Sampling-based motion planning under kinodynamic constraints

December, 2019

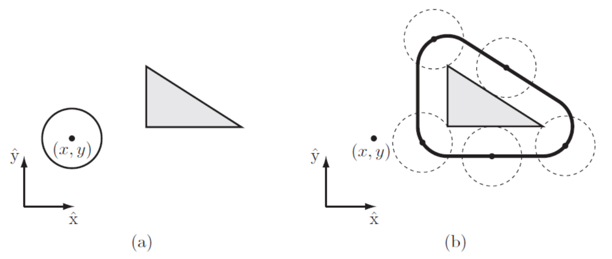
Tel Aviv University

Basic definitions: Configuration space

The d-dimensional space $\mathcal{C}$ containing all possible configurations of the robot is called the configuration space (C-space).

A subset $\mathcal{F} \subset \mathcal{C}$ of all the collision-free configurations is called the free space.

The C-obstacles, defined as $\mathcal{C}_{\text{forb}} = \mathcal{C} \setminus \mathcal{F}$, are rarely represented exactly (may have a complex mathematical representation).



Figures from [Lynch and Park, 16]

An alternative formulation of the motion-planning problem

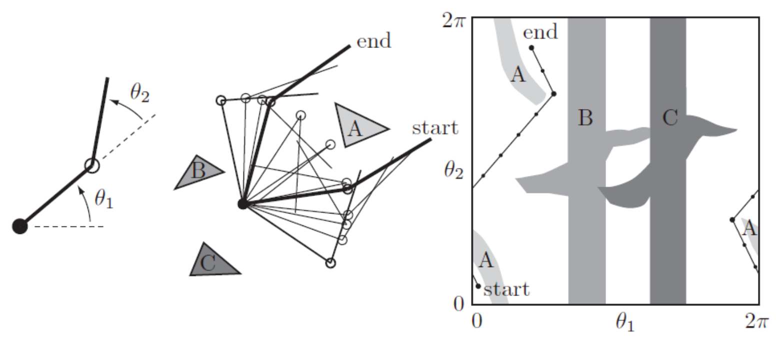
Given:

- A point robot
- A d-dimensional configuration space $\mathcal{C}$ (C-space)
- C-obstacles (often not explicitly given) $\mathcal{C}_{\mathrm{forb}}$
- Free space $\mathcal{F} = \mathcal{C} \setminus \mathcal{C}_{\mathrm{forb}}$
- Initial and final configurations

Goal:

- Plan a continuous path in the free space from the initial configuration to the final configuration

An alternative formulation of the motion-planning problem



Figures from [Lynch and Park, 16]

Path planning vs. Motion planning

- Path planning is a sub-problem of motion planning
- Path planning is purely geometric
- Motion planning also deals with the dynamics, the duration of motion, or constraints on the motion

(non geometric) motion planning

- Suppose that $\mathcal{C} \subset \mathbb{R}^n$
- $\mathcal{U} \subset \mathbb{R}^m$ is the set of control inputs (e.g., steering angle, accelerations) available to drive the robot
- The state of the robot is a generalization of the robot's configuration
- Each state should incorporate the dynamic state of the robot
- $\mathcal{X}_{\text{free}}$ is the free state space

Working in state space

- Allows the planner to incorporate dynamics constraints on the returned paths
- The dimension of the state space is typically $d = 2n$
  - For a configuration of a steerable car represented by (x, y, θ), the corresponding state incorporating the dynamics can be represented by $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$
- Planning in state space means solving a higher dimensional problem

Motion equation

Motion equation: $\dot{x} = f(x, u)$, where x is a state and u is a control

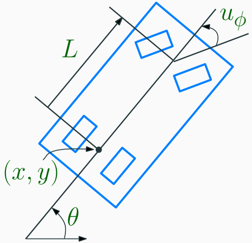
or in integral form, $x(T) = x(0) + \int_0^T f(x(t), u(t))dt$

Example 1: simple (kinematic) car

Each state has (x, y, θ) but $m = 2$ (signed speed u_s and steering angle u_ϕ).

The dynamics of the kinematic car are described as follows:

$$\begin{aligned} \dot{x} &= u_s \cos\theta, \\ \dot{y} &= u_s \sin\theta, \\ \dot{\theta} &= \frac{u_s}{L} \tan u_\phi, \end{aligned}$$



where L is the distance between the front and rear axle of the car

Example 1: simple (kinematic) car

There could be collision-free paths that the car is incapable of following (e.g., slide directly sideways into a parking space)

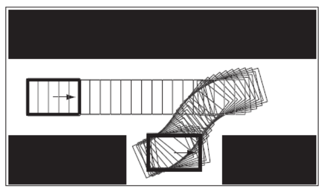


Figure from [Lynch and Park, 16]

Controllable degrees of freedom

- Sometimes not all degrees of freedom are controllable
- An example: a steerable car (an even simpler model than the kinematic car)
  - It has 3 degrees of freedom (x, y, θ)
  - Only one controllable dof (=the steering angle)
- Nonholonomic systems:
  - When #controllable dofs $<$ #dofs
  - Cannot execute an arbitrary path (could be problematic for PRM)

Example 2: second-order (dynamic) car

- Each state keeps (x, y, θ, v, ϕ), where v is the speed and ϕ is the steering angle
- $m = 2$: (u_v, u_ϕ) control the rate of change of v and ϕ

The dynamics of the second-order car are described as follows:

$$\begin{aligned} \dot{x} &= v\cos\theta, \quad \dot{y} = v\sin\theta, \quad \dot{\theta} = \frac{vM}{L}\tan\phi, \\ \dot{v} &= u_v, \quad \dot{\phi} = u_\phi, \end{aligned}$$

where M is the mass of the car, and L is the distance between the front and rear axle of the car.

Steering functions

- Returns a trajectory between two states
- Corresponds to solving the *Two-point boundary value problem (BVP)* in the state space
- For many models of robots it is impractical to generate a BVP solver!
    - There are no steering functions available for such robots
- Certain planners (like PRM) require a steering function

Kinematic vs. kinodynamic constraints

- Kinematic constraints consider only the current position of the robot
- Kinodynamic constraints take into account the forces that caused the motion

Rapidly exploring random tree (RRT)

- Well suited to complex tasks involving kinodynamic constraints:
  Does not require a steering function
- Probably the most commonly used planner

RRT [LaValle and Kuffner, 01]

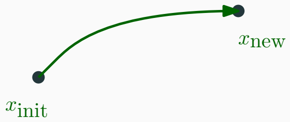
Algorithm 2 RRT ($x_{\mathtt{init}}, \mathcal{X}_{\mathtt{goal}}, k, T_{\mathtt{prop}}, \mathbb{U}$)

$\mathcal{T}$.init(x_{init})
for $i = 1$ to k **do**
    $x_{\text{rand}} \leftarrow$ RANDOM_STATE()
    $x_{\text{near}} \leftarrow$ NEAREST_NEIGHBOR($x_{\text{rand}}, \mathcal{T}$)
    $t \leftarrow$ SAMPLE_DURATION($0, T_{\text{prop}}$)
    $u \leftarrow$ SAMPLE_CONTROL_INPUT($\mathbb{U}$)
    $x_{\text{new}} \leftarrow$ PROPAGATE(x_{near}, u, t)
    if COLLISION_FREE($x_{\text{near}}, x_{\text{new}}$) **then**
        $\mathcal{T}$.add_vertex(x_{new})
        $\mathcal{T}$.add_edge($x_{\text{near}}, x_{\text{new}}$)
return $\mathcal{T}$

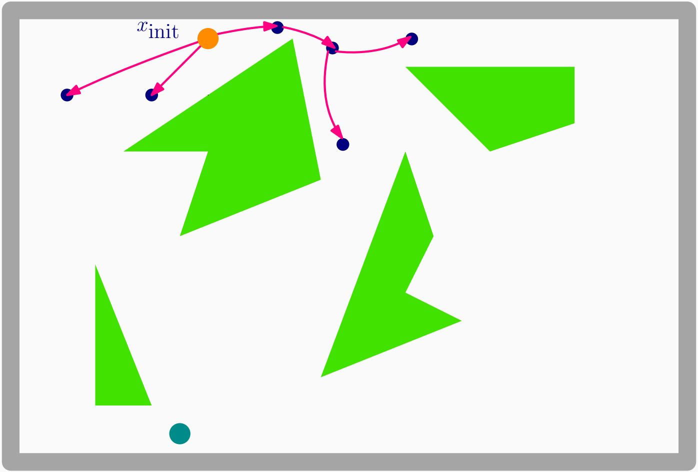
RRT [LaValle and Kuffner, 01]

This variant of RRT uses forward propagation of random controls

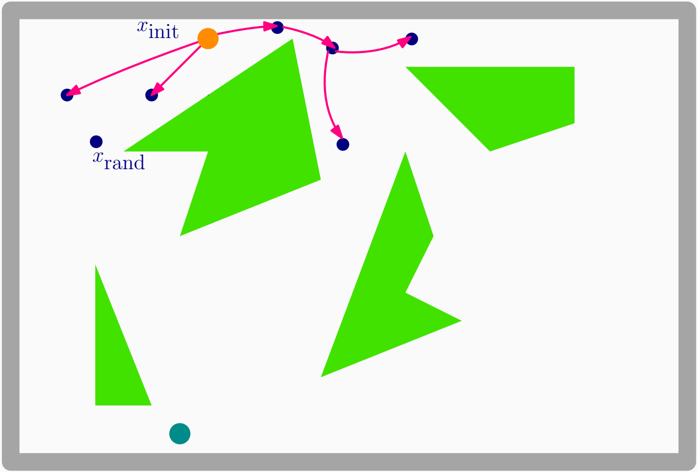
$$u \leftarrow \text{Sample}(\mathcal{U})$$
$$x_{\text{new}} \leftarrow \int_0^t f(x(T), u)dT + x_{\text{init}}$$



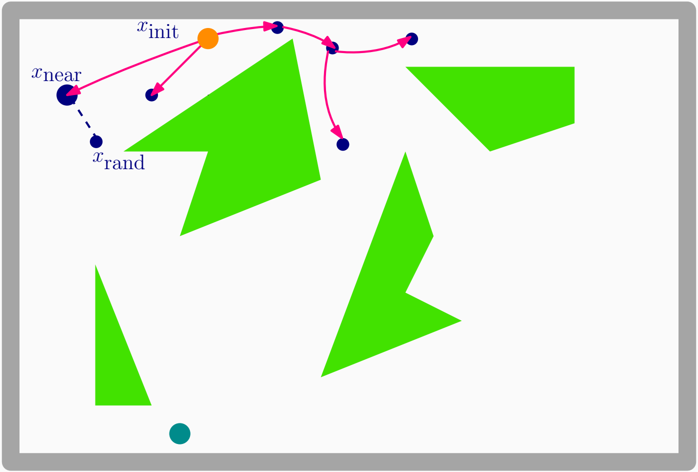
RRT: animation



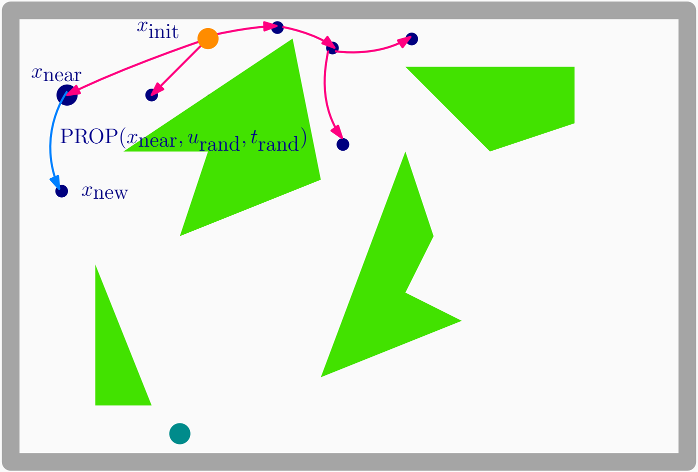
RRT: animation



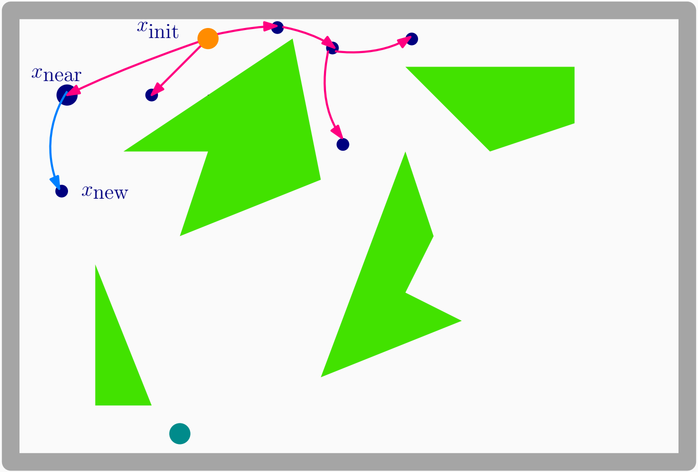
RRT: animation



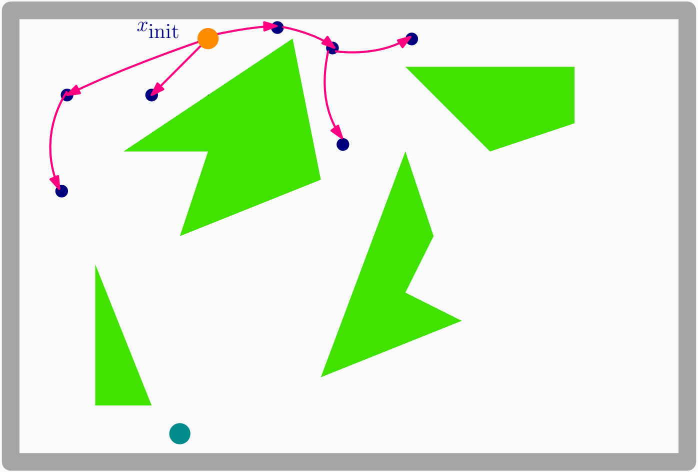
RRT: animation



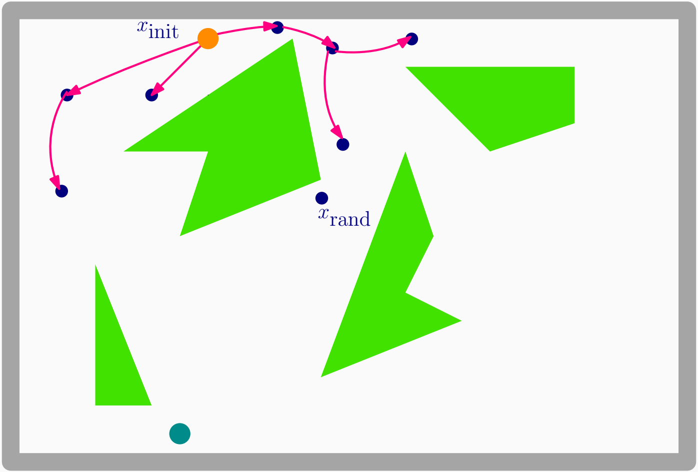
RRT: animation



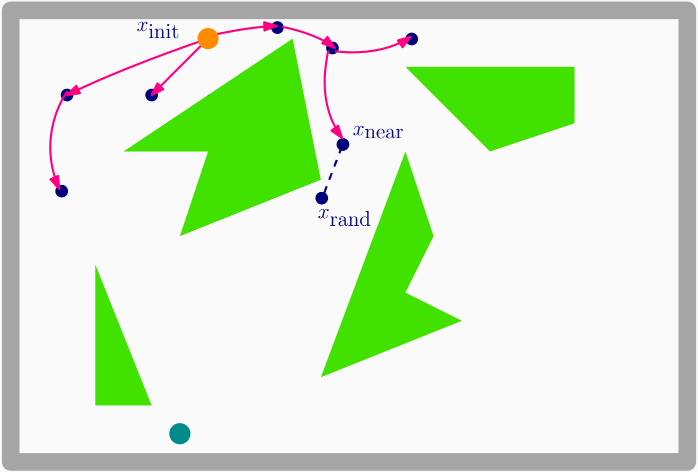
RRT: animation



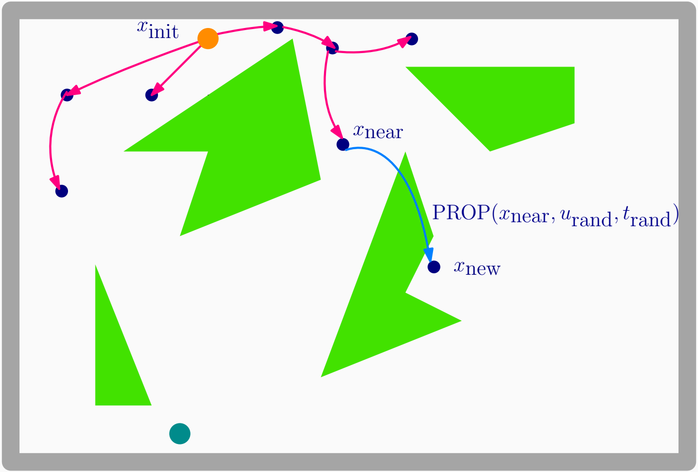
RRT: animation



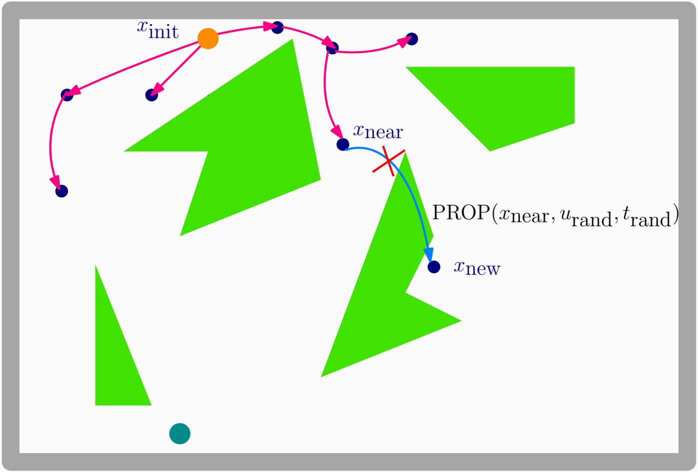
RRT: animation



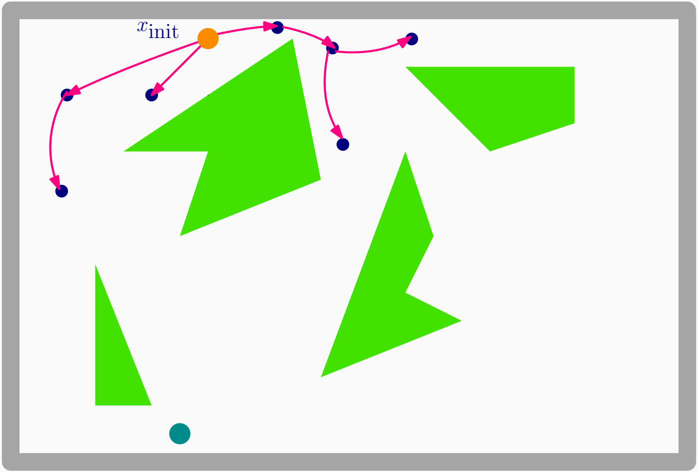
RRT: animation



RRT: animation



RRT: animation



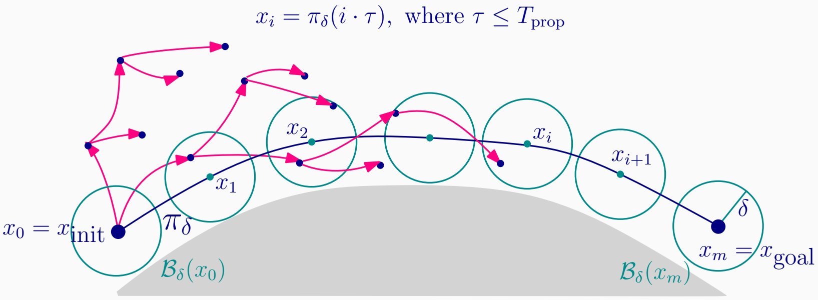
Probabilistic completenesss of RRT

This variant of RRT is proven to be PC assuming that

- the control function is piecewise constant
- the system is Lipschitz continuous: $\exists K_u, K_x > 0$ s.t.
  $\forall x_0, x_1 \in \mathcal{X}, u0, u_1 \in \mathcal{U}$
  $\|f(x_0, u_0) - f(x_0, u_1)\| \leq K_u \|u_0 - u_1\|$
  $\|f(x_0, u_0) - f(x_1, u_0)\| \leq K_x \|x_0 - x_1\|$

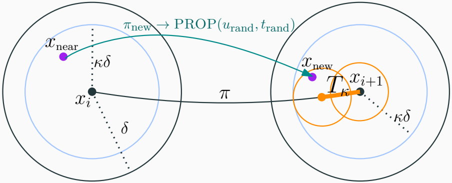
[K., Solovey, Littlefield, Bekris, Halperin, 19]

Proof sketch



Transition between $\mathcal{B}_\delta(x_i)$ and $\mathcal{B}_\delta(x_{i+1})$

- Lemma 1: The probability p_{near} that RRT will grow the tree from $x_{\text{near}} \in \mathcal{B}_\delta(x_i)$, given that a vertex exists in $\mathcal{B}_{\kappa\delta}(x_i)$, is positive.
- Lemma 2: The probability p_{prop} that the propagation step of RRT from $x_{\text{near}} \in \mathcal{B}_\delta(x_i)$ ends in $x_{\text{new}} \in \mathcal{B}_{\kappa\delta}(x_{i+1})$ is positive.



$p = p_{\text{near}} \cdot p_{\text{prop}} > 0$, and is independent of n