

Algorithms for Sensor-based Robotics: Sampling-based Path Planning

Computer Science 336

<http://www.cs.jhu.edu/~hager/Teaching/cs336>

Erion Plaku

<http://www.cs.jhu.edu/~erion>

Path Planning

Construct Configuration Space

Potential Fields/Navigation Functions

Alternative approaches?

CHALLENGES in HIGH-DIMENSIONAL SPACES

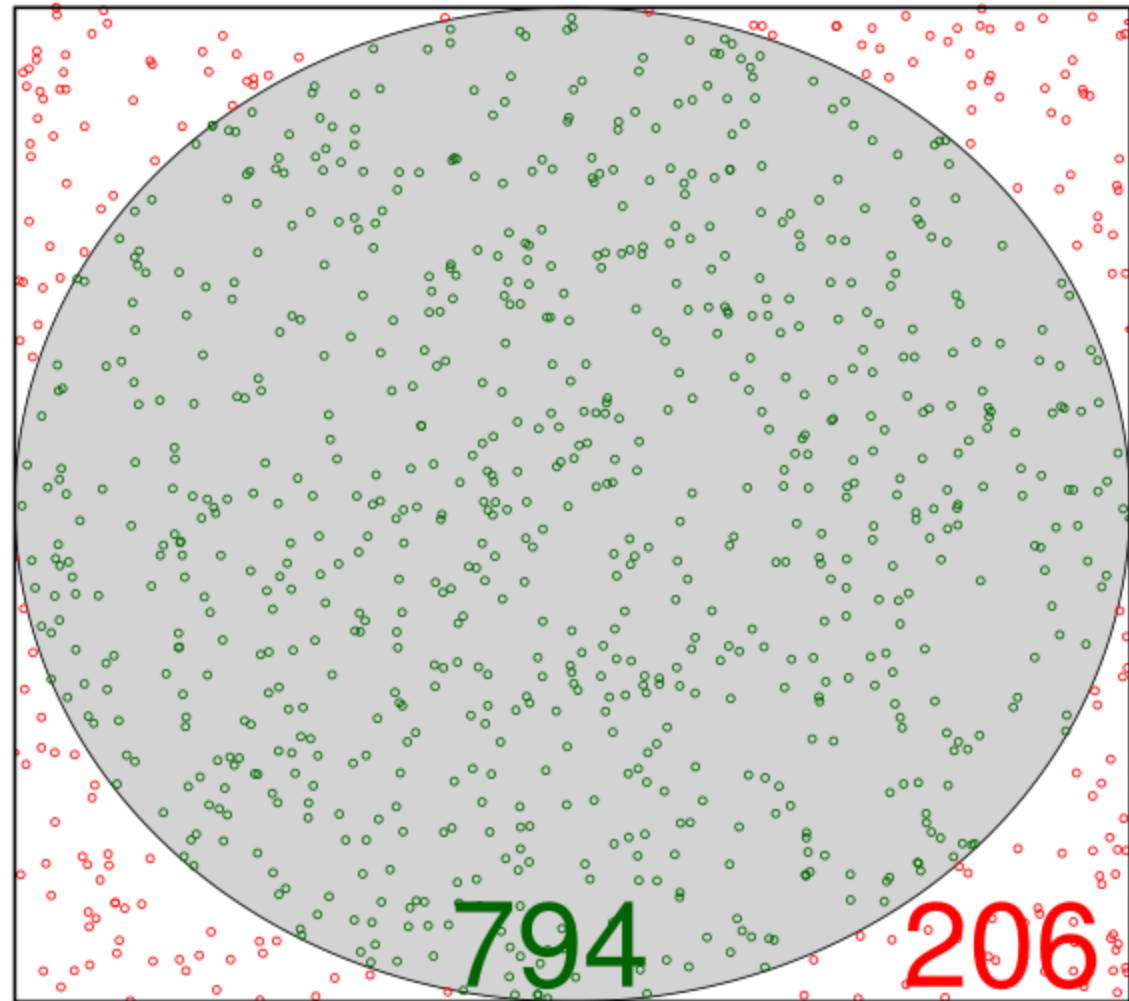
- PSPACE-hard
- Exponential dependency on dimension of configuration space
- No practical implementations
- Rugged landscape
- Multiple local minima
- Difficult to design
- Not generally applicable

Detour: How to Approximate π ?

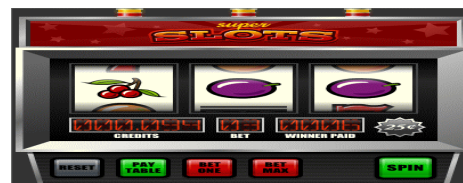


[Figs. from wikipedia]

HINTS



[Fig. from commons.wikimedia.org]



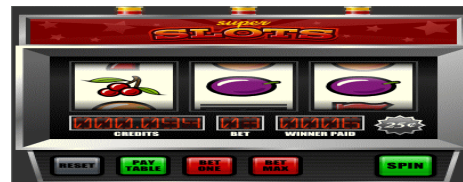
Monte-Carlo Approaches



[Figs. from wikipedia]



- Define input space
- Generate inputs at *random* by *sampling* the input space
- Perform a deterministic computation using the input samples
- Aggregate the partial results into final result



Sampling-based Path Planning

Search for collision-free path from initial to goal configuration
not by explicitly constructing the collision-free configuration space, but
by *sampling* the configuration space

What's the advantage?

- Explores small subset of possibilities
- Computational efficiency
- Widely applicable
- Ease of implementation

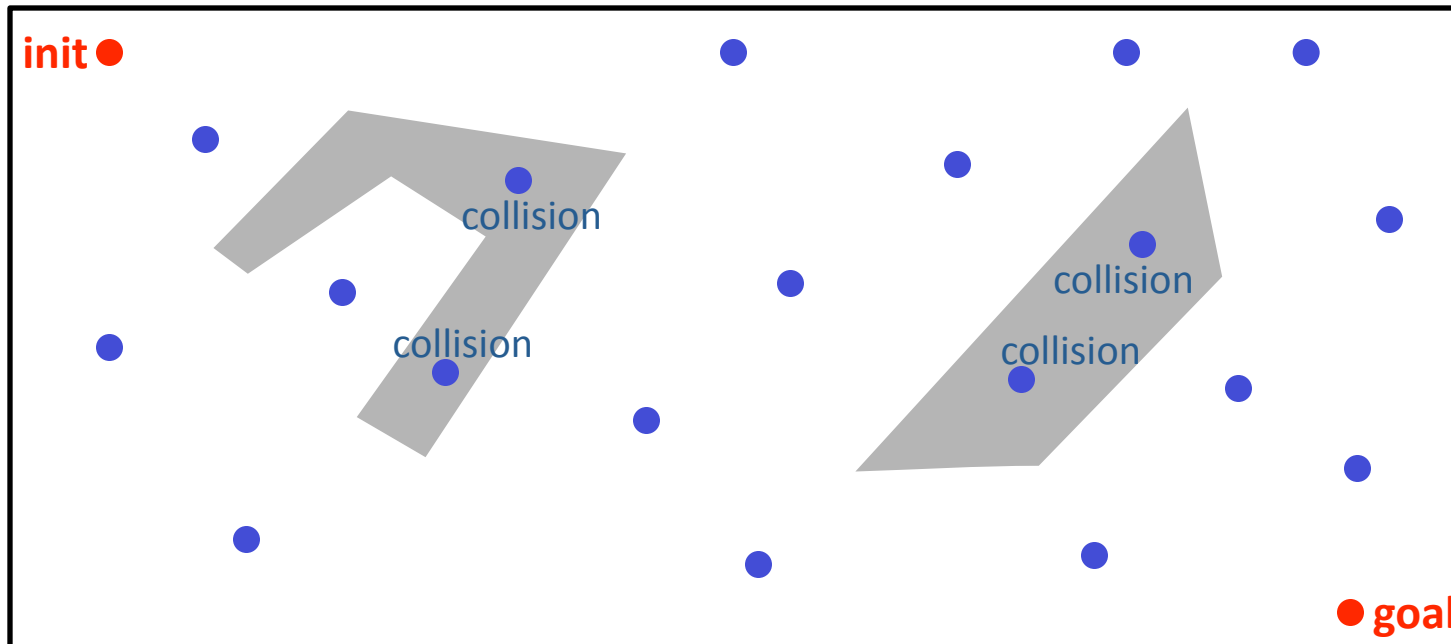
Key Components

- How are samples generated?
- How are samples used?

Sampling-based Path Planning

Search for collision-free path from initial to goal configuration
not by explicitly constructing the collision-free configuration space, but
by *sampling* the configuration space

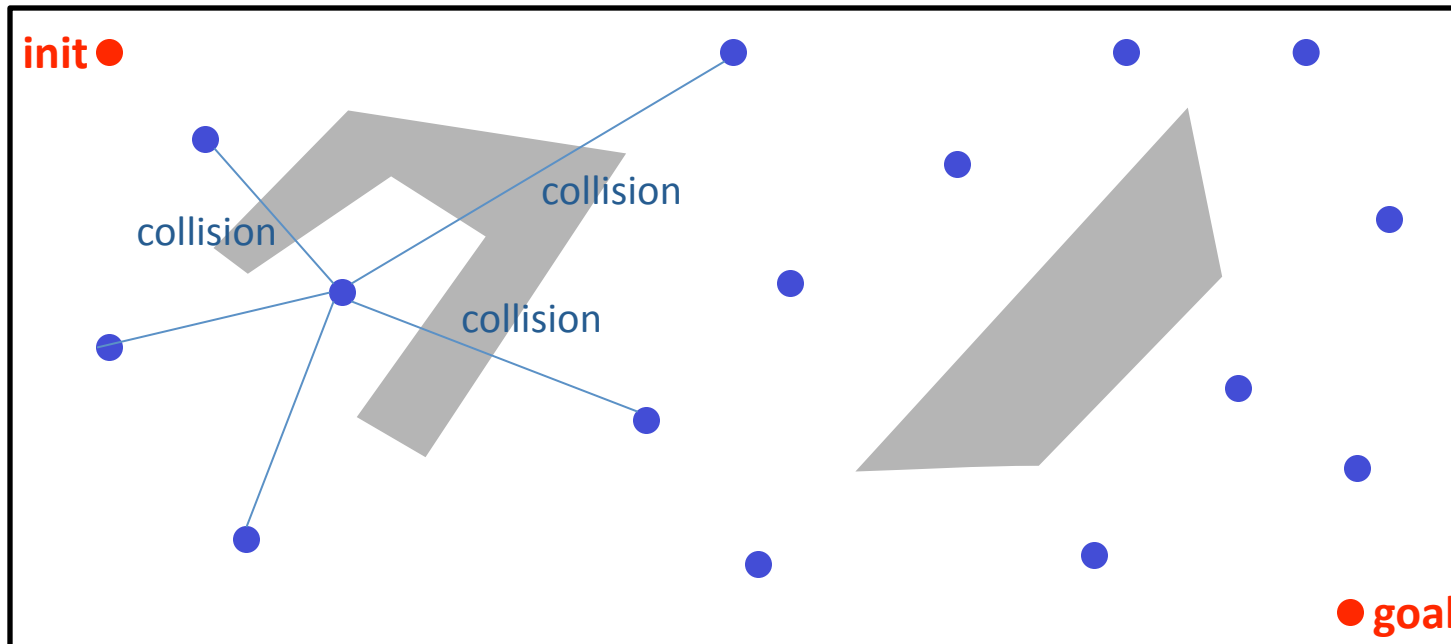
example: 2-dimensional point among polygonal obstacles



Sampling-based Path Planning

Search for collision-free path from initial to goal configuration
not by explicitly constructing the collision-free configuration space, but
by *sampling* the configuration space

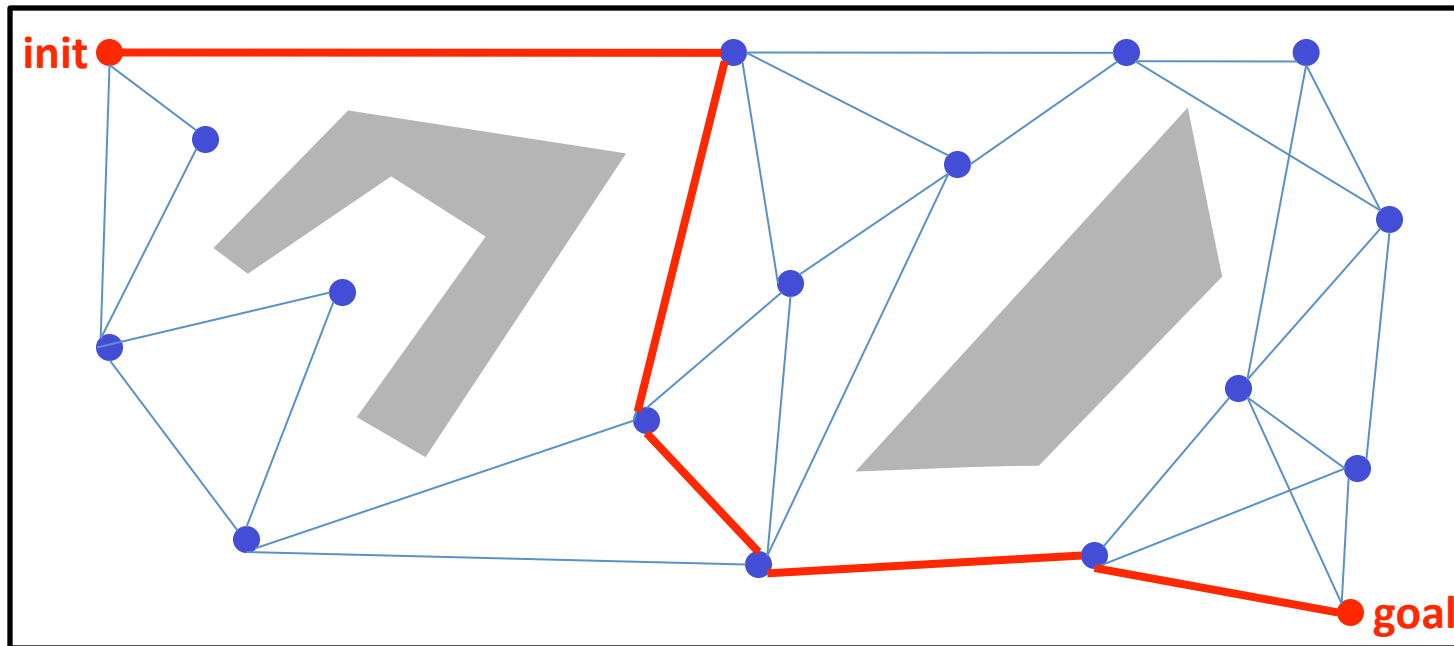
example: 2-dimensional point among polygonal obstacles



Sampling-based Path Planning

Search for collision-free path from initial to goal configuration
not by explicitly constructing the collision-free configuration space, but
by *sampling* the configuration space

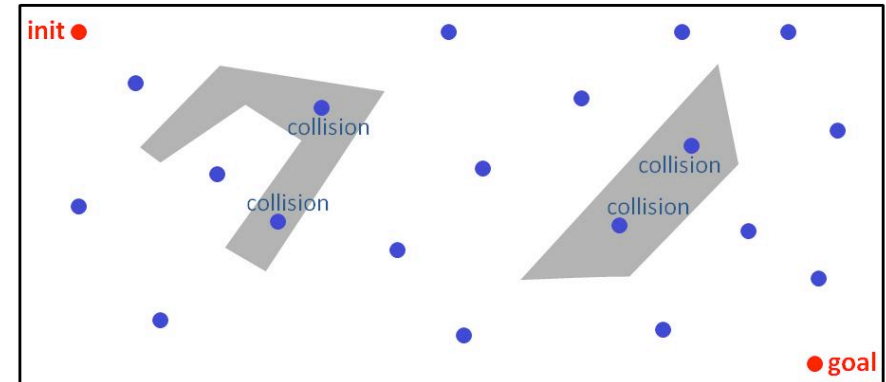
example: 2-dimensional point among polygonal obstacles



Sampling-based Path Planning – Basic Algorithm

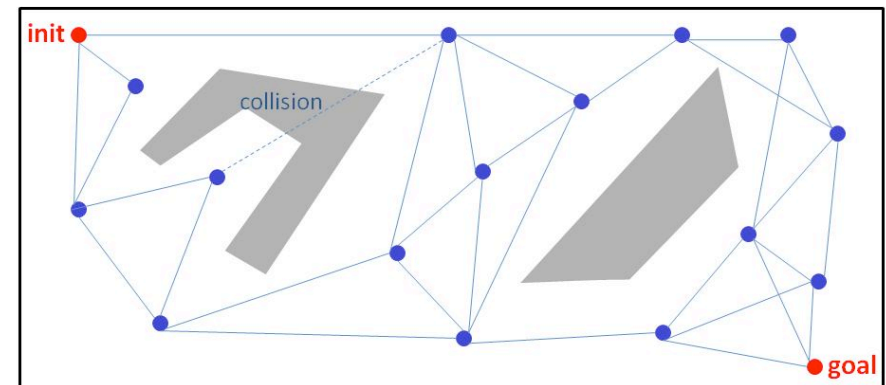
1. Repeat several times

- $q := \text{GenerateSample}()$
- If $\text{IsConfigCollisionFree}(q) = \text{true}$ then
⇒ $V := V \cup \{q\}$



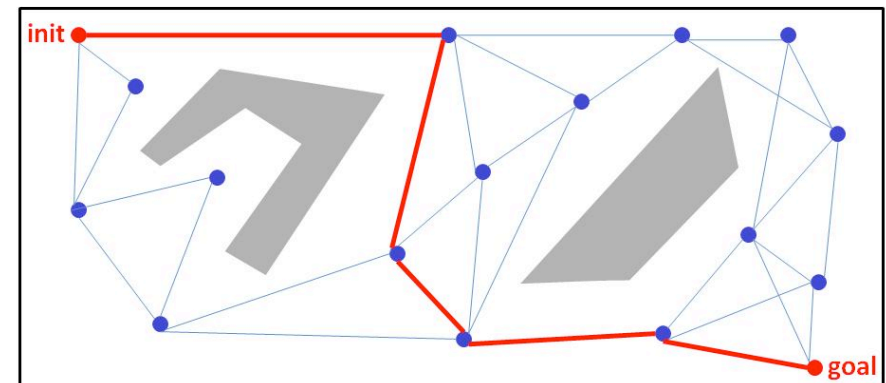
2. Repeat several times

- select a pair (q_a, q_b) from $V \times V$
- $p := \text{GeneratePath}(q_a, q_b)$ [not necessarily collision free]
- If $\text{IsPathCollisionFree}(p) = \text{true}$ then
⇒ $E := E \cup \{(q_a, q_b)\}$
⇒ $(q_a, q_b).\text{path} := p$



3. Search graph for path from initial to goal

- Compute graph sequence q_1, q_2, \dots, q_n connecting q_{init} to q_{goal} , i.e., $q_1 = q_{\text{init}}, q_n = q_{\text{goal}}$ and $(q_i, q_{i+1}) \in E$
- return $(q_1, q_2).\text{path} \cdot \dots \cdot (q_{n-1}, q_n).\text{path}$



1. Abstraction: Sample Generation and Collision Checking

1. Repeat several times

- $q := \text{GenerateSample}()$
- if $\text{IsConfigCollisionFree}(q) = \text{true}$ then add q as graph node

GenerateSample: $\rightarrow Q$

- generates a configuration sample from Q

IsConfigCollisionFree: $Q \rightarrow \{\text{true}, \text{false}\}$

- given any q in Q returns true iff q is collision free

1. Abstraction: Sample Generation and Collision Checking

1. Repeat several times

- $q := \text{GenerateSample}()$
- if $\text{IsCollisionFree}(q) = \text{true}$ then add q as graph node

example: rigid body among polygonal obstacles

GenerateSample: $Q \rightarrow Q$

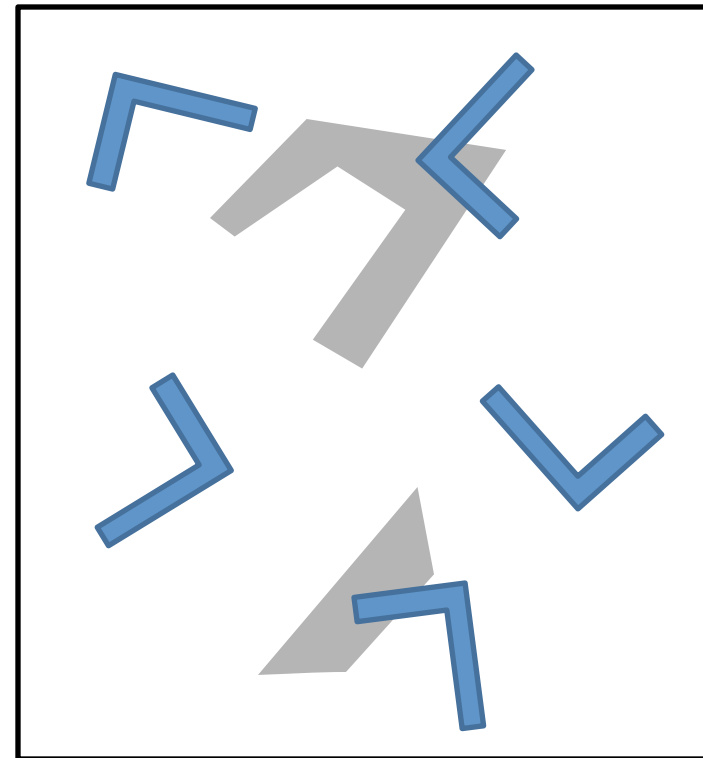
// generates a configuration sample from Q

- $x :=$ sample uniformly at random from $[\min_x, \max_x]$
- $y :=$ sample uniformly at random from $[\min_y, \max_y]$
- $\theta :=$ sample uniformly at random from $[0, 2\pi)$
- return (x, y, θ)

IsCollisionFree: $Q \rightarrow \{\text{true}, \text{false}\}$

// given any q in Q returns true iff q is collision free

- place rigid body in configuration q (affine transformation)
- check collision with polygonal obstacles
- return true iff no collision found



1. Abstraction: Sample Generation and Collision Checking

1. Repeat several times

- $q := \text{GenerateSample}()$
- if $\text{IsConfigCollisionFree}(q) = \text{true}$ then add q as graph node

example: chain with revolute joints among polygonal obstacles

GenerateSample: --> Q

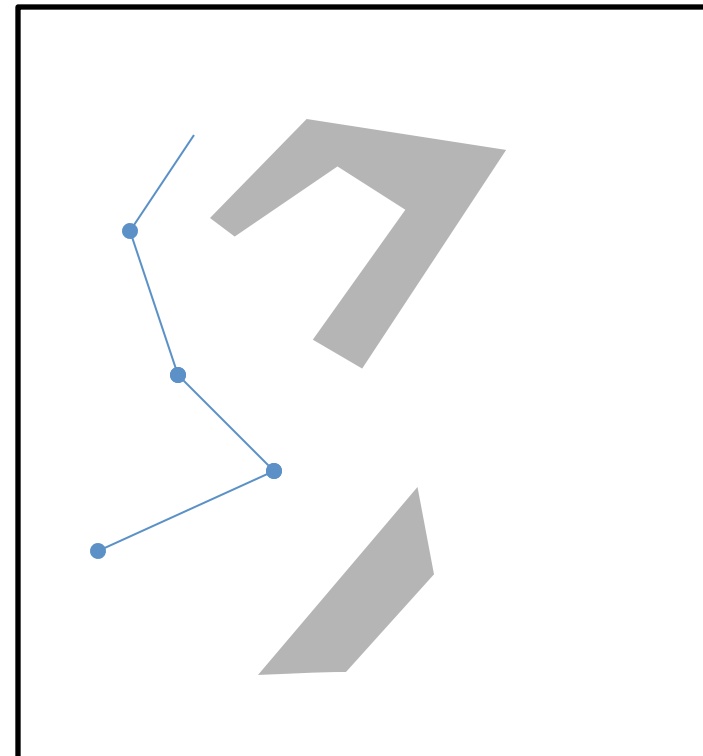
// generates a configuration sample from Q

- for $i = 1$ to n do
 - $\theta_i :=$ sample uniformly at random from $[0, 2\pi)$
- return $(\theta_1, \theta_2, \dots, \theta_n)$

IsConfigCollisionFree: Q --> {true, false}

// given any q in Q returns true iff q is collision free

- place rigid body in configuration q (**forward kinematics**)
- check collision with polygonal obstacles
- return true iff no collision found



2. Abstraction: Connections between Samples

2. Repeat several times

- select a pair (q_a, q_b) from $V \times V$
- $p := \text{GeneratePath}(q_a, q_b)$ [not necessarily collision free]
- If $\text{IsPathCollisionFree}(p) = \text{true}$ then
 - ⇒ $E := E \cup \{(q_a, q_b)\}$
 - ⇒ $(q_a, q_b).\text{path} := p$

path: $[0, T] \rightarrow Q$

- continuous function of path-duration time

GenerateLocalPath: $Q \times Q \rightarrow \text{path}$

- given any a, b in Q it generates local path from a to b
i.e., $\text{path}(0) = a$ and $\text{path}(T) = b$

IsPathCollisionFree: $\text{path} \rightarrow \{\text{true}, \text{false}\}$

- given any path returns true iff path is collision free
i.e., $\text{IsConfigCollisionFree}(\text{path}(t)) = \text{true}$ for all t in $[0, T]$

