Augmented Motion Plans for Planning in Uncertain Terrains

Tsz-Chiu Au and **Ty Nguyen**
School of Electrical and Computer Engineering
Ulsan National Institute of Science and Technology
Ulsan, South Korea 689-798
{chiu, tynguyen}@unist.ac.kr

Abstract

In exploration of unknown planets, ground vehicles such as Mars rovers may not know exactly what terrain they will run into, causing great difficulty in meeting their goals. This paper presents a two-stage approach for motion planning in uncertain terrains. In the first stage, we utilize a specialized planner to generate motion plans to meet some arrival requirements. In the second stage, we augment the motion plans with sensing information and combine them to form a full-fledged controller in order to cope with uncertainty in the environment. This separation of planning and uncertainty management can simplify the development of planners for complicated goals. Our preliminary experiments showed that a vehicle can meet the arrival requirements with a high probability in small random graphs.

Introduction

Our next frontier of space exploration is near earth objects such as Mars and comets. Currently, we rely on robot rovers to explore these extraterrestrial lands. However, without full knowledge of the environment on these lands, it is hard to generate motion plans for these rovers to achieve their objectives. For example, if the vehicle in Figure 1 can only detect the paths within its sensors' range (the orange pie shape), how can it arrive at the destination without knowing entire graph of feasible paths? In this paper, we focus on planning to move a vehicle or a rover to its destination in order to meet certain requirements at the destination, under the condition that the terrain on its way is uncertain.

There have been many works on motion planning under uncertainty in robotics (e.g., (Rekleitis, Meger, and Dudek 2006; Nakhaei and Lamiraux 2008; Um et al. 2013)), but few of them considered complex arrival requirements. Some variants of the popular probabilistic planning methods such as probabilistic roadmap (PRM) (Missiuro and Roy 2006; Kneebone and Dearden 2009) and rapidly-exploring random trees (Melchior and Simmons 2007; Maeda, Singh, and

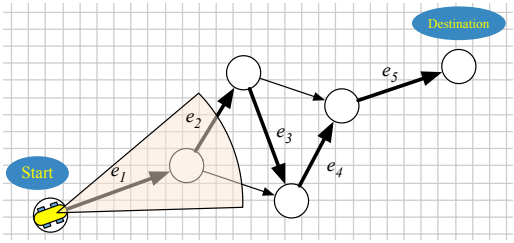

Figure 1: A rover traverses an unknown 3D-terrain to reach the destination. The roads on the terrain are modeled as a directed acyclic graph. The orange pie shape is the sensor's range.

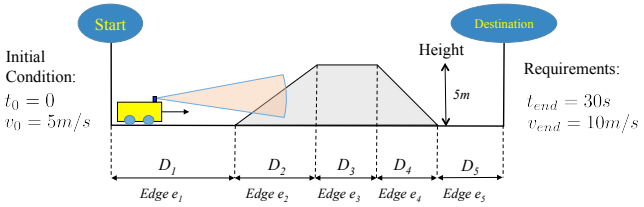

Figure 2: The cross-section of the terrain along the bold path in Figure 1.

Durrant-Whyte 2011; Nikitenko et al. 2013) can be used to generate motion plans in uncertain environments. However, their underlying heuristics provide no guarantee of success. While the framework of POMDPs (e.g., (Ong et al. 2010)) is rich enough to encode planning problems under uncertainty with arbitrary utility functions, some goal conditions can be too complicated for general-purpose POMDP solvers to handle. In fact, without using specialized planners it is hard to satisfy certain goal conditions. We therefore propose a two-stage approach for motion planning under uncertainty: First, use an efficient planner to generate motion plans that meet the arrival requirements in uncertainty-free environments. Second, *combine* the solution plans generated by the planner to form a controller that can work in uncertain environments. Our approach is inspired by the work on the synthesis of strategies from interaction traces in (Au, Kraus, and Nau 2008), which outlined the foundation of our approach: by augmenting motion plans with simulated sensing

information in some possible worlds, these motion plans can be combined to form a full-fleged controller that will meet the arrival requirements in these possible worlds. The separation of the planning and uncertainty management is a key advantage of our approach.

This paper is organized as follows. First, we define the motion planning problem in uncertain terrains. Second, we discuss how to combine the motion plans to cope with uncertain terrains. Our algorithm will then be evaluated experimentally with small random graphs, and the preliminary results will be presented. Finally, we go over some related work and summarize our contributions.

Problem Definition

We assume the terrain has a fixed set of roads, and off-road riding is not allowed. The road network can be modeled as a directed acyclic graph as shown in Figure 1, so that each road corresponds to an edge in the graph. The terrain is 3-dimensional, meaning that each road can have different slopes (see Figure 2). We assume that the *homogeneity* assumption hold in individual roads (i.e., the road condition would not change when the vehicle traverses the road). If a road does not satisfy the homogeneity assumption, we will model it as a sequence of edges in the graph such that each edge satisfies the homogeneity assumption.

As in (Au, Quinlan, and Stone 2012), we consider autonomous vehicles that are controlled by PID-controllers or other non-linear controllers whose control signals are *setpoints*, which are the target velocities for the vehicle. After setting a new setpoint $\hat{v}$, the velocity will not change to $\hat{v}$ immediately; instead it takes a while for the vehicle to settle down at $\hat{v}$. For planning purposes, it is essential to know how long the vehicle takes to settle at $\hat{v}$. We therefore rely on a *performance model* $(T^{\mathsf{stable}}, D^{\mathsf{stable}})$, which conservatively measures the time and distance the vehicle takes to settle at $\hat{v}$ after changing the setpoint. More precisely, the *stable time* $T^{\mathsf{stable}}(v,\hat{v})$ is the *maximum* time the vehicle takes to stabilize at $\hat{v}$, and the *stable distance* $D^{\mathsf{stable}}(v,\hat{v})$ is the *average* distance the vehicle travels for a period of time $T^{\mathsf{stable}}(v,\hat{v})$ after changing the setpoint. Performance models are subject to the road condition, and therefore they can be different on different roads. Assume that we have already obtained the performance model of every road, using the estimation method in (Au, Quinlan, and Stone 2012).

In longitudinal control, a vehicle moves along a road according to a time-dependent velocity function $\hat{v}(\cdot)$ called *setpoint schedule*, where $\hat{v}(t)$ is the setpoint for the vehicle at time t. (Au, Quinlan, and Stone 2012) has discussed at length how to generate a setpoint schedule to control a vehicle to arrive at a specific position on an one-dimensional path at a given arrival time and velocity. But this notion of motion plans is not sufficient for the traversal of a graph since a vehicle also needs to decide which outgoing edges the vehicle should choose when it reaches a node. We therefore extend the notion of setpoint schedules to include control signals which decide which path the vehicle should take in a directed graph.

Consider a vehicle traversing a directed acyclic graph $G=(N,E)$, following a path starting at node n_0 and ending at node n_{end} (n_{end} is called the *destination*). Suppose the vehicle moves along the path using a setpoint schedule $\hat{v}(\cdot)$. In this paper, we assume the controller can act at discrete time points only. Hence, $\hat{v}(\cdot)$ can be represented as a sequence of pairs $\langle (t_0,v_0),(t_1,v_1),\ldots,(t_n,v_n)\rangle$, which means that the vehicle should choose the setpoint v_i at time t_i, for $0 \le i \le n$. We augment the setpoint schedule with information about which outgoing edges the vehicle should choose at each node as follows: a *motion plan* is $\langle (t_0,a_0),(t_1,a_1),\ldots,(t_n,a_n)\rangle$, where $a_i = \langle v_i, e_i\rangle$ is the control signals called an *action* at time t_i. An action a_i is a vector with two components: $\langle v_i, e_i\rangle$, where 1) v_i is the setpoint at time t_i, and 2) e_i is one of the outgoing edges of the node at which the vehicle is located at time t_i. If the vehicle is not located at a node at time t_i or the node it locates at time t_i has no outgoing edges, $e_i = \mathsf{nil}$.

Given a graph $G=(N,E)$, a starting time t_0, and a starting velocity v_0 at n_0, our goal is to generate a motion plan π such that the vehicle will reach the destination n_{end} while satisfying a goal condition $\mathscr{G}$. This motion plan is, of course, subject to the speed limit v_{max} as well as physical constraints of the vehicle as described in the performance model $(T_e^{\mathsf{stable}}, D_e^{\mathsf{stable}})$ of each edge $e \in E$. Formally, we define our problem as follows. A *validation problem* $\mathscr{P}_{\mathsf{valid}}$ is a 4-tuple $\langle (t_0,v_0), \Gamma, v_{\mathsf{max}}, \mathscr{G}\rangle$, where

- (t_0,v_0) is the *initial configuration*;
- $\Gamma = (G, \{(D_e, T_e^{\mathsf{stable}}, D_e^{\mathsf{stable}})\}_{e\in E})$ is the specification of the graph $G=(N,E)$, where D_e is the length of the edge $e \in E$, T_e^{stable} is the stable time function of e, and D_e^{stable} is the stable distance function of e;
- v_{max} is the speed limit of all edges; and
- $\mathscr{G}$ is the goal condition.

We use the velocity function $v(\cdot)$ to denote the velocity of the vehicle over time. We say $v(\cdot)$ is *constructible* if there exists a setpoint schedule $\hat{v}(\cdot)$ such that the velocity function is $v(\cdot)$ if the vehicle follows $\hat{v}(\cdot)$. Let $\rho = \langle e_1, e_2, \ldots, e_m\rangle$ be a path in G connecting n_0 to n_{end}. A velocity function $v(\cdot)$ is *feasible* for a path ρ if it satisfies the following constraints:

C1) $v(t_0) = v(0) = v_0$;
C2) $0 \le v(t) \le v_{\mathsf{max}}$ for $0 \le t \le t_{\mathsf{end}}$, where t_{end} is the arrival time (i.e., the velocity cannot exceed the speed limit or be negative at any time);
C3) $\int_{t_{e_i}}^{t_{e_{i+1}}} v(t)\,dt = D_{e_i}$ for all edge e_i on ρ, where t_{e_i} is the time the vehicle reaches e_i according to $v(\cdot)$ along ρ and $t_{e_{m+1}} = t_{\mathsf{end}}$ (i.e., the distance traveled on an edge e_i must be equal to the length D_i of e_i);
C4) $v(\cdot)$ is constructible; and
C5) $\mathscr{G}$ is true.

A setpoint schedule $\hat{v}(\cdot)$ is *feasible* for a path ρ if the velocity function constructed by $\hat{v}(\cdot)$ is feasible for ρ. A motion plan (say $\pi = \langle (t_i, \langle v_i, e_i\rangle)\rangle_{i=0..n}$) is *feasible* if the corresponding setpoint schedule $(\langle (t_i, v_i)\rangle_{i=0..n})$ is feasible for the corresponding path $(\langle e_i : 0 \le 0 \le n \text{ and } e_i \ne \mathsf{nil}\rangle)$. The objective of $\mathscr{P}_{\mathsf{valid}}$ is to decide whether a feasible motion plan exists. Notice that $\mathscr{P}_{\mathsf{valid}}$ has not yet taken uncertainty into account.

Motion Planning in Uncertain Terrains

This section concerns with situations in which the terrain is not fully observable. At the beginning, the vehicle knows nothing about the terrain beyond the range of its sensors, except the location of the destination n_{end}. Suppose the vehicle starts with a belief about the set $\mathbb{G}$ of *possible graphs*. The belief is defined in terms of a probability distribution P over $\mathbb{G}$, which means that the probability that a possible graph $G \in \mathbb{G}$ is the *real* graph is $P(G)$. As the vehicle traverses the graph, it gathers more and more information about the terrain over time. This information will be helpful for the vehicle to ascertain which graph in $\mathbb{G}$ is real.

Let $\mathbb{G} = \{G_1, G_2, \ldots, G_{m_G}\}$ where $G_i = (N_i, E_i)$ for $1 \leq i \leq m_G$. Some of the nodes and edges are shared by multiple graphs in $\mathbb{G}$. Let $N^s = \bigcup_{1 \leq i \leq m_G} \{N_i\}$ and $E^s = \bigcup_{1 \leq i \leq m_G} \{E_i\}$. The union of all graphs in $\mathbb{G}$ forms a supergraph $G^s = (N^s, E^s)$. Consider the 2D rectangular region R that physically contains G^s. We subdivide the region into a $l_x \times l_y$ grid as shown in Figure 1. Each cell in the grid will generate a *signal* when the sensors on the vehicle gather information about the cell. The signal of a cell reflects some features of the terrain at the cell (e.g., landmarks). From the sensors' viewpoint, a terrain is a mapping $T : [1 \ldots l_x] \times [1 \ldots l_y] \to S$, which specifies the signals at all cells in R, where S is the set of all possible signals. Each possible graph $G_i \in \mathbb{G}$ is associated with a terrain T_i. Suppose the real graph is G_i. At the beginning, the vehicle has only partial knowledge of T_i. The vehicle will use its sensors to gather more information about T_i during traversal. We will make use of the *monotonicity assumption*: Any new knowledge from sensors will not contradict the existing ones.

Suppose the vehicle follows a feasible motion plan $\pi = \langle (t_0, a_0), (t_1, a_1), \ldots, (t_n, a_n) \rangle$, where $a_i = \langle v_i, e_i \rangle$. We assume sensing actions interleave with the execution of actions: Before the execution of an action a_i at time t_i, the vehicle obtains a percept b_i from the environment. We define an *augmented motion plan* as $\tau = \langle (a_1, b_1), (a_2, b_2), \ldots (a_k, b_k) \rangle$, which is basically an *interaction trace* between the vehicle and the environment. Now we make use of the results in (Au, Kraus, and Nau 2008), which states that a set of interaction traces, under certain conditions, can be combined to form an *agent* that will succeed in environments in which the interaction traces are generated. Furthermore, if we carefully select the interaction traces, we can increase the probability that the agent will succeed in an uncertain environment. Algorithm 1 is the architecture of the vehicle's controller that makes use of augmented motion plans. In Algorithm 1. $\mathsf{action}_i(\tau)$ and $\mathsf{percept}_i(\tau)$ be the i'th action and the i'th percept of τ, respectively.

According to (Au, Kraus, and Nau 2008), the interaction traces can be combined to form a *composite agent* if they are *mutually compatible*. In the same vein, we define the compatibility of augmented motion plans as follows.

Definition 1 *Let* $\mathsf{lcp}(\alpha^a, \alpha^b)$ *be the* longest common prefix *of two finite sequences* $\alpha^a = \langle c_1^a, c_2^a, \ldots, c_{k_a}^a \rangle$ *and* $\alpha^b = \langle c_1^b, c_2^b, \ldots, c_{k_b}^b \rangle$, *such that* $\mathsf{lcp}(\alpha^a, \alpha^b) = \langle c_1, c_2, \ldots c_k \rangle$ *where* $c_i = c_i^a = c_i^b$ *for* $1 \leq i \leq k$ *and either (1)* $k = k_a < k_b$, *(2)* $k = k_b < k_a$, *or (3)* $c_{k+1}^a \neq c_{k+1}^b$.

Algorithm 1 The architecture of the vehicle's controller.

```
1:  procedure VehicleController(𝒯)
2:      i := 1; 𝒯_i := 𝒯
3:      while the vehicle has not reached n_end do
4:          if the current time is t_i then
5:              Obtain a percept b_i from sensors.
6:              𝒯_{i+1} := ∅
7:              for all τ ∈ 𝒯_i do
8:                  if b_i = percept_i(τ) then
9:                      𝒯_{i+1} := 𝒯_{i+1} ∪ {τ}
10:             A_i := {action_i(τ) : ∀τ ∈ 𝒯_{i+1}}
11:             if |A_i| ≠ 1 then
12:                 Return Fail since 𝒯 is not compatible.
13:             Let the unique action in A_i be ⟨v_i, e_i⟩
14:             if e_i ≠ nil then steer the vehicle to edge e_i.
15:             Change the current setpoint to v_i
16:             i := i + 1
```

Definition 2 *Two augmented motion plans* τ_1 *and* τ_2 *are* compatible *if and only if* $|\mathsf{lcp}(\mathsf{action}(\tau_1), \mathsf{action}(\tau_2))| > |\mathsf{lcp}(\mathsf{percept}(\tau_1), \mathsf{percept}(\tau_2))|$, *where* $\mathsf{action}(\langle (a_i, b_i) \rangle_{i..k}) = \langle a_i \rangle_{i=1..k_a}$ *and* $\mathsf{percept}(\langle (a_i, b_i) \rangle_{i..k}) = \langle b_i \rangle_{i=1..k_b}$.

Definition 3 *A set* $\mathscr{T}$ *of augmented motion plans is* compatiable *if and only if every pair of augmented motion plans in* $\mathscr{T}$ *is compatiable.*

Theorem 1 states that if the input $\mathscr{T}$ of Algorithm 1 is compatible, the algorithm will never return Fail from Line 12. The proof of the theorem is similar to the proof in Theorem 2 in (Au, Kraus, and Nau 2008).

Theorem 1 *Algorithm 1 will not fail if 1)* $\mathscr{T}$ *is compatible, and 2) one of the augmented motion plans in* $\mathscr{T}$ *is generated by the real graph.*

Theorem 1 implies that if we can find a compatible set $\mathscr{T}$ of *feasible* augmented motion plans, Algorithm 1 will *always* be able to reach its destination while satisfying the goal condition, as long as the real graph is one of the graphs in which some feasible augmented motion plans in $\mathscr{T}$ are generated. Of course, we assume that the underlying planner for the validation problem can generate motion plans that satisfy the goal condition. For instance, if we concern with arriving at the destination at a specific time and at a specific velocity, the algorithms presented in (Au, Kraus, and Nau 2008) will work.

The *probability of success* of Algorithm 1 is equal to $\sum_{\tau_i \in \mathscr{T}} \{P(G_i)\}$, where τ_i is a feasible augmented motion plan generated in G_i. As can be seen, if $\mathscr{T}$ includes one augmented motion plan in every $G \in \mathbb{G}$, the probability of success is 1—the vehicle can guarantee to arrive at the destination in uncertain terrains. However, not every pair of augmented motion plans is compatible. In fact, there can be two possible graphs whose sets of all feasible augmented motion plans are disjoint, meaning that it is *impossible* to have a 100% successful rate. In these uncertain terrains, we can only hope to maximize the probability of success by finding

Algorithm 2 The greedy selection of compatible augmented motion plans.

```
1: procedure FINDCOMPATIABLEPLANS(𝔾)
2:     𝒯 := ∅
3:     for G_i ∈ 𝔾 in descending order of P(G_i) do
4:         for j = 1 to K do
5:             Randomly generate a motion plan π in G_i
6:             Simulate percepts as vehicle follows π in G_i
7:             Let τ be the augmented motion plan of π.
8:             if τ is compatible with all τ' ∈ 𝒯 then
9:                 𝒯 := 𝒯 ∪ {τ}; Break
       return 𝒯
```

a large $\mathscr{T}$ that covers as many possible graphs as possible. Hence, we present a greedy algorithm to find such $\mathscr{T}$. The algorithm, as shown in Algorithm 2, considers the possible graph in the descending order of probability, and then randomly generates K augmented motion plans in these graphs. The augmented motion plan will be added to $\mathscr{T}$ as long as it is compatible with all plans in $\mathscr{T}$. Although the algorithm cannot guarantee to find $\mathscr{T}$ that maximizes the probability of success, it can often find a good set of augmented motion plans with a high probability of success, as shown in the experimental results in the next section.

Preliminary Experimental Results

To evaluate Algorithm 1 and Algorithm 2, we conducted a simulation experiment with four different random graphs. One of the random graphs is shown in Figure 3. First, a directed acyclic graph G^s that connects n_0 to n_{end} was generated by randomly connecting a fixed number of nodes (Graph a). G^s has to be solvable (i.e, there exist paths connecting n_0 to n_{end}). Second, we randomly chose the road condition T^{stable} and D^{stable} for each edge, and assigned landmarks to each of the edges, such that the landmarks on two different edges are different. Third, we randomly removed some edges in G^s to form graphs (Graph b-e). The landmarks on the removed edges were removed too. We repeated the removal of edges from G^s four times to generate a set $\mathbb{G}$ of four possible graphs.

After generating $\mathbb{G}$, we set the starting time $t_0 = 0s$ and the starting velocity $v_0 = 0m/s$. Our goal $\mathscr{G}$ is to reach n_{end} at time $t_{\text{end}} = 100s$ and $v_{\text{end}} = 40m/s$. We devised a motion planning algorithm to generate one augmented motion plan for each possible graph. Then we randomly assigned a probability distribution P over $\mathbb{G}$, and ran Algorithm 2 to find a set $\mathscr{T}'$ of compatible augmented motion plans. We ran Algorithm 1 with $\mathscr{T}'$ 100 times and measured the probability of success. We repeated the measurement 6 times with different probability distributions. The successful rates and their 95% confidence intervals are shown in Table 1.

As can be seen, the success rate of our approach is often higher than 90%, with an overall successful rate of 97.4% While different probability distributions of $\mathbb{G}$ produce similar successful rates, the successful rates heavily depend on the topology of the original supergraph G^s. In G^s_2, the successful rates are 100%, meaning that the augmented motion plans of the possible graphs are highly compatible.

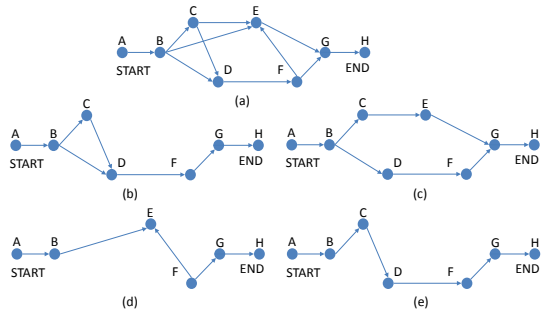

Figure 3: Graph a is G^s, and Graph b-e are the possible graphs.

Table 1: The successful rates of Algorithm 2.

Graph G^s_1		Graph G^s_2	
Prob. Distribution	Success Rate	Prob. Distribution	Success Rate
(0.5,0.2,0.1,0.2)	94.1% ± 5.2	(0.5,0.2,0.1,0.2)	100% ± 0
(0.4,0.3,0.2,0.1)	95.1% ± 3.1	(0.4,0.3,0.2,0.1)	100% ± 0
(0.2,0.1,0.3,0.4)	93.2% ± 3.6	(0.2,0.1,0.3,0.4)	100% ± 0
(0.1,0.5,0.2,0.2)	90.0% ± 3.7	(0.1,0.5,0.2,0.2)	100% ± 0
(0.2,0.1,0.4,0.3)	90.3% ± 2.8	(0.2,0.1,0.4,0.3)	100% ± 0
(0.3,0.3,0.1,0.3)	89.6% ± 2.2	(0.3,0.3,0.1,0.3)	100% ± 0
Graph G^s_3		Graph G^s_4	
Prob. Distribution	Success Rate	Prob. Distribution	Success Rate
(0.5,0.2,0.1,0.2)	98.6% ± 2.3	(0.5,0.2,0.1,0.2)	100.0% ± 0.0
(0.4,0.3,0.2,0.1)	99.1% ± 1.3	(0.4,0.3,0.2,0.1)	99.1% ± 0.4
(0.2,0.1,0.3,0.4)	99.4% ± 0.8	(0.2,0.1,0.3,0.4)	99.5% ± 0.7
(0.1,0.5,0.2,0.2)	99.5% ± 0.7	(0.1,0.5,0.2,0.2)	99.5% ± 0.6
(0.2,0.1,0.4,0.3)	99.3% ± 0.7	(0.2,0.1,0.4,0.3)	99.8% ± 0.4
(0.3,0.3,0.1,0.3)	98.3% ± 1.0	(0.3,0.3,0.1,0.3)	99.0% ± 0.5

Related Work

Probabilistic roadmap methods (PRM) (Kavaki et al. 1996) and rapidly-exploring random trees (LaValle and James J. Kuffner 2000) are both widely used, sampling-based algorithms. These algorithms are incomplete, but some extensions have been made to turn them into complete algorithms. Hirsch and Halperin (2004) and Zhang et al. (2007) proposed a hybrid motion planner that generates complete solutions with PRM. Nonetheless, these modified algorithms will suffer from inefficiency due to their completeness, and the arrival requirement has to be quite simple (e.g., arrive at a position without time and velocity requirement). While interleaving planning and execution is a good strategy to deal with uncertainty in planning (e.g., (Pivtoraiko, Mellinger, and Kumar 2013)), replanning cannot correct wrong decisions in previous steps, thus it is hard to provide any arrival guarantees.

TPOPEXEC (Muise, Beck, and McIlraith 2013) introduces a two-stage approach for planning: First, an offline preprocessor takes a partial-order plan and a set of temporal

constraints to produce a generalized representation. Second, an online component called EXECUTOR selects a temporally consistent, valid plan fragment from the generalized plan. Choset et al. (2000) presented a sensor-based motion planning approach based on a roadmap called hierarchical generalized Voronoi graph (HGVG), which can be incrementally constructed using only line-of-sight sensor data. This approach guarantees that the robot can find a path from start to goal or report that such a path is not feasible. Luna et al. (2014) introduces a two-stage framework for efficient computation of an optimal control policy in the presence of uncertainty. It first generates a bounded-parameter Markov decision process (BMDP) over a discretization of the environment and then quickly selects a local policy within each region to optimize a continuously valued reward function online. As the sensors gather more information about the environment, the BMDP is updated accordingly and the global control policy is recomputed. However, none of the above approaches concern with arrival requirements other than reaching the destination.

Concluding Remarks

In this paper, we proposed a two-stage approach for motion planning in uncertain terrains. More specifically, we proposed to augment motion plans with sensing information and then combine them, in a greedy manner, to form a controller that can handle uncertainty in the environment. This separation of planning and uncertainty management can greatly simplify our task, as we can utilize existing fast planners to generate motion plans that satisfy the goal conditions. In space exploration applications, the goal conditions can be quite specific. Instead of modifying existing planning algorithms for these applications to deal with uncertain terrains, we propose to adopt them directly and combine their solutions to form a contingency controller. If there are enough augmented motion plans, the controller can achieve a high success rate.

Our approach has two drawbacks. First, the number of contingencies can be quite large in an uncertain environment, meaning that we may need a lot of augmented motion plans in order to deal with all these contingencies. However, we believe that in some environments, a small number of augmented motion plans is sufficient because one augmented motion plan can deal with several different contingencies in different possible graphs. We intend to evaluate this possibility in the future. Second, we usually do not know the set of all possible graphs in $\mathbb{G}$ ahead of time, thus some possible graphs are not considered by Algorithm 2. As discussed in (Au, Kraus, and Nau 2008), we can use a backup planner to handle these unknown cases. In the future, we intend to improve our algorithms to address these issues.

References

Au, T.-C.; Kraus, S.; and Nau, D. 2008. Synthesis of strategies from interaction traces. In *Proceedings of the International Joint Conferenceon Autonomous Agents and Multi Agent Systems (AAMAS)*, 855–862.

Au, T.-C.; Quinlan, M.; and Stone, P. 2012. Setpoint scheduling for autonomous vehicle controllers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2055–2060.

Choset, H., and Burdick, J. 2000. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research* 19(2):96–125.

Hirsch, S., and Halperin, D. 2004. Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. In *Algorithmic Foundations of Robotics V*. 239–255.

Kavaki, L. E.; Švestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4).

Kneebone, M., and Dearden, R. 2009. Navigation planning in probabilistic roadmaps with uncertainty. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 209–216.

LaValle, S. M., and James J. Kuffner, J. 2000. Rapidly-exploring random trees: progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, 293–308.

Luna, R.; Lahijanian, M.; Moll, M.; and Kavraki, L. E. 2014. Optimal and efficient stochastic motion planning in partially-known environments. In *AAAI Conf. on Artificial Intelligence*.

Maeda, G. J.; Singh, S. P. N.; and Durrant-Whyte, H. 2011. A tuned approach to feedback motion planning with RRTs under model uncertainty. In *ICRA*, 2288–2294.

Melchior, N., and Simmons, R. 2007. Particle rrt for path planning with uncertainty. In *ICRA*, 1617–1624.

Missiuro, P. E., and Roy, N. 2006. Adapting probabilistic roadmaps to handle uncertain maps. In *ICRA*, 1261–1267.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2013. Flexible execution of partial order plans with temporal constraints. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 2328–2335. AAAI Press.

Nakhaei, A., and Lamiraux, F. 2008. A framework for planning motions in stochastic maps. In *Proceeding of International Conference on Control, Automation, Robotics and Vision*.

Nikitenko, A.; Riga, L.; Ekmanis, M.; and Liekna, A. 2013. Rrts postprocessing for uncertain environments. In *Proceedings of International Conference on Systems, Control and Informatics*, 171–179.

Ong, S. C. W.; Png, S. W.; Hsu, D.; and Lee, W. S. 2010. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research* 29(8):1053–1068.

Pivtoraiko, M.; Mellinger, D.; and Kumar, V. 2013. Incremental micro-uav motion replanning for exploring unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2452–2458.

Rekleitis, I.; Meger, D.; and Dudek, G. 2006. Simultaneous planning, localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems* 54(11):921–932.

Um, D.; Gutiérrez, M. A.; Bustos, P.; and Kang, S. 2013. Simultaneous planning and mapping (spam) for a manipulator by best next move in unknown environments. In *IROS*, 5273–5278.

Zhang, L.; Kim, Y. J.; and Manocha, D. 2007. A hybrid approach for complete motion planning. In *IEEE/RSJ International conference on Intelligent Robots and Systems*.