Persistent Multi-Robot Mapping in an Uncertain Environment

Derek Mitchell and Nathan Michael

Abstract— **This paper proposes a method to deploy teams of robots with constrained energy capacities to persistently maintain a map of an uncertain environment. Typical occupancy map approaches assume a static world; however, we introduce a decay in confidence that degrades the occupancy probability of grid cells and promotes revisitation. Further, sections of the map whose occupancy differs between observations are visited more frequently, while unchanging areas are scheduled less frequently. While naive planning is intractable through the entire space of multi-agent spatio-temporal states, the proposed algorithm decouples planning such that constraints are resolved separately by solving tractable subproblems. We evaluate this approach in simulation and show how the uncertainty of our world model is maintained below an acceptable threshold while the algorithm retains a tractable computation time.**

I. Introduction

Consider the task of deploying a team of robots to persistently map a city block. Incorporating changes due to non-transient effects, such building renovation or the daily relocation of temporary vendors (food trucks, stalls, etc.), can permit a typically static representation to reflect a dynamic environment. It is trivial to recognize these dynamics if robots are deployed to saturate the environment, observing all changes as they occur. However, limitations on team size and energy expenditure restrict how often such observations can be made. We propose a system to mitigate this problem with deployments of energy-constrained robots computed over a sliding-window horizon to factor observations into future deployments.

While many mapping techniques might apply here, we use occupancy grids [1]–[3] to provide a simple and reliable means of storing a probabilistic representation of the physical structure of the environment. Dynamic Pose Graphs [4] offer an alternative approach that stores observations and their corresponding locations in a graph structure, while pruning contributions that disagree with more current measurements. This is an effective means of building a dynamic map while mitigating localization errors, but the transient quality of nodes makes it difficult to track the environment dynamics. Other approaches, such as such as Gaussian Process representations [5] or Hierarchical Gaussian Mixture Models [6], can provide a representation of the full map with sparse data by using the properties of a normal distribution to describe the space between measurements. However, as environments are learned with high confidence, it becomes increasingly difficult to register changes. As occupancy grids maintain the likelihood of intersecting an object over regularly spaced grid cells, we can apply an exponential decay to allow for newer observations to have a greater impact.

In active perception tasks, the utility of observations are typically determined by the entropy reduced in querying the sensors at a location, referred to as the Mutual Information (MI) of an observation. An early work uses a single-step controller to drive robots towards candidate observations with high MI relative to a spatiotemporal Gaussian Process environment model [7]. Subsequent works have improved on the quality of this measure by simplifying computation at the cost of optimality [8] or by refining it with an objective learned from human input [9]. Other works apply MI to unique environment models, such as Gaussian Markov Random Fields [10]. MI has been proven to be a viable utility measure, but these recent works have focused on myopic, reactive controllers to drive robots. Here, we use the improved MI from [8] to determine salient observations, but apply a more sophisticated planner to generate trajectories for a team of energy constrained robots.

The challenge of task scheduling and assignment is addressed in operations research by optimizing load-balancing algorithms starting from the Traveling Salesman Problem and the Vehicle Routing Problem [11], [12] and extending to a host of complex variants [13], [14]. Recent works have sought to draw some of this insight into the robotics field by increasing robustness through the use of finite horizons [15] or by integrating heuristics to improve the computation speed at the cost of optimality [16], [17]. The system presented here applies these techniques to the persistent exploration problem, computing efficient, energy-aware deployments to build a consistent map of a dynamic environment.

The main contributions of this paper are:

- An occupancy grid representation that degrades the contribution of old measurements
- A framework that plans deployments of energy-constrained robots in the above representation
- A system that iterates the planner over a sliding-window horizon to persistently extend plans

II. Problem Definition

Given a map representation that exhibits decay in the contribution of observations, we seek to deploy a team of energy-constrained robots from a starting location, x_s, to persistently improve map accuracy. We use a 3D occupancy grid to provide a probabilistic representation of the environment, with each cell storing the occupancy likelihood $p(m_i|o_{1:n})$ for a cell i in map m given a set of observations $o_{1:n}$. The standard assumptions that apply here are: 1) Cells are independent ($p(m|o_{1:n}) = \prod_i p(m_i|o_{1:n})$) 2) robots have

Derek Mitchell and Nathan Michael are affiliated with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {derekm, nmichael}@cmu.edu.

perfect localization, and 3) unobserved cells begin with a uniform prior ($p(m_i) = p(\neg m_i) = 0.5$). The state of each cell is evaluated by comparing against user-defined threshold values, where $p(m_i|o_{1:n}) \geq \gamma_{\text{occ}}$ indicates an occupied cell, $p(m_i|o_{1:n}) \leq \gamma_{\text{free}}$ indicates a free cell, and all other cells are unknown. Each cell is updated with new observations using the *inverse measurement model* [18]:

$$\text{logit}(p(m_i|o_{1:n})) = \text{logit}(p(m_i|o_n)) - \text{logit}(p(m_i)) + \text{logit}(p(m_i|o_{1:n-1})), \quad (1)$$

where $\text{logit}(p) = \log \frac{p}{1-p}$ is the log-odds representation of a probability p. We induce decay in old measurements by applying an exponential function directly to the log-odds likelihood:

$$\text{logit}(p(t, m_i|o_{1:n})) = e^{-\alpha_i(t - t_{\text{last,i}})}\,\text{logit}(p(m_i|o_{1:n})), \quad (2)$$

where t is the current time, $t_{\text{last,i}}$ is the time cell i was last observed, and α_i is a parameter we tune to adjust the rate of decay. This relation causes $p(m_i, t)$ to approach $p(m_i) = 0.5$ as t approaches infinity, simulating an increase in uncertainty as time progresses.

To compute deployments, given the dynamic map representation described above, we compute a set of energy-constrained routes over a sliding-window horizon by generating plans at a time t_{start} to time t_{end}, then advancing both terms by duration d_{adv}, updating the map, and recomputing. The core problem solved, on a per-horizon basis, is to compute the set of routes that append to or extend previous plans to collect observations that maximally improve map accuracy.

III. Methodology

We approach the above problem with the system described in Fig. 1. Given an occupancy grid that evolves according to (1) and (2), we adapt to perceived changes by adjusting α_i online. As cells are consistently observed as being in the same state α_i decreases, resulting in a slower decay of confidence

$$\alpha_{m_i} = \alpha_{\max} - 0.5(\alpha_{\max} - \alpha_{m_i}).$$

Likewise, cells with states that vary between observations will see α_i increase, thus increasing the frequency that observations will be required

$$\alpha_{m_i} = \alpha_{\min} + 0.5(\alpha_{m_i} - \alpha_{\min}),$$

where $\alpha_{\max}$ and $\alpha_{\min}$ are user-specified bounds on the decay rate. As we only seek to prove the viability of the planner on a time-dependent map, we have chosen a naive strategy which simply steps the decay rate asymptotically towards a set value whenever a measurement is taken. The sensor model simulates a forward-facing ranging sensor that extends beams over a 120 degree horizontal span and a 60 degree vertical span for a maximum range of 1..5 meters. Any cell a beam passes through is updated with $p(m_i|o_n) = \gamma_{\text{miss}}$ and any cell in which a beam intersects and object is updated

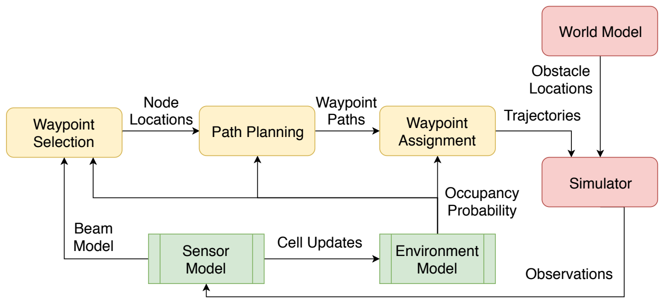


Fig. 1: System diagram. Yellow blocks represent the planning pipeline, green blocks handle observation generation, and red blocks execute plans within a simulated environment.

with $p(m_i|o_n) = \gamma_{\text{hit}}$, where γ_{miss} and γ_{hit} are user-defined parameters.

The following sections show how the planning pipeline computes routes, given the above environment and sensor models. First, Section III-A describes how a set of reachable, informative poses (waypoints) are chosen from the occupancy grid based on the Mutual Information they provide. Then, Section III-B details the path planning algorithm, which generates paths pairwise between all waypoints to evaluate travel cost and feasibility. Finally, Section III-C computes routes that distribute waypoints among robots and updates the plan over the current horizon.

A. Waypoint Selection

The first component of the planning pipeline samples a set of informative waypoints small enough for the system to process in a reasonable amount of time. The value of a set of waypoints is determined by the information they provide. Cauchy-Schwarz Quadratic Mutual Information (CSQMI) [8] provides a measure of this value, reflecting the diminishing returns property exhibited by sets of points that view overlapping regions. For the sake of brevity, we ask the reader to refer to [8] for a derivation of CSQMI and understand that we are able to compute an appropriate utility function from a set of robot poses.

To build the set of waypoints, we first compute the set of reachable poses, directed towards the nearest unknown frontier, then iteratively select the pose that contributes the most additional information until we have collected N_w waypoints. The reachable set is generated using our implementation of Dijkstra's algorithm described in Sect. III-B, where shortest distance paths are grown from x_s until all reachable space is explored.

The Waypoint Selection algorithm is detailed in Algorithm 1. First, the set of all frontier cells is established in V_{frontier} as the set of all cells that are unknown ($\gamma_{free} < p(t, m_i|o_{1:n}) < \gamma_{occ}$), but bordered by at least one free cell. Next, we greedily cluster these cells in C_{frontier} using the GreedyClustering algorithm which initially appends a cell from V_{frontier}, then adds all cells from V_{frontier} within a distance threshold. This process is repeated until all cells from V_{frontier} are assigned to a cluster in C_{frontier}. The set of all reachable free cells is assigned to V_{free} using the DijkstraReachables

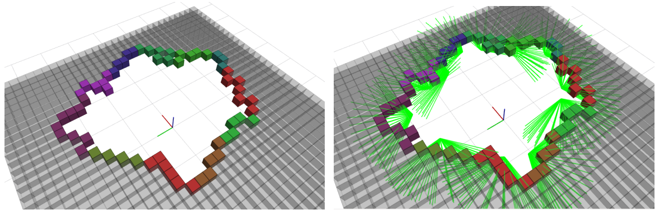


Fig. 2: Depiction of a 2D map, with shaded boxes representing unknown voxels. (Left) Colored blocks represent frontier cells, clustered into groups. (Right) Green lines represent beam measurements from 12 unique poses chosen for maximum information value.

Algorithm 1: Waypoint Selection

```
1: procedure WAYPOINTSELECT($m, x_s, N_w, \epsilon$)
2:     $V_{\text{frontier}} \leftarrow$ Set of all frontier cells in $m$
3:     $C_{\text{frontier}} = \text{GreedyClustering}(V_{\text{frontier}})$
4:     $V_{\text{free}} \leftarrow \text{DijkstraReachables}(m, x_s)$
5:     $P \leftarrow \emptyset$                          ▷ Set of pose candidates
6:     for each $m_f \in V_{\text{free}}$ do
7:         $c_{f,\text{near}} = \arg\min_{c_f \in C_{\text{free}}} \|\text{cent}(m_f) - \text{cent}(c_f)\|^2$
8:         if $\|\text{cent}(m_f) - \text{cent}(c_{f,\text{near}})\| < \text{max\_range}$ then
9:             $\theta = \text{LookAt}(m_f, c_{f,\text{near}})$
10:            $P = P \cup \{\text{cent}(m_f), \theta\}$
11:        end if
12:    end for
13:    $W \leftarrow \emptyset$                          ▷ Final waypoint poses
14:    while $|W| < N_w$ do
15:        $w_{\text{next}} = \arg\max_{l \in L} \text{CSQMI}(m, t_{\text{start}}, W \cup p)$
16:        $d = \text{CSQMI}(m, t_{\text{start}}, W \cup w_{\text{next}})$
17:            $-\text{CSQMI}(m, t_{\text{start}}, W)$
18:        if $d < \epsilon$ then
19:            return $W$
20:        end if
21:        $W = W \cup w_{\text{next}}$
22:    end while
23:    return $W$
24: end procedure
```

algorithm, which uses the algorithm in Sect. III-B to find the shortest distance path to all cells reachable from x_s.

Once the clusters and reachable cells are collected, candidate poses are constructed at each reachable cell. Lines 6 to 12 show this process in detail, where $\text{cent}(x)$ determines the centroid of a cluster or cell and $\text{LookAt}(x, c)$ determines the angle that points to the centroid of cluster c from cell x. After the candidate poses are collected in P, we select the N_w most informative poses. In lines 14 to 22, $\text{CSQMI}(m, t_{\text{start}}, W)$ is used to determine the information value of a set of poses, W, at the horizon start time. The pose which provides W with the most additional information is found (line 15) and appended to W if it provides value added greater than a user defined threshold, ϵ. Otherwise, the algorithm breaks early, preventing the inclusion of uninformative waypoints in the case where the environment is already sufficiently known. Figure 2 shows the clustering and selection aspects of this approach.

Algorithm 2: Dijkstra's Algorithm with Dynamic Occupancy

```
1: procedure DYNAMICDIJKSTRAS($m, s, t_s, v_{\text{traverse}}$)
2:     $Q = \{s\}$                                  ▷ Open node set
3:     for $i = 0, \ldots, |m|$ do
4:         $\text{dist}[i] = \infty$                ▷ Distance from start
5:         $\text{prev}[i] = \emptyset$             ▷ Previous cell index
6:     end for
7:     $\text{dist}[s] = 0$
8:     while $Q \neq \emptyset$ do
9:         $i \leftarrow Q$
10:        $Q = Q \setminus i$
11:        for each $j \in \text{Neighbors}(i)$ do
12:            $d = \text{dist}[i] + \text{length}(i, j)$
13:            $t = d / v_{\text{traverse}} + t_s$
14:            if $d < \text{dist}[j]$ and $p(m_j, t) < \gamma_{\text{free}}$ then
15:                $\text{dist}[j] = d$
16:                $\text{prev}[j] = i$
17:                $Q = Q \cup j$
18:            end if
19:        end for
20:    end while
21:    return dist, prev
22: end procedure
```

B. Path Planning

In this section, we generate the paths to be used in Sect. III-C using a variant of Dijkstra's algorithm, described in Algorithm 2, which is modified to plan only through free cells as determined by $p(t, m|o_{1:n}) \leq \gamma_{\text{free}}$. Paths are generated from a starting cell s and extended through neighbors until all reachable cells are explored. If we also provide the algorithm with the starting time t_s and the assumed constant speed of the robots v_{traverse}, we ensure that paths will always be feasible despite the introduction of confidence decay.

Dijkstra's algorithm typically begins with an open set of cell indexes Q, containing only the start index (line 2). Then neighbors are evaluated to see if the path through the current index belongs to the shortest path. The vectors, dist and prev, store The closest distance to s and the previous index in the shortest path available are stored in the vectors, dist and prev, which are updated for each cell processed. Once the algorithm finishes, we are left with a set of reachable cells whose corresponding paths are determined by iterating backwards through prev. The key change to the traditional Dijkstra's algorithm comes from evaluating the viability of neighbors to ensure that the cell at index j will be free at time t (line 14), where t represents the time of arrival from the start index under the assumption of a constant traversal velocity, v_{traverse}.

Applying this algorithm to each waypoint, we derive the shortest distance trajectories between waypoints. Given that we have encoded the informative value of waypoints in W, we optimize utility by maximizing the number of waypoints addressed per route. This is preferable to heavily investing in computing informative paths at this point, as many paths between waypoints will not contribute to the final solution.

C. Waypoint Assignment

Once the paths connecting W are defined, we rely on the techniques described in [15] to assign waypoints to robots. Their main contribution was to approach periodic task assignment by extending schedules through locally-static finite horizons. New waypoints are generated from periodic task generators and location or timing can be adjusted between horizons to improve the reliability of the overall solution. This effectively blends the quality of a complex convex optimization solver with the responsiveness of closed-loop control. In the system we propose here, this approach is implemented to provide scheduled waypoint visitations over finite horizons concurrently with robots exploring the environment and refining the map. Updates to the map then inform subsequent scheduling horizons to improve the accuracy of the resulting plans.

The basic approach solves a Vehicle Routing Problem with Time Windows (VRPTW) defined by a graph $G = (A, E)$ of nodes A and edges E. The node set A includes a node corresponding to x_s at the start, S_o, and end, S_g of the horizon, a node for each waypoint in W, and a virtual node for each robot that will be active at t_{start}. The virtual nodes, which correspond to the last planned waypoint visit before t_{start}, permit the extension of previously computed plans. Edges in E use the path cost between waypoints as computed by the DynamicDijkstras algorithm. The VRPTW takes the following form:

$$\min_x \sum_{k\in K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{3}$$

$$s.t. \sum_{k\in K} \sum_{j\in V} x_{i,j}^k = 1 \qquad \forall i \in V \tag{4}$$

$$\sum_{j\in V\cup S_g} x_{oj}^k = 1 \qquad \forall k \in K \tag{5}$$

$$\sum_{i\in S_o\cup V} x_{ig}^k = 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{i\in N} x_{ih}^k - \sum_{j\in N} x_{hj}^k = 0 \qquad \forall h \in V, k \in K \tag{7}$$

$$s_i^k - s_j^k + t_{ij} - Z(1 - x_{ij}^k) \leq 0 \qquad \forall (i,j) \in A, k \in K \tag{8}$$

$$a_i \leq s_i^k \leq b_i \qquad \forall i \in N, k \in K \tag{9}$$

$$\sum_{(i,j)\in A} B_{ij} x_{ij}^k \leq B_{\text{max}} \qquad \forall k \in K \tag{10}$$

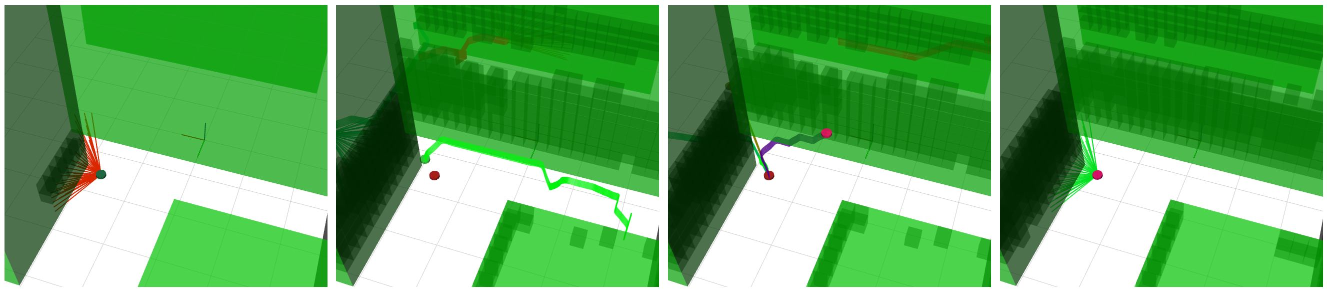
where $x_{ij}^k \in 0,1$ determines if robot k in the set of robots K travels the path between waypoints i and j, $s_i^k \in \mathbb{R}^+$ indicates the time at which waypoint i is visited by robot k, c_{ij} represents the cost of traveling the path between i and j, and a_i and b_i define the lower and upper bounds of the time-window for waypoint i. We set c_{ij} to equal the travel time along the path between waypoints such that the objective (3) will minimize the total travel time over all robots. Constraints (4)-(7) ensure that only one robot leaves each waypoint, each robot begins at the start node S_o and ends at the goal node S_g, and any robots arriving at a waypoint must leave it. Constraint (8) ensures that a robot that travels the edge (i, j) arrives at j after s_i^k plus the duration of travel t_{ij}. Constraint (9) enforces the time-window bounds and constraint (10) limits energy expenditure to the battery limit B_{max}, where $B_{ij} = \max(t_{ij}, a_j - a_i)$ to conservatively estimate the expenditure of energy assuming a constant discharge rate.

The above Mixed-Integer Linear Program can be simplified by relaxing constraint (4), resulting in a set of constraints that are all agnostic of robot assignment. It is shown in [19] that the Lagrangian Dual obtained as a result of this relaxation takes the form:

$$z_{LD}(\lambda) = |K| \left(\min_x (c_{ij} - \lambda_i) x_{ij}^k + \sum_{i\in V} \lambda_i \right), \tag{11}$$

subject to (5)-(10). Note here that $\min_x (c_{ij} - \lambda_i) x_{ij}^k$, subject to the same constraints, is an expression of the Elementary Shortest Path Problem with Resource Constraints (ESPPRC) for a single robot with costs reduced by λ. Maximization of this dual objective is then a matter of testing values of λ against the set of all feasible single-robot routes through the waypoint set. As this is impractical, we rely on the Stabilized Cutting-Plane Algorithm (SCPA) [19], which alternates between building a set of Pareto-optimal paths and searching for the optimal λ. In each iteration, we solve the ESPPRC [20] for a given set of λ values and append the resulting path(s) to the constraint set bounding (11), then we find the values of λ which optimize z_{LD}. When the gap between z_{LD} and the minimal-cost path is small enough, the set of paths that actively constrain the dual objective become the solution set. As the constraint on waypoint visitation has been relaxed, the final solution must be refined such that each waypoint is visited only once. For each waypoint with multiple visitations, we generate a set of problem formulations, each with an additional constraint that either forces or excludes one of the transitions x_{ij}. This process is applied recursively within each branch until all variants of the problem are explored. At this point, we return the solution with the minimum objective value as the set of optimal routes.

In the system presented here, the VRPTW can be constructed in a straight-forward manner. The costs c_{ij} and durations t_{ij} can both be represented by path traversal duration $\text{dist}[i][j]/v_{\text{traverse}}$ between waypoint i and j. The time-window bounds a_i and b_i consider the paths to and from x_s. The lower bound a_i is equal to the travel time between x_s and i ($a_i = t_{\text{start}} + t_{oi}$) as the earliest visitation possible occurs by traveling directly from start. The upper bound b_i is set to the latest moment the edge (i, g) can be traveled before the confidence decay causes a cell to become unknown. We compute this for a single cell with a fairly simple inversion



(a) $(t = 0)$ (b) $(t = 500)$ (c) $(t = 510)$ (d) $(t = 1422)$

Fig. 3: System in operation. Green blocks are obstacles. Cylinders are robots with tails showing the last 5 seconds of motion. Black cubes are voxels designated occupied within the occupancy map. Free and unknown cells are not indicated to preserve visibility. (a) Initial measurement. (b) One robot returns to depot while others explore behind walls. (c) After robots return, two more depart. (d) Final measurement.

of the confidence decay (2):

$$t_{\text{unknown},i} = \frac{1}{\alpha_i} \log \left(\frac{\text{logit}(p(m_i|o_{1:n}))}{\text{logit}(\gamma_{\text{free}})} \right) + t_{\text{last,i}}, \qquad (12)$$

The latest feasible departure for cell i is determined by computing the $t_{\text{unknown},i}$ for each cell along the path from i to g and subtracting the time required to travel to the cell from i, which we approximate as the travel distance divided by v_{traverse}. The latest feasible departure for the whole path is then the earliest time a robot can begin a path before any cell becomes unknown:

$$b_i = \min_{j \in \text{path}(i, x_s)} t_{\text{unknown},j} - \frac{\text{distance}(i,j)}{v_{\text{traverse}}},$$

where distance(i, j) is the path length to cell j along the path connecting waypoint i to x_s. Closing the time window at this value of b_i ensures that the option of returning to the start position is always available.

IV. Simulation Evaluation

We evaluate the proposed system under controlled conditions by computing deployments of aerial robots for extended operation in a simulated environment. Typical active perception applications consider the reduction of entropy in the environment model over time as a means of showing the accuracy and efficiency of the approach. We evaluate entropy for the given approach and compare against a naive, greedy algorithm to show how the proposed system is better suited to handle the constant degradation of certainty. Given the energy-constraints considered, we also evaluate the planned durations and show feasible operation that consistently uses the majority of each robot's capacity. Additionally, we provide insight into the performance of our system by plotting the number of robots deployed over time and showing a comparison graph of computation times.

For the simulation experiments detailed here, we used the world configuration shown in Fig. 3. Robots start from a single position and explore the space bounded by an axis-aligned rectangle while avoiding obstacles (green blocks). The environment model is divided into grid spaces at a resolution of 0.25 meters along each axis. The probability thresholds are set at $\gamma_{\text{free}} = 0.13$ and $\gamma_{\text{occ}} = 0.7$, with $\gamma_{\text{hit}} = 0.7$ and $\gamma_{\text{miss}} = 0.3$. The maximum number of waypoints, N_w, is chosen to be 12 for the simulation tests to allow for fast computation of finite horizons of 40 seconds long. For convenience, we define energy capacity in terms of flight duration, which we set to 100 seconds.

We simulated operation for 3000 seconds with no limit on the number of robots deployed and compared against a greedy algorithm that directs a robot to the next best waypoint immediately after it collects a measurement. With the greedy approach, we assume that the map is updated online and shared between all robots, allowing for the optimal choice of next best observation. We evaluate the efficiency of our approach in the left image of Fig. 4 using entropy, defined as:

$$\begin{aligned} H(m,t) = - \sum_{m_i \in m} & p(m_i,t) \log_2 p(m_i,t) \\ & + (1 - p(m_i,t)) \log_2 (1 - p(m_i,t)), \qquad (13) \end{aligned}$$

and the Kullback-Leibler divergence for a multivariate binomial probability distribution:

$$\begin{aligned} D_{KL}(m,t) = \sum_{m_i \in m} & W(m_i,t) \log_2 \frac{W(m_i,t)}{p(m_i,t} \\ & + (1 - W(m_i,t)) \log_2 \left(\frac{1 - W(m_i,t)}{1 - p(m_i,t} \right), \qquad (14) \end{aligned}$$

where $W(m_i, t)$ is the occupancy likelihood in the oracle model derived from the simulated world configuration. Entropy reflects the confidence in the map, trending towards zero the closer each cell probability approaches 1.0 or 0.0. Divergence, however, highlights model accuracy by trending towards zero the closer the occupancy likelihood approaches the oracle model. The greedy algorithm shows a fast initial decrease of entropy and divergence, but slows under the effect of confidence decay. This trend highlights the weakness of a decoupled planner, where coverage responsibility is not effectively distributed among robots. By applying the principled waypoint assignment of our approach, we are able to consistently improve accuracy and confidence while additionally respecting energy constraints.

The middle image of Fig. 4 shows a histogram of energy expenditure over the course of operation. For the optimal planner, this histogram should be dominated by the rightmost

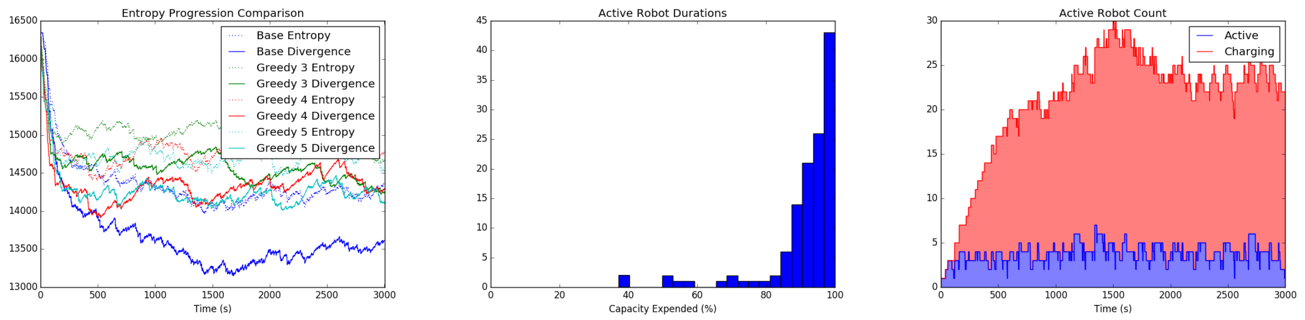


Fig. 4: (Left) The evolution of confidence in the environment model for the proposed system compared against a greedy approach with 3, 4, and 5 robots. *Entropy* provides a measure of confidence in the entire map, while *Divergence* provides a measure of how accurate the model is relative to the true world. The greedy approaches initially converge faster, but are less able to leverage their team size to mitigate the constant increase in uncertainty. (Middle) Histogram of capacity expenditure for planned routes. The more routes in the rightmost bin, the more utility the system is deriving from the limited energy capacity. (Right) Team size required for the simulated operation assuming charge time is 5x flight time. Blue section counts simultaneously active robots while red section counts robots charging.

bin, showing that robots are consistently expending most of their capacity over each route. By extending paths iteratively over multiple horizons, we are able to ensure that robots remain active for the duration of their flight time. As such, we show that 90.3% of the deployed robots expend more than 80% of their capacity.

In the right image of Fig. 4, we evaluate the number of robots deployed when robots must passively charge for five times their operation time before being able to return to operation. Actively operating robots are counted in the blue section, showing that more are deployed as the map is revealed, with an average number of roughly 3.79 robots deployed over the duration of the simulation. The red section shows the additional number of robots charging at any given time, where the total number of robots invested is indicated by the top red line. The data here shows that running the system with the described charging configuration would require at least 30 robots to run without interruption. Similar evaluation can be made for various charging methods, allowing a user to determine how many robots might be required to cover a given type of environment.

This simulation was performed on a Dell PowerEdge T420 with an Intel Xeon E52450 v2 2.50GHz processor and 16 GB of RAM. The computation time for each scheduling horizon is depicted in Fig. 5. The actual computation time may differ depending on hardware, but the figure here provides some insight as to the relative computational burdens of components of the system. Note that the greatest burden comes from the Waypoint Selection algorithm. This is likely due the naive nature of the approach which computes CSQMI for all reachable free spaces N_w times. In future works, we will explore approaches that drastically reduce the burden on this component to allow the other algorithms more time to address harder problems.

V. Conclusion

This paper presents a system designed to deploy energy constrained robots to persistently cover an uncertain environment. We described adaptations to a standard environment model and presented an integration of path planning and task assignment techniques to persistently generate energy-feasible routes. As this is early in the development stage, there is much room for improvement in the selection and implementation of individual components. However, we present a solid foundational structure capable of generating plans quickly that have a high likelihood of driving robots only through free space while ensuring that robots do not exceed their battery capacity while in operation.

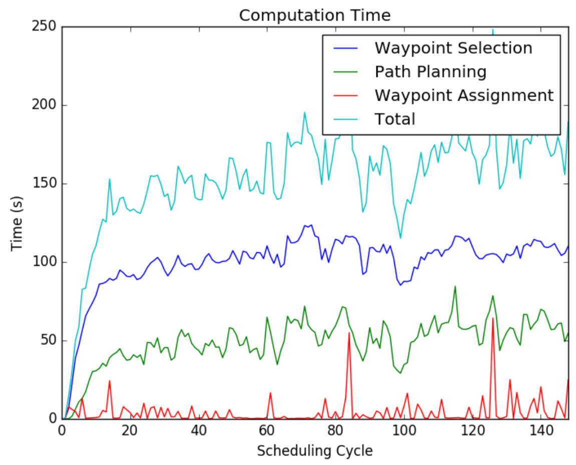


Fig. 5: Computation time required to generate plans for each horizon showing the relative time required by individual components of the algorithm. Complexity remains consistent once steady-state is reached.

As such, there are many areas we will seek to improve in future works. In particular, Waypoint Selection will benefit from attention towards improved computation time and a better spatial distribution of candidate waypoints. Additionally, the development of a complimentary informative path algorithm to refine trajectories after scheduling will improve the rate at which map entropy decreases. In future works, we will explore options such as these and evaluate how they interact within the context of the full system.

References

[1] M. C. Martin and H. Moravec, "Robot evidence grids," March 1996. [Online]. Available: https://www.frc.ri.cmu.edu/~hpm/project.archive/robot.papers/1996/RobotEvidenceGrids.pdf

[2] S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, Sep 2003. [Online]. Available: https://doi.org/10.1023/A:1025584807625

[3] T. Colleens, J. J. Colleens, and D. Ryan, "Occupancy grid mapping: An empirical evaluation," in *Mediterranean Conference on Control Automation*, June 2007, pp. 1–6. [Online]. Available: https://doi.org/10.1109/MED.2007.4433772

[4] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelli. Robots and Systs.*, Oct 2012, pp. 1871–1878. [Online]. Available: https://doi.org/10.1109/IROS.2012.6385561

[5] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012. [Online]. Available: https://doi.org/10.1177/0278364911421039

[6] S. Srivastava and N. Michael, "Approximate continuous belief distributions for precise autonomous inspection," in *IEEE Int. Sym. on Safety Security and Rescue Robotics*, Oct 2016, pp. 74–80. [Online]. Available: https://doi.org/10.1109/SSRR.2016.7784280

[7] A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser, "Modeling and decision making in spatio-temporal processes for environmental surveillance," in *Proc. of the IEEE Intl. Conf. on Robot. Auto.*, May 2010, pp. 5490–5497. [Online]. Available: https://doi.org/10.1109/ROBOT.2010.5509934

[8] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information," in *Proc. of the IEEE Intl. Conf. on Robot. Auto.*, May 2015, pp. 4791–4798. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7139865

[9] H. Liu and M. A. Hsieh, "Neural Network Aided Information Theoretic Exploration," in *IEEE Int. Sym. on Safety Security and Rescue Robotics*, Aug 2018, pp. 1–7. [Online]. Available: https://doi.org/10.1109/SSRR.2018.8468650

[10] C. Wang, T. Li, M. Q.-H. Meng, and C. D. Silva, "Efficient Mobile Robot Exploration with Gaussian Markov Random Fields in 3D Environments," in *Proc. of the IEEE Intl. Conf. on Robot. Auto.*, May 2018, pp. 5015–5021. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460788

[11] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International Transactions in Operational Research*, vol. 7, no. 4, pp. 285 – 300, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0969601600000034

[12] P. Toth and D. Vigo, *The Vehicle Routing Problem*, 2002, ch. 2. Branch-And-Bound Algorithms for the Capacitated VRP, pp. 29–51. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.ch2

[13] Y. Chan and S. Baker, "The multiple depot, multiple traveling salesmen facility-location problem: Vehicle range, service frequency, and heuristic implementations," *Mathematical and Computer Modelling*, vol. 41, no. 8, pp. 1035 – 1053, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717705001639

[14] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *European Journal of Operational Research*, vol. 218, no. 1, pp. 1 – 6, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221711006692

[15] E. Stump and N. Michael, "Multi-robot persistent surveillance planning as a Vehicle Routing Problem," in *IEEE Int. Conf. on Autom. Sci. Eng.*, Aug 2011, pp. 569–575. [Online]. Available: https://ieeexplore.ieee.org/document/6042503/

[16] D. Levy, K. Sundar, and S. Rathinam, "Heuristics for routing heterogeneous unmanned vehicles with fuel constraints," *Mathematical Problems in Engineering*, vol. 2014, Article ID 131450, 12 Pages, 2014. [Online]. Available: https://doi.org/10.1155/2014/131450

[17] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, "Multi-Robot Persistent Coverage with Stochastic Task Costs," in *Proc. of the IEEE/RSJ Int. Conf. on Intelli. Robots and Systs.*, Hamburg, Germany, Sept. 2015, pp. 3401–3406. [Online]. Available: https://doi.org/10.1109/IROS.2015.7353851

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2008.

[19] B. Kallehauge, J. Larsen, and O. B. Madsen, "Lagrangian duality applied to the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 33, no. 5, pp. 1464 – 1487, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054804003028

[20] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks*, vol. 44, no. 3, pp. 216–229, 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20033