Decentralised Mission Monitoring with Spatiotemporal Optimal Stopping

Graeme Best[1,2], Shoudong Huang[3] and Robert Fitch[3,2]

***Abstract*— We consider a multi-robot variant of the mission monitoring problem. This problem arises in tasks where a robot observes the progress of another robot that is stochastically following a known trajectory, among other applications. We formulate and solve a variant where multiple *tracker* robots must monitor a single *target* robot, which is important because it enables the use of multi-robot systems to improve task performance in practice, such as in marine robotics missions. Our algorithm coordinates the behaviour of the trackers by computing optimal single-robot paths given a probabilistic representation of the other robots' paths. We employ a decentralised scheme that optimises over probability distributions of plans and has useful analytical properties. The planned trajectories collectively maximise the probability of observing the target throughout the mission with respect to probabilistic motion and observation models. We report simulation results for up to 8 robots that support our analysis and indicate that our algorithm is a feasible solution for improving the performance of mission monitoring systems.**

I. Introduction

Mission monitoring is a problem in which one mobile robot must observe the progress of another robot that is performing a given task. This problem was first introduced in the context of monitoring an underwater robot with a surface vessel [1], but is applicable in general to a wide variety of task scenarios and types of robots, including environmental monitoring, agricultural robotics, and aerial vehicles [2].

One defining characteristic of mission monitoring is that the *tracker* robot must stop for a period of time in order to observe the *target* robot, whose intended trajectory is known in advance but subject to stochastic disturbances. The physical motivation for this property is that the tracker may need to deploy communication equipment that is most efficient while stationary. The algorithmic problem, then, is how to choose a sequence of stopping locations and durations such that the probability of successfully observing the target is maximised. This problem is important because it is an essential part of employing outdoor robots for certain real-world tasks, such as various underwater missions [3], that depend on timely transmission of sensor observations or system faults. It is also interesting in broader contexts because it applies to

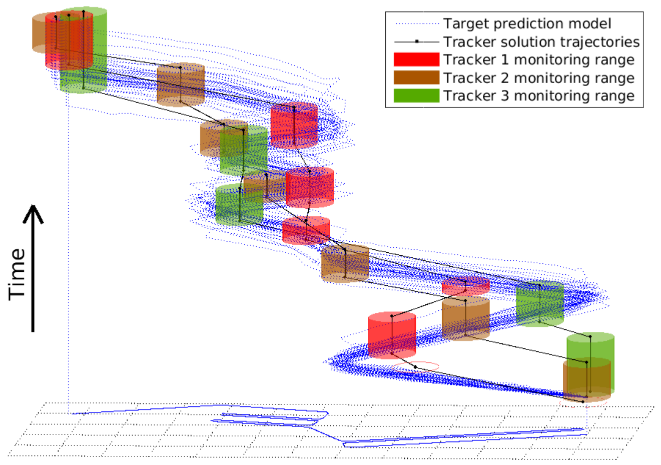

Target prediction model
Tracker solution trajectories
Tracker 1 monitoring range
Tracker 2 monitoring range
Tracker 3 monitoring range
Time

Fig. 1. **The multi-tracker mission monitoring problem.** A probabilistic prediction model for a robot trajectory (30 min AUV mission) is shown as blue sample trajectories. A plan for tracker team (3 surface vessels) is shown in black. Cylinders represent probabilistic monitoring regions at stopping locations. The objective can be interpreted geometrically as maximising the expected overlap between the cylinders and the prediction model.

systems that must stop periodically to conserve energy [4], to provide imagery taken from a stationary viewpoint [5], and for acoustically covert surveillance [6].

Although optimal algorithms exist for the case of a single tracker and target [2], the multi-robot case is not well studied. Various multi-robot problem settings are possible, but the case of multiple trackers observing a single target is of immediate practical value. Optimal single-tracker algorithms guarantee the best solution given a stochastic target trajectory, but do not necessarily guarantee any absolute level of quality. The target trajectory or communication channel may be subject to severe uncertainty that limits the probability of success of any single-tracker solution, or the tracker may be relatively slow-moving. Utilising multiple trackers provides a pathway for improvement by enabling the observation of multiple disparate possible target positions simultaneously.

One challenge in considering the multi-tracker case is that single-tracker algorithms do not extend naturally. It is not useful for trackers to plan independently, because it is likely that all trackers would choose to make the same, rather than complementary, observations. Instead, each tracker must compute its actions *jointly* with the actions of the others. Hence, it is necessary to solve the resulting coordination problem. Ideally, this problem should be solved in a decentralised and asynchronous manner to distribute the computational effort, to avoid having a single point of failure, and to be robust to unreliable communication links.

*This work is supported in part by the Australian Centre for Field Robotics, the Centre for Autonomous Systems at the University of Technology Sydney, and the NSW government.

[1]G. Best is with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR, USA. bestg@oregonstate.edu

[2]G. Best and R. Fitch are with the Australian Centre for Field Robotics (ACFR), The University of Sydney, Sydney, Australia.

[3]S. Huang and R. Fitch are with the Centre for Autonomous Systems (CAS), University of Technology Sydney, Sydney, Australia. {shoudong.huang,rfitch}@uts.edu.au

In this paper, we formulate the multi-tracker variant of mission monitoring and propose a decentralised solution algorithm. Our solution is loosely based on our optimal single-robot planner in [2], which comprises of finding the maximum-weight path through a spatiotemporal graph. For the multi-tracker scenario, the algorithm must be extended to consider the trajectories of the other agents. However, this alone is not enough to ensure successful coordination due to the cyclic dependencies between agents. We overcome this challenge by defining plans as probability distributions over trajectories that are optimised in a decentralised manner; this formulation reduces the likelihood of the algorithm getting stuck in a cycle of suboptimal solutions [7]. While it is difficult to obtain global optimality guarantees for decentralised algorithms, our analysis shows optimality with respect to the *current* plans of the other robots, and we have the proposition that the plans converge towards the probability distribution that minimises the KL-divergence to the optimal *joint* distribution. The algorithm also has the useful properties of being any-time, polynomial runtime per iteration, and small communication bandwidth usage.

Our algorithm produces a trajectory for each tracker that is optimised with respect to probabilistic communication and target trajectory prediction models. Example trajectories for a team of trackers is illustrated in Fig. 1. The trackers perform a dynamic weighted coverage responsive to the probabilistic models and dynamics constraints, which is distinctly different to simple uninformed spatial coverage. We present simulated experiments with realistic trajectory prediction [2] and communication [8] models for an autonomous underwater vehicle (AUV) and a team of monitoring surface vessels. The results demonstrate that the trackers must coordinate to adequately solve this problem, there is significant benefit of using a probabilistic plan representation, and our algorithm outperforms a generic planner. Overall, we show the approach is viable for use in multi-tracker mission monitoring.

Contributions: This paper presents the first formulation and solution for decentralised multi-tracker mission monitoring. While we borrow ideas from our previous work, namely single-tracker mission monitoring [2] and Dec-MCTS [7], this paper contributes more than simply a combination of these two ideas. Relative to [2], this paper contributes a multi-tracker formulation of mission monitoring, a modified single-tracker algorithm for this setting, a decentralised generalisation of the single-tracker algorithm, and new analytical and empirical results. Relative to [7], this paper contributes a new decentralised algorithm designed for mission monitoring, has stronger analytical properties, and is empirically demonstrated to outperform Dec-MCTS.

II. Related Work

The mission monitoring problem was first posed in [1] along with preliminary algorithms and field trials that demonstrate its practical value in the case of AUV missions. In [9], the problem was formulated as the *spatiotemporal optimal stopping* problem and was generalised to admit stochastic prediction models. The solution has guaranteed optimality and polynomial runtime. In [2], experiments demonstrate the value of planning while considering uncertainty in the trajectory and communication models. In this paper, we generalise the formulation for a team of trackers, and propose a decentralised solution algorithm.

The mission monitoring problem has similarities to a variety of related problems. The well-known optimal stopping problem [10] requires the choice of *when* to take a particular action in order to maximise an expected reward. Optimal stopping has motivated other robotics formulations, including selecting when to communicate [11] and sample [12] while moving along a single dimension. Mission monitoring can be described as a generalisation of these formulations where decision points are described in both spatial and temporal dimensions. The problem can be interpreted geometrically (Fig. 1) and thus has similarities to problems in computational geometry [13], such as Voronoi decomposition and the unions of rectangles. Sweep plane algorithms can be used to solve these geometric problems, and this idea forms the basis of the algorithms in [9] [2]. The general approach can equivalently be interpreted as finding the longest-path through a directed acyclic graph, similar to [14] but with a different spatiotemporal graph.

However, all of the problems above are formulated as single-agent problems, whose solutions do not naturally extend to multi-agent scenarios. The dynamic coverage problem [15] is closely related in that a team of robots coordinate to collectively observe a moving set of points. However, [15] assumes the points are static and rely on replanning to react to the dynamics. For mission monitoring, we have access to a prediction model of the dynamics, and thus we can instead plan with respect to this model. Also, [15] is centralised, whereas the algorithms here are decentralised.

Decentralised planners have been developed for many problems, including generic formulations such as Dec-MCTS [16] [7] and max-sum [17]. Problem-specific formulations include those for target search [18], tracking [19], and exploration [20]. While the generic planners may be applicable to mission monitoring, the performance will be poorer than our problem-specific solution; we conduct an empirical comparison to Dec-MCTS.

Coordination between trackers is achieved in our method with a decentralised framework that optimises a product distribution over the joint solution space. This optimisation is performed by considering candidate solution trajectories generated by spatiotemporal optimal stopping. The decentralised framework is similar to the second component of Dec-MCTS [7], which is an adaptation of *probability collectives* [21]. This approach has been shown to be robust to unpredictable communication loss [7], which is particularly useful in marine operations, and can even incorporate communication scheduling to enable more effective use of communication resources [22]. This general approach to decentralisation is analogous to the classic mean-field approximation and related variation inference methods [23], where a global likelihood function is approximated by a collection of structurally simpler distributions.

III. Problem Formulation

In this section we formulate the multi-tracker mission monitoring problem. The problem involves a team of mobile agents: 1) a *target* agent which follows a probabilistic trajectory defined by a mission plan, and 2) a team of *tracker* agents that seek to effectively monitor the target throughout the mission. At each time instant, to monitor effectively, at least one tracker must be stationary and within observation/communication range of the target. The trajectory for each of the trackers can therefore be characterised as a sequence of stopping waypoints in time and space. The optimisation problem is to find the trajectories for the team of trackers that maximises the expected monitoring effectiveness. This problem is to be solved in a decentralised manner. We formally define this problem as follows.

A. Target

The predicted future trajectory of the target is represented as a sequence of random variables $X := (X_1, X_2, ..., X_N)$ with associated timesteps $(t_1, t_2, ..., t_N) = \mathcal{T}$. The timesteps are evenly spaced at Δ_t intervals, with $t_1 = 0$, and $t_N = T$ is the mission duration or a time horizon. Each X_i represents the predicted location of the target at time t_i and has a known probability density function $\rho_i(X_i = x)$ over the domain $\mathcal{X}$.

B. Tracker Team

The target is monitored by a team of R tracker agents $\{1, 2, ..., R\} = \mathcal{R}$. The trajectory of tracker $r \in \mathcal{R}$ is represented by a sequence of positions $Y^r = (y_1^r, y_2^r, ..., y_N^r)$ and states $S^r = (s_1^r, s_2^r, ..., s_N^r)$, with associated timesteps $\mathcal{T}$ (as defined above). The trajectory of the tracker is characterised as alternating between two states $\{\text{STOPPED}, \text{MOVING}\}$. If $s_i^r = \text{STOPPED}$, then at time t_i tracker r is stationary at position y_i^r. Conversely, if $s_i^r = \text{MOVING}$, tracker r is moving between waypoints and therefore not monitoring.

The trajectory of tracker r is equivalently represented by the tuple $\pi^r = [\hat{Y}^r, T^{\text{a}r}, T^{\text{d}r}]$, where $\hat{Y}^r := (\hat{y}_1^r, \hat{y}_2^r, ..., \hat{y}_{M^r}^r)$ is a sequence of waypoint positions with sequences of associated arrival times $T^{\text{a}r} := (t_1^{\text{a}r}, t_2^{\text{a}r}, ..., t_{M^r}^{\text{a}r})$ and departure times $T^{\text{d}r} := (t_1^{\text{d}r}, t_2^{\text{d}r}, ..., t_{M^r}^{\text{d}r})$. We have $\hat{y}_j^r \in \hat{\mathcal{Y}}$, where $\hat{\mathcal{Y}}$ is a discrete set of positions where the trackers may stop.

During the time interval $[t_j^{\text{a}r}, t_j^{\text{d}r})$, tracker r is in the STOPPED state and is stationary at the waypoint position $\hat{y}_j^r \in \hat{\mathcal{Y}}$. During the time interval $[t_j^{\text{d}r}, t_{j+1}^{\text{a}r})$, tracker r is in the MOVING state and is travelling between consecutive waypoints $\hat{y}_j^r, \hat{y}_{j+1}^r$. The sequences of arrival and departure times satisfy the constraints: $t_1^{\text{d}r} \geq 0, t_{M^r}^{\text{a}r} \leq T$, and $t_j^{\text{a}r} < t_j^{\text{d}r} < t_{j+1}^{\text{a}r}, \forall j \in \{1, 2, ..., M^r - 1\}$

The required travel time between two waypoints $\hat{y}_a^r, \hat{y}_b^r$ is defined by a function $\delta(\hat{y}_a^r, \hat{y}_b^r) : \hat{\mathcal{Y}} \times \hat{\mathcal{Y}} \to \mathbb{R}_{\geq 0}$. The proposed algorithm does not depend on the exact trajectory taken to achieve this travel time, but it may be computed by considering the dynamics models of the tracker vehicles. We require $\delta(\hat{y}_a^r, \hat{y}_b^r) = 0$ iff $\hat{y}_a^r = \hat{y}_b^r$. For clarity, we assume δ is the same for all trackers, although this could be easily generalised. We do not plan for collision avoidance between trackers; however, this could typically be handled by a low-level controller during execution with only slight changes to the travel times.

The start position $\hat{y}_1^r$ for each tracker r is a known constant, while the end position $\hat{y}_{M^r}^r$ is to be selected by the planner from a set $\hat{\mathcal{Y}}_\text{end} \subseteq \hat{\mathcal{Y}}$. Alternative start/end assumptions are addressed in [2] with small modifications to the algorithms.

The trajectories for the team of robots collectively is denoted: $\pi = \{\pi^1, \pi^2, ..., \pi^R\}$. The trajectory of all trackers except r is denoted $\pi^{(r)}$, i.e., $\pi^{(r)} := \pi \setminus \pi^r$. These superscript conventions are also used for all tracker variables: s_i, S, y_i, Y, $\hat{y}_j$, $\hat{Y}$, T^a and T^d.

C. Monitoring Effectiveness

At time t_i, the instantaneous monitoring effectiveness for tracker r only is described by a function f^r. This function is defined as the probability of monitoring effectively:

$$f^r(X_i, y_i^r, s_i^r) := \begin{cases} \tilde{f}(\|X_i - y_i^r\|) & \text{if } s_i^r = \text{STOPPED} \\ 0 & \text{if } s_i^r = \text{MOVING} \end{cases} \qquad (1)$$

where $\tilde{f}(d) : \mathbb{R}_{\geq 0} \to [0, 1]$ is the observation (or communication) model. This model $\tilde{f}$ describes the probability of successfully observing the target from a distance of d, although other interpretations of $\tilde{f}$ are possible [2]. This function may be defined as a simple binary r-disk model or a more realistic observation model; we present example definitions in Sec. VI and in [2]. For clarity, we define the observation model as tracker-, translation-, orientation- and time-invariant, however the approach can readily be extended for more general models.

The goal of the *team* of trackers is to collectively monitor the target. At time t_i, the monitoring effectiveness for the team is described by a function f, defined as follows. There is no additional reward for multiple trackers monitoring at the same time. However, having multiple trackers STOPPED at the same time increases the probability that at least one tracker is effectively monitoring. By assuming observation independence, we define f as

$$f(X_i, y_i, s_i) := 1 - \prod_{r \in \mathcal{R}} \left[1 - f^r(X_i, y_i^r, s_i^r)\right], \qquad (2)$$

which specifies the probability that at least one tracker is effectively monitoring at time t_i. The motivation for this formulation of f is that this model encourages the different trackers to observe different parts of the prediction model distribution X_i; if f was instead simply a sum of f^r then all trackers would aim to observe the most likely realisation of X_i, which is likely to result in undesirable behaviour.

The objective function F is defined as the expected monitoring time for the duration of the mission:

$$F(X, \pi) := \mathbb{E}_X \left[\Delta_\text{t} \sum_{i=1}^{N} f(X_i, y_i, s_i)\right] \qquad (3)$$

$$= \Delta_\text{t} \sum_{i=1}^{N} \mathbb{E}_{X_i} \left[f(X_i, y_i, s_i)\right] \qquad (4)$$

where $\{y_i, s_i\}$ are the trajectories derived from π, and the expectation is computed with respect to X. In our results, we express F as a percentage of the mission duration t_N.

D. Problem Statement

The optimisation problem to be solved is stated as follows.

Problem 1: For a given probabilistic model of the predicted target trajectory X, a set of possible waypoint locations $\hat{\mathcal{Y}}$, the start locations $\hat{y}_1^r \in \hat{\mathcal{Y}}, \forall r \in \mathcal{R}$, and the set of feasible end locations $\hat{\mathcal{Y}}_{\text{end}} \subseteq \hat{\mathcal{Y}}$, find for each tracker r the set of stopping waypoints π^r with positions $\hat{y}_j^r \in \hat{\mathcal{Y}}$, $\hat{y}_{M^r}^r \in \hat{\mathcal{Y}}_{\text{end}}$, arrival times T^{ar} and departure times T^{dr}, such that the travel time constraints $t_{j+1}^{\text{ar}} - t_j^{\text{dr}} = \delta(\hat{y}_j^r, \hat{y}_{j+1}^r), \forall j \in \{1, ..., M^r - 1\}, \forall r \in \mathcal{R}$ are satisfied, and the expected monitoring effectiveness for the team $F(X, \pi)$, as defined in (4), is maximised over the mission duration.

Problem 1 is to be solved in a decentralised manner. Specifically, each tracker r optimises its own trajectory π^r based on only the information known to tracker r. We assume tracker r knows the target prediction model X, but does not necessarily know the trajectories $\pi^{(r)}$ selected by the other robots. The trackers can communicate during planning-time to improve coordination, but this communication channel may be unpredictable and intermittent.

IV. Decentralised Planning Algorithm

In this section we present our decentralised planning algorithm as a solution to the multi-tracker mission monitoring problem. The algorithm runs simultaneously and asynchronously on all tracker robots; we present the algorithm from the perspective of tracker r.

The algorithm cycles between three phases: (1) find the optimal solution π_*^r with respect to the currently known information about the other trackers' plans, (2) maintain a set Π^r of possible solutions for π^r and optimise a probability distribution q^r over the set Π^r, and (3) communicate probability distributions with the other robots. These three phases continue regardless of whether or not the communication was successful, until a computation budget is met or the algorithm converges. Pseudocode is provided in Alg. 1.

A. Probability Distributions over Trajectories

The algorithm maintains a probability distribution for each tracker, which represents the predicted plan of each tracker. We define a probability mass function q^r, such that $q^r(\pi^r)$ defines the probability that robot r will select π^r. The domain of q^r is restricted to a *dynamically* selected subset Π^r of all possible solution trajectories. As the algorithm progresses, both the domain Π^r and distribution q^r are optimised. The product distribution of all trackers is denoted (Π, q), and of all trackers except r is denoted $(\Pi^{(r)}, q^{(r)})$.

B. Phase 1: Spatiotemporal Optimal Stopping

The first phase of the algorithm finds the solution π_*^r that is optimal with respect to the current information (probability distributions) available to tracker r. This solution gets incorporated into the set Π^r, which defines the domain of the probability distribution that is optimised later in phase 2.

Algorithm 1 Optimising π^r on-board tracker r.

```
 1: $\mathcal{G} \leftarrow$ GenerateGraph($X$)
 2: $\Pi^r \leftarrow \emptyset$                                  ▷ Set of solutions
 3: define $q^r$ as a probability distribution over $\Pi^r$
 4: $\beta \leftarrow \beta_0$                                   ▷ Temperature parameter
 5: loop
      ▷ Phase 1: Spatiotemporal optimal stopping
 6:     $\pi_*^r \leftarrow$ OptimalStopping($\mathcal{G}, \Pi^{(r)}, q^{(r)}$)
      ▷ Phase 2: Probability distribution optimisation
 7:     $\Pi^r$.RemoveMin($q^r$)             ▷ Remove least likely $\pi^r$
 8:     $\Pi^r$.Add($\pi_*^r$)
 9:     for each $\pi^r \in \Pi^r$ do
10:         $q^r(\pi^r) \leftarrow$ Update($q^r(\pi^r), \beta$)          ▷ Eqn. (6)
11:     $\beta \leftarrow$ Cool($\beta$)
      ▷ Phase 3: Communication
12:     CommunicateTransmit($\Pi^r, q^r$)
13:     $\Pi^{(r)}, q^{(r)} \leftarrow$ CommunicateReceive
    return $\pi^r \leftarrow \text{argmax}_{\pi^r \in \Pi^r} q^r(\pi^r)$
```

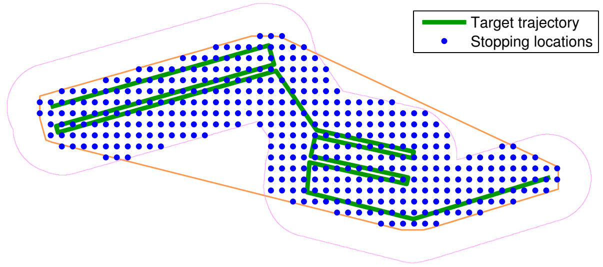


Fig. 2. Example showing possible stopping locations around a deterministic target trajectory moving through 2 spatial dimensions. Also shown are the boundaries of the monitoring region (pink) and convex hull (orange) used for culling the search space. Figure sourced from [2].

We find π_*^r by using a modification of the spatiotemporal optimal stopping algorithm for the single-tracker problem [2]. The algorithm consists of generating a search graph over time and space, followed by a longest-path search through the graph to find the optimal trajectory for the tracker. We extend [2] to also consider the current plans for the other trackers when evaluating the new reward function (2). We also modify the graph generation of [2] to enable more efficient repeated queries, which is particularly useful in the context of Alg. 1. We summarise the algorithm as follows and highlight the main differences.

1) Graph generation: During a precomputation step (Alg. 1 line 1), a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is generated such that any path through this graph represents a trajectory π^r for tracker r. Each vertex $v_\eta \in \mathcal{V}$ represents a potential stopping location in time and space. The set $\mathcal{V}$ is generated by first considering the set of potential stopping locations $\hat{\mathcal{Y}}$. This set is then culled by only keeping positions that are both within the observation range of $\mathcal{X}$, and within the convex hull of $\mathcal{X}$, where $\mathcal{X}$ is the set of all positions that have a non-zero probability of occurrence in X. An example of these sets and the resulting stopping locations $\mathcal{P}$ are depicted in Fig. 2.

Each vertex $v_\eta \in \mathcal{V}$ represents a position $p_\eta \in \mathcal{P}$ and a time interval $[\tau_\eta, \tau_\eta + \Delta_t] \subseteq \mathcal{T}$, denoted by the tuple

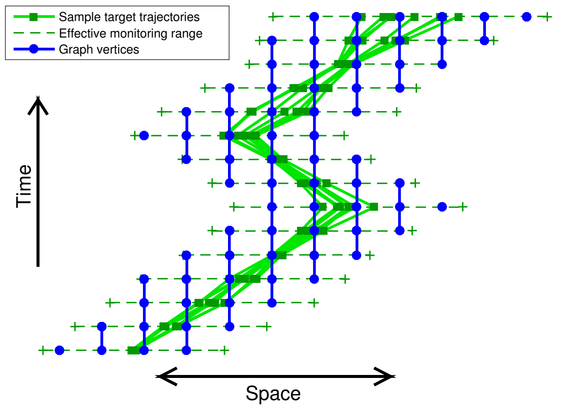


Fig. 3. Example showing graph vertices overlaying the (sampled) probabilistic trajectory of a target moving through a single spatial dimension. Each vertical blue line segment is a vertex in the search graph. Each vertex maps to a potential STOPPED position and time for the tracker.

$v_\eta := [p_\eta, \tau_\eta]$. For each position $p_\eta \in \mathcal{P}$, a vertex is created for each time step $\tau_\eta \in \mathcal{T}$ where the target has a non-zero probability of monitoring the tracker. An example of this vertex generation is illustrated in Fig. 3 overlaying a probabilistic target trajectory.

A solution trajectory is represented by a path through the graph with consecutive vertices connected by directed edges $e_\eta \in \mathcal{E}$. An edge $e_\eta = \langle v_i, v_j \rangle$, where $p_i \neq p_j$, describes travelling from vertex v_i at position p_i to vertex v_j at position p_j. Edges are connected between each pair of vertices that have feasible travel times, i.e., edge $\langle v_i, v_j \rangle \in \mathcal{E}$ iff $\delta(p_i, p_j) \leq \tau_j - \tau_i$. The arrival time at p_j is selected as $t_j^{\text{ar}} = \tau_j$ and the departure time from p_i is $t_i^{\text{dr}} = \tau_j - \delta(p_i, p_j)$. Any vertex that has no feasible path back to the start vertex $[\hat{y}_1^r, 0]$ is excluded from $\mathcal{V}$. Edges are also included for cases where $p_i = p_j$, which represent stopping at position p_i for an extended period of time.

Unlike in [2], to make the repeated queries in Alg. 1 more computationally efficient, we make a further adjustment to the set $\mathcal{E}$. For a fixed vertex v_j, if there are multiple feasible edges $\langle v_i, v_j \rangle$ with different v_i that all have the same location p_i, then only the edge with the latest τ_i is kept, while all others are excluded. Optimality is maintained after this adjustment (corollary of [2, Remark 2]). This improvement yields a runtime complexity for phase 1 that is linear in the resolution of the temporal discretisation (see Sec. V-A).

2) Graph edge weights: In the main loop of the algorithm (line 5), rewards are assigned to the graph edges and then the optimal π_*^r is found. This is performed while considering the plans $(\Pi^{(r)}, q^{(r)})$ of the other trackers, which change each time a communication message is received.

The reward ω_η for edge $e_\eta = \langle v_i, v_j \rangle$ represents the relative value of including edge e_η in the solution path π_*^r. Specifically, we define ω_η as the expected increase in probability of effective monitoring if tracker r were to stop at location p_i at time τ_i, i.e.,

$$\omega_\eta := \mathbb{E}_{X_i, y_i^{(r)}, s_i^{(r)}} \Big[f(X_i, y_i, s_i^{(r)} \cup s_i^r = \text{STOPPED}) \\ - f(X_i, y_i, s_i^{(r)} \cup s_i^r = \text{MOVING}) \Big] \quad (5)$$

where $y_i^r = p_i$, and f is defined in (2). The purpose of computing the *increase* in probability rather than *absolute* probability is to be less affected by noise caused by uncertainty in the other trackers' plans. During the timesteps that tracker r is moving from p_i to p_j, $\omega_\eta = 0$, and thus these timesteps do not need to be evaluated.

The expectation in (5) is computed with respect to several random variables. The $y_i^{(r)}, s_i^{(r)}$ variables represent the position and state of the other trackers, which can be considered by summing over the discrete probability distribution $(\Pi^{(r)}, q^{(r)})$, while evaluating the positions of $\pi^{(r)} \in \Pi^{(r)}$ at time τ_i. If the number of robots or the cardinality of Π is large, then this summation becomes intractable, and therefore should instead be approximated using sampling. The X_i variable is considered by integrating with respect to the PDF ρ_i. The best way to compute this integral would depend on the representation; we use a sampled representation for X and evaluate using Monte Carlo integration.

We note that, unlike the definition of ω_η in [2], in (5) we have ignored the effect of having departure times t_i^{dr} that fall between the discrete time indices. This reduces computation time since edges $\langle v_i, v_a \rangle$ and $\langle v_i, v_b \rangle$ will have the same weight ω_η, and thus (5) only needs to be evaluated once for each vertex, rather than for each edge.

3) Graph search: The optimal tracker trajectory π_*^r is found by searching for the longest-path through the graph $\mathcal{G}$. For general graphs, a longest-path search is NP-hard. However, $\mathcal{G}$ is a directed acyclic graph, and thus we can find the optimal longest-path in polynomial time. This search can be thought of a sweep-plane moving forwards through time. As each vertex v_j is visited, the optimal edge $\langle v_i, v_j \rangle$ is stored, which represents the optimal path if the trajectory were to finish at v_j. The vertex with location in $\hat{\mathcal{Y}}_{\text{end}}$ that has the largest accumulated reward at time t_N is selected as the end vertex. Finally, the trajectory π_*^r is found by backtracking from the end vertex to the start vertex.

C. Phase 2: Decentralised Coordination

In phase 2, the trackers coordinate their plans by jointly optimising a probability distribution over their trajectories in a decentralised manner. The domain of the probability distribution Π^r for tracker r is constructed using trajectories generated in phase 1. The probability distribution q^r optimised in phase 2 is communicated to the other trackers during phase 3, then used by the other trackers when planning their own trajectories.

The domain Π^r is constructed by adding the trajectory π_*^r each time phase 1 is run. The set Π^r should be a small set to keep communication packets small and computation efficient; thus, once a fixed size has been reached, a trajectory is also removed each time one is added. The trajectory with the lowest probability $q^r(\pi^r)$ is selected to be removed. When a new trajectory π_*^r is added, it is assigned a probability $q^r(\pi_*^r) = \max_{\pi^r \in \Pi^r} q^r(\pi^r)$, and then q^r is renormalised. This is performed in Alg. 1 lines 7–8. This construction of Π^r is motivated by, but distinctly different to, Dec-MCTS since the phase 1 of this algorithm generates a new, *optimal* solution at each iteration; in contrast, Dec-MCTS periodically resets Π^r since the equivalent phase 1 is an incremental, converging planner. The benefits of defining Π^r as a compact set with cardinality greater than 1 is analysed in Sec. V and demonstrated empirically in Sec. VI.

The probability distribution q^r is optimised using a decentralised gradient descent scheme that is equivalent to the second phase of Dec-MCTS [7], which is an adaptation of probability collectives [21]. We chose to use this approach since it has interesting theoretical and practical properties, though other similar methods could also be considered here. This descent scheme is formulated as finding the q^r that has minimum KL-divergence to the optimal *joint* probability distribution. Thus, this formulation indirectly optimises the joint plans of the team in a distributed manner. This is performed in Alg. 1 lines 9–10.

Specifically, this optimisation is defined such that during each phase 2, each component $q^r(\pi^r)$ of q^r is updated as

$$\begin{aligned} q^r(\pi^r) \leftarrow q^r(\pi^r) &- \alpha q^r(\pi^r) \\ &\times \left[\frac{\mathbb{E}_\pi[F(X,\pi)] - \mathbb{E}_{\pi^{(r)}}[F(X,\pi) \mid \pi^r]}{\beta} \right. \\ &\qquad \left. + \mathrm{H}(q^r) + \ln\left(q^r(\pi^r)\right) \right] \end{aligned} \tag{6}$$

then q^r is renormalised, where H is entropy, α is a small constant, and β is a temperature parameter that is cooled to slowly decrease the entropy of q^r. The intuition behind this update step (6) is that the probability of selecting π^r is increased if selecting π^r would result in a larger expected reward compared to the expected reward if tracker r were to sample a trajectory from q^r. The last two terms in (6) control the entropy of the distribution, which is reduced slowly to avoid making a decision too quickly and getting stuck in local optima. The expectations in (6) are computed as

$$\mathbb{E}_\pi[F(X,\pi)] := \sum_{\pi \in \Pi} \left[F(X,\pi) \prod_{r' \in \mathcal{R}} q^{r'}(\pi^{r'}) \right] \tag{7}$$

and similarly for $\mathbb{E}_{\pi^{(r)}}[F(X,\pi) \mid \pi^r]$ except tracker r's trajectory is fixed as π^r. Typically, it is necessary to approximate these summations (7) by sampling from q.

D. Phase 3: Communication

In phase 3 (Alg. 1 lines 12–13), tracker r broadcasts its current probability distribution (Π^r, q^r) to the other trackers. If tracker r receives an updated distribution $(\Pi^{r'}, q^{r'})$ from tracker r', then this replaces the locally stored distribution for r'. The updated distribution is used during phases 1 and 2 of the next iteration. If messages are lost, e.g. due to an unreliable communication channel, then each tracker will continue planning based on the most recently received distributions. In [22], we present an extension to this communication phase approach that incorporates *communication scheduling*; this extended approach could also be directly applied to our decentralised mission monitoring algorithm.

V. Analysis

A. Runtime Complexity

The proposed algorithm is an iterative algorithm and a feasible solution is computed at each iteration, and thus is any-time. It is difficult to determine how many iterations are required before the algorithm reaches a satisfactory solution (as discussed in Sec. V-B), however we can analyse the runtime per iteration as follows.

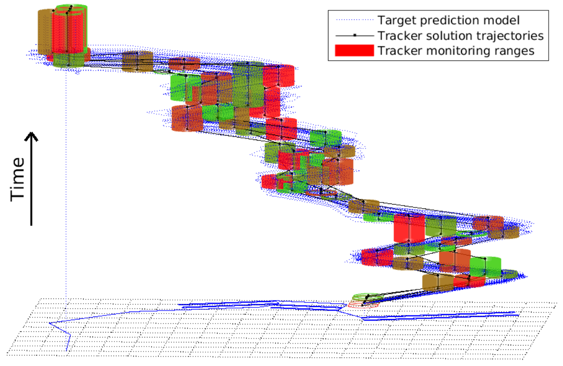
The computation time for the single-tracker spatiotemporal optimal stopping algorithm is $\mathcal{O}(|\mathcal{P}|^2|\mathcal{T}|^2)$ where $|\mathcal{P}|$ is the spatial resolution and $|\mathcal{T}|$ is the temporal resolution [2]. For the multi-tracker problem, we have split this algorithm into a precomputation step, which has runtime $\mathcal{O}(|\mathcal{P}|^2|\mathcal{T}|^2)$, and phase 1, which has runtime $\mathcal{O}(\psi|\mathcal{V}| + |\mathcal{E}|) = \mathcal{O}(\psi|\mathcal{P}||\mathcal{T}| + |\mathcal{P}|^2|\mathcal{T}|)$ where $|\mathcal{V}|$ is the number of graph vertices, $|\mathcal{E}|$ is the number of edges and ψ is the time taken to compute the expectation (5). We note that this runtime for phase 1 is linear in $|\mathcal{T}|$, rather than quadratic as achieved in [2]; this improved runtime is due to the removal of unnecessary edges from $\mathcal{E}$. While these runtimes are polynomial, ψ is exponential in the number of robots if (5) is computed exactly, but can be efficiently and adequately approximated using sampling, as discussed in Sec. IV-B.2. The runtime of phase 2 is dominated by computing the expectations (7) and thus should also be approximated using sampling.

B. Optimality and Convergence

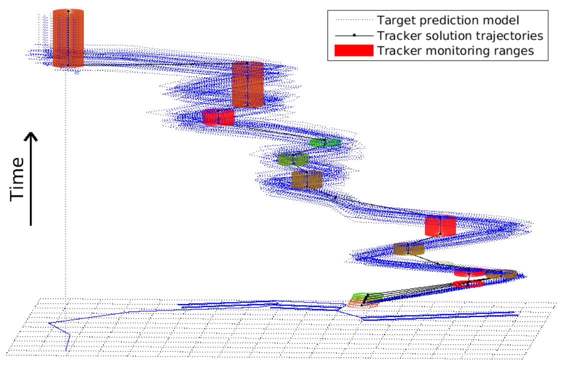
While, due to the inherent challenges of decentralised planning, it is difficult to provide any guarantee of global optimality, we analyse the two main components of the algorithm and their interaction as follows to support the use of these components in our algorithm.

The spatiotemporal optimal stopping algorithm is optimal, including when performing the vertex culling in the graph generation phase [2]. This optimality result directly applies to our phase 1, where optimality is measured with respect to the current information (distributions $\Pi^{(r)}, q^{(r)}$) available to the tracker. However, this optimality result does not imply global optimality for the *joint* plan of the team. In fact, if $\Pi^{(r)}, q^{(r)}$ changes then tracker r may change its decision and vice versa, and this may continue indefinitely.

This observation motivates the need for phase 2 of the algorithm. The probabilistic formulation of phase 2 ensures that each tracker gradually settles upon a solution as the entropy is lowered, rather than responding drastically to changes in $\Pi^{(r)}, q^{(r)}$, and thus will typically overcome the cyclic dependency problem. Additionally, our phase 2 is similar to phase 2 of Dec-MCTS, and thus the analysis of [7, Proposition 1] holds here: the product distribution q asymptotically converges to a distribution that locally minimises the KL-divergence to the optimal *joint* distribution, assuming that the sets Π are selected sufficiently. It is unclear what defines a sufficient selection of solutions for Π^r; however, we argue that our method of using candidate solutions generated by phase 1 is an appropriate heuristic for this purpose. The following experiments empirically support these claims.



(a) **With the proposed algorithm**, trackers successfully coordinate their actions to achieve a monitoring effectiveness of 60%.



(b) **Without coordination**, the monitoring regions significantly overlap, resulting in a lower monitoring effectiveness of 37%.

Fig. 4. Comparing planning with and without communication by a team of 5 trackers. See Fig. 1 caption for a description of this graphical representation.

VI. EXPERIMENTS: AUV MONITORING

These experiments demonstrate the behaviour and performance of the planning algorithm for AUV mission monitoring, and support our theoretical claims. We simulate a scenario where an AUV is monitored by a team of surface vessels that communicate with the AUV via acoustic communication. The AUV probabilistically follows a mission plan, and the surface vessels coordinate using our approach.

A. Scenario

This scenario is a multi-tracker generalisation of the AUV case study in [2]. The prediction model for the AUV simulates the trajectory of a path-following mission. The AUV follows a 7 km mission plan that was previously executed in Middle Harbour, Sydney [1]. Random disturbances are added to simulate realistic sources of uncertainty. The sequence of spatial probability distributions of the AUV's position is represented as a set of sample trajectories. Each sample trajectory is generated by iteratively: (1) updating the state by sampling a stochastic kinematics model, (2) adding localisation noise, and then (3) executing a control policy. The kinematics uses the unicycle model with added noise for modelling ocean currents and control uncertainty. The AUV's angular velocity is controlled using the policy in [24, Ch. 9.3]. We add an adaptive behaviour such that there are three decision points where the AUV may deviate by 200 m from the path. Observations are modelled as acoustic communication between the AUV and the surface vessels. We use the realistic underwater acoustic communication model in [8], which defines the probability of successful communication as a decreasing function of distance.

Each scenario consists of 4 to 8 surface vessels. Parameters are chosen such that the mission has 1 hour duration, the AUV moves at 2 m/s, the trackers move at 3 m/s with a 60 s time penalty for each occasion the tracker stops (e.g., for deploying hardware), the trackers start near the AUV positioned 50 m apart, and the communication model parameters are such that the probability of an observation is defined as a continuous function with 100% probability at 0 m, 50% at 50 m and 0% for ≥ 250 m. The set $\hat{\mathcal{Y}}$ forms a grid with 50 m spacing, and time is discretised at 60 s intervals.

B. Results

First we demonstrate that the trackers must coordinate their actions in order to achieve a reasonable monitoring effectiveness. Fig. 4(a) shows an example solution where the trackers successfully coordinate to observe most of the prediction model. In contrast, in Fig. 4(b) the trackers planned independently, resulting in poor monitoring effectiveness due to the trackers selecting overlapping observations.

The next results, shown in Fig. 5, demonstrate the benefit of planning with a probabilistic representation of the plan. When the set of trajectories Π^r is restricted to only keeping the current trajectory π_*^r, the solution switches between multiple suboptimal solutions as each tracker reacts to the other trackers changing their plan. The trackers do not settle on a single solution, and the monitoring effectiveness remains relatively low. In contrast, planning with a probabilistic representation quickly converges to a stable solution that clearly outperforms the deterministic case. Similar results are achieved between using a distribution size limit of 5 or unlimited; thus a smaller size is recommended for efficiency.

Finally, we compare our proposed approach to Dec-MCTS [7] as an alternative decentralised algorithm. Our planner and Dec-MCTS have a similar three-phase cycle with similar phases 2 and 3. The key difference is that in phase 1 Dec-MCTS employs a generally applicable, incremental planner, which is a novel variant of Monte Carlo tree search. In our algorithm, phase 1 is a problem-specific solution that is optimal with respect to the current information. The results are shown in Fig. 6. Our proposed approach achieves the fastest convergence and the best overall performance. The practical performance of Dec-MCTS is largely dependent on the choice of rollout policy; Dec-MCTS with a rollout policy equivalent to our phase 1 was slower to converge but achieved similar results, Dec-MCTS with a random rollout policy achieved the worst results, and Dec-MCTS with a 50% mixture of the two rollout policies achieved intermediate results. These results show our approach outperforms a generic planner at this problem, and also show that our approach can be used as a guiding heuristic by generic planners. We note that since the problem is new, we do not have algorithms for

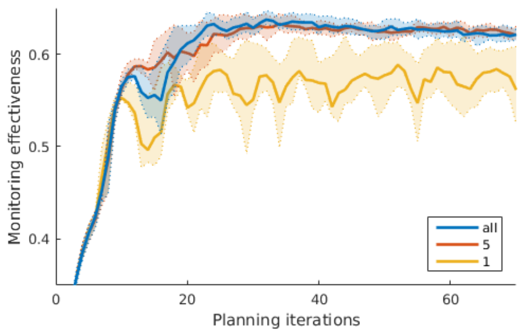


Fig. 5. Convergence of the algorithm with different sizes of the trajectory set Π^r in the plan representation. Using 1 trajectory is a deterministic representation, while 5 and 'all' are probabilistic. 8 robots were used in this scenario. Averages and quartiles shown from 20 trials.

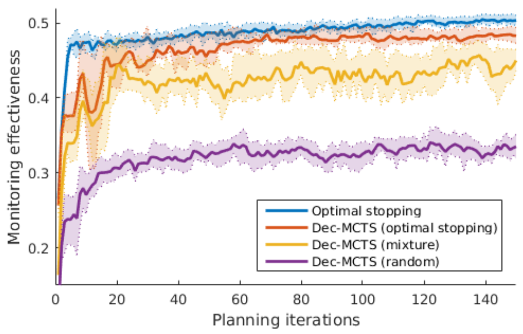


Fig. 6. Comparison between our proposed method and the generic Dec-MCTS decentralised algorithm with three different rollout policies (in parenthesis). 4 robots are simulated in a smaller problem instance than the Fig. 5 scenario. Averages and quartiles shown from 20 trials.

direct comparison other than generic planners. It would be interesting to compare to an optimal *centralised* algorithm though a key challenge that would need to be overcome first is the intractability of searching over this joint solution space that grows exponentially in the number of robots.

The experiments were simulated in MATLAB on a standard desktop computer, and the computation times were on the order of several seconds to minutes for all scenarios. We note that, in practice, computation time can be tuned by varying the number of iterations and the discretisation resolution to meet the requirements of a real-time application.

VII. Conclusion and Future Work

We have formulated and solved the multi-tracker mission monitoring problem, which has broad practical applications, especially for marine robotics operations. The solution is a novel decentralised algorithm that inherits and extends several useful properties from approaches for the single-tracker problem and generic decentralised planning. In our experiments, the solutions are reached after only a small number of communication messages are broadcast. In future work, we would like to consider other multi-robot variants, such as where a team of surface vessels monitor a fleet of AUVs. It would also be interesting to *jointly* optimise the plans of the target and trackers. Our approach is fast enough to accommodate online replanning for adapting to changes, but planning an adaptive policy would also be an interesting challenge. Centralised planners could also be considered, where the key algorithmic challenge would be to develop scalable algorithms for larger teams.

References

[1] G. Best and S. Anstee, "Motion planning for autonomous underwater vehicle supervision," in *Proc. of ARAA ACRA*, 2014.
[2] G. Best, W. Martens, and R. Fitch, "Path planning with spatiotemporal optimal stopping for stochastic mission monitoring," *IEEE Trans. Robot.*, vol. 33, no. 3, 2017.
[3] C. German, M. Jakuba, J. Kinsey, J. Partan, S. Suman, A. Belani, and D. Yoerger, "A long term vision for long-range ship-free deep ocean operations: Persistent presence through coordination of autonomous surface vehicles and autonomous underwater vehicles," in *Proc. of IEEE/OES AUV*, 2012.
[4] R. Brockers, P. Bouffard, J. Ma, L. Matthies, and C. Tomlin, "Autonomous landing and ingress of micro-air-vehicles in urban environments based on monocular vision," *Proceedings of SPIE*, vol. 8031, pp. 803 111 1–12, 2011.
[5] T. Naseer, J. Sturm, and D. Cremers, "FollowMe: Person following and gesture recognition with a quadrocopter," in *Proc. of IEEE/RSJ IROS*, 2013, pp. 624–630.
[6] M. Dunbabin and A. Tews, "Toward robotic visual and acoustic stealth for outdoor dynamic target tracking," in *Proc. of ARAA ACRA*, 2012.
[7] G. Best, O. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *Int. J. Robot. Res.*, 2018, doi:10.1177/0278364918755924.
[8] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. Sukhatme, "Distributed coordination and data fusion for underwater search," in *Proc. of IEEE ICRA*, 2011, pp. 349–355.
[9] G. Best, W. Martens, and R. Fitch, "A spatiotemporal optimal stopping problem for mission monitoring with stationary viewpoints," in *Proc. of Robotics: Science and Systems*, 2015.
[10] Y. S. Chow, H. Robbins, and D. Siegmund, *Great expectations: The theory of optimal stopping*. Houghton Mifflin Boston, 1971.
[11] M. Lindhé and K. Johansson, "Exploiting multipath fading with a mobile robot," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1363–1380, 2013.
[12] J. Das, F. Py, J. B. J. Harvey, J. P. Ryan, A. Gellene, R. Graham, D. A. Caron, K. Rajan, and G. S. Sukhatme, "Data-driven robotic sampling for marine ecosystem monitoring," *Int. J. Robot. Res.*, vol. 34, no. 12, pp. 1435–1452, 2015.
[13] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational geometry*. Springer, 2000.
[14] S. D. Bopardikar, S. L. Smith, and F. Bullo, "On dynamic vehicle routing with time constraints," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1524–1532, 2014.
[15] W. Hönig and N. Ayanian, "Dynamic multi-target coverage with robotic cameras," in *Proc. of IEEE/RSJ IROS*, 2016, pp. 1871–1878.
[16] G. Best, O. Cliff, T. Patten, R. Mettu, and R. Fitch, "Decentralised Monte Carlo tree search for active perception," in *Proc. of WAFR*, 2016.
[17] R. Stranders, A. Farinelli, A. Rogers, and N. Jennings, "Decentralised coordination of mobile sensors using the max-sum algorithm," in *Proc. of IJCAI*, 2009, pp. 299–304.
[18] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *Int. J. Robot. Res.*, vol. 28, no. 2, pp. 201–219, 2009.
[19] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Trans. Cybernetics*, vol. 48, no. 1, pp. 187–198, 2018.
[20] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Auton. Robots*, 2018, doi:10.1007/s10514-018-9778-6.
[21] D. H. Wolpert and S. Bieniawski, "Distributed control by Lagrangian steepest descent," in *Proc. of IEEE CDC*, 2004, pp. 1562–1567.
[22] G. Best, M. Forrai, R. R. Mettu, and R. Fitch, "Planning-aware communication for decentralised multi-robot coordination," in *Proc. of IEEE ICRA*, 2018.
[23] I. Rezek, D. S. Leslie, S. Reece, S. J. Roberts, A. Rogers, R. K. Dash, and N. R. Jennings, "On similarities between inference in game theory and machine learning." *J. Artif. Intell. Res.*, vol. 33, pp. 259–283, 2008.
[24] C. C. de Wit, B. Siciliano, and G. Bastin, *Theory of robot control*. Springer, 2012.