) \setminus C_G^{(\tau)} \\
\mathsf{U}_G^{(\tau)} &= \mathcal{W} \setminus \{C_G^{(\tau)} \cup V_G^{(\tau)}\}, \qquad (7)
\end{aligned}$$

where $C_G^{(\tau)}$, $V_G^{(\tau)}$, and $\mathsf{U}_G^{(\tau)}$ represent the covered, visible, and unexplored cells of global view, respectively. Figure 2(a) and Figure 2(b) illustrate examples of local views whereas the global view, made from those local views, is depicted by Figure 2(c). In the next section, we describe how GP utilizes the global view to generate trajectories for the robots for the horizon τ.

C. Full Trajectory of the Robots

The complete trajectory of a robot is given as the concatenation of the trajectories in all the horizons. The trajectory of the robot $r \in R$ is given by $(\Phi_r^{(0)}(\Phi_r^{(0)}(L^{(0)}))^* \Phi_r^{(1)}(\Phi_r^{(1)}(L^{(1)}))^* \ldots \Phi_r^{(K)}(\Phi_r^{(K)}(L^{(K)}))^*)$, where K is the total number of horizons required for complete coverage of the workspace. We denote by $(\Phi_r^{(\tau)}(L^{(\tau)}))^*$ the finite time waiting by a robot at the end of horizon τ, when the GP aggregates the local views obtained from the robots and generates plans for the next horizon. We denote by $\Phi_r = \Phi_r^{(0)}\Phi_r^{(1)}\ldots\Phi_r^{(K)}$ the active trajectory of robot r. The active trajectory of the multi-robot system is denoted by $\Phi = \Phi^{(0)}\Phi^{(1)}\ldots\Phi^{(K)}$, where $\Phi^(i) = [\Phi_{r_1}^(i), \ldots \Phi_{r_{|R|}}^(i)]$. We assume that the cost incurred by the robots is negligible during their wait. Thus, the goal of the global planner is to minimize $\texttt{length}(\Phi)$ and $\texttt{cost}(\Phi)$.

IV. Global Planner

To solve the coverage planning problem for a large number of robots efficiently in a horizon, the global planner adopts a divide-and-conquer approach. As will be evident from our experimental results, the planning time grows exponentially with the number of robots. To manage the planning complexity, the GP divides the robots into several clusters and solve the coverage planning problems for each cluster independently.

A. Clustering of Robots

The outcomes of the clustering procedure are a set of clusters of robots $\mathcal{C}^{(\tau)} = \{\mathcal{C}_1^{(\tau)}, \ldots, \mathcal{C}_n^{(\tau)}\}$ and the length of the trajectories for the robots in the horizon $L^{(\tau)}$. Note that the number of clusters n is different in different horizon.

1) Clustering algorithm: The clustering process works as follows: The locations of the robots are considered as the vertices of a graph, and Euclidean distance between any two vertices is estimated as the weight of the edge between the two vertices. Next, the global planner deletes all the edges which are greater than $2 \times L_{min} + 1$, where L_{min} represents the minimum length of any planning horizon. By deleting some of the edges of the graph, the global planner partitions the graph into multiple connected but smaller components, and each of those parts is considered as a cluster. Note that the inter-cluster distances are greater than or equal to $2 \times L_{min} + 1$, where the inter-cluster distance is defined as the minimum distance between any two nodes in two different clusters.

2) Estimating the length of the next horizon: In each planning horizon, the GP decides for a horizon length $L^{(\tau)}$ for all the robots in all the clusters. As the planning complexity increases with the number of robots, the GP decides the value of $L^{(\tau)}$ based on the largest cluster (the cluster having the maximum number of robots) so that the planning time for the largest cluster can be minimized. At the very beginning, i.e., for the planning horizon $\tau = 0$, the length of the horizon boundary is initialized to $L^{(0)} = L_{min}$. For the successive intervals, the GP first computes the sizes of the clusters, and then calculates the horizon length based on some simple rule as the one bellow:

$$\begin{aligned}
L^{(\tau)} &= L_{min} && \text{if } \varrho \leq |C_k^{(\tau)}|_{max} \\
&= L_{min} + \varrho - |C_k^{(\tau)}|_{max} && \text{if } |C_k^{(\tau)}|_{max} < \varrho
\end{aligned}$$

Here, $|C_k^{(\tau)}|_{max}$ represents the size of the largest cluster in horizon τ, and ϱ is a user defined threshold value.

3) Bounding Box: Inter-Cluster Collision Avoidance: The GP encloses each of the clusters by a *bounding* box. The bounding boxes partition the workspace into non-overlapping regions. Thus, the bounding boxes help the GP to plan for any cluster independently of others, and to avoid inter-cluster collisions. The formulation of bounding boxes is a two-step process. At first, the GP encloses any operational robot $r_j \in C_k^{(\tau)}$ by a bounding box of area $(2L_{min}+1) \times (2L_{min}+1)$ by keeping the robot at the center of the box. Next, the GP computes the smallest bounding box enclosing the individual bounding boxes of the member robots of cluster $C_k^{(\tau)}$. The smallest bounding box becomes the bounding box for the cluster $C_k^{(\tau)}$. Figure 3 illustrates an example of bounding-boxes. For clarity, the boundaries of individual bounding-boxes and the merged boxes are drawn separately.

4) Selecting Active Robots in a Cluster: After clustering of the robots in any planning horizon τ, the GP classifies the robots of a cluster $C_k^{(\tau)}$ as *Active* or *Dormant* robots. The reason of making some robots dormant is that there could

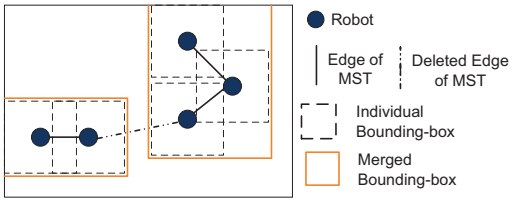


Fig. 3: Bounding-boxes for inter-cluster collision avoidance

be many robots in a compact cluster, and not all robots are required for covering the region surrounded by the bounding box of the cluster in that horizon. The GP counts the number of visible cells within the bounding-box which encloses the cluster $C_k^{(\tau)}$, estimates heuristically the number of robots $N_k^{(\tau)}$ required to visit those cells, and selects from the members of $C_k^{(\tau)}$ only those $N_k^{(\tau)}$ robots which have more visible neighbour cells than the others. For example, if the number of visible cells in the cluster is ν, then $N_k^{(\tau)}$ can be estimated as $\lceil \frac{\rho \times \nu}{L^{(\tau)}} \rceil$, where $\rho \geq 1$ is a parameter to be chosen appropriately. The GP selects $N_k^{(\tau)}$ robots from the robots in $C_k^{(\tau)}$ in the set $Active_k^{(\tau)}$ and the rest of the robots in the set $Dormant_k^{(\tau)}$. The GP generates motion plans for the $Active_k^{(\tau)}$ robots only. The $Dormant_k^{(\tau)}$ robots do not change their locations throughout the planning horizon τ, and thus are considered to be static obstacles.

B. Coverage Planning

In each planning horizon τ, the GP formulates an optimal and safe coverage planning problem for the robots in each cluster $\mathcal{C}_k^{(\tau)} \in \mathcal{C}^{(\tau)}$ based on $\mathcal{W}_G^\tau$, the available information about the workspace at the beginning of the current horizon τ. To solve the coverage planning problem for a cluster of robots $C_k^{(\tau)}$ and horizon length $L^{(\tau)}$ in the τ-th horizon, the global planner needs to synthesize a trajectory $(\Phi^{(\tau)}(0)\ \Phi^{(\tau)}(1)\ \ldots\ \Phi^{(\tau)}(L^{(\tau)}))$ and the corresponding motion plan $(\gamma^{(\tau)}(1)\ \gamma^{(\tau)}(2)\ \ldots\ \gamma^{(\tau)}(L^{(\tau)}))$, where $\forall t \in \{0 \ldots L^\tau\}$, $\Phi^{(\tau)}(t)$ is the vector containing the states of the robots in $C_k^{(\tau)}$, and $\forall t \in \{1 \ldots L^{(\tau)}\}$, $\gamma^{(\tau)}(t)$ is the vector containing the motion primitives to be applied to the robots in $C_k^{(\tau)}$ in state $\Phi^{(\tau)}(t-1)$. At time step t, the state of a robot $r \in C_k^{(\tau)}$ is denoted by $\Phi_r^{(\tau)}(t)$ and the motion primitive applied to the robot is denoted by $\gamma_r^{(\tau)}(t+1)$.

1) Utility Estimation: The primary objective of GP within horizon τ is to maximize the acquired utility by the robots in τ and to minimize the cost of the trajectories of those robots. The overall objective of GP in τ can be expressed as follows:

$$\text{Maximize} \sum_{r \in R} \sum_{t=1}^{L^{(\tau)}} U_r^{(\tau)}(t) - \sum_{r \in R} \sum_{t=1}^{L^{(\tau)}} cost(\gamma_r^{(\tau)}(t)), \quad (8)$$

where, $\sum_{r \in R} \sum_{t=1}^{L^{(\tau)}} U_r^{(\tau)}(t)$ represents the total utility acquired by the robots during that planning horizon τ. On the other hand, $\sum_{r \in R} \sum_{t=1}^{L^{(\tau)}} cost(\gamma_r^{(\tau)}(t))$ is the total cost due to transitions of the robots during τ^{th} horizon. The symbol $U_r^{(\tau)}(t)$ that represents the utility acquired by robot r at time instant t is equal to the utility of the cell w in the planning horizon τ, where, $w = \Phi_r^{(\tau)}(t).l$.

The utility of the cell w in the planning horizon τ, denoted by U_w^τ, can be measured as follows:

$$U_w^\tau = -M \quad \text{if } w \in \mathsf{U}_G^\tau \vee Pr_w(OBS) \geq \eta_{max} \quad (9a)$$
$$= 0 \quad \text{if } w \in C_G^\tau, \quad (9b)$$
$$= u_w \quad \text{if } w \in V_G^\tau, \quad (9c)$$

where M is a large positive number. It can be noted from Eq. (9a) that a large negative utility is assigned to a cell w if it is either unexplored or obstacle-occupied. In Eq. (9b), the GP assigns zero utility to the covered cells. The GP estimates the utility u_w of a visible cell w on the basis of its neighbourhood information and u_w is estimated as follow:

$$u_w = w_U \times u_w^{(U)} + w_V \times u_w^{(V)} + w_C \times u_w^{(C)}, \quad (10)$$

where $u_w^{(U)}$, $u_w^{(V)}$, and $u_w^{(C)}$ are the count of the number of undiscovered, visible, and covered neighbours of w respectively, and w_U, w_V, w_C represent the corresponding weights. If the GP assigns larger values to the weights $u_w^{(U)}$ and $u_w^{(V)}$, the robots *explore* the workspace more. On the other hand, the robots focus to cover more visible cells near to their current locations if the wieght $u_w^{(C)}$ is assigned a larger value.

2) Constraints: To solve the coverage planning problem for a cluster of robots, the GP formulates an optimization problem with the objective function given in (8) subject to the following constraints:

Initial location of the trajectory. At the initial state $\Phi_r^{(\tau)}(0)$ of the trajectory, a robot r will be in the location reached in the final state $\Phi_r^{(\tau-1)}(L^{(\tau-1)})$ of the previous horizon . If the current horizon is the first horizon, then the robot will be in its initial location $I(r)$ in the initial state of the trajectory.

$$\forall r \in C_k^{(\tau)}.\Phi_r^{(\tau)}(0).l = (\tau == 0)?\ I(r) : \Phi_r^{(\tau-1)}(L^{(\tau-1)}).l \quad (11)$$

Ensuring conformance between states and motion primitives: For each robot $r \in \mathcal{C}_k^{(\tau)}$, at each time instant t, the state $\Phi_r^{(\tau)}(t)$ should satisfy the precondition of the motion primitive applied to the robot at time instant t. Moreover, the state $\Phi_r^{(\tau)}(t)$ should satisfy the postcondition of the motion primitive applied to the robot at time instant $t-1$.

$$\forall r \in C_k^{(\tau)}.\ \forall t \in \{0, \ldots, L^{(\tau)} - 1\}.\Phi_r^{(\tau)}(t) \models pre(\gamma_r^{(\tau)}(t+1))$$
$$\forall r \in C_k^{(\tau)}.\ \forall t \in \{1, \ldots, L^{(\tau)}\}.\Phi_r^{(\tau)}(t) = post(\Phi_r^{(\tau)}(t-1), \gamma_r^{(\tau)}(t)) \quad (12)$$

Collision avoidance: The following set of constraints ensures that the robots do not collide with each other.

$$\forall t \in \{0, \ldots, L^{(\tau)} - 1\}.\forall r_i \in C_k^{(\tau)}.\forall r_j \in C_k^{(\tau)} \setminus r_i.$$
$$intermediate(\Phi_{r_i}^{(\tau)}(t), \gamma_{r_i}^{(\tau)}) \cap intermediate(\Phi_{r_j}^{(\tau)}(t), \gamma_{r_j}^{(\tau)}) = \emptyset. \quad (13)$$

The solution of the optimization problem provides the motion plans for the robots for the τ-th horizon. Note that we do not have explicit constraints for ensuring obstacle avoidance. As we maximize the objective function and the utility for visiting a location covered by obstacle is a large negative number, the optimal trajectory is obstacle-free.

C. Optimization on Global Planner

1) Dropping Robots for Enhancing Planning Efficiency: When the robots are near completion of the coverage of the workspace, fewer cells remain to be covered, and involving all the robots is not efficient any more. To deal with such situations, the GP may drop some of the robots in τ^{th} horizon if the planning efficiency in the $(\tau-1)^{th}$ horizon, denoted by $\kappa^{(\tau-1)}$, is below a threshold value. The set of dropped robots is denoted by $\bar{R}$. The GP measures the planning efficiency of τ^{th} horizon, $\kappa^{(\tau)}$ as the ratio of the number of cells covered in the horizon and the number of operational robots. Mathematically,

$$\kappa^{(\tau)} = \frac{|C_G^{(\tau)} \setminus C_G^{(\tau-1)}|}{|R \setminus \bar{R}|}. \tag{14}$$

If the planning efficiency of $(\tau-1)^{th}$ horizon $\kappa^{(\tau-1)}$ is below a threshold value, then GP randomly drops some of the dormant robots in the current horizon, and includes those robots in $\bar{R}$. The set $\bar{R}$ grows with time. In any horizon, only the robots in $R \setminus \bar{R}$ participates in the coverage planning. The robots in $\bar{R}$ are considered as static obstacles in the workspace. As our experimental results demonstrate, dropping robots helps in improving the value of the objective function in (3) significantly.

2) Potential Field based Utility Modification: The GP always strives to maximize its objective, defined by Eq. (8). Consequently, at the beginning of τ^{th} horizon, an isolated visible cell may emerge or a robot r_k may be surrounded only by covered cells. We term a robot as *surrounded robot* if it is surrounded by covered cells only. On the other hand, by *isolated cell*, we mean a visible cell with all immediate neighbours already covered. It is observed that the presence of isolated cells increases the coverage overhead in terms of coverage time and total trajectory length. In case of surrounded-robots, formulating a suitable path is difficult, as the GP plans with receding horizons with local knowledge. It is difficult for the GP to decide in which direction it should steer the robot so that it can reach a visible cell quickly.

We introduce a potential field based method [11] to diminish the imperfection of the local objective maximization by the global planner. The GP generates a potential field to attract any robot to an isolated cell or to direct a surrounded-robot towards a visible cell/region. To deal with an isolated cell, the GP generates a circular potential field of radius $L^{(\tau)}$ centering the cell, and updates the utility of a covered cell within that potential field. On the other hand, to synthesize a trajectory for a surrounded robot r_k, the GP first generates an obstacle-free trajectory from the current location of the robot to its nearest reachable visible cell. Next, the GP, considering the location of r_k as starting location, updates the utilities of the first $L^{(\tau)}$ covered cells. Figure 4 depicts an example of utility modification of the covered cells near an isolated cell and a surrounded-robot based on potential field, where $L^{(\tau)} = 2$.

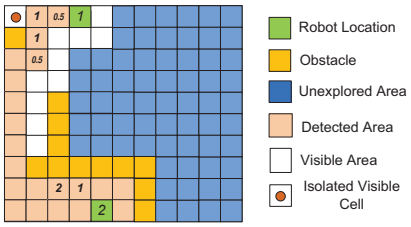


Fig. 4: Utility Modification Based on Potential Field

V. Evaluations

We have evaluated the performance of RHOCOP through extensive simulation along with Robot Operating System (ROS) [23]. The Global Planner generates the motion plans for the robots by using SMT solver Z3 [20]. Further, we employ OpenMP framework [5] to generate motion plans for the clusters concurrently. All the experiments have been performed in a 64-bit Ubuntu 14.04 LTS machine with Intel(R) Core^{TM} i7-4770 CPU @ 3.40GHz $\times$ 8 processor and 16 GB RAM. The values of the parameters used in our experiments are provided in Table VII in Appendix A.

A. Experimental Set up

The experimental set-up can be characterised on the basis of the various parameters, which are enlisted as follows: (i) Size and (ii) visibility of the Workspace, (iii) number of robots, and (iv) planning approaches considered by the Global Planner on the basis of the number of robots.

Workspace: The workspace can be characterized by its (i) *size*, and (ii) *visibility*. Here, we have evaluated the performance of RHOCOP with workspace of different sizes, such as (i) 16×16, (ii) 32×32, (iii) 64×64, and (iv) 128×128. On the other hand, the visibility of workspace can be classified as (i) full or (ii) partial. In a fully observable workspace the locations of the obstacles are known a priori. While evaluating RHOCOP, we consider the workspace to be partially observable to the robots and hence to the Global Planner (GP).

Planning approaches: We have classified the planning approaches, broadly, into two classes: (i) Clustered Planning, and (ii) Total Planning. The classification is based on the number of robots, considered by the GP at the time of planning. In Clustered Planning, the GP plans separately for individual clusters. However, the GP simultaneously considers all the robots during Total Planning.

Benchmark schemes: We have considered three benchmark schemes for comparing the efficiency of our proposed approach RHOCOP. The first scheme is termed as SOLO, which consists of only one robot. The second one is termed as TOTAL planning, in which the GP plans for all the robots simultaneously. Moreover, the workspace is fully visible to the robots as well as the GP. The third benchmark is termed as IDEAL Planning. In IDEAL planning, the GP can fully observe the workspace. Moreover, the GP clusters the robots, and uses receding horizon planning to reduce the overhead of

Benchmarks	Attributes			
	Visibility	Horizon	Planning Type	Total Robots
TOTAL	Full	Fixed	Total	$\|R\| > 1$
SOLO	Partial	Receding	-	$\|R\| = 1$
IDEAL	Full	Receding	Clustered	$\|R\| > 1$
RHOCOP	Partial	Receding	Clustered	$\|R\| > 1$

TABLE I: Characteristics of Planning Schemes

motion planning. Table I illustrates the characteristics of the benchmark schemes, along with RHOCOP.

Benchmark metrics: The efficiency of the different planning schemes are compared with the help of the following benchmark metrics: (i) number of robots supported by a planning scheme, (ii) total time taken by the Global Planner to plan, (iii) average length of the paths traversed by the robots, shortly average path length, (iv) number of iterations or planning horizons required to cover the workspace, and (v) total length of the planning horizons or total horizon length. Total planning time and total horizon length together signify the overall planning overhead. However, average path length represents the overhead of a robot.

B. Robot Model

In our experiments, we have used the specifications of Turtlebot [29], a widely used robot for academic research. The robot has two types of primitives - (i) moving forward and backward, and (ii) rotating a fixed angle while remaining at the same position. We assume that the robot carries a directional range finder which is installed in front of the robot. Thus we consider only the forward motion for the robot.

The robot has four velocity configurations corresponding to the heading towards east, west, north and south while remaining stationary. We design the motion primitives in such a way that the robot can move one cell forward in the current heading direction by changing its velocity instantaneously from 0 to $0.5\mathrm{m\,s^{-1}}$ at the beginning and from $0.5\mathrm{m\,s^{-1}}$ to 0 at the end, or it can rotate by 90° either in the left or in the right direction while being in the same location. Thus we have three motion primitives for each velocity configuration, providing us total 12 motion primitives for the robot.

The dimension of each cell in the workspace is $1\mathrm{m} \times 1\mathrm{m}$, thus the execution of the forward motion primitive takes $2\mathrm{s}$. The angular velocity of the robot is set in such a way that each rotation primitive also requires $2\mathrm{s}$. We assume that the range finder installed on the robot can detect obstacles within $4\mathrm{m}$ distance from the robot with a field of view of 90°.

C. Experimental Results

This subsection is partitioned into three parts. At first, we evaluate the performance of TOTAL planning. In other words, we identify the drawbacks of considering all the robots simultaneously, and fixed horizon length during coverage planning. Subsequently, we compare RHOCOP with SOLO and IDEAL planning approaches. Further, we demonstrate how RHOCOP overcomes the drawbacks of TOTAL planning. At the end, we illustrate the effects of (i) different weight factors of Eq. (10), (ii) dropping of robots during clustering, and (iii) potential field on the performance of RHOCOP.

Sl.	Total Robots	Horizon Length	Iteration	Planning Time (Sec)	Total Horizon Length	Path Length
1.	2	2	209	31.3	418	418
3.		4	54	31.1	216	216
4.		8	26	1348	208	208
9.	4	2	121	67.12	242	242
11.		4	35	302.7	140	140
12.		8	`timeout`	`timeout`	`timeout`	`timeout`
13.	8	2	0	0	0	0
14.		3	29	1140	87	87
15.		4	`timeout`	`timeout`	`timeout`	`timeout`

TABLE II: Summary of experimental result for TOTAL Planning for a 16×16 workspace (`timeout` = 2h)

1) Evaluation of TOTAL Planning: For the evaluation of the TOTAL Planning scheme, we consider a 16×16 fully observable workspace. We have varied the number of robots and also the fixed length of planning horizons in different experiments. ITable II summarizes the results of Total Planning. It can be observed from Table II that the planning time increases exponentially with the number of robots and/or the length of the planning horizon. In other words, planning time dominates other factors of coverage planning when number of robots as well as the horizon length is large. As a consequence, TOTAL planning approach cannot accomplish efficient coverage a large workspace with a large number of robots. These results motivated us to introduce receding horizon coverage-planning, clustering of robots into small groups in RHOCOP, and also to consider variable horizon length depending on the size of the largest cluster in different planning horizons.

2) Performance Evaluation of SOLO, IDEAL, and RHOCOP: Table III, partially, provides the evaluation results of the RHOCOP, SOLO, and the IDEAL planning. Here, we illustrate the general trends, whereas the detailed results are provided in Table VIII of Appendix B. It can be noted from Table III that the coverage overhead for SOLO, generally, is too large. For example, the total exploration time for SOLO to cover a workspace of size 128×128 is more than 15 hours ($(641 + 2 \times 27734)$s, as each motion primitive takes $2\mathrm{s}$ for its execution). On the other hand, with the help of 64 robots, RHOCOP can solve the coverage problem in less than half an hour ($(417 + 2 \times 678)\mathrm{s} = 1773\mathrm{s}$). In terms of total trajectory length (which is an indication of total energy consumption by the robots), the total path length of 64 robots is 32800, which is less than 20% above the total path length 27734 for a single robots. This results indicate that RHOCOP is capable of managing a large group of robots efficiently both in terms of the two cost functions provided in (2) and (3).

Further, it can be observed that, in case of RHOCOP, the number of iterations, total horizon length, and path length decrease uniformly with the increase in the number of robots. Moreover, the planning time increases minimally with the number of robots due to the efficient use of multi-core processor for generating motion-plans for different clusters.

These results demonstrate that RHOCOP can solve multi-robot coverage planning problem efficiently and is capable of utilizing the power of employing a large number of robots effectively.

We include the results for IDEAL planning to show how well the RHOCOP deals with partial observability. As the results show, RHOCOP takes not more than 20% iteration and path length than those of the IDEAL planning in any scenario. One noticeable fact is that IDEAL planning has taken more time for motion planning than RHOCOP in most of the cases. In IDEAL planning, the workspace is fully observable to GP, and more number of cells are visible within any cluster due to full observability. Consequently, the GP chooses fewer robots as dormant in IDEAL planning than that in RHOCOP during clustering in each horizon. As more robots in a cluster means more time for planning, IDEAL planning takes more time for motion planning than RHOCOP.

In Figure 7 of Appendix C, we have illustrated an example scenario where the GP assists 4 robots to cover a 10×10 workspace. Figure 7 depicts the state of the workspace and locations of the robots in different planning horizons.

3) Impact of Weight Factors: The GP approximates utility, u_w, of a cell w_{ij} from its neighborhood information. The GP may formulate different coverage strategies by assigning different weights to the various components of u_w, which, from Eq. (10), comprises three components with associated weight factors. In this work, we consider the range of the weight factors to be in between 0 and 1, and discretize the values of the weight factors into three intervals: *High*, *Medium*, and *Low*. The widths of the intervals are as follows: $1 \geq High \geq 0.75$, $0.75 > Medium \geq 0.25$, and $0.25 > Low \geq 0$. Next, we illustrate the effects of different weight factors. For brevity, we have evaluated only 4 strategies among the 3^3 strategies. The characteristics of those 4 strategies are summarized by Table IV. In HLL or MML, the GP puts more weight to exploration (try to find more visible cells), whereas the GP puts more weight to cover neighbouring cells in LLH or MLM. For evaluation of these strategies, we consider total 16 robots deployed in a 32×32 workspace. Figure 5 depicts the efficiency of different strategies. It can be noted that among these four strategies, MML outperforms the other strategies, though the strategy MLM catches up with MML strategy at the end. In the experimental evaluation of RHOCOP in Table III, the GP considers MLM for coverage strategy. However, estimating appropriate coverage strategy on the basis of number and locations of the robots, and the state of the coverage process is a challenging problem.

4) Clustering: Effects of Dropping Robots: The GP suitably drops robots or reduces the number of operational robots at later stages of coverage planning to eliminate the non-performing robots. Table V shows that the reduction of operational robots always enhances planning efficiency. It is observed that the robots often form large clusters if the GP does not drop the robots. Consequently, the GP estimates short horizon length, and short horizons increase the number of total iterations or horizons. Moreover, not dropping non-performing

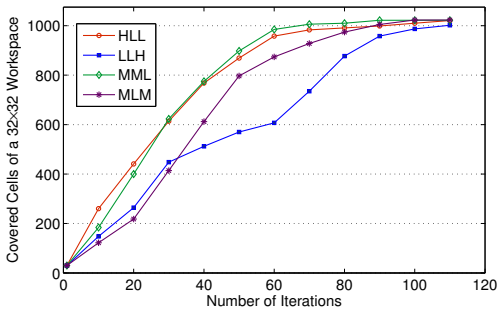


Fig. 5: Impacts of Parameters' Values

robots increases the average path length of the robots. As an example from Table V, the GP fails to generate motion plans for 32 robots in a 32×32 workspace when it does not drop robots, as the robots often form a large cluster. So, it can be concluded that clustering along with dropping robots helps the GP to scale up coverage planning with the number of robots.

Figure 6 illustrates the change in the number of operational robots with respect to the number of detected cells in a 32×32 workspace. It can be observed that when the coverage reaches 80% of the workspace, the GP starts to reduce the number of operational robots to improve the planning efficiency. The experimental result illustrates that the GP is capable of enhancing the planning efficiency by dropping robots at different rate in different scenarios. As an example, the dropping rate of 32 robots is different from 8 or 16 robots.

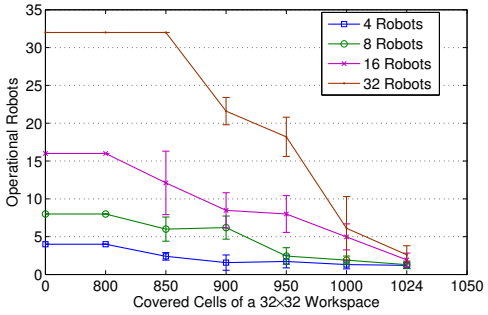


Fig. 6: Number of Operational Robots vs Detected Cells

5) Effects of Potential Field: The Potential Field (PF) helps the GP to accomplish the required coverage by pulling robots towards the isolated visible cells and by pushing a robot, surrounded by detected cells, to a visible cell. To evaluate the effectiveness of PF, we consider the GP *without* PF and *with* PF. Total three test cases are evaluated. In the first, eight robots are used for coverage of a 16×16 workspace. For others, 16 robots provide coverage of 32×32 and 64×64 workspaces. The test result is summarized in Table VI. It can be observed from Table VI that PF is essential for successful coverage or 99.8% coverage of workspaces. Though the GP without PF is successful 80% times to attain required coverage of a 16×16 workspace, the success of GP without PF for larger workspace, such as 32×32 and 64×64, is very limited or null. Moreover,

Sl.	Benchmark	Workspace Size	Total Robots	Iteration		Planning Time (Sec)		Total Horizon Length		Path Length	
				Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
1.	16×16	SOLO	1	98	0.5	7.72	0.25	484	1	484	1
2.		RHOCOP	4	48.83	3.1	13.71	0.34	210.0	17.4	136	2.98
3.			8	30.86	1.86	17.51	0.77	109.4	9.8	61.71	2.90
4.			16	28.63	6.1	20.7	1.47	107.88	28.76	31.65	2.34
5.	32×32	SOLO	1	356.8	3.67	35.12	0.24	1781	19.17	1781	19.17
6.		IDEAL	4	137.8	8.1	61.1	12.2	582.0	38.2	489.75	26.38
7.			8	113.2	29.2	147.6	41.75	480.2	179.8	257.2	12.90
8.			16	54.0	10.1	176.1	53	206	33.1	129.25	11.6
9.			32	53.5	8.3	115.4	29.3	200.2	40	61	1.40
10.		RHOCOP	4	177.8	34.56	37.9	8.68	718.0	127.4	536	24.18
11.			8	128.9	25.75	64.7	22.3	517.3	60.8	268.7	6.90
12.			16	76.23	12.26	84.6	15.96	282.85	43.4	142.2	4.20
13.			32	44.8	6.7	176.3	7.76	152.4	27.2	60.1	2.4
14.	64×64	SOLO	1	1358	17.67	141.12	1.64	6785	85	6785	85
15.		IDEAL	16	158.8	11.9	158.4	15.1	643.8	60.6	464.3	13.8
16.			32	86	6.4	362.7	29.6	296.8	19.40	224.8	2.1
17.		RHOCOP	16	170.7	10.02	124.6	9.3	734.15	44.4	519.2	25.720
18.			32	90.6	3.7	270.3	11.8	302.66	3.4	243	3.16
19.	128×128	SOLO	1	5548	24.67	641	32.24	27734	169.7	27734	169.7
20.		IDEAL	32	257.4	12.2	454.5	27.3	1126	39.1	910.25	16.7
21.			64	188.7	13.73	431.6	16.5	649.2	34.40	476.8	6.20
22.		RHOCOP	32	315.5	32.26	432.8	56.96	1358.85	73.4	1009.3	23.90
23.			64	192.4	11.6	417.3	29.76	678.4	39.2	512.5	9.8

TABLE III: Experimental Evaluation of SOLO, IDEAL, and RHOCOP Planning Approaches

Strategy Name	w_U		w_V		w_D	
	Type	Value	Type	Value	Type	Value
HLL	High	1	Low	0.1	Low	0.1
LLH	Low	0.1	Low	0.1	High	1
MML	Medium	0.5	Medium	0.5	Low	0.1
MLM	Medium	0.5	Low	0.1	Medium	0.5

TABLE IV: Weight Factors of Different Coverage Planning Strategies

Sl.	Total Robots	Dropping Robots	Iteration	Planning Time (Sec)	Total Horizon Length	Path Length
1.	4	No	175.4	53.6	758.45	581.6
2.		Yes	177.8	37.9	818	536
3.	8	No	117.3	68.65	465.6	284.4
4.		Yes	128.9	64.7	517.3	268.7
5.	16	No	93	78.4	298.3	151.34
6.		Yes	76.23	84.6	282.8	142.2
7.	32	No	-	-	-	-
8.		Yes	44.8	176.3	152.4	60.1

TABLE V: Effects of Dropping Robots

it can be observed that the PF reduces the planning overhead by improving the results in all dimensions. In summary, the use of PF is an essential component for the success of receding horizon coverage planning.

VI. Conclusion

In this paper, we have championed the use of large-scale multi-robot systems for various applications involving sensing and data gathering operations. To facilitate the use of multi-robot systems for such applications, we have proposed a framework that can provide a scalable solution for multi-robot coverage of a partially known environment. Simulation results show that the proposed framework ensures coverage quality in presence of incomplete information, and is scalable with respect to the number of robots and the size of the workspace. Though our framework currently works for 2-D environments, our system can be seamlessly extended to 3-D environments and would be useful for various applications involving drones and UAVs.

In our future work, we would like to address several challenging issues that can make our system more efficient and robust. First, in our current work, we have assumed that the obstacles in the workspace are static. Enhancing our planning framework to deal with dynamic obstacles in the workspace would enhance the practical applicability of our system. Second, our system currently deal with a homogeneous set of robots and extending it to deal with heterogeneous robots is our another future goal. For several applications, deploying a heterogeneous set of robots may be more useful and efficient. Finally, our system currently operates in a synchronous manner. All the robots wait when the global planner generates the plan and the global planner remains dormant when the robots execute their plan. We would like to develop an asynchronous version of our system where the global planner will serve the robots in a demand-driven way with the goal that the robots and the global planner can be kept busy for most of the time during the operation of the system.

Acknowledgments

References

[1] O. Arslan and Daniel E. Koditschek. Voronoi-based coverage control of heterogeneous disk-shaped robots. In *ICRA*, pages 4259–4266, 2016.

[2] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 4, chapter 8. IOS Press, 2009.

Sl.	Total Robots	Workspace Size	Potential Field	Successful Simulations (%)	Avg. Covered Cells	Attributes of Successful Simulations			
						Iterations	Planning Time (sec)	Total Horizon Length	Path Length
1.	8	16×16	Absent	79.28	252.9	48.85	28.8	173.1	70.6
2.			Present	100	256	30.86	17.51	109.4	61.71
3.	16	32×32	Absent	72.5	1009.5	92.6	87.8	332.5	185.8
4.			Present	100	1023	76.24	84.6	282.8	142.2
5.	16	64×64	Absent	0	3872	-	-	-	-
6.			Present	100	4090.8	170.7	124.6	734.15	519.2

TABLE VI: Effects of Potential Field

[3] P. Benavidez, M. Muppidi, P. Rad, J. J. Prevost, M. Jamshidi, and L. Brown. Cloud-based realtime robotic visual slam. In *SysCon*, pages 773–777, 2015.

[4] D. M. Prett C. E. Garca and M. Morari. Model predictive control: theory and practice - a survey. *Automatica*, 25(3):335–348, 1981.

[5] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5 (1):46–55, 1998.

[6] A. Desai, I. Saha, J. Yang, S. Qadeer, and S. A. Seshia. Drona: A framework for safe distributed mobile robotics. In *ICCPS*, 2017.

[7] X. Gao and T. Zhang. Unsupervised learning to detect loops using deep neural networks for visual slam system. *Autonomous Robots*, 41(1): 1–18, 2017.

[8] S. Garg and N. Ayanian. Persistent monitoring of stochastic spatio-temporal phenomena with a small team of robots. In *RSS*, 2014.

[9] Ivan Gavran, Rupak Majumdar, and Indranil Saha. ANTLAB: a multi-robot task server. In *EMSOFT*, 2017.

[10] G. Hu, W. P. Tay, and Y. Wen. Cloud robotics: architecture, challenges and applications. *IEEE Network*, 26(3):21–28, 2012.

[11] L. Huang. Velocity planning for a mobile robot to track a moving target-a potential field approach. *Robotics and Autonomous Systems*, 57 (1):55–63, 2009.

[12] W. N. N. Hung, X. Song, J. Tan, X. Li, J. Zhang, R. Wang, and P. Gao. Motion planning with Satisfiability Modulo Theroes. In *ICRA*, pages 113–118, 2014.

[13] O. D. Incel, Amitabha Ghosh, Bhaskar Krishnamachari, and Krishna Chintalapudi. Fast data collection in tree-based wireless sensor networks. *IEEE Transactions on Mobile computing*, 11(1):86–99, 2012.

[14] Austin Jones, Usman Ali, and Magnus Egerstedt. Optimal pesticide scheduling in precision agriculture. In *ICCPS*, pages 12:1–12:8, 2016.

[15] S. Kernbach, D. Haebe, O. Kernbach, R. Thenius, G. Radspieler, T. Kimura, and T. Schmickl. Adaptive collective decision-making in limited robot swarms without communication. *The International Journal of Robotics Research*, 32(1):35–55, 2013.

[16] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[17] S. K. Lee, S. P. Fekete, and J. McLurkin. Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers. *The International Journal of Robotics Research*, 35 (19):1234–1260, 2016.

[18] S. Lindsey, C. Raghavendra, and K. M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):924–935, 2002.

[19] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroeger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, pages 1957–1964, 2016.

[20] L. De Moura and N. Bjrner. Z3: an efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[21] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma. Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63, 2013.

[22] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki. SMT-based synthesis of integrated task and motion plans from plan outlines. In *ICRA*, 2014.

[23] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[24] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *ICRA*, 2012.

[25] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *IROS*, pages 1525–1532, 2014.

[26] I. Saha, R. Ramaithitima, V. Kumar, G. J Pappas, and S. A Seshia. Implan: A scalable incremental motion planning framework for multi-robot systems. In *ICCPS*, pages 1–10, 2016.

[27] K.S. Senthilkumar and K.K. Bharadwaj. Multi-robot exploration and terrain coverage in an unknown environment. *Robotics and Autonomous Systems*, 60:123–132, 2012.

[28] D. E. Soltero, M. Schwager, and D. Rus. Decentralized path planning for coverage tasks using gradient descent adaptive control. *The International Journal of Robotics Research*, 33(3):401–425, 2014.

[29] TurtleBot. http://www.turtlebot.com, 2010.

[30] L.A. Villas, A. Boukerche, H. ABF De Oliveira, R. B. De Araujo, and A. AF Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12:69–85, 2014.

[31] J. Wang, S. Tang, B. Yin, and X.-Y. Li. Data gathering in wireless sensor networks through intelligent compressive sensing. In *INFOCOM*, pages 603–611, 2012.

[32] Y. Wang, N. T. Dantam, S. Chaudhuri, and L. E. Kavraki. Task and motion policy synthesis as liveness games. In *ICAPS*, page 536, 2016.

[33] T. Wongpiromsarn, U. Topcu, and Richard M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57 (11):2817–2830, 2012.

APPENDIX A
DESCRIPTION OF PARAMETERS

Parameter Description	Symbol	Value
Minimum Horizon Length	L_{min}	2
Threshold for Max. Cluster Size	ϱ	4
Parameter for estimating the number of Active robots	ρ	1.5
Utility of Obstacle Occupied or Unexplored Cell	$-M$	-5
Weight factor associated with unexplored cells	w_U	0.5
Weight factor associated with visible cells	w_V	0.1
Weight factor associated with covered cells	w_C	0.5
Threshold for Planning Efficiency	$\kappa^{(\tau)}$	0.5

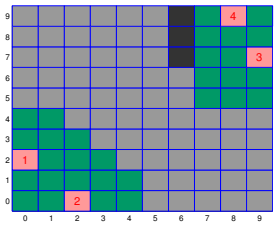
TABLE VII: Summary of Parameter values

APPENDIX B
DETAIL EVALUATION OF SOLO, IDEAL AND RHOCOP

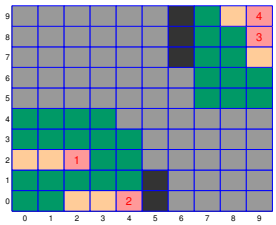
Sl.	Benchmark	Workspace Size	Total Robots	Iteration		Planning Time (Sec)		Total Horizon Length		Path Length	
				Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
1.	16×16	SOLO	1	98	0.5	7.72	0.25	484	1	484	1
2.		IDEAL	4	44	0.24	15.71	0.2	161.0	1.2	131.75	0.38
3.			8	29.0	4.2	14.6	1.75	105.2	9.8	56.2	2.90
4.			16	26.78	3.24	17.9	1.74	103.7	6.56	30.25	2.33
5.		RHOCOP	4	48.83	3.1	13.71	0.34	210.0	17.4	136	2.98
6.			8	30.86	1.86	17.51	0.77	109.4	9.8	61.71	2.90
7.			16	28.63	6.1	20.7	1.47	107.88	28.76	31.65	2.34
8.	32×32	SOLO	1	356.8	3.67	35.12	0.24	1781	19.17	1781	19.17
9.		IDEAL	4	137.8	8.1	61.1	12.2	582.0	38.2	489.75	26.38
10.			8	113.2	29.2	147.6	41.75	480.2	179.8	257.2	12.90
11.			16	54.0	10.1	176.1	53	206	33.1	129.25	11.6
12.			32	53.5	8.3	115.4	29.3	200.2	40	61	1.40
13.		RHOCOP	4	177.8	34.56	37.9	8.68	718.0	127.4	536	24.18
14.			8	128.9	25.75	64.7	22.3	517.3	60.8	268.7	6.90
15.			16	76.23	12.26	84.6	15.96	282.85	43.4	142.2	4.20
16.			32	44.8	6.7	176.3	7.76	152.4	27.2	60.1	2.4
17.	64×64	SOLO	1	1358	17.67	141.12	1.64	6785	85	6785	85
18.		IDEAL	4	441	35.1	160.3	12.44	2182.0	238.2	1959.75	126.38
19.			8	314.5	29.8	222.1	16.75	1422.3	84.8	1064.7	17.30
20.			16	158.8	11.9	158.4	15.1	643.8	60.6	464.3	13.8
21.			32	86	6.4	362.7	29.6	296.8	19.40	224.8	2.1
22.		RHOCOP	4	480	21.9	105.4	8.4	2356.80	111.2	2149.3	155.88
23.			8	241.2	25.6	89	10.3	1129	76.8	971.8	15.90
24.			16	170.7	10.02	124.6	9.3	734.15	44.4	519.2	25.720
25.			32	90.6	3.7	270.3	11.8	302.66	3.4	243	3.16
26.	128×128	SOLO	1	5548	24.67	641	32.24	27734	169.7	27734	169.7
27.		IDEAL	8	814.6	44.2	391.1	21.72	4018	98.4	3702	46.68
28.			16	504.8	47.3	549.36	101.6	2280.7	159.2	1943.2	29.50
29.			32	257.4	12.2	454.5	27.3	1126	39.1	910.25	16.7
30.			64	188.7	13.73	431.6	16.5	649.2	34.40	476.8	6.20
31.		RHOCOP	8	988.7	104.6	314.2	29.6	4682.0	178.4	3968	62.18
32.			16	578.2	30.7	390	56.7	2540.6	145.8	2137.3	50.80
33.			32	315.5	32.26	432.8	56.96	1358.85	73.4	1009.3	23.90
34.			64	192.4	11.6	417.3	29.76	678.4	39.2	512.5	9.8

TABLE VIII: Experimental Evaluation of RHOCOP, SOLO, and IDEAL Planning Approaches

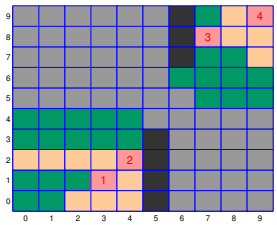
APPENDIX C
COVERAGE PLANNING EXAMPLE



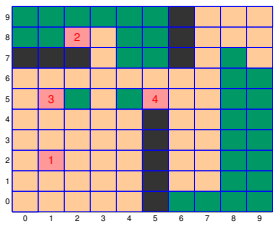
(a) Planning Horizon 0, Active: Robot 1, 2, 3, and 4



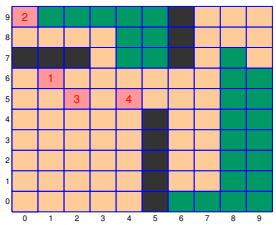
(b) Planning Horizon 1, $L^{(1)}$: 2, Total Covered: 15, Active: Robot 1, 2, 3, and 4



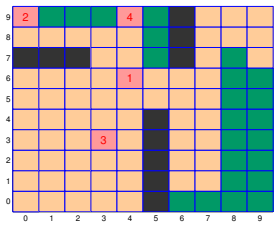
(c) Planning Horizon 2, $L^{(2)}$: 3, Total Covered: 23, Active: Robot 1, 2, and 3, Dormant: Robot 4



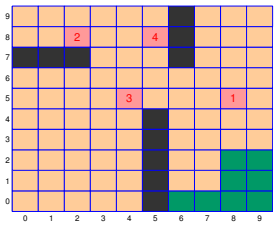
(d) Planning Horizon 8, $L^{(8)}$: 4, Total Covered: 69, Active: Robot 1, 2, 3, and 4



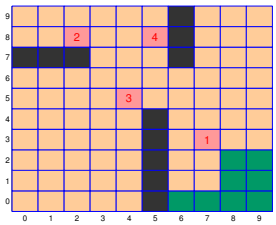
(e) Planning Horizon 9, $L^{(9)}$: 4, Total Covered: 74, Active: Robot 1, 2, 3, and 4



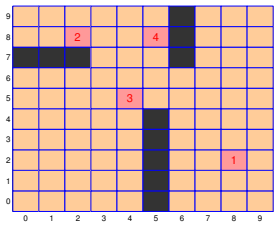
(f) Planning Horizon 10, $L^{(10)}$: 4, Total Covered: 77, Active: Robot 1, 3, and 4, Dormant: Robot 2



(g) Planning Horizon 16, $L^{(16)}$: 5, Total Covered: 92, Active: Robot 1, 3, Dormant: Robot 4, Dropped: Robot 2



(h) Planning Horizon 17, $L^{(17)}$: 5, Total Covered: 92, Active: Robot 1, Dormant: Robot 3 and 4, Dropped: Robot 2



(i) Planning Horizon 22, Coverage is completed

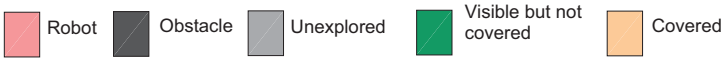


Fig. 7: An Example of Coverage Planning of a 10×10 Workspace by 4 Robots