Asymptotically Optimal Planning by Feasible Kinodynamic Planning in a State–Cost Space

Kris Hauser, *Member, IEEE*, and Yilun Zhou

***Abstract*—This paper presents an equivalence between feasible kinodynamic planning and optimal kinodynamic planning, in that any optimal planning problem can be transformed into a series of feasible planning problems in a state–cost space, whose solutions approach the optimum. This transformation yields a meta-algorithm that produces an asymptotically optimal planner, given any feasible kinodynamic planner as a subroutine. The meta-algorithm is proven to be asymptotically optimal and a formula is derived relating expected running time and solution suboptimality. It is directly applicable to a wide range of optimal planning problems because it does not resort to the use of steering functions or numerical boundary-value problem solvers. On a set of benchmark problems, it is demonstrated to perform, using the expansive space tree (EST) and rapidly-exploring random tree (RRT) algorithms as subroutines, at a level that is superior or comparable to related planners.**

***Index Terms*—Motion planning, nonlinear control systems, trajectory optimization.**

I. Introduction

OPTIMAL motion planning is a highly active research topic in robotics, due to the pervasive need to compute paths that simultaneously avoid complex obstacles, satisfy dynamic constraints, and are high quality according to some cost functions. Recent advances in sampling-based optimal motion planning build on decades of work in the topic of feasible motion planning, in which costs are ignored. However, the field has not yet achieved general-purpose optimal planning algorithms that accept arbitrary black-box constraints and costs as input. In particular, achieving optimality under kinematic and differential constraints remains a major challenge for sampling-based planners.

This paper presents a new state–cost space formulation that transforms optimal motion planning problems into feasible kinodynamic (both kinematically and differentially constrained) planning problems. Using this formulation, we introduce a meta-algorithm, AO-x, to adapt any feasible kinodynamic planner x into an asymptotically optimal motion planner, provided that x satisfies some relatively unrestrictive conditions, e.g., the expected running time is finite. The meta-algorithm accepts arbitrary cost functions, including nondifferentiable ones, and handles whatever kinematic and differential constraints are accepted by the underlying feasible planner.

Manuscript received June 30, 2015; revised January 20, 2016; accepted July 17, 2016. This paper was recommended for publication by Associate Editor T. Bretl and Editor A. Billard upon evaluation of the reviewers' comments. This work was partially supported by the National Science Foundation under Award 1218534. The work of Y. Zhou was partially supported by a Pratt Engineering Undergraduate Research Fellowship.

The authors are with the Department of Electrical and Computer Engineering and with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (e-mail: kris.hauser@duke.edu; yilun.zhou@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2016.2602363

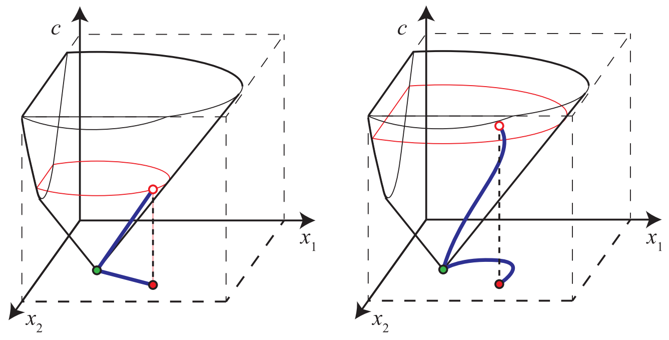

Fig. 1. State–cost space for a 2-D kinematically constrained problem with a path length cost function. State–cost space is 3-D, with a conical reachable set with an apex at the start configuration (green circle). Two paths to the same state–space target (red filled circles) follow trajectories that arrive at different points in state–cost space (red open circles).

The formulation is rather straightforward: the n-dimensional state variable is augmented with an auxiliary cost variable that measures the cost-to-come (i.e., accumulated cost from the start state). This yields an $(n+1)$-dimensional dynamically constrained feasible problem in (state, cost) space (see Fig. 1). The meta-algorithm proceeds by generating a series of feasible trajectories in the state–cost space with progressively lower costs. This is accomplished by first generating a feasible trajectory in state space and then progressively shrinking an upper bound on cost according to the cost of the best path found so far. This meta-algorithm is proven to converge toward an optimal cost under relatively unrestrictive conditions.

The AO-x meta-algorithm is demonstrated on practical examples using the rapidly-exploring random tree (RRT) [16] and expansive space tree (EST) [9] algorithms as subroutines for feasible kinodynamic planning. Due to prior theoretical work on the running time of EST, we are able to prove that the expected running time of the meta-algorithm is $O(\epsilon^{-2} \ln \ln \epsilon^{-1})$, where ϵ is the solution suboptimality. Critically, this is one of the few asymptotically optimal planners that exclusively uses control sampling to handle dynamic constraints, rather than resorting to a steering function or a numerical two-point boundary value problem solver. The new method outperforms prior planners in several toy scenarios including both dynamic constraints and complex cost functions.

II. Background and Related Work

Optimal motion planning has been a topic of renewed activity in robotics largely due to the advent of sampling-based motion planners that are proven to be asymptotically optimal [12]. However, the community has had a long history of interest in optimal motions. Numerical trajectory optimization techniques [2]–[4], [23] have been studied for quite a while but have several drawbacks, including susceptibility to local minima and the need for differentiable constraint and cost representations, which are often hard to produce for complex obstacles. Grid-based planners [7], [18], [24] are often fast in low-dimensional spaces but suffer from the "curse of dimensionality," with performance degrading rapidly in spaces of higher dimension [21]. Sampling-based planners were originally developed to overcome many of these challenges and have been shown to have excellent empirical performance in finding feasible paths in high-dimensional spaces, both without [14] and with dynamic constraints [9], [16]. However, they tend to produce jerky paths that are far from optimal. Some hybrid approaches have combined sampling-based planning with local optimization to produce better paths [20], [26].

More recently, sampling-based optimal planners like RRT* have been proven to produce asymptotically optimal paths whose costs converge in expectation toward optimality. The insight is that paths approaching optimality can be obtained by judiciously "rewiring" a tree of states to add connections that reduce the cost to a given node in the tree. However, this requires a steering function, which is a method to produce a curve between two states that is optimal when obstacles are ignored. In systems without dynamic constraints, this is as simple as generating a straight line. However, steering functions for dynamically constrained systems are much harder to come by.

Several authors have extended RRT* to dynamically constrained systems. It is relatively easy to apply RRT* to dynamically constrained systems if a steering function is available [11]. Proving convergence is more difficult and requires analysis of small-time controllability conditions [13]. Other authors have extended RRT* to systems whose dynamics and costs are (or can be approximated) by linear and quadratic functions, respectively, by definition of a suitable steering function based on the linear quadratic regulator (LQR) principle [22], [27]. It is likely that these approaches will be more effective than our work, which is potentially the most practical in the context in the presence of more complex differential constraints for which a steering function cannot be derived.

In this context, several methods for approximate steering have been developed. One method generated complex maneuvers using RRT* and performed each rewiring step by numerically solving a two-point boundary value problem [10]. This adds a significant computational expense. A similar method performed rewiring using a spline-based trajectory representation that is optimized via a nonlinear program solver [25]. Because they may fall into local minima, such solvers are not guaranteed to find a rewiring solution even if one exists. Convergence of these RRT* modifications is also not proven.

The closest related work to ours in terms of generality of applicability is the Stable-Sparse-RRT planner [19]. Like our work, it avoids the use of a steering function entirely and samples directly in control space. Approximate rewiring is performed by allowing connections to points that are "near enough" according to a distance metric. This scheme was proven to satisfy asymptotic near-optimality, which is the property of converging toward a path with bounded suboptimality [5]. In a more recent paper, the same authors have extended it to an asymptotically optimal planner, stable sparse tree (SST*), by progressively shrinking nearness threshold parameters [17]. However, Stable-Sparse-RRT and SST* and have many parameters to tune, and our experiments suggest that AO-x planners in general outperform both planners.

We note the similarity of our algorithm to Anytime-RRT [6], which includes cost-to-come in the distance metric, except that ours plans in state–cost space rather than simply state space and has fewer parameters to tune. Hence, Anytime-RRT does not obey any theoretical asymptotic- or near-optimality guarantees, and in fact, our experiments suggest that it tends not to converge to an optimum.

III. Theoretical Formulation

This section presents the state–cost space formulation, the meta-algorithms, and theoretical results regarding asymptotic optimality.

A. Terminology

First, we define key concepts of feasible, optimal, and boundedly suboptimal planning problems, as well as complete, probabilistically complete, and asymptotically optimal planners. Let X denote the state space.

Definition 1: A feasible (kinodynamic) planning problem $\mathcal{P} = (X, U, x_I, G, F, B, D)$ asks to produce a trajectory $y(s) : [0, S] \to X$ and control $u(s) : [0, S] \to U$ such that

$$y(0) = x_I \quad \text{(initial state)} \quad (1)$$

$$y(1) \in G \subseteq X \quad \text{(goal state)} \quad (2)$$

$$y(s) \in F \subseteq X \quad \forall s \in [0, S] \quad \text{(kinematic constraints)} \quad (3)$$

$$u(s) \in B(y(s)) \quad \forall s \in [0, S] \quad \text{(control constraints)} \quad (4)$$

$$y'(s) = D(y(s), u(s)) \,\forall s \in [0, S] \quad \text{(dynamic equation).} \quad (5)$$

This is a highly general formulation. Note that kinematic planning problems can simply set the control variable $u(s)$ to the derivative of the path, the control set to $B = \{u \mid \|u\| \leq 1\}$, and the dynamic equation as $D(y, u) = u$. Second-order planning problems (i.e., those with inertia) can be defined with a configuration × velocity state $x = (q, \dot{q})$. Problems with time-variant constraints can be constructed in this same form by augmenting the state variable with the time variable (x, t).

Definition 2: An optimal planning problem $\mathcal{P} = (L, \Phi, X, U, x_I, G, F, B, D)$ asks to produce a trajectory $y(s) : [0, T] \to X$ that minimizes the objective functional

$$C(y) = \int_0^T L(y(s), u(s))ds + \Phi(y(T)) \quad (6)$$

among all feasible trajectories [those that satisfy (1)–(5)]. Here, T is the terminal time, L is the incremental cost, and Φ is the terminal cost function.

For technical reasons, it is more appropriate to speak of an optimal cost rather than an optimal path, because for some problems, an optimal cost may exist as an infimum of costs of trajectories approaching some limit, while a limiting trajectory itself may not exist or may be infeasible [15].

Definition 3: A bounded-suboptimality planning problem $(\mathcal{P}, \epsilon)$ asks to find a trajectory satisfying $C(y) = C^* + \epsilon$, where C^* is the optimal cost and $\epsilon > 0$ is a specified parameter.

Definition 4: A complete planner $\mathcal{A}$ finds a feasible solution to a problem $\mathcal{P}$ when one exists and terminates with "failure" if one does not. Moreover, it does so in finite time. A probabilistically complete planner finds a feasible solution to a problem $\mathcal{P}$, when one exists, with probability approaching 1 as more time is spent planning. Examples include the RRT and EST planners (see Appendix A). A planner is asymptotically optimal for the optimal planning problem $\mathcal{P}$ if the cost $C(t)$ of the generated path approaches the optimum C^* with probability 1 as more time t is spent planning.

B. State–Cost Space Equivalence

Our first contribution is to demonstrate an equivalence of any optimal planning problem with a canonical state–cost form, in which dependence on the incremental cost L is eliminated. In particular, we augment each state x with the cost c taken to reach it from x_I to derive an expanded state $z = (x, c)$.

Theorem 1: The optimal planning problem $\mathcal{P} = (L, \Phi, X, U, x_I, G, F, B, D)$ is equivalent to a state–cost optimal planning problem without incremental costs

$$\hat{\mathcal{P}} = (0, \hat{\Phi}, X \times R^+, U, (x_I, 0), G \times R^+, F \times R^+, B, \hat{D})$$

such that solutions to $\hat{\mathcal{P}}$ are in one-to-one correspondence with solutions to $\mathcal{P}$. Here, the terminal cost $\hat{\Phi}$ is given by

$$\hat{\Phi}(x, c) = c + \Phi(x) \tag{7}$$

and the dynamics $\hat{D}$ are given by

$$z' = \begin{bmatrix} x' \\ c' \end{bmatrix} = \begin{bmatrix} D(x, u) \\ L(x, u) \end{bmatrix}. \tag{8}$$

The proof is straightforward, showing that the projection of solutions to $\hat{\mathcal{P}}$ onto the first $\dim(X)$ elements are solutions to $\mathcal{P}$, and solutions to $\mathcal{P}$ can be mapped to solutions to $\hat{\mathcal{P}}$ via augmenting them with a cumulative cost dimension. Note that even if every state in the original space is reachable, not all points in the state–cost space are reachable. Moreover, the goal set is now a cylinder with infinite extent in the cost direction [see Fig. 2(a)].

Using this state–cost transformation, we can show that a boundedly suboptimal problem with cost at most $\bar{c}$ can be solved by solving a feasible planning problem [see Fig. 2(b)].

Corollary 1: A bounded-suboptimality planning problem $\mathcal{P}$ with $\epsilon = \bar{c} - C^*$ is equivalent to a feasible planning problem $\mathcal{P}_{\bar{c}} = (X \times R^+, U, (x_I, 0), G_{\bar{c}}, F \times R^+, B, \hat{D})$, where $G_{\bar{c}} = \{(x, c) \mid x \in G, c \in [0, \bar{c} - \Phi(x)]\}$ is the set of terminal state–cost pairs satisfying the goal condition and the cost bound, and $\hat{D}$ is given by the state–cost transformation.

Specifically, if y is a solution to $\mathcal{P}_{\bar{c}}$, then it corresponds to a feasible solution of $\mathcal{P}$ with cost no more than $\bar{c}$ (and no less than C^*). In addition, $\mathcal{P}_{\bar{c}}$ has no solution if and only if $\bar{c} < C^*$.

C. Bounded-Suboptimality Metaplanning With a Complete Feasible Planner

The above corollary suggests that bounded-suboptimality planning is equivalent to feasible kinodynamic planning; however, C^* is *a priori* unknown. Hence, we present a bounded-suboptimality metaplanner that repeatedly invokes a feasible planner while lowering an upper bound on cost. This idea builds some intuition for the asymptotically optimal planner presented in the following section.

The meta-algorithm Bounded-Suboptimal($\mathcal{P}, \epsilon, \mathcal{A}$) accepts as input a problem $\mathcal{P}$, a tolerance ϵ, and a complete feasible planning algorithm $\mathcal{A}$ and is listed as follows:

```
Algorithm 1: Bounded-Suboptimal($\mathcal{P}, \epsilon, \mathcal{A}$).
1: Run $\mathcal{A}(\mathcal{P}_\infty)$ to obtain a first path $y_0$. If no solution
   exists, then report '$\mathcal{P}$ has no solution'.
2: Let $c_0 = C(y_0)$.
3: for $i = 1, 2, \ldots$ do
4:     Run $\mathcal{A}(\mathcal{P}_{c_{i-1}-\epsilon})$ to obtain a new solution $y_i$. If no
       solution to $\mathcal{P}_{c_{i-1}-\epsilon}$ exists, then stop.
5:     Let $c_i = C(y_i)$.
```

Step 1 solves for a feasible solution to the original problem, with no limit on cost. In practice, it may be solved in the original state space simply by discarding the cost function. In the loop, Step 4 establishes a new cost upper bound by lowering the best cost found so far c_{i-1} by ϵ. The following theorem proves correctness of this meta-algorithm.

Theorem 2: If $\mathcal{A}$ is a complete planner for the feasible kinodynamic planning problem, then Bounded-Suboptimal($\mathcal{A}, \epsilon$) terminates in finite time and produces a path y_i with cost no more than $C^* + \epsilon$.

Proof: Let $i + 1$ be the index of the final iteration. In the prior iteration, a solution was found, so $c_i \geq C^*$. In the current iteration, no solution is found, and since $\mathcal{A}$ is complete, the current planning problem, $\mathcal{P}_{c_i-\epsilon}$ is infeasible. So, $c_i - \epsilon < C^*$, and therefore, $c_i = C(y_i)$ is within ϵ of optimal.

Running time is finite since each loop reduces the cost by at least ϵ, and hence, it is run no more than $\lceil \frac{c_0 - C^*}{\epsilon} \rceil$ times. ■

D. Asymptotically Optimal Metaplanning With a Randomized Feasible Planner

The need for completeness is too restrictive for high-dimensional problems, where only probabilistically complete planners are practical. We now relax this restriction and also eliminate the dependence on the parameter ϵ. Now, rather than explicitly lowering the upper cost threshold upon each increment, the planner $\mathcal{A}$ searches using the cost of the current best path $\bar{c}$ as an upper cost bound, and hence, if it returns, the next path will have cost no more than $\bar{c}$. Due to probabilistic completeness, the cost of the returned path is almost surely

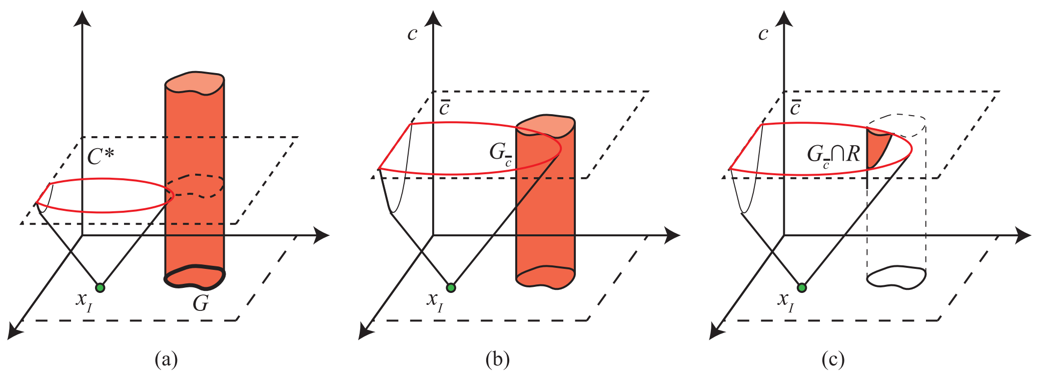

Fig. 2. (a) Goal region in state–cost space extends the state–space goal region G infinitely along the cost axis. The optimal cost C^* is the least-cost point in the reachable set that touches the region. (b) Given the cost of an existing path $\bar{c}$, the optimum is known to lie in the region of space with cost lower than $\bar{c}$. Finding a trajectory that improves upon $\bar{c}$ is a feasible planning problem. (c) Reachable portion of this goal set shrinks as $\bar{c}$ approaches the optimum.

strictly lower than $\bar{c}$. Algorithm 2 gives pseudocode for this meta-algorithm, AO-x.

Algorithm 2: Asymptotically-optimal($\mathcal{P}, \mathcal{A}, n$).

```
1: Run $\mathcal{A}(\mathcal{P}_\infty)$ to obtain a first path $y_0$. If no solution
   exists, report '$\mathcal{P}$ has no solution.'
2: Let $c_0 = C(y_0)$.
3: for $i = 1, 2, \ldots, n$ do
4:     Run $\mathcal{A}(\mathcal{P}_{c_{i-1}})$ to obtain a new solution $y_i$.
5:     Let $c_i = C(y_i)$.
6: return $y_n$
```

This procedure converges to an optimum under a fairly unrestrictive assumption that $\mathcal{A}$ is "well behaved" in the sense that the expected suboptimality value drops by a nonnegligible fraction each iteration. Specifically, we require that $\mathcal{A}$ has a significant chance of finding a path that shrinks the best cost found so far, regardless of $\bar{c}$. This assumption is stated as follows: given some cost upper bound $\bar{c}$, the cost of the next produced solution will be a random variable that follows a cumulative density function $\varphi(C(y); \bar{c})$, which ranges from 0 to 1 on the support $[C^*, \bar{c}]$, i.e., $P(C(y) \leq z) = \varphi(z; \bar{c})$. Our assumption does not prescribe the form of this distribution, but it does require that its moment be well behaved.

$\mathcal{A}$ is considered "well behaved" under two conditions.

1) If there exists a feasible solution and $\bar{c} > C^*$, then $\mathcal{A}$ terminates in finite time.
2) Given a cost bound $\bar{c}$, the expected suboptimality of the computed path is shrunk toward $C^\star$ by a nonnegligible amount each iteration.

Specifically, condition 2 requires that

$$E[C(y)|\bar{c}] - C^* \leq (1 - w)(\bar{c} - C^*) \tag{9}$$

for some positive constant $w > 0$. The value of w does not need to be known; it simply needs to exists. We note that condition 2 is similar to the more straightforward condition that $E[C(y)|\bar{c}] - C^* < (\bar{c} - C^*)$, but the bound w is needed for technical reasons. (Without it, there may exist decreasing sequences of $C(y)$ that never converge to C^*.) In other words, this condition states that there is a nonzero chance $\mathcal{A}$ does not produce the worst-possible path. This condition is not overly restrictive for most randomized planners; the set of paths with $C(y) = \bar{c}$ is a set with measure zero in the space of paths and is unlikely to be sampled at random. We note that it is not possible to explicitly use a contraction bound (9) as an upper cost limit for $\mathcal{A}$ because C^* is unknown.

We are now ready to state the main theorem.

Theorem 3: If $\mathcal{A}$ is a well-behaved randomized algorithm, then asymptotically optimal($\mathcal{A}, n$) is asymptotically optimal. In other words, as n approaches infinity, the probability that y_n is suboptimal approaches zero.

Proof: Let $S_0, \ldots, S_n$ be the nonnegative random variables denoting the suboptimality $C(y_i) - C^*$ during a run of the algorithm. We will show that they converge almost surely to the optimum as n increases. That is we want to show that $P(\lim_{n\to\infty} S_n = 0) = 1$.

Almost sure (a.s.) convergence is equivalent to $\lim_{n\to\infty} P(\sup_{m\geq n} S_m > \epsilon) = 0$. Since $\sup_{m\geq n} S_m = S_n$, a.s. convergence is implied by convergence in probability $\lim_{n\to\infty} P(S_n > \epsilon) = 0$. To prove convergence in probability, we will prove $\lim_{n\to\infty} E[S_n] = 0$ and then use the Markov inequality $P(S_n \geq \epsilon) \leq E[S_n]/\epsilon$.

Conditioning on S_{n-1}, we obtain

$$E[S_n] = \int E[S_n \mid s_{n-1}] P(s_{n-1}) ds_{n-1}$$

and due to (9), we have $E[S_n \mid s_{n-1}] \leq (1 - w)s_{n-1}$. Hence

$$E[S_n] \leq (1 - w) \int s_{n-1} P(x_{n-1}) ds_{n-1}$$
$$= (1 - w)E[S_{n-1}] = (1 - w)^n E[S_0] \tag{10}$$

and thus

$$P(S_n \geq \epsilon) \leq E[S_0](1 - w)^n / \epsilon. \tag{11}$$

Clearly, this approaches 0 as n increases. ■

E. Convergence Rate With Respect to Time

We now take a more detailed analysis of the case in which the feasible planner is probabilistically complete and study the convergence of asymptotically optimal in terms of running time t rather than the number n of planner calls. We show again, under relatively weak assumptions, that asymptotically optimal is asymptotically optimal in terms of time, even though each call to the planner takes increasingly longer to complete as n increases because the reachable portion of the goal set shrinks [see Fig. 2(c)].

A planner is probabilistically complete if the probability that it finds a feasible path, if one exists, approaches 1 as more time is spent planning. Note that a probabilistically complete planner will not necessarily terminate if no feasible path exists.

Note that probabilistic completeness is not a sufficient condition for a planner to be useful, since the convergence rate may be so slow that it is impractical. As an example, let $\mathcal{A}$ be a probabilistically complete planner, and $f(t)$ denote $P(\mathcal{A}$ fails after exactly t seconds of planning$)$. If $f(t) = 1/t$, then expected running time is infinite.

We will assume that for the given X, x_I, F, and $\hat{D}$, the planner $\mathcal{A}$ satisfies an exponential convergence bound, in which $f(t) \leq \min(1, ae^{-bt})$ for some positive values a and b. In practice, an exponential convergence bound implies expected running time is finite

$$E[t] \leq \int_0^\infty tf(t)dt \leq \frac{a}{b}.$$

A more refined analysis [8] gives a tighter bound

$$\int_0^\infty tf(t)dt = \int_0^{(\ln a)/b} tdt + \int_{(\ln a)/b}^\infty tae^{-bt}dt$$
$$= \frac{\ln a}{b} + \frac{1}{1-e^{-b}}. \quad (12)$$

Convergence rate varies, however, depending on the reachable portion of the goal region $G_{\bar{c}} \cap R(x_I)$, where $R(x)$ is the reachability set of x in state–cost space [see Fig. 2(c)]. In particular, $\mathcal{A}$ is less likely to sample a configuration in a small goal region at random, which slows convergence. Hence, the convergence rates are properly defined as a function of the volume of the goal region as

$$a \equiv a(\mu(G_{\bar{c}} \cap R(x_I))), b \equiv b(\mu(G_{\bar{c}} \cap R(x_I))). \quad (13)$$

The following theorem gives an example of such a bound when the EST algorithm is used as the underlying feasible planner.

Theorem 4: In an *expansive* space, the Kinodynamic EST planner [9] satisfies an exponential convergence bound with constants $a(g) = \gamma \ln \frac{1}{g}$ and $b(g) = \delta g$ for positive constants γ and δ, where g is the volume of the reachable goal region. Moreover, $E[t]$ is $O(\frac{1}{g}\ln\ln\frac{1}{g})$ as g approaches 0.

Proof: From the work of Hsu *et al.* [9], Kinodynamic EST with a uniform sampling strategy fails to find a path with probability no more than p if at least $\frac{k}{\alpha}\ln\frac{2k}{p} + \frac{2}{g}\ln\frac{2}{p}$ milestones are sampled, where $k = \frac{1}{\beta}\ln\frac{2}{g}$ and α and β are expansiveness constants that are fixed for the given configuration space. Using a bit of algebra, this expression can be rewritten as a bounded probability of failure

$$f(t) \leq (2k)^{\frac{kg}{kg+2\alpha}} \cdot 2^{\frac{2\alpha}{kg+2\alpha}} \cdot e^{\frac{-t\alpha g}{kg+2\alpha}}.$$

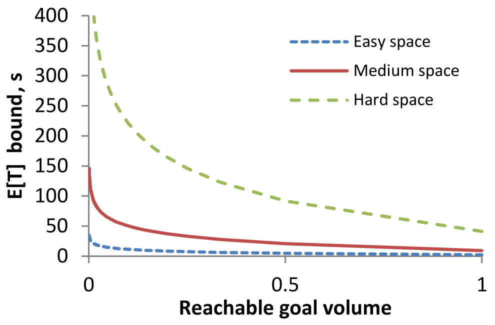

Fig. 3. Running time for each invocation of the underlying planner increases asymptotically to infinity as the reachable goal volume decreases. Visibility characteristics also have major effects on running time. Plots illustrate the theoretical bounds on expected running time of EST for spaces of three different "difficulty" levels. Specifically, visibility constants α and β are set to 0.04, 0.02, and 0.01 for easy, medium, and hard spaces. It is assumed that 1000 samples are generated per second.

Here, we have assumed that each sample takes constant time and the constant factor is ignored.

Now, we will simplify this rather unwieldy expression. First, note that the exponents of the first two terms in the equation are upper bounded by 1, since they are ratios of two positive numbers to their sums, and hence

$$f(t) \leq 4k \cdot e^{\frac{-t\alpha g}{kg+2\alpha}}.$$

Next, we use the fact that $\ln x \leq x$. Hence, $k = \frac{1}{\beta}\ln\frac{2}{g} \leq \frac{2}{\beta g}$. The factor $\frac{\alpha g}{kg+2\alpha}$ in the exponent can now be lower bounded by $\frac{\alpha\beta}{2+2\alpha\beta}g$, and hence, we have the desired expression

$$f(t) \leq \gamma \ln\frac{1}{g}e^{-t\delta g} \quad (14)$$

with $\gamma = \frac{8}{\beta}$ and $\delta = \frac{\alpha\beta}{2+2\alpha\beta}$ constant for a given configuration space. As a result, the running time is bounded by $E[t] \leq \frac{1}{\delta g}(\ln\gamma + \ln\ln\frac{1}{g}) + \frac{1}{1-e^{-\delta g}}$. As g shrinks, the latter term's order of convergence is $\frac{1}{g}$; therefore, we can conclude that $E[t]$ is $O(\frac{1}{g}\ln\ln\frac{1}{g})$ as desired. ■

Fig. 3 illustrates this bound. It is apparent that, since G is fixed by the original problem, $G_{\bar{c}}$ varies only with the parameter $\bar{c}$. Hence, we may also state these functions as $a(\bar{c})$ and $b(\bar{c})$. In order for EST to be "well behaved" as defined above, we must require that as $\bar{c}$ approaches C^*, the volume of $G_{\bar{c}} \cap R(x_I)$ is nonzero as long as $\bar{c} > C^*$.

There is also a remaining question of whether state–cost space is expansive in the sense of [9].

Lemma 1: If state space is expansive, then state–cost space is also expansive.

The proof is given in Appendix B. It should be noted, however, that the constants governing EST's convergence rate will be different in the state–cost space and may indeed be less favorable than in state space alone.

Let us now state our main result regarding AO-$\mathcal{A}$, which is the planner defined as Asymptotically Optimal$(\mathcal{P}, \mathcal{A}, \infty)$.

Theorem 5: If $\mathcal{A}$ is a probabilistically complete, exponentially convergent planner, then AO-$\mathcal{A}$ is asymptotically optimal in total running time t.

Proof: Define $c(t)$ as the cost of the best path found so far in AO-$\mathcal{A}$ after time t has elapsed. We will show that it converges almost surely toward C^* as t increases. Let $Z(t)$ be the random variable denoting the suboptimality $c(t) - C^*$ during a run of Asymptotically-optimal($\mathcal{A}, \infty$). We wish to prove that $\lim_{t\to\infty} P(Z(t) \geq \epsilon) = 0$ for any $\epsilon > 0$.

Let $T(x)$ be the random variable denoting the time at which the cost of the best path found so far decreases below $x + C^*$. It is evident that $P\left(Z(t) \geq \epsilon\right) = P\left(T(\epsilon) \geq t\right)$. We wish to show that $E\left[T(\epsilon)\right] < \infty$, which would in turn imply the claim due to the Markov inequality $P\left(T(\epsilon) \geq t\right) \leq E\left[T(\epsilon)\right]/t$.

If the current cost bound is greater than ϵ, then the expected time to find a path for any iteration is upper-bounded by the expected time it would take to find a path to $G_{C^*+\epsilon}$, which is some finite value t_ϵ since $\mathcal{A}$ is exponentially convergent. Specifically, $t_\epsilon = \frac{\epsilon}{\delta\epsilon^2}$ for kinodynamic EST as shown above. Therefore, if asymptotically optimal first finds a path with suboptimality no more than ϵ on the ith iteration, then the cost expended is no more than it_ϵ.

If we let N denote the random variable of the iteration on which asymptotically optimal first finds a path with suboptimality no more than ϵ, then we can bound $E\left[T(\epsilon)\right]$ as

$$E\left[T(\epsilon)\right] \leq \sum_{i=0}^{\infty} it_\epsilon P\left(I = i\right) = t_\epsilon \sum_{i=0}^{\infty} iP\left(N = i\right) = t_\epsilon E[N].$$

To show that $E[N]$ is finite, we will take a variant of the proof of Theorem 3. Again, let $S_0, \ldots, S_n$ be the nonnegative random variables denoting the suboptimality $C\left(y_i\right) - C^*$ during a run of the algorithm. N is the index of the first S_i that decreases below ϵ. Hence, $P(N \leq i) = P(S_i \leq \epsilon)$, and therefore

$$P(N \leq i) = 1 - P(S_i \geq \epsilon) \geq 1 - E[S_0](1-w)^i/\epsilon \quad (15)$$

where we have applied (11) derived in Theorem 3, and where $(1 - w) < 1$ is defined as before. Let $q(i) = P(S_i \geq \epsilon)$ and $r(i) = E[S_0](1-w)^i/\epsilon$. If we consider

$$\begin{aligned} E[N] &= \sum_{i=1}^{\infty} i(P(N \leq i) - P(N \leq i-1)) \\ &= \sum_{i=1}^{\infty} i(q(i-1) - q(i)) \\ &= \sum_{i=1}^{\infty} (q(i-1) + (i-1)q(i-1) - iq(i))) \\ &= \sum_{i=0}^{\infty} q(i) + \sum_{i=1}^{\infty} iq(i) - \sum_{i=1}^{\infty} iq(i) = \sum_{i=0}^{\infty} q(i) \end{aligned} \quad (16)$$

since $q(i) \leq r(i)$, we have $E[N] \leq \sum_{i=0}^{\infty} r(i) = E[S_0]/(\epsilon w)$, which is finite. ■

To be specific, if we were to use the exponential convergence bound for kinodynamic EST, we may conclude that $E[T(\epsilon)]$ is $O(\frac{1}{\epsilon^2}\ln\ln\frac{1}{\epsilon})$.

We note that although earlier iterations will likely terminate much faster than t_ϵ, the last iteration usually dominates running time since t_ϵ grows rapidly as $\epsilon \to 0$.

F. Complexity Discussion

We have proven that convergence rates of AO-x are chiefly governed by the performance of feasible planning subroutine x in (state, cost) space. However, one might wonder whether x might perform worse in (state, cost) space than in state space. Cost indeed adds an extra dimension, and it also adds drift to problems that may originally be driftless.

To address this potential concern, we note first that the first iteration of x can be run on the original state–space problem; therefore, the time to first solution is unchanged. Second, the control space remains unchanged. Because the performance of many kinodynamic planners are governed chiefly by control complexity rather than state dimensionality, the performance hit is not so severe.

Another question is how do problem characteristics, such as state–space dimensionality, geometric complexity, drift, and "visibility" characteristics affect the performance of AO-x? The answer has two parts. First, they affect the running time of x by changing the value of t_ϵ in the proof above. As a result, performance is strongly affected by the choice of planner x and its parameter choices. We observe empirically that choices of x that perform well on the feasible state–space version of a problem also tend to yield good performance of AO-x.

Second, problem dimensionality potentially affects the value of the expected cost reduction w in the proof above. For example, if the reachable goal region in (state, cost) space is locally shaped at the optimum like a convex cone of dimension d, then a goal configuration sampled at random will achieve an average cost reduction of $O(1/(d+1))$.

IV. Implementations and Experiments

This section describes the application of AO-x to several example problems using the feasible kinodynamic planners EST and RRT. We will refer to the implementations as AO-EST and AO-RRT. All planners are implemented in the Python programming language and, hence, could be sped up greatly by the use of a compiled language.

A. Implementations Using RRT and EST

Both kinodynamic EST and kinodynamic RRT are tree-growing planners that perform random extensions to a state–space tree, rooted at the start, by sampling a node in the tree and a control at random and, then, integrating the dynamics forward over a short time horizon. Their major difference is in the sampling strategy, which is described in more detail in Appendix A but will be summarized here. EST attempts to sample an extension so that its terminal state is uniformly distributed over the reachable set of the current tree. RRT attempts to sample an extension so that it is pulled toward a randomly sampled state in state space (a Voronoi bias). Both methods can also incorporate goal biasing strategies to avoid excessive exploration of the state space in directions that are not conducive to reaching the goal.

1) EST Implementation: EST can be applied directly to state–cost planning. EST samples extensions with probability proportional to the inverse density of existing states in the tree, as an approximation to uniform sampling distribution over the tree's reachable set. We use the standard method to approximate density by defining a grid of resolution h and low dimension k over randomly chosen orthogonal projections of the state–cost space. The density of a state x is estimated as proportional to the number of nodes in the tree $N(x)$ contained in the same grid cell as x. In a manner similar to locality-sensitive hashing, we choose several grids and count the total number of nodes sharing the same cell as x across all grids. For our experiments, we use $\binom{\dim(X)+1}{k}$ grids, $h = 0.1$, and $k = 3$, and scale the configuration space X to the range $[0,1]^{\dim(X)+1}$ before performing the random projection. To extend the tree, we sample ten candidate extensions by choosing ten source states uniformly from the set of occupied grid cells and drawing one random control sample. Among those extensions that are feasible, we select one with probability proportional to $1/(N(x_t)+1)^2$, where x_t is its terminal state.

2) RRT Implementation: RRT can also be applied almost directly, but there are some issues to be resolved regarding the definition of a suitable distance metric. RRT relies on a distance metric to guide the exploration toward previously unexplored regions of state space and is rather sensitive to the choice of this metric, with better performance as the metric approximates the true cost-to-go. We simply augment the state–space distance by adding a cost distance metric. Nearest node selection is accelerated using a KD-tree data structure.

3) Performance Considerations: Rather than planning from scratch each iteration, we maintain RRT and EST trees between iterations, which leads to some time savings. We also save time by pruning the portion of the tree with cost more than $\bar{c}$ whenever a new path to the goal is found. Specifically, smaller trees make EST density updates and RRT nearest neighbor queries computationally cheaper, although RRT shows a greater benefit because a larger fraction of its running time is spent in nearest neighbor queries. We also prune more aggressively if a heuristic function $h(x)$ is available. If $h(x)$ underestimates the cost-to-go, then we can prune all nodes such that $c + h(x) > \bar{c}$. Other sampling heuristics could also be employed to bias the search toward low-cost paths [1].

B. Example Problems

1) Kink: The Kink problem [see Fig. 4(a)] is a kinematically constrained problem in a unit square $[0,1]^2$, in which the optimal solution must pass through a narrow corridor of width 0.02 with two kinks. The objective is to minimize path length. Most planners very easily find a suboptimal homotopy class, but it takes longer to discover the optimal one. The maximum length of each expansion of the tree is limited to 0.15 units.

2) Bugtrap: The Bugtrap problem [see Fig. 4(b)] is a kinematically constrained problem in a unit square $[0,1]^2$ that asks the robot to escape a local minimum. The objective function is path length. This is a challenging problem for RRT planners due to their reliance on the distance metric as a proxy of cost. The maximum length of each expansion of the tree is limited to 0.15 units.

3) Dubins: This problem asks to move a standard Dubins car sideways while keeping orientation relatively fixed (see Fig. 5). The state is (x, y, θ) and the control is (v, ϕ), where θ is the

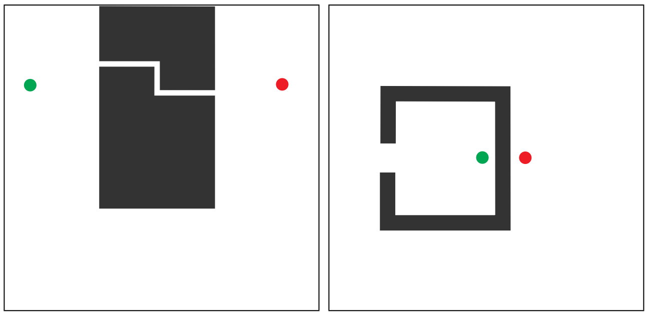
Fig. 4. Kink and Bugtrap problems ask to find the shortest path between the two indicated configurations.

heading, v is the forward velocity, and ϕ is the steering angle. State constraints include $(x,y) \in [0,1]^2$, $v \in \{-1,+1\}$, and $\phi \in [-\pi,\pi]$. For planning, time steps are drawn at random from $[0, 0.25]$ s. The metric is $d((x,y,\theta),(x',y',\theta')) = \sqrt{(x-x')^2 + (y-y')^2 + d_\theta(\theta,\theta')^2/(2\pi)}$, where d_θ measures the absolute angular difference. The goal is to move the car sideways 0.4 units with a tolerance of 0.1 units in state space, with minimal execution time (equivalent to minimum path length).

4) Double Integrator: This asks to move a point with bounded velocities and accelerations to a target location. The state space includes $x = (q, v)$ includes configuration q and velocity v, with constraints $q \in [0,1]^2$, $v \in [-1,-1]^2$, and $u \in [-5,5]^2$, with $\dot{q} = v$ and $\dot{v} = u$. The start is 0.06 units from the left and the goal is 0.06 units from the right, which must be reached with a tolerance of 0.2 units in state space. Distance is Euclidean distance. Time steps are drawn from the range $[0\text{s}, 0.05\,s]$.

5) Pendulum: The pendulum swing-up problem places a point mass of m =1 kg at the end of an L =1 m massless rod. The state space is $x = (\theta, \omega)$. The goal is the set of states such that the rod is within 10° of inverted and absolute angular velocity less than 0.5 rad/s, and the cost is the total time required to complete the task. We take gravitational acceleration to be $g = 9.8$ N·s^2, and a motor can exert a torque at the fixed end of the rod with bang-bang magnitudes $\tau \in \{-2, 0, 2\}$ N·m. The dynamics of the system are described by

$$\dot{\theta} = \omega \tag{17}$$

$$\dot{\omega} = \frac{\tau - mg\sin(\theta)L}{mL^2} = -9.8\sin(\theta) + \tau. \tag{18}$$

The difficulty in this task arises from the fact that the exerted torque cannot make the pendulum complete a full rotation. In fact, the torque will be canceled by gravity at about 11.5°. Therefore, the only way to achieve an inverted position is to take the advantage of gravity by swinging back and forth and accumulating angular momentum. For planning, constant torques are applied for a uniformly chosen duration between 0 and 0.5 s, and trajectories are numerically integrated using a time step of 0.01 s. Fig. 6 shows the first five paths obtained by AO-RRT.

6) Flappy: We devised a simplified version of the once-popular game Flappy Bird. The "bird" has a constant horizontal

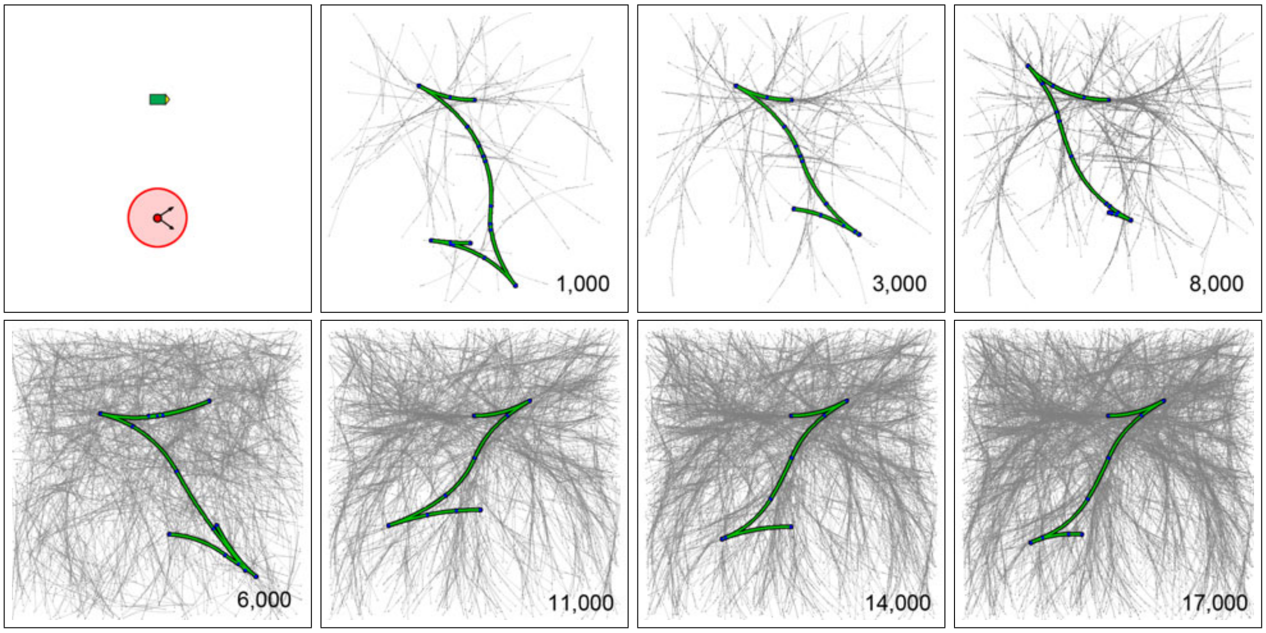

Fig. 5. Planning a sideways maneuver for a Dubins car using AO-RRT (top row) and AO-EST (bottom row). Numbers indicate total number of planning iterations. Green curve indicates best path found so far. Iteration counts are not directly comparable because RRT spends more time per iteration.

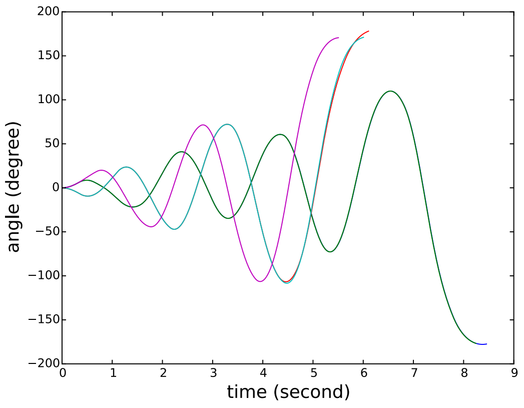

Fig. 6. Plot of angle versus time for the first five trajectories found by AO-RRT on the pendulum example. Shorter execution times (rightmost point on each curve) are preferred. The execution time decreases from 8.46 s in the first solution to 5.51 s in the fifth solution. (Best viewed in color)

velocity and can choose to fall freely under gravity, or apply a sharp upward thrust. The trajectory is a piecewise-parabolic curve. In the original game, the objective is simply to avoid obstacles as long as possible, but in our case, we consider other cost functions. The goal is to traverse from the left of the screen to a goal region on the right. The screen domain is 1000×600 pixels with fixed horizontal velocity of $v_x = 5$ px/s. The gravitational acceleration is $g = 1$ px/s^2 downward. The control u is binary and provides an upward thrust of either 0 or 4 px/s^2. Fig. 7 shows results obtained by our planner.

We represent the state by

$$\vec{x} = (x, y, v_y)$$

where (x, y) is the bird position, and v_y is the vertical velocity. The state evolves according to $\dot{x} = 5$, $\dot{y} = v_y$, and $\dot{v_y} = -1 + 4u$, where $u \in \{0, 1\}$ is the binary control. Time steps are sampled uniformly from the range $[0, 1]$, and the time evolution of the state is solved for analytically. The experiments below illustrate the ability of AO-x to accept unusual cost functions.

C. Experiments

1) Comparing AO-EST and AO-RRT: Fig. 5 illustrates AO-EST and AO-RRT applied to the Car example. Qualitatively, RRTs tend to explore more widely at the beginning of planning, while ESTs tend to focus more densely on regions already explored. As a result, in this example, AO-RRT finds a first path quicker, while AO-EST converges more quickly to the optimum (each iteration of EST is cheaper). Like in feasible planning, the best planner is largely problem dependent, and we could find no consistent winner.

2) Adaptation to Different Costs: Here, we illustrate the generality of the AO method across different cost functions, even those that are nondifferentiable. The Flappy problem and AO-RRT are used here. First, we set cost equal to path length. The second cost metric penalizes the distance traveled only in the lower half of the screen. The optimal path prefers high altitudes and passes through the two upper openings and one lower opening. Fig. 7 shows the results.

3) Benchmarking Against Comparable Planners: We compare against the simpler metaplanner M-x, which simply runs the feasible planner x multiple times, keeping the lowest cost path found so far. We also experimented with a variant, M-x-Prune, which prunes search nodes whose cost is greater

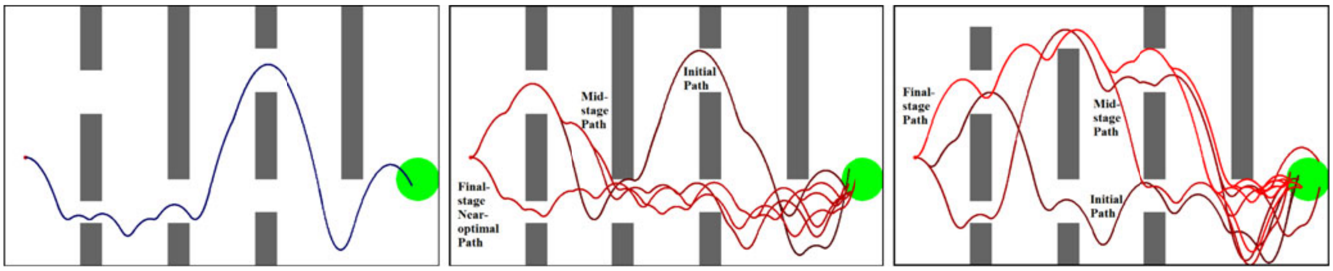

Fig. 7. (Left) Example solution path for Flappy. The planner finds a path that goes from the starting point on the left to the green goal region on the right while avoiding obstacles represented by gray rectangles. (Top) Convergence of Flappy with path length as cost. Brighter paths are of lower cost. The first path is high cost (1866 px), passing through both upper openings, and eventually converges to a path that passes both lower openings (1234 px). (Bottom) Penalizing low altitude paths. The first path passes through most of the lower openings (cost 1330), and the planner converges to a path that passes through upper openings (cost 321).

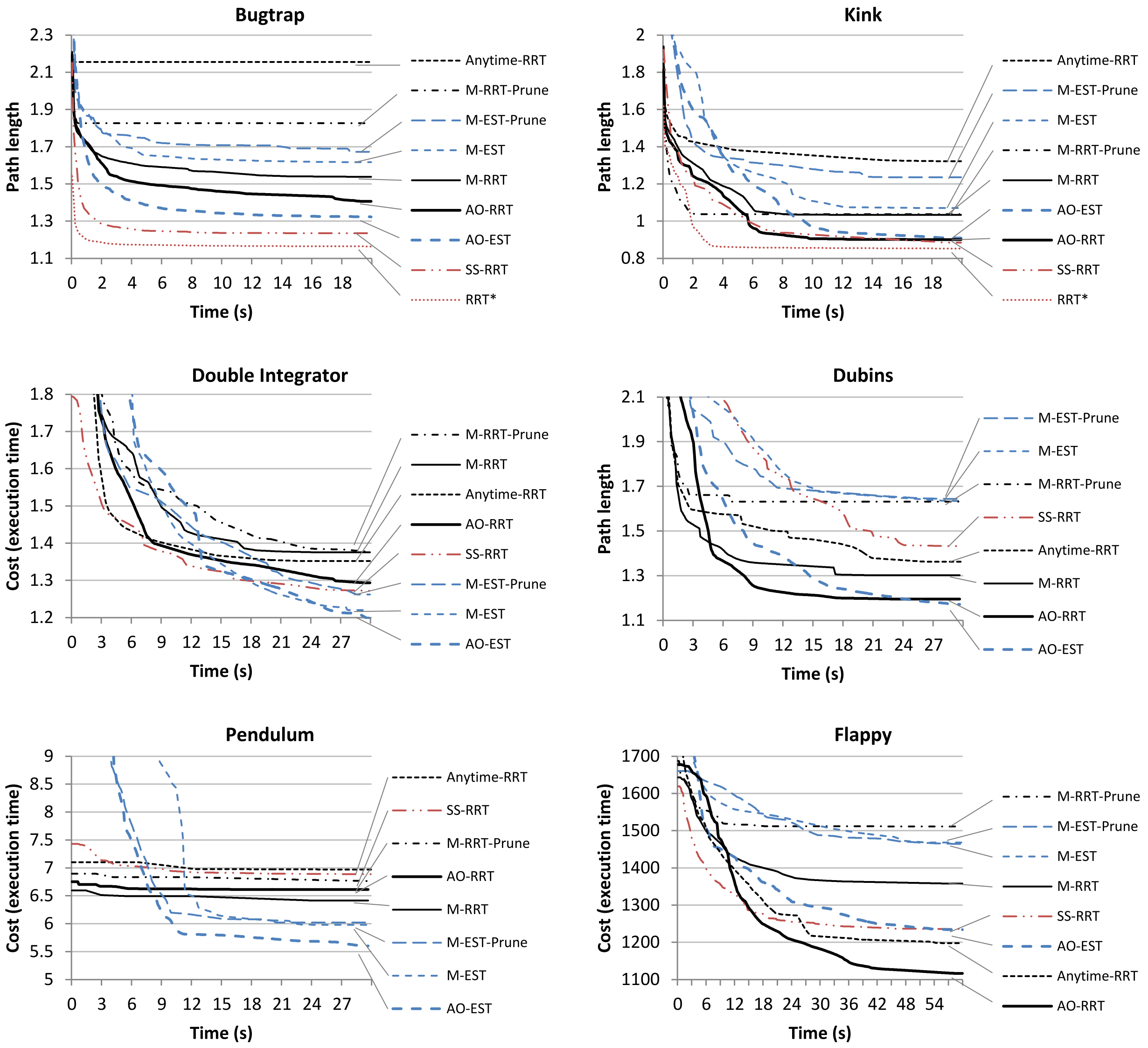

Fig. 8. Results of benchmark tests. Curves measure solution cost versus computation time, averaged over ten runs. (Lower values are better.)

than the cost of the best path found so far. In the 2-D problems, we compare against RRT* [11], and for fair comparison, we provide the other RRT-based planners with the straight line a steering function as well. We also compare against Anytime-RRT [6] and Stable-Sparse-RRT (SS-RRT) [17]. We also compared SST* [17], but its performance was comparable or worse than SS-RRT in all of our tests.

For fair comparison, all algorithms were implemented in Python using the same subroutines for feasibility checking, visibility checking, and distance metrics. All planners used the same parameters as AO-x where applicable. For Anytime-RRT, we used $\epsilon = 0.01$ and $\delta_c = 0.1$ and found that performance was relatively insensitive to these parameters. For SS-RRT, we provided the authors of [17] our test cases and source code, and let them choose parameters via tuning on each of our test cases. KD-trees were used for closest node selection in all of the RRT-based algorithms except Anytime-RRT, in which brute-force selection must be used because it does not select nodes using a true distance metric.

Fig. 8 displays computation time versus solution cost, averaged over ten runs for all of the benchmark problems. These results suggest that AO-EST consistently outperforms M-EST and M-EST-Prune, while AO-RRT usually outperforms M-RRT and M-RRT-Prune. The exception was Pendulum, where all the RRT-based algorithms converged extremely slowly. We find that Anytime-RRT typically does not perform even as well as the simpler M-RRT algorithm, although Anytime-RRT did perform well on Flappy. SS-RRT performed well on the low dimensional Bugtrap and Kink problems, but was generally middle-of-the-road on other problems.

Overall, we observe that AO-EST and AO-RRT are the best or near-best performers in most kinodynamic problems. However, AO-RRT is less consistent. A possible explanation is the well-known metric sensitivity of RRTs: when the distance metric becomes a poor approximation to cost-to-go, then RRT performance deteriorates. This property is inherited by AO-RRT and is most strongly observed in the Pendulum problem.

Visually, it may appear that negligible progress is being made in AO-x for some problems. Although suboptimality decreases significantly each time inner iteration of EST or RRT completes, the amount of time between completions grows substantially as suboptimality shrinks. Hence, the convergence rate is sublinear in time (as predicted by Theorem 5) and hence decreases very slowly.

V. Conclusion

This paper presents an equivalence between optimal motion planning problems and feasible kinodynamic motion planning problems using a state–cost space transformation. This simple but powerful transformation leads to an easily implemented, asymptotically optimal, sampling-based metaplanner that accepts a sampling-based kinodynamic feasible planner as input. It purely uses control-based sampling, making it suitable for problems with general differential constraints and cost functions that do not admit a steering function. The expected convergence rate of the metaplanner is proven to be related to the goal-dependent running time of the underlying feasible planner. Using RRT and EST as feasible planning subroutines, we demonstrate that the proposed method attains state-of-the-art performance on a number of benchmarks. An implementation of the algorithm and benchmarks are available at http://motion.pratt.duke.edu/dynamics/statecost.html.

We hope this new formulation will provide inspiration and theoretical justification for new approaches to optimal motion planning. As an example, an obvious way to improve convergence rate would be to run local optimizations on each trajectory found by the underlying planner; this method has been shown to work well for kinematic optimal path planning [20]. We also observed curious results regarding state–space versus cost–space weighting in the RRT distance metric. Following up on this work may also open up avenues of research in sampling strategies for state–cost space planning.

Appendix A
Kinodynamic RRT and EST Planners

For completeness, this appendix describes the RRT and EST planners as applied to feasible kinodynamic planning problems. We will use the terminology of a planning problem $\mathcal{P} = (X, U, x_I, G, F, B, D)$ in Definition 1. Both RRT and EST expand a tree $\mathcal{T} = (V, E)$ rooted at x_I consisting of states connected by feasible trajectories in F. They incrementally expand $\mathcal{T}$ by integrating the dynamics forward using a constant control input, but they differ in their method for selecting nodes and control inputs.

RRT's strategy expands $\mathcal{T}$ toward sparsely sampled regions by sampling a random state in X and extending the tree toward it. This is known as a Voronoi biasing strategy because the probability of choosing a node in $\mathcal{T}$ is proportional to the volume of its Voronoi cell. RRT takes the following parameters:

1) a state–space distance metric d;
2) a maximum integration duration $\delta > 0$;
3) a number of samples k drawn.

Pseudocode for the RRT algorithm is as follows:

Algorithm 3: RRT($\mathcal{P}, d, \delta, k$).

```
1: $V \leftarrow \{x_I\}, E \leftarrow \{\}$
2: **for** $i = 1, 2, \ldots$ **do**
3:     $x_{\text{rand}} \leftarrow$Sample($X$)
4:     $x_{\text{near}} \leftarrow \arg\min_{x \in V} d(x_{\text{rand}}, x)$
5:     $(y, \Delta t) \leftarrow$Select-Control$(x_{\text{near}}, x_{\text{rand}}, d, \delta, k)$
6:     **if** $y(t) \in F$ for $t \in [0, \Delta t]$ **then**
7:         Let $x = y(\Delta t)$ denote the end of the extension
8:         $V \leftarrow V \cup \{x\}, E \leftarrow E \cup \{x_{\text{near}} \rightarrow x\}$
9:         **if** $x \in G$ **then Return** path in $\mathcal{T}$ from $x_I$ to $x$.
```

Algorithm 4: Select-Control(x_s, x_t, d, δ, k).

```
1: $d_{\text{best}} \leftarrow \infty, y_{\text{best}} \leftarrow nil, \Delta t_{\text{best}} \leftarrow nil$
2: **for** $i = 1, 2, \ldots, k$ **do**
3:     $u \leftarrow$Sample($B(x_s)$)
4:     $\Delta t \leftarrow$Sample($[0, \delta]$)
5:     $y \leftarrow$Integrate($x_s, u, D, \Delta t$)
6:     **If** $d(y(\Delta t), x_t) < d_{\text{best}}$ **then**
7:         $d_{\text{best}} \leftarrow d(y(\Delta t), x_t)$
8:         $y_{\text{best}} \leftarrow y$
9:         $\Delta t_{\text{best}} \leftarrow \Delta t$
**return** $(y_{\text{best}}, \Delta t$best)
```

Algorithm 5: EST($\mathcal{P}, \delta, h, k$).

```
1:  $V \leftarrow \{x_I\}, E \leftarrow \{\}$
2:  for $i = 1, 2, \ldots$ do
3:      for $j = 1, 2, \ldots, k$ do
4:          $x_j \leftarrow$ Sample($\mathcal{T}$)
5:          $u \leftarrow$Sample($B(x_j)$)
6:          $\Delta t_j \leftarrow$Sample($[0, \delta]$)
7:          $y_j \leftarrow$Integrate($x_j, u, D, \Delta t_j$)
8:          $w_j \leftarrow 1/(1 + \rho_h(y_j(\Delta t_j))^2)$
9:      $(y, \Delta t) \leftarrow$Weighted random sample from
        $((y_1, \Delta t_1), \ldots, (y_k, \Delta t_k))$ with weights $(w_1, \ldots, w_k)$.
10:     if $y(t) \in F$ for $t \in [0, \Delta t]$ then
11:         Let $x = y(\Delta t)$ denote the end of the extension
12:         $V \leftarrow V \cup \{x\}$, $E \leftarrow E \cup \{x_{\text{near}} \rightarrow x\}$
13:         if $x \in G$ then Return path in $\mathcal{T}$ from $x_I$ to $x$.
```

Lines 3–4 select the closest node x_{near} in $\mathcal{T}$ to a randomly sampled state. Nearest-neighbor data structures, such as KD-trees, may be used to accelerate this query. Line 5 calls a subroutine that attempts to select a short extension trajectory y that brings x_{near} closer to that state. If the extension is feasible (line 6), it is added to the tree (lines 7 and 8). Note that the most efficient way of testing paths for feasibility is to first test the endpoint and, then, bisect until a collision is found or a minimum resolution has been reached.

The Select-Control subroutine, in its basic form, simply selects k random controls u and durations Δt, and picks the extension whose trajectory ends closest to x_t:

The Integrate subroutine produces a trajectory by integrating the dynamics $D(x, u)$ forward over the duration Δt, in the most basic case using Euler's method, although higher order methods such as Runge–Kutta may also be used. An optimization is to test the feasibility of the extension's end state during Select-Control, which avoids most obviously infeasible extensions. To do this, we replace the condition in line 6 with $(d(y(\Delta t), x_t) < d_{\text{best}}$ and $y(\Delta t) \in F)$.

EST's strategy differs in that it explicitly maintains an estimate of how densely $\mathcal{T}$ covers the state space. The density estimator is a function $\rho_h(x)$, where $h > 0$ is a kernel width parameter. The expansion strategy attempts to explore uniformly by sampling extensions inversely proportional to $\rho_h(x)$. Pseudocode is given in Algorithm 5.

Lines 3–9 sample a set of k random extensions $\{(y_1, \Delta t_1), \ldots, (y_k, \Delta t_k)\}$. The process samples a random node in $\mathcal{T}$, draws a random control sample from U, and integrates the dynamics over a duration Δt sampled from the range $[0, \delta]$. Then, it assigns a sampling weight to each candidate extension, inversely proportional to the density of its end state. As in RRT, as an optimization, we check the feasibility of a candidate end state $y_j(\Delta t_j) \in F$ before adding it to the list of candidate extensions.

To estimate density, several methods may be used. A kernel density estimator $\rho_h(x) = \sum_{y \in \mathcal{T}} \exp(-\|x - y\|^2/h^2)$ is straightforward but requires $O(|\mathcal{T}|)$ time to evaluate. An alternate method stores a hash grid with resolution h over space and counts the number of existing nodes in $\mathcal{T}$ in the same cell as x. Using a grid results in $O(1)$ time for lookup and $O(1)$ time to update. Moreover, we may sample a random node in $\mathcal{T}$ with probability inversely proportional to ρ_h in time $O(1)$ by uniformly sampling a random node from a uniformly sampled occupied cell. However, grids do not scale well to high-dimensional spaces, providing poorer density estimates as d increases. The method we use for $d > 3$ is to store $\binom{d}{3}$ projections of X onto three axes. Grids are stored on each projection, and the overall density is estimated as the average density over each projection.

Appendix B
State–Cost Space Expansiveness

This appendix will prove Lemma 1, which declares that state–space expansiveness implies state–cost space expansiveness. The crux of the issue is whether it is possible to generate a continuum of local trajectories to a given endpoint with a continuum of costs. We show that most spaces satisfy this condition. For those state–cost spaces that do not, we prove that the reachable set is degenerate in dimensionality and, hence, state–cost space planning is identical to state–space planning.

We make two assumptions: $f(x, u)$ is continuous in both arguments, and the cost function $L(x, u) > 0$ is continuous and positive. The following four definitions are given in [9].

Definition 5: The reachable set $\mathcal{R}(x)$ is the set of points reachable via any feasible trajectory, starting from the state x. The reachable set of a set S is the union of reachable sets of the points: $\mathcal{R}(S) = \bigcup_{x \in S} \mathcal{R}(x)$.

Definition 6: The local reachable set $\mathcal{R}_\ell(x)$ is the set of points reachable via feasible trajectories starting from the state x, derived by integrating exactly ℓ controls $u_1, \ldots, u_\ell$ integrated over durations $\Delta t_1, \ldots, \Delta t_\ell$ with $\Delta t_i \in [0, t_{\max}]$ with $t_{\max}$ a given constant for all i.

Definition 7: The β-lookout of a set $S \subseteq X$ is the subset of S that can locally reach a β fraction of the complement of S. In particular, $\beta - Lookout(S) = \{x \in S \mid \mu(\mathcal{R}_\ell(x) \setminus S) \geq \beta \mu(\mathcal{R}(S) \setminus S)\}$.

Definition 8: A space is (α, β)-expansive if for every $x \in F$ and every connected subset S of $\mathcal{R}(x)$, $\mu(\beta - Lookout(S)) \geq \alpha \mu(S)$.

The most important quality to verify is that $\mathcal{R}_\ell(x)$ has nonzero volume relative to the reachable set, that is, $\mu(\mathcal{R}_\ell(x)) \geq \gamma \mu(\mathcal{R}(x))$ for all x and some constant $\gamma > 0$. In state spaces with $\dim(U) < \dim(X)$, a notion of controllability must be satisfied, and the number of chosen controls ℓ must be large enough so that the movement directions span all $\dim(X)$ dimensions of state space. In particular, $\ell \geq \lceil \dim(X)/(\dim(U) + 1) \rceil$ because each subsequent control provides $\dim(U)$ degrees of freedom in U and another in integration time. If local reachable sets have volume zero, then expansiveness will certainly not hold. Otherwise, (α, β)-expansiveness generally holds, except for carefully constructed problems with infinitely thin narrow passages. Since cost space is obstacle free, no such narrow passages exist, and hence, state–cost expansiveness is implied by nonzero volume of local reachable sets.

Let us define notation for state–cost reachability.

Definition 9: The state–cost reachable set $\mathcal{R}^{\text{sc}}(x, c)$ is the set of state–cost points reachable from starting state x and starting cost c via any feasible trajectory. To bound the volume of this region, we restrict c to some maximum value $C_{\max}$ and define state–cost volume metric μ in the usual Lesbegue sense.

Definition 10: The local state–cost reachable set $\mathcal{R}^{\text{sc}}_k(x, c)$ is the set of state–cost points reachable from starting state x and starting cost c via a trajectory derived by integrating exactly k controls and k durations in the range $[0, t_{\max}]$.

We now set out to prove that there exists a value of k such that $\mu(\mathcal{R}^{\text{sc}}_k(x, c)) \geq \gamma\mu(\mathcal{R}^{\text{sc}}(x, c))$ for some $\gamma > 0$, i.e., the local reachable set spans a nonzero volume in the additional cost dimension. We shall determine that $k = 2\ell$ suffices. (This leads to an elegant proof, but usually far fewer than 2ℓ controls are actually needed).

We proceed to show that $\dim(\mathcal{R}^{\text{sc}}_{2\ell}(x, c)) = \dim(X) + 1$ in case-by-case elimination, except for one particularly strange case, in which the true reachable set is degenerate. First, consider the state–cost reachable set with $k = \ell$. In the first case, we may luckily find $\dim(\mathcal{R}^{\text{sc}}_{\ell}(x, c)) = \dim(X) + 1$, and because $\mathcal{R}^{\text{sc}}_{\ell}(x, c) \subseteq \mathcal{R}^{\text{sc}}_{2\ell}(x, c)$, we are done and Lemma 1 is proven. In the second case, $\dim(\mathcal{R}^{\text{sc}}_{\ell}(x, c)) = \dim(X)$. (The case $\dim(\mathcal{R}^{\text{sc}}_{\ell}(x, c)) < \dim(X)$ is impossible because the projection of $\mathcal{R}^{\text{sc}}_{\ell}(x, c)$ onto the state dimensions is identically $\mathcal{R}_{\ell}(x)$, which has dimensionality $\dim(X)$).

At any point y in the interior of $\mathcal{R}_{\ell}(x)$, a cost-to-go surface $c(y)$ can be defined by the span of $\dim(X)$ degrees of freedom of the control trajectory. If there are multiple spans, let us choose one arbitrarily. To make the dependence on initial state explicit, we denote this surface $c(x, y)$.

Consider any point $z \in \mathcal{R}^{\circ}_{\ell}(y)$ in the interior of the ℓ-reachable set of y. Note that z is reachable from x with 2ℓ controls, in particular the ℓ controls from x to y and then the ℓ controls from y to z. Now, consider rerouting the path $x \rightsquigarrow z$ rerouted through some point $y' \in \mathcal{N}(y, \epsilon)$ in the neighborhood of y. Such a neighborhood exists via the following lemma.

Lemma 2: Since the reachable set $\mathcal{R}_{\ell}(y)$ has dimensionality $\dim(X)$ and its boundaries are continuous in y, for any point z in $\mathcal{R}^{\circ}_{\ell}(y)$, there exists a radius $\epsilon > 0$ such that all points $y' \in \mathcal{N}(y, \epsilon)$ have $z \in \mathcal{R}_{\ell}(y')$.

Since the neighborhood of y is locally reachable from x and z is locally reachable from all points, the cost-to-go functions $c(x, y)$ and $c(y, z)$ are locally defined and are moreover continuous in their first and second arguments. The question is whether moving the midpoint affects the overall cost, that is, whether $c(x, y') + c(y', z) = c(x, y) + c(y, z)$ holds? If not, then there exists a continuum of costs that may be accumulated while passing from x to z, and hence, $\dim(\mathcal{R}^{\text{sc}}_{2\ell}(x, c)) = \dim(X) + 1$ as required to prove Lemma 1. Hence, the only remaining case to inspect is whether it is possible to obtain $c(x, y') + c(y', z) = c(x, y) + c(y, z)$ for all $y' \in \mathcal{N}(y, \epsilon)$. Taking y' to be an infinitesimal disturbance of y, we see that the PDE

$$\frac{\partial c}{\partial y}(x, y) + \frac{\partial c}{\partial x}(y, z) = 0 \tag{19}$$

must be satisfied by c. Note that the partial derivatives $\frac{\partial c}{\partial x}$ and $\frac{\partial c}{\partial y}$ are derivatives with respect to the first and second arguments, respectively. Now, taking the derivative of this condition with respect to the initial state x, we see that

$$\frac{\partial^2 c}{\partial x \partial y}(x, y) = 0. \tag{20}$$

The general solution to this equation is $c(x, y) = g(x) + h(y)$. Using the knowledge that $c(x, x) = 0$ for all x, we see that $h(y) = -g(y)$, and hence, $c(x, y) = g(x) - g(y)$. This requirement says that the only problems in which 2ℓ controls will not expand the locally reachable set in the cost dimension are those in which the cost-to-go function is path independent.

Theorem 6: If a state space is expansive with local reachable set $\mathcal{R}_{\ell}$ and the cost-to-go function is path dependent, then the corresponding state–cost space is expansive. Furthermore, expansiveness holds with local reachable set $\mathcal{R}^{\text{sc}}_{2\ell}$.

Proof: The proof is the contrapositive of the above argument. Since the cost-to-go function is not path independent, each locally reachable set with 2ℓ controls will span a nonzero volume in the cost dimension. ■

Corollary 2: Expansiveness holds in reversible systems.

Proof: Cost-to-go is path dependent in such systems: a self-loop has positive cost but a null movement has zero cost. ■

Path dependence is the usual case, and therefore, this theorem holds for almost any conceivable real-world problem. However, it is possible to construct artificial path-independent problems, such as the problem $f(x, u) = 1, L(x, u) = 1$, in which x increases monotonically, and so does cost. However, we can show that for such systems, the reachable set in state–cost space is indeed degenerate, and hence, both the local reachable set and the reachable set have zero volume.

Theorem 7: In path-independent problems, the set of state–cost points reachable with a countable number of piecewise constant controls has degenerate dimensionality $\dim(R^{\text{sc}}(x, c)) = \dim(X)$, and hence, expansiveness holds trivially.

Proof: For any state y reachable from x with a countable number of piecewise-constant controls, we may repeatedly connect to a state on the path with ℓ controls, then to another with 2ℓ controls, etc., showing that path independence holds across the entire trajectory. As a result, the cost of any trajectory from x to y is path independent, and hence, the reachable set is the graph $\mathcal{R}^{\text{sc}}(x, 0) = \{(y, c(x, y)) \mid y \in \mathcal{R}(x)\}$. This set has dimensionality $\dim(X)$. ■

References

[1] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2011, pp. 2640–2645.

[2] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guidance, Control, Dyn.*, vol. 21, no. 2, pp. 193–207, 1998.

[3] J. E. Bobrow, B. Martin, G. Sohl, E. C. Wang, F. C. Park, and J. Kim, "Optimal robot motions for physical criteria," *J. Robot. Syst.*, vol. 18, no. 12,pp. 785–795, 2001.

[4] W. F. Carriker, P. K. Khosla, and B. H. Krogh, "The use of simulated annealing to solve the mobile manipulator path planning problem," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1990, vol. 1, pp. 204–209.

[5] A. Dobson and K. E. Bekris, "A study on the finite-time near-optimality properties of sampling-based motion planners," in *Proc.*

IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2013, pp. 1236–1241, doi: 10.1109/IROS.2013.6696508.

[6] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2006, pp. 5369–5375.

[7] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field D* algorithm," *J. Field Robot.*, vol. 23, no. 2, pp. 79–101, 2006.

[8] K. Hauser and J. -C. Latombe, "Multi-modal planning in non-expansive spaces," *Int. J. Robot. Res.*, vol. 29, no. 7, pp. 897–915, 2010.

[9] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, Mar. 2002.

[10] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. IEEE Conf. Decis. Control Eur. Control Conf.*, Dec. 2011, pp. 3276–3282, doi: 10.1109/CDC.2011.6161521.

[11] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proc. 49th IEEE Conf. Decis. Control*, 2010, pp. 7681–7687.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[13] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5041–5047.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996, doi: 10.1109/70.508439.

[15] J.-P. Laumond, N. Mansard, and J.-B. Lasserre, "Optimality in robot motion: Optimal versus optimized motion," *Commun. ACM*, vol. 57, no. 9, pp. 82–89, Sep. 2014, doi: 10.1145/2629535.

[16] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1999, vol. 1, no. 1, pp. 473–479, doi: 10.1109/ROBOT.1999.770022.

[17] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Proc. 11th Int. Workshop Alg. Found. Robot.*, Istanbul, Turkey, 2014, pp. 263–282.

[18] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on sub-optimality," in *Proc. 16th Int. Conf. Neural Inf. Process. Syst.*, 2003, pp. 767–774.

[19] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Int. J. Robot. Res.*, vol. 35, Apr. 2016, pp. 528–564, doi:10.1177/0278364915614386.

[20] R. Luna, I. A. Sucan, M. Moll, and L. E. Kavraki, "Anytime solution optimization for sampling-based motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5068–5074, doi: 10.1109/ICRA.2013.6631301.

[21] J. Luo and K. Hauser, "An empirical study of optimal motion planning," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Sep. 2014, pp. 1761–1768.

[22] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 2537–2542.

[23] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 489–494, doi: 10.1109/ROBOT.2009.5152817.

[24] J. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci. USA*, vol. 93, pp. 1591–1595, 1996.

[25] S. Stoneman and R. Lampariello, "Embedding nonlinear optimization in RRT* for optimal kinodynamic planning," in *Proc. IEEE Conf. Decision Control*, 2014, pp. 3737–3744.

[26] S. G. Vougioukas, "Optimization of robot paths computed by randomized planners," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 2148–2153.

[27] D. Webb and J. van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear differential constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5054–5061.

Kris Hauser (M'09) received the bachelor's degree in computer science and mathematics from University of California, Berkeley, CA, USA, in 2003 and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 2008.

He is an Associate Professor with Duke University, Durham, NC, USA, with joint appointments with the Department of Electrical and Computer Engineering and the Department of Mechanical Engineering and Materials Science. He was a Postdoctoral Researcher with University of California, Berkeley. He was with the faculty of Indiana University from 2009 to 2014, where he started the Intelligent Motion Laboratory and began his current position with Duke University in 2014. His research interests include robot motion planning and control, semiautonomous robots, and integrating perception and planning, as well as applications to intelligent vehicles, robotic manipulation, robot-assisted medicine, and legged locomotion.

Dr. Hauser received the Stanford Graduate Fellowship, the Siebel Scholar Fellowship, the Best Paper Award at IEEE Humanoids 2015, and the National Science Foundation CAREER Award.

Yilun Zhou is an undergraduate with Duke University, Durham, NC, USA, expecting to receive dual B.Sc. degrees in electrical and computer engineering and in computer science in December 2016.

He plans to pursue the Ph.D. degree in robotics or a related field starting in 2017. He has experience in motion planning, learning from demonstration, task representation, and deep learning. His research interests include developing and applying machine learning methods to improve robot planning and perception.