Bang-Bang Boosting of RRTs

Alexander J. LaValle, Basak Sakcak, and Steven M. LaValle

***Abstract*— This paper presents methods for dramatically improving the performance of sampling-based kinodynamic planners. The key component is a complete, exact steering method that produces a time-optimal trajectory between any states for a vector of synchronized double integrators. This method is applied in three ways: 1) to generate RRT edges that quickly solve the two-point boundary-value problems, 2) to produce a (quasi)metric for more accurate Voronoi bias in RRTs, and 3) to iteratively time-optimize a given collision-free trajectory. Experiments are performed for state spaces with up to 2000 dimensions, resulting in improved computed trajectories and orders of magnitude computation time improvements over using ordinary metrics and constant controls.**

I. INTRODUCTION

Rapidly exploring random trees were originally introduced as an approach to motion planning with differential constraints and dynamics [21]. The idea was to incrementally grow a space-filling tree by applying controls so that two-point boundary-value problems could be avoided if popular methods such as probabilistic roadmaps [17] were applied to these problems. Curiously, RRTs have found more success over the past decades for basic path planning (no differential constraints and dynamics), rather than their intended target, the *kinodynamic planning problem* [6].

Although this phenomenon is partly due to larger mainstream interest since the 1990s in basic path planning, it is primarily due to the additional challenges posed by the harder problems. Computational performance depends greatly how well the RRT nearest-neighbor metric approximates the true, optimal cost-to-go function, which is presumably unattainable. Furthermore, efficient steering methods or motion primitives are often needed to enhance performance, rather than applying constant controls as in [21]. Indeed, the most successful kinodynamic RRT planning methods have exploited the existence of simple cost-to-go functions and steering methods (ignoring obstacles) for special classes of systems [8], [10], [19], [27], [30], [32] or have relied on numerical solutions computed offline for more general systems and optimization objectives [29]. Learning-based approaches to estimate the cost-to-go function have also been proposed [3], [22], [26], [33]. Inspired by all of these works, we enhance RRT performance by using metrics and steering methods based on bang-bang time-optimal controls [28]. It was already shown that RRT exploration seems to improve with bang-bang metrics [10], [15].

We continue in this direction by developing a steering method (and proving its correctness) that completely solves the problem of time-optimally steering a vector of double integrators from any initial state to any goal state with a synchronized arrival time. Optimal solutions are easily and exactly computed as two- to four-piece constant controls for each double integrator, resulting in piecewise-constant controls for the whole system, and trajectories as parabolic arcs in the configuration and state (phase) spaces. The challenge is to ensure that all double integrators arrive at their goal states at the same time, which is often impossible due to momentum, unless some form of time-stretching or waiting is inserted.

We then show experimentally that the steering method improves RRT performance by orders of magnitude when compared to the original method that uses weighted Euclidean metrics and constant controls over fixed time intervals. The study presented here is focused on double integrator dynamics, which lies at the core of fully actuated systems, with the intention of extending it to more general dynamics of arbitrary stabilizable systems (similar to the way it was accomplished in [13]). As a step in this direction, we also present some preliminary results for a non-double integrator system, representing a vehicle on a curved surface.

This paper also presents methods that rapidly optimize collision-free trajectories by iteratively applying the simple time-optimal steering method to the output of sampling-based planners. We consider two cases: 1) directly optimizing the result of a kinodynamic RRT-based planners, and 2) converting the piecewise-linear path produced by an RRT-based planner for basic path planning [18] into a trajectory by applying bang-bang controls along each segment and then further iteratively optimizing the result. Our experiments indicate that the second method is more efficient; however, it is limited to problems in which the initial and goal states are at zero velocity (if some form of completeness is demanded). An alternative to these optimizations would be to apply asymptotically optimal extensions of RRTs, such as RRT* [16] or SST* [23]; however, we are motivated by the evidence that "plan first and optimize later" often produces optimal paths more quickly and consistently than asymptotically optimal planning [12], [25].

The paper continues as follows. Section II defines the problem. Section III develops the algebraic details of bang-bang time optimal control for single and multiple, parallel double integrators. Section IV presents the new planning and optimization methods. Section V gives implementation details, computed results, and performance analysis. Section VI summarizes the results, their implications, and discusses the logical next steps.

The authors are with Center for Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland. Email: firstname.lastname@oulu.fi

II. PROBLEM DEFINITION

Let $\mathcal{C}$ be the robot *configuration space*, assumed to be an n-dimensional smooth manifold with points referenced using local coordinates on $\mathbb{R}^n$. Geometric models (typically piecewise-linear) are given for the robot and its (static) environment, and the robot model transforms for each q depending on robot kinematics. Let $\mathcal{C}_{free}$ be the open subset of $\mathcal{C}$ in which the robot does not intersect obstacles. See [4], [20] for more details.

The first problem is:

Problem 1 (Basic path planning) *Given any $q_I, q_G \in \mathcal{C}_{free}$, compute a path $\tau : [0,1] \to \mathcal{C}_{free}$ such that $\tau(0) = q_I$ and $\tau(1) = q_G$.*

This problem ignores kinematic constraints and dynamics, leading to a second, harder problem that takes these into account. Building upon $\mathcal{C}$ and $\mathcal{C}_{free}$, let $x = (q, \dot{q})$ be a $2n$-dimensional *state* vector for every configuration $q \in \mathcal{C}$. The set of all x forms X, the *state space*, which is the tangent bundle $T(\mathcal{C})$. Assume $\mathcal{C}_{free}$ is lifted into X as $X_{free} = \{(q, \dot{q}) \in X \mid q \in \mathcal{C}_{free}\}$.

Let $\dot{x} = f(x, u)$ define a standard *control system* on X, in which u belongs to a compact *action set* $U \subset \mathbb{R}^m$. For convenience in this paper, we will equivalently define the control system in terms of the accelerations that actions $u \in U$ induce at a particular $q \in \mathcal{C}$. Thus, let $A(x) = A(q, \dot{q})$ be the set of all accelerations $\ddot{q}$ that can be obtained at $(q, \dot{q})$ by applying an action $u \in U$ for the system f. Let $a \in A(x)$ denote a particular acceleration vector ($a = \ddot{q}$) that may be applied. Let $A = \cup_{x \in X} A(x)$. Let $\Phi(x, \tilde{a})$ denote the state trajectory $\tilde{x} : [0, t_F] \to X$ obtained by integrating a control $\tilde{a} : [0, t_F] \to A$ from state x. This leads to the next problem:

Problem 2 (Kinodynamic planning) *Given any $x_I, x_G \in X_{free}$, compute an acceleration* control $\tilde{a} : [0, t_F] \to A$ *for which $\tilde{a}(t) \in A(\tilde{x}(t))$ for all $t \in [0, t_F]$, and that produces a state trajectory $\tilde{x} = \Phi(x_I, \tilde{a})$, with $\tilde{x} : [0, t_F] \to X_{free}$ and $\tilde{x}(t_F) = x_G$.*

We also consider *time optimality* in some cases, which means that a solution is chosen for which t_F is as small as possible among all possible solutions.

These restricted versions will be considered in this paper:

1) *Stabilizable:* For all $x \in X$, $A(x)$ contains an open set that contains the origin $\mathbf{0}$. This implies that for any x_I, x_G, there exists a finite-time acceleration control that solves the kinodynamic planning problem if $X_{free} = X$ (no obstacles).
2) *Rest-to-rest:* These problems require that for $x_I = (q_I, \dot{q}_I)$ and $x_G = (q_G, \dot{q}_G)$, $\dot{q}_I = \dot{q}_G = \mathbf{0}$.
3) *nD-double-integrator:* Each q_i is independently actuated within global acceleration bounds $a_{min,i} < 0$ and $a_{max,i} > 0$. In this case, $A(x)$ is an axis-aligned rectangle that contains the origin, fixed for all $x \in X$.

III. TIME-OPTIMAL ACCELERATIONS

The task in this section is to calculate time-optimal solutions to the kinodynamic problem for the nD double integrator model and no obstacles: $X_{free} = X = \mathbb{R}^{2n}$. The solutions will work out so that controls are piecewise-constant,

$$\tilde{a} = ((a_1, t_1), (a_2, t_2), \ldots, (a_n, t_n)), \tag{1}$$

which means that each a_i is applied for duration t_i, starting at time $t_1 + t_2 \ldots t_{i-1}$. Initially, a_1 is applied at time $t = 0$.

A. Time-optimal control of a double integrator

Consider one double integrator, for which every $q \in \mathbb{R}$. The allowable accelerations form a closed interval $A = [a_{min}, a_{max}]$, in which $a_{min} < 0$ and $a_{max} > 0$. It is a control system of the form $\ddot{q} = a$ for $a \in [a_{min}, a_{max}]$ and has an associated *phase plane*, with coordinates $(q, \dot{q}) \in \mathbb{R}^2$. The task is to determine an acceleration control $\tilde{a} : [0, t_F] \to A$ for which $\Phi(x_I, \tilde{a}) = x_G$ and t_F is as small as possible.

Pontryagin's maximum principle provides necessary conditions on the time-optimal trajectory by considering a co-state vector (λ_1, λ_2) that serves as a generalized Lagrange multipliers for the constrained optimization problem [24]. Following the standard theory, the Hamiltonian is defined as

$$H(x, a, \lambda) = 1 + \lambda_1 x_2 + \lambda_2 a, \tag{2}$$

in which $x_1 = q$ and $x_2 = \dot{q}$, and the optimal $\tilde{a}$ is constrained to

$$\tilde{a}^*(t) = \operatorname*{argmin}_{a \in A} \{1 + \lambda_1(t) x_2(t) + \lambda_2(t) \tilde{a}(t)\}. \tag{3}$$

Solving the adjoint equation $\dot{\lambda}_i = -\frac{\partial H}{\partial x_i}$ results in $\lambda_1(t) = c_1$ and $\lambda_2(t) = c_2 - c_1 t$ for unknown constants c_1 and c_2. If $\lambda_2(t) < 0$, then $\tilde{a}^*(t) = a_{max}$, and if $\lambda_2(t) > 0$, then $\tilde{a}^*(t) = a_{min}$. Thus, the action may be assigned as $\tilde{a}^*(t) = -\mathrm{sgn}(\lambda_2(t))$, if $\lambda_2(t) \neq 0$. At the boundary case in which $\lambda_2(t) = 0$, any $a \in A$ may be chosen. Since $\lambda_2(t)$ is linear, it may change signs at most once, implying that the optimal control involves at most two "bangs", each corresponding to an extremal acceleration applied over a bounded time interval. Thus, the time-optimal control is of the form $\tilde{a}^* = ((a_1, t_1), (a_2, t_2))$, with degenerate possibilities of $t_1 = 0$ or $t_2 = 0$.

Further algebraic analysis is needed to precisely determine a_1, t_1, a_2, and t_2 for a given $(q_I, \dot{q}_I)$ and $(q_G, \dot{q}_G)$. See Figure 1. If a constant acceleration a is applied at any phase $(q_0, \dot{q}_0)$, then a parabolic curve is traced out in the phase plane. If $q_0 = \dot{q}_0 = 0$, then its equation is $q = \frac{1}{2}\dot{x}^2$. More generally, it satisfies $q - \frac{1}{2}\dot{q}^2 = c$, in which the coefficient $c = q_0 - \frac{1}{2}\dot{q}_0^2$ corresponds to the parabola's intersection with the $\dot{q} = 0$ axis.

Next consider four parabolas based on all combinations of initial and goal states, and extremal controls. Let I^+ denote the parabola obtained from setting $(q_0, \dot{q}_0) = x_I = (q_I, \dot{q}_I)$ and applying constant control a_{max}; let $c(I^+)$ denote its $\dot{q} = 0$ intercept. Similarly, I^- is obtained by applying a_{min}. Furthermore, G^+ and G^- are obtained by setting

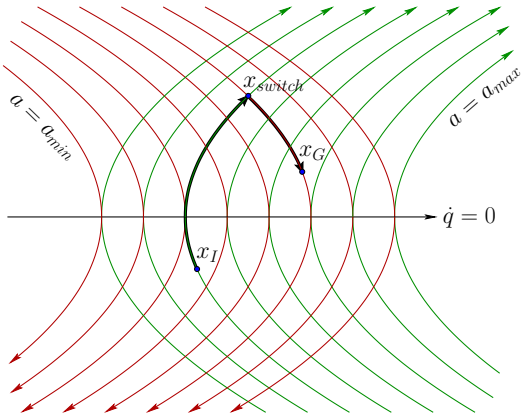


Fig. 1. Intersecting parabolas and traveling forward in time produces the time-optimal trajectory in the phase plane, which generally involves two-piece constant controls and parabolic trajectories.

$(q_0, \dot{q}_0) = x_G$ and applying a_{max} and a_{min}, respectively. Assuming $x_I \neq x_G$, consider all possible intersections of I^+, I^-, G^+ and G^-. There are only two possible types: I^+G^- and I^-G^+. The first results in a_{max} applied until the intersection point, $x_{switch} = (q_{switch}, \dot{q}_{switch})$, is reached, followed by applying a_{min}. The second type applies a_{min} first, followed by a_{max}.

The type I^+G^- intersection occurs if $c(I^+) > c(G^-)$. The intersection position is the midpoint $q_{switch} = (c(I^+) + c(G^-))/2$, and the velocity is $\dot{q}_{switch} = \sqrt{2(q_{switch} - c(I^+))}$. Similarly, the type I^-G^+ intersection occurs if $c(I^-) < c(G^+)$. The intersection position is the midpoint $q_{switch} = (c(I^-) + c(G^+))/2$, and the velocity is $\dot{q}_{switch} = -\sqrt{2(q_{switch} - c(G^+))}$.

To determine the control timings to go from x_I to x_{switch} to x_G note that changing velocity by an amount d with constant acceleration a requires time d/a. The timings are:

$$t_1 = (\dot{q}_{switch} - \dot{q}_I)/a_1 \tag{4}$$

and

$$t_2 = (\dot{q}_G - \dot{q}_{switch})/a_2, \tag{5}$$

respectively.

There will always be at least one intersection, but sometimes there are both I^+G^- and I^-G^+. In this case, t_1 or t_2 may be negative for one intersection, but the other intersection provides a valid control. It is also possible to obtain valid controls for both cases, in which case the one that requires least time must be selected (Figure 2 will provide more details).

The arguments above lead to the following proposition:

Proposition 1 *The time-optimal control for the double integrator problem is $((a_1, t_1), (a_2, t_2))$, in which $a_1 = a_{min}$ and $a_2 = a_{max}$, or $a_1 = a_{max}$ and $a_2 = a_{min}$, and the durations t_1 and t_2 are given by (4) and (5).*

B. Time-optimal control of a vector of double integrators

To extend the result to a vector of double integrators, a waiting method is needed so that each double integrator arrives at its goal in its phase plane at the same time.

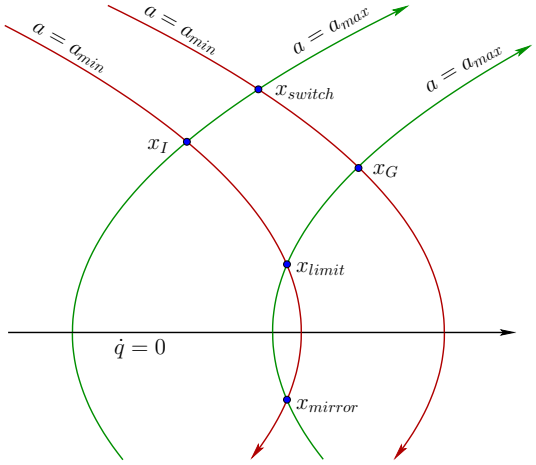


Fig. 2. If both I^+G^- and I^-G^+ intersections occur, then there is a gap interval, which disallows certain waiting times. This poses a significant challenge to synchronizing the arrival times of n double integrators, which is overcome in our paper and is critical to our new planning algorithms.

This problem was considered in [7], [15], but only for the limited case in which their final velocities are zero. Our work builds upon the observations in [11], [19]. We introduce an explicit, complete, provably time-optimal, and computationally efficient solution to the general problem of steering of n independent integrators for any initial and goal state pairs in their respective phase planes in $O(n \lg n)$ time.

For a fixed x_I and x_G, let t^* be the time to reach the goal by applying the bang-bang solution of Section III-A. Now consider some $t_w > t^*$. Does a control $\tilde{u}$ necessarily exist that will cause x_G to be reached at exactly time t_w? The answer is yes if there is only one intersection type (I^+G^- or I^-G^+). However, if both intersections occur, then the situation depicted in Figure 2 occurs (or its symmetric equivalent for $\dot{q} < 0$). The path from x_I to $x_{limit} = (q_{limit}, \dot{q}_{limit})$ to x_G corresponds to the slowest trajectory that reaches x_G while remaining in the $\dot{q} > 0$ half-plane. The critical switching point x_{limit} can be calculated using the parabola intersection algebra from Section III-A. The time taken by this trajectory is calculated as

$$t_{limit} = (\dot{q}_{limit} - \dot{q}_I)/a_{min} + (\dot{q}_G - \dot{q}_{limit})/a_{max}. \tag{6}$$

Thus, if $t_w \in [t^*, t_{limit}]$, then a solution exists (and requires only two constant control segments). If $t_w > t_{limit}$, then x_G can no longer be reached while remaining in the $\dot{q} > 0$ half-plane. The next available time is obtained by continuing to apply $a = a_{min}$ until the second parabolic intersection point, called x_{mirror} is reached, in the $\dot{q} < 0$ half-plane. Note that $x_{mirror} = (q_{limit}, -\dot{q}_{limit})$. Once x_{mirror} is reached, $a = a_{max}$ is applied to arrive optimally at x_G. The time required to traverse this trajectory is

$$t_{mirror} = t_{limit} + 2\dot{q}_{limit}/a_{max} - 2\dot{q}_{limit}/a_{min}. \tag{7}$$

Thus, there may generally be a *gap interval* (t_{limit}, t_{mirror}) for which no solution exists.

Now suppose that $t_w \geq t^*$ and there is no gap interval, or $t_w \in [t^*, t_{limit}] \cup [t_{mirror}, \infty)$. A two-piece control

$((a_1, t_1), (a_2, t_2))$ can be calculated as a solution to the following equations corresponding to the boundary conditions:

$$\dot{q}_I t_1 + \frac{a_1 t_1^2}{2} + (\dot{q}_I + a_1 t_1) t_2 + \frac{a_2 t_2^2}{2} = q_G - q_I, \quad (8)$$

$$a_1 t_1 + a_2 t_2 = \dot{q}_G - \dot{q}_I, \quad (9)$$

while satisfying $a_1, a_2 \in [a_{min}, a_{max}]$. Note that $t_2 = t_w - t_1$ and $0 \leq t_1, t_2 \leq t_w$. The solution is usually not unique, and can be selected either arbitrarily or by optimizing a relevant optimization objective, such as energy.

Proposition 2 *If a solution exists for prescribed time t_w to reach x_G from x_I, then the above waiting method generates a control that achieves it; otherwise, it reports failure (implying that t_w lies in the gap interval).*

Proof: Let $\tilde{a}_u$ and $\tilde{a}_l$ be two control trajectories defined as $\tilde{a}_u = ((a_{max}, t_u), (a_{min}, t_w - t_u))$ and $\tilde{a}_l = ((a_{min}, t_l), (a_{max}, t_w - t_l))$. Furthermore, for given x_I and x_G, t_u and t_l satisfy that $\dot{q}_u(t_w) = \dot{q}_l(t_w) = \dot{q}_G$, in which $(q_u, \dot{q}_u) = \Phi(x_I, \tilde{a}_u)$ and $(q_l, \dot{q}_l) = \Phi(x_I, \tilde{a}_l)$. Suppose a solution exists for t_w and let $\tilde{a} : [0, t_w] \to [a_{min}, a_{max}]$ be a solution control trajectory such that $\tilde{x}(t_w) = x_G$, in which $\tilde{x} = (q, \dot{q}) = \Phi(x_I, \tilde{a})$. For all $t \in [0, t_w]$, it is true that

$$\dot{q}_l(t) \leq \dot{q}(t) \leq \dot{q}_u(t). \quad (10)$$

Suppose this is not true and there exists a $t' \in [0, t_w]$ for which $\dot{q}(t') > \dot{q}_u(t')$. Then, if $t' \leq t_u$ it implies that $\int_0^{t'} \tilde{a}(t)dt > \int_0^{t'} \tilde{a}_u(t)dt$ which violates the condition that $\tilde{a}(t) \leq a_{max}$ for all $t \in [0, t_w]$. The same reasoning can be done for $t' > t_u$ and for $\dot{q}(t) < \dot{q}_l(t)$. It follows from (10) that

$$q_l(t) \leq q(t) \leq q_u(t) \quad (11)$$

for all $t \in [0, t_w]$. In particular, $q_l(t_w) - q_I \leq q_G - q_I \leq q_u(t_w) - q_I$. Then, there exist $a_1, a_2 \in [a_{min}, a_{max}]$ and $0 \leq t_1 < t_w$ that satisfy the boundary conditions given in (8) and (9). This proves that if a solution $\tilde{a}$ exists for t_w, then, there also exists a two-piece solution $((a_1, t_1), (a_2, t_w - t_1))$. As a consequence, if there does not exist a two-piece solution, then, there also does not exist a solution for t_w. This happens for example, if $q_G < q_l(t_w)$ or $q_G > q_u(t_w)$ for given t_w, x_I, and x_G. ■

Putting these results together, a control $\tilde{u}$ can be determined for any t_w, unless it is in the gap interval. If $t_w \geq t_{mirror}$ and $q_{init} \geq 0$, then a four-piece solution is obtained by: 1) maximum deceleration to rest from x_I, 2) waiting for time $t_w - t_{mirror}$, 3) maximum deceleration to x_{mirror}, and 4) maximum acceleration to x_G. A symmetric equivalent solution applies for $q_{init} \leq 0$. If $t_w \in [t^*, t_{limit}]$, then the two-piece solution from (8) and (9) is used (this method could even be applied if $t_w \geq t_{mirror}$, but this was not attempted).

Now suppose that the optimal times have been calculated for n double integrators to start at some x_I and end at some x_G. In the worst case, every double integrator could have a gap interval. The problem is to find the smallest time t such that $t \geq t^*$ and $t \notin (t_{limit}, t_{mirror})$. If a double integrator has no gap interval, then that part of the condition is dropped. A simple algorithm is contained in the proof of the following proposition:

```
BANG_BANG_RRT_BIDIRECTIONAL(x_I, x_G)
 1  T_a.init(x_I); T_b.init(x_G)
 2  for i = 1 to K do
 3      x_n ←BANG-BANG-NEAREST(S_a, α(i))
 4      x_s ←BANG-BANG-STEER(x_n,α(i))
 5      if x_s ≠ x_n then
 6          T_a.add_vertex(x_s)
 7          T_a.add_edge(x_n, x_s)
 8          x'_n ← BANG-BANG-NEAREST(S_b, x_s)
 9          x'_s ← BANG-BANG-STEER(x'_n,x_s)
10          if x'_s ≠ x'_n then
11              T_b.add_vertex(x'_s)
12              T_b.add_edge(x'_n, x'_s)
13          if x'_s = x_s then return SOLUTION
14      if |T_b| > |T_a| then SWAP(T_a, T_b)
15  return FAILURE
```

Fig. 3. Bidirectional RRT with bang-bang steering and quasimetric.

Proposition 3 *The time-optimal steering problem for n double integrators can be solved in $O(n \lg n)$ time.*

Proof: A simple line sweeping algorithm [5] achieves the bound as follows. Sort the optimal, limit, and mirror times for all double integrators into a single array of length $O(n)$. Sweep incrementally across the array, starting from the shortest time. In each step increment or decrement a counter of the number of double integrators that have a solution; t_{limit} causes decrementing and the other two cases cause incrementing. Each step takes $O(1)$ time. The method terminates with the optimal t when all n integrators admit a solution. The overall algorithm takes time $O(n \lg n)$ time due to the initial sorting. ■

IV. PLANNING METHODS

A. Kinodynamic RRT with bang-bang metric and steering

Suppose a kinodynamic planning problem is given for an nD-double integrator system. Figure 3 presents an outline of a balanced bidirectional RRT-based planning algorithm that uses bang-bang methods for both the metric and the steering method. A single-tree *goal-biased* algorithm could alternatively be made [19], [21]. Let $\alpha(i) \in X$ denote the random state obtained in iteration i (α could alternatively be a deterministic sequence that is dense in X [20]). Line 3 returns the nearest state x_n among all points S_a visited by tree T_a. Using the tools from Section III, there are two natural choices for the (quasi)metric, $\rho(x, x')$, which is an estimate of the distance from x to x'. Note it is not symmetric for our problem. The first choice $\rho_1(x, x')$ is the maximum time $t_1 + t_2$ from (4) and (5), taken over all n double integrators. A slower and more accurate metric is $\rho_2(x, x')$

is the time with waiting, t_w, from Section III-B, which is the time it takes for every double integrator to arrive at x'. Note that these metrics are expected to produce a better Voronoi-bias [21] because they are closer to the true optimal cost-to-go function.

Line 4 is the time-optimal steering method from Section III-B. Collision checking is performed along the trajectory, and the steering stops at x_s if an obstacle is hit or $\alpha(i)$ is reached. The new trajectory is added to T_a (and S_a). In practice, this was accomplished in our experiments by inserting into T_a nodes and edges along the trajectory; an exact method could alternatively be developed for representing and computing nearest points in S_a.

Lines 8 and 9 are similar to Lines 3 and 4, except that an attempt is made to connect the newest visited point x_s to the nearest point S_b in the other tree, T_b. Line 14 swaps the roles of the tree so that the smaller one explores toward $\alpha(i)$ and the larger one attempts to connect to the newly reached point.

If a more general, stabilizable system (recall from Section II) is given, then it can be converted into an nD-double integrator by restricting A to an compact, axis-aligned rectangular region that contains the origin. Such a subset of A always exists, and lies in the intersection of the open subsets of $A(x)$ that contain $\mathbf{0}$, for all $x \in X$. If the rectangle is small relative to $A(x)$ at each x, then we expect the solutions produced by the algorithm in Figure 3 to be further from their potential optima; a step toward investigating this problem is taken at the end of Section V.

B. Bang-bang trajectory optimization

Let X, X_{free}, x_I, x_G, and A be fixed for an nD double integrator system. Suppose that a piecewise-constant control $\tilde{a} : [0, t_F] \rightarrow A$ is given so that the resulting $\tilde{x} = \Phi(x_I, \tilde{a})$ is solution to Problem 2 from Section II. The task is to replace $\tilde{a}$ with a new control $\tilde{a}' : [0, t'_F] \rightarrow A$ so that $t'_F < t_F$ while maintaining the constraints that the trajectory maps into X_{free} and arrives at x_G at time t'_F. The new control $\tilde{a}'$ is constructed by selecting t_1 and t_2 such that $0 \leq t_1 < t_2 \leq t_F$.

For a given control, $\tilde{a}$, let $\tilde{a}[t_1, t_2]$ denote its restriction to the interval $[t_1, t_2]$. Thus, $\tilde{a}[t_1, t_2] : [t_1, t_2] \rightarrow A$.[1] The original control $\tilde{a}$ can be expressed as a sequence of three controls $\tilde{a}[0, t_1]$, $\tilde{a}[t_1, t_2]$, and $\tilde{a}[t_2, t_F]$ by applying Φ to each in succession. We replace the middle portion with a bang-bang control $\tilde{a}'[t_1, t'_2]$ using the methods of Section III-B. Since the method is time-optimal, it is known that $t'_2 \leq t_2$ (they are equal only if $\tilde{a}[t_2, t_F]$ is already time-optimal). The new control must satisfy

$$\Phi(\Phi(x_I, \tilde{a}[0, t_1])(t_1), \tilde{a}[t_1, t_2])(t_2) = \Phi(\Phi(x_I, \tilde{a}[0, t_1])(t_1), \tilde{a}'[t_1, t'_2])(t'_2), \quad (12)$$

which implies that the new control sequence $\tilde{a}[0, t_1]$, $\tilde{a}[t_1, t'_2]$, and $\tilde{a}[t_2, t_F]$ arrives at x_G at time $t_f - (t_2 - t'_2)$. Fresh collision checking is needed for the trajectory from $\Phi(x_I, \tilde{a}[0, t_1])(t_1)$ to $\Phi(\Phi(x_I, \tilde{a}[0, t_1]), \tilde{a}'[t_1, t'_2])(t'_2)$.

The general template for *iterative bang-bang optimization* is:

1) Choose t_1 and t_2 according to a random or deterministic rule.
2) Attempt to replace $\tilde{a}[t_1, t_2]$ with the bang-bang alternative $\tilde{a}'[t_1, t'_2]$. If the result is collision free, then update $\tilde{a}$ with the modified control.
3) Go to Step 1, unless a termination criterion is met based on the number of iterations without any significant time reduction.

The rule of choosing t_1 and t_2 should produce a dense sequence of intervals in the following sense: the points of the form $(t_1, t_2) \in \mathbb{R}^2$ must be dense in the triangular region satisfying $0 \leq t_1 \leq t_2 \leq t_F$ (this ignores the fact that t_F decreases in each iteration, and such out-of-bounds intervals can be rejected in the analysis). A simple but effective rule is to first pick t_1 and t_2 uniformly at random. If $t_1 < t_2$, then replace $\tilde{a}[t_1, t_2]$. Otherwise, toss an unbiased coin to replace either $\tilde{a}[0, t_2]$ or $\tilde{a}[t_1, t_F]$. This extra consideration over purely random pairs (e.g., as in [11]) helps focus on the ends. Alternatively, t_1 and t_2 could be picked according to deterministic sequences to ensure convergence.

The termination criterion could be based on a hard limit on the number of iterations, or failure statistics (for example, no significant improvement more than $\epsilon > 0$ has occurred in the past 50 iterations). Note that the approach is not a variational optimization as in a gradient descent in trajectory space [2]; it more resembles path shortening for basic path planning (called *shortcutting* in [9], [25]). Thus, local time-optimality is gradually reached in the sense that the solution cannot be further improved by replacing trajectory segments with time-optimal alternatives, but it is not equivalent to a time optimum in the sense of local perturbations in trajectory space.

C. Basic path planning with bang-bang state-space lifting

Consider taking the output of a basic path planning, lifting it into the state space using bang-bang control, and then applying the bang-bang optimization method of Section IV-B. Suppose we are given a kinodynamic planning problem for nD double integrators for which x_I and x_G are both at rest (zero velocity). Thus, $x_I = (q_I, \mathbf{0})$ and $x_G = (q_G, \mathbf{0})$. The first step is to compute a piecewise-linear path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$, in which $\mathcal{C}_{free}$ is the projection of X_{free} onto the configuration space. This could, for example, be computed by RRT-Connect [18], but the particular planner is unimportant.

We then introduce the *bang-bang transform*, described here for $\mathbb{R}^n$ and $a_{max} = -a_{min} = 1$ (it easily generalizes; see also [11]). For each vertex q along the path τ, extend it to $x = (q, \mathbf{0}) \in X_{free}$. For each edge between consecutive vertices, q, q', execute a bang-bang control; we require that it is constrained to the edge and steers from $(q, \mathbf{0})$ to $(q', \mathbf{0})$. Let $v = q' - q$, normalized as $\hat{v} = v / \|v\|$. Let $s = \max_i(|\hat{v}_i|)$. Let $a_i = \hat{v}_i / s$ and $t = \sqrt{s\|v\|}$. The bang-bang control is

[1] The restrictions will be closed intervals that allow single-point overlaps, but this will not affect the resulting trajectories.

$((a,t),(-a,t))$. This transform is applied to each edge of the path and the resulting controls are concatenated.

The following propositions support the approach of lifting any piecewise-linear collision-free path (which are the typical output of sampling-based planners) into the state space via the bang-bang transform.

Proposition 4 *The bang-bang transform of a path is time-optimal. Furthermore,* $\Phi(x_I, \tilde{a}) = x_G$ *and the resulting trajectory is collision free.*

Proof: Assuming that there are no spurious vertices (lying in the interior of a linear segment), the system must come to rest at each vertex. Thus, applying the time-optimal control from rest to rest over each edge yields a time-optimal control for the whole path. The bang-bang transform is merely a consequence of Proposition 1, applied to the simpler rest-to-rest case. Regarding collision, for each segment, the accelerations yield velocities parallel to it; thus, the path traversed is the edge itself, which is already known to be collision free. Furthermore, this implies that x_G is reached after the full control $\tilde{a}$ is applied. ■

Proposition 5 *Using the bang-bang transform, any piecewise-linear solution to Problem 1 can be converted via the bang-bang transform into a corresponding solution to Problem 2, restricted to double integrators and rest-to-rest.*

Proof: This is a direct consequence of Proposition 4 due to the preservation of the collision-free and goal reachability properties of the bang-bang transform. ■

V. EXPERIMENTS

The algorithms were implemented in Python 3.9.5 on a Windows 10 PC with an AMD Ryzen 7 5800X CPU and 32GB 3200MHz CL16 RAM. Naive methods were used for nearest neighbor searching and collision detection because they are not critical to the experimental analysis. All results are shown in a high-resolution video available at `http://lavalle.pl/videos/IROS23.mp4`.

A. Kinodynamic planning for a 2D vehicle (4D state space)

These examples use a four-dimensional state space corresponding to a 2D workspace in which a planar vehicle moves with double integrator dynamics. Let $a_{max} = -a_{min} = 1$.

Figure 4 compares the new BB-RRT method (Figure 3) to the original kinodynamic RRT-Bi [21] and RRT-Connect [18] on the 2D projection that ignores velocities and dynamics. The state space X is $[-400, 400]^2 \times [-10, 10]^2$. Statistics are reported in Figure 5. The initial and goal states were at rest. The ρ_1 metric introduced in Section IV-A was used. The S_a and S_b sets were approximated by placing new RRT nodes along long edges for every 12 collision checks (see Section 5.5.2 of [20]). The BB-RRT is about 1564 times faster on average than the RRT-Bi. RRT-Connect is even faster, but it only constructs paths on the 2D configuration space, and

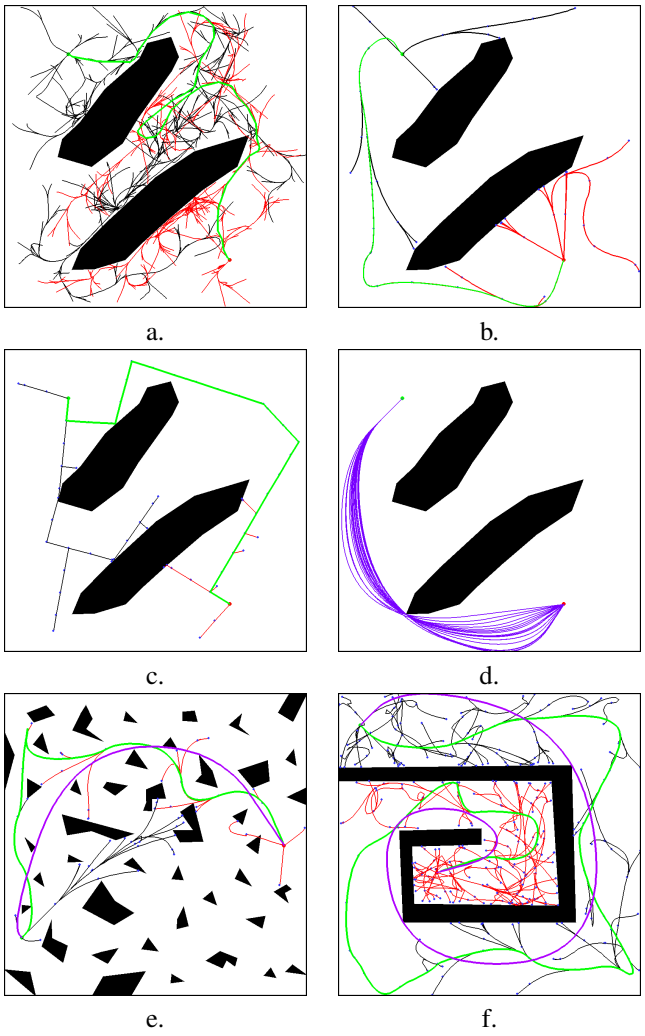


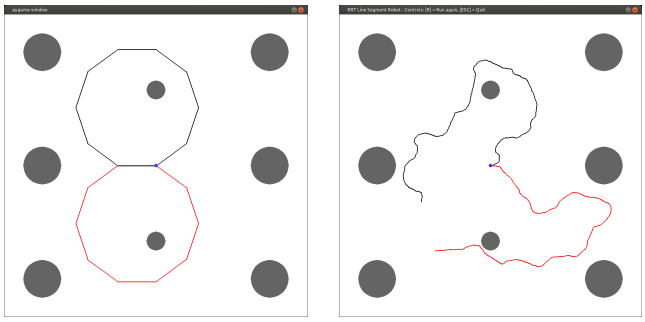
Fig. 4. a) Original kinodynamic RRT-Bi [21], b) The proposed BB-RRT, c) RRT-Connect [18] applied to the 2D projection, d) multiple bang-bang optimizations of a planned path, e & f) two more BB-RRT examples with BB-optimized paths (purple).

Method	RunTime	Nodes	ColChecks	TrajTime
RRT-Bi	27.013	1589.1	4364.8	311.53
BB-RRT	0.017276	57.772	599.48	126.99
RRT-Con	0.004738	60.663	713.25	n/a
BB-Opt	0.021547	n/a	3291.3	72.452

Fig. 5. For each method, the execution time (second), number of RRT nodes, number of collision checks, and trajectory execution time (seconds) are reported (where applicable). All numbers are calculated as averages over 1000 runs.

the number of collision checks is comparable. Figure 4.d shows 50 results for the bang-bang optimizer of Section IV-B, applied to the same initial path; computation times are fairly consistent across runs and problems, depending mainly on path length and collision detector cost.

The BB-RRT has the advantage, much like RRT-Connect, in that there are no parameters to tune. RRT-Bi has parameters for the step size, the set of actions, and the connection distance (the trees do not exactly meet). For the example in Figure 4.a, we used 24 constant acceleration actions, $\Delta t = 5$, and connection distances of $\Delta q = 5$ and $\Delta \dot{q} = 2$;



a. b.

Fig. 6. BB-optimization applied to a planar manipulator with dynamics.

in the weighted-Euclidean metric, the velocity components were weighted 17.32 times more than the configuration components. Figures 4.e and 4.f show two more examples under the same conditions, for which BB-RRT took on average 0.0368s and 0.4072s, respectively, over 1000 runs. The speedup factors over RRT-Bi were 837.6 and 368.5. Again, RRT-Connect on the 2D projection was faster, by factors 6.65 and 12.5, respectively. Original and bang-bang optimized paths are shown green and purple, respectively.

B. Bang-bang optimization for a planar manipulator

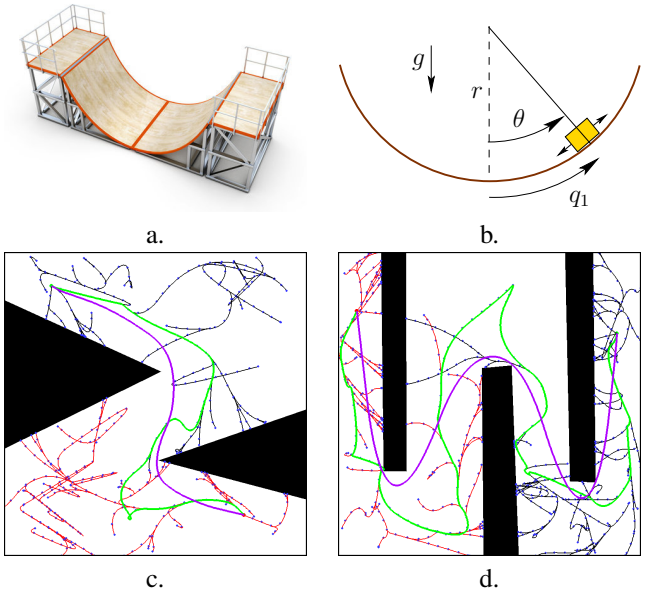
Figure 6 depicts additional experiments, performed for an n-link, fixed-base planar manipulator, modeled as a kinematic chain of line segments of equal length. Again, assume $a_{max} = -a_{min} = 1$. The initial and goal configurations form a regular polygon, as shown in Figure 6.a for $n = 10$ links. Paths were initially computed using RRT-Connect on $\mathcal{C}$ and then lifted into X using the bang-bang transform of Section IV-C. Each joint has limits $\pm\pi$ and is modeled as a double integrator. The BB-Optimization method was applied to computed paths from $n = 10$ (dimension of X is 20) up to $n = 1000$ (dimension of X is 2000). Figure 6.b shows configurations in the RRTs that were grown from initial and goal configurations, respectively. Average running times (10 runs) to fully converge are 1.63s for 10 links, 1.72s for 20 links, 3.75s for 50 links, 20.75s for 100 links, and 1123.35s for 1000 links (rapid increases due to dimension were caused by a naive quadratic-time implementation of the waiting method, rather than the $O(n \lg n)$ method presented in Section III-B). Termination was reached if 200 iterations were attempted with no more than 0.1s reduction in trajectory time; the most dramatic reductions occur in the first few iterations. In a typical run for 100 links, the trajectory execution time was reduced from 57.95s to 8.46s.

C. Beyond pure double integrator dynamics

As a step toward investigating bang-bang boosting of more general, stabilizable systems, suppose that the planar vehicle from Section V-A is instead placed on the interior surface of a level, cylindrical tube of radius r (Figures 7.a-b). The q_1 coordinate dynamics resemble that of an actuated pendulum:

$$\ddot{q}_1 = r\ddot{\theta} = u_1 - g\sin\theta. \tag{13}$$

The q_2 coordinate is the position along the tube in the direction of its central axis, with dynamics $\ddot{q}_2 = u_2$. Assume



a. b. c. d.

Fig. 7. a) Consider a moving vehicle on a cylindrical surface, b) the q_1 coordinate behaves like an actuated pendulum, c & d) computed examples (horizontal and vertical axes correspond to q_1 and q_2, respectively).

$u_1, u_2 \in [-1, 1]$. If $|g\sin\theta| < 1$ for θ, then the system is stabilizable, as defined in Section II. The interval of allowable accelerations $\ddot{q}_1$ becomes $A(\theta) = [-1-g\sin\theta, 1-g\sin\theta]$. To generate a bang-bang trajectory from some θ_I to θ_G, we restrict the system to a double integrator in which a_{min} and a_{max} are set to the minimum and maximum of $A(\theta_I) \cap A(\theta_G)$. We also test and reject any generated bang-bang trajectory for which $\ddot{q}_1 \notin A(\theta)$ at any time.

Two computed examples of both BB-RRT planning and bang-bang optimization are shown in Figures 7.c-d, in which the vehicle must go from rest-to-rest along the curved surface; a top-down view is given by unrolling the cylinder. The state space X is the same as in Section V-A, $r = 300$, and $g = 1$; note that the slope $\theta = q_1/r$ along the left and right edges reaches $\pm 4/3$ radians (76.394 degrees). The allowable horizontal accelerations at these boundaries are approximately $[-0.028, 1.972]$ and $[-1.972, 0.028]$, respectively (substantially shifted from $[-1, 1]$). The computation times averaged over 1000 runs were 0.02963s and 1.1184s, respectively. We also ran 1000 experiments on the geometry of the problem in Figure 4.e, but instead using the tube model, and the resulting average computation time was 0.09594s (approximately 2.61 times slower than for the level-surface case). In general, the planning and optimization methods easily overcame the challenges due to the non-double integrator model.

VI. DISCUSSION

We have proposed, analyzed, and implemented methods that accelerate planning performance and optimize solutions. The key is a steering method that quickly computes bang-bang time-optimal controls using exact, parabolic solutions. Although the study has been limited to RRTs, we expect it could enhance other sampling-based planning methods that rely on distance metrics or steering, such as *probabilistic*

roadmaps [1], [17] or *expansive space trees* [14]. One of the key observations of our experiments is that plan-and-optimize is superior when applicable: It is more reliable to explore the C-space first, lift the solution into the state space, and then use bang-bang optimization. However, this option applies only for rest-to-rest problems; for more general problems, a bang-bang enhanced RRT could be applied to bring each of x_I and x_G to zero velocity by biasing samples to the $(q, \mathbf{0})$ plane.

The encouraging results of this paper lead naturally to many new questions and further studies. The implementation focused mainly on n-double-integrator dynamics; however, with the vehicle-in-the-tube results from Section V-C, we have easily extended it for acceleration bounds that vary with state. This opens exciting directions of research to adapt the method to many more classes of stabilizable systems. Another important direction is to develop bang-bang boosted versions of asymptotically optimal planners, such as RRT* [16] and SST* [23]; this would enable stronger comparisons to the plan-and-optimize approach, both in computation time and solution quality. Also, improvements can be made to the iterative bang-bang optimization through strategic interval selection. Finally, efficient nearest-neighbor algorithms should be developed for the bang-bang metric over a tree of parabolic arcs (analogous to [31]).

ACKNOWLEDGMENTS: This work was supported by a European Research Council Advanced Grant (ERC AdG, ILLUSIVE: Foundations of Perception Engineering, 101020977), Academy of Finland (PERCEPT 322637, CHiMP 342556), and Business Finland (HUMOR 3656/31/2019). We thank Dmitry Berenson, Oren Salzman, Kalle Timperi, and Dmitry Yershov for helpful discussions.

REFERENCES

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Workshop on Algorithmic Foundations of Robotics*, pages 155–168, 1998.

[2] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March-April 1998.

[3] H.-T. L. Chiang, J. Hsu, Marek M. Fiser, L. Tapia, and A. Faust. RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies. *IEEE Robotics and Automation Letters*, 4(4):4298–4305, 2019.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Ed.* Springer-Verlag, Berlin, 2000.

[6] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40:1048–66, November 1993.

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

[8] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, December 2005.

[9] R. Geraerts and M. H. Overmars. Creating high-quality paths for motion planning. *The International Journal of Robotics Research*, 26(8):845–863, 2007.

[10] E. Glassman and R. Tedrake. A quadratic regulator-based heuristic for rapidly exploring state space. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 5021–5028, 2010.

[11] K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *Proceedings IEEE International Conference on Robotics and Automation*, 2010.

[12] E. Heiden, L. Palmieri, S. Koenig, K. O. Arras, and G. S. Sukhatme. Gradient-informed path smoothing for wheeled mobile robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1710–1717, 2018.

[13] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden. Time-optimal trajectories for a robotic manipulator: A provably good approximation algorithm. In *Proc. IEEE International Conference on Robotics & Automation*, pages 150–155, Cincinnati, OH, 1990.

[14] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal Computational Geometry & Applications*, 4:495–512, 1999.

[15] S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control*, pages 7681–7687, 2010.

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

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

[18] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[19] T. Kunz and M. Stilman. Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[20] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://lavalle.pl/planning/.

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

[22] Y. Li and K. E. Bekris. Learning approximate cost-to-go metrics to improve sampling-based motion planning. In *2011 IEEE International Conference on Robotics and Automation*, May 2011.

[23] Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016.

[24] D. Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Princeton, NJ, 2012.

[25] J. Luo and K. Hauser. An empirical study of optimal motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[26] L. Palmieri and K. O. Arras. Distance metric learning for RRT-based motion planning with constant-time inference. In *Proc. IEEE International Conference on Robotics and Automation*, pages 637–643. IEEE, 2015.

[27] A. Perez, R. Platt Jr., G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. LQR-RRT* : Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE International Conference on Robotics and Automation*, 2012.

[28] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *L. S. Pontryagin Selected Works, Volume 4: The Mathematical Theory of Optimal Processes*. Gordon and Breach, Montreux, Switzerland, 1986.

[29] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini. Sampling-based optimal kinodynamic planning with motion primitives. *Autonomous Robots*, 43(7):1715–1732, Oct 2019.

[30] E. Schmerling, L. Janson, and M. Pavone. Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2574–2581, 2015.

[31] V. Varricchio, B. Paden, D. Yershov, and E. Frazzoli. Efficient nearest-neighbor search for dynamical systems with nonholonomic constraints. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2016.

[32] D. Webb and J. van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Proceedings IEEE International Conference on Robotics and Automation*, 2013.

[33] W. J. Wolfslag, M. Bharatheesha, T. M. Moerland, and M. Wisse. RRT-CoLearn: Towards kinodynamic planning without numerical trajectory optimization. *IEEE Robotics and Automation Letters*, 3(3):1655–1662, 2018.