Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies

Lucas Janson, Tommy Hu, and Marco Pavone

Abstract—This paper addresses the problem of planning a safe (i.e., collision-free) trajectory from an initial state to a goal region when the obstacle space is a-priori unknown and is incrementally revealed online, e.g., through line-of-sight perception. Despite its ubiquitous nature, this formulation of motion planning has received relatively little theoretical investigation, as opposed to the setup where the environment is assumed known. A fundamental challenge is that, unlike motion planning with known obstacles, it is not even clear what an optimal policy to strive for is. Our contribution is threefold. First, we present a notion of optimality for safe planning in unknown environments in the spirit of comparative (as opposed to competitive) analysis, with the goal of obtaining a benchmark that is, at least conceptually, attainable. Second, by leveraging this theoretical benchmark, we derive a pseudo-optimal class of policies that can seamlessly incorporate any amount of prior or learned information while still *guaranteeing* the robot never collides. Finally, we demonstrate the practicality of our algorithmic approach in numerical experiments using a range of environment types and dynamics, including a comparison with a state of the art method. A key aspect of our framework is that it automatically and implicitly weighs exploration versus exploitation in a way that is optimal with respect to the information available.

I. INTRODUCTION

This paper addresses the problem of planning a safe (i.e., collision-free) trajectory from an initial state to a goal region when the obstacles in between are a priori unknown and are instead revealed online, e.g., through line-of-sight perception. A fundamental challenge is that, unlike motion planning with known obstacles or obstacles whose uncertainty is fully modeled probabilistically, when parts of the configuration space are simply unknown it is not even clear what an optimal policy to strive for (through, e.g., asymptotic convergence) is. One of the main goals of this paper is to address this shortcoming in the literature by defining a notion of optimality for use as a benchmark and also for conceptual guidance. We then follow this conceptual guidance to propose a novel algorithm for planning in unknown environments that produces low-cost solutions by flexibly incorporating side information about the environment, is guaranteed to be collision-free (even if the side information is incorrect), and requires on the order of 0.5–1s of (serial) computation time per action.

Related Work: Most algorithms for motion planning assume full knowledge of the environment, and can not be used, at least directly, to find a motion plan within an incomplete map [17, 18]. A common heuristic approach is to set temporary goals along the way to the goal region, and re-compute in
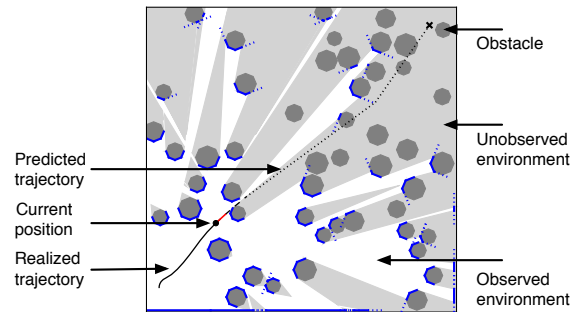
Fig. 1. Planning in unknown environments: as the robot moves (in this example, with double-integrator dynamics and line-of-sight perception), it needs to weigh exploration versus exploitation in a way that is best with respect to the information available. In particular, the robot should never take an action that leads into an inevitable collision state with respect to any unobserved part of the environment.

a receding horizon fashion motion plans within the portion of the environment that has been observed (a technique also referred to as re-planning). An instantiation of this approach is represented by frontier-based exploration, in which the temporary goals are placed at the boundary between the observed and unobserved environment (e.g., on the nearest frontier [1]; see also [6] for more sophisticated heuristics for temporary goal placements). Such a receding horizon approach is also quite effective in the presence of moving obstacles or, more in general, dynamic environments [13, 8, 25].

If one considers a kinodynamic model of a robot (as is the case in this paper), an additional consideration is how to ensure safety when the environment is only partially known. An important concept in this regard is the notion of inevitable collision states (ICS) [10], states for which, no matter what future control inputs the robot applies, a collision with an obstacle must eventually occur. To ensure safety, the robot should then never take an action that leads into an inevitable collision state *with respect to any unobserved part of the environment* (see Figure 1). The notion of planning in unknown environments under ICS constraints has been studied by a number of authors, see, e.g., [11, 10, 21, 2], with a focus on making the computation of ICS tractable.

An additional important consideration is how to reason about actions and/or observations that *might* be taken in the unobserved environment [12]. This is directly related to the notion of selecting actions that best weigh exploration versus exploitation as a function of available information, such as from perception. A possible approach is to bias motion plans toward "next best view" states [7], which capture the notion of amount of unobserved environment that can be seen at a future state. For example, this idea is infused within a receding

horizon planning scheme in [4]. A conceptually similar idea is to choose temporary goals that reward exploration [12]. More in general, one can frame the problem of planning in unknown environments as a partially observable Markov decision process, where the partial observability is with respect to the environment [22]. To make the problem tractable, one needs to consider a number of approximations, for example, replacing collision avoidance constraints with penalties on collision probabilities (possibly learned [22]).

Statement of Contributions: Despite the wealth of algorithms available for planning under uncertainty, and the well-established theoretical foundations for planning in known environments (e.g., [15, 14, 20]), the problem of planning in unknown environments has received relatively little theoretical investigation. Accordingly, the contribution of this paper is threefold. First, we present a notion of optimality for planning in unknown environments in the spirit of *comparative* [16] (as opposed to competitive [5, 9]) analysis, with the goal of obtaining a benchmark that is, at least conceptually, attainable. Second, by leveraging the theoretical benchmark and combining the aforementioned notions of re-planning, ICS, and forward-looking biasing in a self-contained framework, we derive a pseudo-optimal class of policies that can seamlessly incorporate any amount of prior or learned information while still *guaranteeing* the robot never collides. Finally, we demonstrate the practicality of our algorithmic approach in numerical experiments using a range of environment types and robot dynamics, including a comparison with a state of the art method for planning in unknown environments [22]. Importantly, computation times are on the order of 0.5–1s of (serial) computation time per action, making it an algorithm amenable to real-time implementation.

Organization: This paper is structured as follows. In Section II we provide notation for a number of concepts needed to rigorously state the problem of planning in unknown environments. In Section III we develop a notion of optimality that can be used as a benchmark. In Section IV we leverage such a notion of optimality to derive a pseudo-optimal class of policies. In Section V we present results from numerical experiments supporting our statements. Finally, in Section VI we draw some conclusions and discuss directions for future work.

II. NOTATION

This paper concerns planning trajectories for robots with dynamics and perception through static configuration spaces with obstacles, and we will need carefully-defined notation for all these concepts. First the robot's configuration space: we assume it lies in $[0,1]^d$, and it is characterized by its obstacles $\mathcal{X}_{\text{obs}} \subset [0,1]^d$, or more generally the subset of $[0,1]^d$ which the robot is not allowed to enter. The remainder of the configuration space which the robot is free to traverse is denoted $\mathcal{X}_{\text{free}} = [0,1]^d \setminus \mathcal{X}_{\text{obs}}$ and we will assume $\mathcal{X}_{\text{free}}$ is a closed set. We will sometimes informally refer to the configuration space with obstacles defined equivalently by $\mathcal{X}_{\text{obs}}$ or $\mathcal{X}_{\text{free}}$ as the environment. In particular, this paper deals with the case of unknown environment, necessitating notation for the set of all possible $\mathcal{X}_{\text{free}}$: $\mathfrak{X}_{\text{free}} = \{\mathcal{X} \subset [0,1]^d : \mathcal{X} \text{ closed}\}$ and all possible $\mathcal{X}_{\text{obs}}$: $\mathfrak{X}_{\text{obs}} = \{\mathcal{X} \subset [0,1]^d : [0,1]^d \setminus \mathcal{X} \text{ closed}\}$. Define the robot's initial state as $x_0 \in \mathcal{X}_{\text{free}}$ and its goal

region as $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$, where the set $\mathcal{X}_{\text{goal}}$ will also be assumed closed. Next, we define the dynamics $\mathcal{D}$ as the set of differential equations governing the robot's motion. Let $\sigma : [0,T] \to [0,1]^d$ denote a finite-time trajectory of duration T, with $\Sigma_{\mathcal{D}}$ the set of all dynamically-feasible trajectories under $\mathcal{D}$. Let $\text{START}(\sigma) := \sigma(0)$ and $\text{END}(\sigma) := \sigma(T)$, and let $\text{RANGE}(\sigma)$ denote the curve in $[0,1]^d$ traced out by σ. Let $c : \Sigma_{\mathcal{D}} \to \mathbb{R}_{\geq 0}$ be an additive cost function on the set of finite-time dynamically-feasible trajectories. Another important component in this paper is the robot's perception mechanism/function $P : \mathfrak{X}_{\text{free}} \times [0,1]^d \to \mathfrak{X}_{\text{free}} \times \mathfrak{X}_{\text{obs}}$, which takes in the environment and robot state and P_1 (the first output of P) returns a (usually local) set $\mathcal{X}_{\text{free}}^P$ of collision-free states around the robot's state $x \in [0,1]^d$ and P_2 (the second output of P) returns the observed open (except on the boundaries of $[0,1]^d$) subset $\mathcal{X}_{\text{obs}}^P$ of $\mathcal{X}_{\text{obs}}$ (while many robot sensors will only be able to perceive obstacle boundaries, which are closed, we will assume such sensors instead return a very thin open set along the obstacle boundary).[1][2] Note that P_2 is not just redundantly returning $[0,1]^d \setminus \mathcal{X}_{\text{free}}^P$, as it makes the important distinction between obstacle boundary, which the robot must avoid, and unexplored frontier, which the robot should explore in order to reach its goal.
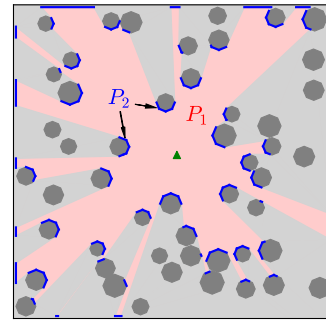

Fig. 2.   For a robot (green triangle) with line-of-sight perception, the observed obstacle-free space is given by P_1 in red while the observed obstacle subsets are given by P_2 in blue. Note that P_1 and P_2 are not completely redundant, as the boundary of P_1 excluding P_2 represents unexplored and ostensibly obstacle-free frontier.

The first argument to P is the configuration space, so the actual perception of the robot at a state x navigating an environment $\mathcal{X}_{\text{free}}$ will always be given by $P(\mathcal{X}_{\text{free}}, x)$, but importantly, the explicit dependence on the first argument will also allow us to reason about hypothetical future perception in a postulated environment $\mathcal{X}_{\text{free}}^{\text{guess}}$ which need not correspond to the true environment $\mathcal{X}_{\text{free}}$. It will also be useful to have a function $\text{ICS} : \mathfrak{X}_{\text{free}} \to \mathfrak{X}_{\text{free}}$, which takes a set of collision-free states $\mathcal{X}_{\text{free}}$ and returns the set of *inevitable collision states* (ICS) $\mathcal{X}_{\text{ICS}}$ defined as states x for which there exists $T_x < \infty$ such that a robot in state x and with dynamics $\mathcal{D}$ is guaranteed to leave $\mathcal{X}_{\text{free}}$ before time T_x (for notational simplicity we suppress the dependence on the dynamics $\mathcal{D}$). Throughout this paper we will assume no noise, that is, there is no stochasticity in the dynamics $\mathcal{D}$, the perception P, and the robot is assumed to always know its current state x. We will also assume that $\mathcal{X}_{\text{goal}} \cap \text{ICS}(\mathcal{X}_{\text{free}}) = \emptyset$, although this is not really necessary. Finally, let $[N] := \{1, \ldots, N\}$.

---

[1] $P(\mathcal{X}_{\text{free}}, x)$ may be defined arbitrarily for $x \notin \mathcal{X}_{\text{free}}$.
[2] P could also take as its second argument an entire trajectory—everything in this paper would still go through with only minor notational changes.

III. A NOTION OF OPTIMALITY FOR PLANNING IN UNKNOWN ENVIRONMENTS

A. Benchmarking Motion Planning in Unknown Environments

A significant algorithmic challenge and central contribution of this paper is how to define the objective for motion planning in unknown environments. As shorthand and to distinguish the settings of known and unknown environments, we will refer to motion planning in unknown environments as MP_{un} and motion planning in known environments as MP_{kn}. Pedagogically, we find it useful to compare with the known-environment setting, where the problem statement is clear:

Find a minimum-cost collision-free trajectory from x_0 to $\mathcal{X}_{goal}$.

Assuming a collision-free and dynamically-feasible trajectory exists, the minimum cost for the MP_{kn} problem with given $\mathcal{D}$, c, $\mathcal{X}_{free}$, x_0, and $\mathcal{X}_{goal}$ is given by (where dependence on $\mathcal{D}$ and c have been suppressed to streamline the notation)

$$c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal}) = \min_{\substack{\sigma \in \Sigma_{\mathcal{D}}, \\ \text{RANGE}(\sigma) \subset \mathcal{X}_{free}, \\ \text{START}(\sigma) = x_0, \ \text{END}(\sigma) \in \mathcal{X}_{goal}}} c(\sigma) \qquad (1)$$

Parsing the constraints: the first line enforces dynamic feasibility, the second line ensures the trajectory is collision-free, and the third line requires the trajectory to start at x_0 and end in $\mathcal{X}_{goal}$. As achieving $c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal})$ exactly is rarely computationally possible, actual MP_{kn} algorithms usually target something weaker, such as feasibility (e.g., [19]) or asymptotic optimality (e.g., [26, 23, 24, 20]), but it seems indisputable that $c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal})$ is the gold standard against which any MP_{kn} algorithm can be compared. Before getting to MP_{un}, let us rewrite (1) in a way that explicitly acknowledges that in practice most planning problems are dealt with in discrete time. As such, we can use $\mathcal{D}$ to define an action-generating function $\mathcal{A}_{\mathcal{D}} : [0,1]^d \to \Sigma_{\mathcal{D}}$ which gives the closed set of actions (dynamically-feasible finite-length trajectories) the robot may follow starting from any given state.[3] We can now write the discretized MP_{kn} problem, the analogue of (1), as

$$c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal}) = \min_{\substack{\sigma_1, \dots, \sigma_N \text{ s.t. } N \in \mathbb{N}, \\ \sigma_k \in \mathcal{A}_{\mathcal{D}}(\text{END}(\sigma_{k-1})) \ \forall k \in [N], \\ \text{RANGE}(\sigma_k) \subset \mathcal{X}_{free} \ \forall k \in [N], \\ \text{END}(\sigma_N) \in \mathcal{X}_{goal}}} \sum_{k=1}^{N} c(\sigma_k) \qquad (2)$$

As written, the minimization is over N as well as the σ_k, but in practice N will be fixed to some large value (an upper-bound for the number of actions in the optimal trajectory) and $\mathcal{A}_{\mathcal{D}}$ will be augmented to always include a null action which takes zero time, has zero cost, and does not change the robot's state. This allows the robot to take any number of null actions once it reaches $\mathcal{X}_{goal}$, so that c^* can accommodate optimal trajectories shorter than N actions. Parsing the constraints: the first line divides the trajectory into a finite number of actions σ_k, the second line requires each action to be dynamically feasible (for notational simplicity, define $\text{END}(\sigma_0) := x_0$), the third

---

[3]For instance, these actions could be traces through a prespecified short time or distance of solutions to the differential equations in $\mathcal{D}$ using parameters defined by a set of low-level controls such as steering angle or acceleration; we abstract this all away for notational convenience to deal only with actions.

---

line requires each action to be collision-free, and the final line requires the final action to end in $\mathcal{X}_{goal}$.

Despite the work in MP_{un} reviewed earlier, we are not aware of any analogue to $c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal})$ for MP_{un}, i.e., a gold standard against which to compare an algorithm's performance and measure its suboptimality. Equation (2) is also useful in that if one could find a minimizer, that trajectory would be optimal for discretized MP_{kn}. For these reasons we seek an MP_{un} analogue. First note that $c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal})$ will generally represent an unattainable objective for MP_{un}, and thus does not itself represent an ideal benchmark for MP_{un}.

Example 1. *Consider a robot with a limited sensing radius and bounded speed and acceleration (which can be applied in any direction, that is, unlike a car, the robot has no 'forward' direction) so that the stopping distance from its top speed is greater than its sensing radius. Then the only way to achieve $c^*(\mathcal{X}_{free}, x_0, \mathcal{X}_{goal})$ when $\mathcal{X}_{obs} = \emptyset$ is for the robot to immediately accelerate to maximum speed in the direction of the nearest point in $\mathcal{X}_{goal}$ and proceed until it reaches $\mathcal{X}_{goal}$. However, this trajectory clearly is not collision-free if there is an obstacle along that path, and in fact even if the robot chooses to follow this trajectory until an obstacle is sensed, it is guaranteed to collide if the obstacle is far enough away from x_0. We emphasize that this fact makes such a choice inherently unsafe, since even if there is prior information suggesting there are no obstacles, the robot cannot verify this assertion because of its limited sensing and so may still end up colliding with an unexpected obstacle.*

The preceding paragraph hints at two important points. First, although it was sufficient in MP_{kn} to talk only about trajectories, in MP_{un} new information actually comes in during a trajectory that the robot in general can react to, so it is more appropriate to discuss *online policies* instead, which are functions that take a priori information about the robot and environment and combine it with a history of perception information gathered by the robot up to a given time to produce the robot's next action. In this paper, we want to find policies which are *safe*.

Definition 1 (Safe Policy). *A safe policy is a function that takes in $\mathcal{A}_{\mathcal{D}}$, c, P, x_0, $\mathcal{X}_{goal}$, and a vector of historical perception information of arbitrary length $(P(\mathcal{X}_{free}, x_0), \dots, P(\mathcal{X}_{free}, x_k))$ and returns an action that is guaranteed not to leave $\mathcal{X}_{free}$, for any value of $\mathcal{X}_{free}$ that agrees with the historical perception information.*

Note that $\mathcal{X}_{free}$ only appears in the inputs to a policy through the perception function P, so the only information about the environment that the policy can use to ensure safety is gathered online by the robot. Also, policies may in general have a random component, in which case we could talk about the *probability* of the policy returning an action that collides with an obstacle, but in this paper we only deem a policy safe if that probability is zero. The goal of this paper is to establish a framework for computing safe policies that produce (collision-free, by definition) trajectories with as low a cost as possible.

Second, the only way to guarantee a policy is safe is to ensure that no action takes the robot into an inevitable collision state *with respect to any unobserved part of the environment*. This is a necessary condition for a policy to be

safe as defined in Definition 1, since if the robot ever enters an ICS with respect to an unobserved part of the environment, then the robot will (inevitably) collide for some possible values of the environment (e.g., for $\bigcup_{j=0}^{k} P_1(\mathcal{X}_{\text{free}}, x_j)$, which matches the observed environment perceived up until the ICS is reached, but for which $\mathcal{X}_{\text{obs}}$ contains all the unobserved environment), contradicting the definition. This is suggestive of the following discretized MP_{un} problem, analogous to the discretized MP_{kn} problem of Equation (2) (in addition to suppressing dependence on the dynamics $\mathcal{A}_{\mathcal{D}}$ and the cost function c, we also now suppress the dependence on P, since all three of these inputs are normally fixed for a given robot):

$$\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}}) = \min_{\sigma_1, \ldots, \sigma_N \text{ s.t. } N \in \mathbb{N},} \sum_{k=1}^{N} c(\sigma_k) \qquad (3)$$
$$\sigma_k \in \mathcal{A}_{\mathcal{D}}\big(\text{END}(\sigma_{k-1})\big) \;\; \forall k \in [N],$$
$$\text{RANGE}(\sigma_k) \cap \text{ICS}\big(\bigcup_{j=0}^{k-1} P_1(\mathcal{X}_{\text{free}}, x_j)\big) = \emptyset \;\; \forall k \in [N],$$
$$\text{END}(\sigma_N) \in \mathcal{X}_{\text{goal}}$$

where the second-to-last constraint line is the only part that changed from Equation (2) (we abuse notation slightly by letting ICS now and for the remainder of the paper denote its discrete analogue, defined as the set of states x for which there exists $T_x < \infty$ such that a robot in state x and following any actions in $\mathcal{A}_{\mathcal{D}}$ is guaranteed to leave $\mathcal{X}_{\text{free}}$ before time T_x). That line states that the kth action cannot be in an ICS with respect to everything that the robot will perceive up through the end of $(k-1)$th action σ_{k-1}. Note that all the states in $\mathcal{X}_{\text{obs}}$ are ICS, so this is at least as strong as just requiring all the actions to be collision-free, and thus $\tilde{c}^* \geq c^*$ when the two take the same arguments. Nevertheless, because any trajectory returned by a safe policy can never enter an ICS (and thus can never violate the second constraint line in Equation (3)), $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$ lower-bounds the cost of any trajectory returned by a safe policy applied in $\mathcal{X}_{\text{free}}$. The natural next question is whether Equation (3) is tight—that is, does there exist a safe policy which attains it?

The answer is a qualified "yes." As we show next, there is a safe policy that achieves the cost $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$, but that policy depends on $\mathcal{X}_{\text{free}}$. A reasonable criticism would be: if we allow the policy to depend on $\mathcal{X}_{\text{free}}$, why not have a policy that just returns a solution to the discretized MP_{kn} problem from Equation (2), which would attain the even lower cost c^*? The answer is that such a policy is not safe as we have defined it, in that it does not in general produce a collision-free trajectory, while the *safety* of the $\mathcal{X}_{\text{free}}$-dependent policy we describe next actually does not depend on $\mathcal{X}_{\text{free}}$. That is, the policy we are about to define produces a trajectory that is collision-free no matter what, even if $\mathcal{X}_{\text{free}}$ is not the true environment. As underscored in Example 1, this distinction is not only conceptually important, but will be practically important when we turn to developing algorithms for real problems, where the true $\mathcal{X}_{\text{free}}$ is never known exactly but may be guessed at based on prior knowledge or information gained along the trajectory, making it crucial that the safety of a policy does not depend on perfect knowledge of $\mathcal{X}_{\text{free}}$.

Consider a policy that starts with some guess $\mathcal{X}_{\text{free}}^{\text{guess}}$ for $\mathcal{X}_{\text{free}}$ and takes actions according to a minimizer of Equation (3)—except with $\mathcal{X}_{\text{free}}^{\text{guess}}$ replacing $\mathcal{X}_{\text{free}}$—until it reaches $\mathcal{X}_{\text{goal}}$ or it first perceives an obstacle boundary not in $\mathcal{X}_{\text{free}}^{\text{guess}}$ (here is

where we first see the utility of P_2, since only P_1 was used in Equation (3) but it does not distinguish between observed obstacles and the simple boundary of observed free space). In the latter case—denote such a waypoint by x_k—the robot updates $\mathcal{X}_{\text{free}}^{\text{guess}}$ to some $\mathcal{X}_{\text{free}}^{\text{guess}(2)}$ that matches its perception (e.g., by taking the union of the obstacles in $\mathcal{X}_{\text{free}}^{\text{guess}}$ and any obstacle boundary perceived thus far) and then follows a minimizer of Equation (3)—with $\mathcal{X}_{\text{free}}^{\text{guess}(2)}$ now replacing $\mathcal{X}_{\text{free}}$, with x_k as its starting point, and still with all perception information up to time k incorporated into the second constraint—until it reaches $\mathcal{X}_{\text{goal}}$ or it first perceives an obstacle boundary not in $\mathcal{X}_{\text{free}}^{\text{guess}(2)}$, and continues this loop indefinitely. This is a safe policy because by definition the trajectory it produces never enters a state that is ICS with respect to the unobserved part of the environment. Furthermore, in the lucky case when $\mathcal{X}_{\text{free}}^{\text{guess}} = \mathcal{X}_{\text{free}}$, an obstacle boundary not in $\mathcal{X}_{\text{free}}^{\text{guess}}$ will never be observed, and the trajectory will simply be a minimizer of Equation (3) and thus, will have cost exactly equal to $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$. The following theorem summarizes what we have shown in this section (this paragraph established the upper-bound and the lower-bound comes from the four sentences following Equation (3)).

Theorem 1. *For a robot constrained to take actions defined by $\mathcal{A}_{\mathcal{D}}$ in an environment $\mathcal{X}_{\text{free}}$ which is unknown to it except through a perception mechanism P, the optimal cost c of any trajectory from x_0 to $\mathcal{X}_{\text{goal}}$ returned by a safe policy is exactly $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$, defined by Equation (3).*

Remark 1. *We stop here to briefly contrast our benchmark with the usual approach of competitive analysis [5], which would benchmark an online algorithm to its best possible offline performance—in this paper that would mean using c^* as a benchmark instead of $\tilde{c}^*$. But since we have shown that c^* is in general unachievable in MP_{un}, increasing c^* to $\tilde{c}^*$ by imposing constraints that are necessary conditions for any safe MP_{un} trajectory provides a more realistic benchmark. Our approach is similar to that of* comparative *analysis [16], with the addition that Theorem 1 also proves that our benchmark is tight.*

B. A Flexible Class of Pseudo-Optimal Policies

The proof of Theorem 1's tightness in the previous subsection exhibited a class of safe policies that achieve cost exactly equal to $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$ when fed a guess $\mathcal{X}_{\text{free}}^{\text{guess}}$ of $\mathcal{X}_{\text{free}}$ that happens to be correct (and because they are safe policies, even if the guess is incorrect they are still guaranteed to never produce a colliding trajectory). This is certainly a desirable property for a safe MP_{un} policy, but perhaps more important from a practical standpoint is ensuring the cost does not depend too dearly on the accuracy of $\mathcal{X}_{\text{free}}^{\text{guess}}$. In simpler terms, we would like our safe policy to achieve *nearly* optimal cost when given a *nearly* correct guess of $\mathcal{X}_{\text{free}}$, and to still perform reasonably well even if no prior knowledge about about $\mathcal{X}_{\text{free}}$. In this subsection we will introduce a flexible class of safe policies that are all pseudo-optimal in that they achieve $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$ when fed a correct guess $\mathcal{X}_{\text{free}}^{\text{guess}} = \mathcal{X}_{\text{free}}$, but also allows for more clever methods of updating $\mathcal{X}_{\text{free}}^{\text{guess}}$ as new obstacles are observed online, so that the cost degrades gracefully as $\mathcal{X}_{\text{free}}^{\text{guess}}$ diverges from $\mathcal{X}_{\text{free}}$. We still ignore computational constraints in this subsection, but

address the approximations needed for computability in the following subsection.

As just alluded to, we first need notation to describe a $\mathcal{X}_{\text{free}}^{\text{guess}}$-updating scheme. Let $U : \mathfrak{X}_{\text{free}} \times \mathfrak{X}_{\text{obs}} \to \mathfrak{X}_{\text{free}}$ take as arguments the thus-far perceived collision-free space $\mathcal{X}_{\text{free}}^P$ and the thus-far perceived subset of $\mathcal{X}_{\text{obs}}$, $\mathcal{X}_{\text{obs}}^P$, and return a guess $\mathcal{X}_{\text{free}}^{\text{guess}}$ of what the true $\mathcal{X}_{\text{free}}$ might be. A minimal requirement for a reasonable guessing function U would be that its output guess at least agrees with its inputs of historical perception, i.e., $\mathcal{X}_{\text{free}}^P \subset U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P)$ and $\mathcal{X}_{\text{obs}}^P \subset [0,1]^d \setminus U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P)$. This leaves many reasonable choices for U, such as if there is an outdated map of the environment $\mathcal{X}_{\text{free}}^{\text{map}}$, then a reasonable choice might be $U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P) = (\mathcal{X}_{\text{free}}^{\text{map}} \cup \mathcal{X}_{\text{free}}^P) \setminus \mathcal{X}_{\text{obs}}^P$, which would produce highly accurate guesses if $\mathcal{X}_{\text{free}}^{\text{map}}$ is pretty close to $\mathcal{X}_{\text{free}}$, while still being able to adapt to unexpected obstacles. At the other end of the spectrum, if very little is known about the environment a priori, since we are only considering safe policies anyway, it may make sense to take an aggressive strategy and simply set $U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P) = [0,1]^d \setminus \mathcal{X}_{\text{obs}}^P$. This generic formalism allows for both extremes and everything in between.

With U in hand, the problem simplifies in the sense that there is a well-defined "best" policy which assumes its guess at the environment is correct, still constrains itself to never enter an ICS with respect to unknown parts of the environment, and is also forward-looking in that it takes into account the consequences of its current action on all future actions. From a given state x with $\mathcal{X}_{\text{free}}^P$ the union of P_1's thus far and $\mathcal{X}_{\text{obs}}^P$ the union of P_2's so far, define the optimal cost-to-go when using a given updating function U as

$$\tilde{c}_U^*\big(x, (\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \mathcal{X}_{\text{goal}}\big) = \min_{\sigma_1, \ldots, \sigma_N \text{ s.t. } N \in \mathbb{N},} \sum_{k=1}^{N} c(\sigma_k) \quad (4)$$

$$\sigma_k \in \mathcal{A}_{\mathcal{D}}\big(\text{END}(\sigma_{k-1})\big) \; \forall k \in [N],$$
$$\text{RANGE}(\sigma_k) \bigcap \text{ICS}\big(F^U_1(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_{k-1}))\big) = \emptyset \; \forall k \in [N],$$
$$\text{END}(\sigma_N) \in \mathcal{X}_{\text{goal}}$$

where again for notational simplicity define $\text{END}(\sigma_0) := x$, and the function $F^U : \mathfrak{X}_{\text{free}} \times \mathfrak{X}_{\text{obs}} \times \{(\sigma_1, \ldots, \sigma_k) : \sigma_j \in \Sigma_{\mathcal{D}} \; \forall j \in [k], \; k \in \mathbb{N}\} \to \mathfrak{X}_{\text{free}} \times \mathfrak{X}_{\text{obs}}$ takes in the current state of perception in $\mathcal{X}_{\text{free}}^P$ and $\mathcal{X}_{\text{obs}}^P$, as well as a sequence of future actions $\{\sigma_1, \ldots, \sigma_k\}$, and pushes forward the perception function P and the updating function U along the hypothetical trajectory $\sigma_1, \ldots, \sigma_k$, returning what the robot would have perceived by the end of σ_k assuming U produces a correct guess of the environment. Function F^U can be explicitly defined recursively as:

$$F^U\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \emptyset\big) = \big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P\big), \quad (5)$$

$$F^U_1\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_k)\big) =$$
$$F^U_1\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_{k-1})\big) \bigcup$$
$$P_1\Big(U\Big(F^U\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_{k-1})\big)\Big), \text{END}(\sigma_k)\Big),$$

$$F^U_2\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_k)\big) =$$
$$F^U_2\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_{k-1})\big) \bigcup$$
$$P_2\Big(U\Big(F^U\big(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_{k-1})\big)\Big), \text{END}(\sigma_k)\Big).$$

Using Bellman's equation and the recursive nature of F^U, $\tilde{c}_U^*$ can also be written recursively as

$$\tilde{c}_U^*\big(x, (\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \mathcal{X}_{\text{goal}}\big) = \quad (6)$$
$$\min_{\substack{\sigma \in \mathcal{A}_{\mathcal{D}}(x), \\ \text{RANGE}(\sigma) \bigcap \text{ICS}(\mathcal{X}_{\text{free}}^P) = \emptyset}} c(\sigma) + \tilde{c}_U^*\big(\text{END}(\sigma), F^U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \{\sigma\}), \mathcal{X}_{\text{goal}}\big)$$

and we can in turn explicitly write the optimal policy similarly

$$\sigma_U^*\big(x, (\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \mathcal{X}_{\text{goal}}\big) = \quad (7)$$
$$\arg\min_{\substack{\sigma \in \mathcal{A}_{\mathcal{D}}(x), \\ \text{RANGE}(\sigma) \bigcap \text{ICS}(\mathcal{X}_{\text{free}}^P) = \emptyset}} c(\sigma) + \tilde{c}_U^*\big(\text{END}(\sigma), F^U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \{\sigma\}), \mathcal{X}_{\text{goal}}\big).$$

(σ as an optimization variable here represents a single action, in contrast to Equation (1) where it is a full trajectory ending at $\mathcal{X}_{\text{goal}}$.) Note that the policy class σ_U^* has the pseudo-optimality property that as long as U is chosen such that it returns the true $\mathcal{X}_{\text{free}}$ until it is given contradicting perception information, the cost of following σ_U^* from x_0, namely, $\tilde{c}_U^*(x_0, P(\mathcal{X}_{\text{free}}, x_0), \mathcal{X}_{\text{goal}})$, will exactly equal the optimal MP_{un} cost proved in Theorem 1: $\tilde{c}^*(\mathcal{X}_{\text{free}}, x_0, \mathcal{X}_{\text{goal}})$. This can be seen by noting that if U always returns $\mathcal{X}_{\text{free}}$, then the second-to-last constraint of Equation (4) matches that of Equation (3), and everything else matches as well. The pseudo-optimality tells us that, despite no explicit reward for exploration in the formulation of σ_U^*, for good-enough U, σ_U^* trades off exploration and exploitation exactly optimally, and we may hope this implicit trade-off is still fairly good for more general U. However, we do not try to characterize the gap between $\tilde{c}_U^*$ and $\tilde{c}^*$ for general U, as such a theoretical investigation would be beyond the scope of this paper, and a simulation experiment would be computationally prohibitive due to the cost of computing $\tilde{c}^*$.

IV. AN APPROXIMATELY PSEUDO-OPTIMAL CLASS OF POLICIES

A. Generic Approximations

We now turn to the implementation and computational tractability of employing the policy σ_U^*. Despite the fairly compact recursive representation of $\tilde{c}_U^*$ in Equation (6), performing dynamic programming would be prohibitively expensive due to the exponential (in N) size of the search space. But luckily, there are a few approximations/simplifications we can make that drastically improve the computational properties of σ_U^*, and we have found in practice that our approximate σ_U^* still performs quite well (as discussed in Section V). We list all these approximations next, which together provide an *approximately* pseudo-optimal class of policies that are also computationally tractable. In the next subsection, we will give the implementation details for these approximations that we found to work well in our numerical experiments.

Approximation #1: First, ICS is outer-bounded by $\widetilde{\text{ICS}}$ which checks only a few stopping actions (e.g., maximum deceleration with no turning angle, maximal left turning angle, and maximal right turning angle). Note that because this approximation is a tight outer-bound, it does not change the guarantee that our policy never results in collision.

Approximation #2: One thing to note about Equation (5) is that if U has the reasonable property that its output $\mathcal{X}_{\text{free}}^{\text{guess}}$

does not change with new perception information as long as that information agrees with $\mathcal{X}_{\text{free}}^{\text{guess}}$, then F^U simplifies and can be written nonrecursively as

$$F^U_1\left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_k)\right) = \quad\quad (8)$$

$$\mathcal{X}_{\text{free}}^P \bigcup_{j=1}^{k} P_1\left(U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \text{END}(\sigma_j)\right)$$

$$F^U_2\left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, (\sigma_1, \ldots, \sigma_k)\right) =$$

$$\mathcal{X}_{\text{obs}}^P \bigcup_{j=1}^{k} P_2\left(U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \text{END}(\sigma_j)\right).$$

The reason is that no matter how far into the future the guess is being propagated, since the hypothetical future actions are all assumed to gather perception information from an environment that matches $U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P)$, and new perception information that matches the existing guess doesn't change the guess, then that guess is the same as just the guess with the current information. This property of U is not necessary, but both of the examples given when U was introduced satisfy it, and even for some U functions that do not, we expect that solving Equation (4) with the simplified constraint given by Equation (8) will give a good approximation to solving it with the full constraint for most reasonable choices of U. Thus we plug in the union in Equation (8) for F^U_1 in the second-to-last constraint line of Equation (4).

Approximation #3: Having just rewritten the second-to-last constraint line of Equation (4) using Equation (8), we now proceed to simplify it further by reducing the union in Equation (8) to only $\mathcal{X}_{\text{free}}^P$ and the last perception output, i.e., $\mathcal{X}_{\text{free}}^P \cup P_1\left(U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \text{END}(\sigma_k)\right)$ if the third argument to F^U_1 has at least one action, or simply $\mathcal{X}_{\text{free}}^P$ if it has none. That is, for checking ICS, we always keep track of all the *actual* perception information gathered by the robot, but only keep track of the most recent *hypothetical* perception information. We expect this approximation to be quite good in most cases, since the robot will usually be moving forward into unknown space, not backward into already-explored space. This approximation gives the optimization problem optimal substructure, which makes it far easier to solve.

Approximation #4: Until $\mathcal{X}_{\text{goal}}$ first intersects $\mathcal{X}_{\text{free}}^P$, we greedily set an intermediate goal region just outside $\mathcal{X}_{\text{free}}^P$ and only optimize our hypothetical trajectory to reach that instead of $\mathcal{X}_{\text{goal}}$. Said intermediate goal $\mathcal{X}_{\text{goal}}^{\text{inter}}\left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \mathcal{X}_{\text{goal}}\right)$ is chosen as the collision-free region of the boundary of the unknown environment that minimizes the sum of an approximate cost-to-come from x to $\mathcal{X}_{\text{goal}}^{\text{inter}}$ and an approximate cost-to-go from $\mathcal{X}_{\text{goal}}^{\text{inter}}$ to $\mathcal{X}_{\text{goal}}$ through the current guess of the environment. This approximation drastically reduces the search space, especially early in the trajectory, and for simple approximations adds little computational cost. The intuition behind this approximation is that the robot, without knowing what lies beyond the next obstacle, is unlikely to be able to make good high-level choices and must instead be greedy at low resolution. For instance, if a robot traversing a hallway encounters a fork, with limited information about the environment beyond its immediate vicinity, it might as well choose the hallway which points more towards $\mathcal{X}_{\text{goal}}$. This

approximation does nothing to hinder the main benefit of the forward-looking nature of σ_U^*, which is highly-intelligent *local* behavior, like how best to approach a blind corner.

Combining these approximations gives us an expression for an approximation to σ_U^* that is more computationally tractable:

$$\sigma_U^{\text{aprx}}\left(x, \left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P\right), \mathcal{X}_{\text{goal}}\right) = \quad\quad (9)$$

$$\underset{\substack{\sigma \in \mathcal{A}_{\mathcal{D}}(x), \\ \text{RANGE}(\sigma) \bigcap \text{ICS}(\mathcal{X}_{\text{free}}^P) = \emptyset}}{\arg\min} \; c(\sigma) + \tilde{c}_U^{\text{aprx}}\left(\text{END}(\sigma), \left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \sigma\right), \mathcal{X}_{\text{goal}}\right),$$

where the new approximate cost-to-go function $\tilde{c}_U^{\text{aprx}}$ is given by (note it has different inputs from $\tilde{c}_U^*$)

$$\tilde{c}_U^{\text{aprx}}\left(x, \left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \sigma_0\right), \mathcal{X}_{\text{goal}}\right) = \min_{\sigma_1, \ldots, \sigma_N} \sum_{k=1}^{N} c(\sigma_k). \quad (10)$$

$$\sigma_k \in \mathcal{A}_{\mathcal{D}}\left(\text{END}(\sigma_{k-1})\right) \; \forall k \in [N],$$

$$\text{RANGE}(\sigma_k) \bigcap \widetilde{\text{ICS}}\left(\mathcal{X}_{\text{free}}^P \cup P_1\left(U(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P), \text{END}(\sigma_k)\right)\right) = \emptyset \; \forall k \in [N],$$

$$\text{END}(\sigma_N) \in \mathcal{X}_{\text{goal}}^{\text{inter}}\left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \mathcal{X}_{\text{goal}}\right)$$

B. Implementation Details

The way we implemented σ_U^{aprx} made various choices about the approximations discussed in the previous subsection, which can be categorized into the choice of U, $\widetilde{\text{ICS}}$, $\mathcal{X}_{\text{goal}}^{\text{inter}}\left(\mathcal{X}_{\text{free}}^P, \mathcal{X}_{\text{obs}}^P, \mathcal{X}_{\text{goal}}\right)$, and how we actually computed a solution for Equation (9). These choices are purely heuristic, and were found through experience and experimentation to work well for a range of problems, but there may be many other good choices as well. In all of our experiments the cost function c was simply the elapsed time of the trajectory.

Choice of U: We chose U so that all unobserved space is guessed to be collision-free, except any obstacle boundaries that are already observed but end at unknown territory are extended by a small amount into the unknown. This choice of U is optimistic while assuming roughly that "walls are locally smooth", and we expect this to be a reasonable choice in almost any setting with little a-priori knowledge of the unknown environment. For observed obstacle boundaries that ended at unknown territory, five evenly-spaced points were marked up to 0.125m from the unknown boundary, and then those points were fit with either a linear or quadratic curve and extended into the unknown until intersection with a known region, up to a maximum length of 0.125m.

Choice of $\widetilde{\text{ICS}}$: Three stopping actions were used: maximal straight-line deceleration in current direction of travel, maximal deceleration while turning left, and maximal deceleration while turning right.

Choice of $\mathcal{X}_{\text{goal}}^{\text{inter}}$: Intermediate goals were selected by dividing the unknown frontier into lengths of 0.3m, and the location defined by the midpoint of each frontier segment is used to define the center of a circular candidate $\mathcal{X}_{\text{goal}}^{\text{inter}}$ of radius 0.3m. The cost-to-come for each candidate was computed by dynamically-constrained (but not ICS-constrained) FMT* [23, 24] and the cost-to-go computed by FMT* [14], both with sample density of 150 samples/m^2 and connection radius of 0.75m. Since FMT* returns a geometric path and not a kinodynamic trajectory, the FMT* solution path distance was converted to time by assuming that from the final speed when the robot reaches $\mathcal{X}_{\text{goal}}^{\text{inter}}$, the robot maximally accelerates to

and continues at maximum speed to $\mathcal{X}_{\text{goal}}$ along the FMT*path.

Choice of $\mathcal{X}_{goal}^{inter}$: Finally, the actual optimization problem was approximated again by dynamically-constrained FMT* but with the addition of the ICS constraint in the third constraint line of Equation (10). In fact, even that constraint is again simplified by replacing RANGE(σ_k) with END(σ_k)—this is purely a computational simplification, since if the endpoints are not in ICS, then no point along the action can possibly be in ICS. Because of this simplification and approximations #2 and #3 in the previous subsection, the ICS constraint can actually be checked pointwise, making it extremely amenable to sampling-based motion planning algorithms such as FMT*, since each sample can be checked for ICS immediately after being sampled and before any edges are drawn. Note that sampling-based algorithms take a slightly different approach to approximate the solution to Equation (9), but the sample density and edges between samples implicitly define the discreteness parameters N and the set of actions $\mathcal{A}_{\mathcal{D}}$.

V. NUMERICAL EXPERIMENTS

In this section we present simulation results on maze, forest, and office building environments to show the versatility of our algorithm in environments of varying complexity. We implemented two dynamics models for experiments. The first is a nonlinear vehicle model with maximum acceleration/braking of 1m/s^2, maximum speed of 9 m/s, maximum curvature of 0.13 m, and maximum curvature rate of 7.5 $(\text{ms})^{-1}$. The second is a 2D double integrator dynamics model with maximum acceleration/braking of 1 m/s^2 and maximum speed of 6 m/s. Experiments were written in the Julia programming language [3] and run on a Intel Core i5-4300U CPU@1.90GHz quadcore laptop, although the four cores were not used in parallel.

We first demonstrate the strength of our algorithm in a large-scale comparison with the method of [22]. We ran our algorithm on randomly-generated hallway (maze with no dead ends) maps with the nonlinear vehicle dynamics model and considered the resulting tradeoff between path cost and collision probability. Hallway maps of width 1.2 m were generated by sampling from a Markov chain hallway generator with turn frequency of 0.4. The path cost for connections considered is total time to goal, and the resulting total path cost on each map was normalized by the path cost of the path planned by the dynamically-constrained (but not ICS-constrained) FMT* algorithm with comparable sample density on the fully known map. Figure 3 shows the resulting comparison between our algorithm and that of [22]. By design, our algorithm is guaranteed not to collide and thus only has one data point denoting zero probability of collision. The data point for our algorithm lies below the curve defining the collision probability-path cost tradeoff of [22]'s algorithm, and for their approach to achieve the same path cost as σ_U^{aprx} would require roughly a 5% probability of collision. Additionally, as [22]'s method's probability of collision approaches zero, the cost begins to climb rapidly, far above that of σ_U^{aprx}. This outperformance can most readily be explained by the forward-looking nature of σ_U^{aprx}, as it optimizes with respect to future actions and perception, in contrast to the method of [22].
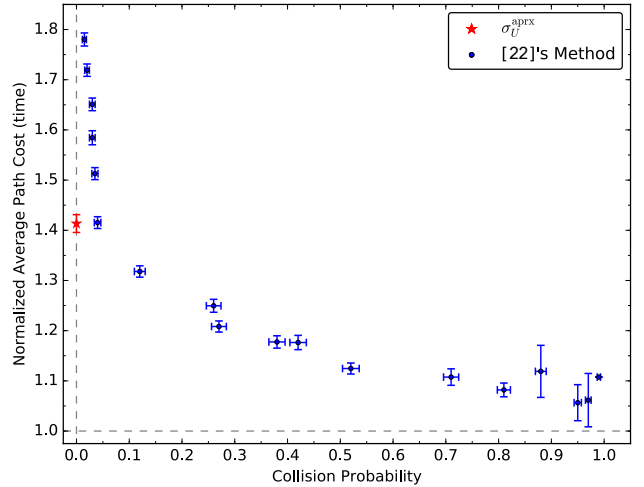

Fig. 3. Normalized average path cost versus collision probability of simulations from 500 random hallway maps using our algorithm (red). Path costs are normalized by the cost of the path planned by the kFMT* algorithm with comparable sample density and with full knowledge of the map, and then averaged over the trials. The data points for the [22] method (blue) were produced by running their algorithm on 1000 randomly generated hallway maps with different cost-of-collision values and averaging over the paths that did not result in collision. Their algorithm was trained on 50,000 sample hallway maps and features used were the same as the ones in the original paper, with appropriate hyperparameter tuning. Error bars denote ±1 standard deviation.

To understand better the behavior of our implementation of σ_U^{aprx}, Figure 4 displays the results of our algorithm on a maze environment with the nonlinear vehicle dynamics. We see that the vehicle takes wide turns when approaching corners, which allows for greater visibility of what is around the corner before actually making the turn and minimizes the deceleration necessary to maintain safety. The optimization *automatically and implicitly* trades off the extra distance required to take the turn more widely and the extra speed allowed by seeing the open hallway earlier and thus not having to slow down to ensure safety in case of a hidden obstacle. Figures 4(a) and 4(b) show the added benefit of this approach—early detection of bad trajectories. While a myopic approach may have optimistically rushed into the path suggested in Figure 4(a), by taking the turn more widely (which, again, was never explicitly programmed into our algorithm) the vehicle detects the dead end early and swings back up to the correct route, having suffered only a minor delay. Figures 5 and 6 show results of our algorithm with double integrator dynamics in a forest environment with scattered tree obstacles and a real office building environment with complex hallways, furniture, and rooms. In the forest environment, greedy exploration and plans to intermediate goals guide the robot to the goal on a direct and fast trajectory with minimal detours despite the lack of any prior knowledge of the environment (the U function used was the same as for the maze) and guaranteed safety. The office building simulation shows each of the safety and optimality features of our algorithm. Initial travel in the narrow hallway is slow and cautious, followed by higher speeds in the wider space, and finally wide turns when traveling back into the narrower hallway and into the room to arrive at the goal.

Table I shows the computation times for each environment
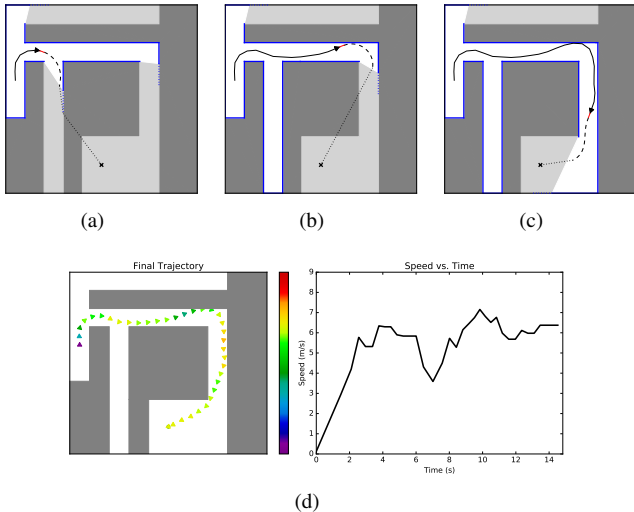
(a) (b) (c)

(d)

Fig. 4. Simulation results on a maze map with dead ends and forks in the path. The receding horizon plan to the goal is shown in the dashed (current to intermediate goal) and the dotted (intermediate goal to final goal) lines. The travelled path up to each waypoint is shown in black, with the action to be taken at the timestep in red. The full path is shown in (d), with speed indicated by the color bar. Note the recovery from exploration of a hallway that turned out to be a dead end shown in (a) and (b).
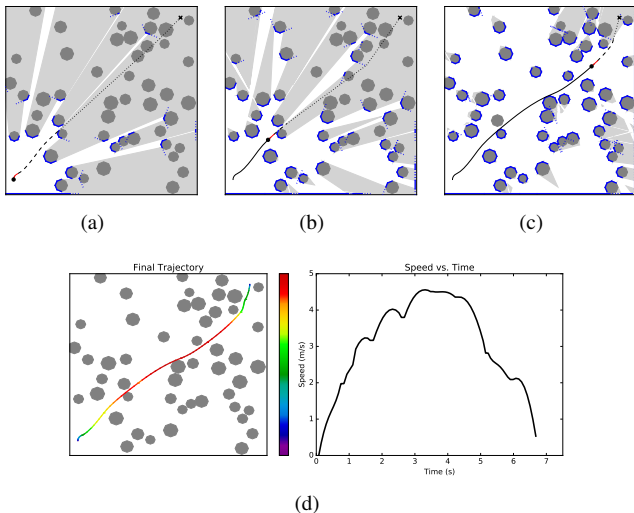

(a) (b) (c)

(d)

Fig. 5. Simulation results on a forest environment with tree obstacles. Formatting is the same as in Figure 4.

with their corresponding complexity defined by number of half-planes defining the obstacles in the map and the average number of half-planes defining the unexplored frontier at the beginning of each action. Each action requires about 0.5 - 1s of (serial) computation time, which makes the algorithm amenable to real-time implementation. Computation times could be significantly improved by considering a parallel implementation, which is left for future work.

VI. CONCLUSIONS AND FUTURE WORK

We have proposed a algorithmic framework and a pseudo-optimal class of policies for the problem of safe motion planning in unknown environments. This work easily raises as
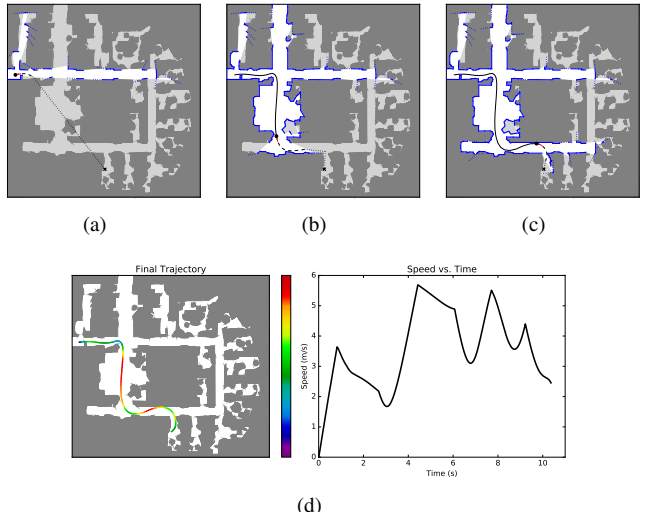

(a) (b) (c)

(d)

Fig. 6. Simulation results on simplified scan of the Stanford Gates building internals. Formatting is the same as in Figure 4.

TABLE I
COMPUTATION TIMES

Environment	Total # of obstacle half-planes	Avg. # of frontier half-planes	Avg. Time (ms)/plan
Maze	96	2	540
Forest	490	15	780
Office Bldg.	1493	25	1340

many interesting questions as it solves, and future work will include (1) improving computation times through parallelization and optimized implementation, (2) deployment on real robot and integration with true dynamics and perception functions, (3) theoretical bounds on the optimality gap of σ_U^* as a function of U, (4) characterization of performance when the problem is infeasible, (5) extension to the setting of uncertainty, both in the dynamics and perception of the robot, for instance by replacing 'inevitable' in the ICS acronym with 'highly-probable', and (6) investigation into the best implementation details for different types of problems, especially the crucial choices of $\widehat{ICS}$ and U. We highlight this last direction, the choice of U, as particularly appealing for its allowance of sophisticated machine learning algorithms to guess at the unexplored environment. While especially the most complex learning algorithms can have spectacular performance, they are often criticized for a lack of reliability and generalizability— but $\sigma_U^*/\sigma_U^{\text{aprx}}$ completely ameliorate this concern by acting as a wrapper around the machine learning in U that guarantees the robot's safety no matter what it returns. Underscoring this point: if U ever returns a grossly inaccurate guess for the environment due to one of the many reasons why machine learning algorithms can fail (e.g., test data which does not resemble the training data), the robot will (safely) begin to follow a less-efficient trajectory. This seems like as good an outcome as possible, since of course no algorithm can perform equally well if given good or bad information, but using the methods in this paper the worst we can lose is time, but never the robot itself due to collision.

REFERENCES

[1] Yamauchi B. A frontier-based approach for autonomous exploration. In *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, 1997.

[2] K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. IEEE Conf. on Robotics and Automation*, 2007.

[3] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing, 2012. Available at http://arxiv.org/abs/1209.5145.

[4] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "Next-Best-View" planner for 3D exploration. In *Proc. IEEE Conf. on Robotics and Automation*, 2016.

[5] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge Univ. Press, 2005.

[6] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[7] C. Connolly. The determination of next best views. In *Proc. IEEE Conf. on Robotics and Automation*, volume 2, pages 432–435, 1985.

[8] D. Ferguson, N. Kalra, and A. Stentz. Replanning with RRTs. In *Proc. IEEE Conf. on Robotics and Automation*, 2006.

[9] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. *SIAM Journal on Computing*, 38(3):881–898, 2008.

[10] T. Fraichard and H. Asama. Inevitable collision states – a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[11] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[12] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys. Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *Proc. IEEE Conf. on Robotics and Automation*, pages 1071–1078, 2015.

[13] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. Journal of Robotics Research*, 21(3):233–255, 2002.

[14] L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast Marching Tree: a fast marching sampling-based method for optimal motion planning in many dimensions. *Int. Journal of Robotics Research*, 34(7):883–921, 2015.

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

[16] E. Koutsoupias and C. H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):300–317, 2000.

[17] S. M. LaValle. *Planning Algorithms*. Cambridge Univ. Press, 2006.

[18] S. M. LaValle. Motion planning: Wild frontiers. *IEEE Robotics and Automation Magazine*, 18(2):108–118, 2011.

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

[20] Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *Int. Journal of Robotics Research*, 35(5):528564, 2016.

[21] S. Petti and T. Fraichard. Partial motion planning framework for reactive planning within dynamic environments. In *Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics*, 2005.

[22] C. Richter, W. Vega-Brown, and N. Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Int. Journal of Robotics Research*, pages 325–341. 2018.

[23] E. Schmerling, L. Janson, and M. Pavone. Optimal sampling-based motion planning under differential constraints: the driftless case. In *Proc. IEEE Conf. on Robotics and Automation*, 2015. Extended version available at http://arxiv.org/abs/1403.2483/.

[24] E. Schmerling, L. Janson, and M. Pavone. Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics. In *Proc. IEEE Conf. on Decision and Control*, 2015.

[25] J. P. van den Berg and M. H. Overmars. Planning the shortest safe path amidst unpredictably moving obstacles. In *Workshop on Algorithmic Foundations of Robotics*, pages 103–118, 2006.

[26] D. J. Webb and J. van den Berg. Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints. In *Proc. IEEE Conf. on Robotics and Automation*, 2013.