TIGRIS: An Informed Sampling-based Algorithm for Informative Path Planning

Brady Moon\textsuperscript{1}, Satrajit Chatterjee\textsuperscript{1}, and Sebastian Scherer\textsuperscript{1}

Abstract—Informative path planning is an important and challenging problem in robotics that remains to be solved in a manner that allows for wide-spread implementation and real-world practical adoption. Among various reasons for this, one is the lack of approaches that allow for informative path planning in high-dimensional spaces and non-trivial sensor constraints. In this work we present a sampling-based approach that allows us to tackle the challenges of large and high-dimensional search spaces. This is done by performing informed sampling in the high-dimensional continuous space and incorporating potential information gain along edges in the reward estimation. This method rapidly generates a global path that maximizes information gain for the given path budget constraints. We discuss the details of our implementation for an example use case of searching for multiple objects of interest in a large search space using a fixed-wing UAV with a forward-facing camera. We compare our approach to a sampling-based planner baseline and demonstrate how our contributions allow our approach to consistently out-perform the baseline by 18.0%. With this we thus present a practical and generalizable informative path planning framework that can be used for very large environments, limited budgets, and high dimensional search spaces, such as robots with motion constraints or high-dimensional configuration spaces.

[Code\textsuperscript{a}] [Video\textsuperscript{b}]

I. INTRODUCTION

Real-world data gathering is an important problem being solved through autonomous robotic platforms. Some examples include spatiotemporal monitoring [1], aerial surveillance [2]–[4], estimating urban wind fields [5], and wildfire monitoring [6]. The traditional methods when using humans to accomplish these tasks may be inefficient, infeasible, tedious, or too risky to carry out. Using autonomous robots to accomplish these tasks allows for a more efficient and safe alternative, as well as opening doors to new applications that were not formerly possible.

Each data gathering problem can present unique challenges depending on the type of data, method of collection, and sampling environment the robot must traverse, but the overall framework for solving the problem can be formulated in a unifying way. A typical way to formulate a data gathering problem is as an informative path planning (IPP) problem which involves finding paths that maximize information gathered while respecting budget constraints. These problems can be challenging to solve as they have been shown to be NP-hard [7] and even PSPACE-hard [8].

Many approaches [9]–[11] attempt to find a global solution by discretizing the space and reducing the number of states to be as small as possible, such as planning on a grid or nodes on a graph. These solutions can be resolution complete, optimal for the given problem conditioning, but are challenging to apply to real-world scenarios and often give solutions that are clearly suboptimal for the situation. This can be due to having to reduce the size of the search space in order to get a solution quickly enough or the oversimplification of the space which results in having to ignore important factors such as data gathered between the nodes on the graph.

The work by [12] adapted common sampling-based motion planning algorithms to IPP with motion constraints and asymptotic optimality analysis. Their work quickly explores the space of possible paths to find a feasible and optimal solution and was shown to find optimal solutions faster than existing solvers.

In this paper we present TIGRIS (Tree-based Information Gathering using Informed Sampling) where we build on the approach of [12] to create a sampling-based information gathering algorithm for IPP with two key improvements:
1) Perform informed sampling within a subset of the configuration space that influences the estimated information gain, allowing for planning in larger and higher dimensional search spaces by prioritizing the exploration of paths with high reward.

2) Include edge rewards using a novel edge cost formulation and demonstrate its advantage in spite of the extra computation.

We demonstrate that our algorithm works well at rapidly planning high information gain paths in high-dimensional and large spaces while being constrained to a relatively small budget relative to the size of the space. An example use-case of this algorithm is shown in Fig. 1 where a fixed-wing UAV with a forward-facing image sensor is searching a region for a missing hiker. In such a situation, it is necessary for a path planner to be able to plan paths through areas of high potential reward in order to maximize the likelihood of collecting the most amount of relevant information while subject to a relatively small budget.

Additionally, in this paper we also detail how we implement TIGRIS for our specific use case—searching for objects of interest using a fixed-wing UAV in an environment that is much larger than the available budget. We step through all the considerations that had to be made to successfully implement TIGRIS for this challenging problem including how to estimate edge rewards and the potential gain in information of a candidate path in the tree. We further detail our environment setup, belief representation, and how we implement our approach in a practical and feasible way. Finally, we release TIGRIS as an open source package with this paper.

The paper is organized as follows: Section II reviews the related works in this field and the contributions of our work. Section III lays out the problem definition. Section IV explains our proposed algorithm. Section V details our approach to implementing TIGRIS for our particular use case and all the considerations that were made in our planning environment. Section VI lays out our testing framework and results compared to the baseline method. Section VII gives the conclusion and future work.

II. RELATED WORK

Many approaches have been used for robotic data gathering in physical spaces. The IPP problem can also be formulated as an adaptive sampling problem or an orienteering problem with each having their own subtle variations in the problem formulation.

Some of the previous works have tried to solve the IPP problem using a receding horizon approach. The work by [13] proposed an adaptive receding horizon controller for marine monitoring that modifies the lookahead step size to not get stuck in a local optima. The work by [3] compares finite horizon tree search, locally optimal myopic search, Monte Carlo tree search, and their own versions of cluster tree search when searching for a target while gliding between thermals. They explain the strengths and weaknesses of each method and show their versions of cluster tree search perform the best in their mission framework. These receding horizon methods reactively plan locally optimal paths and do not fully leverage the known prior environment information.

In regard to planning globally optimal paths, many approaches discretize the space and turn the problem into a graph search problem such as in [7], [14], [15]. However, the work by [16] found that many of the graph based approaches can perform close to that of a “lawnmower” search due to their inability to revisit areas more than once. Allowing for revisits can be very important for finding the optimal path when the information gain function is submodular or when multiple revisits are necessary for a given task. The work in [10] formulates the IPP as an orienteering problem and presents an anytime solution to informative path planning in a discrete space. The authors do note that their approach only offer low run-times when the solution space has multiple paths that are near-optimal, and their approach also does not allow for revisits. The run-times become quickly intractable as the probability of near-optimal paths existing in the solution space becomes small. Discretized approaches can also start to become computationally intractable as the graph size increases [1], [10] which makes them infeasible in large search spaces.

As an alternative to this approach, information gathering problems can also be formulated in continuous spaces. Examples of this approach can be seen in the works by [12], [17], [18]. In all these approaches the authors adopt a sampling-based approach where points in continuous space are sampled and added to a tree. An anytime best path is chosen in the tree that maximizes information gain while being subject to a budget constraint. A drawback of these approaches is that the problem formulations can still suffer in large spaces. As the dimensionality increases, the space needed to search can also explode relative to the size of the space with high information reward.

This raises the question: how do we perform anytime informative path planning in large high-dimensional spaces efficiently? In this work we explore this question and present a practical solution to this challenging problem. Our method uses a sampling-based approach but leverages the prior environment information to perform informed sampling within the space. This prioritizes building the tree in areas of high reward even when the dimensionality of the space is large. Our inclusion of edge rewards also helps in more accurately estimating path rewards thereby quickly expanding the tree in large spaces without sacrificing accurate path reward estimation.

III. PROBLEM DEFINITION

Let \( T \) represent a sensor trajectory in \( \mathbb{T} \) and \( C(T) \) be the cost function for a given trajectory. Let \( I(T) \) be the expected reward or information gain from a trajectory and \( B \) be the budget constraint.

We define the informative path planning problem as maximizing information gain subject to a budget constraint as in the following:

\[ \arg \max_{T} I(T) \text{ subject to } C(T) \leq B \]
where an optimal trajectory \( T^* \) is found that does not exceed the budget \( B \).

In this formulation, the information reward function \( I(\cdot) \) is not constrained to being modular and could also be time-varying modular or even submodular [19].

IV. METHODOLOGY

On a high level, TIGRIS is similar to other sampling-based planners in that points are sampled within the search space and used to build out a tree. For the case of IPP, each node in the tree contains the state, path cost, and node information gain. At the end of the planning time, the node with the highest value of path information is returned. When searching over a large space while subject to a limited budget, our planner seeks to build high reward paths quickly through prioritizing the sampling of regions with high reward. This allows us to generate plans that outperform random sampling approaches throughout the planning time.

The code for TIGRIS is described in Alg. 1.

Let \( \mathcal{X} \) represent the state space and \( x_{start} \) being the starting configuration within \( X_{free} \). \( T \) represents the time allowed for planning and \( \Delta \) is the extend distance when building the tree. For brevity, we will use the notation \( X \leftarrow \{x\} \) and \( X \leftarrow X \cup \{x\} \) to represent the operations \( X \leftarrow X \cup \{x\} \) respectively.

The algorithm starts by calculating the information reward for the starting configuration \( x_{start} \in \mathcal{X} \) with \( I(\cdot) \) and sets the cost to zero (Alg. 1 Line 1). It then initializes the starting node \( n_{start} \), adds it to the vertices set \( V \), and initializes the edges set \( E \) and graph \( G \) (Alg. 1 Line 2-3).

While the computation time has not exceeded the allowed planning time \( T \), the tree will continue to be expanded. This begins with a new configuration \( x_{sample} \) being sampled using \( \text{InformedSample}(\cdot) \) (Alg. 1 Line 5), which in our case we use weighted sampling where the probability is directly proportional to the information reward. Details on how we implemented this sampling efficiently is found in Section V-C.

The nearest node \( n_{nearest} \in V \setminus V_{closed} \) to \( x_{sample} \) is found using the function \( \text{Nearest}(\cdot) \) (Alg. 1 Line 6). Efficient nearest neighbor search could be implemented using structures such as a k-d tree or R-tree.

A feasible point \( x_{feasible} \) is found by extending from the nearest node by a maximum cost \( \Delta \) using the function \( \text{Steer}(\cdot) \) (Alg. 1 Line 7). This feasible point may have an edge cost less than \( \Delta \) because the \( \text{Steer}(\cdot) \) function will extend until a collision or the budget is reached. This is a subtle difference from [12] in that ours will not allow points to extend beyond the budget but will rather reduce the extended distance to create a path that uses the entire budget.

If the feasible point is not the same as the sample point, then near points \( n_{near} \in V \setminus V_{closed} \) within a radius from \( x_{feasible} \) are found using \( \text{Near}(\cdot) \) and set as \( n_{near} \) (Alg. 1 Line 8-9). This radius could be predefined or set using a function like in [20] which defines a closed hypersphere of radius \( r_n \) where \( n \) is the number of nodes.

For each node \( n_{near} \) in the set \( N_{near} \), and new state \( x_{new} \) and edge \( e_{new} \) is extended from \( n_{near} \) to \( x_{feasible} \) using the \( \text{Steer}(\cdot) \) function of cost \( \Delta \) (Alg. 1 Line 11). If the new state \( x_{new} \) is not the same as \( x_{feasible} \) (meaning it is not trapped or in the closed set), then a new node \( n_{new} \) will be created. The information reward for the node is estimated using \( I(\cdot) \) which calculates the reward for the trajectory up to the new state \( x_{new} \), including the new edge \( e_{new} \) and entire trajectory of edges and states preceding it. Depending on the modularity and temporal dependence of your information function, calculating the reward for a new node might be as simple as appending the additional information gain from the new state and edge or, alternatively, the entire trajectory up to that point may need to be calculated.

After calculating the estimated information reward, the cost of the new edge is found with \( \text{Cost}(\cdot) \) and appended to the cost of the near node \( C_{n_{near}} \). With the new state, information, and cost, a new node is now created (Alg. 1 Line 13-15). However, before adding this new node to the tree, it is first checked for if it should rather be pruned. The \( \text{Prune}(\cdot) \) function examines nearby nodes to see if it is worth adding the new node. This could be a simple heuristic of seeing if a nearby node has a lower cost and higher reward, but this heuristic can have implications on
finding the optimal solution. A variety of other heuristics can be used to reduce computational complexity, but, to ensure optimality, the upper bound on reward for the nodes would need to be used within the \textit{Prune(·)} function.

If the node isn’t pruned, the edge is added to the set \( E \), the node is added to the set \( V \), and the graph is updated (Alg. 1 Line 17-19). If the cost of the new node is equal to the budget, then the node is also added to the closed set \( V_{closed} \). Lastly, once the while loop is terminated when the max planning time is exceeded, the path with the highest information reward is returned.

V. IMPLEMENTATION

We implemented TIGRIS with the objective of reducing entropy with a bias toward increasing belief probabilities. Our data gathering is performed using a fixed wing UAV and a static forward facing camera. We discretize our space into grid cells with index \( i \), and let \( X_i \) be the event that an object of interest lies within the cell at index \( i \). Therefore \( P(X_i) \) is the probability of the event, such as a lost hiker being in the cell. In the formulation of entropy reduction, the value within a cell could easily be any other metric that we want to reduce uncertainty for such as a model for radioactive strength in a space or a temperature field. We assume a prior belief over the map that we then leverage to plan a path that reduces the overall entropy.

Let \( x \in \mathbb{R}^3 \times SO(2) \) represent the state of a fixed wing UAV in inertial space. We assume constant curvature and connect states using Dubins paths. The sensor is statically mounted facing forward on the UAV with a downward pitch angle of \( \theta \) and the heading matching the vehicle of \( \psi \) as seen in Fig. 3.

For the \textit{Prune(·)} function, we chose an aggressive heuristic of not adding a new node if a nearby node has a lower cost and higher reward. For our nearest neighbor search we used the Open Motion Planning Library [21]. Our function for \textit{InformedSample(·)} is seen in Alg. 2. We implemented weighted sampling with the probability of selection being proportional to information reward, however calculating the information reward for every state would be computationally infeasible. To handle this, we compute the information reward for viewing each cell at an optimal range and then use that value for a weighted sample over cells (Alg. 2 Line 1). We then sample uniformly within the subset of the space that is independent of the sampled cell, \( z \) and \( \psi \) (Alg. 2 Line 2-3). The full sampled state can then be found using \( v_{opt} \), which is a fraction reflecting the desired vertical placement of the target between the bottom of the frame and the optical axis in the image frame (Alg. 2 Line 4-7).

A. Sensor Model

A sensor model is necessary for belief map updates given sensor measurements. The function approximating the sensor performance could be derived through testing a given perception system and approximated with a high order polynomial or even be a lookup based on the testing data. However the method, it is important to ensure the model maps closely and accurately to the actual system, otherwise the search algorithm could lead to erroneous behaviors. There are many factors that could be included as inputs to a sensor model include target range, sensor velocity, sensor acceleration, shutter speed, sensing angle, altitude, target material type, weather, fog, lighting conditions, and target behavior.

In our implementation, we are modeling the performance of a perception system that uses an electro-optical sensor. Let the variable \( Z \) represent the event of a positive measurement. We model both the true positive rate \( P(Z|X) \) and false positive rate \( P(Z|\neg X) \) using (1), with the false positive rate just being the complement of the equation in our case, and \( r \) being the range of the object from the sensor and \( a, b, \) and \( c \) being parameters to fit the model to the performance of a perception pipeline. \( \beta \) is where the sensors performance drops off to where measurements do not change the belief space. Fig. 2 shows the plots of (1) and its complement.

\[
f(r) = \begin{cases} 
\frac{1}{a + e^{b(r-c)}} & r \leq \beta \\
0.5 & r > \beta
\end{cases}
\]  

(1)

B. Information Reward Function

We base our information reward function on the reduction in entropy [22] from a new measurement. The full equation for Shannon entropy in our case is

\[ H(X) = -P(X) \log P(X) - P(\neg X) \log P(\neg X) \]

Using Bayes’ theorem to find \( P(X|Z) \), the entropy reduction due to a positive measurement would be

\[ \Delta H(X|Z) = H(X) - H(X|Z) \]

![Fig. 2. An example sensor model of (1) with blue being the true positive rate and red being the false positive rate.](image-url)
In order to prioritize reducing entropy in areas that increase our confidence in objects locations, we also add weights to the entropy reduction. \( R_p \) is the weight for increasing probabilities and \( R_n \) is for decreasing probabilities.

Ideally we would compute the expectation of the entropy reduction over the entire path, but computing this for all combinations of outcomes for each path segment would be computationally expensive. We follow the same approach as [18] of using the optimistic approximation of the expected reward to compute the upper bound of the reward. If \( P(X) \geq 0.5 \) then we assume a positive measurement \( Z \) and the converse if less than 0.5. The resulting reward function is the following:

\[
R(X) = \begin{cases} 
R_p \Delta H(X|Z) & P(X) \geq 0.5 \\
R_n \Delta H(X|\neg Z) & P(X) < 0.5 
\end{cases}
\]  

(2)

C. Estimating Edge Rewards

When calculating the information reward over a path, the sensor footprint can be projected onto the surface plane to find all cells within the footprint and the reward for each cell is calculated using (2) and summed. This step becomes more complicated when including edges in the reward calculation. Discretizing the edge into nodes would have imperfect overlaps due to a non-square sensor footprint, thus not including some of the grid cells in calculating the edge reward. To mitigate this issue, large overlaps of the sensor footprints would be needed but would thus lead to a large increase in the computation time for grid cell belief updates. Rather than choosing between the tradeoffs, we compute an approximation of the edge reward by finding all cells within the sliding sensor footprint along the edge and find the minimum distance the UAV will be to each of those cells. This ensures all cells within the edge are accounted for and each belief value is updated using a measurement when the sensor is closest to that cell.

The minimum range \( r_{min} \) of a point to our sensor is first found by calculating the perpendicular distance \( d \) of the point to the edge line. Let \( z \) be the altitude of the camera sensor above the ground, \( k \) be half of the length of the bottom edge of the projected camera footprint on the ground plane, \( \Theta_v = VFOV \), and \( \Theta_h = HFOV \). The distance \( L \) from the edge defines the distance a point will be to the sensor when it transitions from being on the bottom of the frame to the left or right edge of the image frame and is calculated using (3).

\[
L = \begin{cases} 
k & \theta \geq \frac{\Theta_v}{2} \\
z \tan(\frac{\Theta_h}{2}) & \theta \leq \frac{\Theta_h}{2}
\end{cases}
\]  

(3)

We then calculate \( r_{min} \) using (4) where \( v = z \tan(\theta - \frac{\Theta_v}{2}) \) and \( \Delta v = \frac{d - L}{\tan(\theta)} \). The resulting \( r_{min} \) is used to for each cell belief update and reward calculation of cells that are being passed over. For cells at the end of edges or at nodes, euclidean distance is used for the range. A visual representation of these calculations is seen in Fig. 3.

\[
r_{min} = \begin{cases} 
\left\| \begin{bmatrix} d & v & z \end{bmatrix} \right\|_2 & \theta \geq \frac{\Theta_v}{2} \land d < L \\
\left\| \begin{bmatrix} d & 0 & z \end{bmatrix} \right\|_2 & \theta \leq \frac{\Theta_v}{2} \land d < L \\
\left\| \begin{bmatrix} d & v + \Delta v & z \end{bmatrix} \right\|_2 & \theta > \frac{\Theta_v}{2} \land d \geq L \\
\left\| \begin{bmatrix} d & \Delta v & z \end{bmatrix} \right\|_2 & \theta \leq \frac{\Theta_h}{2} \land d \geq L
\end{cases}
\]  

(4)

VI. Testing Results

We compared our TIGRIS algorithm against the RIG-tree algorithm outlined in [12] which was shown to find optimal solutions in similar testing environments. We used the same parameters for each algorithm and implemented both in C++. We also used the same information reward function when evaluating final trajectories for each algorithm. The simulation environment was a 5000x5000 meter area with grid cells of 50x50. The UAV had a budget \( B = 6000 \) and the camera had a pitch of \( \theta = 65^\circ \).

The simulations were run on a 3.4 GHz Intel i7 processor with 16 GB of RAM. 12000 Monte Carlo simulations were carried out with 1-12 randomly placed target belief centroids. For each centroid, a random prior mean belief value and standard deviation were selected for a Gaussian kernel.

The results for the Monte Carlo simulations are seen in Fig. 4. The plot shows the information reward of the best path averaged over the 12000 runs. For both methods a high quality path is quickly found and both seem to be approaching an asymptotic value. It is also clear to see the improvement in TIGRIS over the entire planning time. The mean and standard deviation of information reward for TIGRIS and RIG-tree at 5 seconds is 427.53±126.82 and 362.38±136.26 respectively, which is an 18.0% improvement. This give a P value of well less than 0.0001 for the single tail difference between the two means. It is interesting to note the right side of Fig. 4 which shows that the percent difference in the
The results of 12000 Monte Carlo simulations. The blue line shows the average path information reward for the TIGRIS planner while the orange is for the TIG-tree. The graph on the right shows the difference in the two means.

The mean rewards continue to increase at a near linear rate after the initial jump.

Table I groups the results based on the number of target belief centroids that were in the testing environment. The data shows that as the density of the target belief centroids decreases, the performance difference between the algorithms increases. Informed sampling becomes more of an advantage as it becomes harder for random sampling to find high-reward paths. This shows the strength of our method in sparse environments. Even at the lowest difference between the algorithms, the P-values of the difference between the mean information rewards were all far below 0.0001.

Fig. 5 shows one result within the simulation environment. The planned path by TIGRIS is seen in blue and has an information reward of 531.2 while the RIG-tree path is in pink and has an information reward of 407.2. The green arrows at each node show the $\psi$. The background color shows the information reward for each cell with blue being low reward and red being high. The image on the bottom right shows the 3D view of the environment and helps visualize the viewing frustum and sensor footprint. From looking at the results, you can see how TIGRIS maximizes the amount of information reward for each cell with blue being low reward and red being high. The graph of the best path information reward versus time can be seen in the bottom left of Fig. 5. The best path reward for the baseline decreased at some points in the beginning of the planning. Even with the additional computation time due to edge reward calculations, testing showed TIGRIS found better paths by using our edge information reward approximation. Another benefit to adding edge rewards is that it allows for larger extend distances while still being able to closely approximate the information reward over the trajectory. This means TIGRIS can plan in large spaces with a less dense tree than the baseline. Without the edge rewards, calculating the trajectory reward with only the nodes would be a poor approximation of the path reward and would not reflect the reality that information is also gathered between the nodes.

VII. CONCLUSION

This work presents a novel informed sampling-based algorithm for informative planning called TIGRIS. Our method is tested using Monte Carlo simulations and is shown to have a statistically significant improvement over a comparable
baseline. TIGRIS does well at quickly finding paths that maximize the information reward and reducing entropy over the belief space. As the budget increases to be large relative to the search space, the performance of TIGRIS would be expected to degrade as the problem approaches that of a coverage planner and the density of the tree would need to be intractably high. In these situations, replanning on the fly with TIGRIS could be a potential solution and is slated for future work. Additional future directions of this work could include searching for and tracking moving target, local optimization of the final path, or local optimization over path segments within the planning framework.

REFERENCES

[1] J. Binney, A. Krause, and G. S. Sukhatme, “Optimizing waypoints for monitoring spatiotemporal phenomena,” The International Journal of Robotics Research, vol. 32, pp. 873–888, July 2013.
[2] T. Cabreira, L. Brisolara, and P. R. Ferreira Jr., “Survey on Coverage Path Planning with Unmanned Aerial Vehicles,” Drones, vol. 3, Jan. 2019.
[3] J. L. Nguyen, “Long-term informative path planning with autonomous soaring,” 2015.
[4] B. G. Moon and C. K. Peterson, “Learned search parameters for cooperating vehicles using gaussian process regressions,” in 2018 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 493–502, 2018.
[5] J. Patrikar, B. Moon, and S. Scherer, “Wind and the city: Utilizing UAV-based in-situ measurements for estimating urban wind fields,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
[6] S. M. Towhidul Islam and X. Hu, “Real-time on-board path planning for uas-based wildfire monitoring,” in 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 527–535, 2021.
[7] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, “Efficient informative sensing using multiple robots,” Journal of Artificial Intelligence Research, vol. 34, p. 707–755, Apr 2009.
[8] J. H. Reif, “Complexity of the mover’s problem and generalizations,” in 20th Annual Symposium on Foundations of Computer Science (sfcs 1979), pp. 421–427, 1979.
[9] J. Yu, M. Schwager, and D. Rus, “Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks,” in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 342–349, 2014.
[10] S. Arora and S. Scherer, “Randomized algorithm for informative path planning with budget constraints,” in 2017 IEEE International Conference on Robotics and Automation (ICRA), (Singapore, Singapore), pp. 4997–5004, IEEE, May 2017.
[11] S. Almansari, E. Fata, and S. L. Smith, “Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations,” The International Journal of Robotics Research, vol. 33, no. 1, pp. 138–154, 2014.
[12] G. A. Hollinger and G. S. Sukhatme, “Sampling-based robotic information gathering algorithms,” The International Journal of Robotics Research, vol. 33, pp. 1271–1287, Aug. 2014.
[13] S.-H. Yoo, A. Stuntz, Y. Zhang, R. Rothschild, G. A. Hollinger, and R. N. Smith, “Experimental Analysis of Receding Horizon Planning Algorithms for Marine Monitoring,” in Field and Service Robotics (D. S. Wettergreen and T. D. Barfoot, eds.), vol. 113, pp. 31–44, Cham: Springer International Publishing, 2016. Series Title: Springer Tracts in Advanced Robotics.
[14] J. Binney, A. Krause, and G. S. Sukhatme, “Informative path planning for an autonomous underwater vehicle,” in 2010 IEEE International Conference on Robotics and Automation, pp. 4791–4796, 2010.
[15] H. Zhu, J. J. Chung, N. R. J. Lawrance, R. Siegwart, and J. Alonso-Mora, “Online informative path planning for active information gathering of a 3d surface,” CoRR, vol. abs/2103.09556, 2021.
[16] S. Frolov, B. Garau, and J. Bellingham, “Can we do better than the grid survey: Optimal synoptic surveys in presence of variable uncertainty and decorrelation scales,” Journal of Geophysical Research: Oceans, vol. 119, no. 8, pp. 5071–5090, 2014.
[17] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, “An Efficient Sampling-Based Method for Online Informativie Path Planning in Unknown Environments,” IEEE Robotics and Automation Letters, vol. 5, pp. 1500–1507, Apr. 2020.
[18] G. Hollinger, “Long-horizon Robotic Search and Classification using Sampling-based Motion Planning,” in Robotics: Science and Systems XI, Robotics: Science and Systems Foundation, July 2015.
[19] A. Krause and C. Guestrin, “Submodularity and its applications in optimized information gathering,” ACM TIST, vol. 2, p. 32, 07 2011.
[20] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” CoRR, vol. abs/1005.0416, 2010.
[21] I. A. Şucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” IEEE Robotics & Automation Magazine, vol. 19, pp. 72–82, December 2012.
[22] L. Paull, C. Thibault, A. Nagaty, M. Seto, and H. Li, “Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle,” IEEE Transactions on Cybernetics, vol. 44, no. 9, pp. 1605–1618, 2014.