Distributed Multi-robot Area Examination and Search for Clustered Targets with Limited Field of Views

Jun Chen[a,b], Jiaqing Ma[a], Qi Mao[a,b], Fei Xie[a,b]

[a]School of Electrical and Automation Engineering, Nanjing Normal University, No.2 Xuelin Road, Nanjing, 210023, Jiangsu, China
[b]Jiangsu Key Laboratory of 3D Printing Equipment and Manufacturing, No.2 Xuelin Road, Nanjing, 210023, Jiangsu, China

Abstract

Motivated by coral reef information gathering using multiple unmanned underwater vehicles (UUVs) under the sea, this paper addresses the problem of distributed multi-robot area examination and target search, which requires a team of robots to select, navigate to, and look over a sequence of locations in a known area to search for targets of interest. Robots must reduce the speed during close examination to acquisitive precise information of examined locations and detect if targets exist or not. The key technical challenges for efficiently complete such tasks stem from the facts that the robot has limited sensor field of view and is required to perform the search and examination of targets over an extended area without prior knowledge on their distribution. In such scenario, it is computationally intractable to design the optimal strategy to minimize the time it takes to find all targets. In this paper, we compare two novel target search strategies, the greedy algorithm that thoroughly exploits a detected target cluster, and a Thompson-sampling-based algorithm that balances the trade-off between exploitation – further surveying surroundings of target-detected areas – and exploration – traveling to a randomly selected un-visited area. Simulations are carried out to validate the efficiency of the proposed methods for tasks in which targets are clustered and precise examination is required for target search compared to a baseline sweeping algorithm.

Keywords: multi-robot systems, sensor-based planning, active information gathering

1. Introduction

The world's coral reefs are under threat from stressors such as destructive fishing practices, overfishing, coastal development, and water pollution [1]. Robots are developed in recent years for coral reef monitoring, health assessment, and restoration to help recover an damaged marine ecosystem [2, 3]. An important task is for the robots to explore under sea and search for damaged coral reefs (targets) by gathering and analysing high-precision images so as to conduct recovery actions such as transplantation, coral gardening, and substrate enhancement [1]. This paper seeks to develop efficient target search and examination algorithms using multiple autonomous robots. Similar application scenarios include trash collection in a desert area, flaw inspection and repair on surfaces of buildings and large machinery, detection and sample collection of vegetation in a nature reserve, etc.

The challenges in such tasks are threefold. Firstly, targets are static and tend to be clustered, i.e., they are likely to be located next to one another though their distribution is unknown, and also occupy only a small portion of entire search space, as shown in Figure 1. Secondly, the search space is much larger than robots' field of views for which robots need to navigate a long distance for the target search. Thirdly, the detection of targets requires thorough investigation of search areas which could be time-consuming. Although existing work addresses related problems of target search and environment exploration, reviewed in Section 2, however, none of these problems takes all three challenges into consideration and focuses on improving the search efficiency under such application scenarios.

In this paper, we propose two efficient multi-robot area examination and target search algorithms to address those challenges. A novel brute force greedy algorithm is presented which allows robots to look for target clusters by random search and exploit the surroundings of a detected targets, to compare with a novel probabilistic search algorithm based on Thompson sampling (TS) that learns coarse distribution of target clusters as more detections are gained and drives robots to areas with higher probabilities to find targets while allows them to

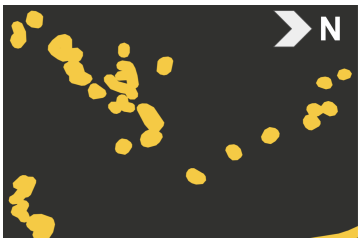


Figure 1: Coral reef locations (yellow) in the red sea at 21°52'30.3"N 38°49'18.2"E. The figure shows an area of approximately 10 km × 6 km. The smallest coral reef cluster shown is of around 0.5 km in diameter. Data source: `https://resourcewatch.org`.

explore un-visited areas at the same time.

The TS-based algorithm adopts a modified Bernoulli TS in which robots repeatedly select search areas based on the posterior of the target locations while taking into account the time it takes to travel and gather information. Additionally, we devise a posterior filtering for the TS-based algorithm to improve robots' search and examination performance near the boundary of target clusters.

We demonstrate the efficacy of our proposed algorithms via series of simulations comparing to a baseline sweeping algorithm. Our results show that the greedy algorithm performs better only when targets are highly aggregated due to its deterministic feature which results in efficient exploitation of a cluster but inefficient exploration of target clusters. On the contrary, the TS-based algorithm balances the trade-off between exploitation and exploration, leading to outperforming in finding targets in varying levels of aggregation.

2. Related Problems

This section investigates two classes of problems that provides potential solutions to our addressed problem, namely the coverage path planning problem and the exploration problem. We demonstrate that though these problems share similar formulation, however, no existing method solves our search and examination problem efficiently.

2.1. Coverage Path Planning

Coverage path planning (CPP) methods, surveyed in [4], find collision-free paths that ensure one or multiple robots to pass over all points in a *known* environment. CPP algorithms often generate zigzag [5, 6] or spiral [7, 8] paths that cover the entire search space. Other methods use randomized or sampling strategies [9, 10] to sweep the complete search space, aiming to cover the majority of areas in the space by successively finding locations where the robot can maximize information gathering. Various task allocation approaches are proposed for a team of robots to complete CPP tasks cooperatively [11, 12, 13, 14]. However, without prior information on the target distribution, robots must spend time investigating all locations thoroughly along a coverage path, resulting in inefficient target search especially when the targets occupy only a small portion of the space.

2.2. Exploration

Exploration addresses the problem of one or more mobile robots planning paths to investigate all locations in an *unknown* environment in order to build a map of it or to ensure that all targets of interest in the environment are detected. Authors propose planning approaches to find trajectories along which one [15, 16, 17] or multiple [18, 19, 20, 21] robots visit frontiers (i.e., regions on the boundary between open space and unexplored space) , or local minima of a potential information field [22, 23] to reduce the uncertainty of the exploration space. While these approaches are more suitable for indoor target search scenarios where the space is relatively small compared to the robot's sensing range, some articles focus on efficiently spreading out a robot team in a *large* search space online to minimize the time for finding all targets cooperatively [24, 25]. All above-mentioned approaches seek to decide the next available goal to sweep the unknown environment while balancing task allocation to agents. Algorithms are also proposed to take advantage of partial prior of target distribution for effective search [26, 27, 28, 29], e.g., targets eliminate odor that can be detected at a distance.

Different from the exploration problem, our addressed search and examination problem assumes that robots perform task in a *known* environment, in which case a pre-assigned path for each robot can always perform better than exploration algorithms through allowing robots to cooperatively sweep all locations with the smallest moving distances, an example of which is the baseline sweeping strategy conducted in Section 6. In other words, instead of exploring the map for target search, patrolling, surveillance, etc, we focus only on planning paths to find targets quickly. Therefore, while exploration algorithms achieve our search and examination goals eventually, the solutions are not efficient.

3. Problem Formulation

A number of static targets are located in a known task space $E \subset \mathbb{R}^2$. The set of all target locations, i.e.,

the areas of interest (AoI) in E is denoted $AoI \subset E$. We denote p the position of a point in E. A homogeneous robot team of n agents is tasked to detect all points $p \in AoI$ using perfect isotropic sensors with finite detection radius fov. Our approach can be extended to handle heterogeneous and anisotropic sensors by applying the similar strategy from our recent work [30], where we map each sensor to an isotropic field of view with equivalent detection capability. Robots are able to localize their position $Q = q_1, ..., q_n$ as well as locations of targets. During traveling, a robot moves at its maximum velocity v_{max} which is too fast for examination tasks, e.g., to gather desired information or interact with the targets. Thus, the robot moves at a slower velocity v_{det} for precise examination once a target is found.

To formulate the task as searching and examining a finite number of locations that cover the entire task space, we tile E by a grid map with K identical grids. To make use of robot's sensing range, K is chosen in a way that a robot can only detect all targets in one grid when it is placed at the center of it. With a slight abuse of notations, k denotes both a grid and the index of it, and K denotes both the entire grid set and the number of total grids. By choosing an *action* $k \in K$, a robot travels to the center of grid k to conduct examination.

3.1. Bernoulli Thompson Sampling

Multi-armed bandits (MAB) [31] are a well-known set of problems where the exploration-exploitation tension plays an importance role for a player (gambler) to select arms that maximize its accumulated reward. In recent work [32, 33], Thompson sampling (TS) [34] is shown to be the most effective and easily implementable stochastic approach to the MAB problems. In Bernoulli TS, a player receives only a binary outcome of either win or loss, i.e., $win \in \{0, 1\}$, from each of the K resources $k \in \{1, 2, ..., K\}$ with a fixed and unknown probability $\theta_k \in [0, 1]$. A reward-maximizing action is sequentially recommended for the next resource to sample from at each discrete time step $t \in T$ using Algorithm 1, where $\textbf{beta}(\alpha, \beta)$ denotes the beta distribution with parameters α and β. The algorithm takes the inputs of the number of resources K and initial parameters for the beta distributions for each resource, i.e., $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]$ and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_K]$. At each discrete time step t, the reward posterior of each resource is sampled and the resource with the highest reward is selected (lines 2-5). The player takes the recommended action, receives the corresponding outcome, and uses a reward r to update the distribution parameters for the selected action (lines 6-7). The update equations for the parameters of the beta distribution are

Algorithm 1 BernTS($K, \boldsymbol{\alpha}, \boldsymbol{\beta}$)

1: **for** $t = 1, 2, \ldots$ **do**
2: **for** $k = 1, \ldots, K$ **do**
3: Sample $\hat{\theta}_k \sim \textbf{beta}(\alpha_k, \beta_k)$
4: **end for**
5: $k^* \leftarrow \arg\max_k \hat{\theta}_k$ ▷ Recommend action
6: Apply action s_{k^*} and observe win_{k^*}
7: Update $\alpha_{k^*}, \beta_{k^*}$ using Equation 1
8: **end for**

$$\begin{cases} \alpha_k \leftarrow \alpha_k + r_{const}, win_k = 1 \\ \beta_k \leftarrow \beta_k + r_{const}, win_k = 0. \end{cases} \tag{1}$$

The tuning reward r_{const} balances between exploration and exploitation, with a higher r_{const} increases the weight of exploitation and vice versa. The beta functions then encode the posterior of getting a reward from each resource, allowing each resource to be sampled with a time-varying probability depending on current knowledge of the posterior distributions and resulting in an exploration behavior. As a growing number of observations are accumulated, the posterior distributions tend to congregate at true expected rewards and the recommended actions approach optimal, bringing about an exploitation behavior.

3.2. Assumptions

We make several assumptions for algorithm implementation and search efficiency guarantee.

Assumption 1 (Environment). *We assume that E is known, which is moderate since usually it is not trivial to obtain maps in advanced through SLAM, GPS, etc. We assume that E is significantly greater AoI, i.e., AoI occupies a small portion of E. This assumption defines a* search *problem instead of* exploration*, in which case a large portion of the search space must be investigated. We assume E to be a convex open space for simplicity, as motion planning in non-convex space is beyond the scope of discussion.*

Assumption 2 (Robot). *Robots must be ability to get both q and all p online during search through external positioning systems or state estimation techniques. We assume that v_{max} is significantly greater than v_{det}, which is often the case in practice. We also assume $E \gg fov$ to set a strong constraint of limited sensing range.*

Assumption 3 (Target). *We assume that targets are static. We assume that all targets must be found, defining a* capture *problem in target search catalog [35]. It is*

also assumed that targets are likely to, though not necessary to always appear in clusters. However, the target distribution is completely random and unpredictable and explicit prior knowledge cannot be accessed.

4. Greedy Algorithm

The idea of the brute force greedy algorithm is to randomly explore the task space and exploit the surrounding of a target thoroughly once it is found. For each robot $r_i \in R$, the search and examination algorithm is outlined in Algorithm 2, taking inputs of r_i's initial position q_0 and ID i, grid set K. A set V, initially empty, is also utilized to store all visited grids and is updated and shared across the team through neighborhoods. An empty action list is initialized prior to the task containing the next grids to examine sequentially. When a robot finds targets at its current examined grids, it pushes all un-examined grids from the 8 surrounding grids to its action list by checking V, shown in Lines 4-5. In the case that all surrounding grids are examined, the robot pushes grids in the action lists of its neighbored robots in order from nearest to farthest without duplication so as to assist other robots sequentially, outlined in Lines 6-9. After examining the current grid, the robot either selects its next goal as the first grid in its non-empty action list, or draws an un-examined grid randomly as its next goal if all surroundings grids are examined and no assistance is needed by neighboring robots, as outlined in Lines 10-16.

The novel greedy algorithm allows robots to thoroughly exploit a target cluster deterministically before exploring other clusters, assuming that targets are located next to each other, i.e., there is only one cluster in the environment. While the assumption does not hold, i.e, there are multiple clusters in the environment, it can be costly for robots to find all clusters as they sample and examine the environment without any prior knowledge of target distribution. As a comparison to the brute force solution, in the next section, we present a probabilistic TS-based algorithm that aims to achieve a balance between searching efficiently for highly clustered targets and slightly clustered ones.

Algorithm 2 GreedySearch(q_0, i, K, V)

1: Initialize an action list $S_i \leftarrow \emptyset$
2: Initialize $q_i \leftarrow q_0$
3: **while** true **do**
4: **if** targets are found at q_i **then**
5: Push un-examined adjacent grids to S_i
6: **if** $S_i = \emptyset$ **then**
7: Push S_j to S_i for all $j \in \mathcal{N}_j$ without duplication in order from nearest to farthest
8: **end if**
9: **end if**
10: **if** $S_i = \emptyset$ **then**
11: Draw $k \in K$ randomly s.t. $g_k \notin V$
12: $goal_i \longleftarrow g_k$
13: **else**
14: $goal_i \longleftarrow S_i(1)$
15: $S_i \longleftarrow S_i \setminus S_i(1)$
16: **end if**
17: $V \leftarrow V \cup goal_i$
18: Broadcast S_i and V to neighbors
19: Travel to and examine $goal_i$
20: **end while**

5. Thompson Sampling Algorithm

5.1. Bernoulli Bandit Modeling

The outcome of an action k is denoted as a binary value based on the detection Z received at that location,

$$win_k = \begin{cases} 0, & Z = \emptyset \\ 1, & \text{else}, \end{cases} \tag{2}$$

i.e., the outcome is 0 if a robot detects nothing in that grid and 1 otherwise. For each grid $k \in K$, a beta function $\mathbf{beta}(\alpha_k, \beta_k)$ is updated by a reward r_k obtained through a reward policy introduced in Section 5.3. Initially, we set $\alpha_k = \beta_k = 1$, meaning that targets are considered with identical probability to appear at any locations in E. Note that outcomes are not immediately received after selecting an action. Instead, there is a time delay between selecting an action and receiving an outcome due to the travel and detecting time it takes.

5.2. Goal Selection

On starting the search task, a robot must select the next grid to search at each time step. Our goal selection takes both current belief of true target distribution and the cost of time to travel to the next goal into consideration to optimize search efficiency. On the one hand, a robot is always rewarded by choosing a grid with maximum probability of finding a target and move to it. On the other hand, a reward obtained by reducing travel time becomes great when targets are repeatedly found in current exploited sub-region. In this way, the robot prioritizes searching surrounding areas and leaves other areas for future exploitation, even if the posterior of surrounding target density is not maximum, so as to shorten the total travel time during search. On the contrary, the robot receives less reward from saving travel time and tends to rely on its best guess of target distribution if it

repeatedly finds no targets, suffering from possible long travel time for sake of finding the next sub-region to exploit.

The robot is assumed to have repeatedly found targets in the last $c_f \in \mathbb{N}$ selected grids, or have repeatedly missed targets in the last $c_m \in \mathbb{N}$ selected grids. Once a goal is decided, we define the "travel-reducing" reward obtained by reducing travel time from its current location q to any $p \in E$ by a non-negative function:

$$\mathcal{H}(q,p) = \begin{cases} (1-norm)\cdot c_f, & c_f > 0 \\ (1-norm)/c_m, & \text{else,} \end{cases} \tag{3}$$

$$norm = \frac{\|q-p\|}{d_{max}}, \tag{4}$$

where d_{max} is the distance between two furthest points in E, and $\|\cdot\|$ denotes the Euclidean norm. Note that the Euclidean distance between goal location and robot location can be replaced with length of trajectories if E is not a convex open space and motion planning algorithms are applied. We normalize the distance between q and p by E's dimension in *norm*. The robot can always be better rewarded by selecting a closer grid to detect, as a smaller *norm* always leads to a greater $\mathcal{H}$. Then we select an action k^* such that

$$\mathcal{U} = c\cdot\hat{\theta}_k + (1-c)\cdot\mathcal{H}(q,p_k), \tag{5}$$

$$k^* \leftarrow \arg\max_k \mathcal{U}, \tag{6}$$

where $\hat{\theta}_k \sim \textbf{beta}(\alpha_k,\beta_k)$, and $c \in [0,1]$ is a tuning weight. If the robot repeatedly gets positive outcomes, it receives increasing reward from deciding to reduce travel distances, which encourages it to exploit nearby locations despite the estimated target distribution. Conversely, if the robot constantly fails to find targets, the reward from reducing travel time decreases and the reward from selecting grids with higher estimated target density dominants.

Our goal selection process is outlined in Algorithm 3, which takes the inputs of K, current $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, as well as a set S containing all grids not being searched at current time step. Different from the regular MAB problem where a player may select an arm repeatedly, a grid is never searched twice in our target search problem, meaning that the same action cannot be selected twice. A sample θ_k is drawn from the beta function of each grid $k \in S$, after which a grid k^* is selected and the temporary goal location g is set to k^*'s location p_k, outlined in lines 1-7. Then the algorithm outputs g and the updated S, shown in lines 8-9.

Algorithm 3 GoalSelection($K, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{S}$)

1: **for** $k = 1,\ldots,K$ **do**
2: **if** $k \in \boldsymbol{S}$ **then**
3: Sample $\hat{\theta}_k \sim \textbf{beta}(\alpha_k,\beta_k)$
4: **end if**
5: **end for**
6: Select k^* using Equation 6
7: $g \leftarrow p_k$
8: $\boldsymbol{S} \leftarrow \boldsymbol{S} \setminus k^*$
9: Return $g, \boldsymbol{S}$

5.3. Recursive Target Search

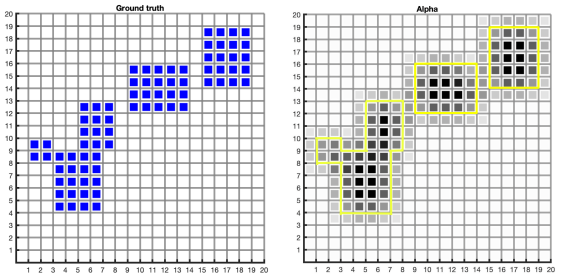
The target search algorithm, outlined in Algorithm 4, allows a robot to search targets online recursively overtime. The algorithm begins with initializing a grid map and beta functions for each grid, outlined in lines 1-2. The temporary goal g is set to the robot's initial position q_0, and the set of all un-visited grids S is initialized with K, as shown in line 3. Lines 4-15 demonstrate the main search process, including a posterior update step and a posterior filtering step, and the goal selection process. Robot terminates when desired task is completed, such as having searched all grids or running out of time.

5.3.1. Posterior Update

To recursively select optimized goals online, a robot must update the target posterior after getting an outcome at g. In standard MAB problems, the posterior of a resource is updated for a player to decide how frequently that resource is to be selected to play. However, in searching for static targets, a same grid will not be explored repeatedly. To take advantage of search outcomes, posteriors of some other grids must also be updated besides the sampled grid, based on the assumption that they are somewhat correlated. Thus, a K-dimensional reward set $\boldsymbol{r} = \{r_1, r_2, \ldots, r_K\}$ is created and initialized with zeros at each time step, after which we choose to set not only k's reward, but also the rewards of k's eight neighboring grids $\mathcal{N}(r_k)$ to r_{const}, as outlined in lines 5-7 in Algorithm 4. The reward set $\boldsymbol{r}$ is then utilized to update the beta functions following the reward policy outlined in lines 8-12, an explicit form of Equation 1 where win_k is determined by Equation 2. The reward policy augments αs of a grid and its neighbors with r_{const} if the outcome is positive, and updates βs similarly for a zero outcome. We demonstrate in Section 6 that such strategy outperforms the baseline sweeping algorithm when targets are distributed in clusters since the neighborhood is believed to have similar target density to a centered grid, attracting robots to search around founded targets.

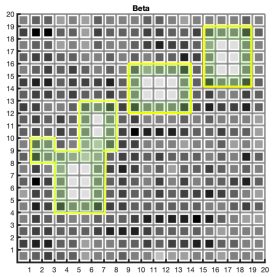
Algorithm 4 TargetSearch(q_0, K)

1: Draw K grids over E
2: Initialize $\alpha(1,1), \beta(1,1)$ for all $\alpha_k \in \boldsymbol{\alpha}, \beta_k \in \boldsymbol{\beta}$
3: Initialize $g \leftarrow q_0, \boldsymbol{S} \leftarrow 1, 2, ..., K$
4: **while** true **do**
5: $\quad \boldsymbol{r} \leftarrow \boldsymbol{0}$
6: $\quad$ Move to g and search
7: $\quad$ Set r_k and all members in $\mathcal{N}(r_k)$ to r_{const}
8: $\quad$ **if** $win_k = 1$ **then**
9: $\quad\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \boldsymbol{r}$
10: $\quad$ **else**
11: $\quad\quad \boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \boldsymbol{r}$
12: $\quad$ **end if**
13: $\quad$ Implement posterior filtering and update $\boldsymbol{\beta}$
14: $\quad$ GoalSelection($K, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{S}$)
15: **end while**

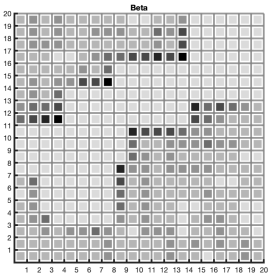


(a) True Target Distribution

(b) $\boldsymbol{\alpha}$ Map



(c) $\boldsymbol{\beta}$ Map Before Filtering



(d) $\boldsymbol{\beta}$ Map After Filtering

Figure 2: Figures show an example of the posterior filtering process in a 20×20 grid map. Figure 2a shows the ground truth of target distribution, with targets are located in blue grids. Figure 2b shows the α value of each grid, with a darker grid indicating a higher α in that grid. Yellow boundaries mark the true locations of target clutters. Figure 2c and Figure 2d show the β value of each grid before and after filtering, respectively, with a darker grid indicating a higher β in that grid. In Figure 2c, yellow boundaries mark the true locations of target clutters and green translucent areas are marginal grids of clusters.

5.3.2. Posterior Filtering

Though a robot benefits from exploring goals close to known targets in target clusters, it may suffer from inaccurate estimation of target density at cluster edges. When the robot detect nothing right next to the edges of target clusters, it tends to not move to these marginal areas, where in fact there are target distributed. Such tendency aggravates as searched space accumulates, resulting in slowdown in finding targets located at cluster edges. An example is shown in Figure 2. After searching for targets distributed as Figure 2a for a period of time in a 20×20 grids squared search space, while grids with targets inside are with higher α values (Figure 2b), not all of them are with lower β values (Figure 2c). The marginal areas of target clusters, marked by translucent green in Figure 2c, have relative high values, resisting being selected to search by the robot.

Therefore, before selecting goals using Algorithm 3, we increment the posterior update with a simple filtering process to improve robot's performance in searching cluster edges, as outlined in line 13 in Algorithm 4. We extract a set $A \subset K$ of all grid k in which $\alpha_k > 1$. In other words, A is the set of all grids that targets are found inside or close to them so far. Then we reset $\{\forall \beta_k = 1 | k \in A\}$, indicating that they will not be disregarded even the robot has detected nothing around it. The $\boldsymbol{\beta}$ map after filtering Figure 2c is shown as Figure 2d.

6. Simulation Results

We conduct a series of `MATLAB` simulations to validate the performance of our algorithms through batches of tests. We show quantitative comparisons of our algorithms with the baseline method. Additionally, we present an ablation study to demonstrate the efficacy of both posterior update and posterior filtering for the TS-based method. For simplicity, in our simulations, robot displacement is with first order dynamics and rotation does not take up time. We choose $r_{const} = 10$ and $c = 0.5$. Search space is square with $1\,\text{m} \times 1\,\text{m}$ grid size based on robot's field of view.

6.1. Quantitative Comparisons

To test whether a team of robot can efficiently examine an area and find targets, eight holonomic robots equipped with an isotropic sensor are tasked to search a $40\,\text{m} \times 40\,\text{m}$ squarespace with striped obstacles. The robot moves at $v_{det} = 0.03\,\text{m/s}$ in order to effectively examine the grids. We test four different target aggregation levels from 1 to 4 depending on the ratio of sensor

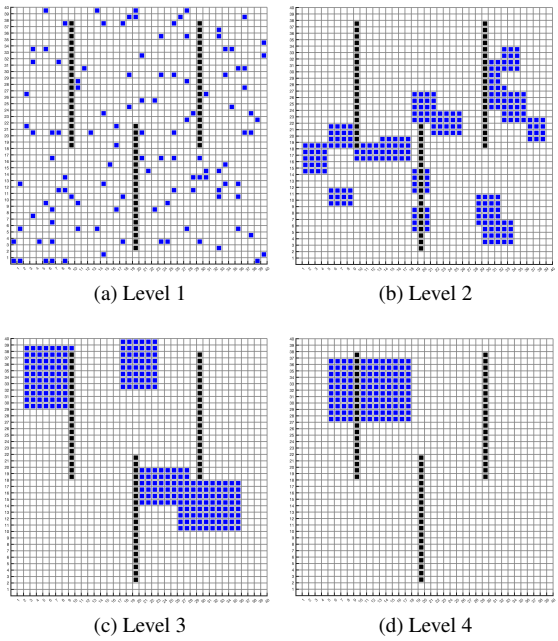
field of view to the average area of clusters, where level 1 is equivalent to uniform distribution. An example is shown as Figures 3. For each aggregation level, target distribution is generated randomly in each trial via arbitrarily placing identical numbers and sizes of clusters inside the search space, with possible overlaps.

Our greedy and TS-base algorithms are compared with a baseline sweeping algorithm, which divides the environment into eight identical rectangular areas for the robots and drives each of them to sweep its assigned area in a zigzag pattern, resulting in the most efficient coverage paths for examining all grids in the environment. We first run four batches of tests of four aggregation levels with $v_{max} = 0.3\,\mathrm{m/s}$ respectively, each with 100 trials. For each trial, robots depart from the lower-left corner of the environment and the task is considered finished when all targets are found and examined. As shown in Figures 4a-4d, sweeping algorithm yields constant performance and completes the task in around or slightly less than 3000 s for all target distributions and outperforms when targets are uniformly distributed, since robots must examine most of the grids sequentially to find all targets. On the contrary, by adopting a deterministic exploitation strategy, the greedy algorithm shows subversively different performances as aggregation level changes and only outperforms when all targets are highly aggregated. However, TS-based algorithm shows significant improvement in efficiency to the sweeping algorithm for targets from slightly to highly clustered due to its ability to trade-off between exploration and exploitation by taking advantage of past examination outcomes.

Instead of examining grids sequentially through a shortest coverage path, both our algorithms actively select the next grid to investigate based on the past outcomes, yet longer trajectories must be traversed. Therefore, the efficiency of our algorithms in searching and examining clustered targets exacerbates when robot's traveling across a grid costs much more time than examination of it. Figures 4c, 4e, and 4f demonstrate this feature and indicate that the more v_{max} greater than v_{det}, the more significant of both greedy and TS-based algorithms.

6.2. Ablation Study for TS-Based Method

We further validate the efficacy of both posterior update and posterior filtering using TS-based method through a series of tests. In the following tests, a robot is tasked to search a $50\,\mathrm{m}\times50\,\mathrm{m}$ square open space with an isotropic sensor. The robot moves at $v_{max} = 1\,\mathrm{m/s}$ and $v_{det} = 0.2\,\mathrm{m/s}$. To randomize target distribution, we select n target clusters from 7 different sizes and shapes shown in Figure 5 and randomly place them inside each tested search space, with possible overlaps.



(a) Level 1 (b) Level 2

(c) Level 3 (d) Level 4

Figure 3: Figures show four target aggregation levels in a $40\,\mathrm{m}\times40\,\mathrm{m}$ tested environment from level 1 to 4, respectively. Black grids are occupied by obstacles which robots are not able to cross. Targets are located in blue grids.

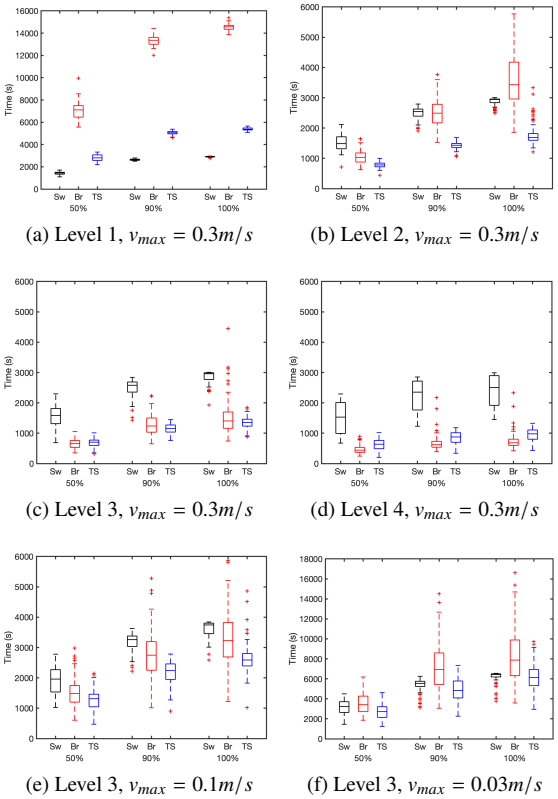
6.2.1. Wasserstein-1 Distance

At each time step, we use Wasserstain-1 distance (W1D) to quantify the error between the distribution of searched locations and the distribution of true target locations. Wasserstein-1 distance is also named the earth-mover distance due to its connection to the optimal transport problem [36]. We denote $\mathcal{P}(M)$ the space of probability measures defined on a metric set M. For two probability measures $\mu, \nu \in \mathcal{P}(M)$, we define the W1D:

$$\mathcal{W}(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \int_{M\times M} \|x-y\| \mathrm{d}\gamma(x,y), \qquad (7)$$

where $\Gamma(\mu,\nu)$ is the set of all joint distributions $\gamma(x,y)$ whose marginals are μ and ν, respectively. W1D calculates the minimum distance for an earth mover to transport one pile of a unit amount of earth into another, and $\gamma(x,y)$ quantifies the "mass" of earth to be transported from x to y in order to transform the distributions μ into ν.

W1D between the searched set and the true target set reflects robot's effectiveness in optimizing its goal selection. A lower W1D indicates the fact that the robot

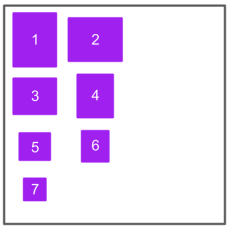


(a) Level 1, $v_{max} = 0.3m/s$ (b) Level 2, $v_{max} = 0.3m/s$

(c) Level 3, $v_{max} = 0.3m/s$ (d) Level 4, $v_{max} = 0.3m/s$

(e) Level 3, $v_{max} = 0.1m/s$ (f) Level 3, $v_{max} = 0.03m/s$

Figure 4: Boxplots show batches of simulation results of utilizing sweeping algorithm (Sw), brute force greedy algorithm (Br), and TS-based algorithm (TS), respectively, with different target aggregation levels and robot traveling speeds v_{max}. We conduct 100 trials for each combination of variables. Figures 4a, 4b, 4c, and 4d plots the time consumed to find and examine 50%, 90%, and 100% targets distributed at different aggregation levels when v_{max} equals to 0.3 m/s, respectively. Figures 4e and 4f plots the time consumed to find and examine all targets distributed at aggregation level 3 when v_{max} equals to 0.1 m/s and 0.03 m/s, respectively.



Cluster	Width	Height
1	1/4	1/3
2	1/3	1/4
3	1/5	1/4
4	1/4	1/5
5	1/7	1/6
6	1/6	1/7
7	1/10	1/10

Figure 5: Figure shows the 7 different sizes and shapes of target clusters we choose from to randomize the simulated target distributions. Blue numbered rectangles are 7 clusters, and their sizes and shapes are compared to the search space drawn in black square. The legend gives the width and height of each cluster, assuming the search space is with unit side length. Widths and heights are rounded to the closed grid numbers in grid map implementation.

assigns more efforts in areas where target density is higher, instead of searching for areas with low target density wastefully, despite the number of targets being found. Additionally, a more rapidly descending W1D illustrates a stronger ability for the robot to optimizing its goal selection.

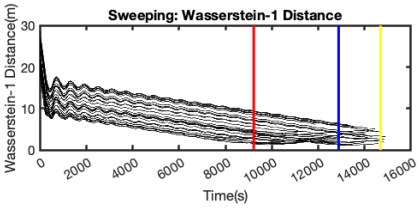
6.2.2. Results

We run 30 trials, for each of which we generate a target distribution using cluster 1-6, one for each cluster. Three methods are compared, namely the sweeping algorithm, the TS method without posterior filtering, and the TS method with posterior filtering, with the same 30 target distributions generated randomly. In Figure 6, we plot the W1Ds for 30 trial using each method along with the median search time used in 30 trials to detect 50%, 90%, and 100% of AoI. The results indicate that TS-based methods outperforms the sweeping strategy, regardless of whether the posterior filtering is applied or not, since TS-based methods lead to significant shorter time to detect AoI. In Figure 6a, the uniformly descending wavy curves result from the zigzag behavior by running the sweeping algorithm. We also see that W1Ds drop much faster using TS-based methods, indicating their ability to optimize goal selections online.

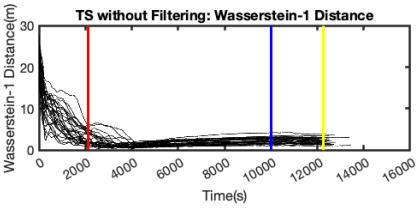
Moreover, the results show a significant reduce in search time by applying the posterior filtering step in finding the last part of AoI, while the speed for finding the first part for it is a bit lower. The reason behind is that the robot cannot locate the edges of clusters precisely at the beginning without enough sampling, which slightly prevents it from exploiting the true clusters by filtering β. However, as samples accumulates, this filtering greatly help the robot to find the edges in a short period of time. By contrast, W1D slowly goes up for a long time after a fast descent in Figure 6b, indicating that the robot makes an large amount of useless effort in searching the last part of the targets without the filtering step.

6.3. Clustering Levels

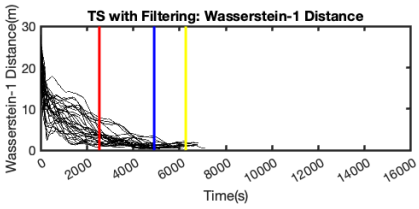
In order to find the adaptability of our TS-based method in environments with target clustered to varying degrees, we continue to run trials using target distributions generated from three sets of target clusters, from highly clustered to somewhat clustered. We choose the cluster sets in a way that for each of the target distributions, $\frac{AoI}{E}$ is similar. We run 30 trials for each set of clusters and record the median search time of the 30 trials to detect 50%, 90%, and 100% of AoI. The results along with the sets of chosen clusters are shown in Table 1. It is shown that our TS-based method performs



(a) W1D of Sweeping



(b) W1D of TS Without Filtering



(c) W1D of TS with Filtering

Figure 6: Figures show results of batch of trials. Three methods are compared, namely the sweeping algorithm (Figure 6a), the TS method without posterior filtering (Figure 6b), and the TS method with posterior filtering (Figure 6c). Black curves show 30 W1Ds over time for each method. Red, blue and yellow lines in each subplot show median time used of 30 trials for finishing detecting 50%, 90%, and 100% of *AoI* respectively.

better with higher target clustering levels, which is consistent with our inference. However, of all trials, our method is found to outperform the sweeping algorithm in searching speed, by roughly compared with Figure 6.

7. Conclusions

This paper studied the area examination and target search problem to allow a team of robot to efficiently find targets that are likely to distribute in random clusters. Robots have no explicit knowledge of target distribution, and the search space is much greater than robot sensing range. Comparing with the greedy algorithm, the TS-based algorithm relies on recursively sampling the next location to search, taking into account both the target posterior obtained from previous search and the weighted penalization of travel time to that location at each time instance. As a result, robots can learn from the environment and are capable of taking advantage of past exploration to optimize goal selection. Our simulations demonstrate the efficacy of our examination and search algorithms explicitly and validate the efficiency of our methods compared to the sweeping algorithm. Our future work will extend the method to 3D surface implementation, and to distributed multi-robot systems. We are seeking to develop an efficient mobile information gathering network that is able to explore coral reef under the sea, find damages from building facades, etc.

Table 1: Searching Time(s) for Different Clustering Levels

Distribution \ Percentage	50%	90%	100%
4×Cluster 1	1989	3812	5494
10×Cluster 6	1928	4739	5930
20×Cluster 7	3858	7261	7978

Funding

This work was supported by research initiation grant 184080H201B60 from Nanjing Normal University.

References

[1] E. Bayraktarov, P. J. Stewart-Sinclair, S. Brisbane, L. Boström-Einarsson, M. I. Saunders, C. E. Lovelock, H. P. Possingham, P. J. Mumby, K. A. Wilson, Motivations, success, and cost of coral reef restoration, Restoration Ecology 27 (5) (2019) 981–991.

[2] T. Manderson, J. Li, N. Dudek, D. Meger, G. Dudek, Robotic coral reef health assessment using automated image analysis, Journal of Field Robotics 34 (1) (2017) 170–187.

[3] S. Mou, D. Tsai, M. Dunbabin, Reconfigurable robots for scaling reef restoration, arXiv preprint arXiv:2205.04612 (2022).

[4] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous systems 61 (12) (2013) 1258–1276.

[5] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, J. Nieto, Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem, in: Field and service robotics, Springer, 2021, pp. 277–290.

[6] S. Bochkarev, S. L. Smith, On minimizing turns in robot coverage path planning, in: 2016 IEEE international conference on automation science and engineering (CASE), IEEE, 2016, pp. 1237–1242.

[7] T. M. Cabreira, C. Di Franco, P. R. Ferreira, G. C. Buttazzo, Energy-aware spiral coverage path planning for uav photogrammetric applications, IEEE Robotics and automation letters 3 (4) (2018) 3662–3668.

[8] C. Wu, C. Dai, X. Gong, Y.-J. Liu, J. Wang, X. D. Gu, C. C. Wang, Energy-efficient coverage path planning for general terrain surfaces, IEEE Robotics and Automation Letters 4 (3) (2019) 2584–2591.

[9] W. Jing, D. Deng, Z. Xiao, Y. Liu, K. Shimada, Coverage path planning using path primitive sampling and primitive coverage graph for visual inspection, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2019, pp. 1472–1479.

[10] B. Englot, F. Hover, Sampling-based coverage path planning for inspection of complex structures, in: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 22, 2012, pp. 29–37.

[11] J. Tang, C. Sun, X. Zhang, Mstc*: Multi-robot coverage path planning under physical constrain, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 2518–2524.

[12] A. Sipahioglu, G. Kirlik, O. Parlaktuna, A. Yazici, Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach, Robotics and Autonomous Systems 58 (5) (2010) 529–538.

[13] L. Collins, P. Ghassemi, E. T. Esfahani, D. Doermann, K. Dantu, S. Chowdhury, Scalable coverage path planning of multi-robot teams for monitoring non-convex areas, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 7393–7399.

[14] N. Karapetyan, K. Benson, C. McKinney, P. Taslakian, I. Rekleitis, Efficient multi-robot coverage of a known environment, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 1846–1852.

[15] L. Heng, A. Gotovos, A. Krause, M. Pollefeys, Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 1071–1078.

[16] B. Yamauchi, A. Schultz, W. Adams, Mobile robot exploration and map-building with continuous localization, in: Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Vol. 4, IEEE, 1998, pp. 3715–3720.

[17] F. Niroui, K. Zhang, Z. Kashino, G. Nejat, Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments, IEEE Robotics and Automation Letters 4 (2) (2019) 610–617.

[18] K. M. Wurm, C. Stachniss, W. Burgard, Coordinated multi-robot exploration using a segmentation of the environment, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2008, pp. 1160–1165.

[19] W. Burgard, M. Moors, C. Stachniss, F. E. Schneider, Coordinated multi-robot exploration, IEEE Transactions on robotics 21 (3) (2005) 376–386.

[20] W. Sheng, Q. Yang, J. Tan, N. Xi, Distributed multi-robot coordination in area exploration, Robotics and autonomous systems 54 (12) (2006) 945–955.

[21] B. Lindqvist, A.-A. Agha-Mohammadi, G. Nikolakopoulos, Exploration-rrt: A multi-objective path planning and exploration framework for unknown and unstructured environments, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 3429–3435.

[22] J. Vallvé, J. Andrade-Cetto, Potential information fields for mobile robot exploration, Robotics and Autonomous Systems 69 (2015) 68–79.

[23] H. Carrillo, P. Dames, V. Kumar, J. A. Castellanos, Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy, in: 2015 IEEE international conference on robotics and automation (ICRA), IEEE, 2015, pp. 487–494.

[24] D. K. Sutantyo, S. Kernbach, P. Levi, V. A. Nepomnyashchikh, Multi-robot searching algorithm using lévy flight and artificial potential field, in: 2010 IEEE Safety Security and Rescue Robotics, IEEE, 2010, pp. 1–6.

[25] A. Baranzadeh, A. V. Savkin, A distributed control algorithm for area search by a multi-robot team, Robotica 35 (6) (2017) 1452–1472.

[26] P. Lanillos, S. K. Gan, E. Besada-Portas, G. Pajares, S. Sukkarieh, Multi-uav target search using decentralized gradient-based negotiation with expected observation, Information Sciences 282 (2014) 92–110.

[27] J. Pugh, A. Martinoli, Inspiring and modeling multi-robot search with particle swarm optimization, in: 2007 IEEE swarm intelligence symposium, IEEE, 2007, pp. 332–339.

[28] H. Xiao, R. Cui, D. Xu, A sampling-based bayesian approach for cooperative multiagent online search with resource constraints, IEEE Transactions on Cybernetics 48 (6) (2017) 1773–1785.

[29] M. S. Couceiro, R. P. Rocha, N. M. Ferreira, A novel multi-robot exploration approach based on particle swarm optimization algorithms, in: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, IEEE, 2011, pp. 327–332.

[30] J. Chen, P. Dames, Distributed multi-target tracking for heterogeneous mobile sensing networks with limited field of views, in: 2021 ieee international conference on robotics and automation (icra), IEEE, 2021, pp. 9058–9064.

[31] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine learning 47 (2) (2002) 235–256.

[32] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen, A tutorial on Thompson sampling, arXiv preprint arXiv:1707.02038 (2017).

[33] O. Chapelle, L. Li, An empirical evaluation of Thompson sampling, Advances in neural information processing systems 24 (2011) 2249–2257.

[34] W. R. Thompson, On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, Biometrika 25 (3/4) (1933) 285–294.

[35] C. Robin, S. Lacroix, Multi-robot target detection and tracking: taxonomy and survey, Autonomous Robots 40 (4) (2016) 729–760.

[36] F. Santambrogio, Optimal transport for applied mathematicians, Birkäuser, NY 55 (58-63) (2015) 94.