Introduction to Robotics

Configuration Space

Erion Plaku

Department of Electrical Engineering and Computer Science
Catholic University of America

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

... a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Collision-Free Configuration

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Collision-Free Configuration

- q is collision free iff the robot does not collide with any obstacles when in configuration q, i.e., $\texttt{Robot}(q) \cap \left(\bigcup_{i=1} \texttt{Obstacle}_i\right) = \emptyset$

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Collision-Free Configuration

- q is collision free iff the robot does not collide with any obstacles when in configuration q, i.e., $\texttt{Robot}(q) \cap \left(\bigcup_{i=1} \texttt{Obstacle}_i\right) = \emptyset$

Collision-Free Configuration Space

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Collision-Free Configuration

- q is collision free iff the robot does not collide with any obstacles when in configuration q, i.e., $\texttt{Robot}(q) \cap \left(\bigcup_{i=1} \texttt{Obstacle}_i\right) = \emptyset$

Collision-Free Configuration Space

- $Q_{free} = \{q \in Q : q \textit{ is collision free}\}$

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Collision-Free Configuration

- q is collision free iff the robot does not collide with any obstacles when in configuration q, i.e., $\texttt{Robot}(q) \cap \left(\bigcup_{i=1} \texttt{Obstacle}_i\right) = \emptyset$

Collision-Free Configuration Space

- $Q_{free} = \{q \in Q : q \textit{ is collision free}\}$

Path-Planning Problem: Compute collision-free path from q_{init} to q_{goal}

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

. . . a key concept in path planning is the notion of a *configuration space*

Configuration (denoted by q)

- a complete specification of the position of every point of the robot

Configuration Space or C-Space (denoted by Q)

- space of all possible configurations of the robot, i.e., $Q = \{q : q \textit{ is a configuration}\}$

Collision-Free Configuration

- q is collision free iff the robot does not collide with any obstacles when in configuration q, i.e., $\texttt{Robot}(q) \cap \left(\bigcup_{i=1} \texttt{Obstacle}_i\right) = \emptyset$

Collision-Free Configuration Space

- $Q_{free} = \{q \in Q : q \textit{ is collision free}\}$

Path-Planning Problem: Compute collision-free path from q_{init} to q_{goal}

- $\texttt{path} : [0, 1] \rightarrow Q_{free}$ is a continuous function with $\texttt{path}(0) = q_{init}$, $\texttt{path}(1) = q_{goal}$

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

Examples of Configuration Spaces

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center
- How can the points on the robot be expressed as a function of its configuration?

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

- How can the points on the robot be expressed as a function of its configuration?

  $$\texttt{Robot}(c_x, c_y) = \{(x, y) : (x - c_x)^2 + (y - c_y) \leq r^2\}$$

Examples of Configuration Spaces

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

- How can the points on the robot be expressed as a function of its configuration?

  $$\texttt{Robot}(c_x, c_y) = \{(x, y) : (x - c_x)^2 + (y - c_y) \leq r^2\}$$

- What is the configuration space Q?

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

- How can the points on the robot be expressed as a function of its configuration?

  $$\texttt{Robot}(c_x, c_y) = \{(x, y) : (x - c_x)^2 + (y - c_y) \leq r^2\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2$ (same as that of a point robot)

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

- How can the points on the robot be expressed as a function of its configuration?

  $$\texttt{Robot}(c_x, c_y) = \{(x, y) : (x - c_x)^2 + (y - c_y) \leq r^2\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2$ (same as that of a point robot)

- What is the free configuration space Q_{free}? Is it the same as that of a point robot?

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

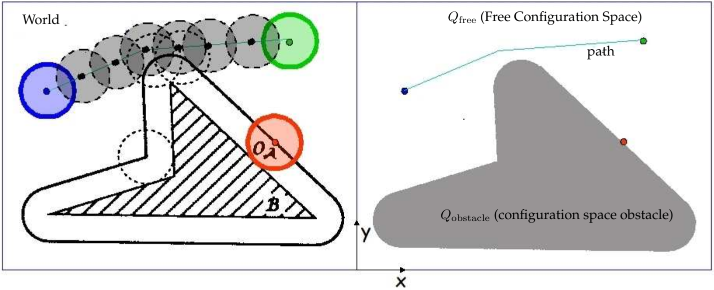
- How can the points on the robot be expressed as a function of its configuration?

  $$\texttt{Robot}(c_x, c_y) = \{(x, y) : (x - c_x)^2 + (y - c_y) \leq r^2\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2$ (same as that of a point robot)

- What is the free configuration space Q_{free}? Is it the same as that of a point robot?



[Fig. courtesy of Latombe]

disk robot with radius r that can translate without rotating in the plane:

- How can the configuration be represented?

  as the two-dimensional position (c_x, c_y) of the robot's center

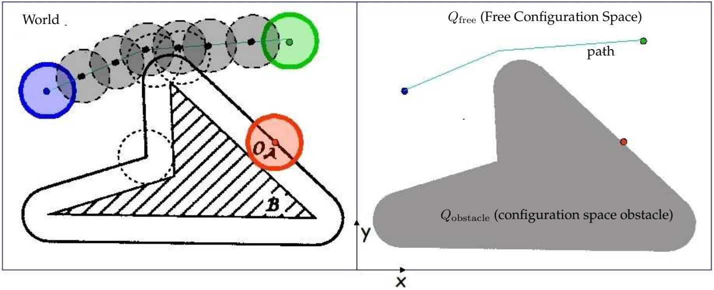
- How can the points on the robot be expressed as a function of its configuration?

  $$\texttt{Robot}(c_x, c_y) = \{(x, y) : (x - c_x)^2 + (y - c_y) \leq r^2\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2$ (same as that of a point robot)

- What is the free configuration space Q_{free}? Is it the same as that of a point robot?



[Fig. courtesy of Latombe]

- How would you compute Q_{free}?

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation

- How can the points on the robot be expressed as a function of its configuration?

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

(c_x, c_y, θ): position + orientation

- How can the points on the robot be expressed as a function of its configuration?

$$\texttt{Robot}(c_x, c_y, \theta) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} : (x, y) \in P \right\}$$

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation

- How can the points on the robot be expressed as a function of its configuration?

$$\texttt{Robot}(c_x, c_y, \theta) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} : (x, y) \in P \right\}$$

- What is the configuration space Q?

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation
- How can the points on the robot be expressed as a function of its configuration?

$$\texttt{Robot}(c_x, c_y, \theta) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} : (x, y) \in P \right\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2 \times S^1$ (S^1 refers to the unit circle)

Examples of Configuration Spaces

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation

- How can the points on the robot be expressed as a function of its configuration?

$$\texttt{Robot}(c_x, c_y, \theta) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} : (x, y) \in P \right\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2 \times S^1$ (S^1 refers to the unit circle)

- What is the free configuration space Q_{free}?

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation

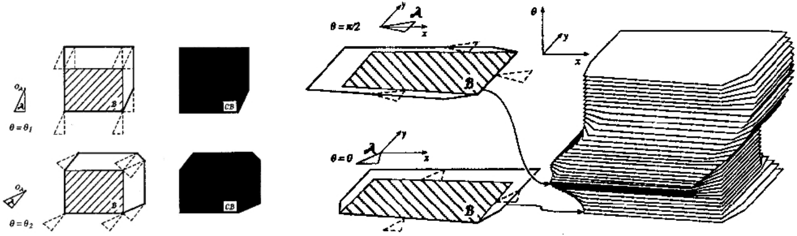
- How can the points on the robot be expressed as a function of its configuration?

$$\mathtt{Robot}(c_x, c_y, \theta) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} : (x, y) \in P \right\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2 \times S^1$ (S^1 refers to the unit circle)

- What is the free configuration space Q_{free}?



[Fig. courtesy of Latombe]

Examples of Configuration Spaces

polygon P that can translate and rotate in the plane:

- How can the configuration be represented?

  (c_x, c_y, θ): position + orientation

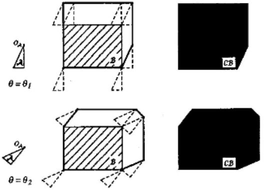
- How can the points on the robot be expressed as a function of its configuration?

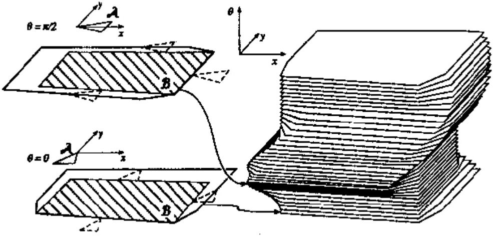
$$\texttt{Robot}(c_x, c_y, \theta) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} : (x, y) \in P \right\}$$

- What is the configuration space Q?

  $Q = \mathbb{R}^2 \times S^1$ (S^1 refers to the unit circle)

- What is the free configuration space Q_{free}?

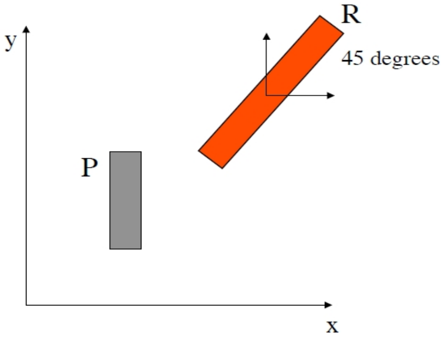


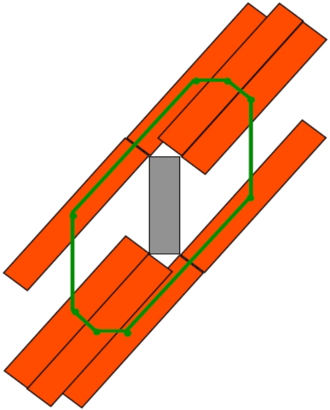


[Fig. courtesy of Latombe]

- How would you compute Q_{free}?

Taking the cros section of configuration space where robot is rotated at 45 degrees:





[Fig. courtesy of Choset, Dodds, Manocha]

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

  $(c_x, c_y, c_z, q_{\mathrm{rot}})$: position + orientation

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

$$(c_x, c_y, c_z, q_{\mathrm{rot}})\text{: position + orientation}$$

Orientation Representations

Examples of Configuration Spaces

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

$(c_x, c_y, c_z, q_{\mathrm{rot}})$: position + orientation

Orientation Representations

- Rotation about x-axis, y-axis, z-axis

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

  $(c_x, c_y, c_z, q_{\mathrm{rot}})$: position + orientation

Orientation Representations

- Rotation about x-axis, y-axis, z-axis

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

  $(c_x, c_y, c_z, q_{\mathrm{rot}})$: position + orientation

Orientation Representations

- Rotation about x-axis, y-axis, z-axis

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Euler angles, e.g., $q_{\mathrm{rot}} = (\alpha, \beta, \gamma)$ (yaw-pitch-roll), so rotation is

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

  $(c_x, c_y, c_z, q_{\mathrm{rot}})$: position + orientation

Orientation Representations

- Rotation about x-axis, y-axis, z-axis

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Euler angles, e.g., $q_{\mathrm{rot}} = (\alpha, \beta, \gamma)$ (yaw-pitch-roll), so rotation is $R_x(\gamma)R_y(\beta)R_z(\alpha)$

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

$(c_x, c_y, c_z, q_{\rm rot})$: position + orientation

Orientation Representations

- Rotation about x-axis, y-axis, z-axis

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Euler angles, e.g., $q_{\rm rot} = (\alpha, \beta, \gamma)$ (yaw-pitch-roll), so rotation is $R_x(\gamma)R_y(\beta)R_z(\alpha)$
- Axis-angle, e.g., $q_{\rm rot} = (u_x, u_y, u_z, \theta)$

$$R(u, \theta) = I\cos\theta + (\sin\theta)[u]_\times + (1 - \cos\theta)u \otimes u$$

$$[u]_\times = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix} \quad u \otimes u = \begin{pmatrix} u_x u_x & u_x u_y & u_x u_z \\ u_y u_x & u_y u_y & u_y u_z \\ u_z u_x & u_z u_y & u_z u_z \end{pmatrix}$$

rigid body P that can translate and rotate in 3D:

- How can the configuration be represented?

$(c_x, c_y, c_z, q_{\mathrm{rot}})$: position + orientation

Orientation Representations

- Rotation about x-axis, y-axis, z-axis

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Euler angles, e.g., $q_{\mathrm{rot}} = (\alpha, \beta, \gamma)$ (yaw-pitch-roll), so rotation is $R_x(\gamma)R_y(\beta)R_z(\alpha)$
- Axis-angle, e.g., $q_{\mathrm{rot}} = (u_x, u_y, u_z, \theta)$

$$R(u, \theta) = I\cos\theta + (\sin\theta)[u]_\times + (1 - \cos\theta)u \otimes u$$

$$[u]_\times = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix} \quad u \otimes u = \begin{pmatrix} u_x u_x & u_x u_y & u_x u_z \\ u_y u_x & u_y u_y & u_y u_z \\ u_z u_x & u_z u_y & u_z u_z \end{pmatrix}$$

- Quaternions

manipulator with revolute joints:

- How can the configuration be represented?

manipulator with revolute joints:

- How can the configuration be represented?

  $(\theta_1, \theta_2, \ldots, \theta_n)$: vector of joint values

manipulator with revolute joints:

- How can the configuration be represented?

  $(\theta_1, \theta_2, \ldots, \theta_n)$: vector of joint values
- How can the points on the robot be expressed as a function of its configuration?

Examples of Configuration Spaces

manipulator with revolute joints:

- How can the configuration be represented?

  $(\theta_1, \theta_2, \ldots, \theta_n)$: vector of joint values

- How can the points on the robot be expressed as a function of its configuration?

  forward kinematics (more later in the course)

manipulator with revolute joints:

- How can the configuration be represented?

  $(\theta_1, \theta_2, \ldots, \theta_n)$: vector of joint values

- How can the points on the robot be expressed as a function of its configuration?

  forward kinematics (more later in the course)

- What is the configuration space Q?

$$Q = \overbrace{S^1 \times S^1 \ldots \times S^1}^{n} \text{ (} S^1 \text{ refers to the unit circle)}$$

Examples of Configuration Spaces

manipulator with revolute joints:

- How can the configuration be represented?

  $(\theta_1, \theta_2, \ldots, \theta_n)$: vector of joint values

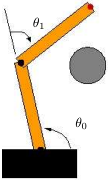
- How can the points on the robot be expressed as a function of its configuration?

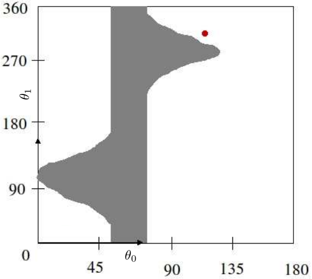
  forward kinematics (more later in the course)

- What is the configuration space Q?

  $$Q = \overbrace{S^1 \times S^1 \ldots \times S^1}^{n}$$ (S^1 refers to the unit circle)

- What is the free configuration space Q_{free}?





manipulator with revolute joints:

- How can the configuration be represented?

  $(\theta_1, \theta_2, \ldots, \theta_n)$: vector of joint values

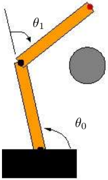
- How can the points on the robot be expressed as a function of its configuration?

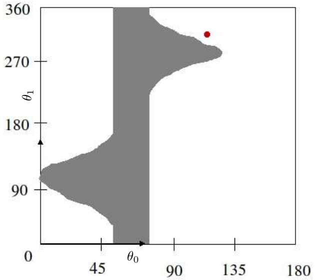
  forward kinematics (more later in the course)

- What is the configuration space Q?

  $$Q = \overbrace{S^1 \times S^1 \ldots \times S^1}^{n}$$ (S^1 refers to the unit circle)

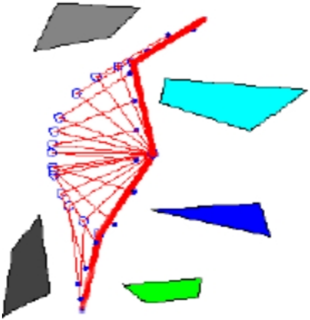
- What is the free configuration space Q_{free}?

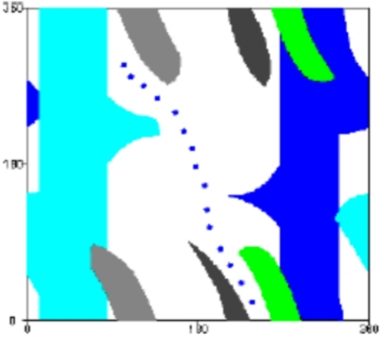




- How would you compute Q_{free}?

Two-link Path





Courtesy of Ken Goldberg

- The Minkowski sum of two sets A and B, denoted by $A \oplus B$, is defined as

$$A \oplus B = \{a + b : a \in A, b \in B\}$$

- The Minkowski difference of two sets A and B, denoted by $A \ominus B$, is defined as

$$A \ominus B = \{a - b : a \in A, b \in B\}$$

How does it relate to path planning?

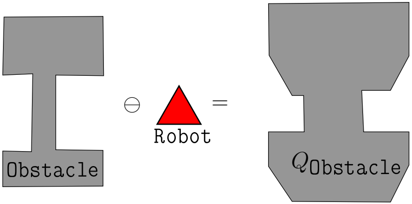
- Recall the definition of the configuration-space obstacle

$$Q_{\texttt{Obstacle}} = \{q : q \in Q \textit{ and } \texttt{Robot}(q) \cap \texttt{Obstacle} \neq \emptyset\}$$

(set of all robot configurations that collide with the obstacle)

- Classical result shown by Lozano-Perez and Wesley 1979

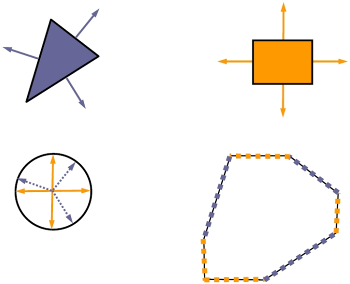
$$\text{for polygons and polyhedra} : Q_{\texttt{Obstacle}} = \texttt{Obstacle} \ominus \texttt{Robot}$$



[Fig. courtesy of Manocha]

Properties of Minkowski Sums

- Minkowski sum of two convex sets is convex
- Minkowski sum of two convex polygons A and B with m and n vertices ...
  - ... is a convex polygon with $m + n$ vertices
  - ... vertices of $A \oplus B$ are "sums" of vertices of A and B
  - ... $A \oplus B$ can be computed in linear time and space $O(n + m)$



[Fig. courtesy of Manocha]

Algorithm

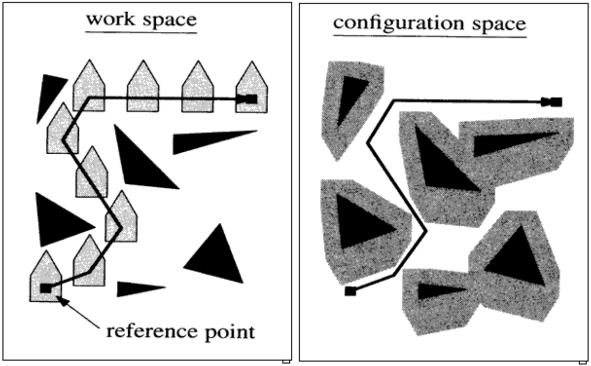
- sort edges according to angle between x-axis and edge normal
- let the sorted edges be $e_1, e_2, \ldots, e_{n+m}$
- attach edges one after the other so that edge e_{i+1} starts where edge e_i ends

- Minkowski sum for nonconvex polygons
  - Decompose into convex polygons (e.g., triangles, trapezoids)
  - Compute the minkowski sums of the convex polygons and take their union
  - Complexity: $O(n^2 m^2)$ (4-th order polynomial)
- 3D Minkowski sums: [convex: $O(nm)$ complexity] [nonconvex: $O(n^3 m^3)$ complexity]

Path Planning: From Point Robots to Robots with Geometric Shapes

- We have seen path-planning algorithms when a robot is a point
- How can we plan a collision-free path when the robot has a geometric shape?

... a key concept in path planning is the notion of a *configuration space*



- reduce robot to a point in the configuration space
- compute configuration-space obstacles (difficult to do in general)
- search for a path for the point robot in the free configuration space

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space

Topology of Configuration Spaces

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space
- Two spaces have different topologies if cutting and pasting is required to make them the same (think of rubber figures — if we can stretch and reshape "continuously" without tearing, one into the other, they have the same topology)

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space
- Two spaces have different topologies if cutting and pasting is required to make them the same (think of rubber figures — if we can stretch and reshape "continuously" without tearing, one into the other, they have the same topology)
- Mathematical mechanisms for talking about topology: homeomorphism/diffeomorphism

$f : X \rightarrow Y$ is called a homeomorphism iff

  - f is a bijection (one-to-one and onto)
  - f is continuous
  - f^{-1} (the inverse of f) is continuous

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space
- Two spaces have different topologies if cutting and pasting is required to make them the same (think of rubber figures — if we can stretch and reshape "continuously" without tearing, one into the other, they have the same topology)
- Mathematical mechanisms for talking about topology: homeomorphism/diffeomorphism

$f : X \rightarrow Y$ is called a homeomorphism iff

- f is a bijection (one-to-one and onto)
- f is continuous
- f^{-1} (the inverse of f) is continuous

examples of homeomorphisms: [disc to square]; [$(-1, 1)$ to $\mathbb{R}$]

X is diffeomorphic to Y iff exists $f : X \rightarrow Y$ such that

- f is a homeomorphism where f and f^{-1} are smooth (derivatives of all orders exist)

Topology of Configuration Spaces

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space
- Two spaces have different topologies if cutting and pasting is required to make them the same (think of rubber figures — if we can stretch and reshape "continuously" without tearing, one into the other, they have the same topology)
- Mathematical mechanisms for talking about topology: homeomorphism/diffeomorphism

  $f : X \rightarrow Y$ is called a homeomorphism iff

  - f is a bijection (one-to-one and onto)
  - f is continuous
  - f^{-1} (the inverse of f) is continuous

  examples of homeomorphisms: [disc to square]; [$(-1, 1)$ to $\mathbb{R}$]

  X is diffeomorphic to Y iff exists $f : X \rightarrow Y$ such that

  - f is a homeomorphism where f and f^{-1} are smooth (derivatives of all orders exist)

  example of diffeomorphism: circle to ellipse

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space
- Two spaces have different topologies if cutting and pasting is required to make them the same (think of rubber figures — if we can stretch and reshape "continuously" without tearing, one into the other, they have the same topology)
- Mathematical mechanisms for talking about topology: homeomorphism/diffeomorphism

  $f : X \rightarrow Y$ is called a homeomorphism iff

  - f is a bijection (one-to-one and onto)
  - f is continuous
  - f^{-1} (the inverse of f) is continuous

  examples of homeomorphisms: [disc to square]; [$(-1, 1)$ to $\mathbb{R}$]

  X is diffeomorphic to Y iff exists $f : X \rightarrow Y$ such that

  - f is a homeomorphism where f and f^{-1} are smooth (derivatives of all orders exist)

  example of diffeomorphism: circle to ellipse

- An n-dimensional configuration space Q is a *manifold* if it locally looks like $\mathbb{R}^n$, i.e., every $q \in Q$ has a neighborhood homeomorphic to $\mathbb{R}^n$

Why study it?

- Extend results from one configuration space to another
- Design specialized algorithms that take advantage of certain topologies

What about the topology?

- Topology is the "intrinsic character" of a space
- Two spaces have different topologies if cutting and pasting is required to make them the same (think of rubber figures — if we can stretch and reshape "continuously" without tearing, one into the other, they have the same topology)
- Mathematical mechanisms for talking about topology: homeomorphism/diffeomorphism

  $f : X \rightarrow Y$ is called a homeomorphism iff

  - f is a bijection (one-to-one and onto)
  - f is continuous
  - f^{-1} (the inverse of f) is continuous

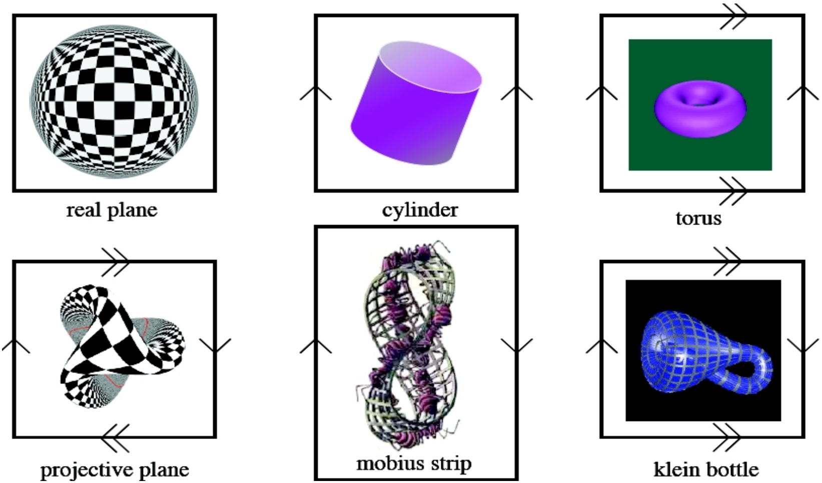
  examples of homeomorphisms: [disc to square]; [$(-1, 1)$ to $\mathbb{R}$]

  X is diffeomorphic to Y iff exists $f : X \rightarrow Y$ such that

  - f is a homeomorphism where f and f^{-1} are smooth (derivatives of all orders exist)

  example of diffeomorphism: circle to ellipse

- An n-dimensional configuration space Q is a *manifold* if it locally looks like $\mathbb{R}^n$, i.e., every $q \in Q$ has a neighborhood homeomorphic to $\mathbb{R}^n$
- A manifold is path-connected if there is a path between any two points



[Fig. courtesy of Choset, Dodds, Manocha]