Cooperative Sensor Scheduling for Long Term Planning

Marek Hilton
School of Engineering
University of Birmingham
Birmingham, UK
0009-0008-9855-4285

Beth Jelfs
School of Engineering
University of Birmingham
Birmingham, UK
b.jelfs@bham.ac.uk

Marco Martorella
School of Engineering
University of Birmingham
Birmingham, UK
m.martorella@bham.ac.uk

Bill Moran
Electrical & Electronic Engineering
University of Melbourne
Melbourne, Australia
wmoran@unimelb.edu.au

Christopher Gilliam
School of Engineering
University of Birmingham
Birmingham, UK
c.gilliam.1@bham.ac.uk

***Abstract*—We present a sensor scheduling algorithm to plan the motion of multiple autonomous platforms for cooperative tracking of targets within a region that contains obstacles and occlusions. The platforms have kinematic constraints and their sensors have restricted field of view and range. The proposed algorithm is a variant of the Rapidly exploring Random Tree star algorithm (RRT*) that has been adapted to the problem of determining paths for multiple independent kinematically constrained platforms to optimise their tracking performance. To guide the scheduling algorithm, we define a tracking cost based on the Posterior Cramér Rao Bound (PCRB) derived from the predicted positions of the platforms and targets. Through simulations of generated paths, we show that the algorithm generates rational plans for tracking targets and that the tracking cost accurately predicts the realised performance of the platforms.**

***Index Terms*—RRT, Planning, Sensor Scheduling, Tracking**

I. Introduction

The recent growth in autonomous systems and sensing capabilities paves the way for large-scale networks of autonomous sensing platforms in applications such as remote sensing [1], search-and-rescue, and surveillance [2]. To realise their full potential, these platforms need to not only fuse multiple sources of information but also organize in a cooperative manner to maximise collection of relevant information for the task at hand. For example, consider multiple mobile platforms attempting to track a target, or targets; the platforms need to cooperatively plan their trajectories to maximize the information gained from their measurements. Importantly, this planning needs to adapt to uncertainty, both in unknown objectives and in sensory measurements. These sources of uncertainty make this problem distinct from typical path planning and route finding problems, where the objective is to find a viable path from a starting state to an end state. Instead, such problems fall into the class of sensor scheduling [3].

Sensor scheduling encapsulates both the problem of fusing the sensory information and the problem of how to organize/manage the sensors to maximise information gained for a given task [3]. Cooperative sensor scheduling for tasks, such as tracking, is a growing area [2]. However, direct solutions to this cooperative problem are, typically, computationally infeasible, arising from the exponential growth in possible actions with both the number of platforms and the length of the time horizon. Instead, current efforts are generally limited to either myopic scheduling (short-time horizon) [1] or an action-constrained long-horizon planner [4]. Another option, from multi-agent systems, is to centrally learn a scheduling policy and then have the scheduler run independently on each sensor [5], [6]. This independence means a sensor's actions are not based on information/behaviour of other sensors, so limiting the advantage of having a network.

An alternative approach for long-horizon planning, that does not restrict the action space of the platform, is to take advantage of the widely used Rapidly exploring Random Tree star algorithm (RRT*) [7]. Our previous work [8]–[10], introduced the concept of using the RRT* to plan the trajectory of a single platform tracking a single target. Traditionally, in robotics, the RRT* algorithm plans the route of a platform to travel from A to B while avoiding obstacles based on minimising the Euclidean distance between A and B [7]. For detailed review of RRT* algorithms, we refer the reader to [11]. In contrast, our previous work adapted the RRT* algorithm so that it constructed a trajectory that minimised an information theoretic tracking cost. The key advantage of this approach is that it offered a computationally efficient method for constructing long-horizon trajectories without restricting the platform's movement to a discrete, finite, set of actions. We now build upon this work to develop a cooperative, long-horizon, scheduling algorithm.

In this paper, we consider the problem of cooperatively planning the trajectories of multiple platforms to track multiple targets. We propose a cooperative sensor scheduling algorithm to generate these trajectories based on adapting

This work was supported by the UK Defence and Security Accelerator.

the traditional RRT* [7]. In contrast to the commonly used Dubins model [12], our approach uses a highly constrained kinematic platform model when constructing the RRT* tree. The advantage of this approach is that we avoid the tendency of Dubins models to promote paths that *weave*; all turns are assumed to be at the maximum rate in the model. Furthermore, since the Dubins model allows motion between any two states, more nodes in the RRT* tree need to be considered when making local refinements, which increases the computational burden of the resulting RRT* algorithm. To avoid this problem, we combine our highly constrained kinematics with a second-stage tree refinement step so as to alter the timing of (re)wired paths without changing the shape of the path itself. Building on previous work [13]–[15], the tree is constructed using a tracking cost based on the posterior Cramèr Rao lower bound (PCRB) [16] that accounts for uncertainty in sensor measurements. The platforms, in our problem, are assumed to have perfect communications and so can maintain a centralised track for each target of interest, but each platform may move independently to obtain measurements of different targets at different times and from different positions. We show through simulation results that the plans generated by the proposed algorithm are rational and accurately predict tracking performance.

The rest of this paper is organised as follows. In Section II we define mathematical models for the platform states, target states, platform kinematics, and physical environment. Section III describes our implementation of the RRT* algorithm, including modifications that allow the algorithm to work for multiple platforms with kinematic constraints. The calculation of the tracking cost used to evaluate the performance of paths and guide the algorithm is then described in Section IV. Section V, provides simulation results and finally Section VI concludes the paper.

II. Mathematical Models

A. Platform State

The RRT* algorithm creates a tree in which each node corresponds to a prospective reachable state for the platforms, and where edges form the potential paths through these prospective states. The state represented by the i^{th} node in the tree, $\boldsymbol{n}_i$, comprises the individual states of each of the N platforms: $\boldsymbol{n}_i = [\boldsymbol{s}_{i,1}^{\mathsf{T}}, \cdots, \boldsymbol{s}_{i,N}^{\mathsf{T}}]^{\mathsf{T}}$, where $\boldsymbol{s}_{i,j}$ is the state of the j^{th} platform at the i^{th} node. The state of each platform is characterised by its position and heading; that is, $\boldsymbol{s}_i = [\boldsymbol{p}_i^{\mathsf{T}}, h_i]^{\mathsf{T}}$ where $\boldsymbol{p}_i \in \mathbb{R}^2$ is the position of the platform in Cartesian coordinates and h_i is the heading of the platform in radians relative to a fixed direction. The speed of the platform is assumed to be constant and is, consequently, omitted from the state representation.

B. Target State

The purpose of the multi-platform planner is to track a number of targets of interest. The j^{th} target at time step k is $\boldsymbol{x}_k^j = [x_k^j, v_{x,k}^j, y_k^j, v_{y,k}^j]^{\mathsf{T}}$, where $\boldsymbol{p}_k^j = [x_k^j, y_k^j]^{\mathsf{T}}$ is the position of the platform in Cartesian coordinates and $\boldsymbol{v}_k^j = [v_{x,k}^j, v_{y,k}^j]^{\mathsf{T}}$ is its velocity vector. We adopt a standard constant velocity model for target motion, perturbed by a continuous white noise acceleration [17], thus, the transition of a target state from time step k to time step $k+1$ is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{e}_k = \left(\begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \otimes \boldsymbol{I}_2 \right) \boldsymbol{x}_k + \boldsymbol{\eta}_k, \qquad (1)$$

where Δ represents the time between step k and $k+1$, $\otimes$ is the Kronecker product and $\boldsymbol{I}_2$ is the 2×2 identity matrix. The vector $\boldsymbol{\eta}_k$ represents the process noise, that is a zero-mean Gaussian variable with a typical covariance $\boldsymbol{Q}_\Delta$:

$$\boldsymbol{Q}_\Delta = q \left(\begin{bmatrix} \frac{\Delta^3}{3} & \frac{\Delta^2}{2} \\ \frac{\Delta^2}{2} & \Delta \end{bmatrix} \otimes \boldsymbol{I}_2 \right), \qquad (2)$$

where q represents the magnitude of the process noise; expressing the deviation of the target motion from the constant velocity model. The planning algorithm is assumed to have prior knowledge of the motion model and process noise covariance, that is, there is no model mismatch.

C. Kinematic Model

Autonomous platforms may have a high degree of manoeuvrability, as, for instance, a multicopter uncrewed aerial vehicle (UAV), or have more highly kinematically constrained movement, for example, a fixed wing UAV, uncrewed sea surface or ground vehicle. The RRT family of algorithms (see, for instance, [7], [9]–[11], [18], [19]) has previously been applied to planning problems where agents are kinematically constrained. We adopt a platform model where the constraints correspond to the movement being similar to a fixed wing UAV. Constrained movement and the use of a forward-looking sensor (Section IV-B) provide a more challenging problem where platforms cannot loiter around targets to improve tracking performance. In this work, the platform kinematic paths are restricted as follows:

1) The speed of a platform is constant;
2) A platform can make turns up to a minimum turn radius (maximum turn rate).

D. Environment

The physical environment within which the platforms and targets move is assumed to be a square region. Throughout the region there can be occluding objects that block line of sight for sensors, as well as "no-go" areas where platforms are not permitted to enter, but can, nonetheless, see through. In our simplified model, these occlusions and obstacles can be any arbitrary polygons. In practice, occluding objects could include terrain features or areas of interference. Similarly, obstacles may be areas of rough terrain or simply restrictions on movement imposed by the operator.

III. Planning for Multiple Platforms with RRT*

In this section, we present a detailed description of the modified RRT* algorithm. The original RRT* algorithm of [7] constructs a tree iteratively, where each iteration adds one node to the tree and consists of four distinct steps: sampling,

steering, wiring, and rewiring. Our modified algorithm uses the same four steps; however, the kinematic constraints of the problem force modifications of wiring and rewiring.

We start with the definition of the steps of the RRT* algorithm, and then outline a further constrained *planning* kinematic model that determines the physical path selected by a platform between any two states. A key definition is the operation of *steering* that underpins our modifications to the algorithm. We then detail the wiring and rewiring steps of our modified RRT* algorithm.

A. The RRT* Iteration

In our variant of the RRT* algorithm, used to build a tree of prospective future states, the root node of the tree represents the starting states of the platforms. Each node in the tree has an associated cost, dependent on the kinematic path from the root to the node in question. By *kinematic path*, we mean the kinematically constrained path that the platform would follow through a sequence of nodes in the tree. The kinematic path between any parent-child nodes is further constrained by a planning specific kinematic model described in Section III-B.

Once the tree is constructed, the optimal path is then chosen by finding the node with the lowest cost and determining the kinematic path to it from the root node. The tree is constructed by N iterations of the modified RRT* algorithm where each iteration of the algorithm proceeds by the following four steps:

1) **Sample** — A random sample state is taken within the explorable region of the problem;
2) **Steer** — An intermediate *reachable* state between the nearest node in the tree and the random sample is determined through steering;
3) **Wire** — For each node in the neighbourhood of the steer state, steer towards the steer state. The resulting node with the lowest cost is adopted into the tree;
4) **Rewire** — The new node adopts nearby nodes in the tree via steering *path shortening* if it can reduce their cost.

The first two steps of an iteration, sampling and steering, represent the exploratory behaviour of the RRT* algorithm: the tree tends to expand into unexplored regions of the state space before generating more nodes in already explored regions [7]. First, a sample state is taken at random anywhere within the explorable region of the state space; that is, states that are not inside regions that the platform is not able to enter. We assume that the distribution of these random samples is uniform throughout the explorable space, but other distributions can be used to bias exploration to different parts of the space; for instance, in a route finding problem where there is a predetermined goal state, a typical strategy might be to alternate between random sampling and sampling the goal state. The algorithm then attempts to 'steer' towards the given sample moving towards it a maximum allowed distance. This produces a new steer state with a kinematic path up to a maximum length. We will refer to this state as the *sample steer state*. Typically, in the standard RRT* algorithm, it is this state that will be added to the tree in the wiring and rewiring steps, however, in our implementation we apply a second round of steering as part of the wiring and rewiring steps, which will be described in Section III-C.

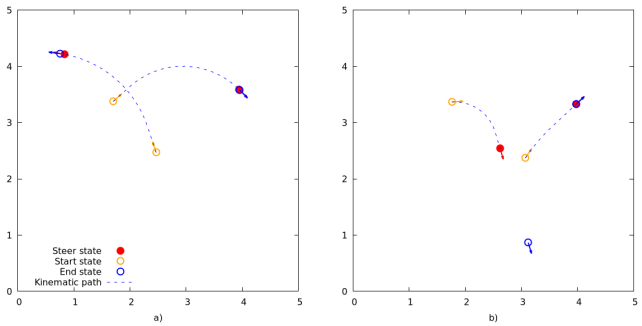


Fig. 1. Steering from a start state to an end state. An *intermediate* kinematically feasible state between the two states is determined where the length of the kinematic path of each platform is equal. In **a)**, the end state can be reached by both platforms simultaneously, whereas in **b)**, the end state cannot be reached simultaneously by both platforms under the kinematic constraints.

B. Planning Kinematic Model & Steering

Typically, an infinite number of paths between two states conform to the kinematic model as defined in Section II-C. To reduce complexity, and facilitate planning, we further constrain the kinematic paths between a parent and child node in the RRT* tree to eliminate ambiguity. We will call the more highly constrained model the *planning kinematic model* to distinguish it from the less stringent kinematic model of the real platforms. To mitigate issues with the commonly used Dubins model, such as weaving and the large number of neighbours to be considered in the wiring and rewiring steps of the algorithm, we have assumed the following simpler planning kinematic model:

KM1 The path between two states must consist of at most one arc of radius greater than $r_{\min}$ and at most one straight line segment;

KM2 The arc, if present, must be of maximum possible radius (the shallowest turn) compatible with reaching the final state;

KM3 Heading changes of greater than 90 degrees (between states and in the course of a path) are disallowed and considered infeasible.

Constraint KM3 ensures that the path length is bounded above by $l_{\max} = 2d/\sin(\pi/4)$, where d is the euclidean distance between platform positions and $l_{\max}$ is the path length. This can be used to efficiently calculate the nearest neighbours of a node in terms of path length. Constraint KM2 along with KM3 discourage generation of extreme heading changes in the early iterations of the RRT* algorithm. These constraints discourage the generation of paths that rapidly alternate between right-hand and left-hand turns (weaving).

We refer to the process of selecting a path between two states according to the planning kinematic model as *steering*. Our use of the term is slightly different from that in the literature: in the original RRT* implementation, steering refers exclusively to the second of the four steps of the algorithm

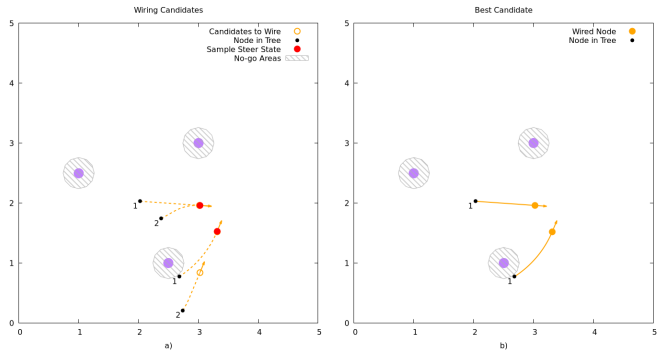


Fig. 2. The wiring step. **a)** A single sample steer state and two neighbouring nodes (numbered) for wiring. For each neighbour node, a candidate steer state (yellow) is determined toward the sampled steer state (red). The best steer state **b)** is chosen based on the tracking cost. Roughly speaking, this causes the path that brings the platforms closest to the targets, marked in purple, to be chosen.

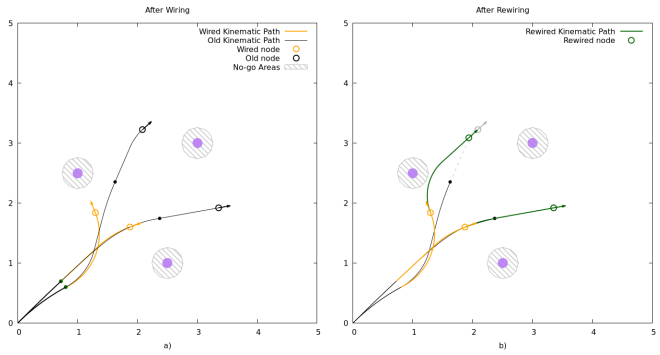


Fig. 3. Rewiring of a single 2 platform state. Two branches of the tree are plotted before **a)** and after **b)** rewiring. The rest of the tree is omitted for clarity. Each branch consists of a pair of kinematic paths. The leaf node of the first branch (yellow) is the newly wired node and the leaf node of the second branch (black) corresponds to a neighbour of the wired node. Rewiring creates a new edge (green) that replaces the existing edge (grey) to the neighbour node. The neighbour node is moved in the process of rewiring. Note, the iteration shown here is not the same as in Fig. 2.

(see Section III-A). Given the highly constrained planning kinematics defined above, it is generally not kinematically feasible to move between two arbitrary states, because each state corresponds to a particular moment in time. Since each platform moves at a constant speed, over a given time interval the lengths of the paths of different platforms must be equal. To mitigate this, the steering process attempts to move from the first state as far as possible towards the second state. We call the resulting state a *steer state*. In the steering step of the algorithm we create the *sample steer state* from the nearest node towards the random sample. Examples of steer states for two different two-platform scenarios are shown in Fig. 1. Note that the lengths of the kinematic paths of each platform are equal.

C. Wiring

To produce optimal paths, the RRT* algorithm uses wiring and rewiring steps that ensure that new nodes are added to the tree in a way that minimises the cost of the new node through the wiring step and the cost of existing nodes in the tree through the rewiring step. Rewiring allows existing paths in the tree to be iteratively improved as more nodes are added, the space is explored further, and new paths are established. Through the rewiring step, the RRT* algorithm provides provably close to optimal solutions [7].

In both the wiring and rewiring steps, neighbours within a radius of the sample steer state η are determined. In the wiring step, a *candidate steer state* is determined from each neighbour by steering toward the current sample steer state generated in the previous step. The cost of the candidate steer states are calculated from the cost model and the candidate steer state with the lowest cost is added to the tree. This second round of steering is not part of the typical RRT* algorithm and is necessary because, for any given sample steer state, only one neighbour will have a feasible kinematic path. An example of the candidate steer states generated by the wiring step is plotted in Fig. 2.

D. Rewiring & Path Shortening

The wiring step of the algorithm is a one step greedy approach to establishing low cost paths in the RRT* tree. Consequently, the paths established by the wiring step are liable to be suboptimal. This is most evident in the route finding context where the paths established can be circuitous and indirect. Good examples of this can be found in the work of Karaman et al. [19]. To address this, the rewiring step of the algorithm is used to progressively improve the paths already established in the tree as more nodes are added.

In the rewiring step, we attempt to steer from the newly wired node to neighbouring nodes. The wired node will adopt a neighbouring node as a child if, by doing so, it reduces the cost of the neighbouring node. In this way, suboptimal paths are replaced by closer to optimal paths, as illustrated in Fig. 3. For our problem, the cost model promotes paths that improve tracking performance, so that rewiring, for us, tends to choose paths that move closer to targets.

The platforms must move at a constant speed and can only make turns up to a maximum turn rate. Since each node in the tree is assumed to occur at one particular point in time, the kinematic path from the root node to any node in the tree must be equal in length for each platform. That is, each platform must reach a given node at the same time. An unfortunate consequence of this is that neighbouring nodes are generally not reachable from the newly wired node.

To alleviate this, we employ a “path shortening” strategy, where the positions of states are adjusted during the rewiring stage. Each node in the rewired subtree is moved up its kinematic path (towards the root) so that the lengths of the kinematic paths of the platforms remain equal after rewiring. An example is shown Fig. 3, where the newly wired node provides a closer to optimal path for a neighbour node that is already in the tree. The path to the node is changed to pass through the newly wired node and bring the platform closer

to one of the targets, thus improving tracking performance. Note that the kinematic path for one of the platforms has been shortened to ensure both platforms reach the node at the same time.

IV. Tracking Cost

In this section, we define a model for a tracking cost that guides the construction of the RRT* tree. Each platform attempts to improve the centralised tracks by gaining observations of the targets in the scene. Each attempted observation has a probability of a successful detection, following which the centralised track can be updated using the resulting noisy measurements. The potential improvement of a given track when a measurement is taken is determined by several factors:

- The current uncertainty in the track;
- The probability of detection;
- The uncertainty in measurements.

Current uncertainty in the track is dependent on the path to the current state, whilst the probability of detection and the uncertainty in measurements are determined by the predicted relative positions of the platforms and targets at the points of measurement.

A. Measurement Model

We assume a range-bearing measurement model. Because of the nonlinear nature of such measurements, we opt for an extended Kalman filter (EKF) for the tracking algorithm. The Jacobians for the measurement and target transition functions are:

$$\boldsymbol{H}_k(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{x}_k^p) = \nabla \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{x}_k^p)|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_{k+1|k}} \tag{3}$$

$$\boldsymbol{F}_\Delta = \nabla \boldsymbol{f}(\boldsymbol{x}_k)|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_{k|k}} = \left(\begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \otimes \boldsymbol{I}_2 \right), \tag{4}$$

where ∇ represents differentiation with respect to the state $\boldsymbol{x}_k^j$.

B. Sensor Model

For simplicity, the sensor is modelled as a range-bearing sensor with a limited forward-looking field of view (FOV) and a range-dependent probability of detection. Targets outside a cone of half-angle β around the boresight direction of the sensor are considered outside the sensor FOV. Within that FOV, the probability of detection is determined by a constant false alarm rate (CFAR) model. Specifically, given a target position at polar coordinates (r, θ) in the frame of reference of the platform, the probability of detection of a target, $P_d(r, \theta)$ at range r and bearing θ, is:

$$P_d(r,\theta) = \begin{cases} \frac{1}{2}\text{erfc}\left(\text{erfc}^{-1}(2P_{\text{FA}}) - \sqrt{\chi(r)}\right) & \text{if } |\theta| < \beta \\ 0 & \text{otherwise, ,} \end{cases} \tag{5}$$

where $\text{erfc}(\cdot)$ is the complimentary error function, P_{FA} is the probability of false alarm, and $\chi(r)$ is the range dependent signal-to-noise ratio. In the case of an active sensor, such as a radar,

$$\chi(r) = \frac{A}{r^4}\,, \tag{6}$$

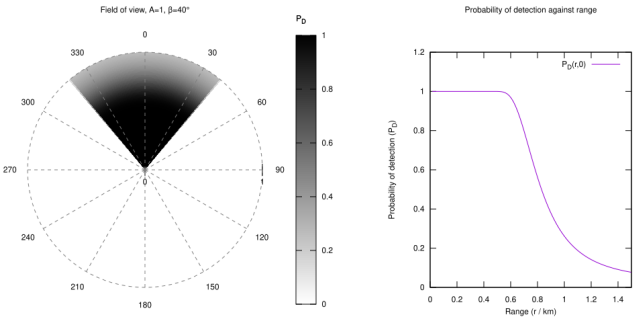


Fig. 4. The probability of detection and FOV of a range-bearing sensor with $\beta = 40°$ and $A = 1$.

where A is the aggregate of the various terms of the radar equation. A typical plot of probability of detection given by this model is shown in Fig. 4.

C. Tracking Cost

We consider a *tracking cost* that promotes the building of paths to provide optimal tracking of targets. For each node in the tree, the expected lower bound on the track covariance for each target is calculated in the form of the Posterior Cramér-Rao Bound (PCRB). Let $\tilde{\mathbf{P}}_{i,j}$ be the PCRB of the j^{th} target track at node $\mathbf{s}_i$ in the RRT* tree, then the tracking cost is expressed as the summed traces of these PCRBs:

$$\mathcal{J}(\mathbf{s}_i) = \sum_{j=1}^{J} \text{tr}\Big(\tilde{\mathbf{P}}_{i,j}\Big). \tag{7}$$

Currently, this cost function promotes behaviours in which the platforms attempt to minimise the uncertainty in all tracks equally. A reduction in the uncertainty of one track is of the same importance to an equal reduction in the uncertainty of another. A weighted version of this would prioritize some targets over others.

The PCRB, $\tilde{\mathbf{P}}_{i,j}$, of the track for target j at a given node s_i in the tree is calculated according to the points along the kinematic path to s_i at which measurements are attempted. An example of the measurement points between a parent and child node is plotted in Fig. 5. The PCRB immediately following the n^{th} measurement, $\tilde{\mathbf{P}}_{n,j}$, based on the measurement model and the PCRB of the preceding measurement point, $\tilde{\mathbf{P}}_{n-1,j}$ is

$$\tilde{\mathbf{P}}_{n,j} = \Big[\mathbf{J}_p(n) + P_d(r,\theta) \cdot \mathbf{J}_m(r,\theta)\Big]^{-1}, \tag{8}$$

where $\mathbf{J}_p(n)$ and $\mathbf{J}_m(r,\theta)$ are the prior information and predicted measurement matrices respectively, and $P_d(r,\theta)$ is the probability of detection. Both the measurement matrix and probability of detection are parametrised in terms of the predicted position of the target in the frame of reference of the platform when the measurement is made. In particular, because the tracking algorithm is an EKF we obtain,

$$\mathbf{J}_m(r,\theta) = \mathbf{H}_{r,\theta}^{\text{T}} \mathbf{R}^{-1} \mathbf{H}_{r,\theta} \tag{9}$$

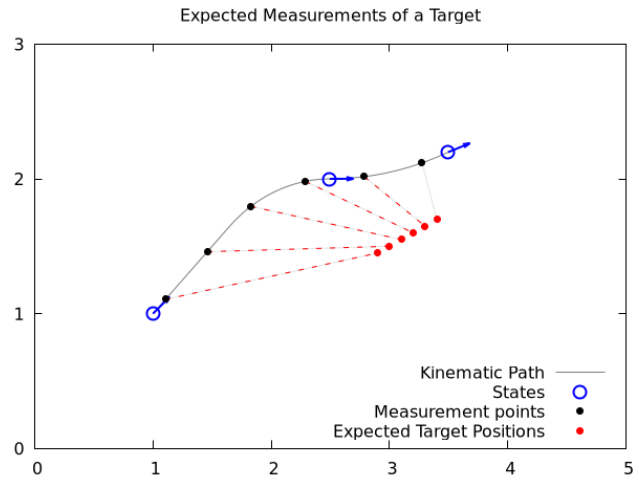


Fig. 5. Measurement points (period of $\Delta = 0.5$) on a kinematic path between three states and the corresponding expected position of the target at those points in time. Each planned measurement expected to be within the FOV of the platform is coloured in red. The final planned measurement is outside the FOV of the platform. As shown in the diagram, planned measurements do not necessarily coincide with states but are evenly spaced along the kinematic path.

where $\mathbf{H}_{r,\theta}$ is the measurement model Jacobian and $\mathbf{R}$ is the measurement model noise [16], [20]. The prior information matrix from the previous measurement time is

$$\mathbf{J}_p(n) = \left[\mathbf{F}_\Delta \tilde{\mathbf{P}}_{n-1,j} \mathbf{F}_\Delta^{\mathrm{T}} + \mathbf{Q}_\Delta\right]^{-1}, \tag{10}$$

where $\mathbf{F}_\Delta$ is the transition matrix and $\mathbf{Q}_\Delta$ is the transition model noise for a time step equal to the update period of the sensor, Δ.

Nodes in the RRT* tree do not necessarily coincide with measurement points, so that, to calculate the PCRB at a given node, we apply a final step of the transition model,

$$\tilde{\mathbf{P}}_{n,j} = \left[\mathbf{F}_\delta \tilde{\mathbf{P}}_{n-1,j} \mathbf{F}_\delta^{T}\right]^{-1}, \tag{11}$$

where δ is the time between the final measurement point and the end state.

V. Simulation Results

In this section, we present an analysis of our proposed algorithm when path planning for two platforms to track three targets. The purpose of these simulations is to test the rationality of the algorithm when given accurate yet uncertain information about the target state. Specifically, before starting their task, the platforms are given initial knowledge of the targets in the scene, and the scheduler needs to design their trajectories in order to reduce the uncertainty in the target tracks when the platforms are in motion. This initial knowledge takes the form of an accurate mean state of a target but with a high associated uncertainty.

Paths are first generated, using the proposed algorithm, for each platform. To test its effectiveness, the platforms proceed to follow the generated paths, attempting to measure the targets periodically. At each measurement point, the platforms have

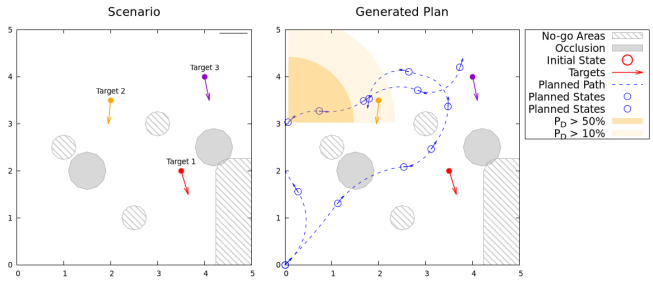


Fig. 6. A scenario where 2 platforms plan to track 3 targets. **Left:** The scenario setup including occluding and obstructing obstacles, the initial platform states, and initial ground truth target states. **Right:** The generated plan after 1500 iterations of the proposed algorithm. For reference, the field of view of one platform is plotted for a single platform and time instance.

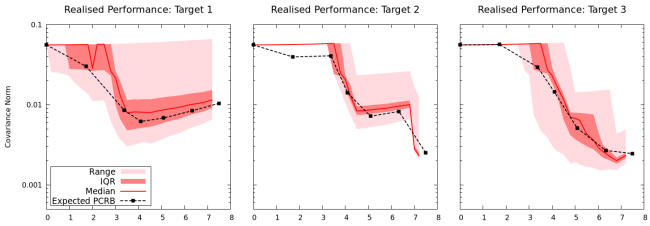


Fig. 7. Expected and realised performance for three target tracks. The median, inter quartile range (IQR), and range are calculated from 2000 realisations of the planned path.

a probability of making a measurement of the target based on the probability of detection model described in Section IV-B. Crucially, after successful detections, the platforms do not regenerate the plan. The simulation of the measurements taken and the implementation of the EKF tracking algorithm uses the Stone Soup Python library [21]–[23].

An example scenario with two platforms and three targets is illustrated in Fig. 6 along with a plan generated after 1500 iterations of the proposed algorithm. The scenario comprises three moving targets within a complex scene of occlusions and no-go regions, which impedes the motion of the platforms. Each platform has a forward-looking bearing-range sensor with high uncertainty in bearing.

The cost optimised by the platforms is a linear combination of the tracking cost defined in IV and a large fixed penalty when a platform is closer than 0.5 to a target position. This is to disincentivise moving too close to a target.

The generated plan demonstrates cooperative behaviour: the platforms approach the targets from multiple complementary perspectives whilst simultaneously navigating between no-go regions and the expected paths of the targets. At first, the two platforms separate. One platform heads directly towards the centre of the three targets while the other loops around to approach from the top. This results in complementary detections from perpendicular angles on targets 2 and 3.

To validate this plan we simulated the platforms following the generated paths over 2000 realisations. Each realisation used a unique random seed for the target transition model noise, measurement model noise, and detection attempts. The performance of the plan in terms of the track covariance norm

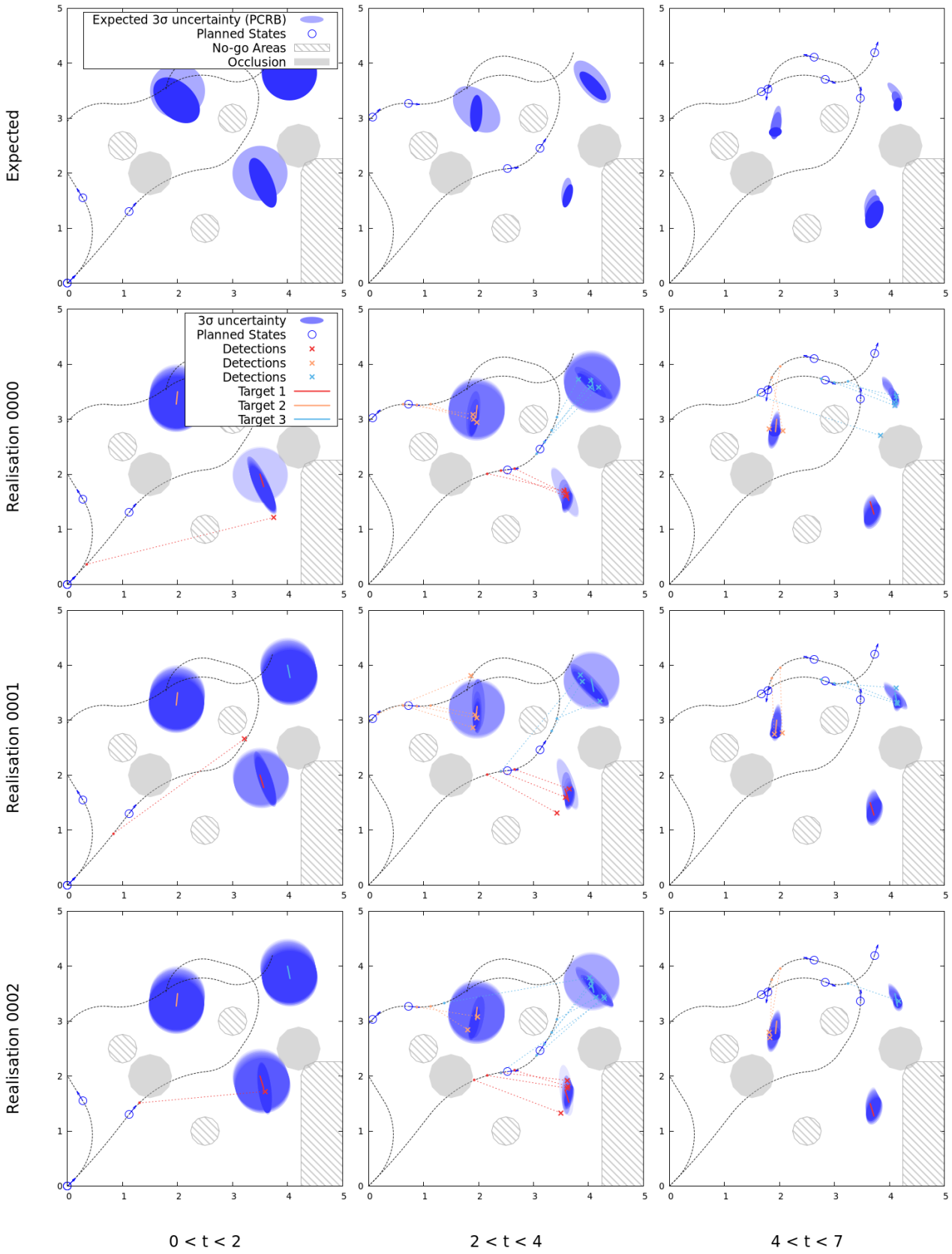


Fig. 8. The expected tracking performance and three realisations of the tracking performance over three time intervals, $t \in (0, 3)$, $t \in (3, 7)$, and $t \in (7, 10)$. The 3σ regions of the target tracks are plotted as blue areas while the successful detection are plotted as dashed lines ending in crosses. The grey dashed line indicates the planned paths of the platforms.

is plotted in Fig. 7 alongside the expected performance in the form of per-target PCRBs. The planned path successfully improves tracking performance across all targets. Moreover, the realised tracking performance closely matches the expected tracking performance predicted by the PCRB.

The expected plan and 3 realisations are graphically depicted in Fig. 8 over 3 time intervals. In each case, the 3σ region of the track is plotted. As can be seen from the plots, the tracking performance in the realisations matches closely with the expected performance in terms of the sizes of the 3σ regions, as well as in their shape and orientation.

Clearly, the proposed tracking cost estimates tracking performance well, though, in some cases, it is overly optimistic. This is most obvious in the performance of the tracks early in the plan (at long range) where better performance is predicted than is realised.

VI. Conclusion

We have presented a modified RRT* algorithm for the purpose of coordinating multiple platforms in the task of tracking multiple targets under highly constraining kinematics. To promote good tracking performance, we have used a tracking cost based on the PCRB. We have demonstrated through simulations that it accurately predicts the tracking performance of plans generated by the proposed algorithm.

We have also shown, again through simulations, that the planning algorithm is capable of producing rational paths across multiple platforms in a cooperative way to improve tracking performance. Specifically, platforms take different but complementary paths that, when possible, converge on targets from complementary directions.

While more work is needed, in particular, on relative timing of platform actions, we believe that we have demonstrated a potentially useful approach to cooperative multi-platform sensor scheduling.

VII. Acknowledgment

The Monte Carlo simulations described in this paper were performed using the University of Birmingham's BlueBEAR High Performance Computing service. See http://www.birmingham.ac.uk/bear for more details.

References

[1] J. Dai, W. Pu, J. Yan, Q. Shi, and H. Liu, "Multi-UAV collaborative trajectory optimization for asynchronous 3-D passive multitarget tracking," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[2] G. Ferri, A. Munafò, A. Tesei, P. Braca, F. Meyer, K. Pelekanakis, R. Petroccia, J. Alves, C. Strode, and K. LePage, "Cooperative robotic networks for underwater surveillance: an overview," *IET Radar, Sonar & Navigation*, vol. 11, no. 12, pp. 1740–1761, 2017.

[3] A. Hero and D. Cochran, "Sensor management: Past, present, and future," *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3064–3075, 2011.

[4] P. Braca, R. Goldhahn, K. D. LePage, S. Marano, V. Matta, and P. Willett, "Cognitive multistatic AUV networks," in *17th International Conference on Information Fusion (FUSION)*, July 2014, pp. 1–7.

[5] S. Layer-Reiss, A. Zwenig, C. Kalender, and D. Braun, "Reinforcement learning based guidance policy optimization for a swarm of cooperative sensor platforms tracking a single target," in *2023 26th International Conference on Information Fusion (FUSION)*, June 2023, pp. 1–8.

[6] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, Feb 2023.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[8] X. Wang, S. Williams, D. Angley, C. Gilliam, T. Jackson, R. Ellem, A. Bessell, and B. Moran, "RRT* trajectory scheduling using angles-only measurements for AUV recovery," in *22nd International Conference on Information Fusion (FUSION)*, Ottawa, ON, Canada, Jul. 2019, pp. 1–6.

[9] S. Williams, X. Wang, D. Angley, C. Gilliam, B. Moran, R. Ellem, T. Jackson, and A. Bessell, "Dynamic target driven trajectory planning using RRT*," in *2019 22th International Conference on Information Fusion (FUSION)*, July 2019, pp. 1–8.

[10] S. Williams, X. Wang, D. Angley, C. Gilliam, B. Moran, T. Jackson, R. Ellem, and A. Bessell, "Smoothing optimal RRT* trajectories for recovery of an AUV by a moving surface vessel," in *Maritime Situational Awareness Workshop organised by NATO Science and Technology Organization Centre for Maritime Research and Experimentation*, Lerici (La Spezia), Italy, 2018, pp. 1–8.

[11] T. Xu, "Recent advances in Rapidly-exploring random tree: A review," *Heliyon*, vol. 10, no. 11, p. e32451, 2024.

[12] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, p. 497, 1957.

[13] D. Angley, S. Mehrkanoon, B. Moran, C. Gilliam, and S. Simakov, "Improving automated search for underwater threats using multistatic sensor fields by incorporating unconfirmed track information," in *Proc. IEEE International Conference on Autonomous Systems (ICAS)*, Aug 2021, pp. 1–5.

[14] C. Gilliam, B. Ristic, D. Angley, S. Suvorova, B. Moran, F. Fletcher, H. Gaetjens, and S. Simakov, "Scheduling of multistatic sonobuoy fields using multi-objective optimization," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, April 2018, pp. 3206–3210.

[15] C. Gilliam, D. Angley, S. Williams, B. Ristic, B. Moran, F. Fletcher, and S. Simakov, "Covariance cost functions for scheduling multistatic sonobuoy fields," in *2018 21st International Conference on Information Fusion (FUSION)*, Cambridge, UK, July 2018, pp. 1–8.

[16] P. Tichavsky, C. H. Muravchik, and A. Nehorai, "Posterior cramer-rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1386–1396, May 1998.

[17] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. Wiley, 2002.

[18] S. M. LaValle and J. J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001. [Online]. Available: https://doi.org/10.1177/02783640122067453

[19] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[20] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the Kalman filter: Particle filters for tracking applications," *Artech House*, 2004.

[21] J. Barr, O. Harrald, S. Hiscocks, N. Perree, H. Pritchett, and J. W. *et. al.*, "Stone Soup open source framework for tracking and state estimation: enhancements and applications," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXI*, L. Grewe, E. Blasch, and I. Kadar, Eds. SPIE, 2022, p. 4.

[22] S. Hiscocks, J. Barr, N. Perree, J. Wright, H. Pritchett, O. Rosoman, M. Harris, R. Gorman, S. Pike, P. Carniglia, L. Vladimirov, and B. Oakes, "Stone Soup: No longer just an appetiser," in *2023 26th International Conference on Information Fusion (FUSION)*, 2023, pp. 1–8.

[23] S. Hiscocks, O. Harrald, J. Barr, N. Perree, L. Vladimirov, and M. H. *et. al.*, "Stone Soup," 1.5V, 2024, Software available at https://zenodo.org/records/14224532.