A Monte Carlo Tree Search Framework for Autonomous Source Term Estimation in Stone Soup

Timothy J Glover, Rohit V. Nanavati,
Matthew Coombes, Cunjia Liu and Wen-Hua Chen
Department of Aeronautical and Automotive Engineering,
Loughborough University, LE11 3TU, UK
Email: {*t.glover, r.nanavati, m.j.coombes, c.liu5, w.chen*}*@lboro.ac.uk*

Nicola Perree and Steven Hiscocks
Cyber and Information Systems,
Defence Science and Technology Laboratory,
Salisbury, UK
Email: *stonesoup@dstl.gov.uk*

***Abstract*—Source term estimation of a hazardous release remains a topic of significant interest in the robotics and state estimation communities, with application to many safety critical scenarios including gas or nuclear release, locating suspicious smells or response to emergency incidents. Limited sensing resources and time constraints mean that deciding on how to act in order to improve efficiency of estimation is also of significant interest. This paper has two main focuses: a sequential Monte Carlo technique for performing source term estimation from gas concentration measurements taken on a mobile sensor platform and a Monte Carlo tree search (MCTS) framework to perform sensor motion planning to maximise Kullback-Leibler divergence (KLD). Both algorithms are implemented in the open source tracking and estimation framework: Stone Soup, creating several key contributions to this Python based toolkit. The presented algorithm demonstrates superior performance when compared to a greedy myopic alternative when considering source position estimation error, release rate error and successful rate performance measures.**

I. Introduction

Localising a source of gas release, estimating its emission rate or determining the presence of chemical, biological radiological and nuclear (CBRN) activity in an environment is a challenge of significant interest across multiple research communities. Source term estimation (STE) presents benefits to human safety and environmental damage mitigation when applied to scenarios including environmental pollution [1], nuclear site decommissioning [2], volcanic eruptions [3], defence [4], [5] and nuclear disasters [6], [7]. The growing capabilities of robots and sensors means that autonomous systems present the ideal solution to STE problems as they can remove humans from high risk tasks, reduce the requirement for skilled operators and improve response times and efficiency. However, an autonomous search platform must tackle two main problems: estimating the source term parameters based on measurement data and planning a suitable path to collect useful sensor measurements.

Considering the first of the two challenges, there are two main approaches to performing STE: optimisation and Bayesian [8]. In order to capture the stochastic and uncertain nature of gas release and prior information in such scenarios, this work is only concerned with the Bayesian approach. For a detailed review of optimisation based approaches, the reader is referred to [8]. Two popular approaches to Bayesian STE are Markov Chain Monte Carlo (MCMC) [9]–[11] and sequential Monte Carlo [12]–[14]. MCMC techniques construct a Markov chain that sequentially samples from a proposal distribution and accepts or rejects the new sample according to a criteria such as Metropolis-Hastings [15], [16]. SMC techniques are also sample based but the formulation allows for samples to be drawn simultaneously, making them largely more efficient than the MCMC alternative in online implementation.

Moving on to consider the planning problem of autonomous STE, coverage path planning presents the most simple approach, which is to generate maximum coverage trajectories offline [17], [18], but these can be inefficient and do not scale well in large environments. Bio-inspired guidance based on instantaneous concentration gradients, such as Chemotaxis [19], [20] and Anemotaxis [21], is a second approach that is also ideal when resources are limited. However, this work is focused on more robust information theoretic approaches, such as Infotaxis [22] and Entrotaxis [23]. These algorithms attempt to reduce uncertainty on the predicted source term distributions in conjunction with a myopic path planner. Tree based planners for information theoretic approaches have also been introduced more recently [24], [25], but this is a new area with many techniques yet to be introduced to this application.

In this work, a Monte Carlo tree search (MCTS) framework is introduced to autonomous STE for the first time. It is an established solution technique for Markov decision processes (MDPs) and partially observable MDPs (POMDPs) of which this problem can be cast into the latter. Initially introduced for competing in complex decision problems like Computer Go [26], MCTS is gaining interest in robotics related problems [27]–[29]. Our work introduces MCTS as a planner for autonomous STE, optimising over Kullback-Leibler divergence (KLD) as the reward. The Bayesian STE algorithm is also implemented to predict future source term beliefs. The algorithms in this paper are implemented in the open source target tracking framework: Stone Soup, creating a set of new features for sensor management.

The remainder of this paper is structured as follows. Sec. II introduces the problem formulation. Sec. III outlines the approach to STE. The MCTS sensor planning is introduced in Sec. IV. Contributions to Stone Soup are highlighted in Sec. V. The simulation scenario, results and discussion are then

presented in Sec. VI before concluding the paper in Sec. VII.

II. Problem Formulation

In the event of a CBR disaster, it is important that estimation of any release is performed as efficiently as possible. The problem is twofold: estimation of the chemical release source term and where to take future measurements to maximise the reward function linked to the task objective.

The platform is equipped with a gas intensity sensor providing noisy, pointwise concentration measurements, given by $z_k \subset \mathbb{R}^+$ at timestep k. A search area is defined in bounded 3D space, $\Omega \subset \mathbb{R}^3$ which will contain the source location and the platform will be constrained to this area ($\mathbf{p}_k \in \Omega$).

The source term to be estimated, given by Θ, is defined by $\Theta = [\mathbf{p}_s^\intercal \quad q_s \quad u_s \quad \phi_s \quad \zeta_s^\intercal]^\intercal$, where $\mathbf{p}_s = [x_s \quad y_s \quad z_s]^\intercal \in \mathbb{R}^3$ is the position in 3D Cartesian space, $q_s \in \mathbb{R}^+$ is the release rate in g/s, $u_s \in \mathbb{R}^+$ and $\phi_s \in \mathbb{R}$ are the wind speed and direction in m/s and rad respectively and $\zeta_s = [\zeta_{s1} \quad \zeta_{s2}]^\intercal \in \mathbb{R}^+$ are the gas diffusion and lifetime parameters [30]. Considering evaluation, focus will be on source location estimation and release rate as these are most relevant in a CBR disaster.

A. Dispersion Model

The dispersion of gas in the environmnet will be modelled according to the isotropic plume model [22], given by

$$\mathcal{M}(\mathbf{p}_k, \Theta_k) = \frac{q_s}{4\pi\zeta_{s1}\|\mathbf{p}_k - \mathbf{p}_s\|} \exp\left[\frac{-\|\mathbf{p}_k - \mathbf{p}_s\|}{\lambda}\right] \exp\left[\frac{-(x_k - x_s)u_s\cos\phi_s}{2\zeta_{s1}}\right] \exp\left[\frac{-(y_k - y_s)u_s\sin\phi_s}{2\zeta_{s1}}\right], \tag{1}$$

where $\lambda = \sqrt{\frac{\zeta_{s1}\zeta_{s2}}{1+(u_s^2\zeta_{s2})/(4\zeta_{s1})}}$. As the name suggests, this model assumes isotropic diffusion at the source location.

B. Sensor Motion Model

The platform will be aware of its true position in this work, given by $\mathbf{p}_k = [x_k \quad y_k \quad z_k]^\intercal$. At each timestep, the sensor platform will take a movement action or remain at the current position. A discrete, deterministic grid based motion model is adopted here that allows the platform to move to neighbouring positions, such that

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{u}_k, \tag{2}$$

where $\mathbf{u}_k = [\delta_{x,k} \quad \delta_{y,k} \quad \delta_{z,k}]^\intercal$ is the control action. A feasible action set, $\mathbf{u}_k \in \mathcal{U}_k$, is constructed such that $\mathbf{p}_{k+1} \in \Omega$. The platform will also only move in the xy plane ($\delta_{z,k} = 0$). The simplicity of this motion model prevents constriction to a specific platform.

III. Source Term Estimation

In order to conduct the STE part of this work, we perform Bayesian inference in order to sequentially update the posterior belief of the source term, $p(\Theta_k|z_{1:k})$, where k denotes the current timestep and z_k is the latest measurement. According to Bayes rule, this posterior is calculated as

$$p(\Theta_k|z_{1:k}) = \frac{g(z_k|\Theta_k)p(\Theta_k|z_{1:k-1})}{\int g(z_k|\Theta_k)p(\Theta_k|z_{1:k-1})d\Theta_k}. \tag{3}$$

In this work, the source term is constant ($\Theta_k = \Theta_{k-1}$) which implies that the predicted distribution is equivalent to the prior distribution ($p(\Theta_k|z_{1:k-1}) = p(\Theta_{k-1}|z_{1:k-1})$).

A threshold is applied to sensor measurements, z_{thr}, as they are a combination of gas detections and background noise. The value should minimise false detections without affecting sensitivity to gas. Since measurements can therefore be categorised as detection and non-detections, the likelihoods can be split in a similar way [31]. If detections are defined by $\bar{z}_k$ and non-detections by $\underline{z}_k$, then the measurement likelihood can be defined by

$$g(z_k|\Theta_k) = \begin{cases} g(\bar{z}_k|\Theta_k) & \text{if } z_k > z_{\text{thr}} \\ g(\underline{z}_k|\Theta_k) & \text{if } z_k \leq z_{\text{thr}}. \end{cases} \tag{4}$$

Assuming a detection, we define the measurement model as

$$\bar{z}_k = \mathcal{M}(\mathbf{p}_k, \Theta_k) + \bar{\nu}_k, \tag{5}$$

where $\bar{\nu}_k$ accounts for the sources of error arising between measured and modelled concentrations. Due to the uncertain nature of gas sensors, a Gaussian likelihood is implemented. This makes the likelihood function under a detection

$$g(\bar{z}_k|\Theta_k) = \frac{1}{\sigma_k\sqrt{2\pi}} \exp\left[-\frac{(\bar{z}_k - \mathcal{M}(\mathbf{p}_k, \Theta_k))^2}{2\sigma_k^2}\right], \tag{6}$$

where the standard deviation, σ_k, is a function of the concentration, given by $\sigma_k \propto \mathcal{M}(\mathbf{p}_k, \Theta_k)$.

In this work, non-detections arise from intermittent sensing due to turbulent flow (miss-detection) or concentration under the threshold value. Background noise is not considered as physical sensing is not performed. If we denote the miss-detection event by ϵ_m and small concentration event by ϵ_s then the non-detection likelihood function is formed as

$$g(\underline{z}_k|\Theta_k) = p(\epsilon_m)g(\underline{z}_k|\epsilon_m, \Theta_k) + p(\epsilon_s)g(\underline{z}_k|\epsilon_s, \Theta_k), \tag{7}$$

where $p(a)$ is the probability of event a occurring and $p(\epsilon_m) + p(\epsilon_s) = 1$. Substituting for each of the probability densities, we can now write the likelihood function as

$$g(\underline{z}_k|\Theta_k) = p(\epsilon_m) + \left((1 - p(\epsilon_m)) \times \frac{1}{2}\left[1 + \text{erf}\left(\frac{z_{\text{thr}} - \mathcal{M}(\mathbf{p}_k, \Theta_k)}{\sigma_k\sqrt{2}}\right)\right]\right), \tag{8}$$

where erf$(\cdot)$ is the error function. The value for $p(\epsilon_m)$ is to be selected based on the sensor and environment conditions.

A. Sequential Monte Carlo Implementation

The above equations are approximated with a set of N weighted particles of the source term distribution and a particle filter used to update the source term belief. The particle set is given by $\{\Theta_k^{(i)}, w_k^{(i)}\}_{i=1}^N$ and approximates the posterior distribution $p(\Theta_k|z_{1:k})$. Each sample contains a candidate

source term $\Theta_k^{(i)}$, and associated weight $w_k^{(i)}$. This allows for the usual approximation of the posterior distribution as

$$p(\Theta_k|z_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta_{\Theta_k^{(i)}}(\Theta_k), \tag{9}$$

where $\delta_{\mathbf{b}}(\cdot)$ is the Dirac delta function at point $\mathbf{b}$.

Since the source is static in this work, i.e. $\Theta_k^{(i)} = \Theta_{k-1}^{(i)} \forall\, (i)$, the typical unnormalised weight update can be simplified to

$$\tilde{w}_k^{(i)} = w_{k-1}^{(i)} g(z_k|\Theta_k^{(i)}), \tag{10}$$

where the weights can then be normalised via $w_k^{(i)} = \tilde{w}^{(i)} / \sum_{i=1}^{N} \tilde{w}_k^{(i)}$. Further detail is found in [30].

The particle degeneracy problem, that is exaggerated in the absence of a dynamic model, is alleviated with the well documented effective sample size resampling technique [32]. In order to mitigate particle impoverishment, the Markov chain Monte Carlo move step is carried out using the Metropolis-Hastings algorithm to accept or reject particles that have been regularised according to a Gaussian kernel [33].

IV. Sensor Planning

Once an estimate of the source term is obtained, the task is then to decide on where the sensor should move to for future measurements. This is a POMDP problem as gas concentration measurements are available but not direct observations of the true source. To solve this problem, an MCTS framework has been designed, allowing for non-myopic evaluation of action utility. Before detailing the MCTS process, the method to evaluate actions is first described as this is specific for the scenario presented here.

A formal definition of the current belief state, b_k, is introduced as a set containing the sensor position and posterior distribution of the source term estimate, given by

$$b_k = \left\{ \mathbf{p}_k \quad \{\Theta_k^{(i)}, w_k^{(i)}\}_{i=1}^{N} \right\}, \tag{11}$$

since the posterior in this work is the weighted particle set.

When evaluating the utility of candidate actions, the position of the sensor platform and the posterior source term estimate are progressed to the next timestep. In this work, sensor kinematics are trivial to evaluate according to (2), but progressing source term estimates is more involved. For clarity, let us consider process of progressing the belief state from $k+n-1$ to $k+n$ where $n > 0$.

The result of taking an action on the sensor platform is a new position. From this position, a predicted gas concentration measurement is generated according to the model in Sections II and III, using the initial posterior source term estimation as the ground truth and the same conditions as the real sensor (miss-detection probability and measurement threshold). This gives the predicted measurement at $\mathbf{p}_{k+n}$ as

$$\tilde{z}_{k+n}^{*} = \begin{cases} \mathcal{M}(\mathbf{p}_{k+n}, \hat{\Theta}_{k+n-1}) + \bar{\nu}_{k+n} & \text{if } \epsilon_m = 0 \\ 0 & \text{if } \epsilon_m = 1, \end{cases} \tag{12}$$

where $\epsilon_m = 1$ with $p(\epsilon_m)$, $*$ indicates that the measurement threshold z_{thr} has not been applied and $\hat{\Theta}_{k+n-1}$ is the expected source term estimate. The measurement threshold is applied such that $\tilde{z}_{k+n} = \tilde{z}_{k+n}^{*}$ if $\tilde{z}_{k+n}^{*} > z_{\text{thr}}$.

The expected source term estimate is calculated according to

$$\hat{\Theta}_{k+n-1} = \mathbb{E}[\Theta_{k+n}|z_{1:k}, \tilde{z}_{k:k+n-1}] \approx \sum_{i=1}^{N} w_{k+n-1}^{(i)} \Theta_{k+n-1}^{(i)}, \tag{13}$$

where $\tilde{z}_{k:k+n-1}$ accounts for the predicted future measurements from k to $k+n-1$ that lead to the predicted posterior of the source term estimate at $k+n-1$.

Once a predicted measurement has been generated, the initial source term estimate is updated according to (10), yielding the predicted belief state

$$b_{k+n} = \left\{ \mathbf{p}_{k+n} \quad \{\Theta_{k+n}^{(i)}, w_{k+n}^{(i)}\}_{i=1}^{N} \right\}. \tag{14}$$

Resampling and MCMC move steps are not implemented in the sensor management process. Once updated with the predicted measurement, the POMDP problem has been recast into a belief MDP that can be solved using regular MDP techniques, for example, by calculating the expected reward. In this work, the KLD is implemented as the reward to discriminate between $p(\Theta_{k+n-1}|z_{1:k}, \tilde{z}_{k:k+n-1})$ and $p(\Theta_{k+n}|z_{1:k}, \tilde{z}_{k:k+n})$ [34]. Maximising the difference between the distributions attempts to maximise the information gain under the assumptions described by (12-14). The KLD is therefore, under the particle approximation, defined by

$$\mathcal{D}_{\text{KL}}(\Theta_{k+n-1} \| \Theta_{k+n}) = \sum_{i=1}^{N} w_{k+n-1}^{(i)} \cdot \log \left(\frac{w_{k+n-1}^{(i)}}{w_{k+n}^{(i)}} \right). \tag{15}$$

This is implemented as the reward, or action value, associated with taking the candidate action that led to sensor platform state $\mathbf{p}_{k+n}$ from $\mathbf{p}_{k+n-1}$.

A. Monte Carlo Tree Search

Now that the action evaluation routine has been outlined, the MCTS framework will be introduced. MCTS sequentially expands and evaluates the search tree, $\mathcal{T}$, in an iterative process consisting of selection, expansion, simulation and backpropagation. The tree consists of nodes where $\mathcal{T}(h) = \langle N(h) \quad Q(h) \rangle$ represents a node with history h, number of visits $N(h)$ and cumulative reward $Q(h)$. The history is an ordered set containing the initial belief state and subsequent action belief state pairs, e.g., $h = \{b_k \quad \mathbf{u}_k^2 \quad b_{k+1} \quad \mathbf{u}_{k+1}^1 \quad b_{k+2} \quad \mathbf{u}_{k+2}^2 \quad b_{k+3}\}$ where $\mathbf{u}^a$ is action a from the set of feasible actions $\mathcal{U}$ at each timestep. This example history is also illustrated in Fig. 1. Nodes in the subtree of h may be represented with the addition of the candidate action and belief state, i.e. $\mathcal{T}(h\mathbf{u}_{k+n}b_{k+n})$. In this work, action nodes are not included in the tree due to the deterministic relationship between actions and the subsequent belief state. Each of the 4 processes are now outlined in detail.

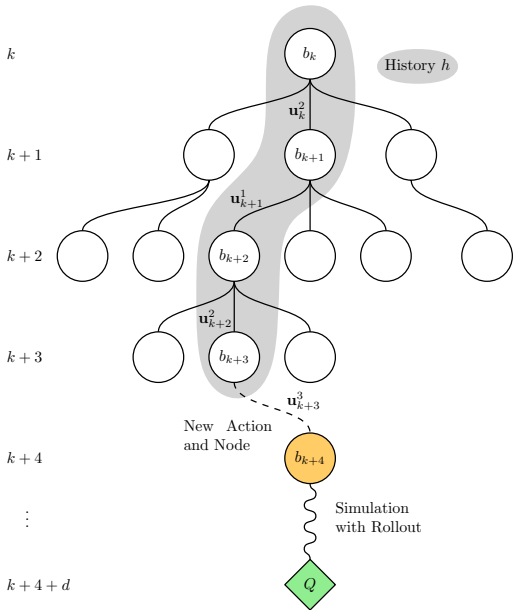


Fig. 1: A visual representation of MCTS, with the history and new candidate node highlighted.

1) Selection: During the selection process, the tree is descended by iteratively selecting the best child node to the current node and making it the current node. The criteria for selection implemented in this work is the upper confidence bound for trees (UCB1) which balances maximising reward per visit and selecting nodes with a low visit count [35]. Considering parent node $\mathcal{T}(h)$, the best child node according to UCB1 is calculated according to

$$\mathbf{u}_{k+n} = \underset{\mathbf{u}_{k+n}\in\mathcal{U}_{k+n}}{\operatorname{argmax}} \quad \frac{Q(h\mathbf{u}_{k+n}b_{k+n+1})}{N(h\mathbf{u}_{k+n}b_{k+n+1})} + c\sqrt{\frac{\log N(h)}{N(h\mathbf{u}_{k+n}b_{k+n+1})}}, \quad (16)$$

where c is the user defined exploration factor to determine how exploratory the UCB1 selection should be. It is clear from (16) that a candidate future node with no visits ($N(h\mathbf{u}_{k+n}b_{k+n+1}) = 0$) will result in the exploration term becoming ∞. Thus, all actions at a node are evaluated before extending the tree any further.

2) Expansion: With the lowest level best child selected, the tree is expanded from this node. This is where a new node is added to the tree corresponding to taking an unvisited action. This sets the initial belief state to the parent node belief state and progresses the sensor platform state according to (2), e.g. $\mathcal{T}(h\mathbf{u}_{k+n}b_{k+n+1})$. The visit count is also initialised to 1 for this new node.

3) Simulation: Simulation is where the expected reward is estimated. It is typical in MCTS to perform a Monte Carlo rollout to gain a better reward expectation [36], even though only one new node is added to the tree. For a rollout from belief state b_{k+n+1}, until $k+n+d$ where d is the the rollout depth, the reward resulting from $\mathbf{u}_{k+n}$ is given by

Algorithm 1: Monte Carlo Tree Search

Initialise tree $\mathcal{T} \leftarrow b_k$
while *Computation budget not reached* **do**
  $n = 0$
  while $\{h\mathbf{u}_{k+n}b_{k+n+1}\} \in \mathcal{T}\ \forall \mathbf{u}_{k+n} \in \mathcal{U}_{k+n}$ **do**
    Select the best child node from history h according to (16)
    Append best child to history: $h \leftarrow \{h\mathbf{u}_{k+n}b_{k+n+1}\}$
    Increment tree level: $n = n + 1$
  end
  Select unvisited action: $\mathbf{u}_{k+n} \in_R \mathcal{U}_{k+n}$ where $\{h\mathbf{u}_{k+n}b_{k+n+1}\} \notin \mathcal{T}$
  Initialise visit count: $N(h\mathbf{u}_{k+n}b_{k+n+1}) = 1$
  Progress sensor state according to $\mathbf{u}_{k+n}$: $\mathbf{p}_{k+n+1} \leftarrow \mathbf{p}_{k+n} + \mathbf{u}_{k+n}$
  Predict concentration measurement according to (12)
  Predict source term posterior according to (10)
  Update the belief state: $b_{k+n+1} \leftarrow \left\{\mathbf{p}_{k+n+1} \quad \{\Theta^{(i)}_{k+n+1}, w^{(i)}_{k+n+1}\}_{i=1}^{N}\right\}$
  Evaluate the new node through rollout: $Q(h\mathbf{u}_{k+n}b_{k+n+1}) \leftarrow \textsc{Rollout}(\mathcal{T}(h\mathbf{u}_{k+n}b_{k+n+1}), d, \gamma)$
  Backpropagate reward and visit count: $N(h) = N(h) + 1$, $Q(h) = Q(h) + Q(h\mathbf{u}_{k+n}b_{k+n+1}) \forall \{\mathbf{u}, b\} \in h$
  Append node to the tree: $\mathcal{T} \leftarrow \{h\mathbf{u}_{k+n}b_{k+n+1}\}$
end
return $\underset{\mathbf{u}_k}{\operatorname{argmax}}\, Q(b_k\mathbf{u}_k b_{k+1})$
function $\textsc{Rollout}(\mathcal{T}(h\mathbf{u}_{k+n}b_{k+n+1}), d, \gamma)$
  Calculate expected reward for new node: $Q(h\mathbf{u}_{k+n}b_{k+n+1}) = \underset{\tilde{z}_{k+n+1}}{\mathbb{E}} [\mathcal{D}_{\text{KL}}(\Theta_{k+n} \| \Theta_{k+n+1})]$
  Append node to history: $h \leftarrow \{h\mathbf{u}_{k+n}b_{k+n+1}\}$
  for $t = n+1 : d$ **do**
    Generate next action from Monte Carlo rollout: $\mathbf{u}_{k+t} \leftarrow \pi_{\text{MC}}(\mathcal{T}(h))$
    Progress sensor state according to $\mathbf{u}_{k+t}$: $\mathbf{p}_{k+t+1} \leftarrow \mathbf{p}_{k+t} + \mathbf{u}_{k+t}$
    Predict concentration measurement according to (12)
    Predict source term posterior according to (10)
    Calculate expected discounted reward and add to existing reward: $Q(h) = Q(h) + \underset{\tilde{z}_{k+t}}{\mathbb{E}} \left[\gamma^{t-n}\mathcal{D}_{\text{KL}}(\Theta_{k+t} \| \Theta_{k+t+1})\right]$
  end
  return $Q(h)$
end

$$Q(h\mathbf{u}_{k+n}b_{k+n+1}) = \underset{\substack{\tilde{z}_{k+t} \\ t=n,\ldots,n+d+1}}{\mathbb{E}} \left[\sum_{t=n}^{n+d} \gamma^{t-n}\mathcal{D}_{\text{KL}}(\Theta_{k+t} \| \Theta_{k+t+1})\right], \quad (17)$$

where γ is the discount factor and KLD calculated according to (12-15). n represents the depth of the immediate parent node in the tree. It should be noted that implementation in this work does not incorporate expectation over measurements as the mean source term is used to generate a single measurement. Its inclusion is to provide generality to the formulation.

4) Backpropagation: The last stage of the MCTS process is backpropagation. Here the expected reward and visit counts are updated in the history h. This is done by adding the latest reward $Q(h\mathbf{u}_{k+n}b_{k+n+1})$ to each node back to the root

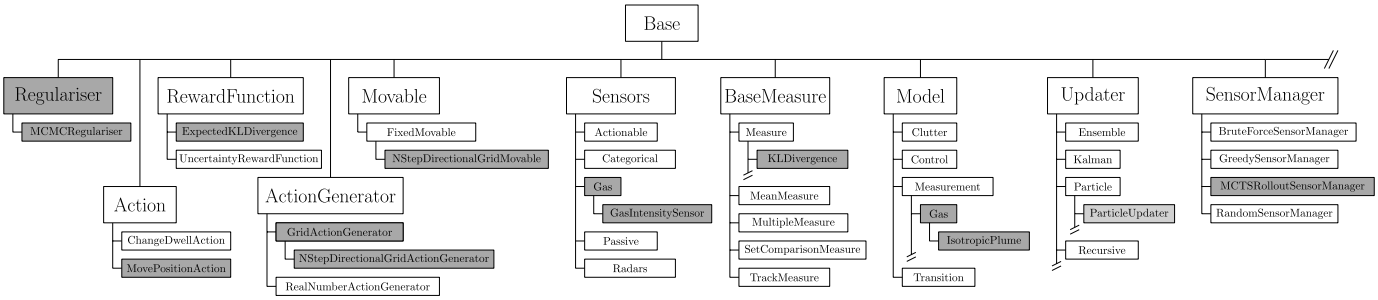


Fig. 2: A visual representation of new components (dark grey) and modified components (light grey) contributed to Stone Soup and how they fit within each respective section.

node in the tree. The visit counts for each of these nodes are also incremented by 1. The process is important for capturing future rewards in the selection process that influences the growth of the tree in the next iteration.

The MCTS algorithm, including rollout, is presented in Algorithm 1 and demonstrates the full flow of the processes. Selection occurs on lines 3-8, expansion on 9-10, simulation on 11-15 and backpropagation on 16-17.

V. Implementation

Both STE and MCTS based sensor planning has been implemented in the open source tracking and estimation framework: Stone Soup [37]. Stone Soup is an open source Python package that is available as a toolkit to learn or teach target tracking problems and techniques, compare new algorithms against many existing approaches or quickly implement a solution to real tracking problems. Through many contributions from the community, Stone Soup is quickly becoming a very powerful tracking, state estimation and sensor management resource, containing state-of-the-art techniques [38].

Stone Soup has been designed in a way that makes it modular, allowing for smaller sections of an algorithm to be interchanged to accommodate novelty more easily and reducing time to develop and implement new techniques. Implementation of the work presented here includes a number of new modules and components for the Stone Soup community. These are

- *Regulariser* module - A module enabling the possibility of particle regularisation techniques preventing particle impoverishment.
- *MCMCRegulariser* - Implementing the Metropolis-Hastings acceptance criteria, this forms the MCMC move step used in this work.
- *Gas* measurement model module - A module incorporating gas modelling techniques.
- *IsotropicPlume* gas measurement model - This is the model described in Sec. III.
- *Gas* sensor module - A category of sensors implementing gas measurement models.
- *GasIntensitySensor* sensor - A sensor type that implements the *IsotropicPlume* model.
- *MovePositionAction* - An action object that enables changes to a sensor or platform position.
- *GridActionGenerator* - A Base type to all grid based action generators.
- *NStepDirectionalGridMovable* - A movable that can be actioned to change its position in a grid based manner, like that described in Sec. II-B.
- *NStepDirectionalGridActionGenerator* - Creates the *MovePositionActions* based on the *NStepDierctional-GridMovable* movable.
- *KLDivergence* measure - Implements the KL divergence as a measure between two distributions.
- *ExpectedKLDivergence* reward - Implements the *KL-Divergence* as a reward for sensor management.
- *MCTSRolloutSensorManager* sensor manager - This implements the MCTS sensor planning technique presented in Sec. IV.

These components are also visualised in Fig. 2. Documentation for each contribution is available online[1]. Each contribution has been designed in a way to make them as general as possible, allowing for implementation in many scenarios and algorithm configurations.

VI. Simulation and Results

In order to demonstrate the algorithm, multiple simulation scenarios have been designed that will each be carried out over Monte Carlo simulations. Fig. 3 illustrates each sensor starting location and each source location and associated wind direction. Specific source parameters are provided in Table I. Each sensor starting location is simulated with each individual source location making a total of 25 simulation configurations. 250 Monte Carlo simulations are conducted to analyse performance, 10 for each configuration. The simulations are

Parameter	Source Number					Prior Distribution
	1	2	3	4	5	
x_s, m	30	30	35	20	5	$\mathcal{U}(0, 50)$
y_s, m	40	25	45	40	35	$\mathcal{U}(0, 50)$
z_s, m	1					$\mathcal{U}(0, 5)$
q_s, g/s	5	5	6	4	7	$\mathcal{G}(2, 5)$
u_s, m/s	4	4	3	2	5	$\mathcal{N}(u_{s,\text{truth}}, 2^2)$
ϕ_s, rad	$\pi/2$	$\pi/2$	$\pi/4$	$3\pi/4$	π	$\mathcal{N}(\phi_{s,\text{truth}}, (10\pi/180)^2)$
ζ_{s1}	1					$\mathcal{N}(\zeta_{s1,\text{truth}}, 0.1^2)$
ζ_{s2}	8					$\mathcal{N}(\zeta_{s2,\text{truth}}, 1^2)$

TABLE I: Source term parameters and prior distributions implemented in simulation.

[1] https://stonesoup.readthedocs.io

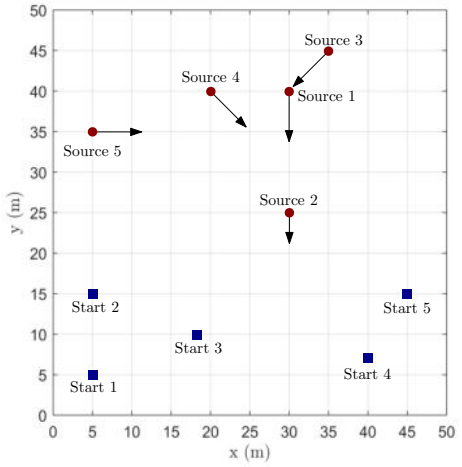


Fig. 3: An illustration of the different source locations, associated wind directions and sensor starting positions. Each sensor start location will be simulated with each source location forming 25 total configurations.

conducted over a maximum of 500 seconds with a sample time of 1 second. The algorithm will be stopped if the trace of the source term estimation covariance is < 2. The bounds of the search space, defined by Ω, are $x_{\text{min}} = 0$ m, $x_{\text{max}} = 50$ m, $y_{\text{min}} = 0$ m, $y_{\text{max}} = 50$ m, $z_{\text{min}} = 0$ m, $z_{\text{max}} = 5$ m.

1) Measurements: Measurements are generated according to (5) and noise is sampled from a Gaussian distribution such that $\bar{\nu}_k \sim \mathcal{N}(0, \sigma^2_{k,\text{truth}})$ with standard deviation calculated by $\sigma_{k,\text{truth}} = \sigma \cdot \mathcal{M}(\mathbf{p}_k, \Theta_k)$. The value set for the standard deviation factor, σ, is 0.5. The measurement threshold z_{thr} is applied such that if a measurement generated by (5) is less than z_{thr} then it will be set to 0. z_{thr} is fixed at 5×10^{-4}. The miss-detection probability, $p(\epsilon_m)$, is set to 0.3.

In terms of evaluating measurement likelihood in the filter, the same parameters are implemented but the standard deviation is calculated according to

$$\sigma_k^{(i)} = \sigma \cdot \mathcal{M}(\mathbf{p}_k, \Theta_k^{(i)}) + \nu_{\text{min}} \text{ for } i = 1 \dots N, \qquad (18)$$

where ν_{min} is a minimum noise value set to 1×10^{-4}.

2) STE Parameters: The STE particle filter is initialised with an equally weighted prior set of particles where distributions for each parameter of the source term are described in Table I. The prior has been designed to reflect knowledge of prior information in a real world scenario, such as wind speed, direction and diffusion parameters of the gas, hence these are a function of the ground truth values. The parameters of interest are the position of the source estimate in the xy plane and release rate, hence the uniform and gamma distributions chosen respectively. The number of particles, N, set for the filter is 10000.

3) Sensor Planning Parameters: The parameters used for the MCTS algorithm are 1) computational budget to run the algorithm over, per timestep, is 100 iterations 2) exploration factor for UCB1 is set to 0.05 3) rollout depth and discount factor are set to 5 timesteps and 0.9 respectively. Where appropriate, the same parameters from the STE filter are implemented in predicting the future measurements and source term distribution.

	Start Location			Source Number				
	$\mathbf{p}_{0,x}$	$\mathbf{p}_{0,y}$	$\mathbf{p}_{0,z}$	**1**	**2**	**3**	**4**	**5**
1	5	5	0	1.0	0.5	1.0	1.0	0.5
2	5	15	0	0.75	0.5	1.0	1.0	0.75
3	18	10	0	1.0	0.5	1.0	1.0	0.75
4	40	7	0	0.75	0.5	1.0	1.25	1.0
5	45	15	0	0.5	0.5	1.0	1.0	1.0

TABLE II: Sensor platform step sizes in metres.

The initial positions of the sensor platform are provided in Table II. The action space is limited to the xy plane as indicated in Sec. II. Step sizes have been chosen for each scenario in order to reflect the distance required to travel to get a good source term estimate, which varies in each situation. The step sizes, δ_x and δ_y, are provided in Table II. The platform can only traverse in one direction per timestep, or remain stationary, producing 5 possible actions per time step unless actions are rejected according to Ω.

In order to benchmark the MCTS algorithm presented, it is compared to a fully myopic alternative, also over 250 Monte Carlo simulations. This has been implemented with the same action set as above and follows the same process described by lines 11-13 in Algorithm 1 for predicting the future belief of the source term. It then selects the next action to take according to $\mathbf{u}_{k+1} = \text{argmax}_{\mathbf{u} \in \mathcal{U}_k} \mathbb{E}_{\tilde{z}_{k+1}} [\mathcal{D}_{\text{KL}}(\Theta_k \| \Theta_{k+1})]$. An example showing how to construct the algorithm presented here in a similar simulation environment using the Stone Soup components contributed here is available online[2].

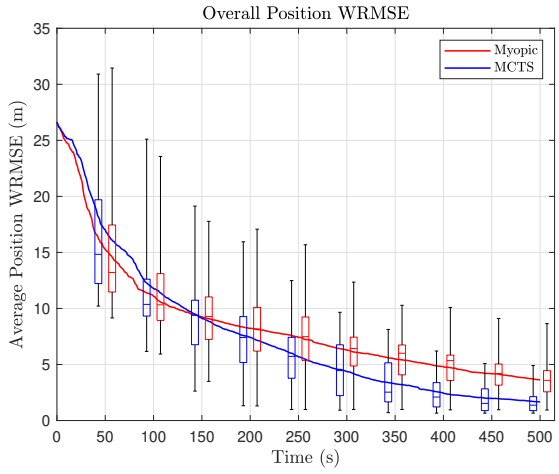
A. Results and Discussion

In order to demonstrate the performance of the proposed algorithm against the myopic alternative, three performance

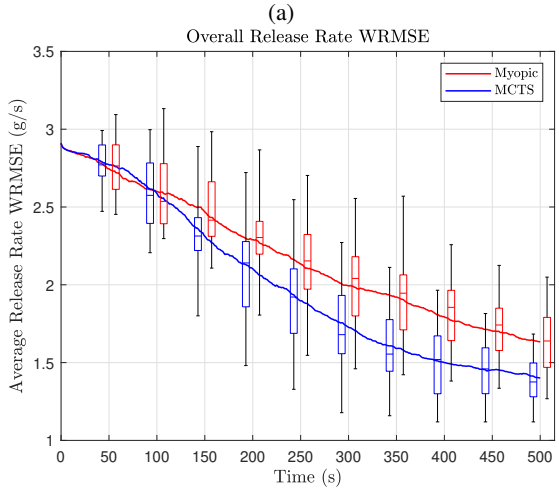
Algorithm	Start Location	Parameter	Source Number 1	2	3	4	5
MCTS	1	WRMSE_{xy}	**0.878**	**2.94**	**1.92**	**0.886**	**1.08**
		WRMSE_q	**1.17**	1.68	**1.42**	**1.17**	**1.34**
		SR	10	7	8	10	8
	2	WRMSE_{xy}	**1.46**	**0.706**	**1.20**	**0.708**	**4.89**
		WRMSE_q	**1.28**	1.67	**1.46**	**1.34**	**1.69**
		SR	8	10	9	10	8
	3	WRMSE_{xy}	**0.784**	**1.70**	**2.87**	**1.20**	**1.13**
		WRMSE_q	**1.12**	**1.37**	**1.48**	**1.45**	**1.33**
		SR	10	7	7	9	9
	4	WRMSE_{xy}	**1.37**	**2.06**	**1.08**	**2.33**	**0.662**
		WRMSE_q	**1.18**	**1.43**	**1.27**	**1.60**	1.29
		SR	8	8	8	8	10
	5	WRMSE_{xy}	**2.43**	**0.763**	**1.42**	**3.51**	**1.74**
		WRMSE_q	**1.23**	**1.55**	**1.48**	**1.68**	**1.36**
		SR	4	9	9	7	7
Myopic	1	WRMSE_{xy}	3.77	4.33	5.89	3.90	2.54
		WRMSE_q	1.31	**1.61**	1.90	1.64	1.91
		SR	5	4	0	2	4
	2	WRMSE_{xy}	3.30	1.70	4.19	2.35	8.64
		WRMSE_q	1.39	**1.57**	1.61	1.44	1.83
		SR	2	7	4	5	3
	3	WRMSE_{xy}	2.45	3.28	3.35	2.59	1.71
		WRMSE_q	1.33	1.89	1.67	1.54	1.34
		SR	6	5	4	6	7
	4	WRMSE_{xy}	4.48	4.01	4.89	3.57	2.31
		WRMSE_q	1.48	1.75	1.72	1.65	**1.27**
		SR	3	6	3	2	5
	5	WRMSE_{xy}	4.75	0.94	2.77	4.42	2.87
		WRMSE_q	1.84	2.05	1.62	1.77	1.69
		SR	1	9	5	3	2

TABLE III: Final average WRMSE for estimated source position, release rate and SR for all simulations. Bold text indicates better performance.

[2]https://stonesoup.rtfd.io/MCTS_STE



(a)



(b)

Fig. 4: WRMSE$_{xy}$ (a) and WRMSE$_q$ (b) averaged over all Monte Carlo simulation runs. Boxes indicate the minimum, maximum, lower quartile, upper quartile and median across all data, every 50 s, to provide insight into the wider data set.

metrics are shown. The first two are the weighted root mean squared error (WRMSE) calculated for the source term by

$$\text{WRMSE}_\Theta = \sqrt{\sum_{i=1}^{N} w^{(i)}(\Theta^{(i)} - \Theta)^2}. \tag{19}$$

The values considered are for the estimated source location (WRMSE$_{xy}$) and estimated release rate (WRMSE$_q$). The successful rate (SR) is the third metric which demonstrates how many simulations, for a specific source configuration and start position, achieved < 2 m WRMSE$_{xy}$ error at the end of the simulation run. The maximum SR is 10 as this is the number of Monte Carlo simulations for each configuration.

Fig. 4a illustrates the overall position estimation performance of MCTS and myopic algorithms, averaged across all sensor starting locations and source configurations. The box plots provide insight into the wider set of data with indication of the upper and lower quartiles, median and minimum and maximum data points across all simulations, shown every 50 s for both algorithms. It is clear from this data that the MCTS algorithm presented here demonstrates improved position estimation performance, converging to an average WRMSE error of 1.67 m compared to 3.65 m for the myopic. The myopic algorithm begins to reduce the WRMSE more quickly than MCTS, which is a reflection of the greedy nature of this approach, compared to the exploratory nature of the UCB1 algorithm when KLD reward is low at the beginning of the simulation.

Fig. 4b shows the overall release rate estimation performance of the two algorithms, also averaged across all simulations with the same box plot configuration. Again, the results clearly demonstrate improved performance of the MCTS algorithm. The final average WRMSE is 1.40 g/s for MCTS compared to 1.63 g/s for myopic. The larger reduction rate of the myopic algorithm is not as pronounced here, which is caused by the weaker connection between sensor position and estimated release rate when compared to position estimation.

Table III shows the mean final WRMSE$_{xy}$, final WRMSE$_q$ and SR of Monte Carlo simulations. All position error values for MCTS are lower than those achieved by the myopic approach. SR demonstrated better performance across all scenarios apart from 1 where both algorithms achieved 9. Almost all release rate error values are also lower for MCTS with myopic providing marginally better performance in 3 scenarios considering this metric.

It is clear from Figs. 4a and 4b and Table III that MCTS is able to achieve consistently better performance than the myopic approach presented in this work. It is able to achieve lower WRMSE$_{xy}$ in every scenario, lower WRMSE$_q$ and SR in most scenarios. Factors not explored in this work include computational burden which is much higher for MCTS than myopic but MCTS presents significantly lower computation burden than a fully exhaustive non-myopic approach.

VII. Conclusion

This paper has presented a technique for conducting Bayesian STE in the form of a particle filter and an MCTS framework for deciding on future sensing locations. KLD has been implemented as the reward function to maximise information gain of the source term estimate. Both STE and MCTS algorithms have been developed in and have been released to Stone Soup for the tracking and estimation community to apply and develop for their own problems.

An extensive set of simulations have been designed to test the robustness of the algorithm and compare to a greedy myopic alternative implementing the same reward. Across almost all simulated start locations and source locations, the MCTS algorithm presents better source position error, release rate error and demonstrates a higher SR, justifying its application to such a scenario.

Future work could include extending the proposed algorithm to consider more complex cluttered environments where obstacles can create local concentration minima that can be difficult to escape. This could include integration with higher fidelity simulation environments that provide a more

realistic set of measurements that are generated in isolation of the assumptions incorporated into the simulated measurement model. Finally, extending comparisons of the MCTS algorithm to longer term horizon planning such as exhaustive search and alternative tree based planning such as RRT would further establish the capability of MCTS and also highlight potential benefits in computational load.

ACKNOWLEDGEMENTS

The authors acknowledge support from the Defence Science and Technology Laboratory (Dstl), part of the UK Ministry of Defence.

REFERENCES

[1] W. Tsujita, A. Yoshino, H. Ishida, and T. Moriizumi, "Gas sensor network for air-pollution monitoring," *Sensors and Actuators B: Chemical*, vol. 110, no. 2, pp. 304–311, Oct. 2005.

[2] I. Tsitsimpelis, C. J. Taylor, B. Lennox, and M. J. Joyce, "A review of ground-based robotic systems for the characterization of nuclear environments," *Progress in Nuclear Energy*, vol. 111, pp. 109–124, Mar. 2019.

[3] L. Mastin, M. Guffanti, R. Servranckx, *et al.*, "A multidisciplinary effort to assign realistic source parameters to models of volcanic ash-cloud transport and dispersion during eruptions," *Journal of Volcanology and Geothermal Research*, vol. 186, no. 1-2, pp. 10–21, Sep. 2009.

[4] A. Norige, J. Thornton, C. Schiefelbein, and C. Rudzinski, "High-Density Distributed Sensing for Chemical and Biological Defense," *Lincoln Laboratory Journal*, vol. 18, no. 1, pp. 25–40, 2009.

[5] P. E. Bieringer, L. M. Rodriguez, F. Vandenberghe, *et al.*, "Automated source term and wind parameter estimation for atmospheric transport and dispersion applications," *Atmospheric Environment*, vol. 122, pp. 206–219, Dec. 2015.

[6] A. Stohl, P. Seibert, G. Wotawa, *et al.*, "Xenon-133 and caesium-137 releases into the atmosphere from the Fukushima Dai-ichi nuclear power plant: Determination of the source term, atmospheric dispersion, and deposition," *Atmospheric Chemistry and Physics*, vol. 12, no. 5, pp. 2313–2343, Mar. 2012.

[7] O. Saunier, A. Mathieu, D. Didier, *et al.*, "An inverse modeling method to assess the source term of the Fukushima Nuclear Power Plant accident using gamma dose rate observations," *Atmospheric Chemistry and Physics*, vol. 13, no. 22, pp. 11 403–11 421, Nov. 2013.

[8] M. Hutchinson, H. Oh, and W.-H. Chen, "A review of source term estimation methods for atmospheric dispersion events using static or mobile sensors," *Information Fusion*, vol. 36, pp. 130–148, Jul. 2017.

[9] M. Borysiewicz, A. Wawrzynczak, and P. Kopka, "Bayesian-Based Methods for the Estimation of the Unknown Model's Parameters in the Case of the Localization of the Atmospheric Contamination Source," *Foundations of Computing and Decision Sciences*, vol. 37, no. 4, pp. 253–270, Dec. 2012.

[10] A. Keats, E. Yee, and F.-S. Lien, "Bayesian inference for source determination with applications to a complex urban environment," *Atmospheric Environment*, vol. 41, no. 3, pp. 465–479, Jan. 2007.

[11] G. Johannesson, B. Hanley, and J. Nitao, "Dynamic Bayesian models via Monte Carlo - an introduction with examples," Tech. Rep., Sep. 2004.

[12] R. Lane, "Approximate bayesian computation for source term estimation," *Mathematics in Defence*, 2009.

[13] A. Gunatilaka, B. Ristic, A. Skvortsov, and M. Morelande, "Parameter estimation of a continuous chemical plume source," in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–8.

[14] A. Wawrzynczak, P. Kopka, and M. Borysiewicz, "Sequential Monte Carlo in Bayesian assessment of contaminant source localisation based on the sensors concentration measurements," in *International COnference on Parallel Processing and Applied Mathematics*, 2014, pp. 407–417.

[15] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. 1995.

[16] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953.

[17] V. Hombal, A. Sanderson, and D. R. Blidberg, "Multiscale adaptive sampling in environmental robotics," in *2010 IEEE Conference on Multisensor Fusion and Integration*, Salt Lake City, UT, USA: IEEE, Sep. 2010, pp. 80–87.

[18] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.

[19] A. Dhariwal, G. Sukhatme, and A. Requicha, "Bacterium-inspired robots for environmental monitoring," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, New Orleans, LA, USA: IEEE, 2004, 1436–1443 Vol.2.

[20] R. Russell, A. Bab-Hadiashar, R. L. Shepherd, and G. G. Wallace, "A comparison of reactive robot chemotaxis algorithms," *Robotics and Autonomous Systems*, vol. 45, no. 2, pp. 83–97, Nov. 2003.

[21] D. Harvey, Tien-Fu Lu, and M. Keller, "Comparing Insect-Inspired chemical plume tracking algorithms using a mobile robot," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 307–317, Apr. 2008.

[22] M. Vergassola, E. Villermaux, and B. I. Shraiman, "'Infotaxis' as a strategy for searching without gradients," *Nature*, vol. 445, no. 7126, pp. 406–409, Jan. 2007.

[23] M. Hutchinson, H. Oh, and W.-H. Chen, "Entrotaxis as a strategy for autonomous search and source reconstruction in turbulent conditions," *Information Fusion*, vol. 42, pp. 179–189, Jul. 2018.

[24] C. Rhodes, C. Liu, P. Westoby, and W.-H. Chen, "Autonomous search of an airborne release in urban environments using informed tree planning," *Autonomous Robots*, vol. 47, no. 1, pp. 1–18, Jan. 2023.

[25] S. An, M. Park, and H. Oh, "Receding-horizon RRT-Infotaxis for autonomous source search in urban environments," *Aerospace Science and Technology*, vol. 120, p. 107 276, Jan. 2022.

[26] C. B. Browne, E. Powley, D. Whitehouse, *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[27] Z. Sunberg and M. Kochenderfer, *Online algorithms for POMDPs with continuous state, action, and observation spaces*, Sep. 2018.

[28] K. Sun, B. Schlotfeldt, G. J. Pappas, and V. Kumar, "Stochastic Motion Planning Under Partial Observability for Mobile Robots With Continuous Range Measurements," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 979–995, Jun. 2021.

[29] S. Ragi and H. D. Mittelmann, "Random-Sampling Monte-Carlo Tree Search Methods for Cost Approximation in Long-Horizon Optimal Control," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1759–1764, Nov. 2021.

[30] M. Hutchinson, C. Liu, and W. H. Chen, "Information-based search for an atmospheric release using a mobile robot: Algorithm and experiments," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 6, pp. 2388–2402, 2019.

[31] E. Yee, "Automated computational inference engine for Bayesian source reconstruction: Application to some detections/non-detections made in the CTBT international monitoring system," *Applied Mathematical Sciences*, vol. 11, no. 32, pp. 1581–1618, 2017.

[32] A. Doucet, A. M. Johansen, *et al.*, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 3, pp. 656–704, 2009.

[33] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.

[34] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[35] P. Auer and N. Cesa-Bianchi, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 2002.

[36] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. Cambridge, Massachusetts: The MIT Press, 2022.

[37] P. A. Thomas, J. Barr, B. Balaji, and K. White, "An open source framework for tracking and state estimation ('Stone Soup')," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVI*, I. Kadar, Ed., vol. 10200, 2017, p. 1 020 008.

[38] S. Hiscocks, J. Barr, N. Perree, *et al.*, "Stone Soup: No Longer Just an Appetiser," in *2023 26th International Conference on Information Fusion (FUSION)*, Charleston, SC, USA: IEEE, Jun. 2023, pp. 1–8.