Robot Autonomy

Lecture 16:
Planning with Constraints

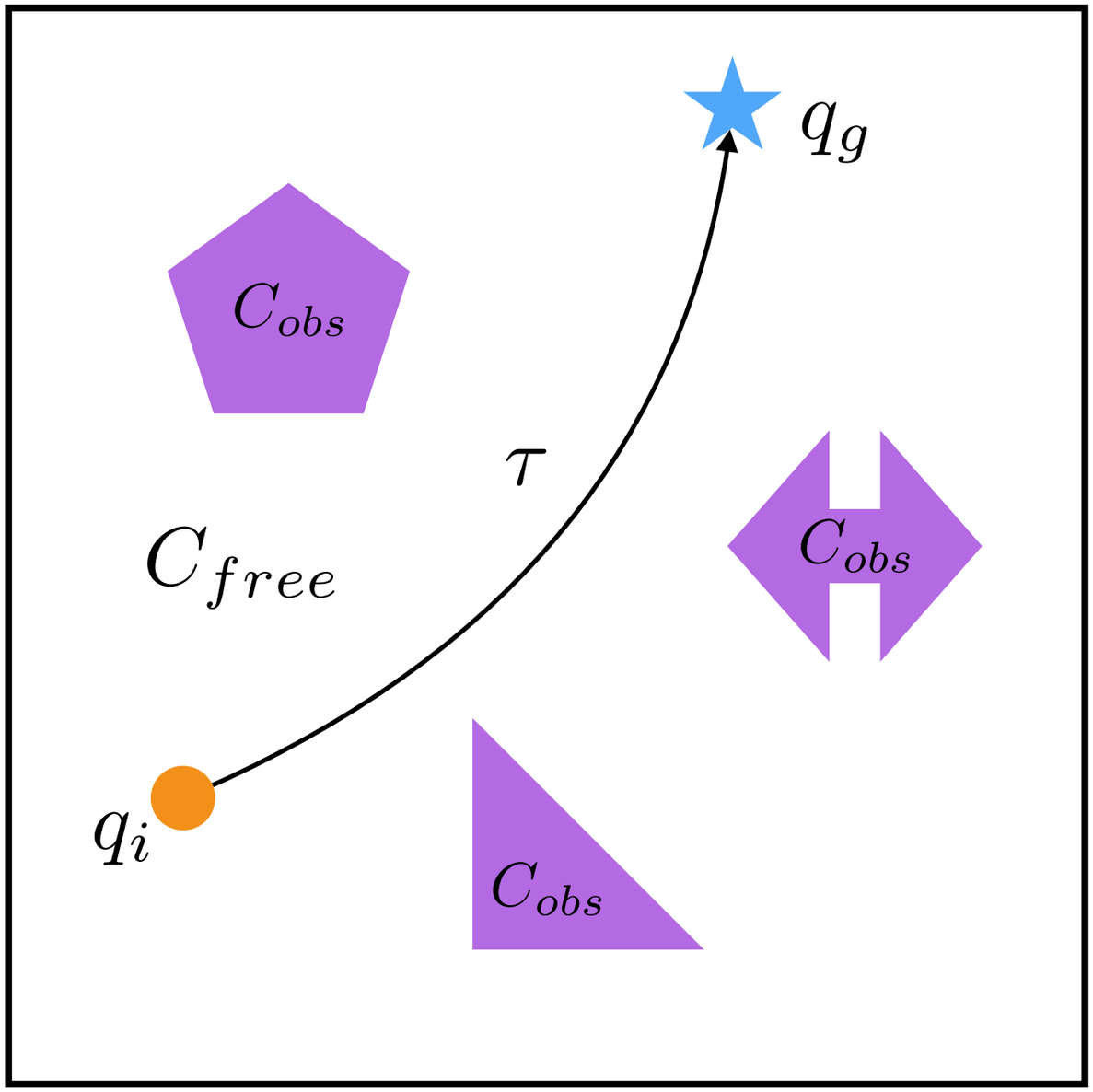
Oliver Kroemer

Piano Mover Problem Constraints

$$\tau(0) = q_i$$

$$\tau(1) = q_g$$

$$\tau : [0, 1] \rightarrow C_{free}$$



$$F(q) = 1 \text{ if } q \in C_{obs}$$

$$F(q) = -1 \text{ if } q \in C_{free}$$

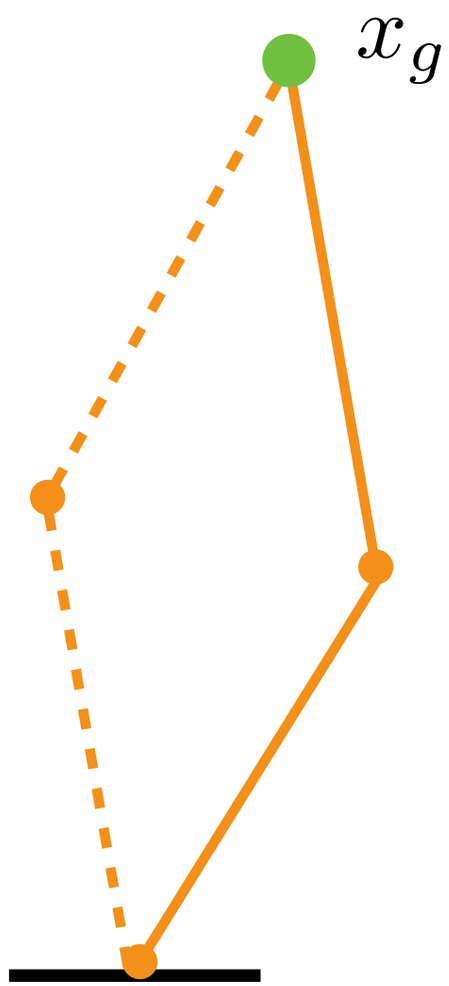
$$F(\tau(t)) = -1 \text{ for all } t \in [0, 1]$$

Goal Constraints

- Goal constraint

Forward Kinematics

$$FK(g) = [x, y, z, \theta_r, \theta_p, \theta_y]^T$$



Goal Set

$$Q_g = \{q | FK(g) = x_g\}$$

Equality Constraint

- Constraint only applies at the end of the trajectory

Equality and Inequality Constraints

- Holding an object upright
  - Equality constraint

$$\theta_r = 0$$

$$\theta_p = 0$$

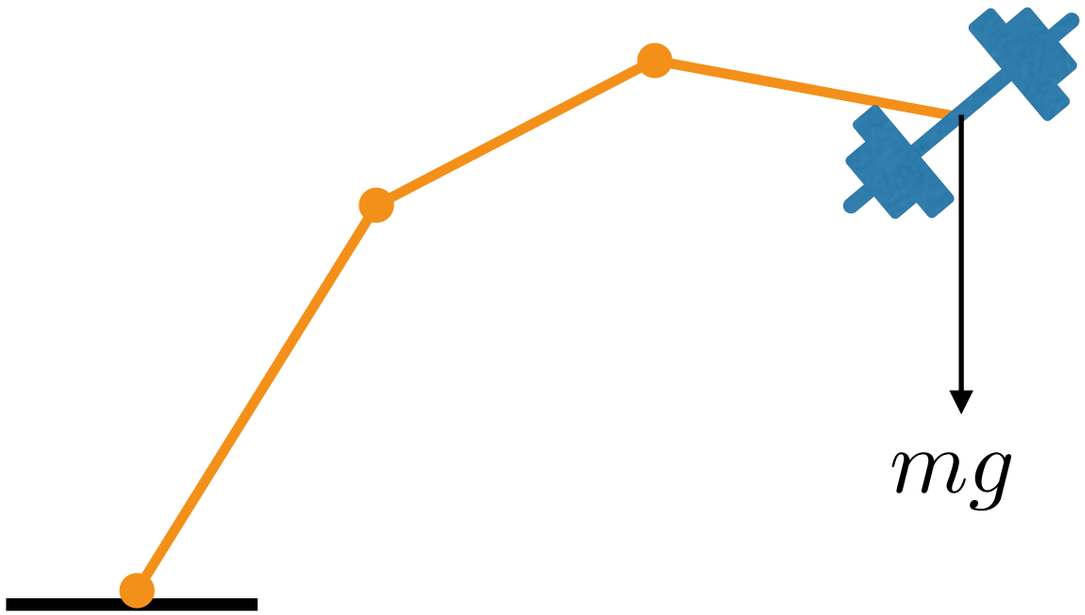
  - Inequality constraint

$$\theta_r \in [r_1, r_2] \qquad \theta_p \in [p_1, p_2]$$

- Constraints may apply along entire trajectory

Torque Limits

- Torque limits



Torque

$$J(q)^T \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix} = T$$

$$T \in \begin{bmatrix} T_{1min} & T_{1max} \\ T_{2min} & T_{2max} \\ T_{3min} & T_{3max} \end{bmatrix}$$

Inequality Constraint

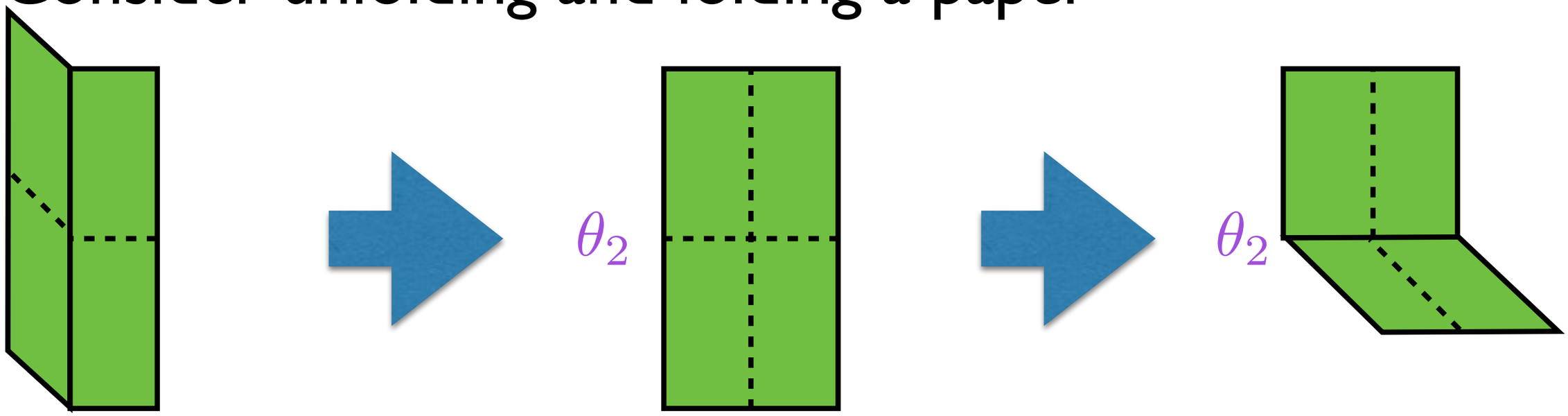
- Constraint applies trajectory-wide

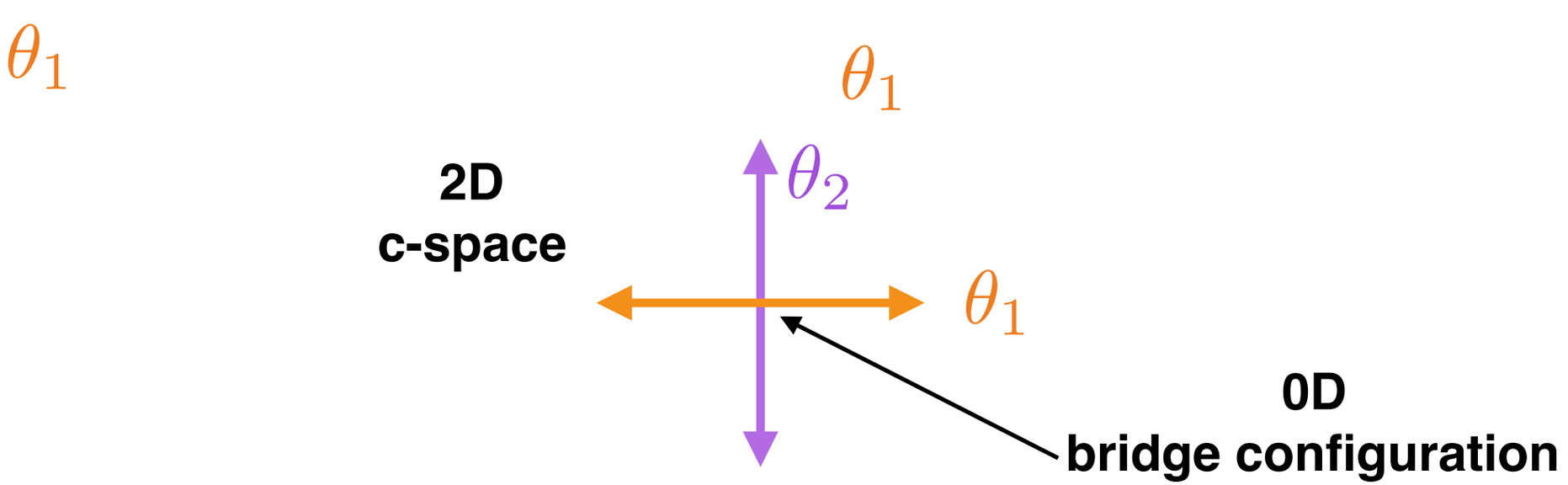
Dimensionality

- Some constraints reduce the dimensionality of allowed q
  - Full dimensional constraints - no reduction
    - Holding up cup - inequality constraint
    - Torque constraint
  - Lower dimensional constraints - reduction
    - Goal constraint
    - Holding up cup - equality constraint

Bridge Configurations

- Consider unfolding and folding a paper

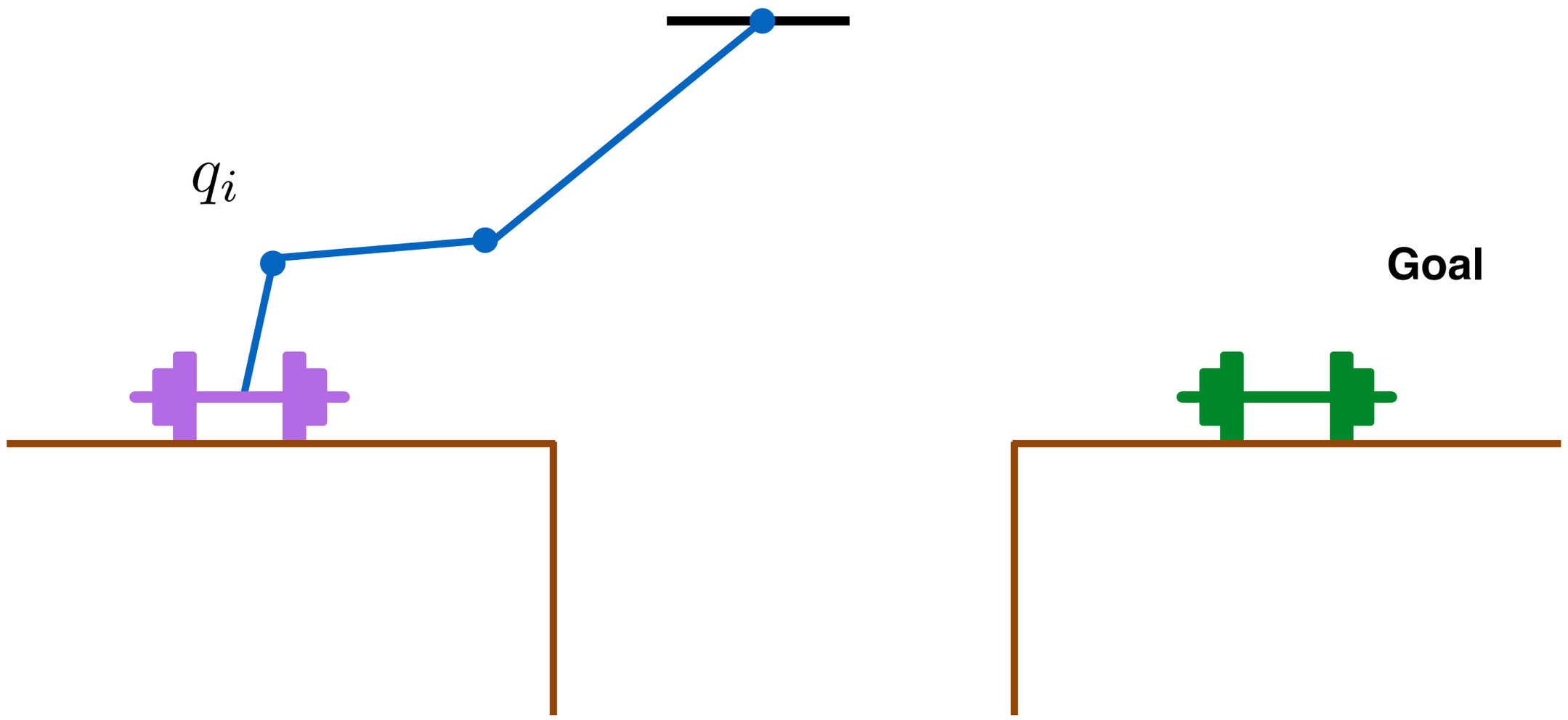




- Bridge configurations connect c-space manifolds
- Need to plan through bridge configurations

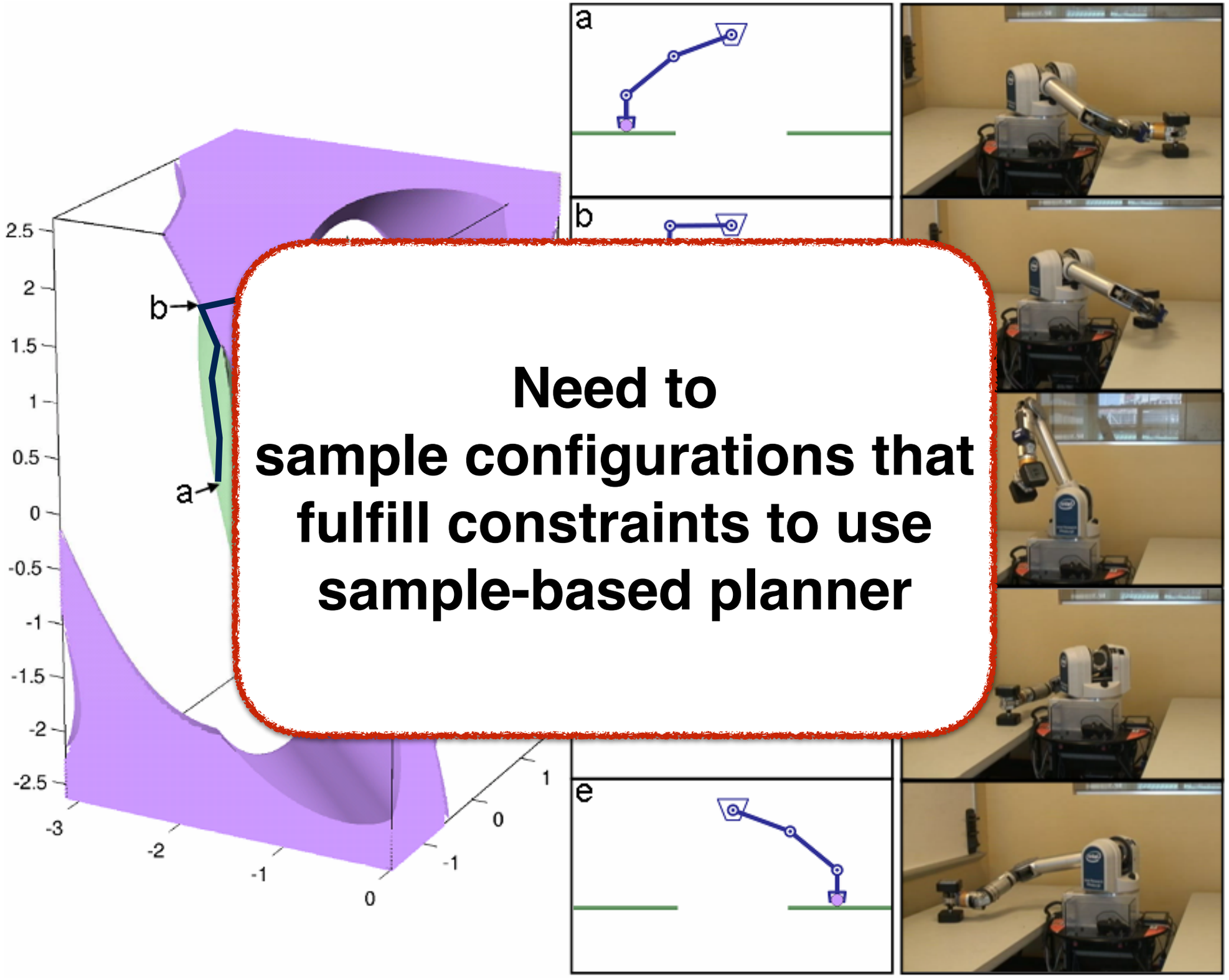
Planning with Constraints

- Torque-limited robot needs to move object to other table



- Want to plan a trajectory using a sample-based planner

Planning with Constraints



Need to sample configurations that fulfill constraints to use sample-based planner

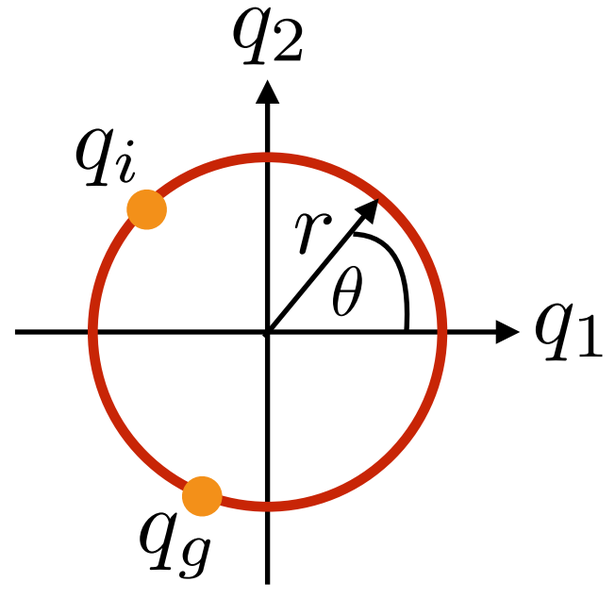
Berenson et al. Manipulation Planning on Constraint Manifolds, ICRA 2009

Explicit and Implicit Constraint Representations

- Explicit constraint representation

$$q_1 = r\cos(\theta)$$

$$q_2 = r\sin(\theta)$$



  - Directly sample from constraint space

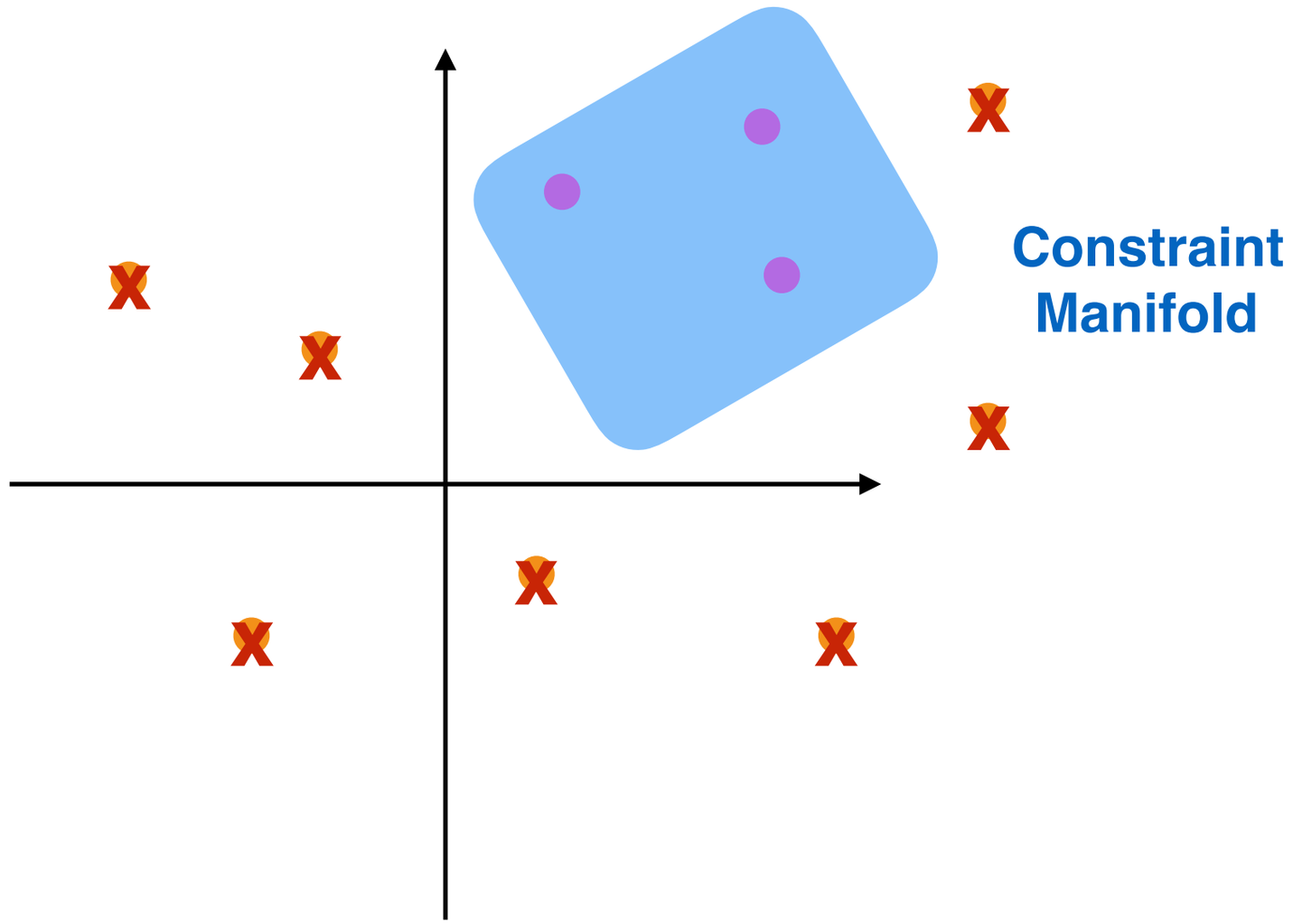
- Implicit constraint representation

$$C_{valid} = \{q | q_1^2 + q_2^2 = r^2\}$$

  - Cannot sample directly from constraint space

Rejection Sampling

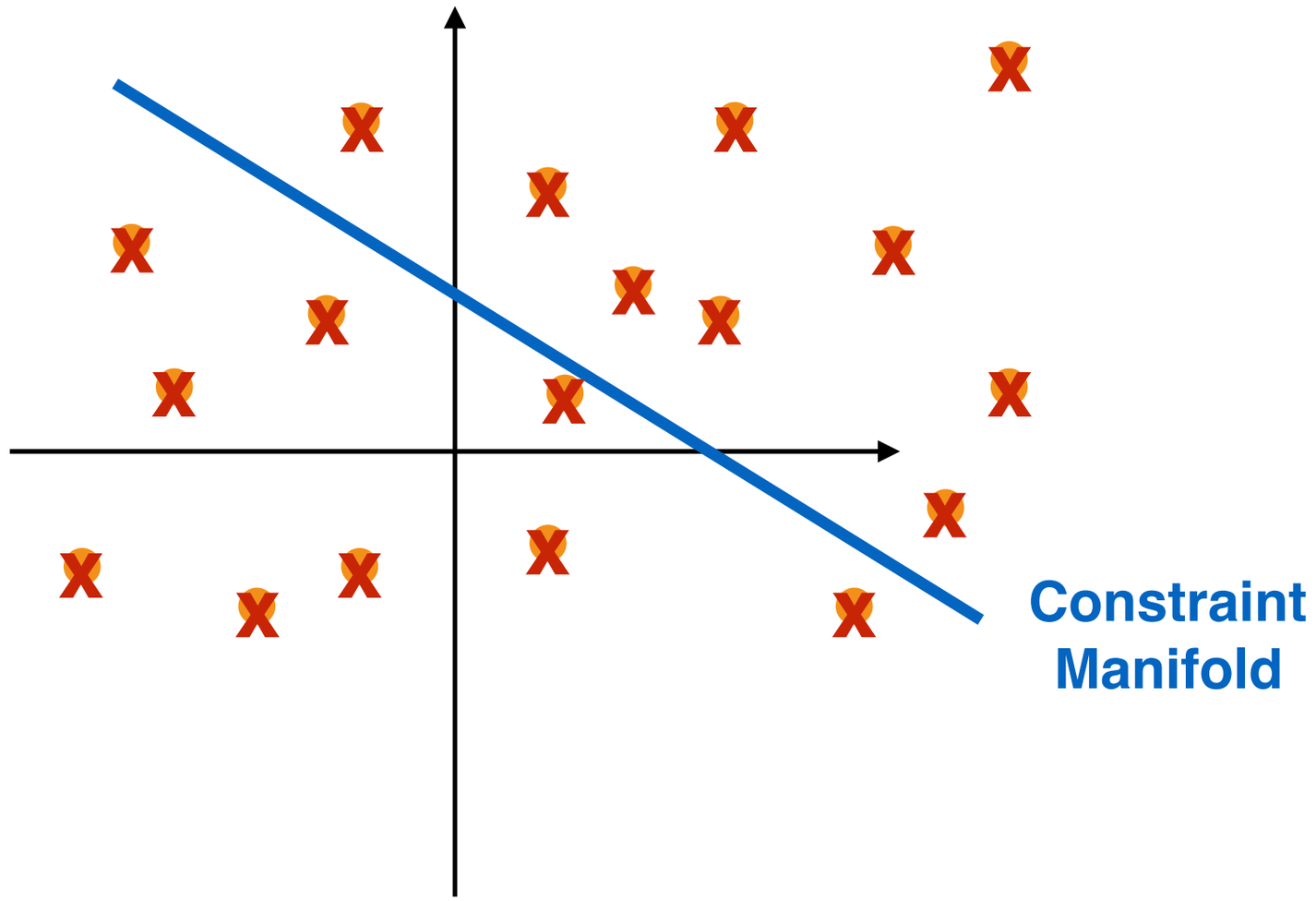
- Sample configurations until a sample is valid



- Only applicable if constraint manifold is full dimensional

Rejection Sampling

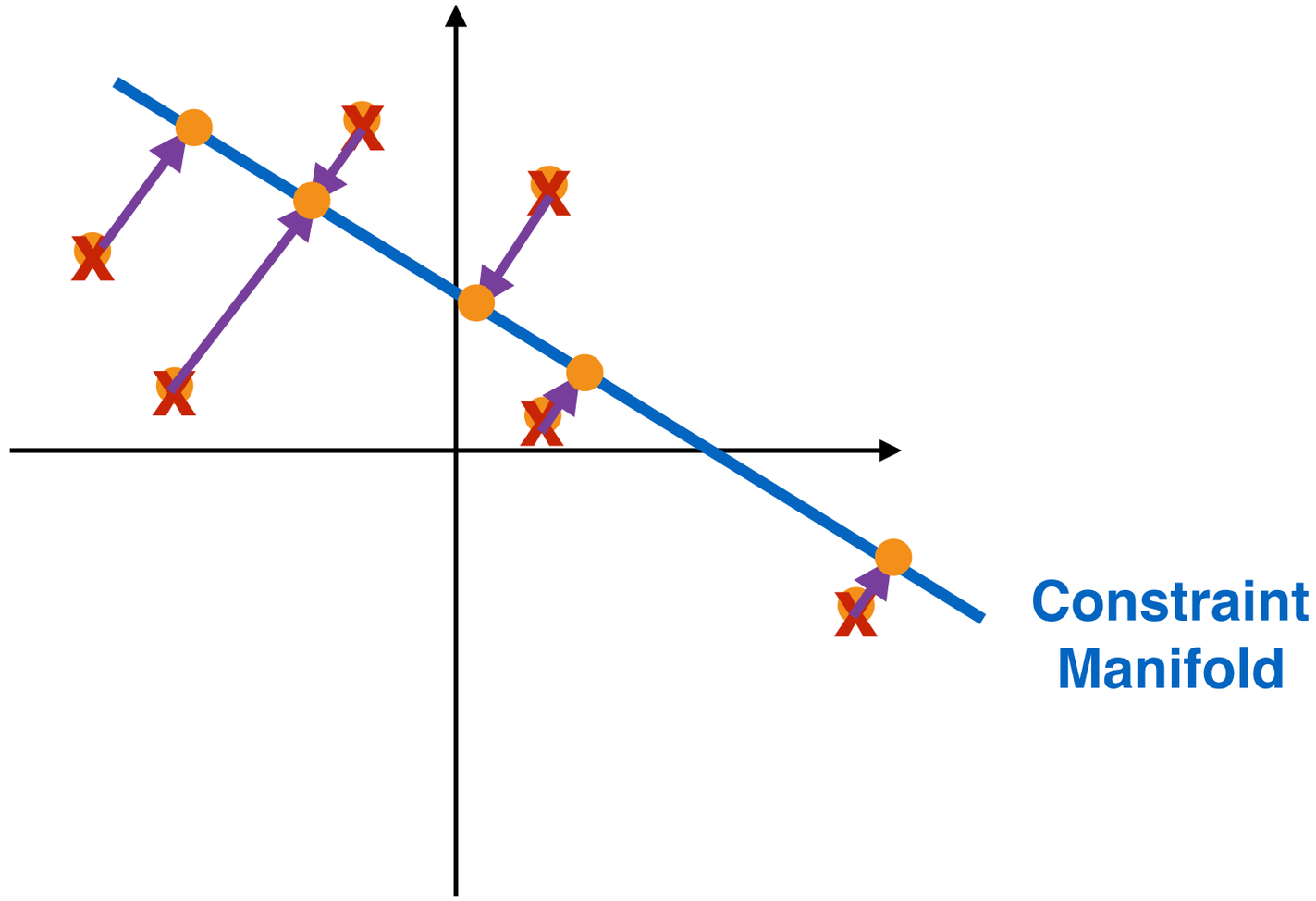
- Sample configurations until a sample is valid



- Only applicable if constraint manifold is full dimensional
- Zero probability of sampling lower dimensional space

Projection Sampling

- Projection sampling:

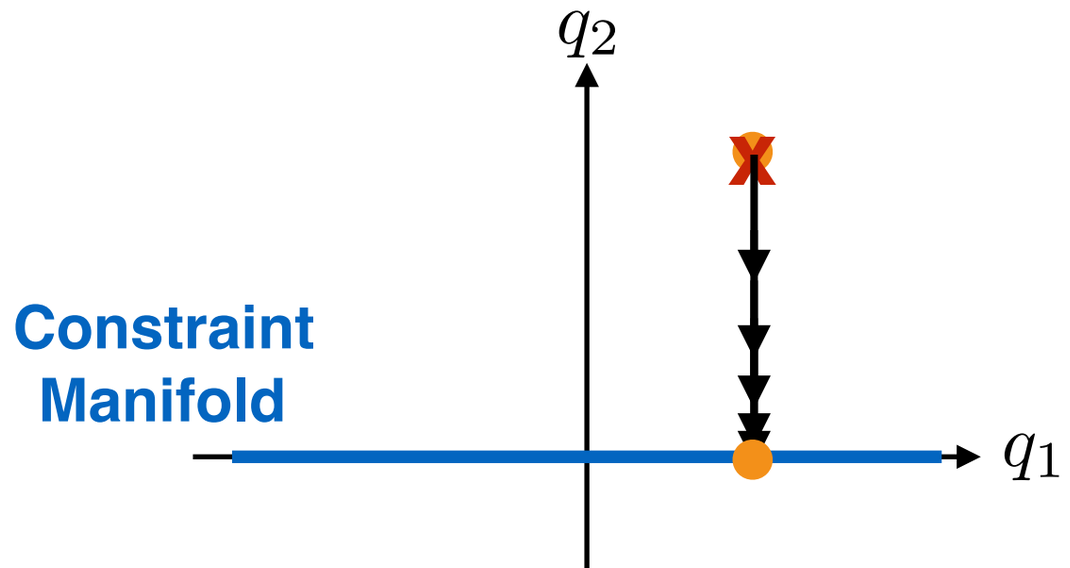


- Sample random configuration in c-space
- Project sampled into constraint manifold

Projection Sampling

- Define constraint as function s.t. $C(q) = 0$ when satisfied

$$C(q) = \frac{1}{2} q_2^2$$



- Project using gradient descent

$$\frac{\partial C}{\partial q_1} = 0 \qquad \frac{\partial C}{\partial q_2} = q_2$$

$$\nabla_q C = \begin{bmatrix} 0 \\ q_2 \end{bmatrix}$$

small step size

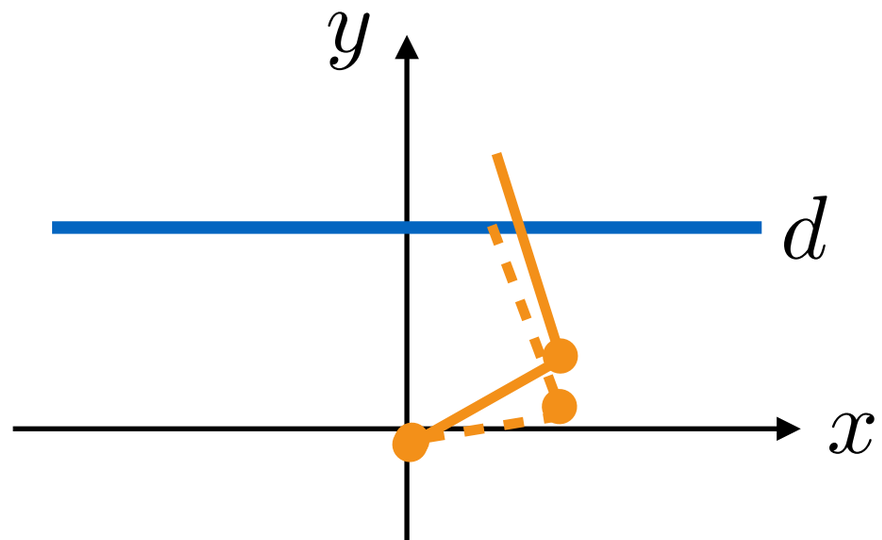
$$q_{new} = q_{prev} - \epsilon \nabla_q C$$

Repeat until $C(q) \simeq 0$

Projection Sampling

- End effector pose constraint $C(q) = (y - d)^2$
- Apply gradient descent again

$$q_{new} = q_{prev} - \epsilon \nabla_q C$$



apply chain rule

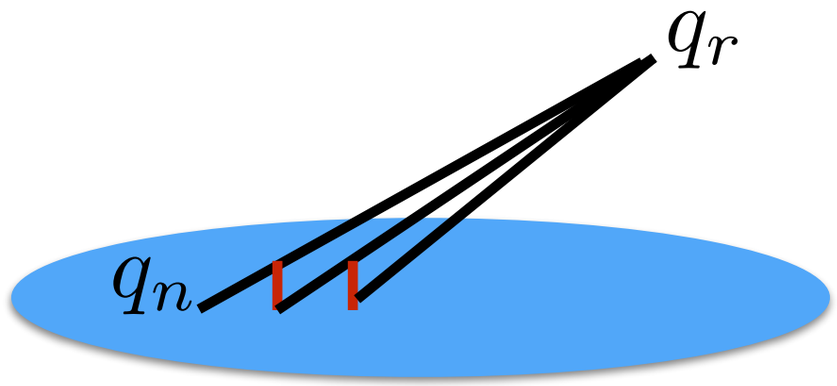
$$\nabla_q C = \left[\frac{\partial x}{\partial q}\right] \nabla_x C$$

$$= J^T \nabla_x C = J(q)^T \begin{bmatrix} 0 \\ 2y \end{bmatrix}$$

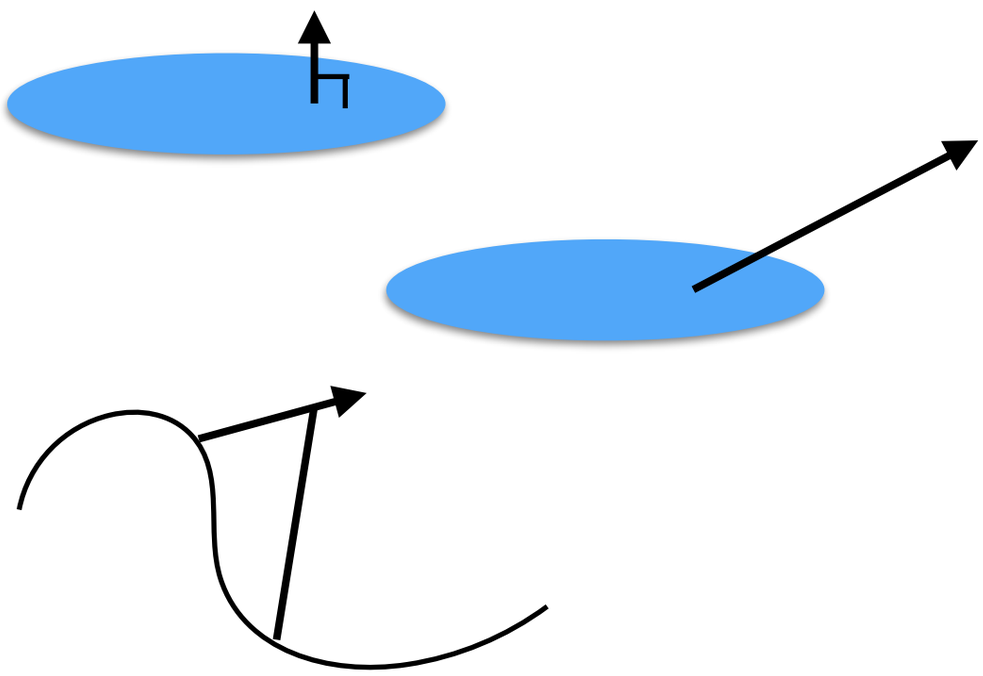
$$q_{new} = q_{prev} - \epsilon J^T \nabla_x C$$

Constrained RRT Extend Function

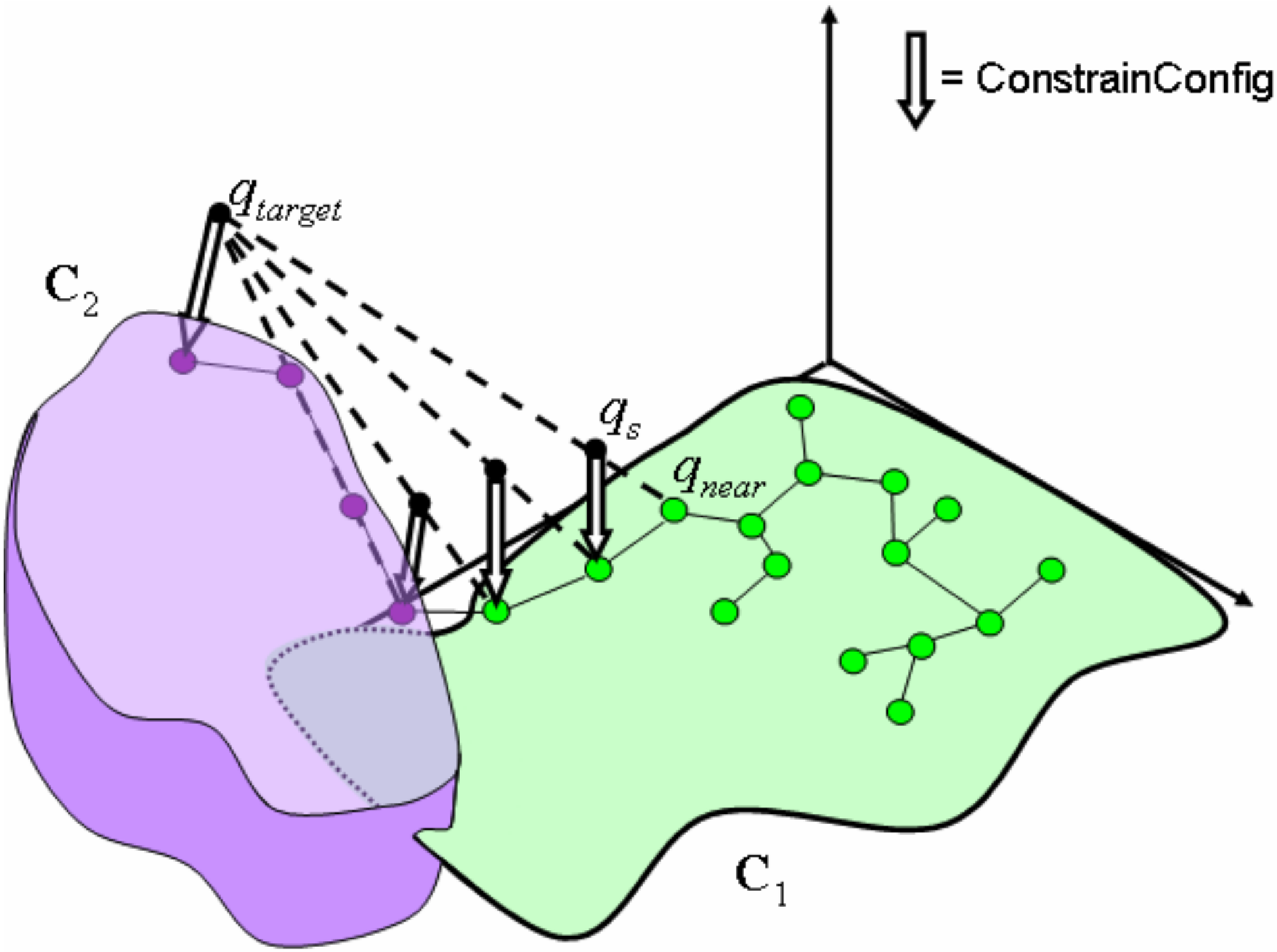
- Extend from q_n to q_r
  - Take a step towards q_r
  - Project until $C(q) \simeq 0$
  - Repeat



- Repeat until:
  - No progress
  - Projection fails
  - Projected point is too far away



Constrained RRT Extend Function



Berenson et al. Manipulation Planning on Constraint Manifolds, ICRA 2009

Constrained RRT Extend Function

Algorithm 2: ConstrainedExtend(T, q_{near}, q_{target})

1 $q_s \leftarrow q_{near}$; $q_s^{old} \leftarrow q_{near}$;
2 **while** *true* **do**
3 **if** $q_{target} = q_s$ **then**
4 **return** q_s;
5 **else if** $|q_{target} - q_s| > |q_s^{old} - q_{target}|$ **then**
6 **return** q_s^{old};
7 **end**
8 $q_s^{old} \leftarrow q_s$;
9 $q_s \leftarrow q_s + \min(\Delta q_{step}, |q_{target} - q_s|) \frac{(q_{target} - q_s)}{|q_{target} - q_s|}$;
10 $q_s \leftarrow \text{ConstrainConfig}(q_s^{old}, q_s)$;
11 **if** $q_s \neq \text{NULL}$ ***and*** $\text{CollisionFree}(q_s^{old}, q_s)$ **then**
12 T.AddVertex(q_s);
13 T.AddEdge(q_s^{old}, q_s);
14 **else**
15 **return** q_s^{old};
16 **end**
17 **end**

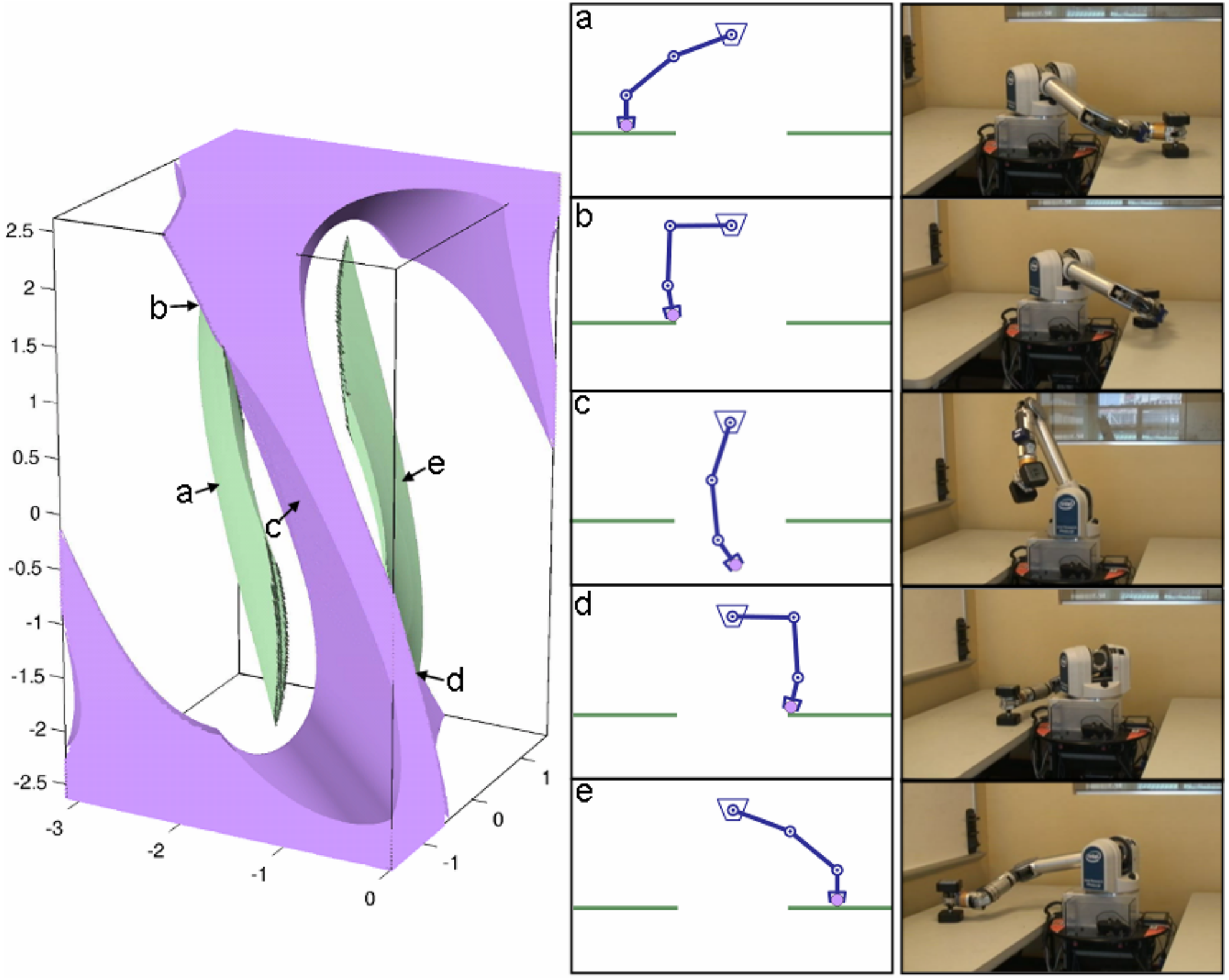
Reached target

Moving further away

Make (limited) step

Add new vertex if possible

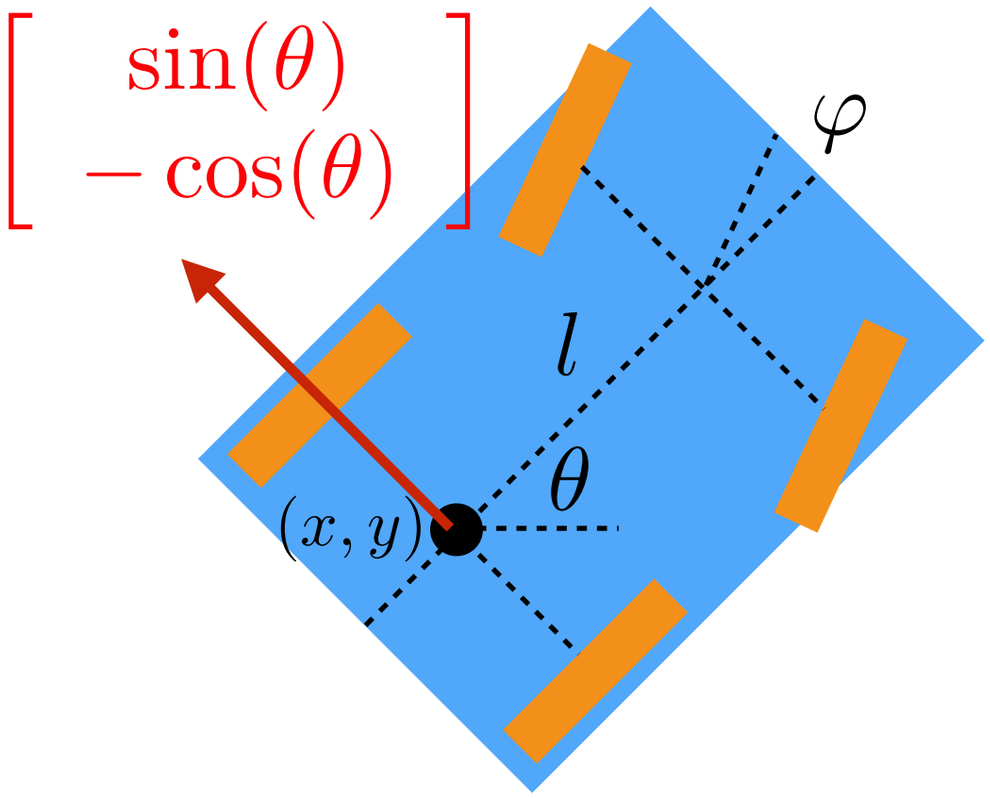
Constrained BiRRT



Planning with Differential Constraints

Car Example

- Consider a simple car model (not modeling dynamics)



State (configuration)

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Controls

$$u = \begin{bmatrix} v \\ \varphi \end{bmatrix}$$

Differential Constraint:

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = 0$$

System Equation

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{l}\tan(\varphi) \end{bmatrix}$$

Differential Constraints

- Tangent space generated by controls is given by

$$Q_T = \{\dot{q} | \dot{q} = f(q, u), u \in U\}$$

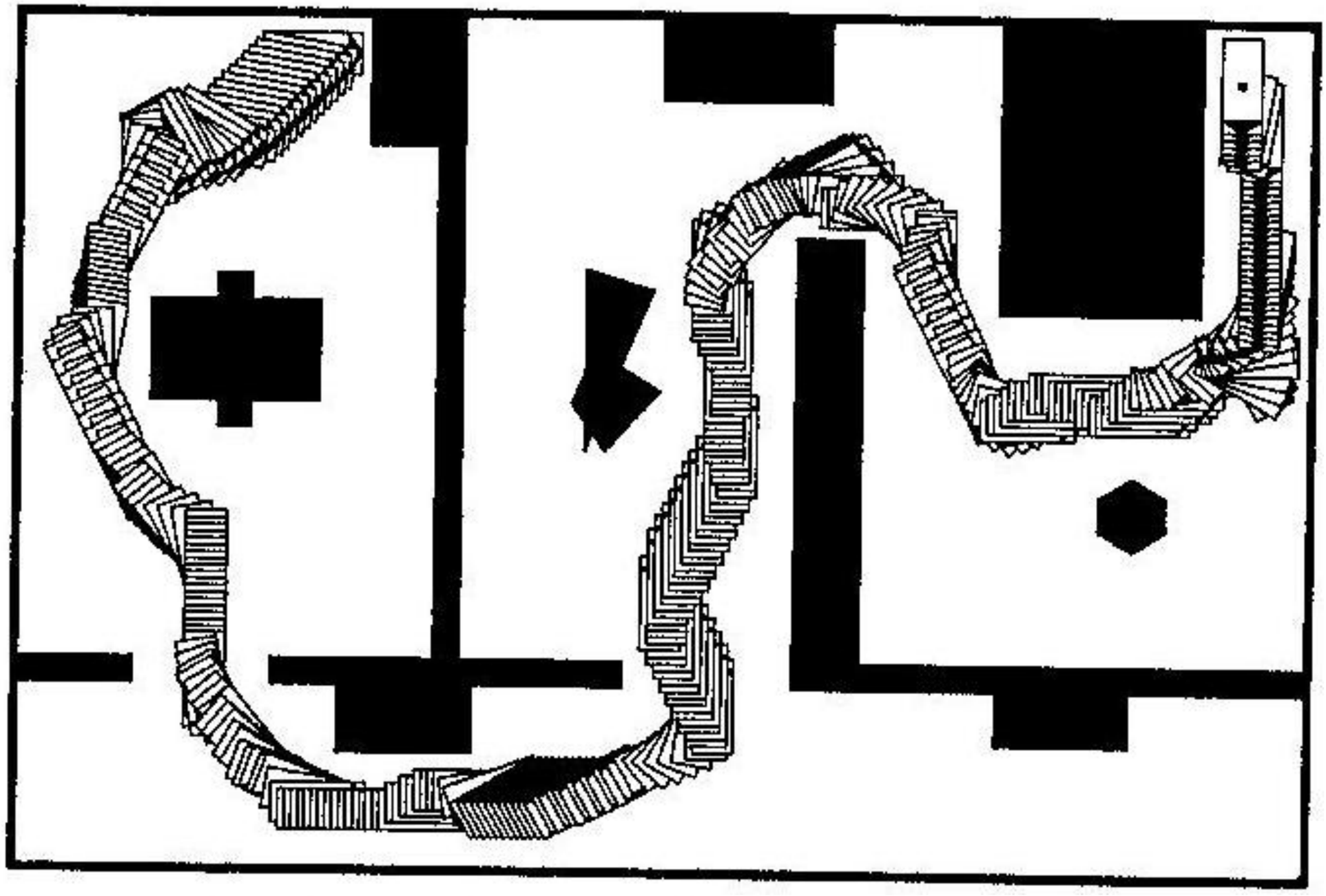
- For a differential constraint we get

$$\dim(Q_T) < \dim(q)$$

  - Not all degrees of freedom are directly controllable

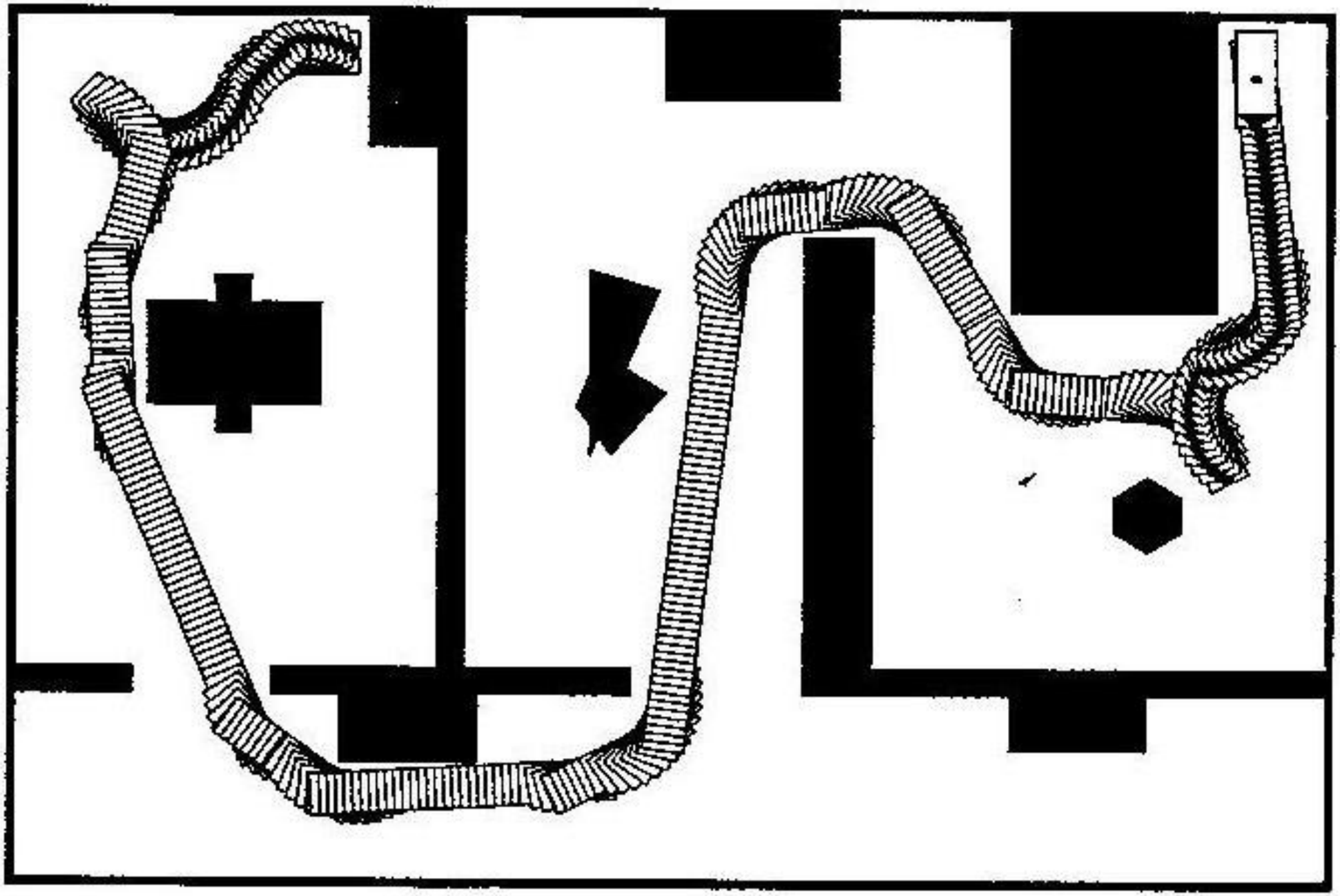
- Differential constraint implies a non-holonomic system

Car Example Plan



IGNORING the differential constraint

Car Example Plan



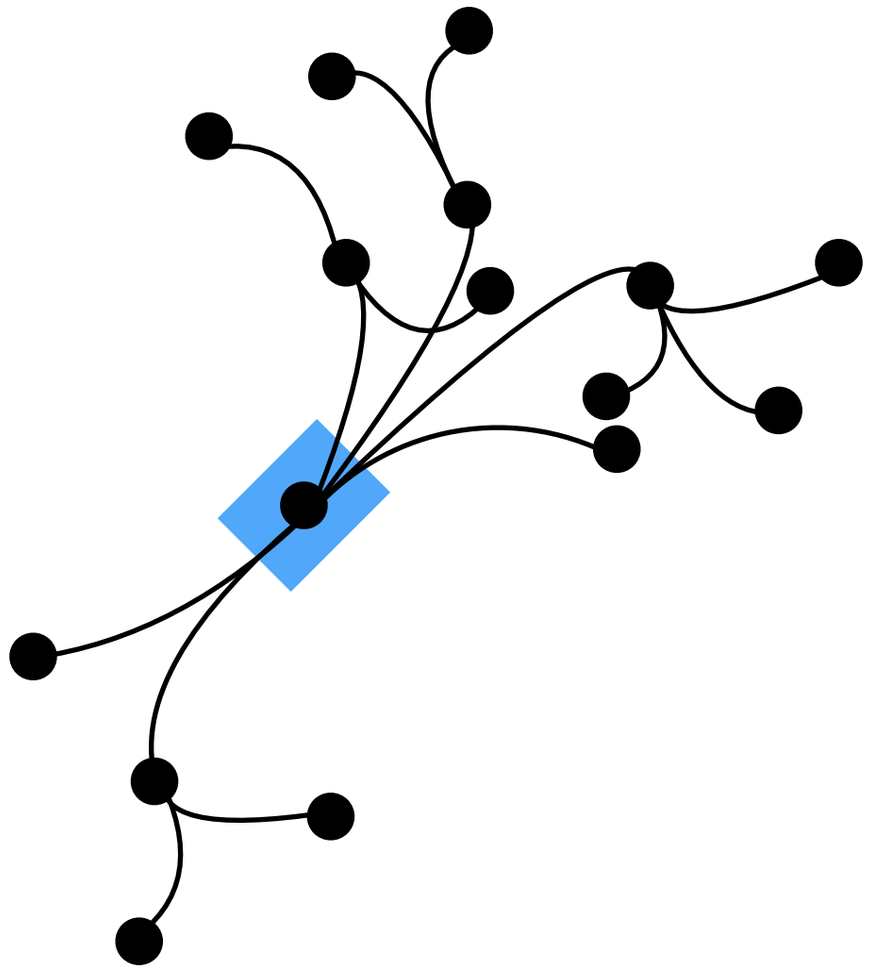
with the differential constraint

Basic Control-based Tree Growing

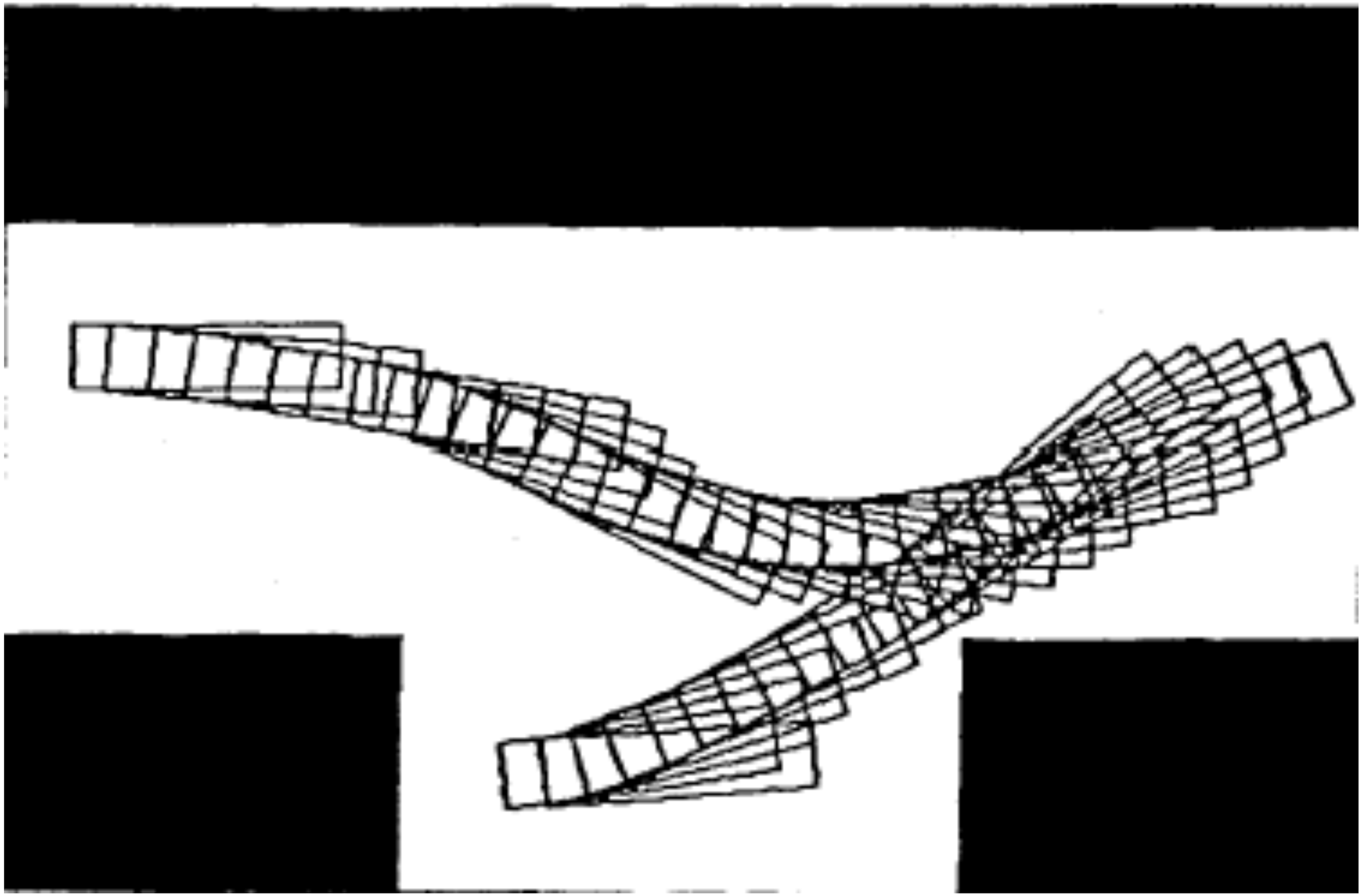
- Want to generate a tree for planning (not RRT yet!)
- Initialize tree T with start configuration $V = q_i$
- Grow tree by repeatedly:
  - Sample a configuration from the tree $q \in T$
  - Sample a random control input u
  - Integrate system equation over short duration (random/fixed)
  - Add vertex and edge if resulting motion is collision free
- Terminate once tree is within small distance of q_g

Basic Control-based Tree Growing

- Creates a tree structure for planning



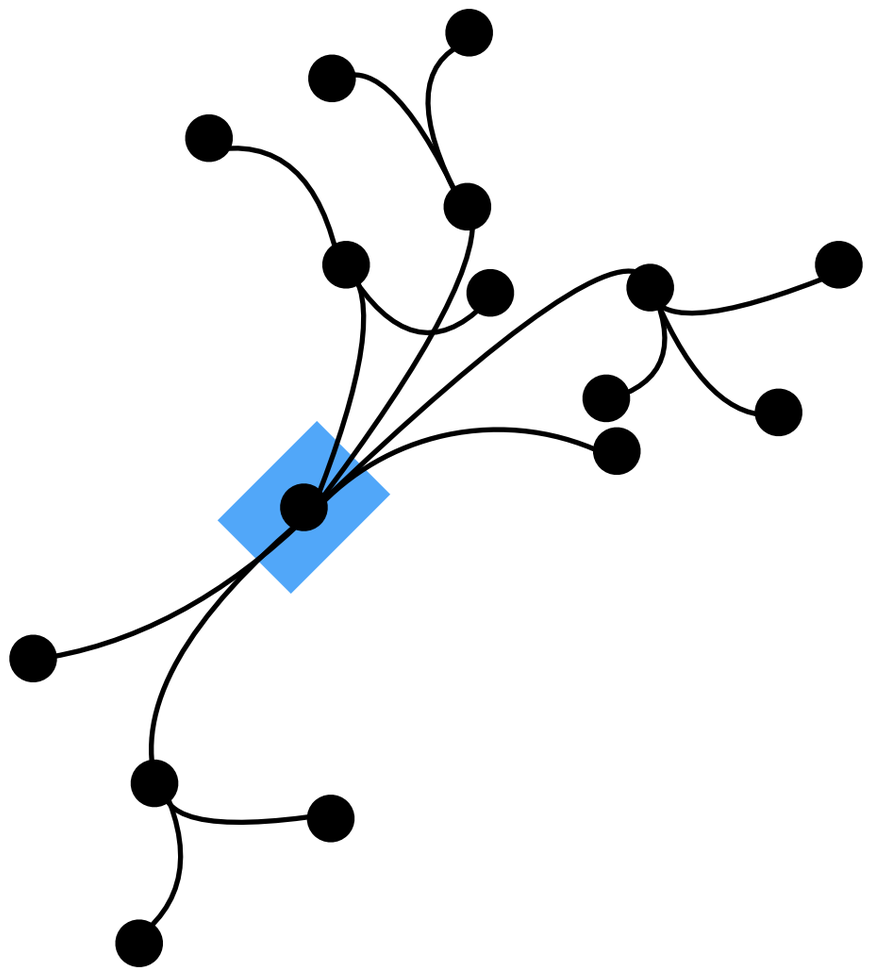
Parking a Car Example



Barraquand and Latombe, Nonholonomic Multibody Mobile Robots, Algorithmica 1993

Basic Control-based Tree Growing

- Creates a tree structure for planning

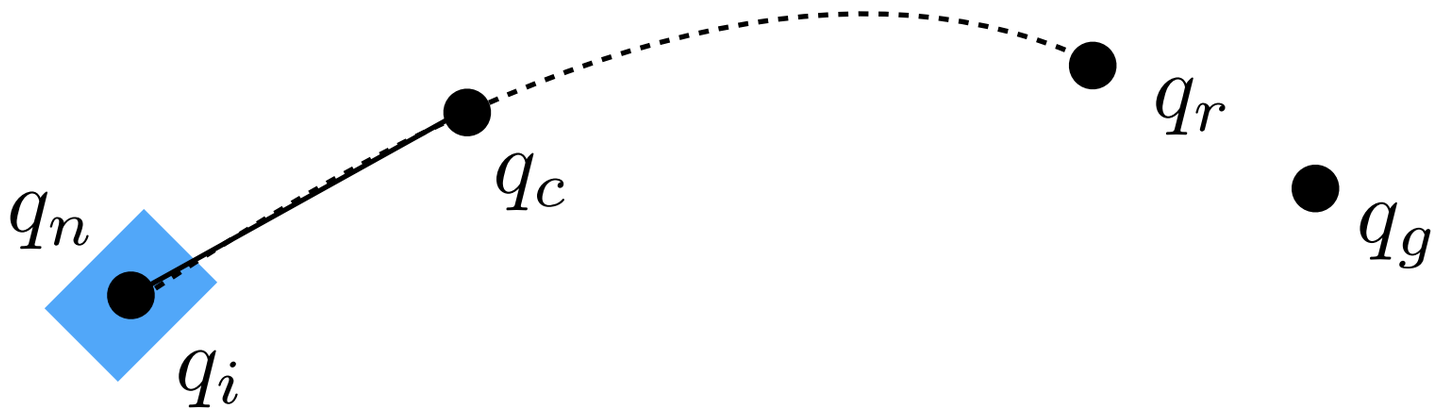


- Not rapidly exploring though
  - Want to bias growth towards unexplored regions - RRTs!

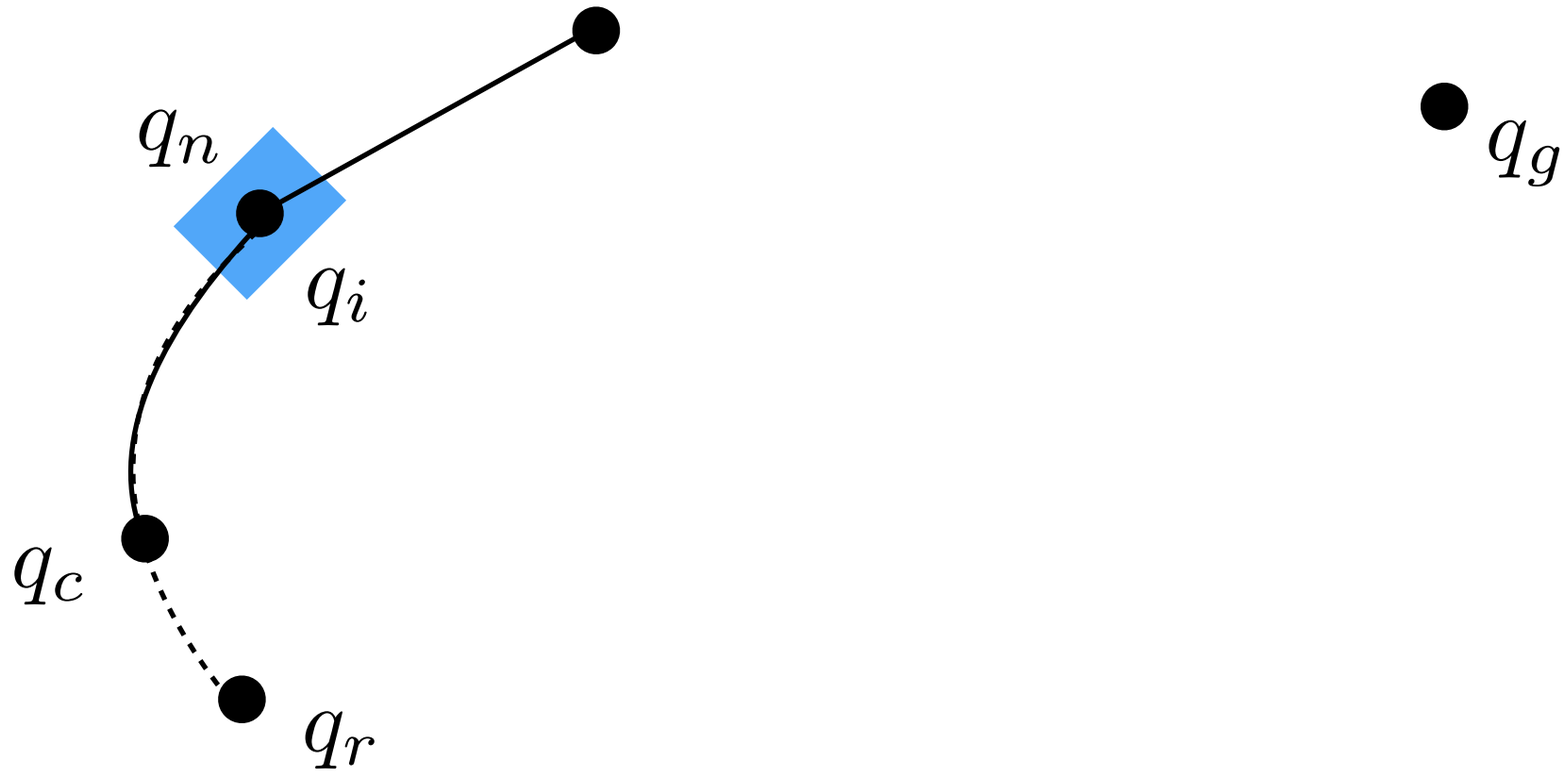
RRTs with Kinodynamic Constraints

- Initialize tree T with start configuration $V = q_i$
- Grow tree by repeatedly:
  - Sample a random configuration q_r from C
  - Find nearest neighbour $q_n \in V$ to q_r
  - Use local planner to compute u to steer from q_n towards q_r
  - Integrate system equation over short duration (random/fixed)
  - Add vertex and edge if resulting motion is collision free
- Terminate once tree is within small distance of q_g

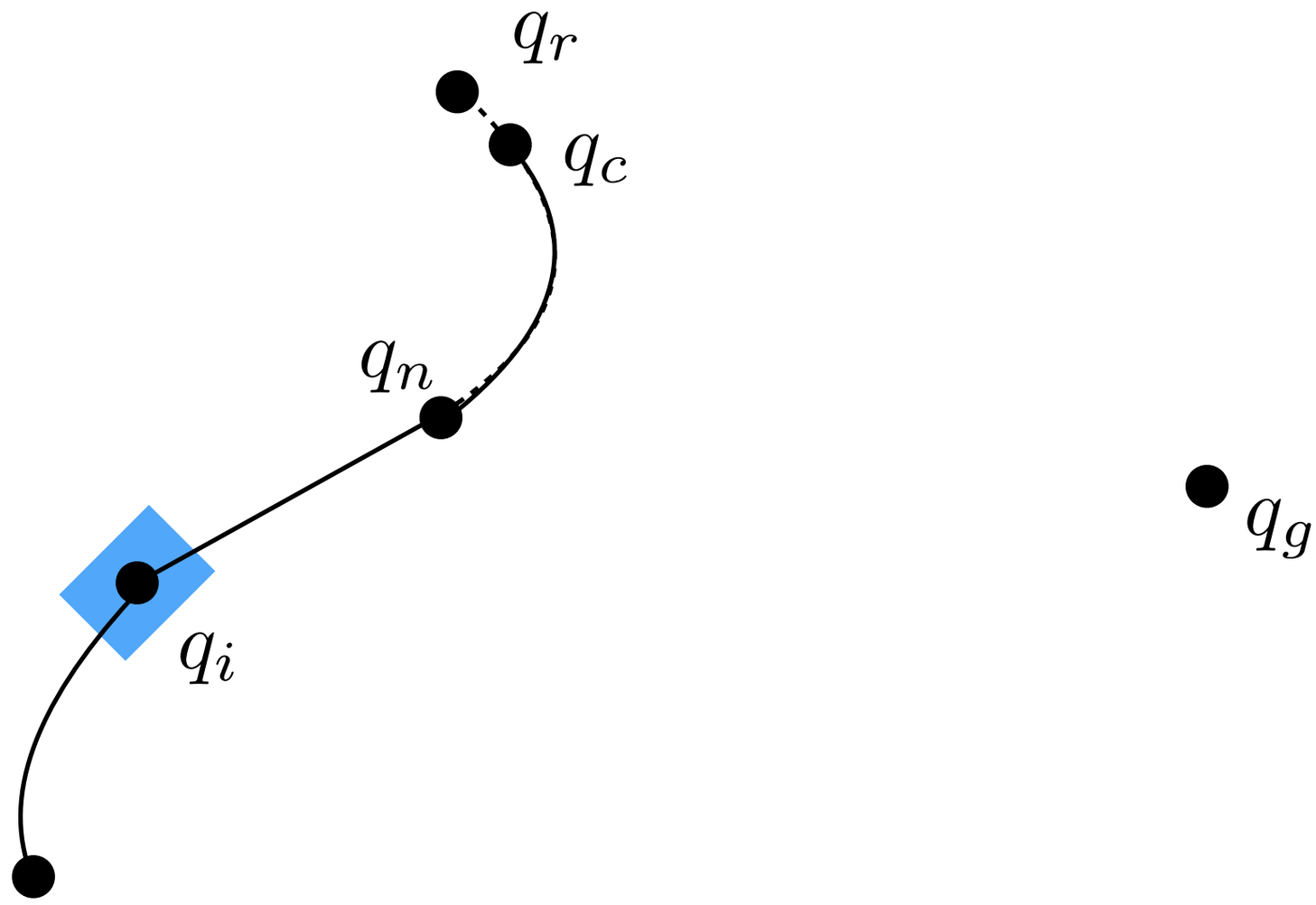
RRTs with Kinodynamic Constraints



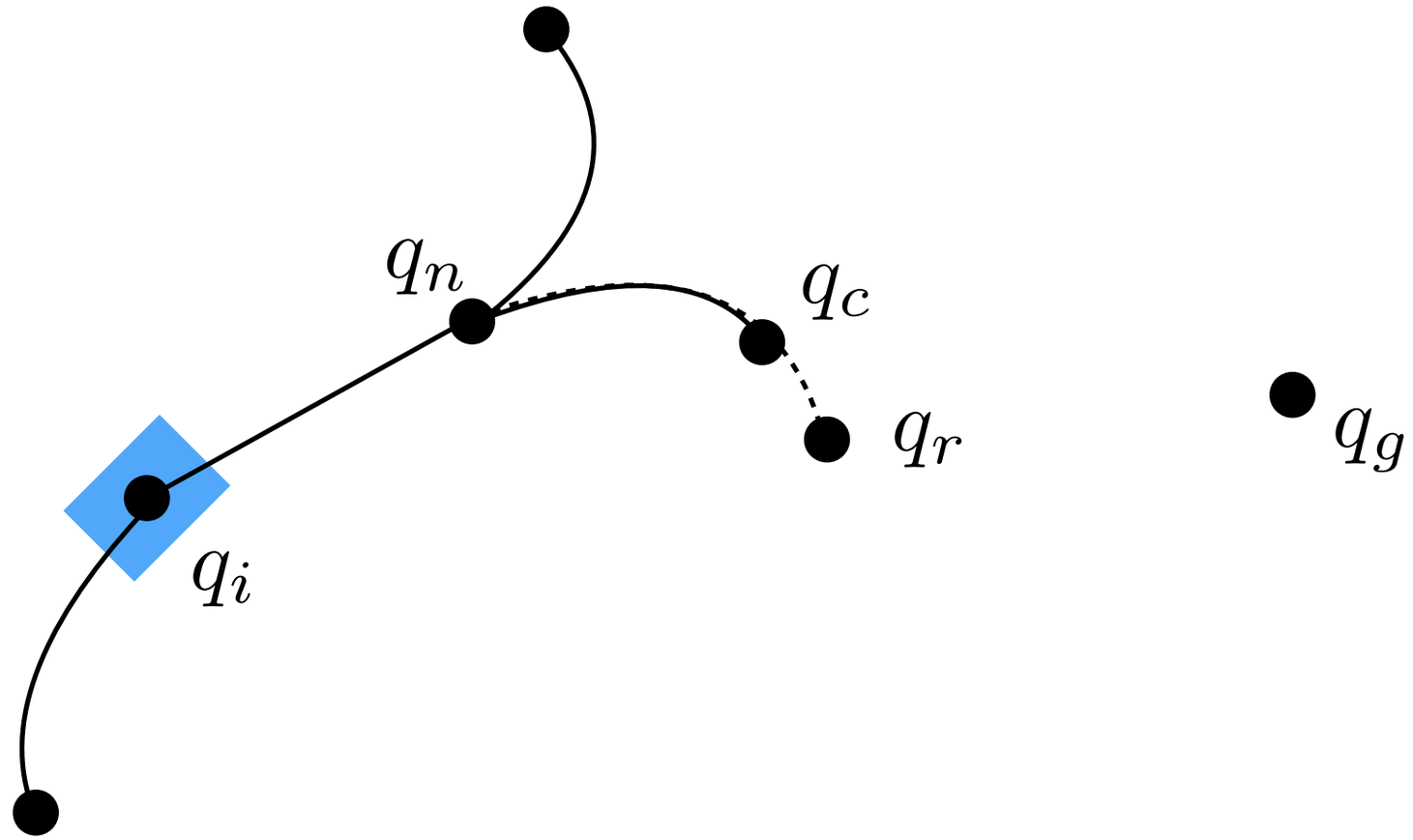
RRTs with Kinodynamic Constraints



RRTs with Kinodynamic Constraints



RRTs with Kinodynamic Constraints

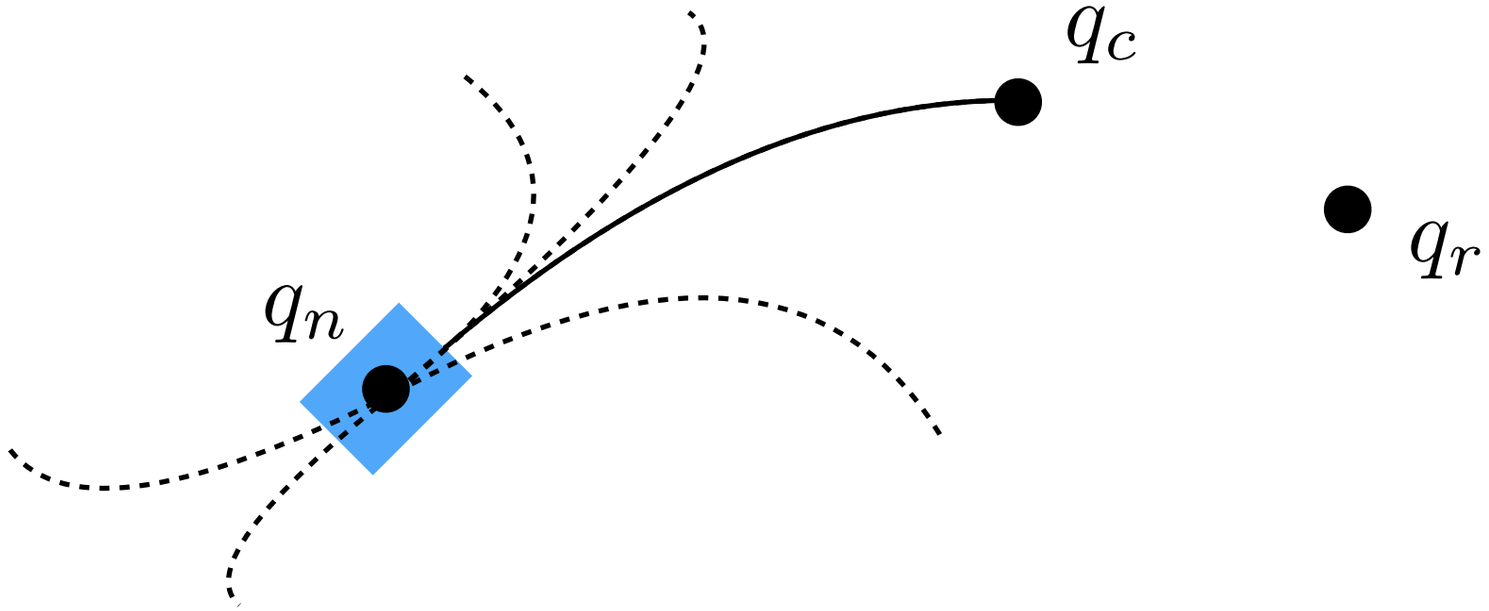


Nearest Neighbours

- Selecting "nearest neighbour" is not trivial
  - May be next to each other but cannot move there
- Naive/standard approaches use Euclidean metrics
- Ideal metric would be cost-to-go from q_n to q_r
  - Effectively as difficult as original problem…
  - Can approximate it, e.g. ignore obstacles

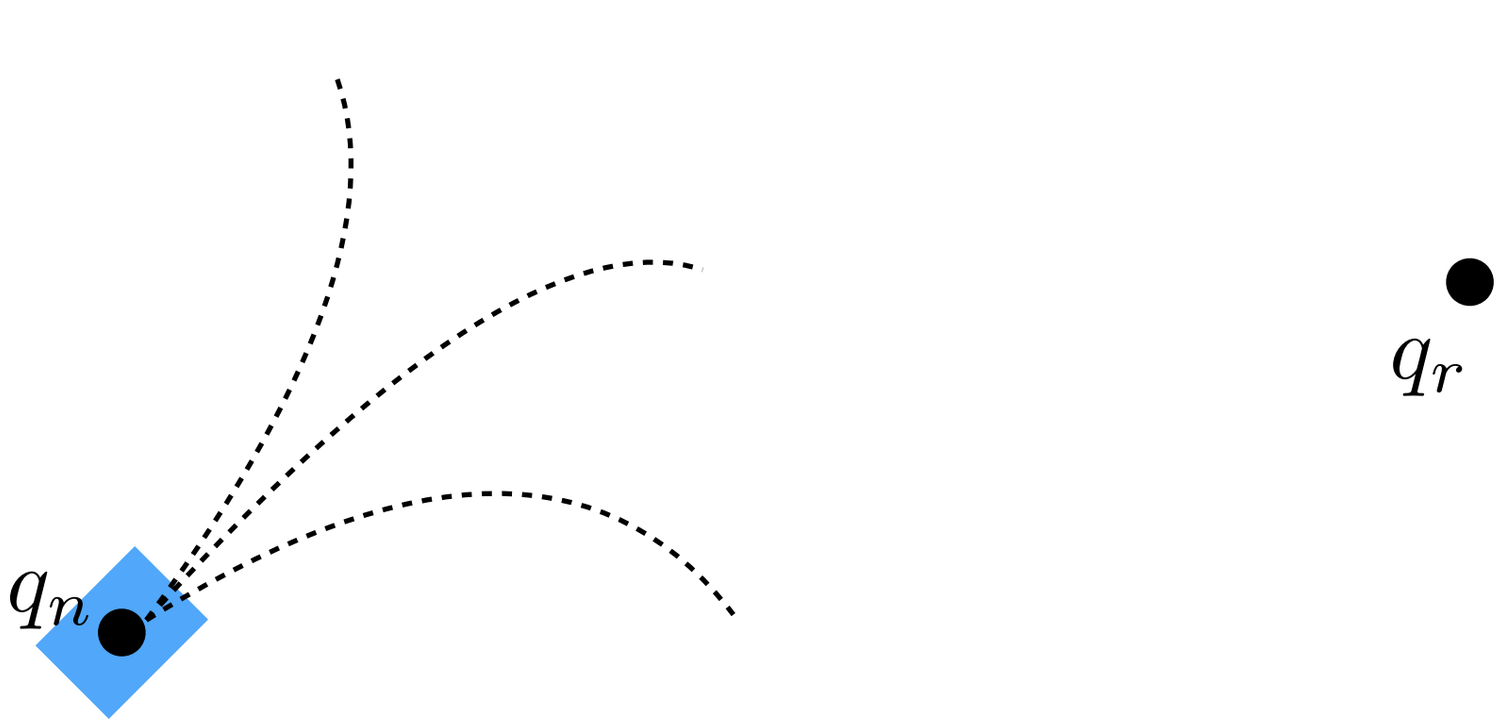
Shooting Algorithm

- Need to find path from q_n to q_r s.t. $\dot{q} = f(q, u)$
- Solving for the control input is difficult
- Approximate with shooting algorithm

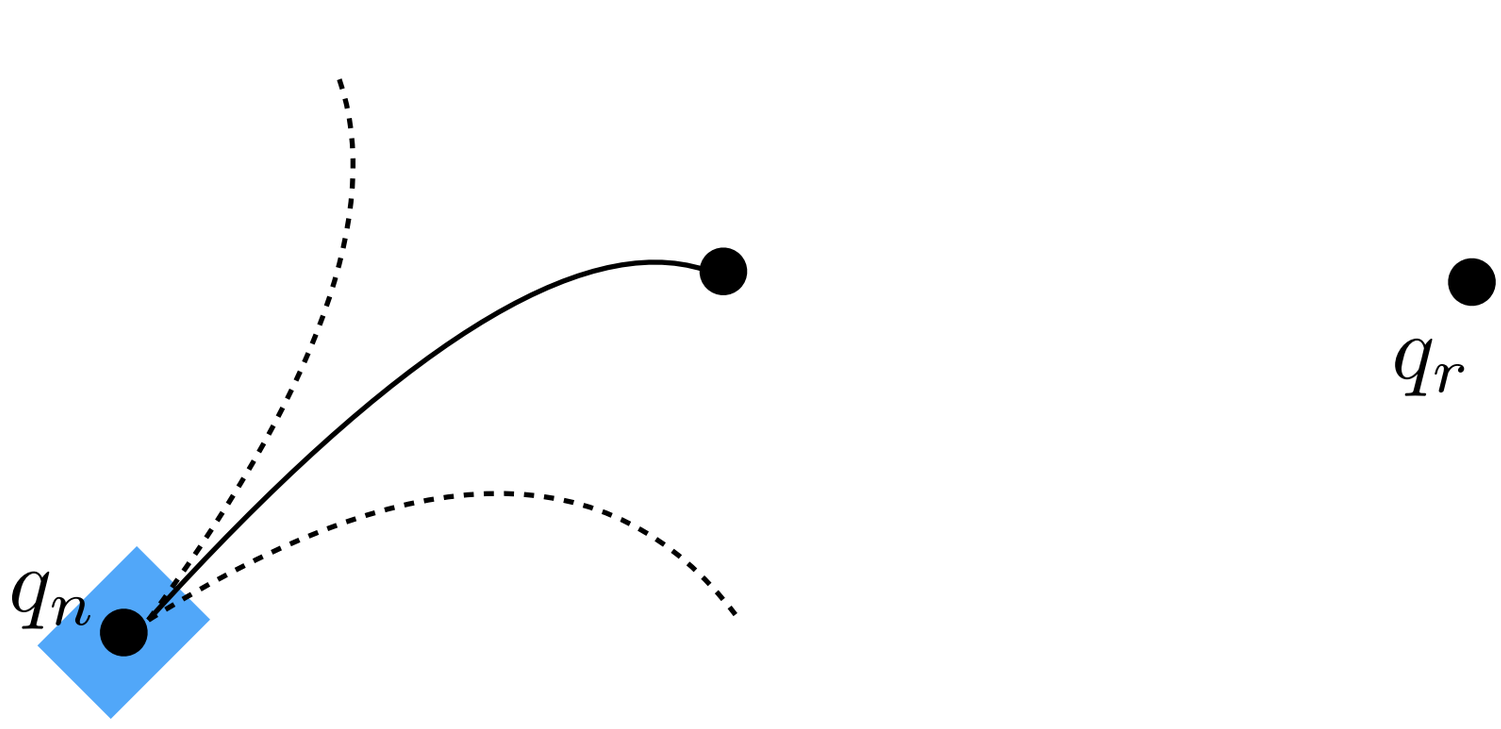


- Sample multiple trajectories from q_n with different controls u
- Select the one that gets the closest to q_r

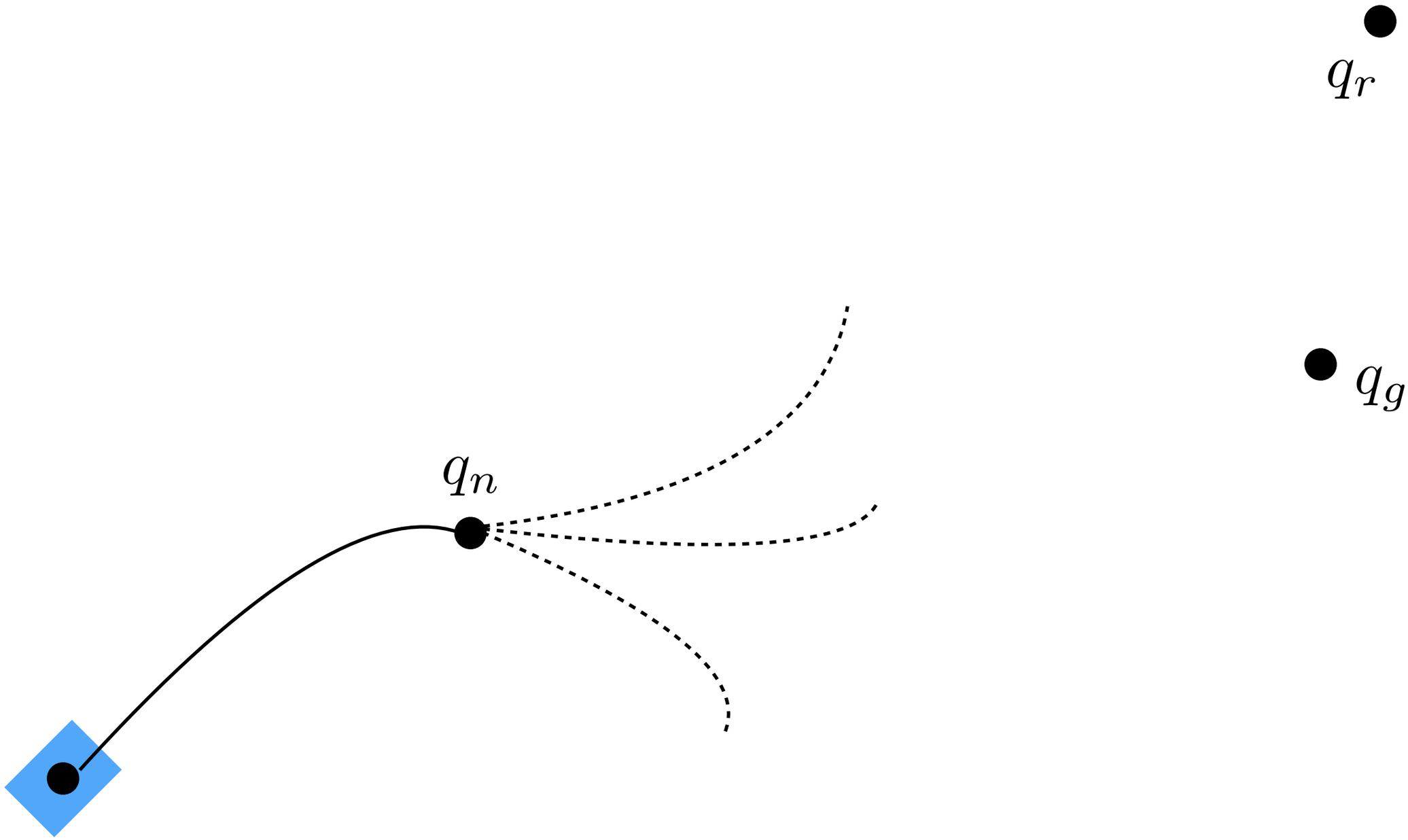
RRT with Shooting Approach



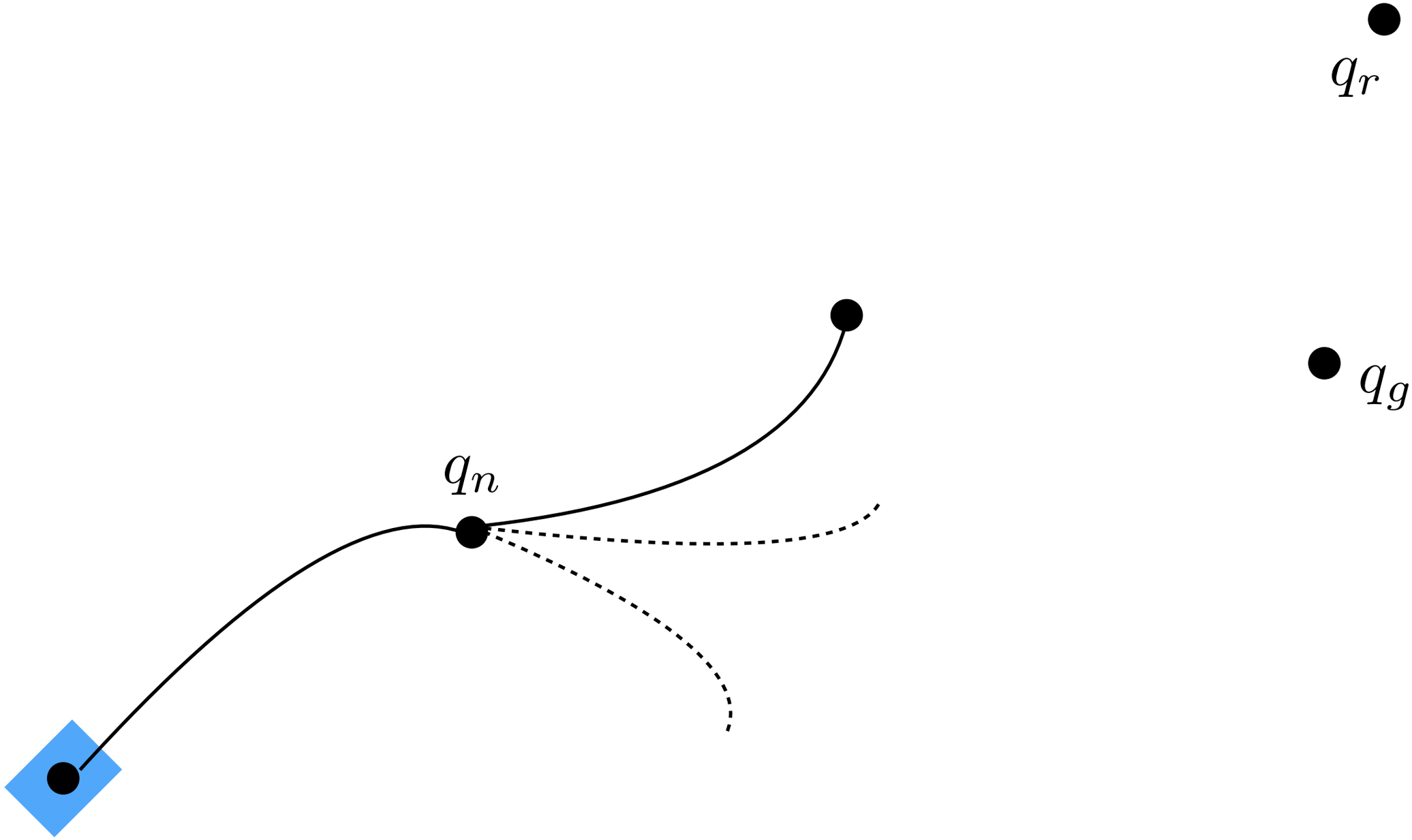
RRT with Shooting Approach



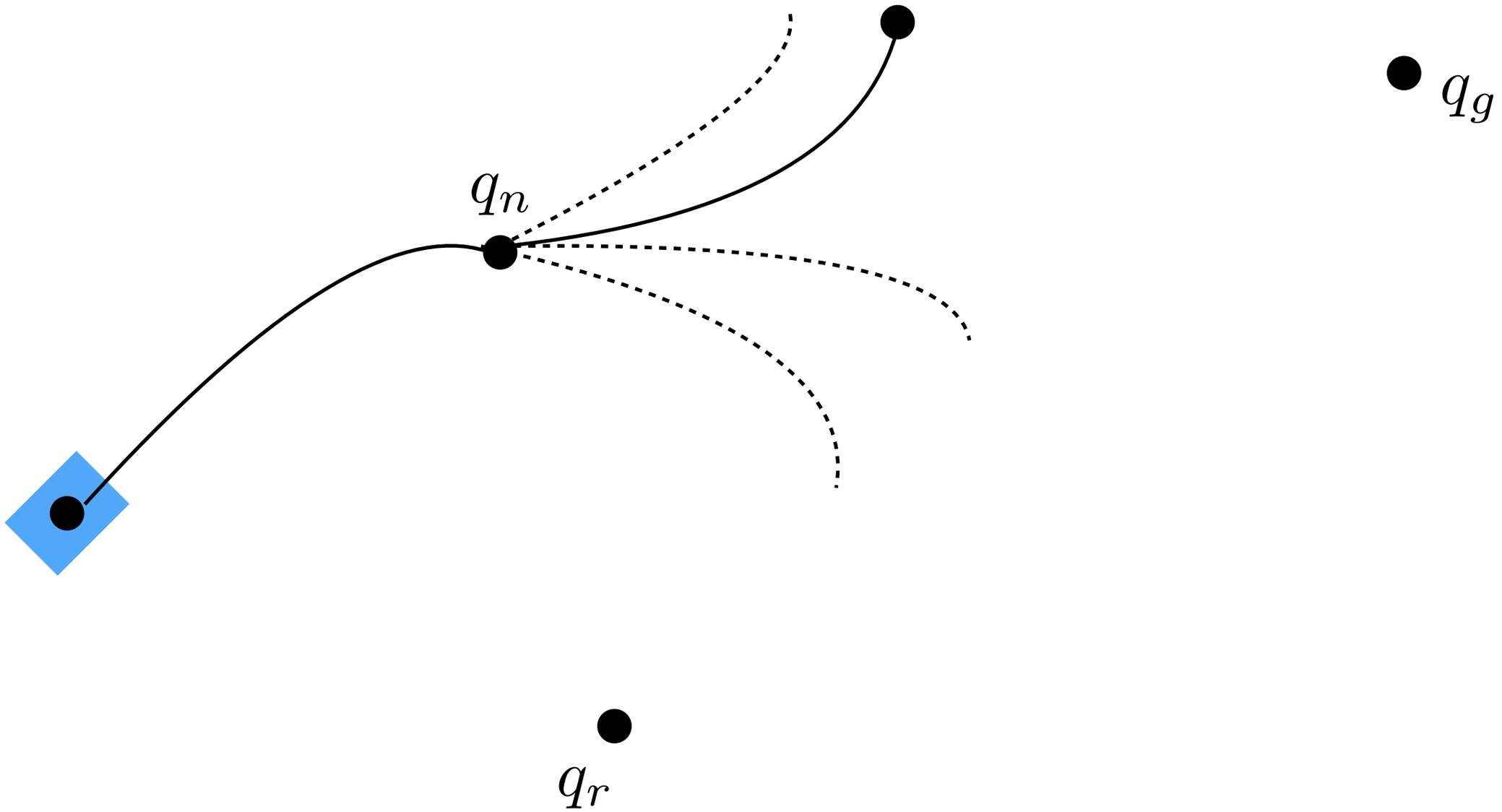
RRT with Shooting Approach



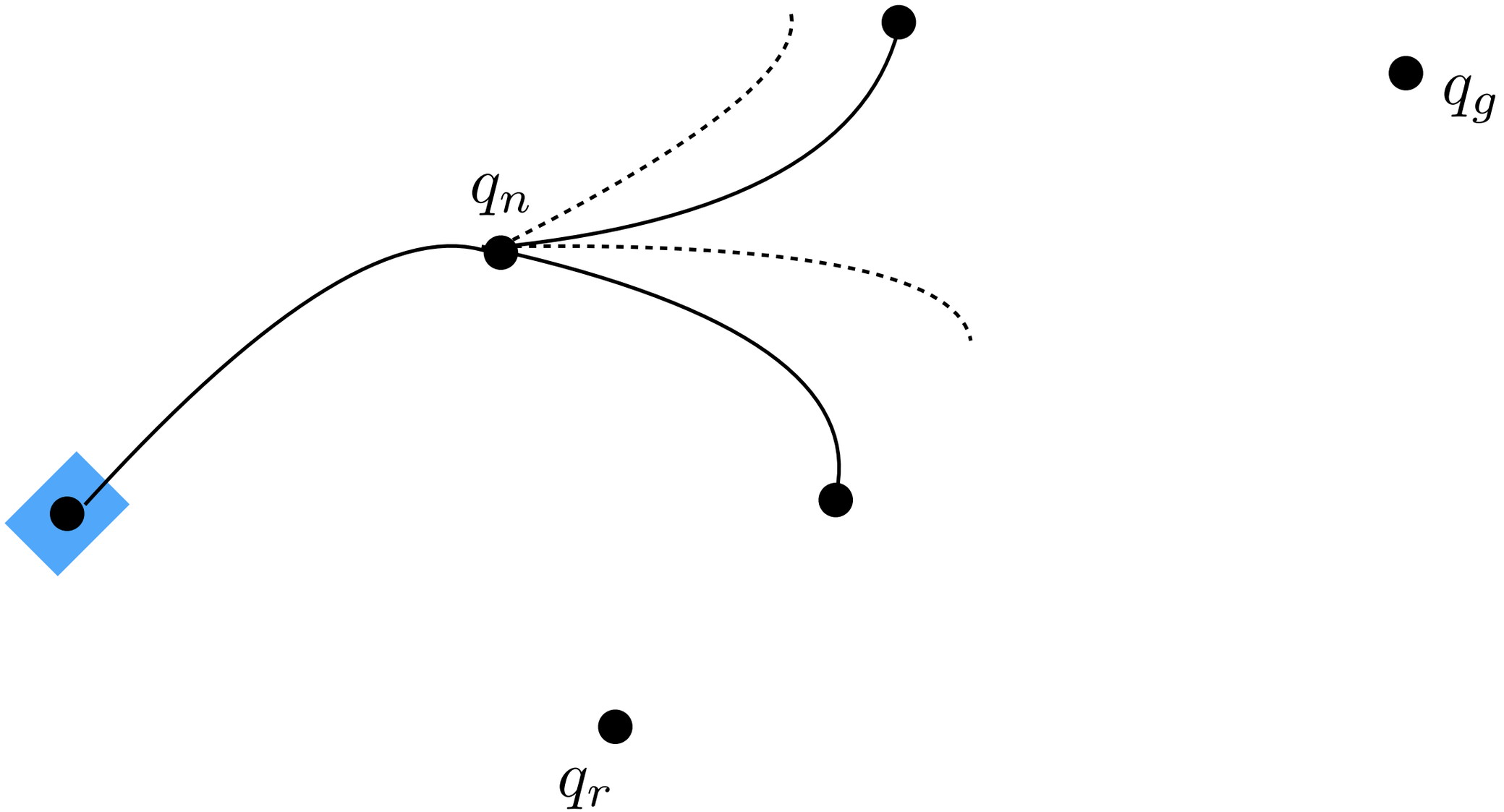
RRT with Shooting Approach



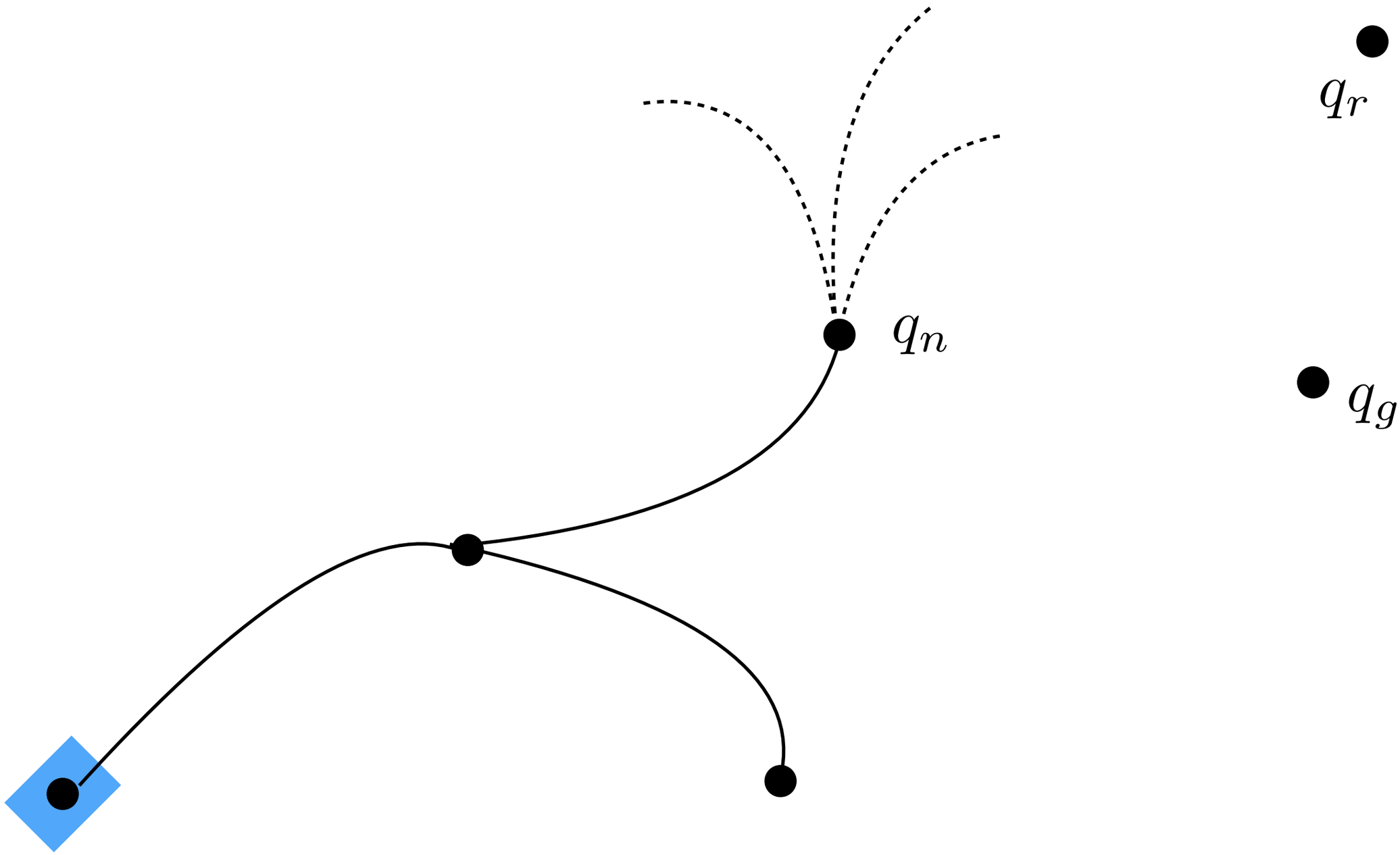
RRT with Shooting Approach



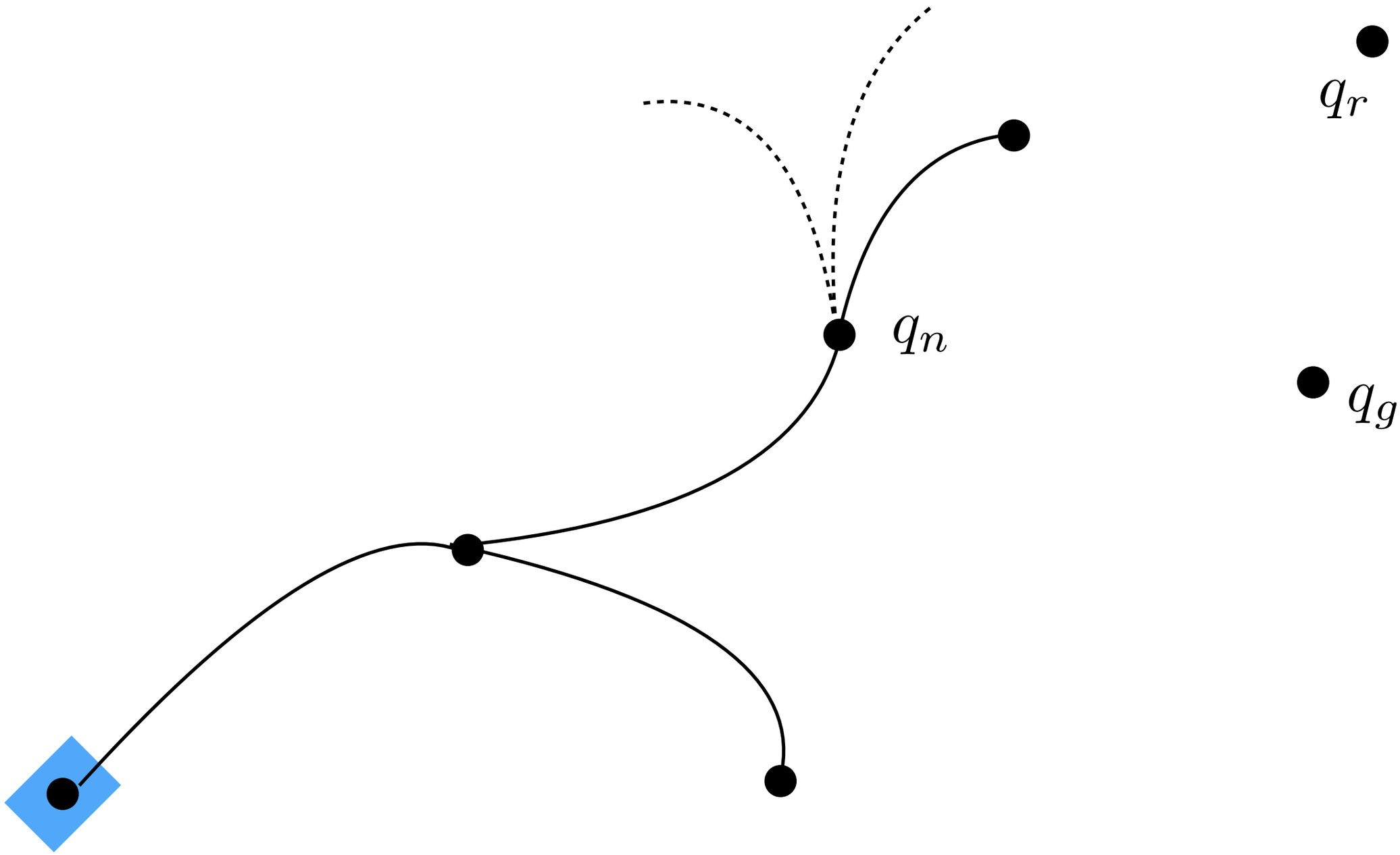
RRT with Shooting Approach



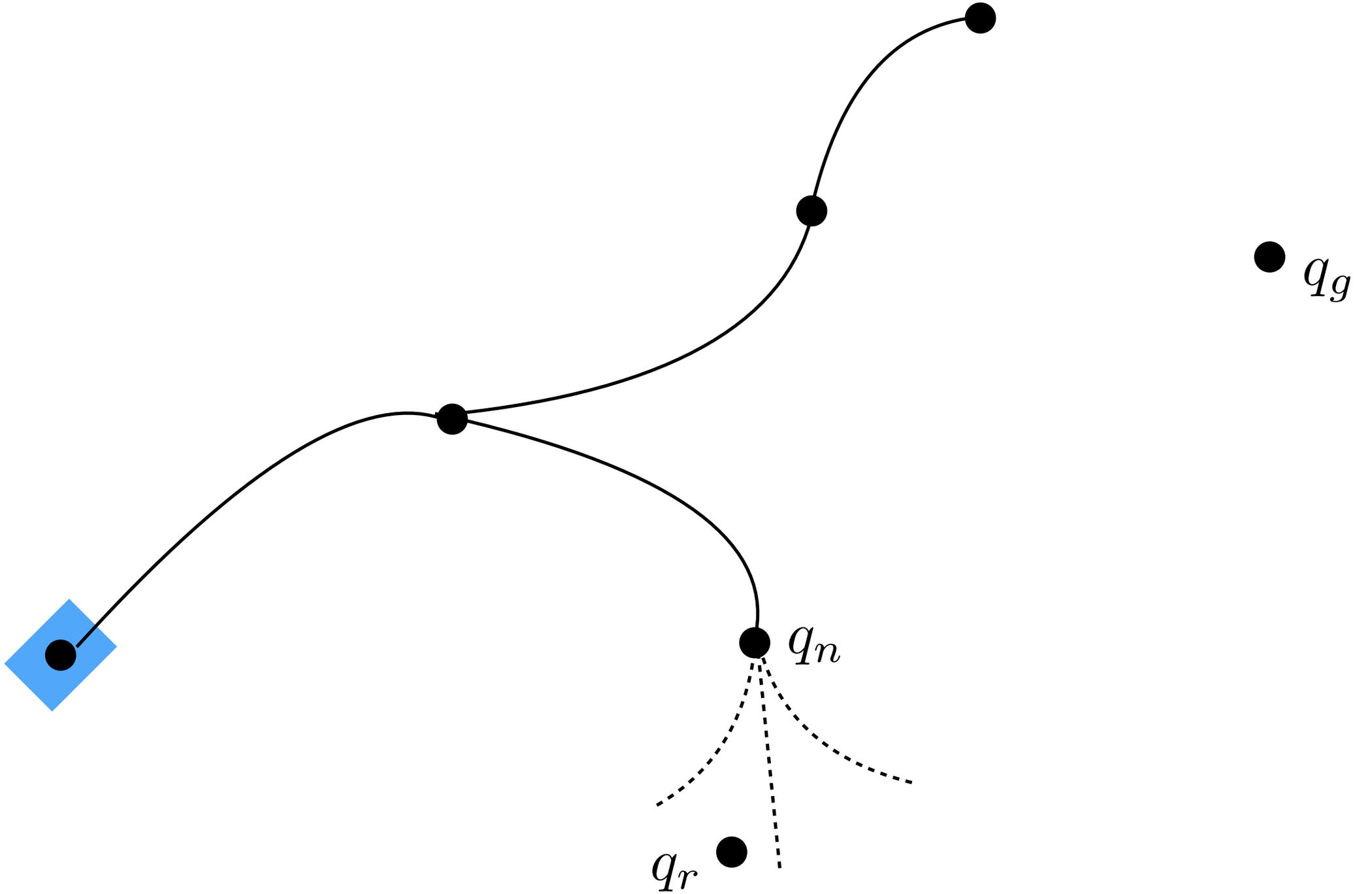
RRT with Shooting Approach



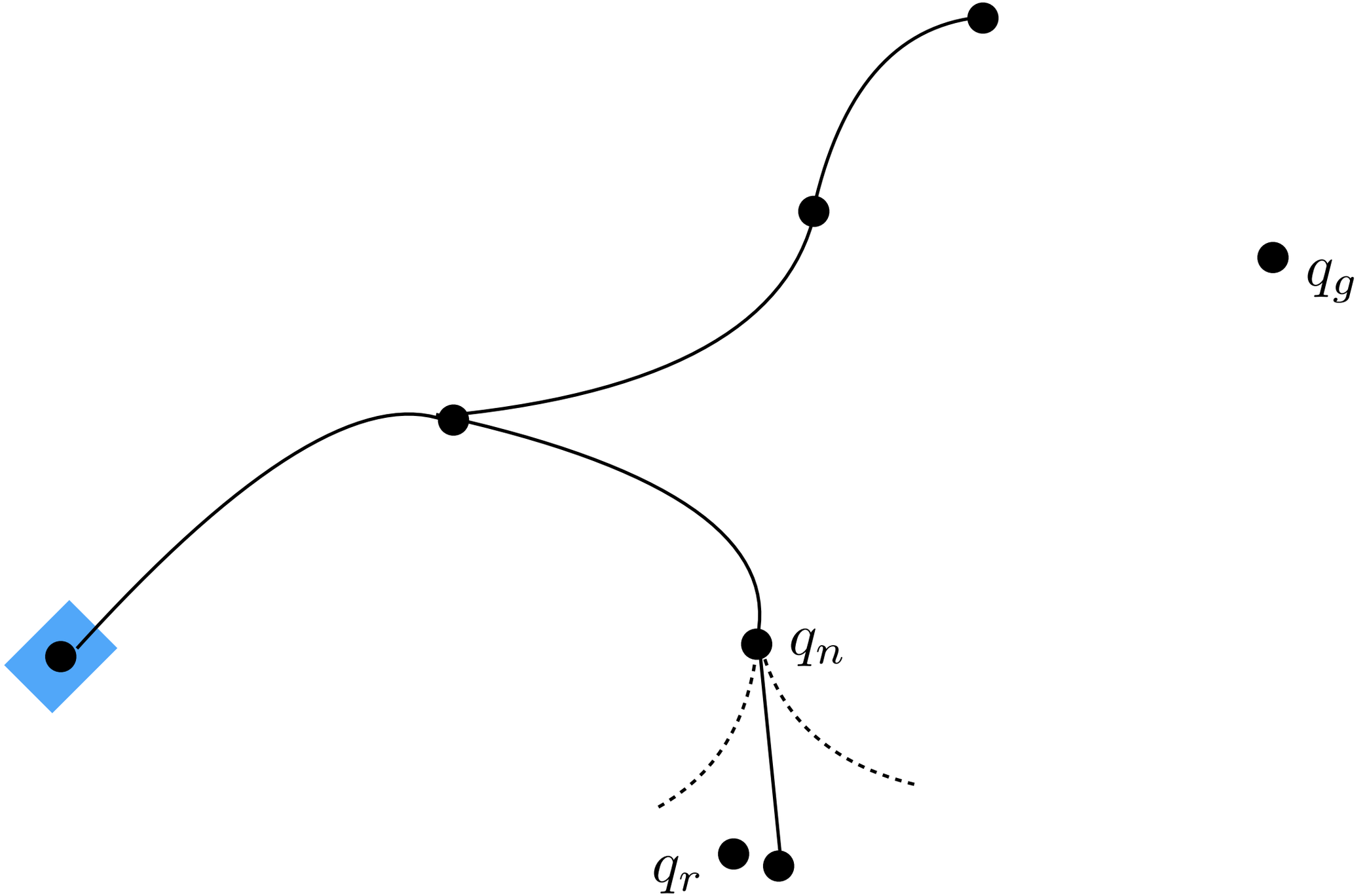
RRT with Shooting Approach



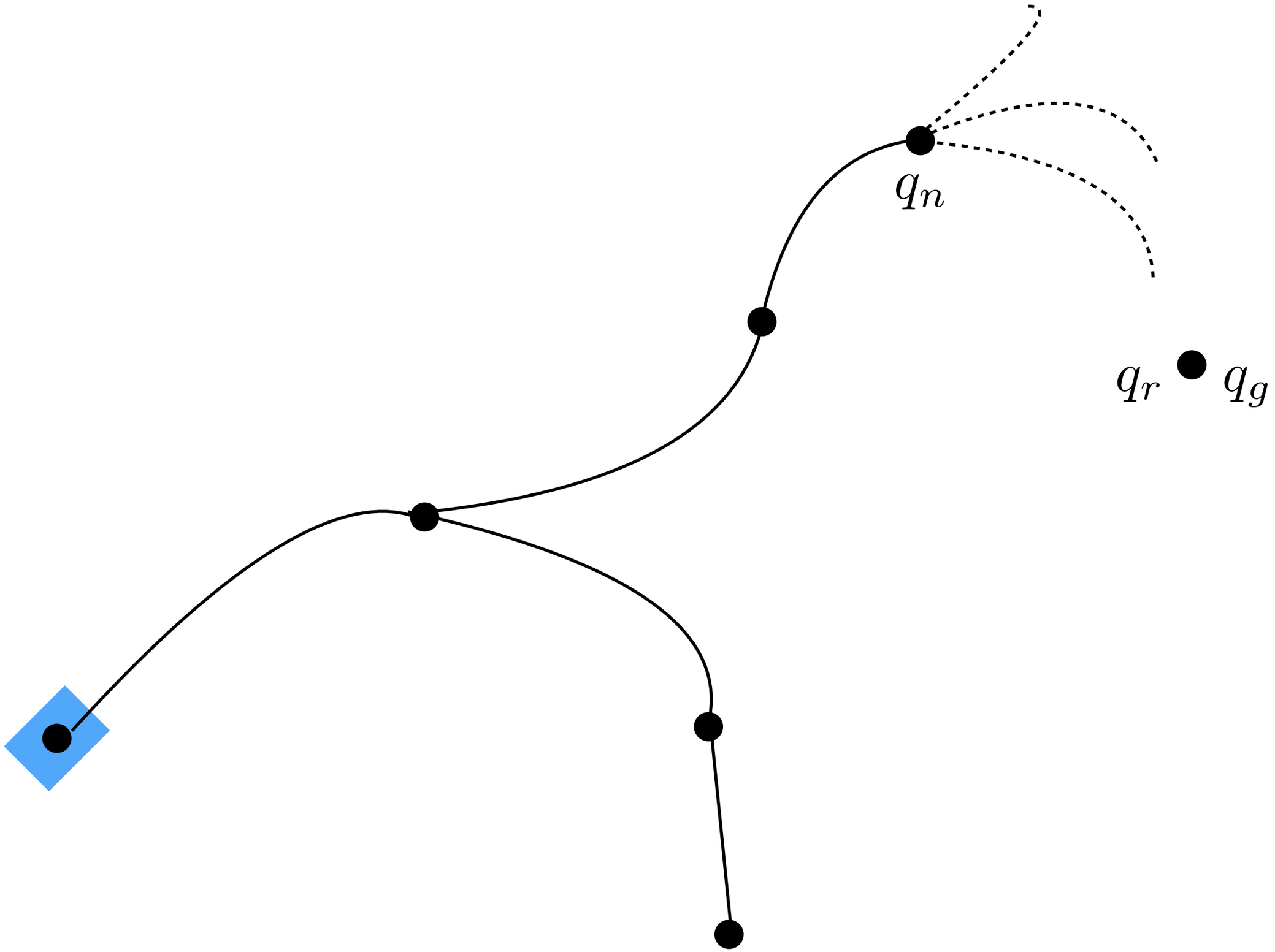
RRT with Shooting Approach



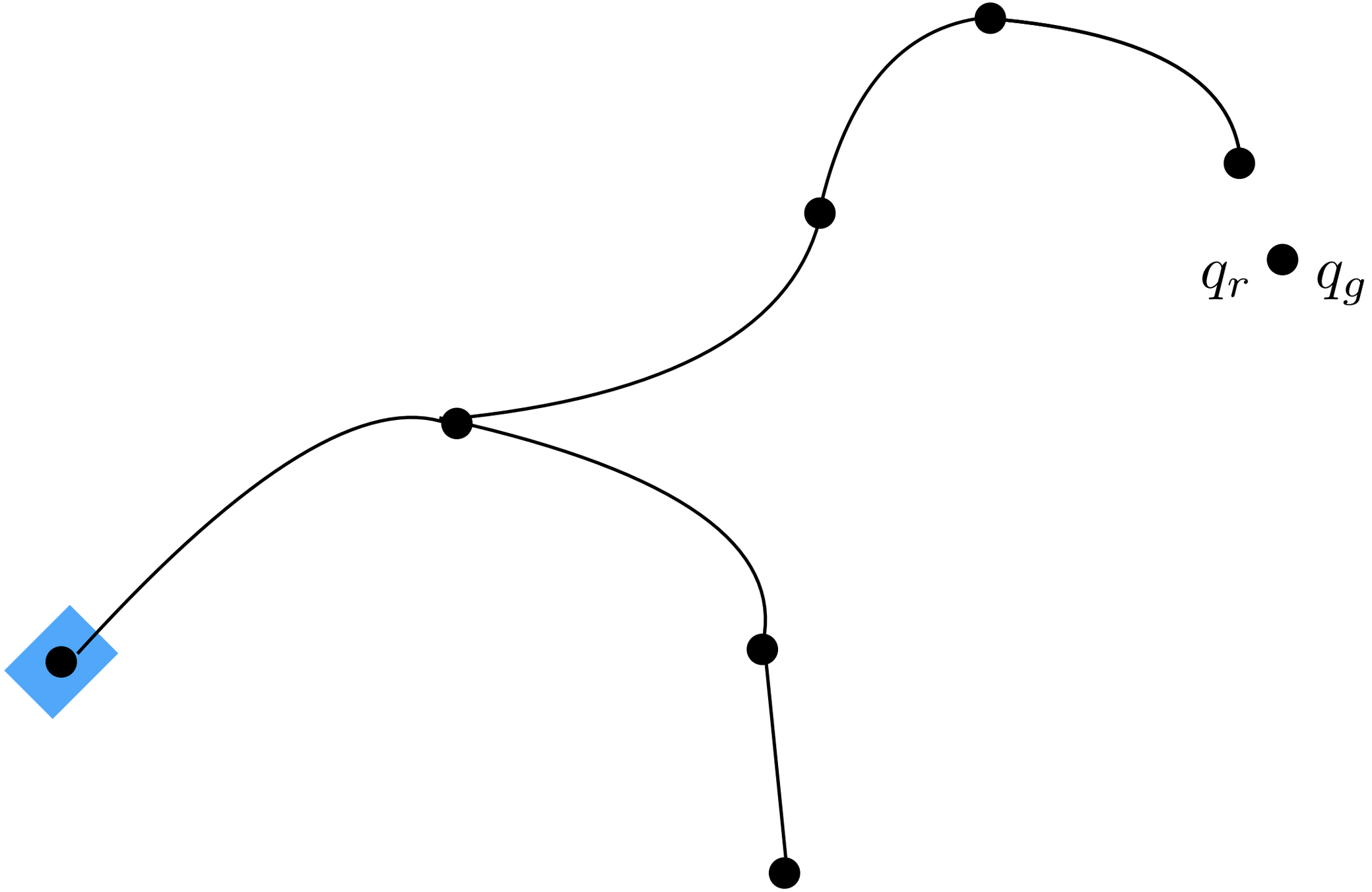
RRT with Shooting Approach



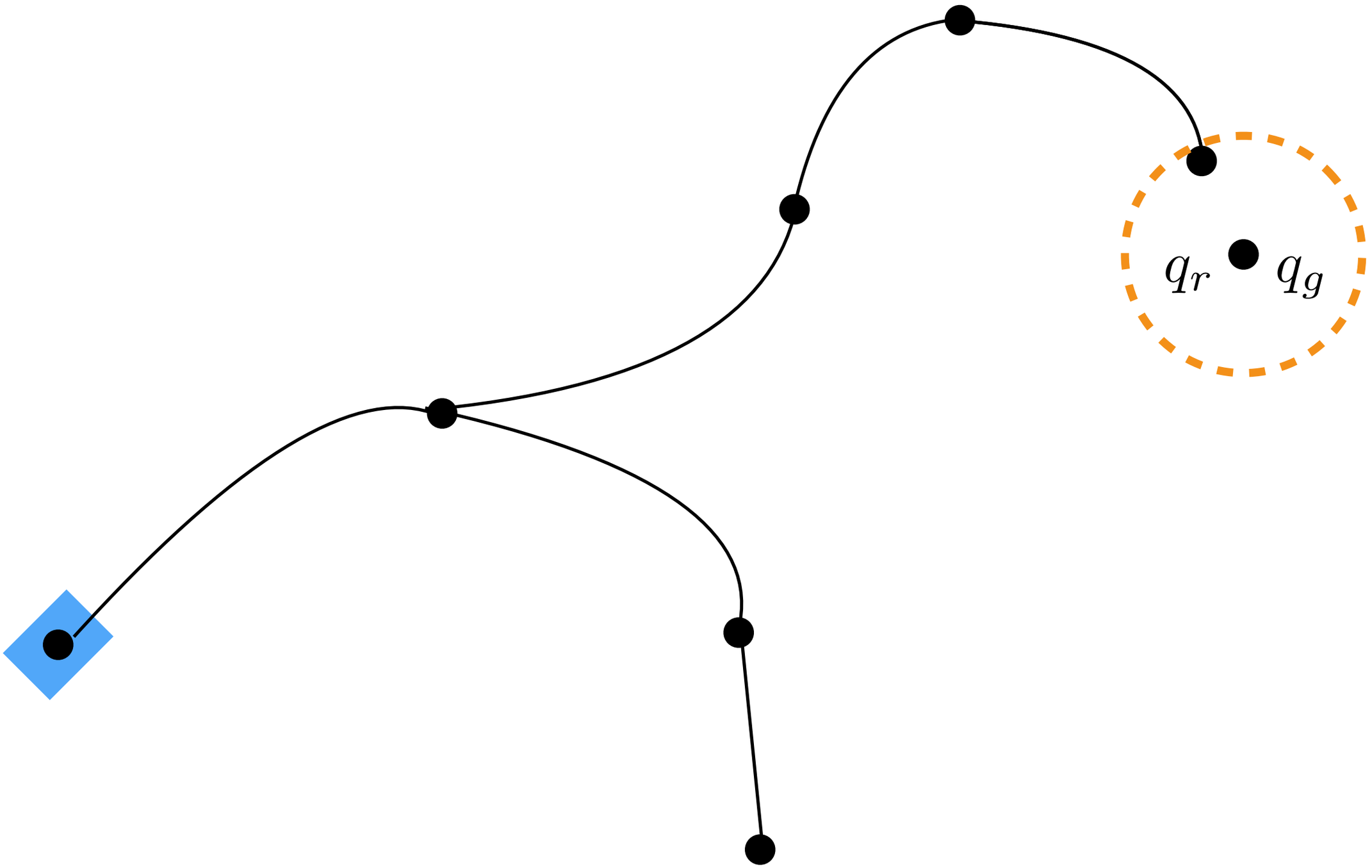
RRT with Shooting Approach



RRT with Shooting Approach

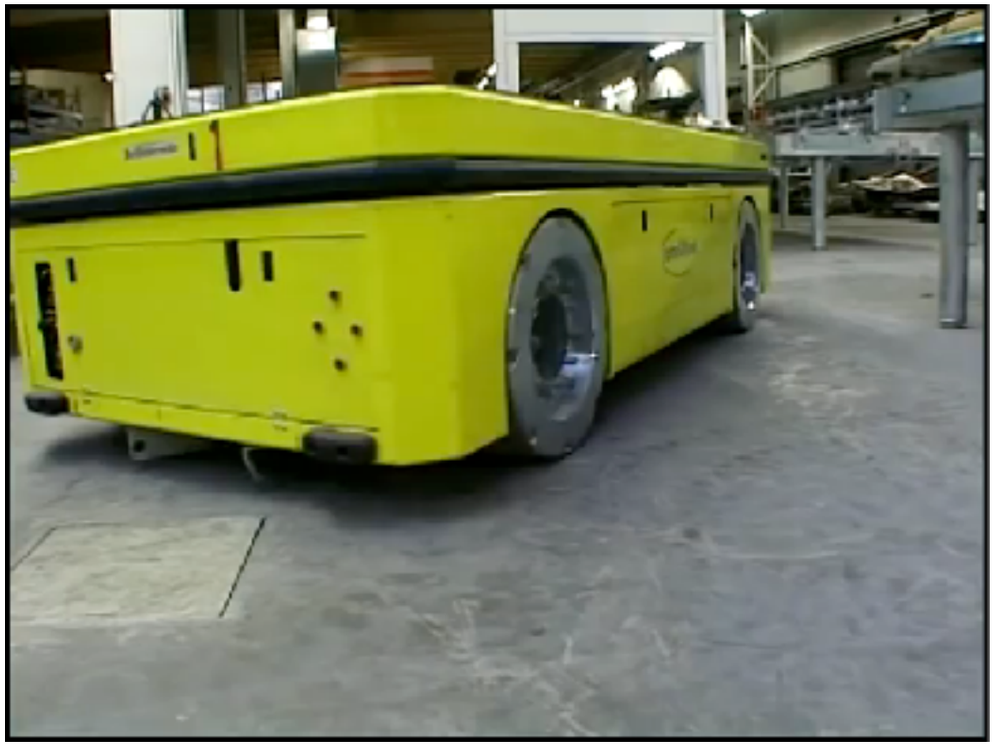


RRT with Shooting Approach



… or engineer the problem away

- Omniwheels can be used make the system holonomic



videos: Kuka

Questions?