# Dynamic Target Driven Trajectory Planning using RRT*

Simon Williams[2], Xuezhi Wang[1], Daniel Angley[2], Christopher Gilliam[1], Bill Moran[2], Richard Ellem[3], Trevor Jackson[3], Amanda Bessell[3]
[1] School of Engineering, RMIT University, Australia
[2] Electrical & Electronic Engineering, University of Melbourne, Australia
[3] Maritime Division, Defence Science and Technology Group, Australia

***Abstract*—In this paper, we focus on dynamic trajectory planning for an autonomous underwater vehicle (AUV). Specifically, we are interested in planning the trajectory of an AUV as it returns to a moving recovery vessel. To aid in this task, the AUV is equipped with a passive, angle-only, sensor to enable localization of the recovery vessel. Accordingly, we present an algorithm that is capable of dynamically updating the trajectory of the AUV given measurement data from the passive sensor. Our approach is based on adapting a static trajectory planning algorithm from robotics, known as Rapidly-exploring Random Tree (RRT*), to allow for localization and tracking of a dynamic target (i.e. the recovery vessel). In contrast to dynamic programming or fixed grid trajectory planning, the RRT* offers a computationally efficient method for long-term trajectory planning with probabilistic guarantees of optimality. In this framework, we explore two options: trajectory planning based on minimising the distance to the target; and trajectory planning based on maximising the tracking accuracy of the target using an information theoretic cost. Using AUV recovery as an evaluation scenario, we analyse and evaluate the proposed trajectory planning algorithm against traditional dynamic programming methods. In particular, we consider trajectory planning in noisy and obstructed environments.**

***Index Terms*—Trajectory optimisation, path planning, passive target tracking, RRT*, autonomous underwater vehicle recovery**

## I. Introduction

In this paper, we focus on dynamic trajectory planning for an autonomous underwater vehicle (AUV). Specifically, we are interested in planning the trajectory of an AUV as it returns to a moving recovery vessel. To aid in this task, the AUV is equipped with a passive, angle-only, sensor to enable localization of the recovery vessel. Accordingly, we present an algorithm that is capable of dynamically updating the trajectory of the AUV given measurement data from the passive sensor. Our approach is based on adapting a static trajectory planning algorithm from robotics, known as Rapidly-exploring Random Tree (RRT*), to allow for localization and tracking of a dynamic target (i.e. the recovery vessel).

This research is supported by DST Group under the research agreement "Trajectory Optimisation for a Moving Observer with 3-D Angle Only Sensor Measurements".

### A. Retrieving Autonomous Underwater Vehicles

Autonomous Underwater Vehicles (AUV) are expected to play an increasingly important role in a wide range of undersea applications such as conducting surveys to map the seabed and locate bottomed objects, sensing and characterising the undersea environment, and monitoring the movements and behaviours of surface vessels, underwater vehicles, other sub-surface objects and/or marine life. Vehicles operating in the underwater environment typically rely on sonar (i.e. underwater acoustics) as the primary method for sensing and communication over extended ranges due to the significant attenuation of electromagnetic signals. Here we consider several sonar based sensor and navigation applications for an AUV operating in cooperation with a surface vessel that supports deployment and recovery of the AUV.

The problem of path planning for a mobile surface vessel that maintains supervisory contact (using an acoustic communications link) with an AUV executing a pre-planned mission was considered in [1]. Here we consider an AUV that conducts survey missions in a more autonomous manner, and on completion of the mission or when signalled by the support vessel via the communication link, uses its on-board sensors and optimal trajectory path planning to navigate safely back towards the support vessel to rendezvous and surface within a safe standoff distance for subsequent recovery. The support vessel is assumed to be mobile and is expected to transit along a straight line trajectory at a constant relatively slow speed towards a safe region (away from other vessels) to prepare for recovery of the AUV. Signals transmitted via the

acoustic communications link can be used to assist the AUV with the detection and identification of the support vessel but are not used to control or program the trajectory or destination of the AUV. The final recovery location will not be known to the AUV or the support vessel in advance and will depend on how long it takes for the AUV to navigate to a safe recovery location.

The AUV is equipped with a forward facing high-frequency sonar sensor array that is used for sensing and navigation, which operates primarily as a passive acoustic sensor. The sensor array receives the acoustic energy emitted from all nearby vessels or objects located within the sensor's field of view, defined by the angular sector containing bearing and elevation angles within $\pm 60°$, with a finite angular beam resolution of $\pm 3.5°$. An on-board sonar processing system determines the direction of arrival of the received acoustic signals and uses this to estimate and track the positions and velocities of all observable signal sources over time. The estimated state information associated with each track is then used for optimal trajectory path planning to enable the AUV to safely navigate towards the recovery vessel in the shortest amount of time while avoiding any other vessels or objects in the area.

The presence of other vessels or objects will impose sensor measurement and physical space constraints within the optimal trajectory path planning algorithms. Sensor measurement constraints result from reduced detectability of the support vessel caused by interference from other noise sources within the same angular sector or increases in overall received noise levels from noise sources in close proximity to the sensor. Physical space constraints result from maintaining minimum safe separation from other vessels or objects, and, from specified non-navigable areas such as shallow water areas, shorelines, islands, physical structures in the water, shipping channels or other restricted areas. These latter 'non-navigable' physical constraints are implemented as pre-programmed geographic regions within the on-board processing system of the AUV.

### *B. Randomly Exploring Random Trees*

Figure 1 shows a scenario where an AUV needs to navigate a fleet of interfering sources using its passive sonar to rendezvous with a moving recovery vessel.

Previous work [1], [2] was based on $n-$step lookahead on the tracker processing. The new algorithm is based on RRT* developed in the robotics literature [3], [4] for motion planning. The probabilistic nature allows soft constraints to be implemented as required.

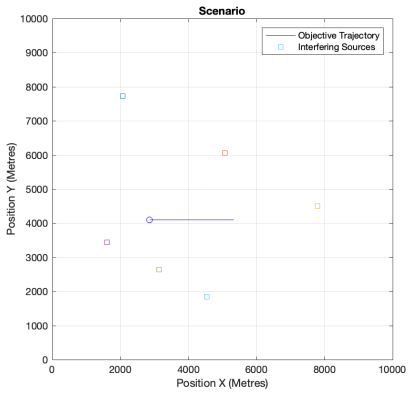



Fig. 1: Example scenario for an AUV to navigate. The path of the moving objective is indicated by the blue line and the squares indicate the position of the interfering sources. Note that the AUV is initialised at the origin.

**function** RRT*($o, g, \eta$, N)
  $t \leftarrow \{o\}$
  $n \leftarrow 0$
  **while** $n < N$ **do**
    $x \leftarrow$SAMPLE
    $x' \leftarrow$STEER$(t, x)$
    $t \leftarrow t + \{x'\}$
    $t \leftarrow$REWIRE(t)
    $n \leftarrow n + 1$
  **end while**
  **return** $t$
**end function**

Fig. 2: Original RRT* algorithm

The RRT* algorithm is provably optimal for a fixed objective and arbitrary allowable manoeuvres at each time step, which does not hold in our case. Figure 2 shows the original RRT* algorithm. It begins with picking a point at random in the region of interest (SAMPLE). Then you move from the nearest point in the tree towards that point as far as is feasible (STEER) usually by giving a maximum allowable distance travelled $\eta$. Limits on feasibility are placed by speed restrictions and obstructions in the region. The cost of this step is then added to the cost recorded at the originating node. But, before this cost is encoded in the tree, a search is conducted locally to see if there is an alternative route to the new point that is lower cost (REWIRE). One of the keys to the RRT* algorithm is that this search need only proceed *locally*, within a distance

$r$ given by

$$r = \min\left\{\left(\frac{\gamma \log n}{\xi_d n}\right)^{1/d}, \eta\right\} \quad (1)$$

where $d$ is the dimension of the space, $n$ is the number of sample points, $\gamma$ is a constant and $\xi_d$ is the volume of the unit ball in $\mathbb{R}^d$ Figure 3

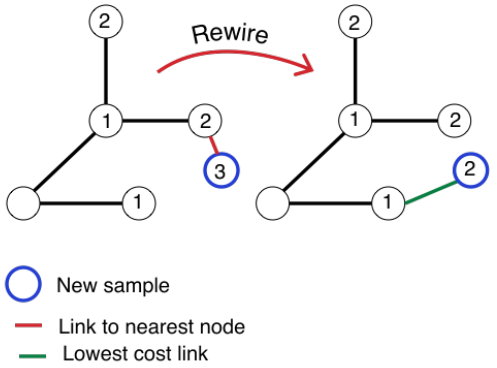



Fig. 3: Diagram of rewire step in RRT* algorithm. The new sample (blue) is linked in to the tree at its nearest neigbour (red). A search, that only extends a maximum of one link length around the sample, reveals a lower cost link. The old edge is deleted and the new edge created (green).

shows how, if a new lower cost link is possible then, the original link is deleted and the alternative instantiated and the new (lower) cost encoded at the new node.

This algorithm is probabilistically complete and asymptotically optimal for finding a path to a goal through an unobstructed region. Specifically the probability of the algorithm converging to the optimal path increases to 1 as the number of samples $N$ increases.

### C. Plan of the Paper

In Section II we detail the addition we have made to the RRT* algorithm to allow its use in our scenario. Section III describes our RRT*-based Trajectory Planning algorithm. Section IV discusses the results obtained.

## II. Motion Planning for Trajectory Optimisation

In this section we shall describe our extensions to the RRT* algorithm that address the use of mobile objectives, dynamic cost functions, kinematic constraints, focused sampling and track estimation accuracy costs.

**function** COST($x_i, l_i, s$)
  $c \leftarrow \eta$
  **if** Angle($x_i, s$) - Angle($x_i, l_i$) $\leq 3.5°$ **then**
    $c \leftarrow c + BW$
  **else if** Angle($x_i - x_{i-1}, l_i$) $\geq 60°$ **then**
    $c \leftarrow c + FOV$
  **end if**
  **return** $c$
**end function**

Fig. 4: Dynamic Cost function: $BW$ is the penalty for an interfeering source lining up with the target and FOV is the penalty for the target leaving the AUV sensor's field of view

### A. Mobile Objective and Dynamic Costs

In order to incorporate the motion of the objective into the algorithm we take advantage of the fact that the AUV uses a tracker to keep an estimate of the position and velocities of its recovery vessel and the other vessels in its field of view. Using this information and the fact that each step in the tree corresponds to an elapsed time, we can use an updated position of the objective when calculating in the tree.

For example, if our new point $x$ is attached to the tree at a depth $k$ from the origin, then the cost function can be evaluated using the new objective location $l_i = l_0 + kv$ where $l_0$ is the original location and $v$ is the velocity estimate in units of tree steps $\eta$.

This enables the implementation of dynamic cost functions where we penalise the alignment of an interfering source and the objective within the same received beamwidth ($\pm 3.5°$), and maintain the objective within the sensor's field of view ($\pm 60°$). This avoids the loss of detections that would result from the recovery vessel being obscured by interfering noise sources associated with other nearby vessels, or located outside the sensor's field of view, which would impact the ability of the tracker to provide accurate target state estimates. The form of these dynamic cost functions is shown in Figure 4, where $x_i$ is the location of the new node which is $i$ steps from the root, $l_i$ is the expected location of the objective after $i$ steps and $s$ are the locations of the interfering sources. The values for the penalties were set at $BW = 50$ and $FOV = 25$

### B. Kinematic Constraints

The RRT* algorithm [3] has been modified to incorporate motion constraints for the AUV by limiting the maximum change in direction at each time step to 10°. This alters the STEER procedure so that the vector joining the new point $x'$ to the

tree always lies within 10°of the previous leg of the tree.

Figure 5 is the result of our modified RRT* algorithm. 1000 samples were taken from a uniform distribution across the region which is a 6x6x6 cube centred at (0,0,0). The AUV starts at (-3,-3,-3) pointing directly at the recovery vessel and has a maximum speed of $\eta = 0.2$ per unit time. The recovery vessel starts at (2,2,-2) moving in direction (0,0,1) at a constant speed of 0.09 per unit time.

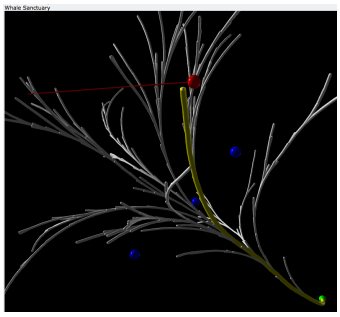



Fig. 5: An example RRT tree with the ship and its motion marked in red, interfering sources in blue and the AUV's initial position in green and optimal path in yellow.

### C. Using a spatial index to speed tree searches

A costly step in the RRT* algorithm is finding the nearest node in the tree to the randomly sampled point, which is $O(n^2)$. The usual K-d and balanced-box trees will not work in this situation as we need to build the index incrementally as the tree is constructed. Fortunately there is an incremental spatial index, the R-Tree [5] with a C++ implementation `libspatialindex` [6] and a python wrapper `rtree` [7].

Table I shows expurgated views of the a profile for the RRT* algorithm run for 4 steps each of 1000 samples comparing raw tree search performance to the spatial index, where `extend` is the main loop with `nearest` and `near` the nearest neighbour functions for both implementations. Finally, the `insert` entry shows the cost of building the spatial index.

| ncalls | time | filename:lineno(function) |
|---|---|---|
| Tree search | | |
| 4000 | 55.758 | rrtstar.py:92(`extend`) |
| 4020 | 24.965 | rrtstar.py:49(`nearest`) |
| 4000 | 28.599 | rrtstar.py:69(`near`) |
| Spatial index | | |
| 4000 | 6.914 | rrtstar.py:112(`extend`) |
| 4020 | 3.388 | rrtstar.py:61(`nearest`) |
| 4000 | 1.067 | rrtstar.py:84(`near`) |
| 4011 | 0.848 | rtree.py:341(`insert`) |

TABLE I: Comparison of the tree search and spatial index methods for nearest neighbour calculation

### D. Trackers and Focussed Sampling

We model the measurements made by the AUV's sensor using a simplified sonar equation with a simplified uniform propagation model and a discrete beam pattern for each of the sensor received beam components.

The AUV's tracker is a Sequential Monte-Carlo Multi-Hypothesis Tracker (SMC-MHT) described in [1], [8]. It is a multiple hypothesis tracker which detects and estimates the underlying target state with unknown measurement origin by enumerating and evaluating all possible measurement origin hypotheses. The sequential Monte Carlo sampling technique is used to approximate the posterior probabilities of the association hypotheses and make the approach computationally tractable.

It is important to be able to separate out the effect of the tracker on the trajectory optimisation algorithm. We shall see one approach by processing the data in the next section. Our approach is using what we have termed a 'Ground Truth' tracker, which outputs the actual position of the target with a simulated diagonal covariance an order of magnitude smaller than those predicted by the actual trackers. This enables repeatable runs with out interference from the random nature of the tracking algorithms.

Usually the SAMPLE procedure uses a uniform distribution over the area to be covered. Here we use the position and covariance estimates from the tracker and the parameters of a multivariate normal distribution and sample the space focussed around the objective location.

### E. Accuracy Cost

An accuracy cost estimate can be defined after [1] as the expectation of the penalty functions for field-of-view and beamwidth obstructions

$$J_a = \det P \, \mathbb{E}_\pi \left[ FOV(x) BW(x) \right] \qquad (2)$$

where $P$ is the posterior covariance of AUV position estimate and $\pi$ is the prior distribution of the objective track estimate. $FOV$ is the field of view penalty and BW(x) is the Beamwidth penalty from the dynamic cost function in Figure 4 for the objective at position $x$. In order to satisfy the completeness and optimality criteria for RRT* (additivity and continuity) we use $\log J_a$ as the cost function.

An example of the resulting paths is shown in Figure 6. The loop and the slalom-like final chase to the target shows the accuracy cost is having

the correct effect on the bearing-only sensor by requiring changes in relative bearing to increase the information content of measurements.

That the loop is possible is due to the scale of the accuracy measure and the requirements of the RRT* algorithm. The RRT* algorithm requires that the cost function be additive, while the determinant of the covariance updates multiplicatively. In order to accomodate this detail we take the log of the accuracy. This rescaling of the accuracy reduces the effect of the FOV penalty so that it is essentially moot. This is another argument for developing an information-based criterion that does not rely on arbitrary parameters to enforce its requirements on the planned trajectory.

The results of comparing the original implementation using the 2 step lookahead algorithm with the newly integrated RRT* algorithm are shown in Figures 11– 15

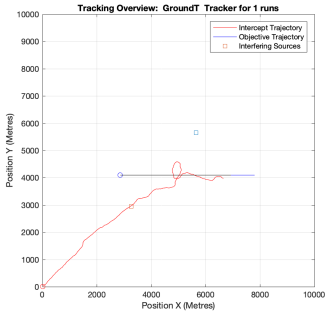



Fig. 6: A single RRT* optimal trajectory created by optimising the accuracy cost. The AUV is initialised at the origin.

## III. Implementing RRT*-based Trajectory Optimisation

The tree shown in Figure 5 is one step in our trajectory optimisation algorithm. Ultimately only the first step of the yellow path will be used to initialise another run of the RRT* motion planner. The full trajectory optimisation algorithm is described in Figure 7. This involves generating an RRT* tree at each step and using the first edge of the tree that ends closest to the target, where the FirstStep$(t, i)$ returns the first step in the branch of tree $t$ at depth $i$ which is closest to the goal. This is calculated by moving in the direction of the first leg of the optimal path for a distance corresponding to the speed of the AUV.

### A. Goal oriented stopping criteria

In the original RRT* implementation the cost minimised the distance to the objective. However, a more realistic criteria for a moving observer is

**function** OptimiseTrajectory$(length, samples)$
  $path \leftarrow [InitialNode]$
  **for** $i$ in 1..length **do**
    $tree \leftarrow$RRT*$(samples)$
    $path \leftarrow path +$ FirstStep$(tree, length - i)$
  **end for**
  **return** $path$
**end function**

Fig. 7: RRT*-based Trajectory Optimisation

to use its passive sensor to approach the target, and then to terminate the approach or switch to other close-range sensing methods to avoid complex nonlinear near-field estimation effects associated with angles-only tracking problems. Accordingly the stopping criteria used here was changed to closing within 300m of the target and the cost function adjusted to minimise distance from the target rather than distance travelled.

This is a fundamental distinction between our approach and the other greedy approaches, which are limited in the number of steps ahead they look by the exponential growth of the number of possibilities to consider. Instead, we can set goal-oriented stopping criteria for the trajectory optimisation, like the one detailed above: Stop when you are within 300m of the objective.

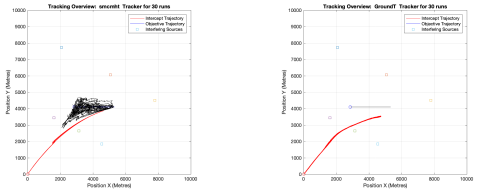
## IV. Results and Discussion

In order to fully exercise the trackers and the trajectory optimisation algorithms comprehensively every combination of tracker, scenario and algorithm was run 30 times.

### A. Distance Cost

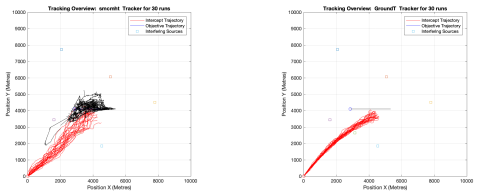
Figures 8–10 show the results for those simulations using the distance cost. For each scenario we plot the raw tracks and resulting trajectories for each tracker and algorithm, then a density map of relative trajectories to isolate the effect of the tracker and illustrate the properties of the trajectories produced by the respective algorithms. Finally, we assess the performance of the competing algorithms by calculating the final distance to the target for each algorithm and comparing the two.

Figure 8 shows the 30 Trajectories optimised using 2-step lookahead trajectory planning for the scenario in Figure 1. This has 6 interfering sources (3 in the near-field, 3 in the far) with the target tracing a path between the two sets of interfering sources. We can see the strong dependence of the



(a) SMC-MHT Tracker (b) Ground Truth

Fig. 8: 30 Trajectories optimised using 2-step lookahead trajectory planning for the scenario in Figure 1 using the distance cost. The AUV is initialised at the origin.

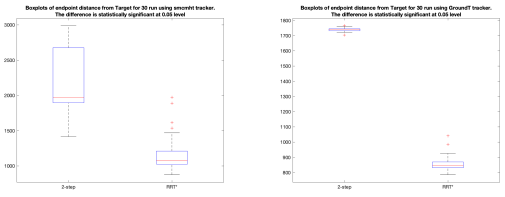


(a) SMC-MHT Tracker (b) Ground Truth

Fig. 9: 30 Trajectories optimised using RRT* trajectory planning for the scenario in Figure 1 using the distance cost. The AUV is initialised at the origin.

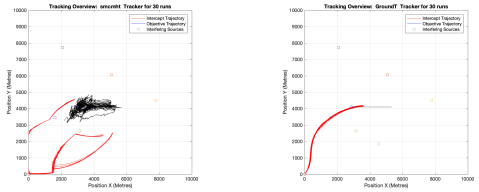
trajectory on the track by comparing each of the trackers to the Ground truth tracker.

The fan structures in the optimised trajectories are most likely caused by inaccurate predictions of the target location interacting with the middle interfering source in the far field. The n-step lookahead algorithm has hard constraints that forbid a co-linear relationship of the target, observer, and interfering source. So an incorrect target location can cause wild trajectories. In the end all the trajectories seem to converge to a standard approach to the target. In Figure 10 we see that the RRT*-based algorithm produces trajectories that close much quicker with the object than the two step look ahead algorithm. This is due to the soft constraints and the longer lookahead horizon.



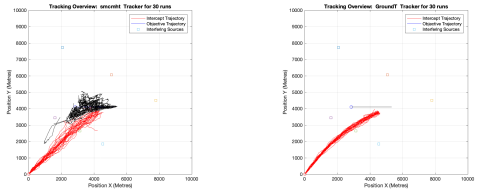
(a) SMC-MHT Tracker (b) Ground Truth

Fig. 10: Distance to objective after 10 minutes for the scenario in Figure 1 using distance cost.



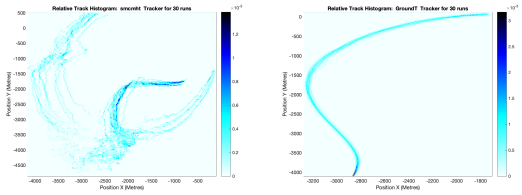
(a) SMC-MHT Tracker (b) Ground Truth

Fig. 11: 30 Trajectories optimised using 2-step lookahead trajectory planning for the scenario in Figure 1 using Accuracy Cost. The AUV is initialised at the origin.



(a) SMC-MHT Tracker (b) Ground Truth

Fig. 12: 30 Trajectories optimised using RRT* trajectory planning for the scenario in Figure 1. The AUV is initialised at the origin.

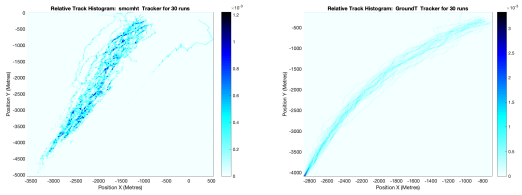


(a) SMC-MHT Tracker (b) Ground Truth

Fig. 13: Probability of relative observer-target track using 2-step lookahead trajectory planning for the scenario in Figure 1.

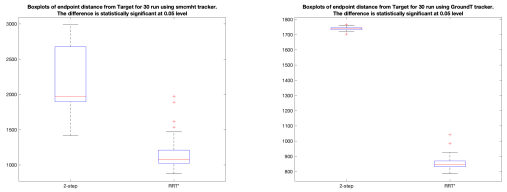
### B. *Accuracy Cost Results*

The main difference shown in Figures 11 – 14 is that even though no distance optimisation has been performed, the trajectories still move towards the objective. This is due to focused sampling where the new additions to the tree are sampled from the distribution of the objective estimate produced by the AUV's tracker.



(a) SMC-MHT Tracker (b) Ground Truth

Fig. 14: Probability of relative observer-target track using RRT* trajectory planning for the scenario in Figure 1.



(a) SMC-MHT Tracker (b) Ground Truth

Fig. 15: Distance to objective after 10 minutes for scenario in Figure 1 using accuracy cost

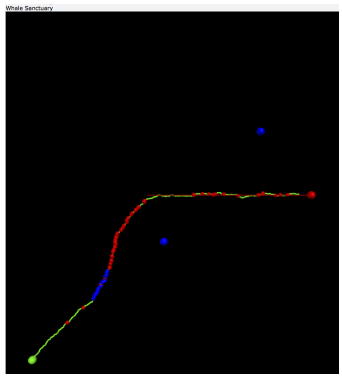


Fig. 16: Optimal path for 1km exclusion zone. Blue indicates points on the path that the beamwidth constraint is violated and red indicates the same for the field-of-view constraint.

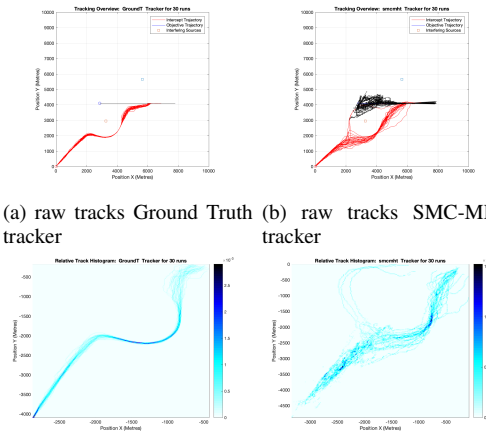
Also present is the same lack of smoothness in the RRT* paths that increases their overall length while a qualitative viewing of the paths in Figure 12 shows they should match or even exceed the performance of the $n$-step lookahead algorithm.

Figure 15 shows the distance to the objective after 10 minutes of simulation for both the 2-step lookahead and the RRT* algorithms, showing the latter outperforming the former. In both cases the difference in the means is statistically significant for $\alpha = 0.05$ but the difference is much greater for the Ground Truth tracker which shows it is the errors in the tracker that cause the variation in the distance to the objective. The larger variation is due to the roughness of the RRT* algorithm.

### C. Exclusion around interfering sources

As a first step towards an improved measurement model, we implemented a 1km exclusion zone around each of the interfering sources. This was implemented by discarding any of the randomly sampled points from the SAMPLE function in the RRT* algorithm of Figure 2 that fell inside the exclusion zone. Figure 16 shows a single trajectory for the target distance cost.

Figure 17 shows the results for 30 runs using both the Ground truth and SMC-MHT trackers and



(a) raw tracks Ground Truth tracker (b) raw tracks SMC-MHT tracker

(c) relative track histogram Ground Truth tracker (d) relative track histogram SMC-MHT tracker

Fig. 17: Simulations with 1km exclusion zone 30 runs. The AUV is initialised at the origin.

the relative trajectory probability distributions. The RRT* paths are clearly avoiding the interfering source and choosing the minimum distance path around the interfering source. In Figure 17(b) it appears that when the raw track is further away then it is best to go clockwise round the interfering source rather than counter-clockwise as the vast majority of the paths do. Further work will be done to understand this behaviour.

## V. CONCLUSIONS AND FURTHER WORK

We have extended the original RRT* model into a fully fledged optimisation algorithm capable of handling moving objectives, focussed sampling, goal centered lookahead and accuracy maximisation and shown that it is more flexible, computationally simpler and better performing than the corresponding tracker based greedy 2-step lookahead algorithm.

Further work is need to address some shortcomings in our existing algorithm and simulation implementations. A more accurate sensor model would allow us to more accurately represent the effect of the interfering sources on the measurements made by sensor and thus provide optimal trajectories that more closely reflect the real-world sitution. The arbitrary penalties hard coded in Figure 4 will be replaced with an information criterion that takes into account the effect that the missed sensor measurements will have on the accuracy of the objective position estimate from the tracker. Smoothing of the RRT*-based trajectories would be useful and we are currently incorporating the Dubins model as a method for smoothing the kinematics of the paths produced [9], [10].

## REFERENCES


[1] M. Morelande and R. Ellem, "Trajectory optimisation for 3D angle-only tracking (milestone a)," University of Melbourne, Tech. Rep., May 2014.

[2] ——, "Trajectory optimisation for 3D angle-only tracking (milestone b)," University of Melbourne, Tech. Rep., Sep. 2014.

[3] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, p. 2, 2010.

[4] ——, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] A. Guttman, "Rtree: Spatial indexing for Python," in *Conference on Management of Data*, ACM-SIGMOD, 1984, pp. 47–57.

[6] M. Hadjieleftheriou. (2014). Libspatialindex. version 1.8.5, [Online]. Available: http : / / libspatialindex.github.io.

[7] S. Gillies, Ed. (2011). Rtree: Spatial indexing for python. version 0.7.0, [Online]. Available: http://toblerity.org/rtree/.

[8] D. B. Reid, "An algorithm for tracking multiple targets.," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[9] L. Dubins, "On the curves of minimal length on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[10] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control*, 2010.