amcs

ITERATIVE METHODS FOR EFFICIENT SAMPLING–BASED OPTIMAL MOTION PLANNING OF NONLINEAR SYSTEMS

JUNG-SU HA [a], HAN-LIM CHOI [b,*], JEONG HWAN JEON [c]

[a]Mechanical Engineering Research Institute, Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong, Deajeon 34141, Republic of Korea
e-mail: `wjdtn1404@kaist.ac.kr`

[b]Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong, Deajeon 34141, Republic of Korea
e-mail: `hanilmc@kaist.ac.kr`

[c]nuTonomy Inc.
1 Broadway, Cambridge, MA 02142, USA
e-mail: `jhjeon@alum.mit.edu`

This paper extends the RRT* algorithm, a recently developed but widely used sampling based optimal motion planner, in order to effectively handle nonlinear kinodynamic constraints. Nonlinearity in kinodynamic differential constraints often leads to difficulties in choosing an appropriate distance metric and in computing optimized trajectory segments in tree construction. To tackle these two difficulties, this work adopts the affine quadratic regulator-based pseudo-metric as the distance measure and utilizes iterative two-point boundary value problem solvers to compute the optimized segments. The proposed extension then preserves the inherent asymptotic optimality of the RRT* framework, while efficiently handling a variety of kinodynamic constraints. Three numerical case studies validate the applicability of the proposed method.

Keywords: optimal motion planning, sampling-based algorithm, nonlinear dynamics.

1. Introduction

Robotic motion planning designs a trajectory of robot states from a given initial state to a specified goal state through a complex configuration space. While motion planning algorithms can be categorized into two groups: combinatorial and sampling-based approaches (LaValle, 2011), the latter (such as the probabilistic road map (PRM) and rapidly exploring random tree (RRT) algorithms) have been successful in practice because their computational advantage over combinatorial methods allows the handling of complex planning environments (Pepy *et al.*, 2009). In particular, the rapidly exploring random tree star (RRT*) algorithm proposed by Karaman and Frazzoli (2011b) is one of the most influential algorithms of this type, because it guarantees probabilistic completeness and asymptotic optimality at the same time. In other words, RRT*

guarantees that, if the planning problem is feasible, the probability of the algorithm failing to find a solution drops to zero as the number of iterations increases, and that the solution asymptotically approaches the optimal solution; this can be utilized as an "anytime" algorithm that finds a feasible trajectory quickly and refines the solution within the allowed computation time. In addition, RRT* inherits the key advantage of RRT: it explores the unexplored search space rapidly (LaValle, 1998). Due to these advantages, the algorithm has been successfully extended to many applications such as (stochastic) optimal control problems and differential games for various systems (Karaman and Frazzoli, 2011a; Huynh *et al.*, 2012; 2014).

When the objective of planning is to minimize a given cost function defined by the state and control trajectory under system dynamics constraints (Szynkiewicz and Błaszczyk, 2011), the motion planning problem is called *optimal kinodynamic*

---
*Corresponding author

motion planning. The problem is challenging because the resulting trajectory should not only satisfy system dynamics but should also lie over a highly non-convex state space because of the obstacle field. Due to the aforementioned properties, RRT* can provide a good framework for an optimal kinodynamic motion planner, supposing the following two issues are appropriately addressed. First, the distance metric should be able to take into account kinodynamic constraints of the problem. RRT-based algorithms take advantage of the Voronoi bias for rapid exploration of the state space: with a wrong distance metric, the configuration space may not be effectively explored. Second, there should be a way to construct an optimal trajectory segment under kinodynamic constraints for a given cost form, because the RRT* algorithm improves the quality of the solution by refining the segments of the trajectory so that the solution asymptotically converges to the optimal one.

There have been many attempts to handle the kinodynamic planning problem in the framework of RRT* by tackling the aforementioned two issues in certain ways. The minimum-time/length planning for holonomic and non-holonomic vehicles (Karaman and Frazzoli, 2010; 2013; Karaman *et al.*, 2013) was first addressed in the RRT* framework; a method tailored to high-speed off-road vehicles taking tight turns (Jeon *et al.*, 2011) was also proposed. Several recent studies were devoted to dealing with kinodynamic constraints in the form of linear differential constraints (Webb and van den Berg, 2013; Perez *et al.*, 2012; Goretkin *et al.*, 2013); these works in particular proposed to adopt optimal control theory for linear systems for cost functions of certain linear- (Perez *et al.*, 2012; Goretkin *et al.*, 2013) or affine-quadratic regulator (Webb and van den Berg, 2013) (LQR or AQR) types. Despite these recent examples of progress, the question of a systematic and efficient method to handle generic nonlinear dynamics, which inevitably involves computation of two-point boundary value problem (TPBVP) solutions in the RRT* process, remains unsettled.

The main contribution of this work is to present a methodology that effectively handles nonlinear dynamics in the framework of RRT*. The methodology determines an optimal trajectory for an affine-quadratic cost functional under nonlinear differential constraints while allowing rapid exploration of the state space. To do so, we adopt the AQR-based pseudo-metric from Glassman and Tedrake (2010) as well as Webb and van den Berg (2013) as an approximation of the optimum distance under nonlinear differential constraints, and two iterative methods are presented to efficiently solve the associated TPBVPs. The proposed extension of RRT* preserves asymptotic optimality of the original RRT*, while taking into account a variety of kinodynamic constraints. Three numerical case studies are presented to demonstrate the applicability of the proposed methodology. While one of the two iterative methods in this paper was first introduced in our earlier work (Ha *et al.*, 2013), this article includes a more extended description of the methodology, in particular proposing one more iterative algorithm, as well as more diverse/extensive numerical case studies.

2. Problem definition

A kinodynamic motion planning problem is defined for a dynamical system as follows:

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

where x denotes the state of the system defined over the state space $\chi \subset \mathbb{R}^n$ and u denotes the control input defined over the control input space $U \subset \mathbb{R}^m$. Let $\chi_{\text{obs}} \subset \chi$ and $\chi_{\text{goal}} \subset \chi$ be the obstacle region and the goal region that the system tries to avoid and to reach, respectively. Then, the feasible state and input spaces are given by $\chi_{\text{free}} \subset \chi \setminus \chi_{\text{obs}}$ and $U_{\text{free}} \subset U$, respectively.

The trajectory is represented as $\pi = (x(\cdot), u(\cdot), \tau)$, where τ is the arrival time at the goal region, $u : [0, \tau] \to U$, $x : [0, \tau] \to \chi$ are the control input and the corresponding state along the trajectory. The trajectory, π_{free}, for a given initial state x_{init} is called feasible if it does not cross the obstacle region and eventually achieves the goal region while satisfying the system dynamics (1), i.e., $\pi_{\text{free}} = (x(\cdot), u(\cdot), \tau)$, where $u : [0, \tau] \to U_{\text{free}}$, $x : [0, \tau] \to \chi_{\text{free}}$, $x(0) = x_{\text{init}}$ and $x(\tau) \in \chi_{\text{goal}}$.

In order to evaluate a given trajectory π, the following form of the cost functional is considered in this paper:

$$c(\pi) = \int_0^\tau \left[1 + \frac{1}{2} u(t)^T R u(t) \right] \, \mathrm{d}t. \tag{2}$$

The above cost functional denotes the trade-off between the arrival time of a trajectory and the expanded control effort. R is a user-defined value; the cost function penalizes more for the trajectory spending large control effort than for late arrival time as R is larger. This type of cost functional is widely used for kinodynamic planning problems (Glassman and Tedrake, 2010; Webb and van den Berg, 2013).

Finally, the problem is defined as follows.

Problem 1. *(Optimal kinodynamic motion planning)* Given χ_{free}, x_{init} and χ_{goal}, find a minimum cost trajectory $\pi^* = (x^*(\cdot), u^*(\cdot), \tau^*)$ such that $\pi^* = \arg\min_{\pi \in \Pi_{\text{free}}} c(\pi)$, where Π_{free} denotes a set of feasible paths.

3. Background

3.1. RRT* algorithm. This section summarizes the RRT* algorithm (Karaman and Frazzoli, 2011b),

Algorithm 1. RRT* algorithm.

```
 1: (V, E) ← ({x_init}, ∅);
 2: for i = 1,,N do
 3:     x_rand ← SAMPLING(χ_free);
 4:     x_nearest ← NEAREST(V, x_rand);
 5:     x_new ← STEER(x_nearest, x_rand);
 6:     π_new ← TPBVPSOLVER(x_nearest, x_new);
 7:     if OBSTACLEFREE(π_new) then
 8:         X_near_b ← NEARBACKWARD(V, x_new);
 9:         X_near_f ← NEARFORWARD(V, x_new);
10:         c_min ← COST(x_nearest) + c(π_new);
11:         for x_near ∈ X_near_b do
12:             π'_new ← TPBVPSOLVER(x_near, x_new);
13:             c' ← COST(x_near) + c(π'_new);
14:             if OBSTACLEFREE(π'_new) ∧ c' < c_min then
15:                 c_min ← c'; π_new ← π'_new;
16:             end if
17:         end for
18:         V ← V ∪ x_new; E ← E ∪ π_new;
19:         for x_near ∈ X_near_f do
20:             π'_near ← TPBVPSOLVER(x_new, x_near);
21:             c' ← COST(x_new) + c(π'_near);
22:             if OBSTACLEFREE(π'_near) ∧ c' < COST(x_near)
                    then
23:                 E ← (E \ π_near) ∪ π'_near; {r}eplace existing
                        edge
24:             end if
25:         end for
26:     end if
27: end for
28:
29: return T ← (V, E);
```

on which this paper builds an extension to deal with kinodynamic constraints. RRT* is a sampling-based algorithm that incrementally builds an optimal trajectory from a specified initial state x_{init} to a specified goal region X_{goal}. The overall structure of the algorithm is summarized in Algorithm 1.

At each iteration, the algorithm randomly samples a state x_{rand} from χ_{free}; it then finds the *nearest* node $x_{nearest}$ in the tree to this sampled state (lines 3–4). Then, the algorithm steers the system toward x_{rand} to determine an x_{new} that is closest to x_{rand} and that stays within a certain specified distance from $x_{nearest}$; then, it adds x_{new} to the set of vertices V if the trajectory from $x_{nearest}$ to x_{new} is obstacle-free (lines 5–7 and 18). Next, the best parent node for x_{new} is chosen from near nodes in the tree so that the trajectory from the parent to x_{new} is obstacle-free and of minimum cost (lines 8, 10–17). After adding the trajectory segment from the parent to x_{new} (line 18), the algorithm rewires the near nodes in the tree so that the forward paths from x_{new} are of minimum cost (lines 9,

19–25); then, the algorithm proceeds to the next iteration.

Below we provide a more detailed description of the key functions of the algorithm:

- SAMPLING(χ_{free}): randomly samples a state from χ_{free}.

- NEAREST(V, x): finds the nearest node from x among nodes in the tree V under a given distance metric $dist(x_1, x_2)$ representing distance from x_1 to x_2.

- STEER(x, y): returns a new state $z \in \chi$ such that z is closest to y among all candidates:

$$\text{STEER}(x, y) := \arg\min_{z \in B^+_{x,\eta}} \text{dist}(z, y),$$

  where $B^+_{x,\eta} \equiv \{z \in \chi \mid \text{dist}(x, z) \leq \eta\}$, with η representing the maximum length of an one-step trajectory forward.

- TPBVPSOLVER(x_0, x_1): returns the optimal trajectory from x_0 to x_1, *without* considering the obstacles.

- OBSTACLEFREE(π): returns indication of whether or not the trajectory π overlaps with the obstacle region χ_{obs}.

- NEARBACKWARD(V, x), NEARFORWARD(V, x): return the set of nodes in V that are within the distance of $r_{|V|}$ from/to x, respectively. In other words,

$$\text{NEARBACKWARD}(V, x) := \{v \in V \mid v \in B^-_{x, r_{|V|}}\},$$

$$\text{NEARFORWARD}(V, x) := \{v \in V \mid v \in B^+_{x, r_{|V|}}\},$$

  where $B^-_{x, r_{|V|}} \equiv \{z \in \chi \mid \text{dist}(z, x) \leq r_{|V|}\}$, $B^+_{x, r_{|V|}} \equiv \{z \in \chi \mid \text{dist}(x, z) \leq r_{|V|}\}$, and $|V|$ denotes the number of nodes in the tree.

  Also, $r_{|V|}$ needs to be chosen such that a ball of volume $\gamma \frac{\log |V|}{|V|}$ is contained by $B^-_{x, r_{|V|}}$ and $B^+_{x, r_{|V|}}$ with γ large enough. For example, with the Euclidean distance metric this can be defined as $r_{|V|} = \min\{(\frac{\gamma}{\zeta_d} \frac{\log n}{n})^{1/d}, \eta\}$ with the constant γ (Karaman and Frazzoli, 2011b); ζ_d is the volume of the unit ball in $\mathbb{R}^d$.

- COST(x): returns the cost-to-come for node x from the initial state.

- PARENT(x): returns a pointer to the parent node of x.

- $c(\pi)$: returns the cost of trajectory π defined by (2).

Note, specifically, that when the cost of a trajectory is given by the path length and no kinodynamic constraint is involved (as was in the first version presented by Karaman and Frazzoli (2011b)), the Euclidean distance can be used as the distance metric $\text{dist}(x_1, x_2)$. Thus, NEARBACKWARD(V, x) becomes identical to NEARFORWARD(V, x) because $\text{dist}(x, z) = \text{dist}(z, x)$ and TPBVPSOLVER(x_0, x_1) simply returns a straight line from x_0 to x_1.

3.2. Optimal control with affine dynamics. This section presents a procedure to compute the optimal solution for an affine system with an affine-quadratic cost functional which, later in this paper, will be taken advantage of for quantification of the distance metric for generic nonlinear systems. Consider an affine system

$$\dot{x}(t) = Ax(t) + Bu(t) + c \qquad (3)$$

and the performance index

$$J = \int_0^\tau \left[1 + \frac{1}{2}u(t)^T Ru(t) \right] \, \mathrm{d}t, \qquad (4)$$

with the boundary conditions

$$x(0) = x_0, \qquad x(\tau) = x_1 \qquad (5)$$

and *free* final time τ.

The Hamiltonian is given by

$$H = 1 + \frac{1}{2}u(t)^T Ru(t) + \lambda(t)^T (Ax(t) + Bu(t) + c).$$

The minimum principle yields that the optimal control takes the form of

$$u(t) = -R^{-1}B^T \lambda(t),$$

with a reduced Hamiltonian system (Lewis *et al.*, 1995):

$$\dot{x}(t) = Ax(t) - BR^{-1}B^T \lambda(t) + c,$$
$$-\dot{\lambda}(t) = A^T \lambda(t), \qquad \text{(BVP1)}$$

where $x(0) = x_0$, $x(\tau) = x_1$. The solution of (BVP1) with the boundary conditions (5) is the optimal trajectory from x_0 to x_1, supposing that the final time τ is given.

Note that, for a given τ, the terminal values of $x(t)$ and $\lambda(t)$ in (BVP1) can be expressed as

$$x(\tau) = x_1, \quad \lambda(\tau) = -G(\tau)^{-1}(x_1 - x_h(\tau)), \qquad (6)$$

where a homogeneous solution of (3), $x_h(\tau)$, and the weighted continuous reachability Gramian, $G(\tau)$, are the final values of the following initial value problem:

$$\dot{x}_h(t) = Ax_h(t) + c,$$
$$\dot{G}(t) = AG(t) + G(t)A^T + BR^{-1}B^T,$$
$$x_h(0) = x_0, \quad G(0) = 0. \qquad \text{(IVP1)}$$

With (6), the optimal trajectory can be obtained by integrating (BVP1) backward.

For a given final time τ, the performance index of the optimal trajectory can be written as

$$C(\tau) = \tau + \frac{1}{2}(x_1 - x_h(\tau))^T G(\tau)^{-1}(x_1 - x_h(\tau)) \quad (7)$$

and the optimal final time can be expressed as

$$\tau^* = \arg\min_{\tau \geq 0} C(\tau). \qquad (8)$$

Equation (7) implies that $C(\tau) \geq \tau$ since $G(\tau)$ is positive (semi-)definite. Therefore, τ^* can be computed by calculating $C(\tau)$ with increasing τ until τ equals the incumbent best cost $\widetilde{C}(\tau) \triangleq \min_{t \in [0,\tau]} C(t)$. With this optimal final time τ^*, the optimal cost can be obtained as

$$C^* = C(\tau^*). \qquad (9)$$

4. Efficient distance metric and steering method

To take advantage of the explorative property of RRT*, it is crucial to use an appropriate distance metric in the process. The Euclidean distance, which cannot consider the system dynamics, is certainly not a valid option in the kinodynamic motion planning problem—for example, it would fail to find the nearest node and thus would not be able to steer toward the sample node. Hence, a distance metric that appropriately represents the degree of closeness while taking into account the dynamic constraints and the underlying cost measure needs to be defined/quantified for a kinodynamic version of RRT*.

An AQR-based pseudo-metric was first proposed by Glassman and Tedrake (2010) as a distance metric for kinodynamic planning to consider first-order linear dynamics. In this work, an AQR-based pseudo-metric is adopted as an approximate distance measure for problems with nonlinear differential constraints. For kinodynamic planning with the cost functional (2) and the dynamic constraint (1), the distance from x_0 to x_1 is computed as

$$\text{dist}(x_0, x_1) = C^* \equiv \min_{\tau \geq 0} C(\tau), \qquad (10)$$

where $C(\tau)$ is calculated from (7) with

$$A = \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}, u=\hat{u}}, \quad B = \left.\frac{\partial f}{\partial u}\right|_{x=\hat{x}, u=\hat{u}},$$

where $\hat{x}$ and $\hat{u}$ represent the linearization points that are set to the initial (or the final) state, i.e., $\hat{x} = x_0$ (or x_1), and $\hat{u} = 0$, in the framework of RRT*. In other words, the distance from x_0 to x_1 is approximated as the cost of the optimal control problem for a linearized system with the same cost functional and boundary conditions. Note that

this pseudo-metric is not a metric (or a distance function) because it is not symmetric, i.e., the cost from x_0 to x_1 and the cost from x_1 to x_0 are not necessarily the same.

In this work, an AQR-based pseudo-metric is adapted as a distance metric in the RRT* framework. The exactness of a metric is related to the property of rapid exploration of the RRT* algorithm: for a randomly chosen sample, the algorithm finds the nearest node with the NEAREST procedure and expands the tree toward the sample using the STEER procedure. Calculation of the exact metric between the two states is equivalent to solving a nonlinear optimal control problem, which is computationally expensive; the algorithm needs to solve such optimal control problems for every pair of states, from the sample state to the nodes in the tree and vice versa. Although the AQR-based pseudo-metric does not take into account the nonlinear dynamics, it can measure the metric for all pairs of states by simple integration and produces a much more exact degree of closeness than is possible when using the Euclidean distance. Details of the implementation are given in Table 1.

5. Efficient solver for the two-point boundary-value problem

As mentioned previously, the TPBVPSOLVER(x_0, x_1) function returns the optimal trajectory from x_0 to x_1. A straight-line trajectory, which is optimal for the problem without dynamic constraints, cannot be a valid solution in general, because it would be not only suboptimal but also likely to violate the kinodynamic constraints. Therefore, this section derives a two-point boundary value problem (TPBVP) involving nonlinear differential constraints and presents methods to compute the solution of this TPBVP.

Let us consider the optimal control problem (OCP) with nonlinear system dynamics as in (1) to minimize the cost functional in (4), where the boundary conditions are given as $x(0) = x_0$, $x(\tau) = x_1$ with free final time τ. The Hamiltonian of this OCP is defined as

$$H(x(t), u(t), \lambda(t))$$
$$= 1 + \frac{1}{2}u(t)^T R u(t) + \lambda(t)^T f(x(t), u(t)).$$

From the minimum principle, we have

$$\left(\frac{\partial H}{\partial u}\right)^T = Ru(t) + \left(\frac{\partial f}{\partial u}\right)^T \lambda(t) = 0,$$

which allows the optimal control to be expressed in terms of $x(t)$ and $\lambda(t)$:

$$u(t) = h(x(t), \lambda(t)). \tag{11}$$

Thus, a system of differential equations for the state $x(t)$

and the costate $\lambda(t)$ is obtained:

$$\dot{x}(t) = f(x(t), h(x(t), \lambda(t))),$$
$$-\dot{\lambda}(t) = \left(\frac{\partial H}{\partial x}\right)^T = \left(\frac{\partial f}{\partial x}\right)^T \lambda(t), \qquad \text{(BVP2)}$$

with boundary conditions $x(0) = x_0$, $x(\tau) = x_1$. The system of differential equations in (BVP2) is nonlinear in general and has boundary conditions at the initial and final time; thus, it is called a nonlinear two-point boundary value problem (TPBVP). An analytic solution to a nonlinear TPBVP is generally unavailable because it is not only nonlinear but also has the boundary conditions that are split in two time instances; numerical solution schemes have often been adopted to solve this problem.

In the present work, two types of numerical *iterative* approaches are presented: methods based on successive approximation (SA) and the variation of extremals (VE). These two approaches find the solution of a nonlinear TPBVP by successively solving a sequence of more tractable problems; in the SA-based method, a sequence of linear TPBVPs are solved and, in the VE approach, a sequence of nonlinear initial value problems (IVPs) are iteratively solved. The main concept of the methods has already been presented, i.e., to solve a *free final-state and fixed final-time* problem (Tang, 2005; Kirk, 2012); however, this work proposes variants that can handle *fixed final-state and free final-time* problems for implementation in the RRT* framework.

In an iterative method, an initial guess of the solution is necessary and often substantially affects the convergence of the solution. In the proposed RRT* extension, the optimal trajectory with the linearized dynamics can be a good choice of the initial condition, particularly because such a trajectory is already available in the RRT* process by calculating the AQR-based distance metric using NEARBACKWARD or NEARFORWARD.

5.1. Successive approximation. Let $\hat{x}$ be a linearization point that is set to be the initial or the final value of a trajectory segment in the RRT* implementation; $\hat{u} = 0$ is also the corresponding linearization point. By splitting the dynamic equation in (1) into the linearized and remaining parts around $\hat{x}$ and $\hat{u}$,

$$\dot{x}(t) = Ax(t) + Bu(t) + g(x(t), u(t)),$$

where

$$A \triangleq \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}, u=\hat{u}}, \quad B \triangleq \left.\frac{\partial f}{\partial u}\right|_{x=\hat{x}, u=\hat{u}}$$

and

$$g(x(t), u(t)) \triangleq f(x(t), u(t)) - Ax(t) - Bu(t),$$

Table 1. Implementation of the AQR metric in the RRT* framework.

$x_{\text{nearest}} \leftarrow \text{NEAREST}(V, x_{\text{rand}})$	• The system dynamics are linearized at x_{rand}. Since x_{rand} is the final state, $x_h(t)$ and $G(t)$ are integrated from $t = 0, x_h(0) = x_{\text{rand}}, G(0) = 0$ in the backward direction, i.e., $t < 0$. • With integration, the cost from the i-th node $v_i \in V$ at the time $-t$ is calculated as $C_i(-t) = -t - \frac{1}{2}(v_i - x_h(t))^T G(t)^{-1}(v_i - x_h(t))$ while the minimum cost is saved as a distance, $d_i = \min_{t<0} C_i(-t)$.
$X_{\text{near\_b}} \leftarrow \text{NEARBACKWARD}(V, x_{\text{new}})$	• The procedure is similar to NEAREST. • The system dynamics are linearized at x_{new}. $x_h(t)$ and $G(t)$ are integrated from $t = 0, x_h(0) = x_{\text{new}}, G(0) = 0$ for the backward direction, i.e., $t < 0$. • With integration, the cost for the time $-t$ and for each $v_i \in V$, the corresponding cost is calculated while the minimum cost is saved as a distance, $d_i = \min_{t<0} C_i(-t)$. • The integration stops when $-t \geq r_{|V|}$ and the set of backward near nodes, $X_{\text{near\_b}} \leftarrow \{v_i \in V \mid d_i \leq r_{|V|}\}$, is returned.
$X_{\text{near\_f}} \leftarrow \text{NEARFORWARD}(V, x_{\text{new}})$	• The procedure is exactly the same as NEARBACKWARD except for the direction of integration, $t > 0$.
$x_{\text{new}} \leftarrow \text{STEER}(x_{\text{nearest}}, x_{\text{rand}})$	• The system dynamics are linearized at x_{nearest}. • Then, the optimal trajectory is calculated by the procedure in Section 3.2 for the linearized system. • If the resulting cost is less than η, x_{rand} is returned as a new node. • Otherwise, it returns x' such that x' is in the trajectory and the cost to x' from x_{nearest} is η.

the optimal control and the reduced Hamiltonian system are expressed as

$$u(t) = -R^{-1}B^T\lambda(t) - R^{-1}g_u^T\lambda(t) \qquad (12)$$

and

$$\dot{x}(t) = Ax(t) - BR^{-1}B^T\lambda(t) - BR^{-1}g_u^T\lambda(t) \\ + g(x(t), u(\lambda(t))),$$
$$-\dot{\lambda}(t) = A^T\lambda(t) + g_x^T\lambda(t), \qquad (13)$$

respectively. The boundary conditions are given by

$$x(0) = x_0, \quad x(\tau) = x_1$$

and the final time τ is *free*. The TPBVP in (13) is still nonlinear, as $g_u^T\lambda$, g and $g_x^T\lambda$ in (13) are *not* necessarily linear with respect to x or λ.

Consider the following sequence of TPBVPs:

$$\dot{x}^{(k)}(t) = Ax^{(k)}(t) - BR^{-1}B^T\lambda^{(k)}(t) \\ \qquad - BR^{-1}g_u^{(k-1)T}\lambda^{(k-1)}(t) + g^{(k-1)}, \quad (14)$$
$$-\dot{\lambda}^{(k)}(t) = A^T\lambda^{(k)}(t) + g_x^{(k-1)T}\lambda^{(k-1)}(t), \qquad (15)$$

with boundary conditions $x^{(k)}(0) = x_0$, $x^{(k)}(\tau) = x_1$, where $g^{(k)} \equiv g(x^{(k)}, u^{(k)})$, $g_x^{(k)} \equiv g_x(x^{(k)}, u^{(k)})$ and $g_u^{(k)} \equiv g_u(x^{(k)}, u^{(k)})$.[1] Note that this system of differential equations is linear with respect to $x^{(k)}(t)$ and $\lambda^{(k)}(t)$ for given $x^{(k-1)}(t)$ and $\lambda^{(k-1)}(t)$. It can be converted into an initial value problem as follows. From (15), $\lambda^{(k)}(t)$ can be expressed as

$$\lambda^{(k)}(t) = e^{A^T(\tau-t)}\lambda^{(k)}(\tau) + \lambda_p^{(k)}(t), \qquad (16)$$

where $\lambda_p^{(k)}(t)$ is a solution of

$$\dot{\lambda}_p^{(k)}(t) = -A^T\lambda_p^{(k)}(t) - g_x^{(k-1)T}\lambda^{(k-1)}(t), \qquad (17)$$

---

[1]Here, subscripts denote partial derivatives, $g_x \equiv \frac{\partial g}{\partial x}$, $g_u \equiv \frac{\partial g}{\partial u}$.

with terminal condition $\lambda_p^{(k)}(\tau) = 0$. Plugging (16) into (14) yields

$$
\begin{aligned}
\dot{x}^{(k)}(t) \\
= Ax^{(k)}(t) + g^{(k-1)} - BR^{-1}B^T e^{A^T(T-t)}\lambda(\tau) \\
- BR^{-1}B^T \lambda_p^{(k)}(t) - BR^{-1}g_u^{(k-1)T}\lambda^{(k-1)}(t).
\end{aligned}
$$
(18)

Then, the final state $x^{(k)}(\tau)$ is expressed as

$$
x^{(k)}(\tau) = x_h^{(k)}(\tau) - G(\tau)\lambda^{(k)}(\tau), \tag{19}
$$

where $x_h(\tau)$ and $G(\tau)$ are the solution of

$$
\begin{aligned}
\dot{x}_h^{(k)}(t) = Ax_h^{(k)}(t) - BR^{-1}B^T\lambda_p^{(k)}(t) \\
- BR^{-1}g_u^{(k-1)T}\lambda^{(k-1)}(t) + g^{(k-1)}, \quad (20)
\end{aligned}
$$

$$
\dot{G}(t) = AG(t) + G(t)A^T + BR^{-1}B^T, \tag{21}
$$

$$
x_h^{(k)}(0) = x_0, \quad G(0) = 0.
$$

Using (19) and $x^{(k)}(\tau) = x_1$, the final costate value can be obtained:

$$
\lambda^{(k)}(\tau) = -G(\tau)^{-1}(x_1 - x_h^{(k)}(\tau)). \tag{22}
$$

Finally, $x^{(k)}(t)$ and $\lambda^{(k)}(t)$ for $t \in [0, \tau]$ are calculated by backward integration of the following differential equation:

$$
\begin{aligned}
\begin{bmatrix} \dot{x}^{(k)}(t) \\ \dot{\lambda}^{(k)}(t) \end{bmatrix} &= \begin{bmatrix} A & -BR^{-1}B^T \\ 0 & -A^T \end{bmatrix} \begin{bmatrix} x^{(k)}(t) \\ \lambda^{(k)}(t) \end{bmatrix} \\
&+ \begin{bmatrix} -BR^{-1}g_u^{(k-1)T}\lambda^{(k-1)}(t) + g^{(k-1)} \\ -g_x^{(k-1)T}\lambda^{(k-1)}(t) \end{bmatrix},
\end{aligned}
\tag{23}
$$

with given boundary values $x^{(k)}(\tau)$ and $\lambda^{(k)}(\tau)$. The optimal control $u^{(k)}(t)$ can also be computed accordingly:

$$
u^{(k)}(t) = -R^{-1}B^T\lambda^{(k)}(t) - R^{-1}g_u^{(k-1)T}\lambda^{(k-1)}(t). \tag{24}
$$

The optimal final time can be found numerically using a gradient descent scheme. The derivative of J for the trajectory at the k-th iteration with respect to the final time is given as

$$
\begin{aligned}
\left[\frac{dJ}{d\tau}\right]^{(k)} &= 1 - \frac{1}{2}\lambda^{(k)}(\tau)^T BR^{-1}B^T\lambda^{(k)}(\tau) \\
&+ \frac{1}{2}\lambda^{(k-1)}(\tau)^T g_u^{(k-1)} R^{-1} g_u^{(k-1)T}\lambda^{(k-1)}(\tau) \\
&+ \lambda^{(k)}(\tau)^T(Ax_1 + g^{(k)}).
\end{aligned}
\tag{25}
$$

Then, a gradient descent-based update rule of the final time is given as

$$
\tau^{(k+1)} = \tau^{(k)} - \eta\left[\frac{dJ}{d\tau}\right]^{(k)}. \tag{26}
$$

With the process described thus far, the TPBVP solver for a nonlinear system based on successive approximation can be summarized as Algorithm 2.

Algorithm 2. $\pi \leftarrow$ SA-based TPBVPsolver(x_0, x_1).

1: Initialize $x^{(0)}$, $\lambda^{(0)}$, $\tau^{(0)}$ and set $k = 0$ {Section 3.2}

2: **repeat**
3:     $k = k + 1$;
4:     Update $\tau^{(k)}$ {Eqns. (25)–(26)}
5:     Get $x^{(k)}(t)$ and $\lambda^{(k)}(t)$ {Eqns. (16)–(23)}
6: **until** Converge
7: Calculate $u(t)$ {Eqn. (24)}
8: **return** $\pi \leftarrow (x^{(k)}(\cdot), u(\cdot), \tau^{(k)})$

5.2. Variation of extremals. The VE approach is a technique employed to successively find the initial value of the costate $\lambda(0)$ using the Newton–Rapson method (Kirk, 2012). The method has been well established, but its implementation in sampling-based planning is not trivial.

Suppose that the initial value of the costate $\lambda(0)$ and the final value of the state $x(\tau)$ are related via a nonlinear function:

$$
x(\tau) = F(\lambda(0)).
$$

The function F takes an initial costate λ_0 as its argument and outputs a final state from (BVP2). Though an analytical expression for this function is unavailable, the initial costate $\lambda(0)$ that leads to $x(\tau) = x_1$ can be successively found as follows:

$$
\begin{aligned}
\lambda^{(k+1)}(0) &= \lambda^{(k)}(0) \\
&- [P_x(\lambda^{(k)}(0), \tau)]^{-1}(x^{(k)}(\tau) - x_1),
\end{aligned}
\tag{27}
$$

where $P_x(\lambda^{(k)}(0), t)$ is called the *state influence function matrix* and is a matrix of partial derivatives of the components of $x(t)$, evaluated at $\lambda^{(k)}(0)$,

$$
P_x(\lambda^{(k)}(0), t) \equiv \begin{bmatrix} \frac{\partial x_1(t)}{\partial \lambda_1(0)} & \cdots & \frac{\partial x_1(t)}{\partial \lambda_n(0)} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n(t)}{\partial \lambda_1(0)} & \cdots & \frac{\partial x_n(t)}{\partial \lambda_n(0)} \end{bmatrix}_{\lambda^{(k)}(0)}.
$$

Similarly, the *costate influence function matrix* is defined and given by

$$
P_\lambda(\lambda^{(k)}(0), t) \equiv \begin{bmatrix} \frac{\partial \lambda_1(t)}{\partial \lambda_1(0)} & \cdots & \frac{\partial \lambda_1(t)}{\partial \lambda_n(0)} \\ \vdots & \ddots & \vdots \\ \frac{\partial \lambda_n(t)}{\partial \lambda_1(0)} & \cdots & \frac{\partial \lambda_n(t)}{\partial \lambda_n(0)} \end{bmatrix}_{\lambda^{(k)}(0)}.
$$

The dynamics of the *influence function matrices* can be derived from the state and costate equations

$$
\dot{x}(t) = \left(\frac{\partial H}{\partial \lambda}\right)^T, \quad \dot{\lambda}(t) = -\left(\frac{\partial H}{\partial x}\right)^T.
$$

Taking the partial derivatives of these equations with respect to the initial value of the costate yields

$$\frac{\partial}{\partial \lambda(0)}\left[\dot{x}(t)\right] = \frac{\partial}{\partial \lambda(0)}\left[\left(\frac{\partial H}{\partial \lambda}\right)^T\right],$$

$$\frac{\partial}{\partial \lambda(0)}\left[\dot{\lambda}(t)\right] = \frac{\partial}{\partial \lambda(0)}\left[-\left(\frac{\partial H}{\partial x}\right)^T\right]. \quad (28)$$

With the assumption that $\frac{\partial}{\partial \lambda(0)}[\dot{x}(t)]$, $\frac{\partial}{\partial \lambda(0)}[\dot{\lambda}(t)]$ are continuous and by using the chain rule on the right hand side of (28), the differential equations of the *influence function matrices* are obtained as

$$\begin{bmatrix} \dot{P}_x(\lambda^{(k)}(0), t) \\ \dot{P}_\lambda(\lambda^{(k)}(0), t) \end{bmatrix}$$
$$= \begin{bmatrix} \frac{\partial^2 H}{\partial x \partial \lambda} & \frac{\partial^2 H}{\partial \lambda^2} \\ -\frac{\partial^2 H}{\partial x^2} & -\frac{\partial^2 H}{\partial \lambda \partial x} \end{bmatrix} \begin{bmatrix} P_x(\lambda^{(k)}(0), t) \\ P_\lambda(\lambda^{(k)}(0), t) \end{bmatrix}, \quad (29)$$

where $P_x(\lambda^{(k)}(0), 0) = 0, P_\lambda(\lambda^{(k)}(0), 0) = I$. The matrix components on the right-hand side are evaluated at $x(t)$ and $\lambda(t)$ by integrating (BVP2) together.

On the other hand, since the final time of the problem is free, the Hamiltonian should vanish:

$$H(x(\tau), \lambda(\tau), \tau) = 0.$$

The update rule for the initial costate value and the final time can then be obtained as

$$\begin{bmatrix} \lambda^{(k+1)}(0) \\ \tau^{(k+1)} \end{bmatrix} = \begin{bmatrix} \lambda^{(k)}(0) \\ \tau^{(k)} \end{bmatrix}$$
$$- \begin{bmatrix} P_x(\lambda^{(k)}(0), \tau^{(k)}) & \frac{\partial x^{(k)}(\tau^{(k)})}{\partial \tau} \\ \frac{\partial H^{(k)}(\tau^{(k)})}{\partial \lambda(0)} & \frac{\partial H^{(k)}(\tau^{(k)})}{\partial \tau} \end{bmatrix}^{-1}$$
$$\times \begin{bmatrix} x^{(k)}(\tau^{(k)}) - x_1 \\ H^{(k)}(\tau^{(k)}) \end{bmatrix}, \quad (30)$$

where entries of the matrix on the right hand side can be obtained from

$$\frac{\partial H(\tau^{(k)})}{\partial \lambda(0)} = \frac{\partial H}{\partial x}\frac{dx(\tau^{(k)})}{d\lambda(0)} + \frac{\partial H}{\partial \lambda}\frac{d\lambda(\tau^{(k)})}{d\lambda(0)}$$
$$= -\dot{\lambda}(\tau^{(k)})^T P_\lambda(\tau^{(k)}) + \dot{x}(\tau^{(k)})^T P_x(\tau^{(k)}), \quad (31)$$

$$\frac{\partial x^{(k)}(\tau^{(k)})}{\partial \tau^{(k)}} = f(x^{(k)}(\tau^{(k)}), h(x^{(k)}(\tau^{(k)}), \lambda^{(k)}(\tau^{(k)}))), \quad (32)$$

$$\frac{\partial H(\tau^{(k)})}{\partial \tau^{(k)}} = \frac{\partial H}{\partial x}\frac{dx(\tau^{(k)})}{d\tau^{(k)}} + \frac{\partial H}{\partial \lambda}\frac{d\lambda(\tau^{(k)})}{d\tau^{(k)}}$$
$$= 0. \quad (33)$$

Finally, the TPBVP solver with the variation of extremals-based method can be summarized as Algorithm 3.

Algorithm 3. $\pi \leftarrow$ VE-based TPBVPsolver(x_0, x_1).

1: Initialize $\lambda^{(0)}(0)$, $\tau^{(0)}$ and set $k = 0$ {Section 3.2}
2: **repeat**
3:    $k = k + 1$;
4:    Integrate $x^{(k)}, \lambda^{(k)}, P_x, P_\lambda$ {(BVP2), Eqn. (29)}
5:    Calculate $\lambda^{(k+1)}(0), \tau^{(k+1)}$ {Eqns. (30)–(33)}
6: **until** Converge
7: Calculate $u(t)$ {Eqn. (11)}
8: **return** $\pi \leftarrow (x^{(k)}(\cdot), u(\cdot), \tau^{(k)})$

5.3. Discussion. Both the methods presented in this section find the solution to (BVP2) iteratively. At each iteration, the SA-based method computes the optimal solution for a linear approximation of the original problem linearized at the solution of the previous iteration, while the VE-based method updates the estimate of the initial costate and the final time. It should be pointed out that these solvers guarantee local (not global) optimality, because the underlying TPBVP is derived as a necessary (not sufficient) condition for optimality. However, this locality would not have a significant impact on the overall motion planning, since the TPBVP itself is posed for optimally linking a small segment of the overall plan. With a similar philosophy, the planning algorithm can drop local trajectory segments whose control sequence lie outside of input space, U_{free}, so as not to make the overall plan violate input feasibility. One limitation is that, since the AQR-based metric is valid and the Gramian, $G(\tau)$, is nonsingluar only for controllable linear systems, the proposed methods cannot make a plan on the part of the state space of a nonlinear system where its linearization is uncontrollable even though the original system is controllable. Also, there can be a singularity issue in (27) or (29). We utilized pseudo-inverse in simulations to avoid such an issue and found no crucial problem in practice, but if it is the case, a pair of states that cause singularity can be discarded. Another option that can be considered is to add a scaled identity matrix, μI, with a small μ to the matrix being inverted; this regularized update method is also used in the Levenberg–Marquardt algorithm to prevent the singularity issue when the Newton–Rapson method suffers from singular Hessian matrix.

Compared to an existing approach that treated *linearized* dynamics (Webb and van den Berg, 2013) and thus did not require an iterative process for solving a TPBVP, the proposed methods need to solve a few additional first-order ODEs until reachingconvergence in computing a single optimal trajectory segment in the RRT* framework. However, the proposed schemes can take into account nonlinear kinodynamic constraints, which was not accurately realized in the previous work.

Note also that the two methods exhibit different characteristics from the computational point of view.

First, the storage requirement of the VE-based method has an advantage over that of the SA-based method: at every iteration, the VE solver only needs to store 2 $(n \times n)$ matrices, $P_x(t)$ and $P_\lambda(t)$, while the SA solver needs to store the trajectories, $\lambda_p^{(k)}(t)$, $x^{(k)}(t)$ and $\lambda^{(k)}(t)$ for $t \in [0, \tau]$ and a $(n \times n)$ matrix, $G(t)$. On the other hand, the SA solver demands integration of $4n$ first-order ODEs in (17), (20) and (23) at each iteration (n^2 first-order ODEs in (21) need to be integrated only in the first iteration and then reused), while the VE solver needs to integrate $2n(n + 1)$ first-order ODEs in (29) and (BVP2) in each iteration. Thus, assuming that both methods converge after a similar number of iterations, the VE-based approach exhibits better memory complexity while the SA-based method gives better time complexity in general. However, detailed convergence characteristics such as convergence time might be problem-dependent.

6. Numerical examples

6.1. Pendulum swing-up.
The control objective of the pendulum swing-up problem is to move the pendulum into an upright position from its downward stable equilibrium. The dynamics of the system are given as

$$I\ddot{\theta} + b\dot{\theta} + mgl_c \sin\theta = u, \tag{34}$$

where we used $I = m = l_c = 1$, $g = 9.81$ and $b = 0.1$ for the simulation. The angular position and velocity, $x(t) = [\theta(t)\ \dot{\theta}(t)]^T$, are the state variables; the torque, $u(t)$, is the control input to the system. The cost of the trajectory is given by (2). The initial and goal states are given as $x_{\text{init}} = [0\ 0]^T$, $x_{\text{goal}} = \{[\pi\ 0]^T,\ [-\pi\ 0]^T\}$. To comparatively investigate the capability of RRT* variants in handling nonlinearity in dynamics, the proposed iterative methods are compared with kinodynamic-RRT* (Webb and van den Berg, 2013), which takes into account the first-order Taylor approximation of the dynamics.

Figure 1 illustrates state trajectories in the phase plane; an open-loop trajectory and a closed-loop trajectory are depicted, as well as the planned trajectory. The open-loop trajectory is generated by using the planned control input trajectory $u(t)$ as a feedfoward control term, while the closed-loop trajectory is produced by adding an LQR trajectory stabilizer (Tedrake, 2009) that utilizes feedback control in the form of $u(t) - K\bar{x}$, with LQR gain K, where $\bar{x}$ is the deviation from the planned trajectory. It was found that both of the proposed iterative solvers produce very similar results; thus, only the result of the SA-based solver is presented herein.

The solid line represents the trajectories planned using the algorithm with the SA-based solver or with the solver for linearized dynamics; the dash-dotted and dashed lines denote the open-loop and closed-loop state trajectories, respectively. Note that as the

linearization is only valid around the nodes of the RRT* tree, the optimized trajectory segment obtained using the linearization-based scheme (Webb and van den Berg, 2013) is dynamically infeasible for the original (nonlinear) system as the state moves far away from the nodes. As shown in Fig. 1(b) and (d), the system may not follow the planned trajectory when using open-loop control or even when using a feedback trajectory stabilizer; while, as shown in Fig. 1(a) and (c), the trajectories planned using the algorithm with the iterative TPBVP solver are followed not only by feedback control but also by open-loop control.

The control input histories are shown in Fig. 2. The solid line represents the input from the planning algorithm and the dashed line denotes the input with feedback control implementation. When the planned trajectory is inconsistent with the real dynamics, a large amount of control effort is required for the feedback controller, which results in a cost increase. This can be observed in the case in which the linearization-based planner is used; note that the control effort for the proposed method is significantly smaller than that for the linearization-based method. Table 2 presents average costs of ten simulations for varying cost functions with the feedback controller; the 'planned cost' stands for the cost returned from the algorithm, and the 'executed cost' represents the cost that includes control effort from the trajectory stabilization feedback controller. It is shown that the cost achieved using the proposed method is smaller and more consistent with the planned cost.

6.2. Two-wheeled mobile robot.
The second example addresses the design of the trajectory of a two-wheeled mobile robot. The states and inputs of the system are $x = [p_x\ p_y\ \theta\ v\ w]^T$ and $u = [F_1\ F_2]^T$, where (p_x, p_y) and θ represent robot position and orientation, respectively, v, w denote the linear and angular velocity of the robot, and F_1, F_2 are the force from each wheel. The dynamic equation of the system is given as

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ w \\ F_1 + F_2 \\ F_1 - F_2 \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ w \\ u_1 + u_2 \\ u_1 - u_2 \end{bmatrix}.$$

Table 2. Average costs from ten simulations.

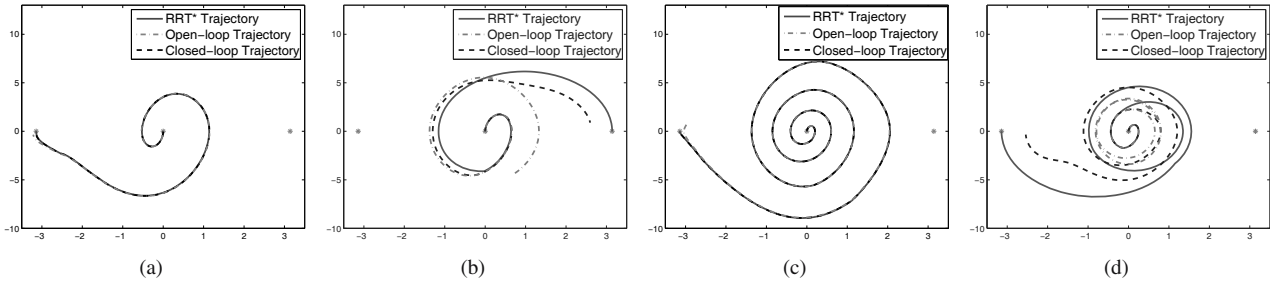|	SA-based		Linearization-based	
	Planned	Executed	Planned	Executed
$R = 1$	6.4621	6.4727	4.5122	8.4837
$R = 5$	18.9790	18.9929	9.7229	31.2797
$R = 10$	33.0323	33.7034	12.9783	55.9197
$R = 15$	52.8430	52.9865	15.1409	83.6147

Fig. 1. State trajectories for different TPBVP solvers with varying R for the pendulum swing up example: SA-based solver, $R = 1$ (a), solver for linearized dynamics, $R = 1$ (b), SA-based solver, $R = 10$ (c), solver for linearized dynamics, $R = 10$ (d).
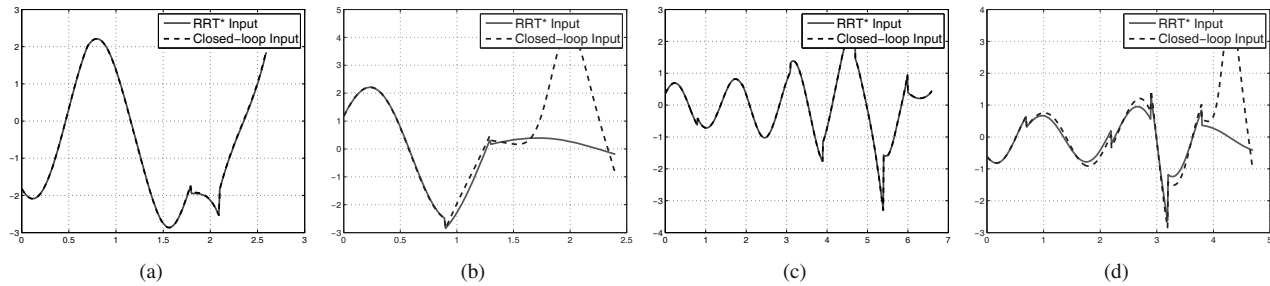

Fig. 2. Planned control input vs. executed (feedforward + feedback) control input containing feedback for different TPBVP solvers with varying R for the pendulum swing up example: SA-based solver, $R = 1$ (a), solver for linearized dynamics, $R = 1$ (b), SA-based solver, $R = 10$ (c), solver for linearized dynamics, $R = 10$ (d).

Table 3. Cost of the best trajectories in the tree out of 100 trials.

# of nodes	300	500	1000	3000	5000
# of feasible	81	100	100	100	100
Mean	inf	21.81	20.51	19.52	19.18
Variance	NaN	2.26	0.79	0.23	0.17

The initial and the goal states are given as $x(0) = [0.5 \, 0.5 \, \pi/4 \, 1 \, 0]^T$ and $x(\tau) = \{[p_x \, p_y \, \theta \, v \, w]^T | 23 \le p_x \le 24, \, 9 \le p_y \le 10, \, 0 \le \theta \le \pi/2, \, 0.8 \le v \le 1.2, \, -0.2 \le w \le 0.2\}$, respectively, with free final time τ. The cost functional is given in (2) with $R = 20 \, \mathrm{diag}(1,1)$.

As the two proposed iterative methods have already been shown to have similar levels of performance, only the result for the VE-based solver case is depicted in this example. Figure 3, by projecting the tree in the five-dimensional space onto the two-dimensional position space, $(p_x, p_y) \in \mathbb{R}^2$, shows the progression of the RRT* tree. The red star and the square represent the initial position and the goal region; the thick line represents the best trajectory found for the corresponding progression. Observe that the resulting trajectories smoothly connect the initial state and the goal state. Also, as the number of nodes in the tree increases, the tree fills up the feasible state space and finds a lower-cost trajectory.

Table 3 presents the results of a Monte-Carlo simulation with 100 trials; it shows the number of trials needed to find a feasible trajectory, and the mean and variance of the minimum costs in each iteration. It is found that the mean and variance of the cost decrease as the number of nodes increases, implying convergence to the same solution.

Finally, Fig. 4 depicts the resulting trajectories obtained using the VE-based solver and the same linearization-based solver that, was used in the first example (Webb and van den Berg, 2013). It is observed that with the solver for linearized dynamics, there is a portion of sideway-skid movement in the middle of the trajectory, which is dynamically infeasible for the robot. On the other hand, the result of the proposed solver shows feasible movement across the whole trajectory.

6.3. SCARA type robot arm. The third example involves generating a motion plan for a three-degree-of-freedom SCARA robot with two rotational joints (represented by θ_1 and θ_2), and one prismatic joint (θ_3), as shown in Fig. 5. The dynamics of the robot are given as

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + N(\theta, \dot{\theta}) = u,$$

where

$$M(\theta) = \begin{bmatrix} \alpha + \beta + 2\gamma \cos\theta_2 & \beta + \gamma \cos\theta_2 & 0 \\ \beta + \gamma \cos\theta_2 & \beta & 0 \\ 0 & 0 & m_3 \end{bmatrix},$$
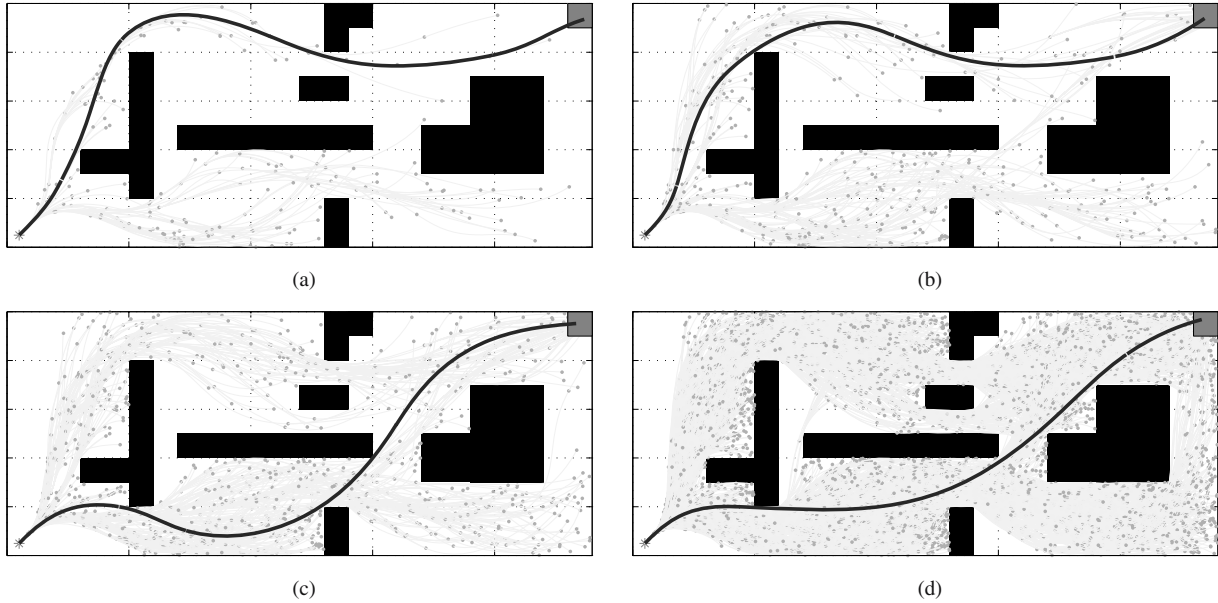
(a)

(b)

(c)

(d)

Fig. 3. Two-dimensional representation of evolution of the RRT* tree for the mobile robot example, when the number of nodes is 200 (a), 500 (b), 1000 (c) and 5000 (d).

$$C(\theta, \dot{\theta}) = \begin{bmatrix} -\gamma \sin\theta_2 \dot{\theta}_2 & -\gamma \sin\theta_2(\dot{\theta}_1 + \dot{\theta}_2) & 0 \\ \gamma \sin\theta_2 \dot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$N(\theta, \dot{\theta}) = \begin{bmatrix} 0 \\ 0 \\ m_3 g \end{bmatrix}$$

and

$$\alpha = I_{z1} + r_1^2 m_1 + l_1^2 m_2 + l_1^2 m_3,$$
$$\beta = I_{z2} + I_{z3} + l_2^2 m_3 + m_2 r_2^2,$$
$$\gamma = l_1 l_2 m_3 + l_1 m_2 r_2,$$

with m_1, m_2, m_3 and l_1, l_2 representing the masses and lengths of each link, respectively. Also, r_1 and r_2 denote the length between a joint axis and the center of mass of each link; I_{z1}, I_{z2} and I_{z3} stand for the moments of inertia about the corresponding rotation axes. The system has a six-dimensional state vector, $x = [\theta_1 \, \theta_2 \, \theta_3 \, \dot{\theta}_1 \, \dot{\theta}_2 \, \dot{\theta}_3]^T$, and three-dimensional control inputs, $u = [\tau_1 \, \tau_2 \, f_3]^T$ representing the torques of the two rotational joints and the force of the prismatic joint. For the simulation, we set the parameters as $I_{z1} = I_{z2} = 2 \times 10^{-2}$, $I_{z3} = 10^{-2}$, $m_1 = m_2 = 1$, $m_3 = 0.4$, $l_1 = l_2 = 1$, $r_1 = r_2 = 0.25$ and $g = 9.81$, respectively.

The problem is to find an optimal motion trajectory that leads to the end-effector of the robot reaching the goal region (shown as a circle in Figs. 6–8) from its initial state, $x(0) = [\pi/2 \, 0 \, 3 \, 0 \, 0 \, 0]^T$ (the corresponding end-effector position is shown as a star in Figs. 6–7), while avoiding collision with obstacles. The cost of the

trajectory takes the form of (2). With other parameters being fixed, to see the effect of the cost functional on the resulting trajectories, the solutions are obtained by varying the input penalty matrix, R. Two values of R are considered:

$$R = 0.05 \operatorname{diag}(1, 1, 0.5), \quad 0.05 \operatorname{diag}(0.5, 0.5, 10).$$

The proposed VE-based method is implemented to solve nonlinear TPBVPs in the process of RRT*. There are two homotopy classes for the solutions from the initial state to the final states: the end effector (a) goes over the wall or (b) makes a detour around the wall.

Figures 6–7 show the resulting motions of the robot when 5000 nodes are added to the tree for the two cases; the dashed line and the circle on the ground represent the end-effector trajectory and the goal region, respectively. For the first case ($R = 0.05 \operatorname{diag}(1, 1, 0.5)$), which has a smaller control penalty for joint 3, the resulting trajectory goes over the wall, as shown in Fig. 6; going over the wall by using a cheap control input of joint 3 leads to a lower cost in this case. On the other hand, in the second case ($R = 0.05 \operatorname{diag}(0.5, 0.5, 10)$), shown in Fig. 7, the proposed algorithm generates a trajectory that makes a detour because the control penalty for joint 3 is larger; going over the wall is so expensive in this case that the detouring trajectory is chosen as the best one.

Considering the control penalties for joints 1 and 2, the penalty is smaller in the second case than in the first one; thus, the duration of the trajectory has greater weight more in the second case. Figure 8 shows top-views of the SCARA robot motion at each time step in Figs. 6–7, as well as the corresponding control inputs; the motion
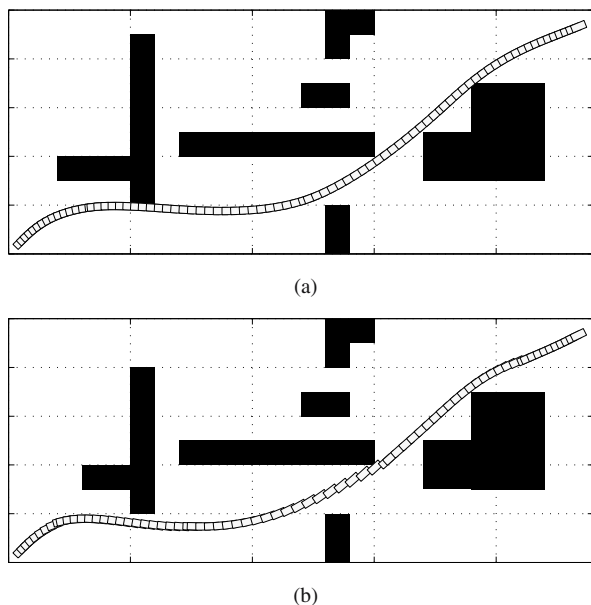
(a)

(b)

Fig. 4. Resulting trajectories with the proposed solver (a) and
the solver for linearized dynamics (b) for the two-
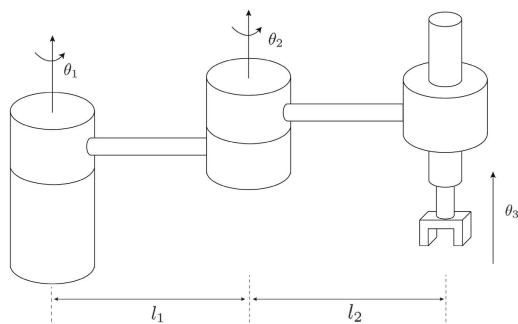wheeled mobile robot example.

Fig. 5. SCARA robot.

starts at the position colored in light-gray and ends at
the position colored in dark-gray. It can be seen that
the trajectory reaches the goal state more quickly in
the second case even though it requires a large amount
of input-energy to joints 1 and 2 (the arrival times of
the minimum-cost trajectory are 2.15 and 1.25 seconds,
respectively).

7. Concluding remarks

In this work, an extension of RRT* to handle nonlinear
kinodynamic differential constraints was proposed. In
order to tackle two caveats, choosing a valid distance
metric and solving two point boundary value problems
to compute an optimal trajectory segment, an affine
quadratic regulator (AQR)-based pseudo-metric was

adopted and two iterative methods were proposed. These
methods were tested on three numerical examples,
highlighting their capability of generating dynamically
feasible trajectories in various settings.

Even though the proposed methods focused on
being implemented in the RRT* framework, they are
expected to help other state-of-the-art motion planning
algorithms, like RRT# (Arslan and Tsiotras, 2013),
GR-FMTs (Jeon *et al.*, 2015), FMT* (Janson *et al.*,
2015), the path planner for multi vehicles (Klaučo
et al., 2016) or a manipulator (García-Rodríguez *et al.*,
2016; Rodríguez-Liñán *et al.*, 2017) to address nonlinear
dynamical systems.

References

Arslan, O. and Tsiotras, P. (2013). Use of relaxation methods
in sampling-based algorithms for optimal motion planning,
*IEEE International Conference on Robotics and Automa-
tion, Karlsruhe, Germany*, pp. 2421–2428.

García-Rodríguez, R., Segovia-Palacios, V., Parra-Vega, V.
and Villalva-Lucio, M. (2016). Dynamic optimal
grasping of a circular object with gravity using robotic
soft-fingertips, *International Journal of Applied Math-
ematics and Computer Science* **26**(2): 309–323, DOI:
10.1515/amcs-2016-0022.

Glassman, E. and Tedrake, R. (2010). A quadratic
regulator-based heuristic for rapidly exploring state space,
*IEEE International Conference on Robotics and Automa-
tion, Anchorage, AK, USA*, pp. 5021–5028.

Goretkin, G., Perez, A., Platt, R. and Konidaris, G. (2013).
Optimal sampling-based planning for linear-quadratic
kinodynamic systems, *IEEE International Conference
on Robotics and Automation, Karlsruhe, Germany*,
pp. 2429–2436.

Ha, J.-S., Lee, J.-J. and Choi, H.-L. (2013). A successive
approximation-based approach for optimal kinodynamic
motion planning with nonlinear differential constraints,
IEEE Conference on Decision and Control, Florence, Italy,
pp. 3623–3628.

Huynh, V.A., Karaman, S. and Frazzoli, E. (2012). An
incremental sampling-based algorithm for stochastic
optimal control, *IEEE International Conference on
Robotics and Automation, Minneapolis, MN, USA*,
pp. 2865–2872.

Huynh, V.A., Kogan, L. and Frazzoli, E. (2014). A Martingale
approach and time-consistent sampling-based algorithms
for risk management in stochastic optimal control, *IEEE
Conference on Decision and Control, Los Angeles, CA,
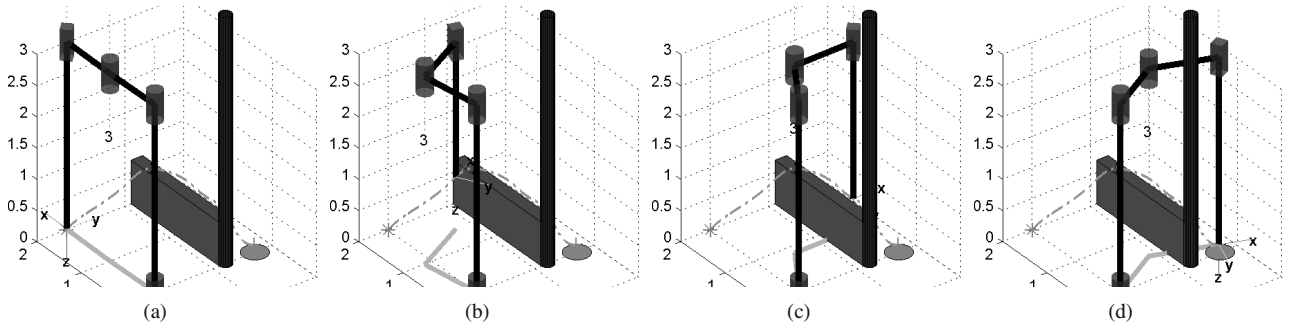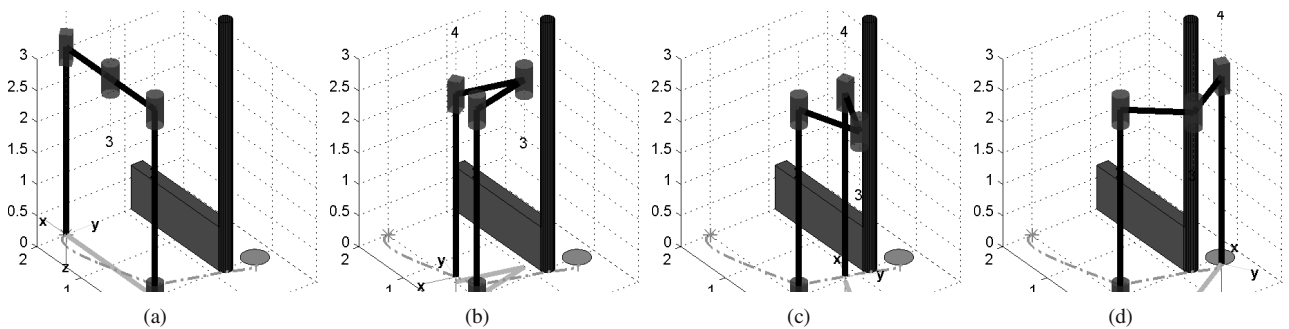USA*, pp. 1858–1865.

Fig. 6. Motion of the SCARA robot corresponding to the resulting state trajectory when 5000 nodes are added to the tree with $R = 0.05 \, \mathrm{diag}(1, 1, 0.5)$: $t = 0$ sec. (a), $t = 0.7$ sec. (b), $t = 1.45$ sec. (c), $t = 2.15$ sec. (d).

Fig. 7. Motion of the SCARA robot corresponding to the resulting state trajectory when 5000 nodes are added to the tree with $R = 0.05 \, \mathrm{diag}(0.5, 0.5, 10)$: $t = 0$ sec. (a), $t = 0.4$ sec. (b), $t = 0.85$ sec. (c), $t = 1.25$ sec. (d).
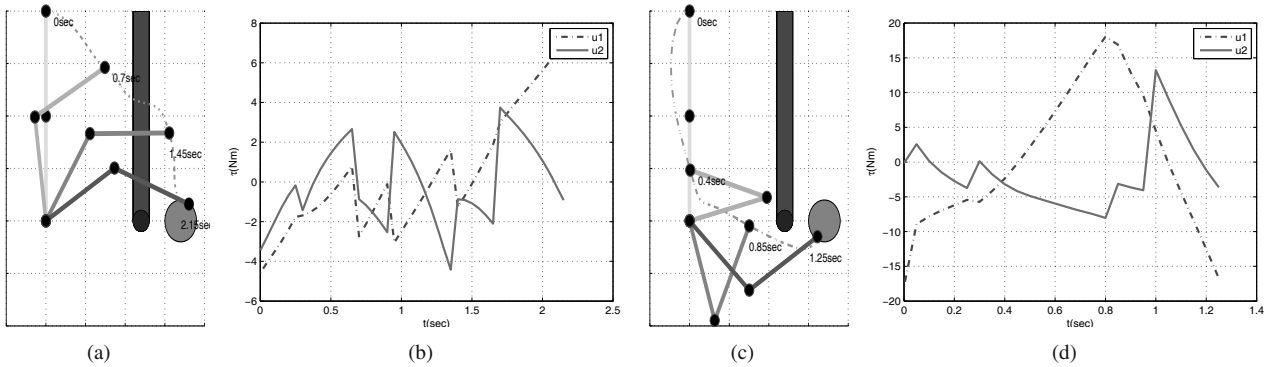

Fig. 8. Top-views of resulting motion and the corresponding inputs: case 1, top-view (a), case 1, input (b), case 2, top-view (c), case 2, input (d).

Janson, L., Schmerling, E., Clark, A. and Pavone, M. (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions, *The International Journal of Robotics Research* **34**(7): 883–921.

Jeon, J., Karaman, S. and Frazzoli, E. (2011). Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*, *IEEE Conference on Decision and Control, Orlando, FL, USA*, pp. 3276–3282.

Jeon, J., Karaman, S. and Frazzoli, E. (2015). Optimal sampling-based feedback motion trees among obstacles for controllable linear systems with linear constraints, *IEEE International Conference on Robotics and Automation, Seattle, WA, USA*, pp. 4195–4201.

Karaman, S. and Frazzoli, E. (2010). Optimal kinodynamic motion planning using incremental sampling-based methods, *IEEE Conference on Decision and Control, Atlanta, GA, USA*, pp. 7681–7687.

Karaman, S. and Frazzoli, E. (2011a). Incremental sampling-based algorithms for a class of pursuit-evasion games, *in* D. Hsu *et al.* (Eds.), *Algorithmic Foundations of Robotics IX*, Springer, Berlin/Heidelberg, pp. 71–87.

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

Karaman, S. and Frazzoli, E. (2013). Sampling-based optimal motion planning for non-holonomic dynamical systems,

IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, pp. 5041–5047.

Karaman, S., Walter, M.R., Perez, A., Frazzoli, E. and Teller, S. (2011). Anytime motion planning using the RRT*, *IEEE International Conference on Robotics and Automation, Shanghai, China*, pp. 1478–1483.

Kirk, D.E. (2012). *Optimal Control Theory: An Introduction*, Courier Corporation, Dover Publications, Inc. Mineola, NY.

Klaučo, M., Blažek, S. and Kvasnica, M. (2016). An optimal path planning problem for heterogeneous multi-vehicle systems, *International Journal of Applied Mathematics and Computer Science* **26**(2): 297–308, DOI: 10.1515/amcs-2016-0021.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning, *Technical Report 98-11,* Iowa State University, Ames, IO.

LaValle, S.M. (2011). Motion planning, *IEEE Robotics & Automation Magazine* **18**(1): 79–89.

Lewis, F.L., Vrabie, D. and Syrmos, V.L. (1995). *Optimal Control*, John Wiley & Sons, New York, NY.

Pepy, R., Kieffer, M. and Walter, E. (2009). Reliable robust path planning with application to mobile robots, *International Journal of Applied Mathematics and Computer Science* **19**(3): 413–424, DOI: 10.2478/v10006-009-0034-2.

Perez, A., Platt, R., Konidaris, G., Kaelbling, L. and Lozano-Perez, T. (2012). LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics, *IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA*, pp. 2537–2542.

Rodríguez-Liñán, M.C., Mendoza, M., Bonilla, I. and Chávez-Olivares, C.A. (2017). Saturating stiffness control of robot manipulators with bounded inputs, *International Journal of Applied Mathematics and Computer Science* **27**(1): 79–90, DOI: 10.1515/amcs-2017-0006.

Szynkiewicz, W. and Błaszczyk, J. (2011). Optimization-based approach to path planning for closed chain robot systems, *International Journal of Applied Mathematics and Computer Science* **21**(4): 659–670, DOI: 10.2478/v10006-011-0052-8.

Tang, G. (2005). Suboptimal control for nonlinear systems: A successive approximation approach, *Systems & Control Letters* **54**(5): 429–434.

Tedrake, R. (2009). Underactuated robotics: Learning, planning, and control for efficient and agile machines, *Course Notes for MIT 6.832*, MIT, Cambridge, MA, `http://ocw.mi t.edu/courses/electrical-engineering-a nd-computer-science/6-832-underactuate d-robotics-spring-2009/`.

Webb, D.J. and van den Berg, J. (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics, *2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, pp. 5054–5061.
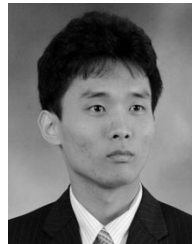
Jung-Su Ha is a post-doctoral researcher at the Mechanical Engineering Research Institute of the KAIST (Korea Advanced Institute of Science and Technology). He received his BS degree in electrical engineering from Hanyang University, Seoul, Korea, in 2011, his MS degree in electrical engineering from the KAIST, Daejeon, Korea, in 2013, and his PhD degree in aerospace engineering from the KAIST, in 2018. His research interests are in the area of stochastic optimal control and reinforcement learning problems in uncertain environments.

Han-Lim Choi is an associate professor of aerospace engineering at the KAIST (Korea Advanced Institute of Science and Technology). He received his BS and MS degrees in aerospace engineering from the KAIST, Daejeon, Korea, in 2000 and 2002, respectively, and his PhD degree in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2009. His current research interests include the planning and control of multi-agent systems, planning and control under uncertainty, resource management in radars, and Bayesian inference for large-scale systems. He (together with Dr. Jonathan P. How) is the recipient of the 2011 Automatica Applications Prize.

Jeong Hwan Jeon is a senior research scientist at nuTonomy. He received his BS degree in mechanical and aerospace engineering from Seoul National University, Seoul, Korea, in 2007, and his MS and PhD degrees in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2009 and 2015, respectively. His current research interests include algorithmic, computational, and control-theoretic approaches to decision making, planning and control architectures for autonomous systems, including self-driving cars.