Sampling-based robotic information gathering algorithms

Citation	Hollinger, G. A., & Sukhatme, G. S. (2014). Sampling-based robotic information gathering algorithms. The International Journal of Robotics Research, 33(9), 1271-1287. doi:10.1177/0278364914533443
DOI	10.1177/0278364914533443
Publisher	SAGE Publications
Version	Accepted Manuscript
Terms of Use	http://cdss.library.oregonstate.edu/sa-termsofuse

Sampling-based Robotic Information Gathering Algorithms

Geoffrey A. Hollinger[*] and Gaurav S. Sukhatme[†]

August 12, 2014

Abstract

We propose three sampling-based motion planning algorithms for generating informative mobile robot trajectories. The goal is to find a trajectory that maximizes an information quality metric (e.g., variance reduction, information gain, or mutual information) and also falls within a pre-specified budget constraint (e.g., fuel, energy, or time). Prior algorithms have employed combinatorial optimization techniques to solve these problems, but existing techniques are typically restricted to discrete domains and often scale poorly in the size of the problem. Our proposed rapidly-exploring information gathering (RIG) algorithms combine ideas from sampling-based motion planning with branch and bound techniques to achieve efficient information gathering in continuous space with motion constraints. We provide analysis of the asymptotic optimality of our algorithms, and we present several conservative pruning strategies for modular, submodular, and time-varying information objectives. We demonstrate that our proposed techniques find optimal solutions more quickly than existing combinatorial solvers, and we provide a proof-of-concept field implementation on an autonomous surface vehicle performing a wireless signal strength monitoring task in a lake.

1 Introduction

Mobile robots are increasingly being tasked with gathering information about their environment. Emerging application domains include marine monitoring (Smith et al., 2011), aerial surveillance/search (Hollinger et al., 2009), tactile object recognition (Hsiao et al., 2010), and facility inspection (Hollinger et al., 2013). In all of these domains, the goal is to maximize some metric of information quality (e.g., probability of locating a target, accuracy of the inspection, or quality of the survey) while satisfying constraints on fuel, energy, or time.

The *informative motion planning* problem of maximizing information gathered subject to a budget constraint is particularly challenging because it typically requires searching over a large and complex space of possible trajectories. Such problems have been shown to be NP-hard (Singh et al., 2009), or even PSPACE-hard (Reif, 1979a), depending on the form of the objective function and the space of possible trajectories. Prior work has leveraged approximation algorithms (Singh et al., 2009) and branch and bound solvers (Binney and Sukhatme, 2012) to provide informative trajectories for mobile robots. However, these prior algorithms are limited to discrete domains and often scale poorly in the amount of budget and the size of the environment.

[*]G. Hollinger is with the School of Mechanical, Industrial & Manufacturing Engineering, Oregon State University, Corvallis, OR 97330 USA, `geoff.hollinger@oregonstate.edu`

[†]G. Sukhatme is with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089 USA, `gaurav@usc.edu`

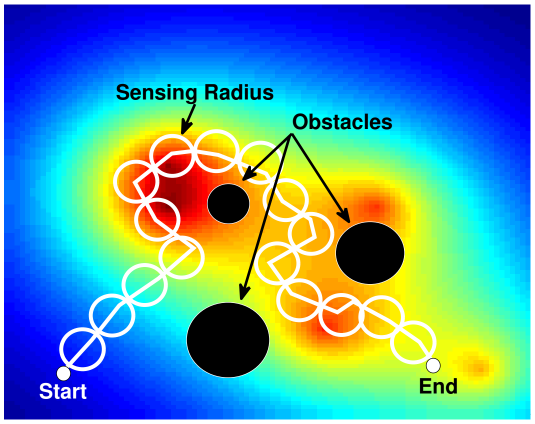


Figure 1: Example information landscape and trajectory generated by the proposed algorithm for an autonomous vehicle monitoring a phenomenon of interest (e.g., an ocean event, seismic activity, or an area to be surveyed). The red areas have higher information quality than the blue areas. Our algorithms extend ideas from sampling-based motion planning to provide informative trajectories that satisfy a budget constraint.

In this paper, we provide a suite of rapidly-exploring information gathering (RIG) algorithms that solve the informative motion planning problem using iterative sampling. Our algorithms combine ideas from asymptotically optimal rapidly-exploring random trees (RRT*), rapidly-exploring random graphs (RRG), and probabilistic roadmaps (PRM*) (Karaman and Frazzoli, 2011) with insights from branch and bound optimization (Binney and Sukhatme, 2012) to provide improved efficiency and generality. Adapting theoretical analysis from prior work also allows us to show asymptotic optimality for a class of objective functions.

Prior algorithms, including information-rich RRTs (Levine, 2010) and belief-space RRTs (Bry and Roy, 2011), have been shown to solve related problems for certain objective functions. The key novelty of this paper is the introduction of three incremental sampling-based algorithms (built off of RRT*, RRG, and PRM* respectively) for generating informative trajectories that are asymptotically optimal, outperform existing combinatorial solvers, and allow for general motion constraints. An early version of this work was presented in our prior conference paper (Hollinger and Sukhatme, 2013). The journal version includes the introduction of two additional variants of the RIG algorithm and more extensive quantitative comparisons of the proposed techniques.

The remainder of this paper is organized as follows. We first discuss related work to highlight the need for efficient informative motion planning algorithms (Section 2). We then provide a formalization of the informative motion planning problem (Section 3), and we introduce the proposed sampling-based algorithms along with an analysis of asymptotic optimality (Section 4). To test our approach, we compare the proposed algorithms to existing combinatorial solvers, and we also provide a field implementation on an autonomous surface vehicle (Section 5). Finally, we draw conclusions and discuss avenues for future work (Section 6).

2 Related Work

Autonomous information gathering problems have a rich history in mathematics, computer science, and robotics dating back to early work in sequential hypothesis testing (Wald, 1945). Early research was concerned with the problem of determining which experiments to run to best determine the nature of an unknown. Experimental design has also been used extensively in the statistics and physics communities (Weber, 2010). Similar ideas were later generalized to account for constraints on physical systems, which resulted in a vibrant community studying active perception (Bajcsy, 1988).

The problems discussed here are closely related to the problem of adaptive sampling, which has been studied extensively in the robotics (Fiorelli et al., 2006), AI (Low et al., 2009), and machine learning (Thompson et al., 2010) communities. In adaptive sampling, the goal is to choose observation locations that minimize prediction uncertainty or maximizing some measure of information gain. Our work extends sampling-based motion planning techniques to provide improved solutions to a broad class of adaptive sampling problems.

Active perception and adaptive sampling problems have been extended to account for mobile sensing within the framework of Bayesian reasoning (Cameron and Durrant-Whyte, 1990). Subsequent works have resulted in an extensive suite of solutions that incorporate information theoretic measures for problems like object recognition, mapping, and scene reconstruction (see (Chen et al., 2011) for a survey). Modern applications of next best view planning and belief space planning continue to push the envelope of gradient-based optimization in these domains (Bourgault et al., 2002; van den Berg et al., 2012). While such algorithms have been shown to be useful for a range of domains, they typically rely on restrictive assumptions on the objective function and do not have guarantees on global optimality.

Finite-horizon model predictive control methods (Bourgault et al., 2003; Ryan and Hedrick, 2010) provide improvement over myopic techniques, but they do not have performance guarantees beyond the horizon depth. An alternative is to formulate the information gathering problem as a POMDP (Myers and Williams, 2010), but such approaches are notoriously difficult to apply to large problem instances. Sampling-based approaches have been applied to POMDPs in the past (Thrun, 1999). However, these past methods depend on approximating the entire belief space, which is often an extremely high-dimensional space.

A number of general solvers for robotic information gathering problems utilize combinatorial optimization techniques to search over a discrete grid. The recursive greedy algorithm (Singh et al., 2009) is one example that achieves bounded performance for submodular objective functions. Branch and bound techniques have also been proposed that only require the objective function to be monotonic (Binney and Sukhatme, 2012). The resulting solvers typically require computation exponential in the size of the problem instance due to the large blowup in the search space with increasing budget. An alternative is to utilize a finite-horizon solver that solves the problem for only a portion of the budget at a time (Hollinger et al., 2009). While such solvers perform heuristically well, they do not carry guarantees for behavior outside the horizon length.

Motion and path planning are fundamental problems in robotics and have been studied extensively in the past two decades (Latombe, 1991; LaValle, 2006). Increasing the degrees of freedom of the robot or the dimensionality of the environment typically causes an exponential increase in the computation required to solve the planning problem optimally. Thus, motion and path planning problems are generally computationally hard (NP-hard or PSPACE-hard) (Reif, 1979b). Modern planning methods have focused on the generation of approximate plans with limited computation (e.g., RRT* algorithms (Karaman and Frazzoli, 2011)). Our work extends these ideas to domains where the robot seeks to gather information along the trajectory.

The use of sampling-based motion planning algorithms, such as the rapidly-exploring random tree (RRT) (LaValle and Kuffner, 2001) and the probabilistic roadmap (PRM) (Kavraki et al., 1996), has increased enormously in past years. Such algorithms have the advantage of quickly exploring a space of interest to achieve a feasible solution. The development of the RRT* and PRM* algorithms has led to algorithms that asymptotically approach the optimal solution with increasing computation time (Karaman and Frazzoli, 2011). Variants have also been proposed that have anytime capabilities and utilize branch and bound techniques to trim the search tree (Karaman et al., 2011). Recent work has shown that extensions of these sampling-based algorithms can solve problems with uncertainty in position while preserving asymptotic optimality guarantees (Bry and Roy, 2011). Similar algorithms have also been designed to generate low-cost cycles that provide persistent monitoring of key points in the environment (Lan and Schwager, 2013). These algorithms are concerned with minimizing a cost function without integral constraints along the trajectory. They do not consider the problem of maximizing an information metric subject to a budget constraint. As such, they cannot directly be used to solve informative motion planning problems.

Sampling-based algorithms are natural candidates for generating motion plans for information gathering tasks. The information-rich RRT (iRRT) was designed to maximize the accuracy of tracking a mobile target (Levine, 2010). The iRRT extends sampling-based algorithms to solve a class of information gathering problems. However, the authors do not provide asymptotic optimality guarantees, and the application is limited to tracking problems. In the current paper, we introduce the RIG algorithms that combine ideas from the iRRT (Levine, 2010), RRT* (Karaman and Frazzoli, 2011), RRBT (Bry and Roy, 2011), and combinatorial branch and bound (Binney and Sukhatme, 2012) algorithms to provide efficient information gathering for a rich space of objective functions.

3 Problem Setup

Informative motion planning on mobile platforms with finite resources requires solving the following maximization problem:

$$\mathcal{P}^* = \underset{\mathcal{P}\in\Psi}{\operatorname{argmax}}\ I(\mathcal{P}) \text{ s.t. } c(\mathcal{P}) \leq B, \tag{1}$$

where Ψ is the space of possible trajectories for a robot or team of robots, B is a budget (e.g., time, fuel, or energy), and $I(\mathcal{P})$ is a function representing the *information quality* gathered along the trajectory $\mathcal{P}$.[1] We will assume the the robots are modeled using discrete time dynamics, and that the trajectory is deterministic given the environment and the control inputs. As a notational convention, we will denote the portion of a trajectory from times t_0 to t_f as $\mathcal{P}^{t_0:t_f}$. Where it will not cause confusion, we will also denote a partial trajectory $\mathcal{P}^t$ as the segment of a trajectory centered around time t. We allow for restrictions on the space of valid trajectories, including kinodynamic constraints, obstacles, and vehicle speed limitations.

[1] We note that the proposed algorithms can potentially be used to optimize any quantifiable metric of interest along the trajectory. The scope of this paper is limited to information gathering objectives, but the examination of additional objectives of a similar form is an interesting avenue for future work.

We assume in this paper that the form of the cost and information objectives are known a priori. Regarding the cost function, we assume it is strictly positive, monotonically increasing, bounded, and additive. If we let $\mathcal{P}_a$ be a partial trajectory and we let the $+$ denote the operation of concatenating two trajectories into a single trajectory, a cost function can be considered monotonically increasing if for all trajectories $\mathcal{P}_b$ that can be concatenated with $\mathcal{P}_a$, we have that $c(\mathcal{P}_a) < c(\mathcal{P}_a + \mathcal{P}_b)$. If we have two partial trajectories $\mathcal{P}_a$ and $\mathcal{P}_b$ that can be concatenated to yield a trajectory $\mathcal{P}_{ab}$, a cost function is considered additive if $c(\mathcal{P}_{ab}) = c(\mathcal{P}_a) + c(\mathcal{P}_b)$. These assumptions include most objective functions utilized in prior sampling-based motion planning literature (e.g., distance, energy, etc.) (Karaman and Frazzoli, 2011). Note that these assumptions ensure that all trajectories satisfying the budget will be of finite length.

We consider information objective functions of the following three types: modular, time-varying modular, and submodular. If we let $\mathcal{P}_a$ and $\mathcal{P}_b$ be two partial trajectories, and we let $\mathcal{P}_{ab}$ be the trajectory found by concatenating them, we can define a modular information objective as one where $I(\mathcal{P}_{ab}) = I(\mathcal{P}_a) + I(\mathcal{P}_b)$. A time-varying modular objective is the same, except that the value of $I(\mathcal{P}_{ab})$ depends on the time when $\mathcal{P}_a$ and $\mathcal{P}_b$ are executed. For instance, if t is an arbitrary start time, t_a is the duration of partial trajectory $\mathcal{P}_a$, a time-varying modular objective would have $I(\mathcal{P}_{ab}, t) = I(\mathcal{P}_a, t) + I(\mathcal{P}_b, t + t_a)$. In contrast, a submodular objective is one where $I(\mathcal{P}_{ab}) + I(\mathcal{P}_{a \cap b}) \leq I(\mathcal{P}_a) + I(\mathcal{P}_b)$, where $\mathcal{P}_{a \cap b}$ is the intersection of trajectories $\mathcal{P}_a$ and $\mathcal{P}_b$. In the context of robot motion planning, one key property of submodular objectives is that the amount of information gathered in the future is dependent on the prior trajectory, whereas with modular objectives it is not (see Krause and Guestrin (2011) for a more thorough discussion of submodularity and its properties).

Two major factors make solving these optimization problems particularly difficult: (1) for nearly all interesting objective functions, finding the optimal solution is formally hard (NP-hard or PSPACE-hard) (Reif, 1979a; Singh et al., 2009), and (2) the space of trajectories Ψ grows with increasing budget, making exhaustive searches intractable. We propose three variants of incremental algorithms that all utilize sampling to generate increasingly informative trajectories satisfying the cost budget constraints. These sampling-based methods allow for the generation of informative trajectories that maximize complex information quality objectives.

4 Algorithm Descriptions

We now discuss the proposed motion planning algorithms that generate trajectories to maximize an information quality metric while also maintaining budget constraints. The key idea is to sample the configuration space of the vehicle and to build up a tree or graph of possible trajectories by extending candidate trajectories towards the sampled points. Unlike many prior motion planning algorithms, where vertices in the graph represent locations in the environment, vertices in the RIG tree/graph represent a tuple of location, cost, and information. We will refer to these tuples as "nodes" in the tree/graph.

We propose three variants of the RIG algorithm: (1) one that builds up a roadmap of possible trajectories (RIG-roadmap), (2) one with a fixed start point that incrementally extends a tree of possible trajectory (RIG-tree), and (3) one with a fixed start point that builds a graph of solution trajectories (RIG-graph). These variants respectively are inspired by the PRM*, RRT*, and RRG algorithms from prior work (Karaman and Frazzoli, 2011). We note, however, that prior versions of PRM*, RRT*, and RRG have not been applied to problems where the constraints are integrated over the trajectory (e.g., budget constraints). The key difference is that the problem at hand optimizes an information quality metric subject to a cost constraint rather than optimizing a cost metric subject to local constraints (e.g., obstacles).

The RIG-roadmap variant is described in Algorithm 1. The idea behind this variant is to generate a number of possible samples (lines 5–8) in the configuration space of the vehicle in a pre-processing step and then step through these samples to connect them into trajectories (lines 9–36). The RIG-roadmap differentiates itself from the other variants in that the number of roadmap locations is specified in advance, and the connections between the locations and resulting trajectories are built afterwards. This approach allows the user to control the effective density of the locations prior to the computationally expensive step of building the roadmap connections.

The RIG-tree variant is is described in Algorithm 2. The main loop of the algorithm begins with a start node and then incrementally generates randomly sampled points in the configuration space. For each new point that is sampled, the algorithm extends the nearest node on the tree to create a new node (lines 7–8). All nearby nodes are then extended towards the new node to generate a tree of solutions (lines 10–29). Nodes in the tree are only extended if the resulting cost does not exceed the budget (line 24), and nodes that are not promising can be pruned (line 18, see Section 4.3 for more detail). The main advantage of the RIG-tree variant is that it substantially reduces the number of stored trajectories by maintaining a tree of solutions and extending them incrementally. However, these solutions only represent a subset of those contained in the full graph, which can lead to reduced information quality of the final trajectory.

The RIG-graph variant is described in Algorithm 3. The major difference between RIG-graph and RIG-tree is that additional edges are added between nodes to form a graph structure. Similarly to RIG-tree, points in the space are sampled (line 7), and the nearest node on the graph is extended towards the sampled point (line 8). After near nodes are extended towards the new node (lines 10–13), the information and cost is recursively updated along all the new edges (lines 16-34). During this recursive update, new nodes are generated to store the information and cost generated by the trajectories formed from the newly added edges (line 25). More nodes are then generated at the neighbors of these newly added nodes and so on until all eligible nodes have been expanded (termination condition on line 16).[2] Pruning can also be employed during this step to limit the number of nodes added (line 21). After the algorithm has run for many iterations, the graph will contain a number of possible informative trajectories within the budget constraint.

All of the RIG variants are designed to be run in an "anytime" fashion where they continue to improve the solution until terminated by an operator. The operator may choose to terminate the algorithm after an allotted amount of processing time has passed or if the solution no longer seems to be improving. The development of more sophisticated termination conditions is left as an avenue for future work.

The RIG variants allow for constraints on the vehicle's motion through the use of a Steer() function to extend nodes towards newly sampled locations. Requirements on the steer function are discussed in the next section. Candidate points on the graph to be extended are identified using a Near() function. The near function can be heuristically set or set based on pre-specified ball around the sampled point (as in (Karaman and Frazzoli, 2011)). The algorithm can also account for obstacles in the environment by limiting feasible trajectories to a free configuration space $\mathcal{X}_{free}$.

The main challenges presented by the information gathering problem are (1) focusing the graph

[2]This step requires that the vehicle be able to effectively "reverse" its movements. If dynamics are considered, this may not be possible, and this step would need to use an additional call to the Steer() function.

Algorithm 1 Rapidly-exploring Information Gathering Roadmap (RIG-Roadmap)

```
1: Input: Step size $\Delta$, Budget $B$, Workspace $\mathcal{X}_{all}$, Free space $\mathcal{X}_{free}$, Environment $\mathcal{E}$, Start configuration $\mathbf{x}_{start}$, Near radius $R$, Number of samples $K$
2: % Initialize cost, information, starting node, node list, edge list, and roadmap locations list
3: $I_{init} \leftarrow InitialInformation(\mathbf{x}_{start}, \mathcal{E})$, $C_{init} \leftarrow 0$, $n \leftarrow \langle \mathbf{x}_{start}, C_{init}, I_{init} \rangle$
4: $V \leftarrow \{n\}, V_{closed} \leftarrow \emptyset$, $V_{roadmap} \leftarrow \emptyset$, $E \leftarrow \emptyset$
5: for 1 to K do
6:    % Sample configuration space of vehicle
7:    $\mathbf{x}_{roadmap} \leftarrow Sample(\mathcal{X}_{all})$, $V_{roadmap} \leftarrow V_{roadmap} \cup \{\mathbf{x}_{roadmap}\}$
8: end for
9: while not terminated do
10:    for all $\mathbf{x}_{samp} \in V_{roadmap}$ do
11:       % Find nearest node
12:       $n_{nearest} \leftarrow Nearest(\mathbf{x}_{samp}, V \setminus V_{closed})$
13:       $\mathbf{x}_{feasible} \leftarrow Steer(\mathbf{x}_{n_{nearest}}, \mathbf{x}_{samp}, \Delta)$
14:       % Find near points within radius
15:       $N_{near} \leftarrow Near(\mathbf{x}_{feasible}, V \setminus V_{closed}$,R)
16:       for all $n_{near} \in N_{near}$ do
17:          % Extend towards new point
18:          $\mathbf{x}_{new} \leftarrow Steer(\mathbf{x}_{n_{near}}, \mathbf{x}_{feasible}, \Delta)$
19:          if $NoCollision(\mathbf{x}_{n_{near}}, \mathbf{x}_{new}, \mathcal{X}_{free})$ then
20:             % Calculate new information and cost
21:             $I_{new} \leftarrow Information(I_{n_{near}}, \mathbf{x}_{new}, \mathcal{E})$, $c(\mathbf{x}_{new}) \leftarrow EvaluateCost(\mathbf{x}_{n_{near}}, \mathbf{x}_{new})$
22:             $C_{new} \leftarrow C_{n_{near}} + c(\mathbf{x}_{new})$, $n_{new} \leftarrow \langle \mathbf{x}_{new}, C_{new}, I_{new} \rangle$
23:             if PRUNE($n_{new}$) then
24:                Delete $n_{new}$
25:             else
26:                % Add edges and node
27:                $E \leftarrow E \cup \{(n_{near}, n_{new}\}$, $V \leftarrow V \cup \{n_{new}\}$
28:                % Add to closed list if budget exceeded
29:                if $C_{new} > B$ then
30:                   $V_{closed} \leftarrow V_{closed} \cup \{n_{new}\}$
31:                end if
32:             end if
33:          end if
34:       end for
35:    end for
36: end while
37: return $G = (V, E)$
```

generation such that candidate trajectories satisfy the budget requirements, (2) managing computational complexity for computing the information gathered at each node on the graph, and (3) pruning partial trajectories that cannot lead to a more informative final trajectory than the current best. We now discuss how the RIG variants overcome these challenges.

Algorithm 2 Rapidly-exploring Information Gathering Tree (RIG-Tree)

```
1: Input: Step size $\Delta$, Budget $B$, Workspace $\mathcal{X}_{all}$, Free space $\mathcal{X}_{free}$, Environment $\mathcal{E}$, Start configuration $\mathbf{x}_{start}$, Near radius $R$
2: % Initialize cost, information, starting node, node list, edge list, and tree
3: $I_{init} \leftarrow InitialInformation(\mathbf{x}_{start}, \mathcal{E})$, $C_{init} \leftarrow 0$, $n \leftarrow \langle \mathbf{x}_{start}, C_{init}, I_{init} \rangle$
4: $V \leftarrow \{n\}, V_{closed} \leftarrow \emptyset$, $E \leftarrow \emptyset$
5: while not terminated do
6:   % Sample configuration space of vehicle and find nearest node
7:   $\mathbf{x}_{samp} \leftarrow Sample(\mathcal{X}_{all})$, $n_{nearest} \leftarrow Nearest(\mathbf{x}_{samp}, V \setminus V_{closed})$
8:   $\mathbf{x}_{feasible} \leftarrow Steer(\mathbf{x}_{n_{nearest}}, \mathbf{x}_{samp}, \Delta)$
9:   % Find near points to be extended
10:  $N_{near} \leftarrow Near(\mathbf{x}_{feasible}, V \setminus V_{closed}, R)$
11:  for all $n_{near} \in N_{near}$ do
12:    % Extend towards new point
13:    $\mathbf{x}_{new} \leftarrow Steer(\mathbf{x}_{n_{near}}, \mathbf{x}_{feasible}, \Delta)$
14:    if $NoCollision(\mathbf{x}_{n_{near}}, \mathbf{x}_{new}, \mathcal{X}_{free})$ then
15:      % Calculate new information and cost
16:      $I_{new} \leftarrow Information(I_{n_{near}}, \mathbf{x}_{new}, \mathcal{E})$, $c(\mathbf{x}_{new}) \leftarrow EvaluateCost(\mathbf{x}_{n_{near}}, \mathbf{x}_{new})$
17:      $C_{new} \leftarrow C_{n_{near}} + c(\mathbf{x}_{new})$, $n_{new} \leftarrow \langle \mathbf{x}_{new}, C_{new}, I_{new} \rangle$
18:      if PRUNE($n_{new}$) then
19:        Delete $n_{new}$
20:      else
21:        % Add edges and node
22:        $E \leftarrow E \cup \{(n_{near}, n_{new}\}$, $V \leftarrow V \cup \{n_{new}\}$
23:        % Add to closed list if budget exceeded
24:        if $C_{new} > B$ then
25:          $V_{closed} \leftarrow V_{closed} \cup \{n_{new}\}$
26:        end if
27:      end if
28:    end if
29:  end for
30: end while
31: return $\mathcal{T} = (V, E)$
```

4.1 Budgeted Trajectory Generation

The problem of maximizing information subject to a budget constraint differs in several ways from problems typically solved using sampling-based motion planners. In many problem domains, a vehicle must move from one point to another while minimizing the trajectory cost (Karaman and Frazzoli, 2011). For the problem of information optimization, there is no fixed goal point. Instead, there is a hard constraint on budget at which point the mission time has expired or the vehicle has run out of fuel.[3]

[3]We note that our proposed algorithms can also be utilized if a fixed goal point is specified. In this case, nodes would enter the closed list when the minimum cost to reach the goal exceeds the remaining budget.

Algorithm 3 Rapidly-exploring Information Gathering Graph (RIG-Graph)

```
1: Input: Step size $\Delta$, Budget $B$, Workspace $\mathcal{X}_{all}$, Free space $\mathcal{X}_{free}$, Environment $\mathcal{E}$, Start configuration $\mathbf{x}_{start}$, Near radius $R$
2: % Initialize cost, information, starting node, node list, edge list, and graph
3: $I_{init} \leftarrow InitialInformation(\mathbf{x}_{start}, \mathcal{E})$, $C_{init} \leftarrow 0$, $n \leftarrow \langle \mathbf{x}_{start}, C_{init}, I_{init} \rangle$
4: $V \leftarrow \{n\}, V_{closed} \leftarrow \emptyset$, $E \leftarrow \emptyset$
5: while not terminated do
6:    % Sample configuration space of vehicle and find nearest node
7:    $\mathbf{x}_{samp} \leftarrow Sample(\mathcal{X}_{all})$, $n_{nearest} \leftarrow Nearest(\mathbf{x}_{samp}, V \setminus V_{closed})$
8:    $\mathbf{x}_{feasible} \leftarrow Steer(\mathbf{x}_{n_{nearest}}, \mathbf{x}_{samp}, \Delta)$
9:    % Find near points to be extended
10:   $N_{near} \leftarrow Near(\mathbf{x}_{feasible}, V \setminus V_{closed}, R)$
11:   for all $n_{near} \in N_{near}$ do
12:      % Extend towards new point
13:      $\mathbf{x}_{new} \leftarrow Steer(\mathbf{x}_{n_{near}}, \mathbf{x}_{feasible}, \Delta)$
14:      if $NoCollision(\mathbf{x}_{n_{near}}, \mathbf{x}_{new}, \mathcal{X}_{free})$ then
15:         $Q \leftarrow \{\mathbf{x}_{new}\}$
16:         while $Q \neq \emptyset$ do
17:            $\mathbf{x}_q \leftarrow Pop(Q)$, $N_{qnear} \leftarrow Near(\mathbf{x}_q, V \setminus V_{closed}, R)$
18:            for all $n_{qnear} \in N_{qnear}$ do
19:               $I_q \leftarrow Information(I_{n_{qnear}}, \mathbf{x}_q, \mathcal{E})$, $c(\mathbf{x}_q) \leftarrow EvaluateCost(\mathbf{x}_{n_{qnear}}, \mathbf{x}_q)$
20:               $C_q \leftarrow C_{n_{qnear}} + c(\mathbf{x}_q)$, $n_{added} \leftarrow \langle \mathbf{x}_q, C_q, I_q \rangle$
21:               if PRUNE($n_{added}$) then
22:                  Delete $n_{added}$
23:               else
24:                  % Add edges and node
25:                  $E \leftarrow E \cup \{(n_{qnear}, n_{added})\}$, $V \leftarrow V \cup \{n_{added}\}$
26:                  % Add to closed list if budget exceeded
27:                  if $C_q > B$ then
28:                     $V_{closed} \leftarrow V_{closed} \cup \{n_{added}\}$
29:                  else
30:                     $Q \leftarrow Q \cup \{n_{added}\}$
31:                  end if
32:               end if
33:            end for
34:         end while
35:      end if
36:   end for
37: end while
38: return $G = (V, E)$
```

To apply sampling-based motion planners to these problems, we make the following modification: if a candidate trajectory would exceed the budget, it will never be extended towards a sampled point. All three RIG variants maintain a *closed list* of nodes that represent completed trajectories.

This modification also allows the proposed methods to be used in a receding-horizon manner by planning over a budget increment and then replanning after the budget increment is expended. As new points are generated, trajectories that are not completed will be extended and eventually lead to completed trajectories. This approach builds up a large number of completed trajectories that efficiently explore the space of trajectories and provide different levels of information maximization.

4.2 Computational Complexity

As described above, we assume that when given a partial trajectory $\mathcal{P}^{t_0:t_f}$ for times t_0 to t_f, we can calculate the information gathered using some known function $I(\mathcal{P}^{t_0:t_f})$. For modular and time-varying modular functions, it is straightforward to build trajectories in an incremental fashion by storing the information gathered at $t-1$ and then adding the incremental information gathered at t. For submodular functions, the entire partial trajectory is necessary to calculate the information at time t. Thus, the trajectory must be reconstructed by traversing the graph backwards. This traversal is an $O(N)$ operation, where N is the number of nodes.

For some objective functions, the cost of calculating the information gathered may also increase as the length of the trajectory grows. The computational requirement for adding a new node grows as $O(N + f(N))$, where $f(N)$ is the computational cost of calculating $I(\cdot)$ for a trajectory of length N. For complex objective functions, $f(N)$ could be exponential in N, which limits the applicability of the approach. For many modular objectives, $f(N)$ is simply an $O(N)$ counting operation along the trajectory. For submodular objectives, $f(N)$ may be polynomial in N (e.g., variance reduction in a Gaussian process or minimization of Fisher information). If the cost of calculating the information gathered along a trajectory is substantial, more aggressive pruning approaches may need to be employed to reduce the number of nodes, or approximation may be used to estimate the information gathered without calculating it exactly.

The locations of the active nodes are stored in a KD-tree, which allows for efficient retrieval of the near nodes. The memory requirements grow linearly in the number of nodes since the stored information remains constant. The computational complexity of the steer function must also be considered, but it will typically not grow in the number of nodes. One of the key differences between RIG-roadmap, RIG-tree, and RIG-graph is the way in which they keep the number of nodes in the tree low. RIG-roadmap allows for the number of locations to be specified beforehand, RIG-tree only adds trajectories that form a tree, and RIG-graph incrementally increases the number of locations until memory or processing time are exhausted. We will demonstrate the advantages and disadvantages of each of these approaches in Section 5.

Despite the ability of the RIG variants to keep the number of nodes low, the computational and memory requirements can still become infeasible, particularly in high-dimensional configuration spaces. Since all of the near nodes are extended for each new sample, denser graphs will add more nodes per iteration than sparser graphs. This additional computation provides motivation for pruning away nodes that are no longer useful for finding the best information gathering trajectory. We will next discuss how to develop pruning strategies using formal analysis of asymptotic optimality.

4.3 Theoretical Analysis

In this section, we show that the RIG-graph variant is asymptotically optimal for stationary modular objective functions, time-varying modular objective functions, and submodular objective functions.

The asymptotic optimality requires that a *conservative* pruning strategy is used that does not remove nodes that could potentially lead to the most informative plan.

We first state a number of modest assumptions that are required for this analysis. These assumptions are adapted from the rapidly-exploring random belief tree (RRBT) (Bry and Roy, 2011) and the rapidly-exploring random graph (RRG) (Karaman and Frazzoli, 2011) algorithms.

Assumption 1 *Let x^a, x^b, and x^c be three points within radius Δ of each other. Let the trajectory e_1 be generated by $Steer(x^a, x^c, \Delta)$, e_2 be generated by $Steer(x^a, x^b, \Delta)$, and e_3 be generated by $Steer(x^b, x^c, \Delta)$. If $x^b \in e_1$, then the concatenated trajectory $e_2 + e_3$ must be equal to e_1 and have equal cost and information.*

This assumption is nearly equivalent to the assumption required for RRBT (Bry and Roy, 2011) except that we require that both the cost and information be equal for the concatenated trajectory. The assumption states that the Steer() function is consistent for intermediate points and that both the cost and information functions are also consistent. This assumption is necessary because of the refinement as additional samples are added, which in the limit leads to samples that are infinitely close together. For general robot dynamics, this property requires that we can simulate the continuous system for any intermediate time step.

Assumption 2 *There exists a constant $r \in \mathbb{R}_+$ such that for any point $x^a \in \mathcal{X}_{free}$ there exists an $x^b \in \mathcal{X}_{free}$, such that (i) the ball of radius r centered at x^a lies inside $\mathcal{X}_{free}$ and (ii) x^a lies inside the ball of radius r centered at x^b.*

This assumption is equivalent to the corresponding assumption in the RRG and RRT* algorithms (Karaman and Frazzoli, 2011) that requires that some free space is available around the optimal trajectory to allow for convergence. Finally, we require the following assumptions on the sampling function.

Assumption 3 *Points returned by the sampling function Sample() are i.i.d. and drawn from a uniform distribution.*[4]

We now have the following lemma regarding the asymptotic performance of RIG-graph without pruning.

Lemma 1 *Let Ψ^B denote the set of all finite length trajectories through $\mathcal{X}_{all}$ that satisfy a budget B such that for every $x \in \mathcal{P}$ and $\mathcal{P} \in \Psi^B$, $x \in \mathcal{X}_{free}$. Let Ψ_i^B denote the set of trajectories contained in the graph built by RIG-graph at iteration i for budget B if pruning is not employed. We have that $\lim_{i\to\infty} \Psi_i^B = \Psi^B$.*

Proof This claim follows from Assumptions 1, 2, and 3 along with the analysis of the asymptotic optimality of RRBT in (Bry and Roy, 2011). From Lemma 1 in Bry and Roy (2011), the following properties are necessary for all feasible trajectories to be generated in an RRBT graph: (1) an infinitely dense connected graph must be created around each point in $\mathcal{X}_{free}$, and (2) all trajectories must be of finite length. We have condition (2) from the assumption that the cost function is

[4]Results should also extend to any absolutely continuous sampling distribution with density bounded away from zero on $\mathcal{X}_{all}$ (as in (Karaman and Frazzoli, 2011)).

monotonically increasing and additive. For condition (1), the graph within a ball around a point in $\mathcal{X}_{free}$ approaches infinite density if the distance between adjacent nodes approaches zero, and the graph within the ball remains fully connected. If there is sufficient space to sample within a ball around all points in $\mathcal{X}_{free}$ then, in the limit with Assumption 3, the density of the graph within the ball around all points will approach infinity and will remain fully connected. As in Bry and Roy (2011), if the Near() function returns every node within a ball of radius $r \propto (log(n)/n)^{1/d}$, where n is the number of uniquely located nodes in the graph and d is the dimensionality of the state, then using Assumption 2, we have that such a ball exists around all points. Additionally, Assumption 1 states that the Steer() actions can be infinitely subdivided, which means that all trajectories of infinitesimal cost will be generated by it. Thus, in the limit, the graph density around each point in $\mathcal{X}_{free}$ approaches infinity, and the graph remains fully connected within the ball. Finally, if a cost budget is substituted for the chance constraint in RRBT, the trajectories generated by RIG-graph are equivalent to those generated by RRBT, since the node generation and connection techniques are the same for the two algorithms when pruning is not employed. ■

From Lemma 1, we know that the RIG-graph algorithm will produce all possible budget-constrained trajectories in the limit. Thus, in the limit it will produce the optimal plan (i.e., it is asymptotical optimal). We now examine requirements for pruning strategies that preserve asymptotic optimality. We will first need an additional definition regarding the relationship between two nodes.

Definition 1 *Let $I(n)$ be the information gathered by the trajectory leading to node n, $c(n)$ be the cost along the trajectory to node n, and $x(n)$ be the point in the configuration space of node n. Given two nodes $n_a = \langle x(n_a), c(n_a), I(n_a)\rangle$ and $n_b = \langle x(n_b), c(n_b), I(n_b)\rangle$, these nodes are considered to be co-located if $x(n_a) = x(n_b)$.*

Co-located nodes arise when the Near() function is called to extend multiple nodes towards a new node and several of the near nodes are within the step size of the steer function. In practice, two nodes can be considered to be co-located if they are within some small distance ϵ. We also define an ordering relation between two co-located nodes, denoted as n_a and n_b. The ordering $n_a > n_b$ implies that n_b should be pruned if it is co-located with n_a. A precise specification of the ordering relation is not required as long as it satisfies the condition in Proposition 1 below. We give several examples of ordering relationships that can be used for pruning in the following section. We now have the following proposition regarding the asymptotic optimality of RIG-graph.

Proposition 1 *Given two nodes n_a and n_b that are co-located with different cost and information values. Let p_a^g be the maximally informative partial trajectory originating from n_a that satisfies the remaining budget. Similarly, let p_b^g be the maximally informative partial trajectory originating from n_b that satisfies the remaining budget. If a pruning strategy is employed that removes co-located nodes that are dominated by a partial ordering $n_a > n_b$, RIG-graph is asymptotically optimal if the following condition holds:*

$$n_a > n_b \Rightarrow I(p_a^g) + I(n_a) > I(p_b^g) + I(n_b). \tag{2}$$

Proof The condition above states that a partial ordering exists such that the most informative final trajectory from n_b is always less informative than the most informative final trajectory from n_a. Thus, no maximally informative trajectory will ever be disallowed by pruning as long as the partial ordering is upheld. The result now directly follows from Lemma 1, which states that all feasible trajectories satisfying the budget will be generated. ■

The asymptotic optimality of RIG-tree and RIG-roadmap are not as straightforward. To see why, notice that RIG-tree and RIG-roadmap with a decreasing Near() radius may fail to find optimal solutions that contain long looping trajectories. As the Near() radius decreases, nodes corresponding to long looping trajectories will be expanded progressively more slowly. We conjecture that RIG-tree and RIG-roadmap will be asymptotically optimal given that a sufficiently large radius is used for the Near() function relative to the length of the longest looping path in the solution. In Section 5, we show that all RIG variants converge to the optimal solution in practice.

4.4 Pruning of Nodes

In order to ensure that we generate the maximally informative trajectory in the limit, we need to guarantee that we do not remove any nodes that may lead to the optimal plan. When a new node is added to the graph, all near nodes are extended towards it. For a dense graph, the new node will be reached by multiple nodes in the graph, which will lead to multiple new partial plans with co-located endpoints. When this occurs, it is advantageous to prune nodes that cannot lead to the optimal solution. We now describe conservative pruning strategies to this effect.

For a modular objective function, it is straightforward to show that if n_a and n_b are co-located and $c(n_a) < c(n_b)$, then $I(p_a^g) \geq I(p_b^g)$. To see this, note that for a modular objective function the information landscape is not affected by the prior trajectory. Thus, a trajectory with more cost left to spend from the same point will always gather at least as much information. In this case, we can prune node n_b if an alternative co-located node n_a exists such that $I(n_a) > I(n_b)$ and $c(n_a) \leq c(n_b)$. This relationship defines one possible ordering of $n_a > n_b$. It is important to note that when nodes are pruned, the entire trajectory leading to them is not pruned. In fact, other nodes in the partial trajectory could be candidates for extension.

For submodular objective functions, the pruning strategy above may remove nodes that lead to the optimal trajectory. To see this, note that the maximally informative trajectory from a node $I(p_a^g)$ depends on the prior path taken to reach that node. Thus, we cannot guarantee that paths should be pruned even if they are dominated both in information and in cost. To achieve a conservative pruning strategy in this case, we must generate an upper bound on $I(p_a^g)$. We can do this by calculating the reachable information $I^{max}(n_a)$, i.e., the information within the remaining budget (Binney and Sukhatme, 2012). The reachable information can be calculated either through integration, sampling, or by enumeration (if the information function is evaluated at discrete points). We note that if the reachable information is calculated through approximation, the pruning strategy must be adjusted accordingly to preserve asymptotic optimality.

Given a way to calculate the reachable information, we can now prune node n_b if the following are true: it is co-located with node n_a, $I(n_a) > I^{max}(n_b) + I(n_b)$, and $c(n_a) \leq c(n_b)$. This relationship defines another potential ordering for $n_a > n_b$. In practice, the more aggressive pruning strategy for modular objectives can alternatively be used as a heuristic for submodular objectives to sacrifice asymptotic optimality for computational efficiency.

5 Experimental Results

We now discuss simulations and field tests validating the proposed informative motion planning algorithms. The simulations were run on a single desktop with a 3.2 GHz Intel i7 processor with 9 GB of RAM. The RIG algorithm variants were implemented in C++ on Ubuntu Linux. Nearest neighbor queries were provided by the Open Motion Planning Library (OMPL) (Şucan et al., 2012).

5.1 Comparison to Branch and Bound

We first compare the RIG-tree algorithm to a combinatorial branch and bound algorithm from prior work (Binney and Sukhatme, 2012). The branch and bound algorithm incrementally extends partial paths in a discrete space and maintains an upper bound on the solution quality of each partial path using the bounding strategy for calculating $I^{max}(\cdot)$ described above. The branch and bound algorithm is guaranteed to terminate with the optimal solution. As such, it provides an apt comparison to validate the scalability improvements of RIG as well as the advantage of operating in continuous space.

The simulations model a synthetic information gathering problem in a 10 km $\times$ 10 km environment. To provide a comparison with the branch and bound algorithm, the vehicle is given a simple motion model where it is assumed to move holonomically on a discrete 1 km grid. The RIG algorithm was modified such that nodes could only be created on a 1 km grid, which effectively yields a discrete variant of the algorithm. In some simulations, circular obstacles are placed randomly in the environment with radii (also chosen randomly) between 1 km to 5 km.

The information objective for these simulations was specified by the random placement of five Gaussian information sources. The intensity of each information source degrades exponentially with a randomly chosen length scale and intensity. In the time-varying case, the information sources move with a known trajectory. For the submodular case, no additional information is gathered from a point once it has been observed by the vehicle. Note that this restriction is taken into account during the planning phase, and the actual information landscape does not change. A changing information landscape would require online replanning, which is left as an avenue for future work. An example problem and solution trajectory is shown in Figure 1, and an accompanying video in Multimedia Extension 1 shows animations of growth over time for RIG-tree refining a trajectory with a submodular information objective.

Given the specifications described above, 100 random scenarios were chosen for each budget, and the RIG-tree algorithm was compared to the combinatorial branch and bound algorithm. RIG-tree is used in these experiments due to its simplicity, and we provide comparisons between the RIG variants in the next section. The use of a discrete environment allows the optimal solution to be found using the branch and bound solver. We are then able to compare the time it takes for RIG-tree to achieve an optimal solution. Figure 2 shows the computation time to find the optimal discrete path for increasing budget lengths. For the modular and time-varying modular objectives, the RIG-tree algorithm is able to find the optimal solution quickly even at higher budgets.

Here, the optimal solution was known from running the branch and bound algorithm. It is possible for multiple optimal solutions to exist with the same information quality. However, in nearly all of the cases observed, RIG-tree found the same solution as the branch and bound solver. In general, the optimal solution can be identified at the point where the graph stops improving. Refining an upper bound on optimal as the algorithm progresses is a potential area for future work.

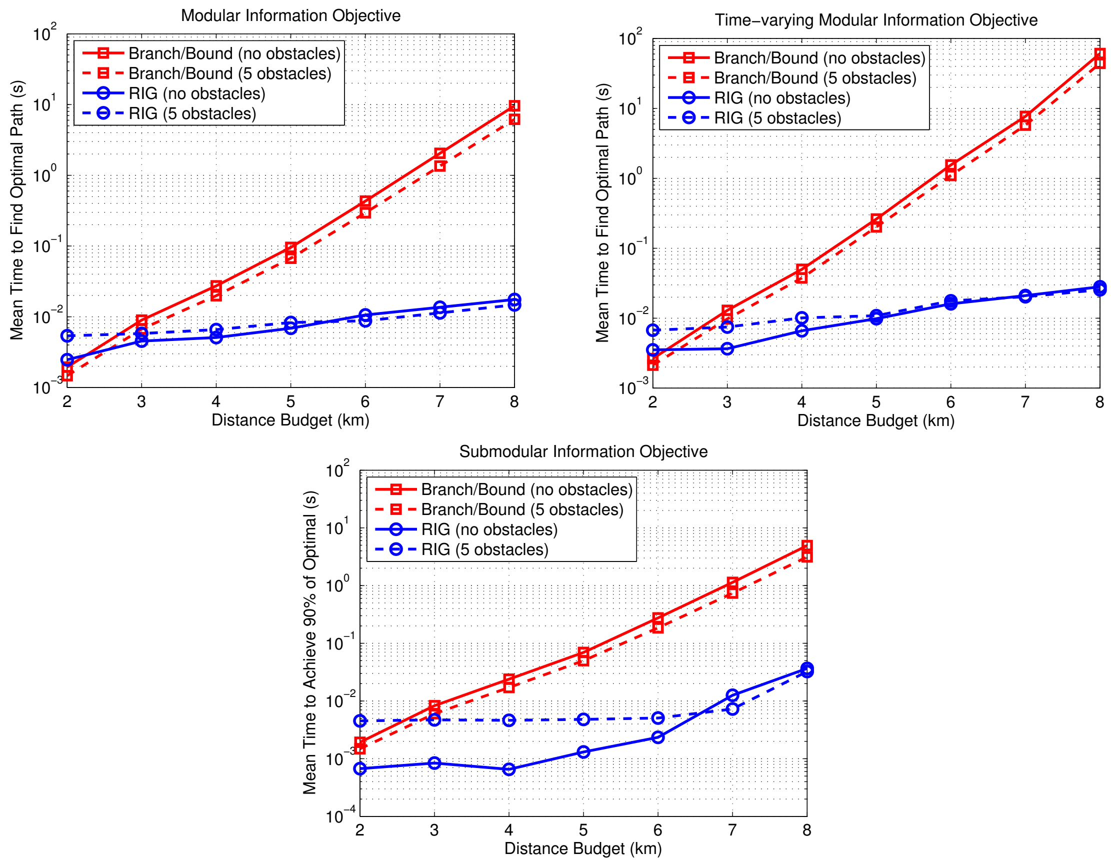


Figure 2: Simulated results for the RIG-tree algorithm in a 10 km $\times$ 10 km environment. Each data point is averaged over 100 simulated trials with a random information landscape derived using a mixture of Gaussians. For comparison to the discrete branch and bound algorithm, waypoints are restricted to fall on a 1 km grid. For modular and time-varying modular information objectives, the proposed RIG-tree algorithm quickly approaches the optimal discrete solution even at high budgets. For submodular objective functions, RIG-tree quickly approaches a near-optimal solution.

Adding obstacles to the environment can either increase or decrease the computation time. Obstacles restrict the number of possible paths through the environment, but they also can prevent the algorithm from converging quickly. With larger budgets, the computational gains from restricting the number of paths outweigh the additional convergence time as obstacles are added, and with smaller budgets the opposite occurs.

The gain in computation time for RIG-tree over the combinatorial algorithm is less pronounced for the submodular information objective, due to the pruning criterion necessary to guarantee convergence to the optimal solution. Initial simulations (not shown) demonstrated that RIG provides a marginal improvement over the combinatorial algorithm for finding the optimal solution. However, a strength of RIG-tree is that it rapidly explores the environment and quickly generates a near-optimal solution. Figure 2 shows significant improvement in computation time for achieving a solution within 90% of optimal (again the branch and bound solution is used to find the optimal solution in these discrete domains). In Section 5.3, we will also demonstrate how using the RIG-tree algorithm in continuous space can be used to improve path planning for optimizing a submodular information objective in a wireless signal strength mapping application.

5.2 Comparison of RIG Variants

We now examine the performance of the three RIG variants to determine the tradeoff between them. Simulations were run using the same problem domains as described above in the branch and bound comparisons. The discrete environments were used because they allow for finding the optimal solution using the branch and bound solver, which makes it possible to compare the mean time to reach optimal for the competing algorithms. Figure 3 shows a comparison of the performance of RIG-roadmap, RIG-tree, and RIG-graph for modular, time-varying modular, and submodular objective functions in environments with and without obstacles (budget = 8 km).

The comparisons provide the average time it takes for each RIG variant to achieve various percentages of the optimal solution. We see that in the modular and time-varying modular case, the three variants seem to perform competitively, with RIG-tree lagging behind somewhat. However, the RIG-tree variant is able to find the optimal solution more quickly than RIG-graph in all cases. The advantage of using the graph representation, as opposed to the tree representation, is apparent here because the graph representation builds up a richer space of paths more quickly. However, the graph representation does not present much advantage when a near-optimal solution is sought, due to the requirement for sampling more points to generate a broader coverage of the workspace.

In the case of a submodular objective function, the RIG-graph variant performs poorly relative to the other variants. In these cases, the pruning function allows for a large number of suboptimal trajectories to be generated by RIG-graph, which consumes processing time but does not yield a better solution. The RIG-roadmap and RIG-tree variants do not create as many trajectories relative to the number of sampled locations, which turns out to be advantageous in this case. The disadvantage of the RIG-roadmap is that the number of sampled points must be pre-specified, which can be difficult if the necessary point density is not known beforehand. Thus, RIG-tree seems to provide the most reliable performance without the necessary tuning parameter.

We also provide results on the effectiveness of pruning when running the proposed algorithms (Figure 4). With RIG-roadmap, the increase in nodes has a "rise-and-plateau" quality that stems from iterating through the locations in the roadmap. When roadmap locations near the previously built trajectory are queried, it leads to a large increase in the number of nodes. In contrast, when roadmap locations far from the built trajectory are queried, the number of nodes increases more slowly. For RIG-tree, pruning leads to a reduction in the number of nodes and a reduction in the iterations to find the optimal solution (optimal was found previously using the branch and bound algorithm). Finally, when RIG-graph is used, pruning can actually increase the number of required iterations. This is due to the additional nodes that would have been pruned being extended and then subsequently generating new nodes through the Near() function. Some of these node extensions may lead to an optimal trajectory that otherwise would not have been built until a later iteration. We note, however, the benefit of reducing the iterations is outweighed by the increase in running time caused by the additional nodes.

Figure 4 also provides plots of the running times and percent of optimal achieved for the three RIG variants versus the number of iterations run. We see from these results that RIG-graph has the highest running time by far, and RIG-roadmap and RIG-tree have competitive running times. It is also apparent that pruning reduces the running time significantly for RIG-graph, but does

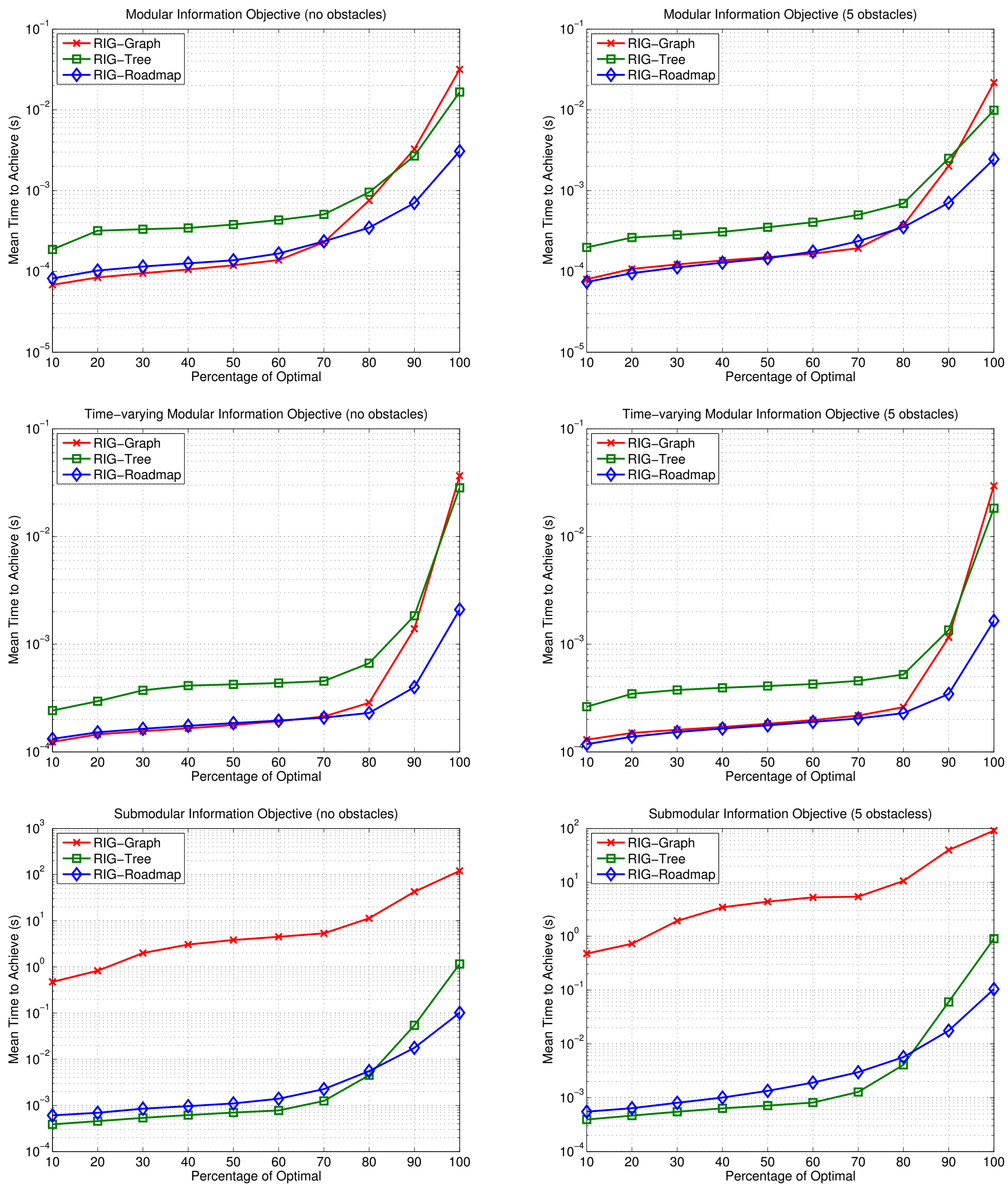


Figure 3: Simulated results for the three RIG variants in a 10 km × 10 km environment with a budget of 8 km. Each data point is averaged over 100 simulated trials with a random information landscape derived using a mixture of Gaussians.

not present as much of a benefit for RIG-tree and RIG-roadmap. We do note, however, that the running time upon finding the optimal solution is smaller in all cases when pruning is used. Another interesting observation from these results is that pruning does not have an effect on the number of

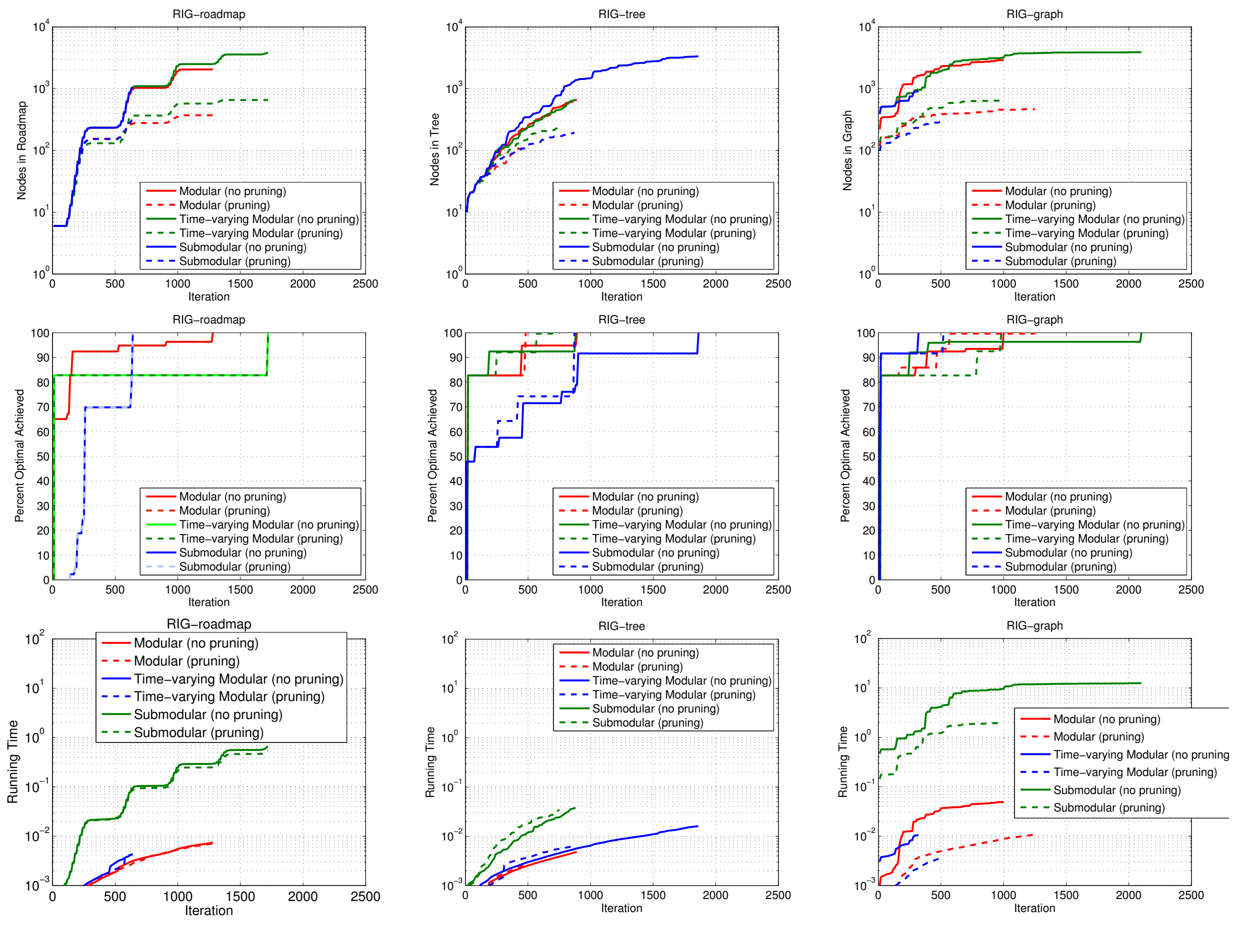


Figure 4: Number of iterations versus number of nodes in the tree (top), percent of optimal achieved (middle), and running time (bottom) using the RIG-roadmap, RIG-tree, and RIG-graph algorithms in a 10 km $\times$ 10 km environment with budget of 5 km. The plots are terminated after the optimal solution (previously determined using a branch and bound solver) is found. Note that the pruned and unpruned paths achieve the same percent optimal at each iteration for RIG-roadmap.

iterations to achieve optimal for RIG-roadmap (the pruned and unpruned solutions are equivalent relative to number of iterations). This occurs because the roadmap provides a systematic way of generating the nodes, which is not changed by pruning. On the other hand, the number of nodes that have been generated on the roadmap and the running time are both reduced by pruning.

Figure 5 shows a visual comparison of the three RIG variants progressively building up nodes and trajectories for an information gathering task with a submodular information objective (i.e., observing a location multiple times does not gather more information). For this example, RIG-roadmap generates fewer nodes and fewer connections due to the fixed sampling locations in the roadmap limiting the search space. In contrast, RIG-graph generates the most connections between nodes by building up a large number of trajectories through the recursive update. It is interesting to observe that some of the trajectories generated by RIG-graph actually loop back on themselves (as seen in the current best trajectory in iterations 10 and 110). RIG-tree strikes a balance by

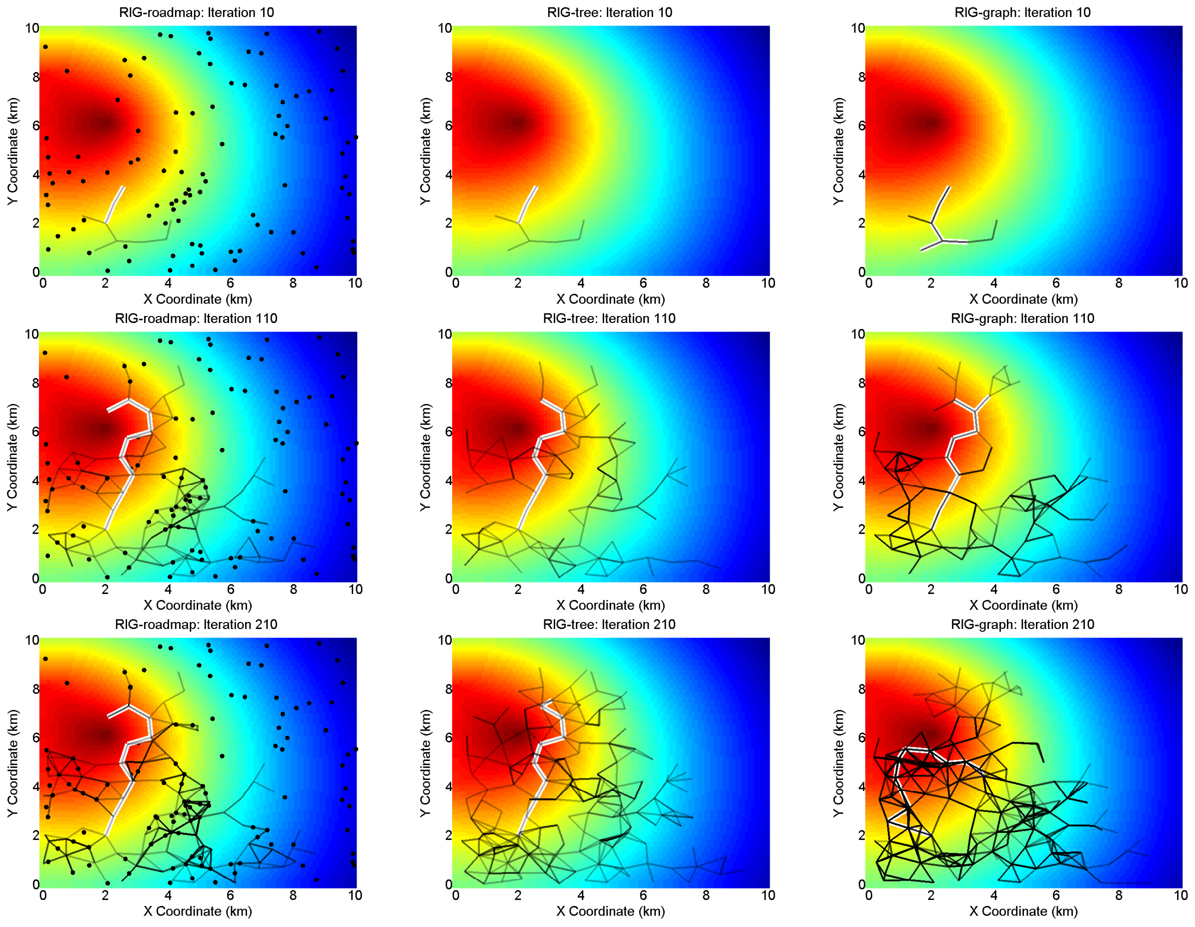


Figure 5: Comparison of the three RIG variants building up trajectories to maximize a submodular information objective (i.e., additional information is not gathered by observing a location multiple times). Pruning is used here to limit the number of nodes. The connections between nodes are shown as black lines (darker the more connections are made between those nodes). The current best trajectory is highlighted in white. For RIG-roadmap, the points on the roadmap are also shown as black dots. Red areas have high information quality, and blue areas have lower information quality. The starting point is at coordinate (2,2).

providing a tree that fills the space but does not generate as many connections as RIG-graph. All three variants are able to find a trajectory that explores the area of high information without exceeding the budget.

5.3 Wireless Signal Strength Mapping

We now demonstrate our proposed approach using a lake monitoring scenario with an autonomous surface vehicle (ASV). The ASV is propeller driven and is capable of moving at speeds up to 2 knots. It is equipped with a GPS unit and a Doppler Velocity Log (DVL), which provide localization capabilities for the vehicle. The vehicle communicates with a ground station through a standard 802.11 wireless connection. The vehicle is capable of measuring the wireless signal strength at its current location in dBm. Experiments were conducted at Puddingstone Lake in San Dimas, CA (Lat. 34.088854°, Lon. -117.810667°).

The vehicle is maneuverable enough to accurately follow waypoints given by a planner. As a result, we are able to generate trajectories in 2D space, which significantly reduces the complexity of the planning problem versus planning in the full 6D space of the vehicle's position, orientation, and velocities. Turning restrictions were incorporated into the Steer() function to ensure that the trajectories are executable. A minimum turning radius of 10 meters was assumed.[5] The RIG-tree algorithm was used in these experiments due to (1) the tree variant's ability to reduce the number of trajectories versus the other two RIG variants, and (2) the size of the planning space being large enough to prevent convergence in a reasonable amount of time.

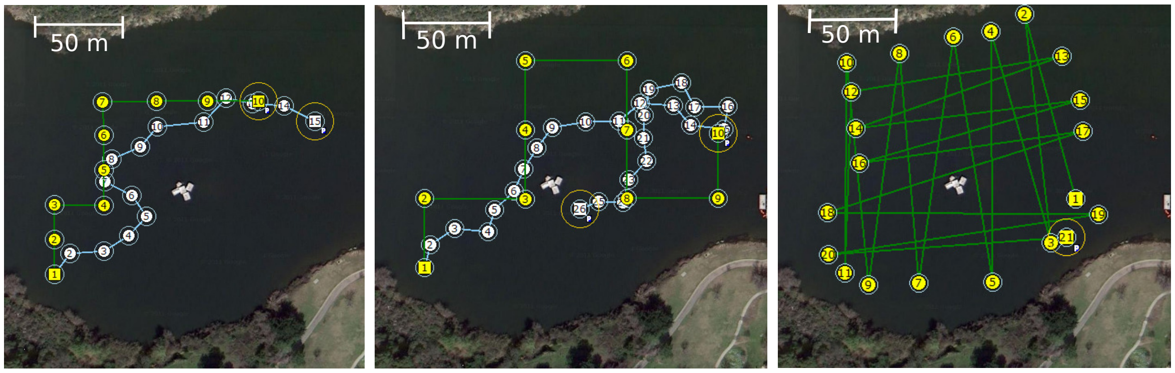
The goal in this experiment is to reconstruct the wireless signal strength over the entire area of interest, which would be useful for constructing an ad hoc network and planning surfacing locations for underwater exploration missions. In the context of informative motion planning, we want to minimize the root mean square error (RMSE) of the wireless signal strength predictions over the area of interest given a budget on the trajectory length. We note that there is no straightforward way to calculate the expected RMSE after executing a given trajectory due to the difficulty in predicting fluctuations in signal strength. As an alternative, we use a surrogate metric for planning that correlates with RMSE.

We utilize nonparametric regression in the form of Gaussian processes (GPs) to provide a prediction of the wireless signal strength across the area of interest. For simplicity, we employ a standard squared exponential kernel (Rasmussen and Williams, 2006) that captures the fact that predictions that are nearer to each other are more correlated than those that are further apart. We use the predicted variance from the GP as a proxy for RMSE, which has been shown in prior work to correlate (Vasudevan et al., 2009). Hyperparameters were determined using standard gradient ascent on the marginal likelihood. Due to computational limitations, a downsampled subset of 1000 points was used to learn the hyperparemeters. For inference on the GP, a local approximation is employed where only the 100 nearest data points are used to calculate the prediction and variance. For a given trajectory, we can now calculate the expected reduction in variance over some sampled set of points in the space of interest. Using this expected reduction in variance as an information quality metric fully defines an informative motion planning problem.

Trajectories were planned for budgets of 200 m and 400 m using RIG-tree and the discrete branch and bound algorithm (shown in Figure 6). RIG-tree was run for 1 minute using the aggressive pruning strategy that prunes all paths that are dominated in both distance and information. We note that the variance reduction information objective is not modular, which means this pruning strategy is a heuristic here. The branch and bound algorithm was run on the finest grid possible that would lead to completion in less than 1 minute (22 m × 22 m for the 200 m path and 44 m × 44 m for the 400 m path). Both algorithms were given a 1 minute running to time to allow for a fair comparison between them. The trajectories were then uploaded to the vehicle and executed. The vehicle did not replan the trajectory online, though this is a particularly interesting avenue for future work.

The vehicle measured the wireless signal strength at a rate of 1 Hz along the trajectory. The 200 m paths were executed in approximately 5 minutes (300 data points), and the 400 m trajectories

[5]We note that it may be possible to generate a trajectory in 6D that gathers more information for a given trajectory length than the optimal trajectory in 2D. Further analysis of full 6D planning with RIG is an interesting avenue for future work.



(a) Budget = 200 m (b) Budget = 400 m (c) Full survey = 2500 m

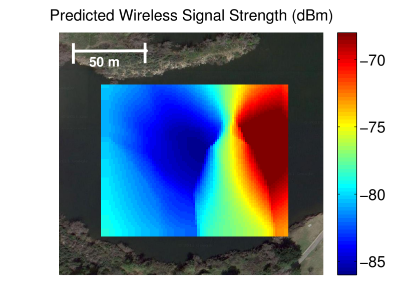
Figure 6: **Left and Center:** Trajectories taken by the branch and bound algorithm (shown as yellow waypoints) versus the RIG-tree algorithm (shown as white waypoints) by an autonomous vehicle monitoring wireless signal strength in a lake. The RIG-tree algorithm is able to operate in continuous space to provide a richer set of possible trajectories for monitoring the environment. **Right:** The full survey is also shown (planned by a human operator), which was used to compare the predictive accuracy after executing the autonomously planned trajectories.

were executed in approximately 10 minutes (600 data points). A full survey was also run that took 45 minutes to complete (2700 data points). Since the full survey contains substantially more data than the shorter trajectories, it was used as a proxy for ground truth. The mean squared reconstruction errors after executing the different trajectories (relative to the full survey) are shown in Figure 8. For both the 200 m and 400 m budgets, the trajectory generated by the proposed RIG-tree algorithm provides lower RMSE than the trajectory generated by the discrete branch and bound algorithm. Figure 8 also shows that RIG provides improvements in the mean variance estimate across the lake for both trajectory lengths. The resulting wireless signal strength maps (shown in Figure 7) show a qualitative improvement for the paths generated by RIG-tree. These results demonstrate the benefit of planning in continuous space. The experiments on the autonomous surface vehicle also show the ease of implementation of the proposed algorithm on a fielded system.

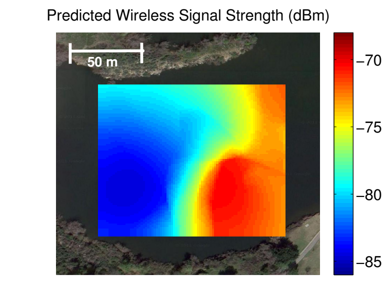
6 Conclusions and Future Work

We have shown that it is possible to generate highly informative motion plans while respecting a budget constraint through the use of iterative sampling-based motion planning algorithms. Variants of the proposed RIG-roadmap, RIG-tree, and RIG-graph algorithms apply to modular, time-varying modular, and submodular information objectives. We have also shown that the RIG-graph variant is asymptotically optimal, in that it approaches the optimal solution with increasing runtime, as long as a conservative pruning strategy is used to eliminate suboptimal plans. We have shown through simulations that the proposed algorithm is capable of finding optimal solutions in discrete domains quickly. We have also demonstrated the effectiveness of the RIG-tree algorithm in a wireless signal strength mapping domain where it is able to improve the accuracy of a wireless signal strength map given a limited budget on mission time.

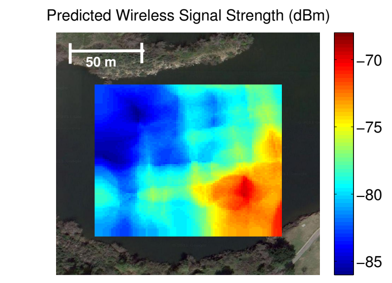
The suite of algorithms proposed here opens up a number of avenues for future work. More



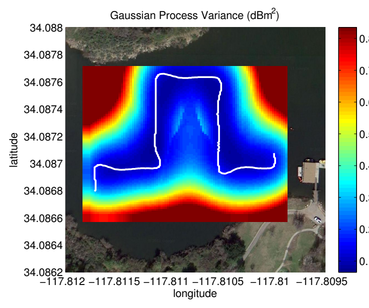
(a) After 400 m branch/bound path



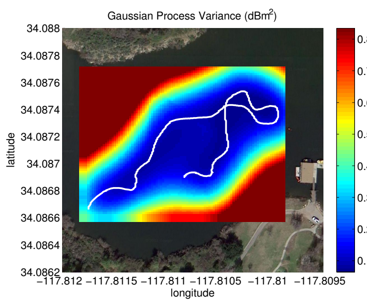
(b) After 400 m RIG-tree path



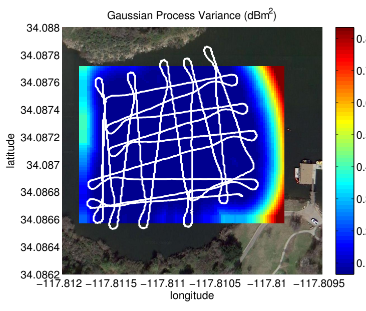
(c) After 2500 m survey



(d) After 400 m branch/bound path



(e) After 400 m RIG-tree path



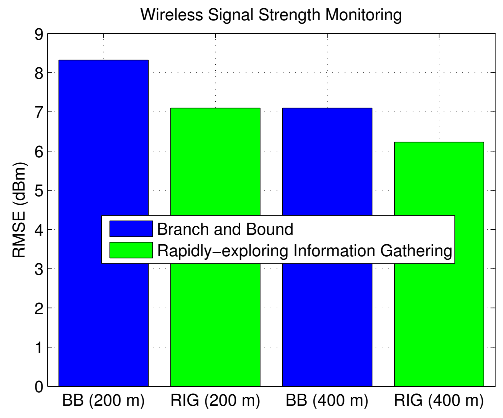
(f) After 2500 m survey

Figure 7: **Left and Center:** Wireless signal strength map and variance map built after executing the 400 m trajectories generated by branch and bound and RIG-tree. The executed trajectories are shown in white. **Right:** Wireless signal strength map and variance map built after executing the full 2500 m survey. The map generated by RIG-tree is qualitatively closer to the one generated by the full survey.

aggressive pruning strategies may allow for asymptotically optimal behavior with submodular objective functions while pruning away more suboptimal trajectories. Additional work could also provide more efficient methods for calculating the information of a given trajectory for complex objective functions. More future work lies in generating more intelligent sampling strategies that focus the candidate trajectories towards particularly informative areas of the environment. RRTs can bias samples towards the goal by adjusting the sampling distribution. Similarly, the RIG variants could be biased towards high-information states. Quantifying the benefit of such biased sampling would require a careful empirical study across domains. In the current formulation, each node in the graph or tree encodes a single trajectory through the environment. It may be beneficial to examine alternative trajectories through the existing nodes in the graph. Dynamic programming methods may be candidates for this alternative trajectory identification.

We can also extend the proposed algorithm to multiple vehicles by sampling in the space generated by the cross product of the vehicle's individual configuration spaces. In the multi-vehicle case, a large number of samples may be required to sufficiently cover the sampling space and generate desirable trajectories. A more efficient alternative is to run multiple single-robot RIGs sequentially to avoid the exponential blowup in the configuration space. For the case of submodular objectives, such an approach would yield bounded performance relative to optimal (Singh et al., 2009).

A key future extension of the RIG variants is to apply them to the case where online replan-



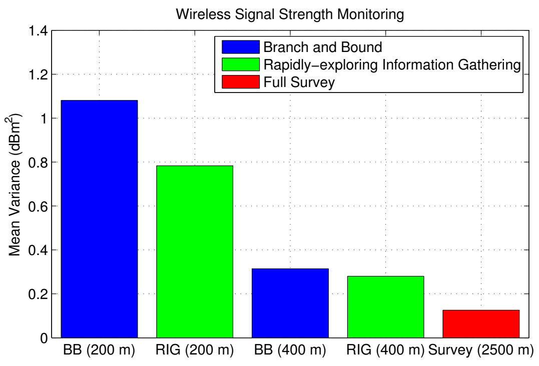


Figure 8: Root mean square error (left) and mean Gaussian process variance (right) of the predicted wireless signal strength in a lake after executing four different data collection trajectories. The trajectories planned by the branch an bound algorithms are based on a discrete grid, and the trajectories planned by the proposed RIG-tree algorithm operate in continuous space (trajectories are shown in Figure 6). The RIG-tree algorithm is able to achieve lower RMSE (relative to the full survey) and mean variance for a given trajectory length.

ning is allowed if the objective function changes from the initial estimate. An interesting question is whether parts of the previous solution could be reused to improve the computational requirements once the information objective has changed. This extension would allow RIG to be used in environments where the vehicle must act adaptively as new information is received. However, it would require careful management of computational resources to reason over the space of *possible* information objectives rather than a known information objective.

Finally, the proposed algorithms can be used to solve general motion planning problems with integral constraints (e.g., budget along a path). Other problems in optimization that require integral constraints may also benefit from the algorithms developed here. Therefore, the RIG variants may lead to stable numerical algorithms for a wide range of problems reaching beyond information collection. One potential roadblock is to extend the RIG variants to incorporate objective functions that do not have the property of submodularity, which would require alternative pruning strategies than those presented here. Ultimately, the work we have presented here provides a foundation for the application of sampling-based motion planning algorithms to robotic information gathering and related optimization problems.

Acknowledgments

The authors gratefully acknowledge Jonathan Binney from Willow Garage for assistance with data processing.

Funding

This research has been funded in part by the Office of Naval Research [grant numbers N00014-09-1-0700, N00014-07-1-00738, N00014-06-10043]; and the National Science Foundation [grant numbers CCR-0120778, CNS-1035866, IIS-1317815].

Appendix: Index to Multimedia Extensions

The multimedia extensions to this article can be found online by following the hyperlinks at www.ijrr.org.

Table 1: Index to multimedia extensions

Extension	Media Type	Description
1	Video	RIG-tree algorithm incrementally refining a trajectory

References

Bajcsy, R. (1988). Active perception. *Proc. IEEE, Special Issue on Computer Vision*, 76(8):966–1005.

Binney, J. and Sukhatme, G. S. (2012). Branch and bound for informative path planning. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2147–2154.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. F. (2003). Coordinated decentralized search for a lost target in a Bayesian world. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 48–53.

Bourgault, F., Makarenko, A. A., B.Williams, S., Grocholsky, B., and Durrant-Whyte, H. F. (2002). Information based adaptive robotic exploration. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, volume 1, pages 540–545.

Bry, A. and Roy, N. (2011). Rapidly-exploring random belief trees for motion planning under uncertainty. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 723–730.

Cameron, A. and Durrant-Whyte, H. (1990). A Bayesian approach to optimal sensor placement. *Int. J. Robotics Research*, 9(5):70–88.

Chen, S., Li, Y., and Kwok, N. M. (2011). Active vision in robotic systems: A survey of recent developments. *Int. J. Robotics Research*, 30(11):1343–1377.

Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Mag.*, 19(4):72–82.

Fiorelli, E., Leonardi, N. E., Bhatta, P., Paley, D. A., Bachmayer, R., and Fratantoni, D. M. (2006). Multi-AUV control and adaptive sampling in Monterey Bay. *IEEE J. Oceanic Engineering*, 31(4):935–948.

Hollinger, G., Englot, B., Hover, F., Mitra, U., and Sukhatme, G. (2013). Active planning for underwater inspection and the benefit of adaptivity. *Int. J. Robotics Research*, 32(1):3–18.

Hollinger, G., Singh, S., Djugash, J., and Kehagias, A. (2009). Efficient multi-robot search for a moving target. *Int. J. Robotics Research*, 28(2):201–219.

Hollinger, G. and Sukhatme, G. (2013). Sampling-based motion planning for robotic information gathering. In *Proc. Robotics: Science and Systems Conf.*

Hsiao, K., Kaelbling, L. P., and Lozano-Pérez, T. (2010). Task-driven tactile exploration. In *Proc. Robotics: Science and Systems Conf.*

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Research*, 30(7):846–894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the RRT*. In *IEEE Int. Conf. Robotics and Automation*, pages 1478–1483.

Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580.

Krause, A. and Guestrin, C. (2011). Submodularity and its applications in optimized information gathering. *ACM Trans. Intelligent Systems and Technology*, 2(4).

Lan, X. and Schwager, M. (2013). Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2407–2412.

Latombe, J. C. (1991). *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA.

LaValle, S. M. (2006). *Planning Algorithms.* Cambridge University Press, Cambridge, U.K.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *Int. J. Robotics Research*, 20(5):378–400.

Levine, D. S. (2010). Information-rich path planning under general constraints using rapidly-exploring random trees. Master's thesis, Massachusetts Institute of Technology.

Low, K. H., Dolan, J. M., and Khosla, P. (2009). Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proc. Int. Conf. Automated Planning and Scheduling*, pages 233–240.

Myers, V. and Williams, D. (2010). A POMDP for multi-view target classification with an autonomous underwater vehicle. In *Proc. IEEE OCEANS Conf.*

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning.* The MIT Press.

Reif, J. H. (1979a). Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science*, pages 421–427.

Reif, J. H. (1979b). Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science*, pages 421–427.

Ryan, A. and Hedrick, J. K. (2010). Particle filter based information-theoretic active sensing. *Robotics and Autonomous Systems*, 58(5):574–584.

Singh, A., Krause, A., Guestrin, C., and Kaiser, W. (2009). Efficient informative sensing using multiple robots. *J. Artificial Intelligence Research*, 34:707–755.

Smith, R., Schwager, M., Smith, S., Jones, B., Rus, D., and Sukhatme, G. (2011). Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *J. Field Robotics*, 28(5):714–741.

Thompson, D. R., Chien, S., Chao, Y., Li, P. P., Cahill, B., Levin, J., Schofield, O., Balasuriya, A., Petillo, S., Arrott, M., and Meisinger, M. (2010). Spatiotemporal path planning in strong, dynamic, uncertain currents. In *Proc. IEEE Int. Conf. Robotics and Automation.*

Thrun, S. (1999). Monte carlo POMDPs. In *Neural Information Processing Systems Conf.*, pages 1064–1070.

van den Berg, J., Patil, S., and Alterovitz, R. (2012). Motion planning under uncertainty using iterative local optimization in belief space. *Int. J. Robotics Research*, 31(11):1263–1278.

Vasudevan, S., Ramos, F. T., Nettleton, E. W., and Durrant-Whyte, H. F. (2009). Gaussian process modeling of large scale terrain. *J. Field Robotics*, 26(10):812–840.

Wald, A. (1945). Sequential tests of statistical hypotheses. *Ann. Mathematical Statistics*, 16(2):117–186.

Weber, T. (2010). Simple methods for evaluating and comparing binary experiments. *Theory and Decision*, 69(2):257–288.