Regionally Growing Random Trees: A synergistic motion planning and control algorithm for dynamic systems

Siamak G. Faal and Cagdas D. Onal

Abstract— **Nonlinearities, differential constraints, and input limitations preclude the use of regular feedback control algorithms in a range of complex dynamic systems. This article introduces the concept of Regionally Growing Random Trees (RGRT) as a powerful tool that synergistically combines motion planning and control tasks. RGRT is a forest of Dynamics-based Rapidly Expanding Trees (DRETs) that grow in the state-space of a dynamic system without requiring any distance function or explicit solutions of the differential equations of motion. The growth of multiple DRETs results in paths between the tree roots and forms a roadmap which is utilized in a planning algorithm to find a feasible path between a current state and a goal state. A path tracking algorithm is then used to convert the open-loop commands of the planner into a feedback controller, which provides robustness against disturbances and modeling errors. The RGRT motion planning and control scheme allows complete utilization (instead of avoidance) of system nonlinearities, which provides solutions for overcoming actuator constraints and eliminates the limitations imposed on the system by traditional feedback control approaches.**

I. Introduction

Although the inherent complexity of nonlinear systems complicate the controller design process, taking advantage of these nonlinearities provides more elegant solutions for the control problem. Some of the examples that have been discussed in the literature are: control of a 2-degree-of-freedom (DoF) pendulum with a passive shoulder joint known as Acrobot [1], [2]; control of a pendulum that is connected to an active cart (also known as cart-pole or inverted pendulum system) [3], [4]; and aggressive maneuvers of small-scale quad rotors [5], [6].

From the state-space perspective (for the sake of clarity, please refer to Section II for detailed definitions of the terms used in this article), a control action is a manipulation of state variables from a current state to a goal state through a specific path. Due to the range of possible control inputs, the path between the current and goal states is not unique and is subjected to constraints dictated by system dynamics. Moreover, for a generic nonlinear system, different control actions lead to diverse paths in state-space of the system, in which some may never reach the desired goal state. An example of such behavior is observed in a simple pendulum with input torque limitations, as explained in Section III-B.

Utilizing a search algorithm in the state-space may lead to solutions that overcome specific actuator limitations and allow taking advantage of nonlinearities to improve system performance. This type of control is widely observed in nature. For instance, when a cardinal bird attempts to perch on a tree branch, it shapes its wings to a parachute-like form to generate a rapid speed reduction. Unlike biological flyers, common aircraft controllers avoid high angles of attack and stall conditions due to invalidity of linear models in these nonlinear regions. Although linear models and approximations work fine for common industrial systems, modern robotic systems will require more agile and constrained maneuvers for state to state transitions, due to their high-dimensional state-space, complex and potentially underactuated dynamics, and interactions with the environment.

A possible method to determine control actions that satisfy differential and static constraints of the system is to formulate the control problem as a motion-planning problem. Two well known approaches that resemble this technique are: LQR trees [7], [8] and randomized kinodynamic planning [9]. An LQR tree is a tree of linear quadratic regulators (LQR) that are constructed by computing the corresponding stability regions of each regulator. The LQR nodes cover a controllable subset of the state-space to provide regions of attraction for the initial conditions of the system. LQR trees were used in the control of a fixed-wing glider to perform a bird-like perching maneuver [10], [11] and to stabilize an Acrobot in the upward configuration [2]. A similar approach is followed in [6] to demonstrate aggressive maneuvers with quad rotors. This method treats the control problem as a set of controllers that are derived based on system dynamics and each has a specific initial and goal region. These controllers are then used to control the system over a family of trajectories that are defined as a sequence of motions. To account for errors in the dynamic model, the controller for each trajectory segment is refined through successive trials. In this class of controllers, the state-space path is either predefined in forms of position and velocity profiles or it is a result of a controller that is active for that specific region of the state-space.

On the other hand, randomized kinodynamic planning approach [9] focuses on the construction of rapidly exploring random trees (RRTs) [12] in the state-space of the system. This approach allows the utilization of system dynamics to find feasible paths to the goal state. RRT [12] is a specific subcategory of rapidly exploring dense trees (RDT) [13]. RRT randomly chooses samples from the configuration space and seeks a path from the closest neighbor on the tree to the newly sampled point. This approach allows RRT to cover the space with a relatively uniform distribution [14] if a suitable sampling technique is used. Thus, the state-space application of RRT requires two main tools: a controller

Siamak G. Faal and Cagdas D. Onal are with the Soft Robotics Laboratory, Worcester Polytechnic Institute, Worcester, MA 01609, USA {sghorbanifaal, cdonal}@wpi.edu

that can manipulate any state of the system to the randomly chosen state; a method to define the distance between two states. These two requirements make it difficult to apply kinodynamic planning to systems with complex dynamics.

This article focuses on a different approach to address the problems associated with existing methods and proposes a synergistic algorithm that combines motion-planning and control parts into a unified problem. The proposed algorithm addresses: 1) Searching for a feasible state-space path between a current state and a goal state (or a goal region) by utilizing system dynamics and satisfying static and differential constraints; and 2) Providing robustness against disturbances and modeling errors as the system advances through the discovered path. The general structure of the presented algorithm is composed of two phases. 1) It utilizes dynamics-based rapidly expanding trees (DRET) to create a forest of regionally growing random trees (RGRT) to discover a feasible path between the two states; 2) A simplified DRET is used to create a local feedback loop, which provides robustness against disturbances and modeling errors. Since Phase 2 can be computed in parallel with Phase 1, the algorithm can constantly seek to find better solutions for the planning and tracking problems. The further growth of the DRETs in RGRT forest can help the algorithm to find probabilistically optimal solutions for planning and control problems of increasing complexity.

The rest of the article is organized as follows. Section II discusses the fundamentals of DRET, RGRT forest, and the tracking feedback loop in detail. Some case studies of the proposed algorithm are presented in Section III. The paper is concluded with discussions and future work in Section IV.

II. Methodology

Before discussing details of the proposed algorithms, a brief introduction of the terms and notations used in this manuscript is provided in what follows. These definitions are used to eliminate possible ambiguities regarding terms used in the literature. In this article, a *state* of a system is referred to a distinct configuration of the system, denoted by $\underline{x}$. A state of a system is a vector of state variables which provides enough information to predict system behavior in response to the system inputs. The term *space*, denoted by S, is the union of all the states of a system. A *configuration space* (c-space or S_c) is a space, in which the state progressions are governed by a *state transition function*. It is also assumed that in a c-space there is a known function, $\phi(x_c, x_g)$, which returns the necessary system inputs for transitions from a current to a goal state. The term *state-space* is specifically used for systems with a continuous space, in which the state transition function is a differential equation of the general form:

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

where $\dot{\underline{x}} \in \mathbb{R}^n$ is a vector of time derivatives of the state variables for an n dimensional system. Parameter t indicates time and $\underline{u}$ is a vector of system inputs. It is assumed that function ϕ is unknown or hard to compute in state-spaces.

There are two main approaches for planning problems: discrete planning and planning in continuous spaces [13]. Although discrete planning algorithms are designed to solve problems with a countable space, they are also applicable to continuous space problems by discretization [15], [13], [16]. This discretization is achieved by overlaying either a uniform or a geometrically computed grid on top of the space [17]. The main drawbacks of the discrete planning algorithms on continuous spaces are grid resolution problems [13] and the associated curse of dimensionality [18], [13]. On the other hand, by eliminating the resolution problem, planning directly in continuous space provides a larger feasible space for solutions. Also, since the state-space of physical systems is continuous, planning in continuous-space eliminates the discretization phase. A class of continuous-space planners that is widely used in the last decade and has shown promising practicality is sampling-based algorithms [19]. Due to their sampling nature, this class of algorithms can avoid local minima [20], and require less computation by avoiding explicit construction of obstacle space [21], but they are only probabilistically complete [22]. RRT and probabilistic roadmap (PRM) [23] are two of the successful sampling-based algorithms that are widely used in path planning problems [24], [25], [20], [19].

RRT is constructed incrementally from vertices that are randomly drawn from the space, and edges which connect new vertices to their neighbors on the tree, where each vertex represents a state and each edge is a path between two states. Random sampling of states allows RRT to cover the space with the same probability distribution of the random number generator used in the algorithm. Thus, the state-space application of RRT requires a state to state controller that can traverse the edges between a randomly chosen state and its closest state on the tree. This requirement imposes many challenges on the application of RRT in the state-space of systems with complex dynamics. An alternative is to construct a tree incrementally by expanding edges from randomly selected vertices to find new states. Although this method does not require any controller and edges are created by simulating system dynamics forward from the chosen initial states, the resulting tree will be biased toward the old vertices and hence, grow much slower than an RRT tree. This behavior is explained in [9]. This article introduces two methods to eliminate the bias toward specific vertices and to increase the growth rate of the tree. Similiar to RRT, PRM samples the space for reachable configurations and tends to build a roadmap between the samples [26]. The idea behind PRM forms the basis of RGRT as explained below.

A. Regionally Growing Random Trees (RGRT)

RGRT algorithm combines the concepts of PRM with RDT. PRM takes random samples from the c-space and uses a local planner to find paths which connect these samples to other nearby samples. To use RDTs in state-space, it is required to find an expansion method that takes the differential state transition function of the form (1) into account. This expansion algorithm is explained in

Algorithm 1 Growing a DRET

procedure GROW(ψ)
  $i \leftarrow \Gamma(\mathcal{N}_\psi)$
  $j \leftarrow N_\psi + 1$
  $u_{ij} \leftarrow \Gamma(U)$
  $\delta t_{ij} \leftarrow \Gamma([0, \delta t_{max}])$
  $x_{new} \leftarrow \int_{t_i}^{t_i+\delta t_{ij}} f(t, x, u_{ij})\mathrm{d}t + x_i$
  if $x_{new} \in X_{feasible}$ **then**
    $v_j \leftarrow \{t_i + \delta t_{ij}, x_{new}, i\}$
    $e_{i,j} \leftarrow \{x_i, u_{ij}, \delta t_{ij}\}$
    $N_\psi \leftarrow N_\psi + 1$
    return $\{x_{new}, \psi\}$
  else
    return $\{\psi\}$

Algorithm 1 and a tree that expands based on this algorithm is called dynamics-based rapidly expanding tree (DRET). A DRET $\psi = \{v_i, e_{mn}\}, \forall i \in \mathcal{N}_\psi$ and for some m and n in $\mathcal{N}_\psi$ is a network with vertices (v_i) and edges (e_{mn}) defined as:

$$v_i = \{t_i, x_i, p_i\}, \tag{2}$$
$$e_{mn} = \{u_n, \delta t_n\}, \tag{3}$$

where x_i is the state vector, t_i is the time index, and $p_i \in \mathbb{N}^+$ indicates the index of the parent for vertex i. The edge from vertex m to vertex n, denoted by e_{mn}, is a set composed of a control action, u_n and a time difference δt_n. The set $\mathcal{N}_\psi$ is the set of all vertex indices of ψ and it is defined as:

$$\mathcal{N}_\psi = \{i \in \mathbb{N}^+ \quad | \quad i \leq N_\psi\}, \tag{4}$$

where N_ψ is the total number of vertices of ψ. The root of ψ is defined as its vertex with index 1 and denoted by $\mathcal{R}_\psi$. For any generic set X, the function $\Gamma(X)$ chooses a random sample from the members of X. Finally, $X_{feasible} \subset X$ defines the set of all feasible members of X that satisfy all the imposed constraints.

Algorithm 2 Finding edges between forest trees

procedure CONNECTFOREST($\mathcal{F}$)
  while $\kappa < 1 \vee \text{quest} = 1$ **do**
    for all $\psi_n \in \mathcal{F}$ **do**
      $\{x_{new}, \psi_n\} \leftarrow$ GROW(ψ_n)
      if $\exists \mathcal{R}_\psi; \|x_{new} - x_r\| \leq \epsilon$ **then**
        $E_{new} \leftarrow$ EDGE(ψ_n)
        if $\mathcal{C}(E_{new}) < \mathcal{C}(E_{n,m})$ **then**
          $E_{n,m} \leftarrow E_{new}$
    $\kappa = \mathcal{K}(\mathcal{F})$
  return $\mathcal{F}$

RGRT is a forest composed of regionally growing DRETs form root vertices that are randomly sampled from the state-space. If a leaf of a tree reaches to the vicinity of a root of another tree, an edge will be created between the two trees. If the iterative growth of the trees results in an improved path between the two trees, the previous path will be replaced by the more optimal path. This optimality is defined by a cost function $\mathcal{C}(E_{n,m})$ evaluated on the paths between the trees. Algorithm 2 explains the expansion of the RGRT forest. A RGRT forest $\mathcal{F} = \{\psi_i, E_{mn}\}$, $\forall i \in \mathcal{N}_\mathcal{F} = \{i \in \mathbb{N}^+ \mid i \leq N_\mathcal{F}\}$ and some m and n in $\mathcal{N}_\mathcal{F}$, is a network in which each vertex is a DRET tree and each edge E_{mn} is a path that connects $\mathcal{R}_m$ to $\mathcal{R}_n$.

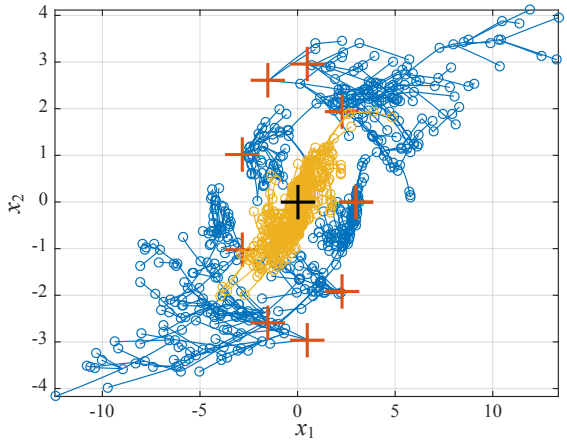

Fig. 1. A comparison between the growth of a single DRET tree with 500 vertices (illustrated with yellow color) and a forest of RGRT with 10 tress that each have 50 vertices (illustrated with blue color) for a Double-Integrator system explained in Section III-A. The roots of the RGRT and the DRET trees are depicted with red and black crosses, respectively.

After the construction of the roadmap, a graph search algorithm is used to find the optimal path in the forest between a current and a goal state. If the current or goal states are not in the set of initial states, the algorithm will return the path between the closest tree roots to the current and goal states. A comparison between RGRT and a single DRET is shown in Fig. 1. In this example, the total number of vertices in both the RGRT forest and the DRET is 500. As seen in this figure, RGRT explores a larger space for the same number of vertices while following the system dynamics.

B. Path Tracking using a Local DRET

Once a path between a current state and a desired state is discovered, a tracking controller can be used to traverse the path. Since the discovered path is composed of a sequence of open-loop inputs and corresponding time intervals, the modeled system is expected to follow the path exactly using feedforward commands, and utilizing a feedback loop allows compensation for disturbances, errors caused by planning resolution (ϵ), and modeling deficiencies. The tracking controller uses a local DRET that is only allowed to grow from its root vertex (current state) and it is biased by the open-loop action that is already discovered in the planning part as a feedforward term. The feedback control action is defined to be the action that minimizes the distance between the current state, x_c, and a point on the planned path x_{n+1}. Details of the tracking controller are given in Algorithm 3. Once the state variables approach the desired point within a certain

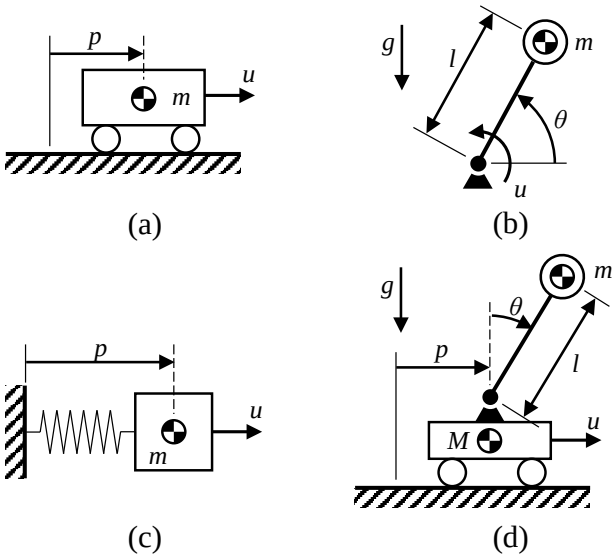

Fig. 2. The schematics of the systems that are used as case studies. (a) A simple mass on a friction-less surface; (b) A pendulum system with a point mass; (c) A mass-spring system with a nonlinear spring coefficient; (d) The cart-pole (inverted pendulum) system.

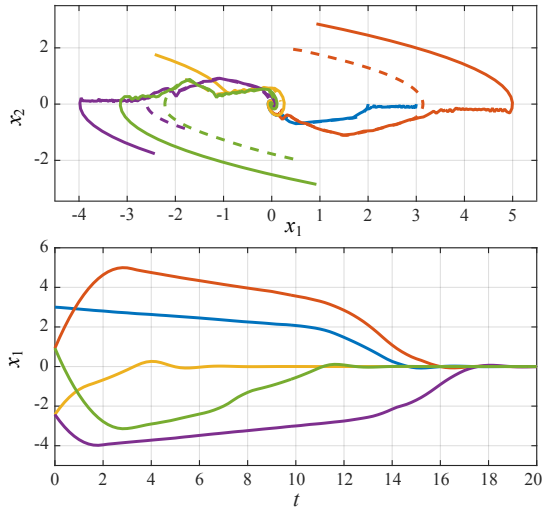

Fig. 3. Path planning and tracking controller response for a RGRT forest for the double-integrator system. The dashed lines show the paths between DRET roots that are close to the initial and goal states. The solid lines represent the paths followed by utilizing the tracking algorithm from initial states that are not coinciding with the roots.

stability region, a traditional feedback controller can take over and stabilize the system around the desired state.

Algorithm 3 Computing the feedback the current state, x_c.

procedure CONTROLACTION(x_c, P)
  $U_s \leftarrow \{u | u \pm u_o \in U\}$
  $n \leftarrow k \mid \forall x \in P \ \|x_c - x_k\| \leq \|x_c - x\|$
  while $i \leq i_{max}$ **do**
    $u_i \leftarrow \Gamma(U_s)$
    $x_i \leftarrow \int_{\delta t_s} f(t, x_c, u_i)\mathrm{d}t + x_c$
    $i \leftarrow i + 1$
  $u^* = u_k | \ \forall i \ \|x_k - x_c\| \leq \|x_i - x_c\|$
  return u^*

III. RESULTS AND DISCUSSION

This section describes four case studies of increasing complexity to investigate the performance and response of the proposed RGRT control scheme, as depicted in Fig. 2. Three of the four considered systems have a 2-D state-space and allow visualization of phase portraits.

A. Double-Integrator

The double-integrator is a second-order linear differential equation which models the dynamics of a simple mass under the effect of a time-varying force input in 1-D space. Two popular examples of double-integrator systems are: a simple mass moving on a frictionless surface, as depicted in Fig. 2(a); and a simple satellite which is modeled as a rotational inertia that is controlled by a pair of thrusters. The differential equations governing the state transition of a normalized double-integrator are: $\dot{x}_1 = x_2$ and $\dot{x}_2 = u$ where u is the input force to the system.

The implementation of the algorithm starts with constructing a RGRT forest. After the construction of the forest, it can be used to find feasible paths between a current and desired state. Some examples of the discovered path and the response of the tracking controller are depicted in Fig. 3. The desired point for all the initial conditions in this figure is $x = \{0, 0\}$. The dashed lines in the phase portrait illustrate the paths that are discovered from the closest roots to the initial and goal states. The solid lines represent the paths followed based on the tracking algorithm from initial conditions that do not coincide with the roots. As the system state gets close enough to the desired state ($\|x_d - x\| \leq \delta = 0.5$), a linear proportional-derivative (PD) controller ($k_p = 5$, $k_d = 2$) takes over the tracking control and stabilizes the system. The parameters used in this simulation are: $U \in [-1, 1]$, $\epsilon = 0.1$, $\delta t_{max} = 2$, and $N_{\mathcal{T}} = 15$.

B. Pendulum

Another interesting system that can be considered as a test-bed for the proposed algorithm is a simple pendulum model. Due to their interesting characteristics, 1-DoF pendulum systems are used in feedback control literature to test the performance of different controllers [27]. The system of differential equations which governs the motion of a pendulum with a massless rod under the effect of an applied input torque is:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \frac{1}{ml^2}u - \frac{g}{l}\cos(x_1). \end{aligned} \tag{5}$$

In the above equation, x_1 and x_2 represent θ and $\dot{\theta}$, respectively. The input torque is denoted by the control input u; g is the gravitational acceleration and m is the mass of the pendulum. Parameter l denotes the distance from the revolute joint to the center of mass, which is considered to

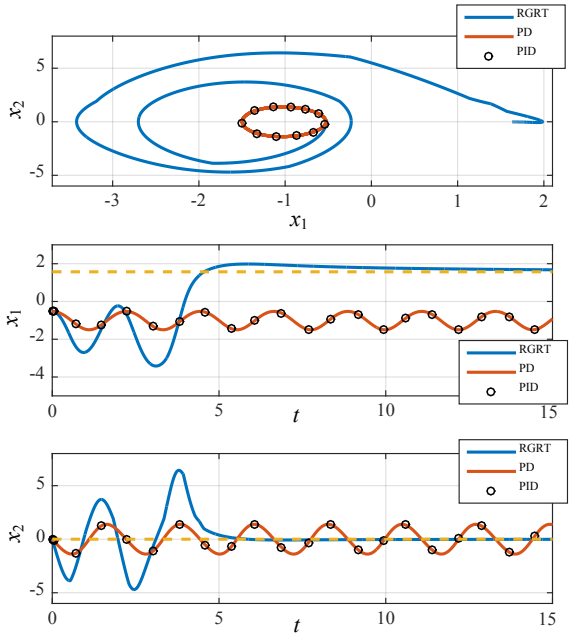

Fig. 4. RGRT based motion planning of a pendulum with input torque limitations. In contrast to PD and PID controllers, RGRT successfully swings up the pendulum by gaining enough momentum. The desired $x_1 = \pi/2$ and $x_2 = 0$ lines are depicted with dashed yellow lines.

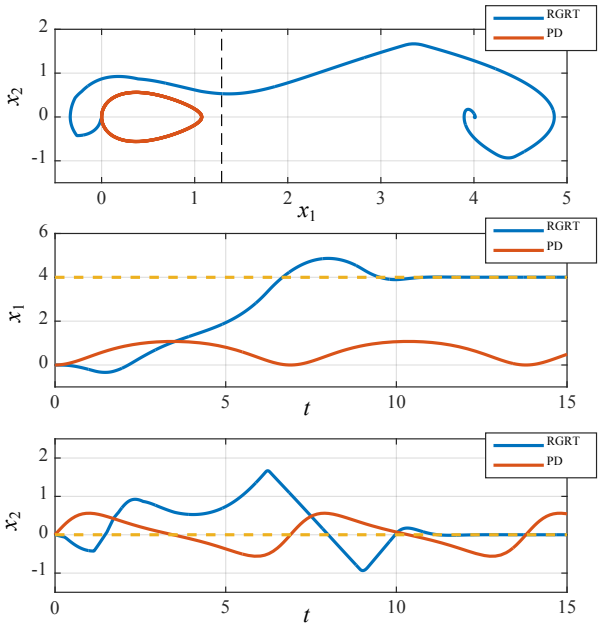

Fig. 5. RGRT based motion planning of the nonlinear mass-spring system. In contrast to a PD controller, RGRT successfully moves the mass to the desired position. The dashed black line in the top figure illustrates x_1 values where the spring force is maximized. The desired $x_1 = 4$ and $x_2 = 0$ are depicted with dashed yellow lines.

be at the center of the pendulum bob. Here, the maximum input torque is defined to be less than the torque needed to hold the pendulum at a horizontal configuration (the gravity induced torque applied to the pendulum is maximum at $\theta = (2n-1)\pi,\ \forall n \in \mathbb{R}$). Thus, without enough momentum, the pendulum can not pass through a horizontal configuration to reach a vertical configuration. This situation will fail most of the common controllers to control the pendulum to swing up ($\theta = \pi/2$) from initial positions below the horizontal line.

In order to test the capabilities of RGRT, three systems are considered for the swing up control of the pendulum: 1) A PD controller with hand-tuned coefficients $k_p = 10$ and $k_d = 2$; 2) A proportional-integral-derivative (PID) controller with $k_p = 10$, $k_i = 5$, and $k_d = 2$; and 3) Our RGRT forest control scheme. The responses of each of these control approaches are illustrated in Fig. 4. As depicted in this figure, RGRT can successfully swing the pendulum to an upward position by following a trajectory that allows it to gain enough momentum and overcome the actuation limitations while the simple PD and PID controllers fail to pass the $\theta = 0$ line and cause the system to undergo an oscillatory motion. The considered pendulum has a mass of 1 kg and length of 1 m. The RGRT parameters used for this simulation are: $U \in [-5, 5]$, $\epsilon = 0.2$, $\delta t_{max} = 1$, and $N_{\mathcal{F}} = 3$. Note that the output of the PD controller at $x = \{0, x_2\}^T$ is $u = \langle \{k_p, k_d\}, (x_d - x) \rangle = 5\pi - 2x_2$ which is greater than the maximum input torque ($U \in [-5, 5]$ Nm) for any $x_2 < 5.3540$ rad/s. As observed in Fig. 4, for the PD controller, x_2 does not exceed 1.4 rad/s. Thus the output of the PD controller saturates by the limits of the maximum input torque. In addition, for any initial state $x_i \in \{x \in \mathbb{R}^2 \mid -\pi/2 < x_1 \leq 0,\ x_2 = 0\}$, the input torque, u, will be positive and its value will be saturated by the maximum input torque. Considering that $U \in [-5, 5]$ which is less than 9.81 Nm (torque required to hold the pendulum at $x_1 = 0$), the PD controller can not possibly move the pendulum to the upward configuration regardless of the gains used.

As observed in Fig. 4, the response of the PD controller matches with the response of the PID controller. This is due to the fact that the integral term of the PID controller keeps increasing. Thus, the output of the controller will remain saturated by the torque limitations of the system. Due to this saturation, the control input to the system will remain equal to the maximum allowable value and it will not behave as a function of x_2. Moreover, since the system model (excluding the controller) does not have any frictional term, the energy stored in the system will not dissipate, which results in an oscillatory response.

In contrast to simple PD or PID controllers, RGRT finds a path that allows the system to gain enough momentum to pass the $\theta = 0$ line to reach the upward configuration. This is observed in the phase portrait of the system depicted in Fig. 4. Similar to the double-integrator, when the system state gets within a close neighborhood of the desired state ($\|x_d - x\| \leq \delta = 1$), a PD controller ($k_p = 10$ and $k_d = 2$) takes over the tracking and stabilizes the system.

C. Nonlinear mass-spring system

To further examine the behavior of RGRT, a mass-spring system with a nonlinear stiffness coefficient is considered.

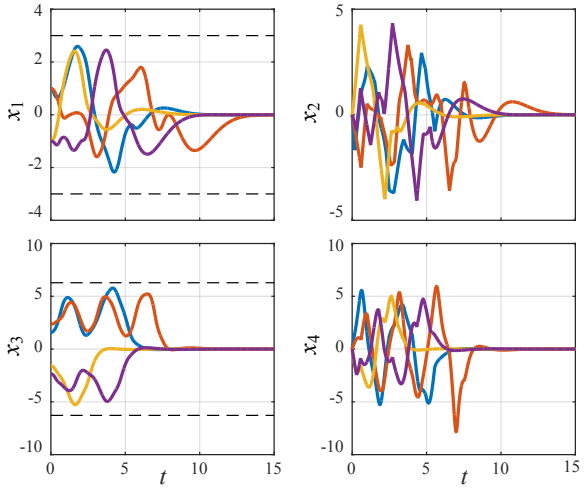

Fig. 6. RGRT based motion planning of the cart-pole system for different initial conditions. The desired state of the system for all the initial conditions is the upward configuration of the pole as the cart rest on $x_1 = 0$. The dashed black lines define the regions of feasible x_1 and x_3 values. As seen in the figure, throughout the trajectory, x_1 and x_3 values remain in the feasible regions.

The governing differential equations of this system are given as:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \frac{1}{m}(u - ke^{(-|x_1|)}\sin(x_1)), \end{aligned} \tag{6}$$

where m is the mass and k is the stiffness coefficient. By taking the derivative of the spring force with respect to x_1 it is observed that the spring force reaches its maximum value at $x_1 = \pi/4$ and it is equal to:

$$f_{max} = \frac{k\sqrt{2}}{2}e^{(-\frac{\pi}{4})}. \tag{7}$$

Similar to the pendulum example, with limitations on the input force, the system can not pass $x_1 = \pi/4$ point without gaining enough momentum. This is observed in the responses of an RGRT motion planner and a PD controller ($k_p = 10$ and $k_d = 2$) illustrated in Fig. 5. In this simulation, the mass is considered to be 1 kg and the stiffness gain is assigned to be 4 N/m. The input force of the system is limited between -1 and 1 N. As shown in this figure, while the PD controller fails to move the mass to the desired state (defined as $x_d = \{4, 0\}$), RGRT successfully overcomes the input force limitation by increasing system momentum and eventually moves the mass to the desired position. The same justification about the effects of the PD controller gains and utilization of an integral term that are explained for the pendulum example (section III-B) are also valid for this nonlinear mass-spring system. As the system states reach the vicinity of the desired state ($\|x_d - x\| \leq \delta$), a PD controller takes over the tracking controller and stabilizes the system. The parameters used in the construction of the RGRT forest for this example are listed as: $U \in [-1, 1]$, $\epsilon = 0.2$, $\delta t_{max} = 1$, and $N_{\mathcal{F}} = 7$.

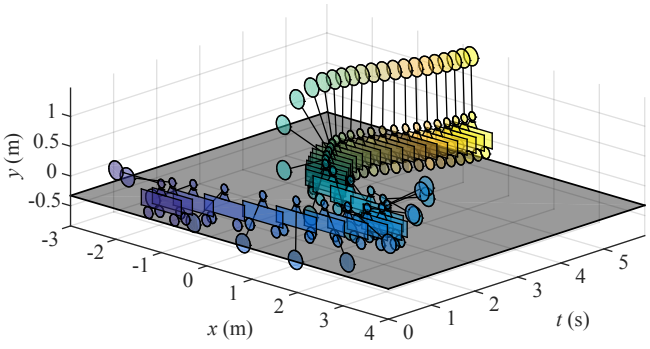

Fig. 7. An action-shot of the cart-pole system as it controlled by RGRT based motion planning for initial condition x_{i3}.

D. Cart-pole

A cart-pole (inverted pendulum) is an under-actuated system with a 4-D state-space. The governing differential equations of motion of this system are presented in what follows. Figure 2(d) illustrates the system schematics and the parameters used in the differential equations.

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \frac{u + \mathcal{C}_1}{2(M + m\sin^2(x_3))}, \\ \dot{x}_3 &= x_4, \\ \dot{x}_4 &= \frac{g\sin(x_3)(M+m) + \mathcal{C}_2 - u\cos(x_3)}{l(M + m(1 - \cos^2(x_3)))}, \end{aligned} \tag{8}$$

where $x_1 = p$ and $x_2 = \theta$. Parameters m and M represent the mass of the pendulum and the cart, respectively. The gravitational acceleration is denoted by g and l measures the length of the pendulum, as depicted in Fig. 2(d). The rest of the parameters are defined as:

$$\begin{aligned} \mathcal{C}_1 &= m(2l{x_4}^2\sin(x_3) - x_2x_4\sin(2x_3) - g\sin(2x_3)) \\ \mathcal{C}_2 &= x_2x_4\sin(x_3)(M+m) - \cos(x_3)(ml{x_4}^2\sin(x_3)) \end{aligned}$$

Figure 6 illustrates the behavior of the cart-pole system with a RGRT control scheme for 4 different initial states: $x_{i1} = \{1, 0, \pi/2, 0\}$, $x_{i2} = \{1, 0, 3\pi/4, 0\}$, $x_{i3} = \{-1, 0, -\pi/2, 0\}$, and $x_{i1} = \{-1, 0, -3\pi/4, 0\}$. Note that, all these initial conditions represent configurations in which the pendulum is oriented downward ($|\theta| \geq \pi/2$); Thus, a full state-feedback controller or any other controller that is derived using a linearized system model around $x_d = \{0, 0, 0, 0\}$ is not guaranteed to be able to move the system from the presented initial conditions to the desired state. This is due to the fact that the linearized models for sine and cosine functions are not valid for all four quadrants. But, as depicted in Fig. 6, RGRT can successfully control the system to the desired state ($x_d = \{0, 0, 0, 0\}$) from the discussed initial conditions. In these simulations, the feasible region for the states are defined as:

$$x = \{x \in \mathbb{R}^4 \mid |x_1| \leq 3,\ |x_3| \leq 2\pi\}. \tag{9}$$

Similar to the other case studies, when the system state gets within a defined neighborhood of the desired state (i.e. $\|x_3\| \leq \delta = \pi/4$) a full state feedback controller $u = -K(xd - x)$ designed based on a linearized model around x_d with $K = \{-0.41, -1.0, -21, 2, -6.0\}$ takes over and stabilizes the system. Rest of the parameters that are used in this simulation are: $M = 1$ kg, $m = 0.1$ kg, $l = 1$ m, $U \in [-10, 10]$, $\epsilon = 0.5$, $\delta t_{max} = 1$, and $N_{\mathcal{F}} = 5$. An action-shot of the cart-pole system controlled by RGRT for the initial condition x_{i3} is depicted in Fig. 7.

IV. Conclusions and Future Work

This paper discussed the details of solving a control problem for a generic dynamic system with input limitations by utilizing a synergistic motion planning and control algorithm, RGRT. The details of the RGRT forest construction and expansion of DRET tress are discussed in detail. The proposed control scheme is then used for four different case studies with 2- and 4-dimensional state-spaces. It is shown that RGRT can successfully utilize system dynamics to overcome control input limitations and manipulate the system from an initial state to a goal state.

Due to its sampling nature, DRETs are also compatible with systems that have discrete inputs (e.g. fluidic systems that are controlled by directional valves) or discrete state-spaces. This allows the utilization of RGRT for a wide range of applications. As discussed in Section II, solving a control problem with RGRT is composed of two phases: 1) expansion of DRET trees and construction of the roadmap of the forest, which is a pre-processing phase; 2) solving a planning problem and finding a path between a current and desired states using the constructed roadmap, which is performed online. In parallel to the execution of the second phase, the forest trees can continue expanding to find improved paths in the forest and substitute the previous paths with more optimal solutions.

The time required for the pre-processing phase depends on the expansion rate of DRETs. Thus, improving the DRET growth algorithm can significantly reduce the pre-processing time and even allow online construction of the roadmap. A possible approach to achieve a faster expansion is to use acceleration profiles that are designed based on system dynamics and input limitations. Formulation of such profiles and studying the effect of different horizons on the behavior of the tracking controller are some of the future work of this research.

Acknowledgments

The authors would like to thank Shadi T. Kalat for her generous help and thoughtful comments on this manuscript.

References

[1] M. W. Spong, "The swing up control problem for the acrobot," *Control Systems, IEEE*, vol. 15, no. 1, pp. 49–55, 1995.

[2] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4054–4061.

[3] F. Grasser, A. D'arrigo, S. Colombi, and A. C. Rufer, "JOE: a mobile, inverted pendulum," *Industrial Electronics, IEEE Transactions on*, vol. 49, no. 1, pp. 107–114, 2002.

[4] Q. Wei, W. P. Dayawansa, and W. Levine, "Nonlinear controller for an inverted pendulum having restricted travel," *Automatica*, vol. 31, no. 6, pp. 841–850, 1995.

[5] T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3)," *arXiv preprint arXiv:1003.2005*, 2010.

[6] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, p. 0278364911434236, 2012.

[7] B. Lincoln and B. Bernhardsson, "Efficient pruning of search trees in LQR control of switched linear systems," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 2. IEEE, 2000, pp. 1828–1833.

[8] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, 2010.

[9] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[10] J. Moore and R. Tedrake, "Control synthesis and verification for a perching UAV using LQR-Trees," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 3707–3714.

[11] J. Moore, R. Cory, and R. Tedrake, "Robust post-stall perching with a simple fixed-wing glider using LQR-Trees," *Bioinspiration & biomimetics*, vol. 9, no. 2, p. 025013, 2014.

[12] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," 2000.

[13] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[14] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[15] T. Guan-Zheng, H. Huan, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279–285, 2007.

[16] R. Bohlin, *Robot path planning*. Chalmers University of Technology,, 2002.

[17] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots:: Ii. a review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283–317, 2003.

[18] R. Bellman, *Dynamic Programming*, ser. Dover Books on Computer Science Series. Dover Publications, 2003.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[20] N. M. Amato and G. Song, "Using motion planning to study protein folding pathways," *Journal of Computational Biology*, vol. 9, no. 2, pp. 149–168, 2002.

[21] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *arXiv preprint arXiv:1005.0416*, 2010.

[22] S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Robotics Research. The Eleventh International Symposium*. Springer, 2005, pp. 36–54.

[23] L. E. Kavraki and J.-C. Latombe, "Probabilistic roadmaps for robot path planning," 1998.

[24] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *Robotics, IEEE Transactions on*, vol. 26, no. 4, pp. 635–646, 2010.

[25] J. Cortés and T. Siméon, "Sampling-based motion planning under kinematic loop-closure constraints," in *Algorithmic Foundations of Robotics VI*. Springer, 2005, pp. 75–90.

[26] R. Geraerts and M. H. Overmars, "A comparative study of probabilistic roadmap planners," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 43–57.

[27] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.