# Risk-Aware Graph Search with Dynamic Edge Cost Discovery

Journal Title
XX(X):1–13

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

**Jen Jen Chung[1], Andrew J. Smith[2], Ryan Skeele[3] and Geoffrey A. Hollinger[2]**

**Abstract**

*In this paper we introduce a novel algorithm for incorporating uncertainty into lookahead planning. Our algorithm searches through connected graphs with uncertain edge costs represented by known probability distributions. As a robot moves through the graph, the true edge costs of adjacent edges are revealed to the planner prior to traversal. This locally revealed information allows the planner to improve performance by predicting the benefit of edge costs revealed in the future and updating the plan accordingly in an online manner. Our proposed algorithm, Risk-Aware Graph Search (RAGS), selects paths with high probability of yielding low costs based on the probability distributions of individual edge traversal costs. We analyze RAGS for its correctness and computational complexity and provide a bounding strategy to reduce its complexity. We then present results in an example search domain and report improved performance compared to traditional heuristic search techniques. Lastly, we implement the algorithm in both simulated missions and field trials using satellite imagery to demonstrate the benefits of risk-aware planning through uncertain terrain for low-flying unmanned aerial vehicles.*

**Keywords**

Planning under uncertainty, graph search, risk-aware planning

## 1 Introduction

When planning in unstructured environments, there is a greater need for fast, reliable path planning methods capable of operating under uncertainty. Planning under uncertainty allows for robustness when faced with unknown and partially known environments. We introduce a method, Risk-Aware Graph Search (RAGS), for finding paths through graphs with uncertain edge costs (similar to Stochastic Shortest Path Problems with Recourse (Polychronopoulos and Tsitsiklis 1996)). In the domain of interest, a robot moves through an environment represented by a graph, and the true costs for edges adjacent to the robot's location are dynamically revealed. Our method accounts for both the uncertainty in the edge costs and the dynamic revealing of these costs. Thus, we bridge the gap between traditional search methods (Hart et al. 1968; Koenig and Likhachev 2005) and risk-aware planning under uncertainty (Hollinger et al. 2016; Murphy and Newman 2013).

Traditionally, graphs are composed of nodes and edges, with a node representing some state and an edge representing the transition between states. Effectively searching through a graph with known edges has been extensively researched and applies across disciplines in robotics, computer science, and optimization. We aim to expand the capabilities of graph search algorithms by allowing for uncertainty in traversal costs and dynamic discovery of those costs, while avoiding the blowup in computation from more expressive frameworks (e.g., POMDPs). Our novel approach searches over uncertain edge costs with known distributions and properly adjusts as new information about the edge costs becomes available. This formulation allows for computationally efficient methods of reducing the risk of traversing paths with high cost.

The main novelty of this work is the introduction of a non-myopic graph search algorithm for risk-aware planning with uncertain edge costs and dynamic local edge cost discovery. RAGS is an online planning mechanism that incorporates live feedback for deciding when to be conservative and when to be aggressive. With edge costs modeled as probability distributions, we can derive a principled way of leveraging information further down a path. This leads to a trade-off between revealing the true cost of a large number of edges (exploration) and traversing uncertain edges with low mean cost (exploitation). RAGS addresses this trade-off in a principled way, and the result is a low probability of executing a path with high cost.

We compare our method to *A**, *sampled A** (Murphy and Newman 2013), and a greedy approach on a large number of random connected graphs. The results show that RAGS reduces the number of high cost runs compared to all other tested methods. In addition to testing over random graphs, we validate RAGS using a real-world dataset. By applying a series of filters to satellite imagery data, we are able to extract potential obstacles that may impede a robot moving through the map. To deal with the imprecise nature of obstacle extraction, we can utilize the benefits of RAGS to find paths that are less likely to be substantially delayed. We show examples of different cases in which RAGS finds preferable

[1]Eidgenössische Technische Hochschule Zürich, Switzerland
[2]Oregon State University, USA
[3]Spectrum Geomatix, USA

**Corresponding author:**
Jen Jen Chung, Autonomous Systems Lab, Eidgenössische Technische Hochschule Zürich, Leonhardstrasse 21, 8092 Zürich, Switzerland.
Email: jenjen.chung@mavt.ethz.ch

paths. The algorithm is run on 96 satellite images taken from a broad set of landscapes. The resulting path costs over the image database confirm the benefit of the RAGS algorithm in a real-world planning domain. Finally, we conducted flight trials using a DJI Matrice 100 quadcopter (DJI 2017b) to demonstrate the real-time application of the RAGS algorithm using onboard sensor feedback.

This paper is an extended and revised version of our prior work, Skeele et al. (2016), which was presented at the 12th International Workshop on the Algorithmic Foundations of Robotics. The main extensions are an improved metric for initially bounding the set of best paths to consider during execution, extending and improving the simulations using real-world data, and the inclusion of flight trial results demonstrating the application of the algorithm in a real-world environment.

The remainder of this paper is organized as follows. Section 2 provides a review of related work and research in probabilistic planners. We formulate the path search problem with uncertain traversal costs and dynamic edge cost discovery in Section 3. In Section 4 we derive a method for quantifying path risk (Section 4.1) and present a way to reduce the search space by removing dominated paths (Section 4.2). The complete RAGS algorithm is presented in Section 4.3 and we show comparisons to existing search algorithms in Section 5. We then highlight the utility of RAGS by demonstrating its effectiveness for planning through various terrain captured from satellite imagery in Section 6 with flight trial results presented in Section 7. Finally, in Section 8, we draw conclusions and propose future directions for this line of research.

## 2 Related Work

Planning under uncertainty is a challenging problem in robotics. An underlying assumption in many existing planning algorithms is that the search space is not stochastic, that is there is high certainty in the search space. Under this assumption, researchers have developed many powerful algorithms like $A*$ (Hart et al. 1968) and $RRT*$ (Karaman and Frazzoli 2011) that perform efficient point to point planning over expected costs (see Latombe (2012) and LaValle (2006) for surveys). These algorithms are efficient for problems with deterministic actions and a well-defined search space, but are often not well suited to planning under uncertainty. Recent work has explored ways of incorporating uncertainty into similar planners in field robotics applications (Hollinger et al. 2016; Bohren et al. 2011).

Reasoning probabilistically in robotic planning allows performance to degrade gracefully when encountering the unexpected. Notable work has been done on incorporating uncertainty from sensors into the state estimation of the system (Kalman 1960; Kurniawati et al. 2008), or in the path planning itself (Chaves et al. 2015; Bry and Roy 2011). However, uncertainty can lie in both the state and the world model, so we must address both sources of uncertainty to plan effectively.

Prior work in uncertain traversability of graph edges has focused on a binary status of the edge. This family of problems is a variant of the shortest path problem known as the Canadian Traveler Problem (CTP) (Papadimitriou and Yannakakis 1991). The CTP is inspired by drivers in northern Canada who sometimes have to deal with snow blocking roads and causing delays. The focus of CTP, and variations like it, is to plan paths/policies when graph connections are uncertain. A slight variation, known as Stochastic Shortest Path with Recourse (SSPR) (Polychronopoulos and Tsitsiklis 1996), adds random arc costs. Our problem formulation, similar to the CTP and SSPR, provides local information during traversal. This is also consistent with the algorithm $PAO*$ for domains where there is a hidden state in the graph (Ferguson et al. 2004). The hidden state relates this work to Partially Observable Markov Decision Processes (POMDPs) (Monahan 1982), which provides an expressive framework for uncertain states, actions, and observations. In our case, we assume there is only uncertainty in the transition costs, which avoids the computational blowup often found in large POMDPs.

Planning over the expected cumulative cost is another relevant variant of the shortest path problem (Hou et al. 2014). Risk-Sensitive Markov Decision Process are one approach to such stochastic planning problems. These solutions address cases where large deviations from the expected behavior can have detrimental effects (Carpin et al. 2016). Previous approaches have used a parameter for risk aversion to solve Risk-Sensitive MDPs (Marcus et al. 1997). Like these techniques, we aim to avoid large deviations from the expected value while planning for low costs; however we also look to exploit local information available during execution. We build on work in risk-aware planning (e.g., Risk-Sensitive MDPs), which deal with probability distributions over outcomes. Other risk-aware planning techniques in the literature use bounding of likelihood (Sun et al. 2015) by minimizing the path length with respect to a lower bound on the probability of success. This is similar to work in Feyzabadi and Carpin (2014), which instead bounds the average risk. These algorithms define reasonable ways of assigning a value for trading off between risk and a primary search objective like distance, but they do not incorporate dynamically revealed information as part of the search.

The stochastic edge cost problem has been approached using an iterative sampling method when dealing with uncertainty in terrain classification (Murphy and Newman 2013). In this prior work, the edge values between landmarks are sampled from modeled cost distributions, and $A*$ is used over each sampling to generate a list of paths. The paths most frequently taken are considered the most likely to return low cost paths, which yields a method (*sampled* $A*$) that we test our algorithm against. In our work, we derive a formula for reducing the risk of a path based on the uncertainty of the traversal costs themselves, which allows for a more expressive framework than heuristic searches and reduced computational requirements compared to sampled approaches.

## 3 Problem Formulation

In this paper, we consider the problem of planning and executing a risk-aware path through an uncertain environment where knowledge of the true traversal costs are revealed only as we arrive within some proximity of an area.

This planning scenario can be described over a graph with edge costs initially represented by some set of distributions, with the true edge costs identified as we arrive at the parent node of each edge during the path execution. Throughout this paper we use the term "risk" in its general sense to describe the uncertainty of an outcome, i.e. committing to any particular path from start to goal has an associated risk since the true cost of the traversal is unknown at the start of execution. The generality of the RAGS formulation is such that the cost can represent any metric, e.g. travel time, path length, energy expenditure or probability of collision/failure (as in SSPR), as long as the outcomes can be modeled as a distribution. Although we focus on a path cost minimization objective in this paper, we note that this formulation could represent informative path planning objectives by considering the equivalent maximization problem (Meliou et al. 2007). We now formulate the path search problem with uncertain edge costs and introduce notation that will be used throughout the paper.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the cost of traversing edge $E \in \mathcal{E}$ is drawn from a normal distribution $\mathcal{N}\left(\mu_E, \sigma_E^2\right)$. The cost of a path $\mathcal{P} \subset \mathcal{E}$ is the sum of the edge traversal costs, each of which are normally distributed. Thus the total path cost is drawn from $\mathcal{N}\left(\mu_\mathcal{P}, \sigma_\mathcal{P}^2\right)$, where $\mu_\mathcal{P} = \sum_\mathcal{P} \mu_E$ and $\sigma_\mathcal{P}^2 = \sum_\mathcal{P} \sigma_E^2$. This formulation is similar to that of the random network used in Polychronopoulos and Tsitsiklis (1996).

**Assumption 1.** *The true edge costs are drawn as i.i.d. samples from the respective cost distributions when queried.*

An important implication of Assumption 1 is that although multiple paths may share a subset of edges, all paths can be treated as having independent cost distributions. This is analogous to a problem where traversal costs are not fixed, but instead vary over time, and edge costs are queried upon arrival at the parent node. This assumption simplifies the computation of risk and dominance (shown later in Section 4) without significantly changing the nature of the problem.

Given any pair of start and goal vertices in the graph, $V_s, V_g \in \mathcal{V}$, the task is to traverse the graph along the *acyclic* path of least risk. More precisely, since each edge transition prunes the available set of paths to the goal, the path of least risk retains the highest probability of traversing the overall least-cost path as each transition is executed from $V_s$ to $V_g$.

The decision at each vertex can be formulated by considering the next available transitions and their associated path sets. For example, say edge connections exist between the current vertex $V_t$ and each of the vertices in the set $\mathcal{V}_{t+1} = \{A, B, C, \cdots\}$; furthermore, $m$ acyclic paths exist from vertex $A$ to $V_g$, while $n$ acyclic paths exist from vertex $B$ to $V_g$, etc. Let $\mathcal{A}$ be the set of random variables $A_i$, $i = \{1, \cdots, m\}$, where $A_i \sim \mathcal{N}\left(\mu_{A_i}, \sigma_{A_i}^2\right)$ represents the cost distribution of path $i$ from vertex $A$ to $V_g$ (see Figure 1). Let $c_{A_i}$ be a sample of the random variable $A_i$ and let $c_{A_0}$ be the known cost of transitioning between $V_t$ and $A$, then the lowest-cost path from $V_t$ to $A$ and onwards to $V_g$ has cost:

$$c_{A_{min}} = c_{A_0} + \min_{A_i \in \mathcal{A}} c_{A_i}. \tag{1}$$

Similar statements can be made for path sets $\mathcal{B}, \mathcal{C}, \cdots$. To traverse the path of least-risk, each edge transition must select the next vertex $V \in \mathcal{V}_{t+1}$ such that the following probability holds,

$$P\left(c_{V_{min}} < c_{V'_{min}}\right) \geq 0.5, \quad \forall V' \in \mathcal{V}_{t+1} \setminus V. \tag{2}$$

That is, transitioning to vertex $V$ results in a higher probability of executing a lower cost path than transitioning to any other neighboring vertex $V'$.

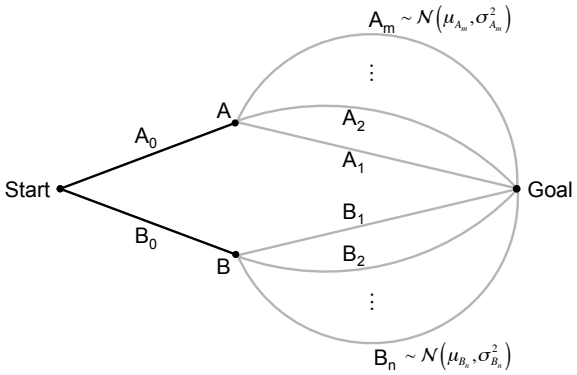



**Figure 1.** Setup of pairwise comparison for sequential lookahead planning. Given the path cost distributions, we can directly compute the probability that traveling from Start to Goal through A will ultimately yield a cheaper path than traveling via B.

## 4 Risk-Aware Graph Search (RAGS)

### *4.1 Quantifying Path Risk*

The pairwise comparison $P\left(c_{A_{min}} < c_{B_{min}}\right)$ describes the probability that the lowest-cost path in the set $\mathcal{A}$ is cheaper than the lowest-cost path in the set $\mathcal{B}$. Given Assumption 1, this probability can be expanded to,

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = \int_{-\infty}^{\infty} P\left(c_{B_{min}} = x\right) \cdot P\left(c_{A_{min}} < x\right) \mathrm{d}x. \tag{3}$$

We can now express each term in the integral according to the path cost distributions of each respective set. Let,

$$f(x, \mathcal{A}) = P\left(c_{A_{min}} < x\right),$$
$$g(x, \mathcal{B}) = P\left(c_{B_{min}} = x\right).$$

Since each of the path costs are drawn from normal distributions, then

$$f(x, \mathcal{A}) = 1 - \prod_{i=1}^{m} \tfrac{1}{2}\mathrm{erfc}\left(d(A_i)\right), \tag{4}$$

$$g(x, \mathcal{B}) = \sum_{j=1}^{n} \left[ \tfrac{1}{\sqrt{2\pi}\sigma_{B_j}} \exp\left(-d(B_j)^2\right) \cdot \prod_{\substack{k=1 \\ k \neq j}}^{n} \tfrac{1}{2}\mathrm{erfc}\left(d(B_k)\right) \right], \tag{5}$$

where $d\left(\zeta_i\right) = \frac{x - c_{\zeta_0} - \mu_{\zeta_i}}{\sqrt{2}\sigma_{\zeta_i}}$.

As an aside, note that $f'(x, \cdot) = g(x, \cdot)$. Thus, using integration by parts, we can show that

$$P\left(c_{A_{min}} < c_{B_{min}}\right) = 1 - P\left(c_{B_{min}} < c_{A_{min}}\right),$$

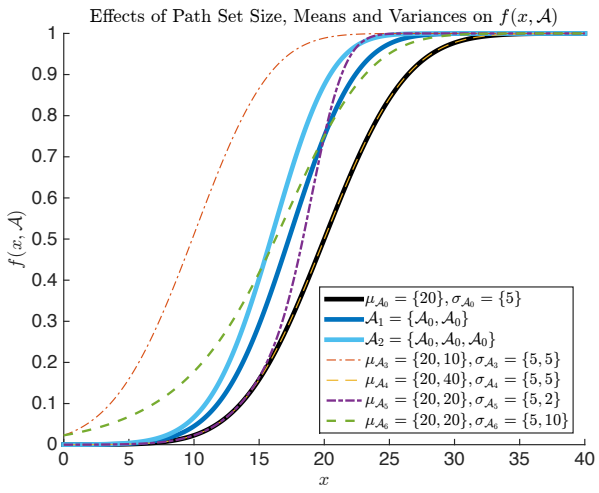



**Figure 2.** The plot of $f(x, \mathcal{A})$, probability of the cost of the path being less than $x$, for seven path sets; $\mathcal{A}_0$ has a single path with cost drawn from $\mathcal{N}\left(20, 5^2\right)$. This plot shows the effects of including additional paths to the set whose costs are drawn from varying distributions. This figure is best viewed in color.

confirming that these two events are indeed complementary.

Equations (4) and (5) give some intuitive insight into the calculation of path risk. This formulation performs a trade-off between the number of available paths in each set and the quality of the paths in each set, the latter represented by the means and variances of the path cost distributions. For example, $f(x, \mathcal{A})$ calculates the probability that the best path in $\mathcal{A}$ has a path cost less than $x$ (the plot of $f(x, \mathcal{A})$ is shown in Figure 2 for $\mu_{\mathcal{A}_0} = \{20\}$ and $\sigma_{\mathcal{A}_0} = \{5\}$ as well as six other path set variants). From Equation (4), we note that as $m \to \infty$, $f(x, \mathcal{A}) \to 1$, $\forall x \in (-\infty, \infty)$. This is shown in the solid curves plotting $f(x, \mathcal{A}_{0,1,2})$ in Figure 2 (black, dark blue and light blue, respectively). Path set $\mathcal{A}_1$ has twice as many paths (of equal means and variances) as $\mathcal{A}_0$, while $\mathcal{A}_2$ has three times as many. The curves show a trend towards an earlier and more rapid transition to 1 as $m$ increases; however the difference between $f(x, \mathcal{A}_2)$ and $f(x, \mathcal{A}_1)$ is much smaller than the difference between $f(x, \mathcal{A}_1)$ and $f(x, \mathcal{A}_0)$, suggesting that adding more paths results in diminishing returns in terms of driving $f(x, \mathcal{A}) \to 1$, $\forall x \in (-\infty, \infty)$.

Other trends can be observed when adding paths of varying cost distributions to the set. $\mathcal{A}_3$ includes a second path with lower mean and equivalent variance to $\mathcal{A}_0$. The corresponding $f(x, \mathcal{A}_3)$ (red, dot-dashed) is shifted significantly to the left of $f(x, \mathcal{A}_0)$, causing the transition towards 1 to occur much earlier. On the other hand, the addition of a second path with higher mean and equivalent variance results in almost no change to the curve, as shown by the overlap between $f(x, \mathcal{A}_4)$ (yellow, dashed) and $f(x, \mathcal{A}_0)$. Adding a path with equivalent mean but a lower variance results in the plot of $f(x, \mathcal{A}_5)$ (purple, dot-dashed), which shows a much more rapid transition to 1 compared to either $f(x, \mathcal{A}_0)$ or $f(x, \mathcal{A}_1)$. However, the addition of a path with equal mean and higher variance increases the overall uncertainty associated with the path set, as shown by the shallower gradation in the probability curve of $f(x, \mathcal{A}_6)$ (green, dashed). Furthermore, this means that although the transition towards 1 begins at lower values of $x$, as $x \to \infty$, $f(x, \mathcal{A}_6) \to 1$ more slowly than for $f(x, \mathcal{A}_1)$ or $f(x, \mathcal{A}_5)$. Indeed, the cross-over occurs at the mean, $x = 20$. For $x < 20$, $f(x, \mathcal{A}_6)$ lies above both $f(x, \mathcal{A}_1)$ and $f(x, \mathcal{A}_5)$, after this point, their ordering reverses.

This analysis motivates a bounding approach that compares the values of path cost means and variances to retain only the most relevant paths for the calculation of Equation (3). Bounding the path set is especially desirable since the computation of each pairwise comparison has complexity $\mathcal{O}(n^2 m)$, where $n$ and $m$ are the sizes of the path sets under consideration.

## 4.2 Non-Dominated Path Set

Existing graph search algorithms, such as A*, use a total ordering of vertices based on the calculated cost-to-arrive to find a single optimal path between defined start and goal vertices. In contrast, an implementation of RAGS requires two sweeps of the graph, since an initial pass is required to gather cost-to-go information at each node for quantifying the path risk at execution. If all possible acyclic paths between start and goal are to be considered, then this initial pass is exponential in the average branching factor of the search tree. To reduce this computation, we introduce a partial ordering condition based on the path cost means and variances to sort the priority queue and terminate the search. This bounds the set of resultant paths to be considered during execution by accepting only those that exhibit desirable path cost mean and variance characteristics.

As discussed in Section 4.1, the addition of paths with higher mean costs to the existing path set results in little improvement in the overall risk of committing to that set. Similarly, adding paths with higher variances on the path cost results in a slower convergence of $f(x, \mathcal{A})$ to 1. In our prior work (Skeele et al. 2016), we used a simple comparison between the path cost distribution properties to bound the path set by removing all paths with strictly higher means and variances. However, since we consider costs defined by normal distributions that are evenly distributed about the mean, a path with high cost variance has an equal probability of resulting in both more or less expensive traversals. Furthermore, these probabilities are greater than that of a path with the same mean cost but a lower cost variance. Thus, naïvely penalizing paths with high variance can result in a loss of optimality. In this work, we update the partial ordering criterion from Skeele et al. (2016) to formally trade off path cost mean and variance according to our probabilistic definition of path risk introduced in the previous section.

In practice, the partial ordering considers whether a path is *dominated* by an existing path, either in the open or closed set. For any two paths, $A$ and $B$, between the same set of start and end vertices, we can consider the probability that the cost of traversing $A$ is lower than the cost of traversing $B$,

$$P\left(c_A < c_B\right) = 1 - P(c_B - c_A \leq 0).$$

Since both path costs are represented by normal distributions, then $c_B - c_A \sim \mathcal{N}\left(\mu_B - \mu_A, \sigma_B^2 + \sigma_A^2\right)$, such that,

$$P\left(c_A < c_B\right) = 1 - \Phi\left(\frac{\mu_A - \mu_B}{\sqrt{\sigma_B^2 + \sigma_A^2}}\right),$$

where $\Phi(\cdot)$ is the cumulative distribution function of a normal distribution. Here we introduce the *domination threshold*, $d_{thresh} \in [0.5, 1)$, such that if $P(c_A < c_B) > d_{thresh}$, then we say that $A$ *dominates* $B$. That is

$$A \succ B \Leftrightarrow 1 - \Phi\left(\frac{\mu_A - \mu_B}{\sqrt{\sigma_B^2 + \sigma_A^2}}\right) > d_{thresh}.$$

Expanding $\Phi(\cdot)$, this becomes

$$A \succ B \Leftrightarrow \mu_A < \mu_B + \sqrt{2\left(\sigma_B^2 + \sigma_A^2\right)}\,\mathrm{erf}^{-1}\left(1 - 2d_{thresh}\right). \tag{6}$$

The inverse error function term in Equation (6) produces a value between $(-\infty, 0]$, which scales the contribution of the distribution variances in the path domination comparison. The effect of this is such that as $d_{thresh} \to 1$, the right hand side of the inequality in Equation (6) goes to $-\infty$, and the number of dominated paths goes to 0. In other words, the number of paths in the non-dominated path set grows to contain all possible paths from start to goal as $d_{thresh} \to 1$. On the other hand, when $d_{thresh} = 0.5$, the non-dominated path set reduces to those paths with optimal mean cost. This is formalized in Lemma 1 and Theorem 1 below.

**Lemma 1.** *Given the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as defined in Section 3, the non-dominated path sets $\mathcal{P}_{ND}$ and $\mathcal{P}'_{ND}$ are generated by using domination thresholds $d_{thresh}$ and $d'_{thresh}$, respectively. If $d_{thresh} \leq d'_{thresh}$, then $\mathcal{P}_{ND} \subseteq \mathcal{P}'_{ND}$.*

**Proof.** Proof by contradiction: If $\mathcal{P}_{ND} \nsubseteq \mathcal{P}'_{ND}$, then there exists a path $B$ where $B \in \mathcal{P}_{ND}$ and $B \notin \mathcal{P}'_{ND}$. It follows from Equation (6) that there exists a path $A$ such that the following inequalities hold true:

$$\mu_A < \mu_B + \sqrt{2\left(\sigma_B^2 + \sigma_A^2\right)}\,\mathrm{erf}^{-1}\left(1 - 2d'_{thresh}\right),$$
$$\mu_A \geq \mu_B + \sqrt{2\left(\sigma_B^2 + \sigma_A^2\right)}\,\mathrm{erf}^{-1}\left(1 - 2d_{thresh}\right).$$

Giving,

$$\mathrm{erf}^{-1}\left(1 - 2d_{thresh}\right) < \mathrm{erf}^{-1}\left(1 - 2d'_{thresh}\right).$$

Since $\mathrm{erf}^{-1}$ is a strictly increasing function, then

$$1 - 2d_{thresh} < 1 - 2d'_{thresh} \Rightarrow d_{thresh} > d'_{thresh},$$

which contradicts the required condition. □

**Theorem 1.** *For any $d_{thresh} \in [0.5, 1)$, the non-dominated path set includes the A* path calculated on the mean of the edge cost distributions.*

**Proof.** Let path $B$ be the $A^*$ path calculated on the mean, which has the lowest total mean cost, $\mu_B = \mu_{min}$. According to Equation (6), when $d_{thresh} = 0.5$, the domination condition is given by $\mu_A < \mu_B$. However, by definition, for all paths $A$, $\mu_A \geq \mu_{min}$. Thus the A* path calculated on the mean of the edge cost distributions will be contained in the non-dominated path set for $d_{thresh} = 0.5$. It follows from Lemma 1 that this path will be included in all non-dominated path sets for $d_{thresh} \in [0.5, 1)$. □

From a practical perspective, the purpose of the domination threshold is to determine the cutoff point for deciding which paths are included in the non-dominated path set and which are excluded. For example, with a domination threshold of 0.6, all paths in the non-dominated set have at least a 60% probability of incurring a lower cost than any path excluded from the set, i.e. if path $A$ dominates path $B$, then path $B$ has $< 40\%$ chance of being the better path. As we increase the domination threshold, more paths are included in the non-dominated set, reducing the chances that the true best path is excluded from the set. However, at the same time, these newly added paths to the non-dominated set also have a successively lower probability of actually being the best path. In our comparative experiments in Section 5 we show that beyond a certain domination threshold, there does not appear to be value in adding more paths to the non-dominated path set, i.e. the additional paths serve to confound the online decision making rather than provide useful options.

### 4.3 RAGS Dynamic Execution

Given the non-dominated path set, path execution can occur by conducting edge transitions at each node to select the path set of least risk according to Equation (2). The true edge transition costs, which become available for all neighboring edges to the current node, are included by directly substituting the known value of $c_{V_0}$ into Equation (1).* The pseudo code for the complete RAGS algorithm is provided in Algorithm 1; an initial sweep of the graph is conducted to search for the non-dominated path set, and a second sweep is conducted during execution to incorporate edge cost information as it is received.

The initial sweep, shown in lines 1-17 of Algorithm 1, is similar to the $A^*$ procedure with three notable exceptions. First, in lines 8-9, if the current node contains a path from start to goal, then it is included in the non-dominated path set. Second, in line 14, each child node of the current expanded node is checked against the ancestor list to explicitly exclude looping paths. Furthermore, a child node is also excluded at this point if it is found to be dominated by any node in the closed set with the same end vertex. Third, in line 16, the initial sweep is terminated either when open set is empty or when the next node to be expanded from the open set is found to be dominated by any node in the current non-dominated path set.

The RAGS online execution begins by constructing the directed graph formed by the non-dominated path set (line 18). Beginning at the start vertex (line 20), each edge transition first considers the set of all neighboring vertices (line 23), and computes a total ordering of the possible traversals according to Equation 3 (line 24). It is here that RAGS incorporates the observed costs of the next transitions if that information is available. The edge traversal to the best vertex from this set is executed (line 25), and this loop continues until the goal vertex is reached.

*Note that knowledge of the true neighboring edge costs is not required for RAGS to formulate a path. The immediate transition costs $c_{A_0}$ and $c_{B_0}$ can be included as distributions in Equation (3) to determine a path from start to goal. Dynamic replanning is only necessary if new edge cost information is discovered.

**Algorithm 1** RISK-AWARE GRAPH SEARCH

// INITIAL SWEEP
Initialize open, closed, $\mathcal{P}_{ND} \leftarrow \emptyset$ ▷ Initialize open and closed sets, and non-dominated path set
$V_s \leftarrow$ start, $V_g \leftarrow$ goal
$N_s \leftarrow \emptyset$, $N_s$.append($V_s$) ▷ Initialize start node
open.push($N_s$) ▷ Place the start node in the open set
**while** open $\neq \emptyset$ **do**
    $N_0 \leftarrow$ open.pop() ▷ Current search node
    closed.push($N_0$)
    **if** $N_0$.getVertex() = $V_g$ **then**
        $\mathcal{P}_{ND}$.push($N_0$) ▷ Non-dominated path to goal found
    **else**
        $\mathcal{V}_{t+1} \leftarrow$ getNeighbors($N_0, \mathcal{G}$) ▷ Expand the current node
        **for** $V$ in $\mathcal{V}_{t+1}$ **do** ▷ Assess all neighboring vertices
            $N \leftarrow N_0$.append($V$) ▷ Compute child node
            **if** notAncestor($V, N_0$) $\wedge$ nonDom($N$, closed) **then**
                open.push($N$) ▷ Open set sorted according to dominance
    **if** $\neg$nonDom(open.top(), $\mathcal{P}_{ND}$) **then** ▷ End search if all nodes in the open set are
        **break** dominated by the non-dominated path set

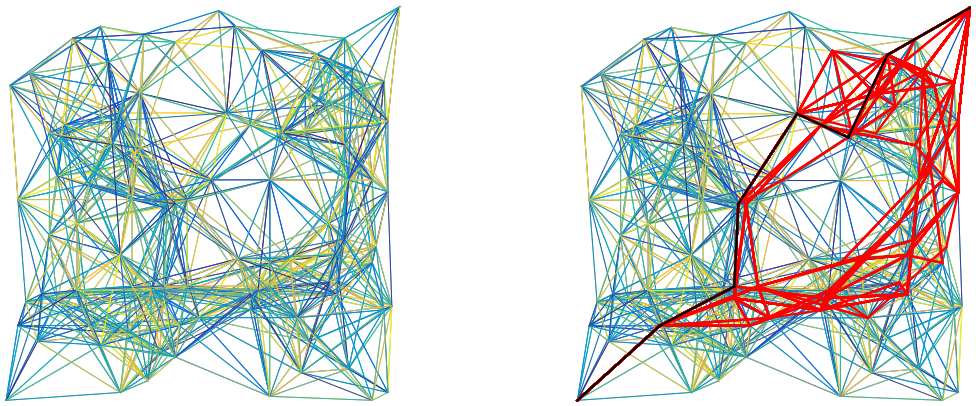
// PATH EXECUTION
$\mathcal{G}_{ND} \leftarrow \mathcal{P}_{ND}$ ▷ Directed graph formed by non-dominated path set
$N \leftarrow \emptyset$
$V_0 \leftarrow V_s$
**while** $V_0 \neq V_g$ **do**
    $N$.append($V_0$)
    $\mathcal{V}_{t+1} \leftarrow$ getNeighbors($N, \mathcal{G}_{ND}$)
    $\mathcal{V}_{ordered} \leftarrow$ ComparePathSets($\mathcal{V}_{t+1}, \mathcal{G}_{ND}$) ▷ Total ordering of vertices from Equation (3)
    $V_0 = \mathcal{V}_{ordered}$.pop() ▷ Execute edge traversal

## 5 Comparison to Existing Search Algorithms

We compared RAGS against a *naïve* $A*$ implementation, a *sampled* $A*$ method, and a *greedy* approach. *Naïve* $A*$ finds and executes the lowest-cost path based on the mean edge costs and does not perform any replanning. The *greedy* search is performed over the set of non-dominated paths and selects the cheapest edge to traverse at each step. The *sampled* $A*$ method searches over graphs constructed by sampling over the edge cost distributions and executes the path that is most frequently found. To provide a fair comparison, the planning time of *sampled* $A*$ (related to the number of sampled graphs it plans over before selecting the most frequent path) is limited to the time RAGS needed to find a path. We chose $A*$ to compare against as a simplified solution to the problem and *sampled* $A*$ because the method was previously introduced in a similar domain Murphy and Newman (2013) . The *greedy* approach provides a baseline for comparison to a purely reactive implementation.

The search algorithms were tested on a set of graphs generated with a uniform random distribution of 100 vertices over a space 100[redacted]0 in size and connected according to the PRM* radius (Karaman and Frazzoli 2011). Edge costs were represented by normal distributions with mean equal to the Euclidean distance between vertices plus an additional cost, $c_E \sim \mathcal{N}\left(\mu_E, \sigma_E^2\right)$. The mean over each edge cost distribution was drawn from a uniform random distribution, $\mu_E \sim \mathcal{U}\left(0, 100\right)$, while the variance was also drawn from a uniform distribution $\sigma_E^2 \sim \mathcal{U}\left(0, \sigma_{max}^2\right)$, where $\sigma_{max}^2 = \{5, 10, 20\}$ for each unique set of graph vertices and mean edge costs. Note that a minimum cost of the Euclidean distance was enforced in the following experiments. The start vertex was defined at $V_s = (0, 0)$, with the goal at $V_g = (100, 100)$. Figure 3a gives an example of a randomly generated graph, with edge variance shown in blue-yellow scale (darker blue lines having a smaller variance). In Figure 3b the non-dominated path set is shown in red, and the RAGS path is shown in black.

In Figure 4 we show the resulting path costs for 100 trials over a single graph configuration, where each trial drew new edge costs from the distributions as described above. For this set of experiments RAGS used a domination threshold of $d_{thresh} = 0.60$ to compute the non-dominated path set. Of the four tested path planning methods, *naïve* $A*$ is the only technique that always executes the same path. This is because the $A*$ path is calculated according to the mean edge costs, which are fixed for each graph. *Naïve* $A*$ performs relatively well in the mean but is prone to incurring expensive traversals. For example, the most expensive $A*$ path in Figure 4c costs 83.5% more than the optimal path, while this number drops to 62.3% for RAGS. RAGS is able to mitigate against severely high cost outcomes by trading off the means and variances of available routes as well as maintaining more future path options, demonstrating the benefits of risk-aware planning. Furthermore, the difference between the path cost results for the *greedy* planner and RAGS shows that although the initial sweep for the non-dominated path set can remove the most severely expensive path options, the more sophisticated decision making used



**(a)** **(b)**

**Figure 3.** A sample search graph is shown in (a). The mean cost is the sum of the Euclidean distance plus a random additional cost. Edge variances are represented using a scale from blue to yellow on the graph. The darker (more blue) the edges, the less variance there is on the cost. The non-dominated path set (red) and the executed RAGS path (black) shown in (b) demonstrate the algorithm's ability to account for both the path cost distributions as well as the available path options to goal. Not only does it favor traversing edges with balanced mean costs and variances, it also trades off against the number of remaining path options to the goal.

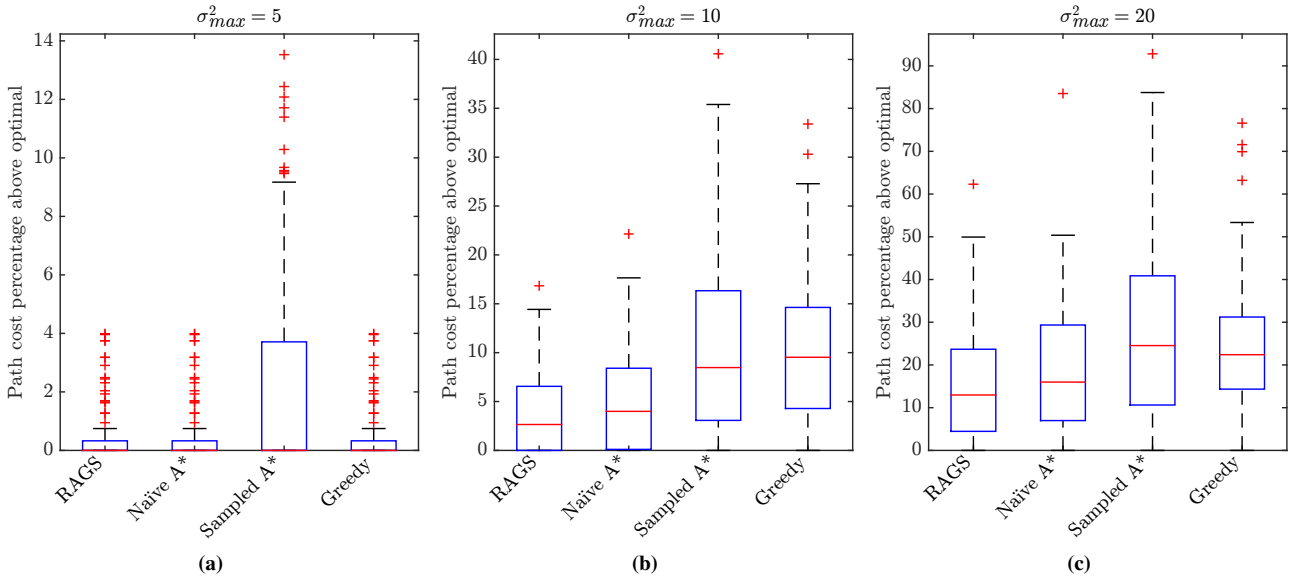



**Figure 4.** Path search results over 100 samples of one graph configuration (unique vertices and mean edge costs); edge cost variances are drawn uniformly between 0 and $\{5, 10, 20\}$ for the three plots (a)-(c), respectively. RAGS computed the non-dominated path set according to a domination threshold on $d_{thresh} = 0.60$. The difference in cost of the executed path and the true optimal path as a percentage of the optimal path cost is shown. Note that the true optimal path cost can only be calculated in hindsight.

in the RAGS plan execution contributes significantly to the overall success of RAGS. Due to the myopic nature of its planning, *greedy* is fallible to arriving at vertices with few path alternatives that all turn out to have high cost. RAGS does not suffer from the same performance decay since it accounts for the full path-to-goal cost distribution at each decision instance. The *greedy* results suggest that similar replanning strategies based only on the immediate edge cost updates may suffer from similar one-step lookahead myopia if downstream risks are not incorporated into the planning.

The *sampled A** approach can account for variability in costs along the edges as long as it can sample enough paths. However, like *naïve A**, this method is similarly prone to risky paths that yield high-cost outliers because it cannot incorporate dynamic local information in the way that RAGS does. In fact, for this set of trials, *sampled A** produced the poorest path choices compared to all other methods. This is partly because the domination threshold for these trials was set relatively low, and so the number of equivalent Monte Carlo graphs used by *sampled A** was also relatively low.

The collated results from 100 unique graph configurations, with each graph sampled 100 times, are shown in Figure 5. Figure 5a shows the final path cost performance for a range of domination threshold values, $d_{thresh} \in [0.50, 0.75]$. As the domination threshold is increased, the performance of *greedy* planning on the non-dominated path set deteriorates rapidly. On the other hand, the performance of *sampled A** improves as $d_{thresh}$ increases. This is largely because the computation time for RAGS increases as well, allowing *sampled A** to use a larger number of Monte Carlo graph

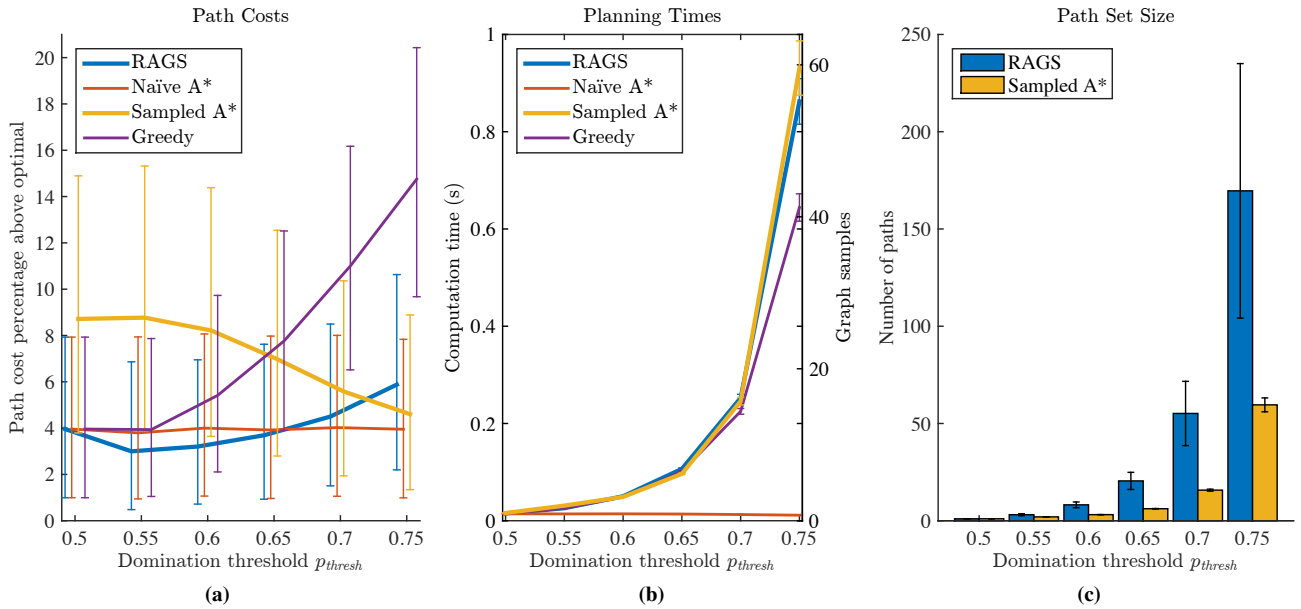



**Figure 5.** (a) Final median path cost performance, (b) computation times and (c) path set size across varying domination thresholds for $\sigma^2_{max} = 10$. In (a), the bars indicate the average first and third quartiles over the 100 graphs (each sampled 100 times). Note that the curves are slightly offset along the x-axis to aid clarity. In (b) and (c), the error bars show the 95% confidence intervals.

samples to determine the final executed path. For $d_{thresh} \in [0.5, 0.65]$, RAGS outperforms *naïve A** but begins to perform poorly for larger values of the domination threshold. This confirms that the additional paths introduced into the non-dominated path set at higher thresholds are less likely to produce better paths. In fact, these results indicate that those paths become confounding factors during the online path execution. RAGS is most beneficial at lower values of $d_{thresh}$ since it is able to identify the critical subset of paths that ultimately provide the best path options when traversing a graph with fluctuating edge costs.

Figure 5b shows the path search and execution times for each method and the total number of Monte Carlo graph queries performed by *sampled A**. Graph search and execution was calculated on a 2.1GHz Intel® Core™ i7-3612QM laptop. The computation time for the *greedy* planning method on the non-dominated path set closely follows that of RAGS, indicating that the majority of the computation cost is incurred during the initial sweep rather than during the online path execution. As expected, the computation time for A* planning is stable and the number of Monte Carlo graphs queried by *sampled A** is proportional to the RAGS computation time. By comparing Figures 5a and 5b we can see that *sampled A** requires over 50 Monte Carlo samples of the graph in order to improve upon the performance of RAGS.

The number of paths stored in the non-dominated path set for increasing $d_{thresh}$ values are also plotted against the number of *sampled A** graphs in Figure 5c. Note that all paths in the RAGS non-dominated set are unique, whereas the *sampled A** paths may not be. RAGS is able to consider a significantly larger set of possible paths compared to *sampled A** for two main reasons. First, *sampled A** needs to draw values from all edge cost distributions and run the *A** search for each Monte Carlo query. In comparison, RAGS only runs a single search through the graph during the initial sweep and leverages the path domination metric to handle the node expansion ordering throughout. Second, as mentioned above, the RAGS online execution is relatively fast compared to the initial sweep. This is due in part to the natural path set pruning that occurs during execution, where after each edge traversal, the remaining path segments to goal in the non-dominated set successively shrinks, thereby making the calculation of Equation (3) faster.

## 6 Satellite Data Experiments

We also applied RAGS to a real world domain using satellite data. In robotic path planning there is often prior information available on the environment, but this information is not necessarily reliable. An example of this is using overhead satellite imagery to provide prior information for path planning. In these trials, we collected and filtered satellite images to identify potential obstacles for a low-flying UAV. To convert the image into a useful mapping of obstacles, we filtered the satellite images to identify trees. The filtering process begins by applying a Gaussian filter to locally homogenize the images. Then the trees were extracted from the filtered images based upon pixel color; an example satellite image input and the resulting obstacles are shown below in Figures 6a-6b.

After the filtering process, the images provided a rough estimate of obstacles that could force the vehicle to slow down or alter its path. Unfortunately, due to the resolution of the satellite imagery, limitations of the tree/obstacle filter, and temporal differences between when the images were taken, the satellite imagery cannot be used to provide a guaranteed cost to traverse an area. For example, the obstacle detection cannot determine the height of the detected trees to determine if the UAV can fly above them or will be forced to detour around them. Consequently, the imagery can only be used as an estimate of the obstacles in the environment and

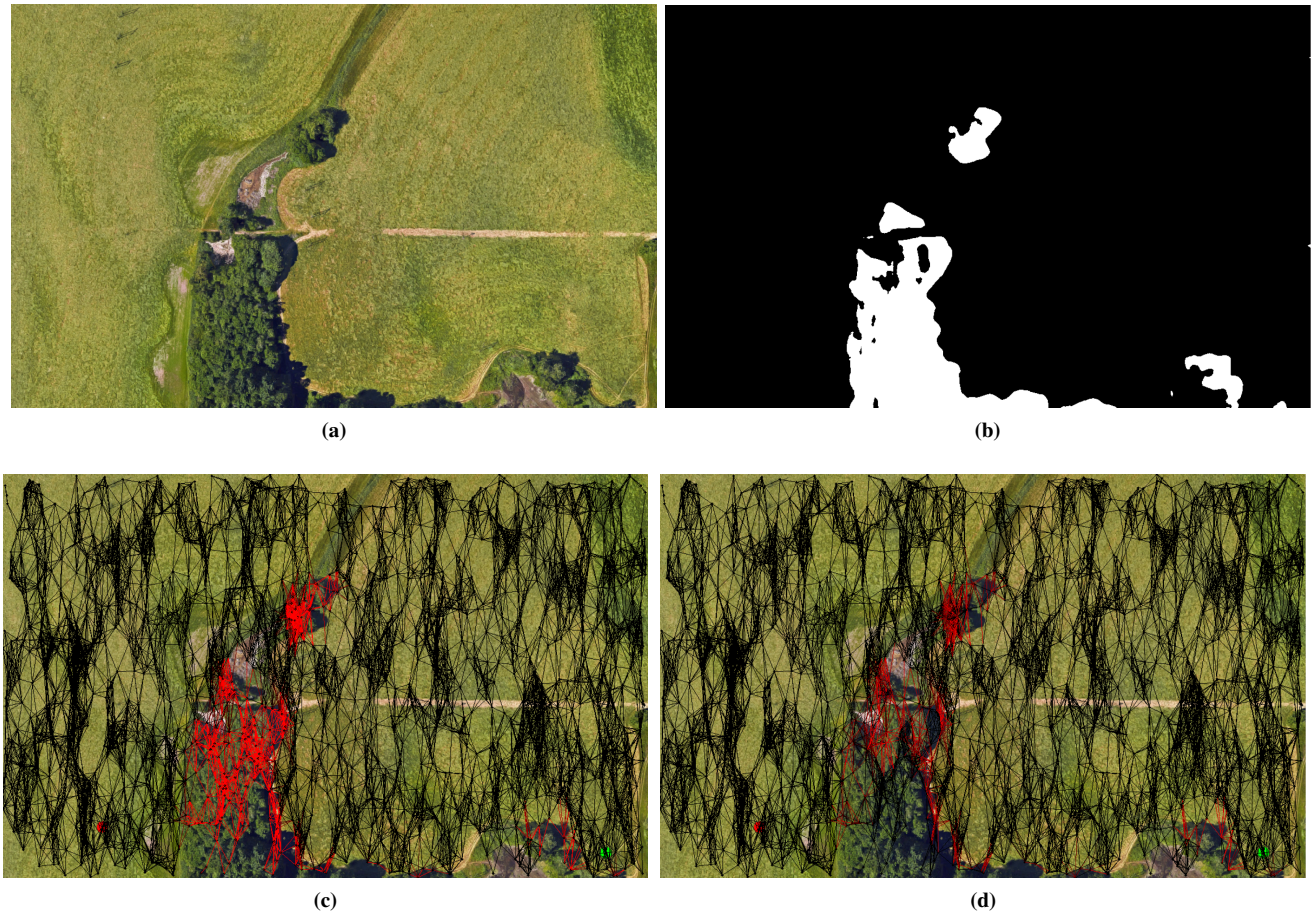


**Figure 6.** An example of a raw satellite image (a) and the extracted obstacles shown in white (b). The mean estimated obstacle edge costs (c) and variance (d) for the raw satellite image in Figure 6a. Lower values are black and higher values transition to red, best viewed in color.

the additional costs the obstacles will add to the flight path. Using the same method as before, we randomly sampled the space to generate a connected graph of flight paths. To estimate the costs of flight paths we calculated the mean and variance of the detected obstacle pixels over each edge and used these values to characterize the edge cost distributions. This process was then repeated for every edge in the graph. Example graph means and variances are shown in Figures 6c and 6d. The mean pixel lightness of the obstacle map along an edge was used to scale the Euclidean distance to generate mean edge costs. This was used to represent a penalty for the increased likelihood of an obstacle and a slower flight speed. The variance in pixel lightness across an edge in the obstacle map was directly used as the edge cost variance in our calculations. After the edges were assigned distributions, RAGS with $d_{thresh} = 0.60$ was used to search through the graph for a path from the top left start vertex to the bottom right goal vertex. $d_{thresh} = 0.60$ was chosen because it allows for exploration while not growing the non-dominated set to be too large. Actual values of the edge costs were drawn from the distribution as the simulated robot moved through the terrain.

## *6.1 Results*

We compared the performance of RAGS to the three other planning algorithms across 96 satellite images. The images are of fields with trees of varying tree densities and may also contain houses or other built structures. Images were captured at different resolutions as well as at different altitudes. The majority of the data have tree clusters scattered throughout the image to provide interesting path planning dilemmas. Four distinct environments are shown in Figures 7, 8, 9, and 10. To avoid visual clutter, only the RAGS, $A^*$, and hindsight optimal paths are shown here. The hindsight optimal path contains no notion of risk and is calculated solely on the "true" edge traversal costs, which are not observable to the planners until they reach a neighboring node. As in the previous experiments in Section 5, for each sample of an image, we draw the true costs from the computed distributions and use these to evaluate the executed paths of each planner. Compiled results for all four planning algorithms on the satellite imagery dataset are presented in Figure 11.

In Figure 7 the paths through an empty field are straight from start (magenta triangle) to goal (cyan circle). As expected, there is little variance in edge costs across the open field, and the trajectories for RAGS and $A^*$ are identical and follow the optimal path. In Figure 8 the two planners, RAGS and $A^*$, again follow identical paths in the presence of an obstacle field with limited options. This map results in one obvious intuitive path for the planners to find

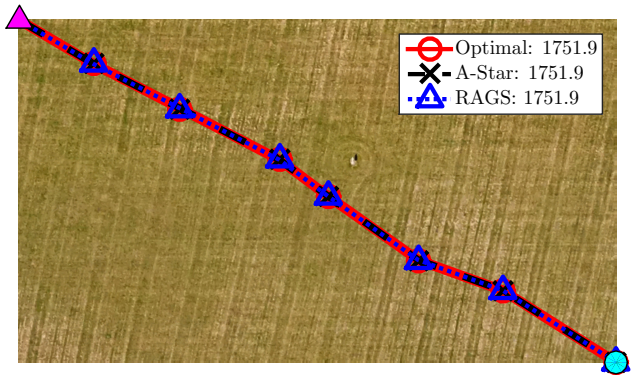



**Figure 7.** The RAGS, $A*$, and the global optimal (known only in hindsight) paths are shown in the figure with the final path costs. The goal is to traverse from the top left start vertex to the bottom right goal vertex. The empty field is a test case showing that both algorithms plan direct paths as expected in an obstacle-free environment. Note that although only the final executed RAGS path is shown, the entire non-dominated path set is considered during the online traversal.

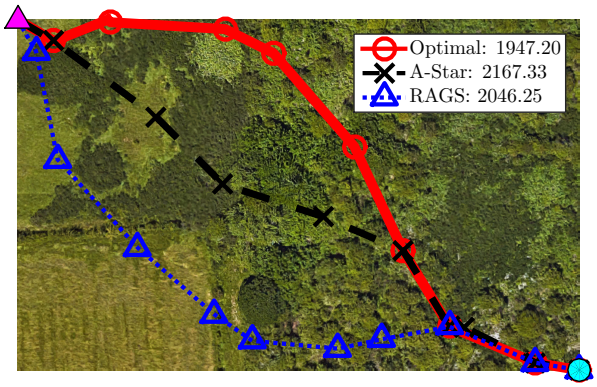



**Figure 9.** The RAGS path is shown traveling through an initial cluster of trees to take full advantage of the clear route over the open field. In contrast, $A*$ finds a path that attempts to navigate straight to the goal through the dense forest without consideration of the cost variability of that region. The costs after execution show that the RAGS path is actually cheaper due to balancing the risk of finding a path through the initial tree cluster that connects to a less risky path to goal. This demonstrates the benefit of RAGS, knowing when to take risks and when to act conservatively.

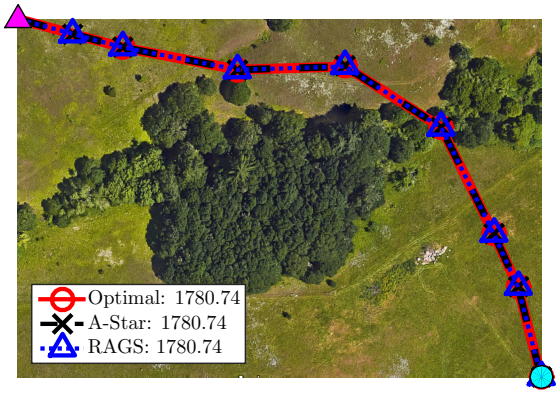



**Figure 8.** This test case demonstrates that both $A*$ and RAGS plan an intuitive path in the presence of obstacles. The planners remain in the empty fields for the duration of the path and only enter the forest briefly as expected.

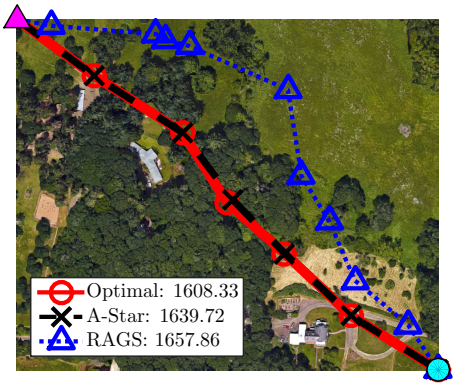



**Figure 10.** Paths are planned through a dense cluster of trees surrounding the goal. Here we can see RAGS plans around the cluster, where the path is ultimately shorter but has a higher risk, before traversing through a less cluttered area. RAGS takes advantage of the wide open region instead of searching for narrow tracks within the tree cluster.

and demonstrates their ability to plan a trajectory around obstacles.

Analyzing the paths found in Figure 9 is more interesting than the previous two examples. Here we start to see the benefit of RAGS in obstacle-dense environments. The path from start to goal is blocked by large clusters of semi-permeable forest. $A*$ executes a path through the center of the cluster that has a low cost in the mean but does not allow for easy deviations if the path is found to be untraversable. On the other hand, the path executed by RAGS demonstrates the nonmyopic nature of this algorithm. RAGS executes a path that travels around the main cluster to minimize the portion of the path within the dense section of the forest. The executed path balances the risk of traversing high-cost edges at the start and end of the path with the benefit of exploiting the sections of open field in the bottom left. Although the RAGS path length is longer than the $A*$ path, it ultimately achieves a lower total cost. Values are drawn from the edge distributions to calculate what would have been the optimal path in hindsight. The optimal path (known only after execution) is shown in red, and we can see that it also avoids the center route executed by $A*$.

The final test case can be seen in Figure 10, here $A*$ actually outperforms RAGS. In this test case the hindsight-optimal and $A*$ path is to push through the center of the cluster along edges with potentially higher penalties, but a shorter Euclidean distance. The RAGS planner alternatively takes a path that largely avoids the cluster along a safer, though ultimately slightly higher cost, path.

The compiled results for all tested algorithms are shown in a box plot of percentage above the hindsight optimal in Figure 11. From the comparison on the three randomly generated graphs in Section 4.3, we showed that the relative performance of RAGS increases as edge variance increases. This is accounted for by the fact that the other planning algorithms are merely searching over the single heuristic of mean traversal cost. If variance is low then this can be enough to solve for a path that is close to the optimal solution. However, Figure 11 reveals that real world data sets can contain significant noise, and it is valuable to account for that variability during planning.

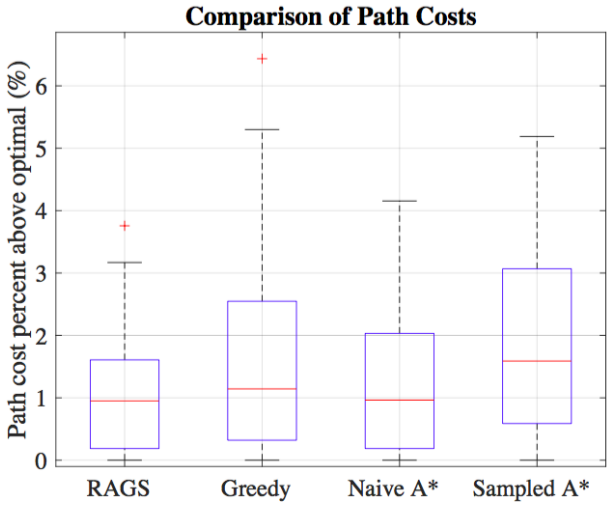



**Figure 11.** Results from satellite data experiments, using 100 samples of 96 images. Box plots represent the path cost percentage above what would have been, in hindsight, the optimal path. The same trends in performance are seen as with the simulated graphs. By accounting for the uncertainties in travel cost, RAGS is able to reduce the risk of executing expensive paths.

## 7 Flight Trials

In addition to the satellite data experiments, the RAGS planner was tested on a quadrotor traversing through an unstructured wooded environment. The quadrotor is provided a graph representation of the environment, consisting of the edge cost distributions calculated from satellite imagery of the area. It then uses either $A*$ to plan a route, or RAGS to generate the non-dominated path set, from the start to finish locations. The goal of the quadrotor is to fly to the finish location as quickly as possible while avoiding obstacles in the environment and following either the $A*$ or RAGS path, the latter of which is updated online given sensor feedback. The quadrotor uses on-board sensing and computing to plan its trajectory, detect and avoid obstacles, and re-plan as needed autonomously and in real time while navigating the path.

The quadrotor used in these experiments is the DJI Matrice M100 DJI (2017b) equipped with a ZED RGBD camera (StereoLabs 2017) and Nvidia Jetson TX2 Development Board (Nvidia 2017), as shown in Figure 12[†]. The Jetson TX2 uses Ubuntu 16.04 and Robot-Operating System (ROS) Kinetic-Kame (jkay 2017). The motion of the Matrice is restricted to a 2D plane by controlling the Matrice to a fixed altitude of 10 m AGL. The Matrice's heading is controlled to orient the ZED RGBD camera in the direction of travel; effectively always facing the Matrice forward. The ZED's depth image, accessed via the ZED-ROS-wrapper (Brehmer 2017), is used to detect obstacles at the quadrotor's operating altitude. As obstacles are detected, they are inflated and added to a 2D occupancy map at 100 Hz. The Matrice then uses the ROS navigation package to plan a path in the 2D occupancy map from the Matrice's current location to the next goal vertex in the $A*$ or RAGS route at 1 Hz.

The movement of the Matrice is controlled at 50 Hz using proportional-derivative controllers for heading, altitude, and horizontal velocities to follow the path to the next vertex. The DJI on-board ROS SDK (DJI 2017a) is used to acquire positioning information and control the motion of the Matrice. The Matrice's horizontal velocity is determined by the mean distance reading in the platform's operating plane. In an open environment with few obstacles the Matrice's velocity increases. As more obstacles are detected and become closer to the Matrice, the platform slows down. Using this method, the Matrice controller guides the platform along the path between vertices. When it reaches a distance within 3 m of the intermediate goal vertex, the Matrice queries either the $A*$ or RAGS on-board planner for the next vertex and repeats this process.

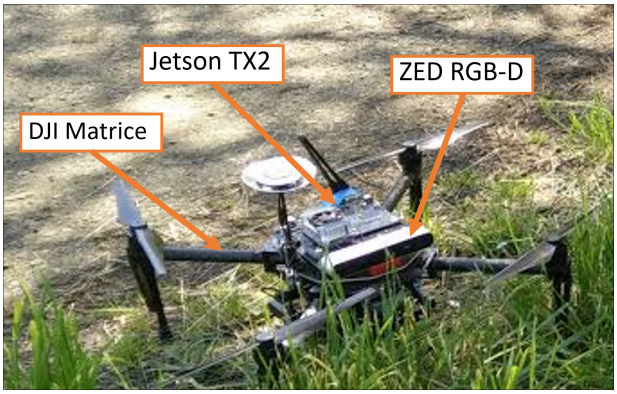



**Figure 12.** The DJI Matrice M100 used in the hardware trials.

The hardware experiments were conducted in the unstructured wooded environment shown in Figure 13. This environment was chosen because it provides multiple paths from the start location, green triangle, to the goal location, red circle. The length of the trial was constrained by the limited battery life of the Matrice and the foot speed of the safety pilot. As seen in Figure 13 the Matrice executed different paths for the RAGS and $A*$ planners, metrics are provided in Table 1. The RAGS planner took the slightly longer path, 209.7 m compared to 201.7 m for the $A*$ path. However, the RAGS path can be seen to be less risky, with fewer nearby obstacles which allowed the quadrotor to travel at a faster speed. The Matrice was able to complete the RAGS path in 162.7 s, compared to the $A*$ path, which was completed in 174.8 s. The time and distance reported is from when the Matrice autonomously departs the starting vertex and autonomously reaches within 3 m of the goal vertex.

**Table 1.** Comparison of results from hardware trials between RAGS and $A*$. Notice that although the RAGS path is 8.0 m longer, RAGS is 12.1 s faster as it selects a lower risk path containing fewer obstacles.

| **Method** | **Path Length** (m) | **Travel Time** (s) |
|---|---|---|
| RAGS | 209.7 | 162.7 |
| $A*$ | 201.7 | 174.8 |

We note that this experiment serves as a case study of the performance of RAGS and does not represent a statistically meaningful result. However, it does confirm

[†]The hardware flight trials conducted for this research were conducted under Oregon State University's (OSU) Federal Aviation Administration Certificate of Authorization and logged in OSU's compliance software, Drone Complier.

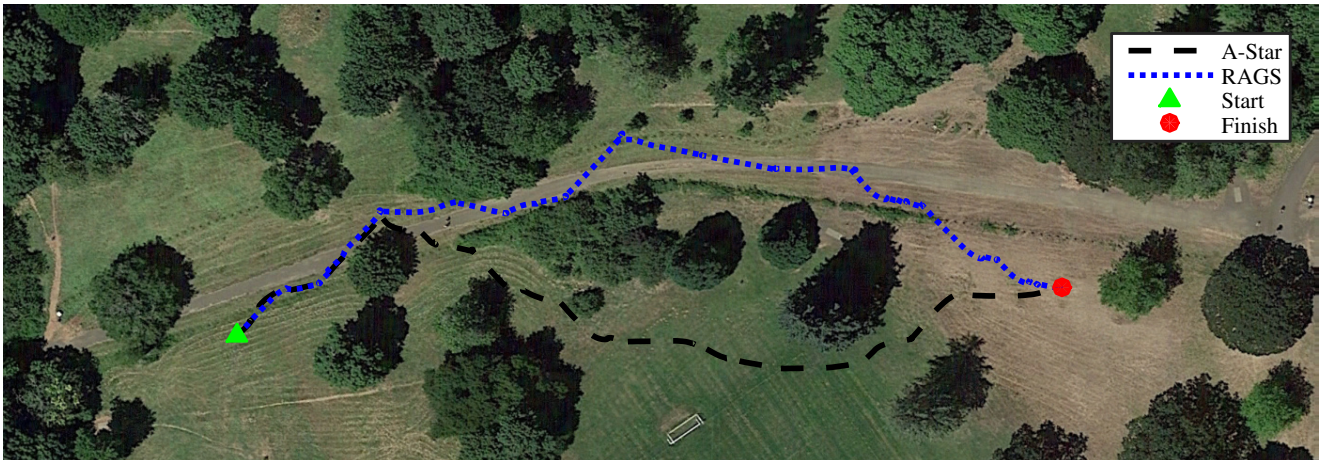



**Figure 13.** The paths taken in the hardware trials by the Matrice for the RAGS and $A*$ planners. The RAGS path is slightly longer, 209.7 m, compared to the $A*$ path, 201.7 m. However, the RAGS path allowed for a slightly higher travel speed and was completed in 162.7 s compared to the 174.8 s taken along the $A*$ path. Notice that although the RAGS path is 8.0 m longer than the $A*$ path, it is completed 12.1 s faster. This is because the RAGS planner accounts for risk along the path and chooses a path from the non-dominated set with fewer obstacles, allowing the quadrotor to travel at a higher speed.

a number of properties of the algorithm in a real-world field trial. First, these hardware trials verify that the RAGS algorithm can be run online in real time on existing hardware in an unstructured environment. Second, it confirms both the trends found in the simulations, that RAGS will attempt to mitigate risk by choosing a safer path, and our own intuition about how the RAGS algorithm should perform, i.e. it chooses the uncluttered path along the road over the shorter path through the trees, which is qualitatively safer and easier for the vehicle to traverse.

## 8 Conclusion

In this paper we have introduced a novel approach to incorporating and addressing uncertainty in planning problems. The proposed RAGS algorithm combines traditional deterministic search techniques with risk-aware planning. RAGS is able to trade off the number of future path options, as well as the mean and variance of the associated path cost distributions to make online edge traversal decisions that minimize the risk of executing a high-cost path. The algorithm was compared against existing graph search techniques on a set of graphs with randomly assigned edge costs, as well as over a set of graphs with traversability costs generated from satellite imagery data. In all cases, RAGS was shown to reduce the probability of executing high-cost paths over $A*$, *sampled* $A*$ and a *greedy* planning approach.

In addition to simulation experiments, we also implemented the RAGS algorithm in a hardware demonstration on board a DJI Matrice M100 platform equipped with a ZED RGBD camera. Our experiments compared the $A*$ and RAGS flight trajectories and showed that although the RAGS planner selected a longer path, the Matrice was able to complete the full RAGS trajectory in shorter time than the $A*$ trajectory. RAGS traded off a slightly longer path in order to maintain greater distance to the nearest obstacles. As a result, the Matrice was able to fly the RAGS path at faster speeds compared to flying the $A*$ path when the platform needed to slow down and maneuver around obstacles.

Our RAGS code is available open source at `https://github.com/osurdml/RAGS`.

### 8.1 Future Work

In this work, we have experimented on graphs with edge costs represented by normal distributions. However, the RAGS framework is designed to accommodate any edge cost distribution function. Indeed one argument against using normal distributions to represent costs in path planning problems is that they result in non-zero probabilities for sampling negative values, thus they may be a poor representation of the true cost distribution. Other distributions, such as the log-normal or beta distributions, may be more accurate. However, computing the non-dominated path set and the pairwise comparisons for general distributions is a non-trivial task that is worth further investigation.

Another promising extension of this work is to consider RAGS as applied to planning for information optimization. In this case, edge cost probability distributions would represent environmental information, and paths would be generated based on the amount of information the vehicle may collect in different parts of the map. The foreseen challenges of this work will be to manage the changes in expected information gain on-the-fly in a principled manner, since properties such as submodularity of the data will mean that future observations downstream no longer reap the gains expected at the start. We believe that incorporating information theoretic planning into the RAGS framework could potentially lead to wider applications in probabilistic planning.

## Acknowledgements

This research was conducted at Oregon State University and has been funded in part by NASA grant NNX14AI10G and Office of Naval Research grant N00014-14-1-0509.

## References


Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the PR2. In *Proc. IEEE International Conference on Robotics and Automation*, pages 5568–5575, 2011.

Alexandre Brehmer. Zed-ros-wrapper. http://wiki.ros.org/zed-ros-wrapper, 2017. Last Edited: 2016-11-28 12:05:30.

Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Proc. IEEE International Conference on Robotics and Automation*, pages 723–730, 2011.

Stefano Carpin, Yin-Lam Chow, and Marco Pavone. Risk aversion in finite Markov Decision Processes using total cost criteria and average value at risk. In *Proc. IEEE International Conference on Robotics and Automation*, pages 335–342, 2016.

Stephen M. Chaves, Jeffrey M. Walls, Enric Galceran, and Ryan M. Eustice. Risk aversion in belief-space planning under measurement acquisition uncertainty. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2079–2086, 2015.

DJI. Dji developer onboard sdk: Turn your aerial platform into an autonomous flying robot. https://developer.dji.com/onboard-sdk/, 2017a. Accessed : 2017-05-10.

DJI. Matrice 100: The quadcopter for developers. `https://www.dji.com/matrice100`, 2017b. Accessed : 2017-04-10.

Dave Ferguson, Anthony Stentz, and Sebastian Thrun. PAO for planning with hidden state. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 2840–2847, 2004.

Seyedshams Feyzabadi and Stefano Carpin. Risk-aware path planning using hierarchical constrained Markov Decision Processes. In *Proc. IEEE International Conference on Automation Science and Engineering*, pages 297–303, 2014.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Geoffrey A. Hollinger, Arvind A. Pereira, Jonathan Binney, Thane Somers, and Gaurav S. Sukhatme. Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles. *Journal of Field Robotics*, 33(1):47–66, 2016.

Ping Hou, William Yeoh, and Pradeep Reddy Varakantham. Revisiting risk-sensitive MDPs: New algorithms and results. In *Proc. International Conference on Automated Planning and Scheduling*, 2014.

jkay. Ros kinetic kame. http://wiki.ros.org/kinetic, 2017. Last Edited: 2016-05-23 22:07:18.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3): 354–363, 2005.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

Jean-Claude Latombe. *Robot Motion Planning*. Springer Science & Business Media, 2012.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

Steven I. Marcus, Emmanual Fernández-Gaucherand, Daniel Hernández-Hernandez, Stefano Coraluppi, and Pedram Fard. Risk sensitive Markov decision processes. In *Systems and Control in the Twenty-First Century*, pages 263–279. Springer, 1997.

Alexandra Meliou, Andreas Krause, Carlos Guestrin, and Joseph M. Hellerstein. Nonmyopic informative path planning in spatio-temporal models. In *Proc. AAAI Conference*, pages 602–607, 2007.

George E. Monahan. State of the Art—A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

Liz Murphy and Paul Newman. Risky planning on probabilistic costmaps for path planning in outdoor environments. *IEEE Transactions on Robotics*, 29(2):445–457, 2013.

Nvidia. Nvidia jetson: The embedded platform for autonomous everything. `http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html`, 2017. Accessed : 2017-05-10.

Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

George H. Polychronopoulos and John N. Tsitsiklis. Stochastic Shortest Path Problems with Recourse. *Networks*, 27:133–143, 1996.

Ryan Skeele, Jen Jen Chung, and Geoffrey A. Hollinger. Risk-aware graph search with dynamic edge cost discovery. In *Proc. 12th International Workshop on the Algorithmc Foundations of Robotics*, 2016.

StereoLabs. Zed 2k stereo camera: The world's first 3d camera for depth sensing and motion tracking. `https://www.stereolabs.com/`, 2017. Accessed : 2017-05-10.

Wen Sun, Sachin Patil, and Ron Alterovitz. High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Transactions on Robotics*, 31(1):104–116, 2015.