CS 460/560
Introduction to Computational Robotics
Fall 2019, Rutgers University

Lecture 08-09
Configuration Space & Rigid Body Transformations

Instructor: Jingjin Yu

Outline

Rigid body, links, and joints

The configuration space

Modeling of robots as linked rigid bodies

Degrees of freedom

- ⇨Single rigid body
- ⇨Multiple joined bodies

Task space and workspace

Rigid body transformations

- ⇨Coordinate frames
- ⇨2D rotations and translations
- ⇨3D rotations and translations
- ⇨Special Euclidean group in three dimensions, $SE(3)$

C-space topology, revisited

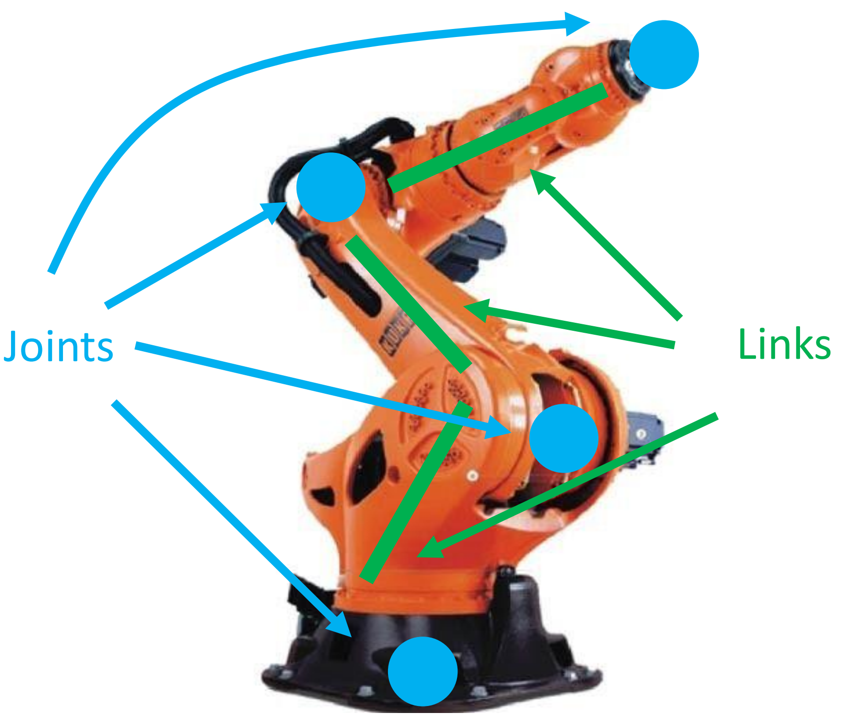
Obstacles and the free C-space

Minkowski sum for computing free C-space

Rigid Body, Links, and Joints

For unified notations

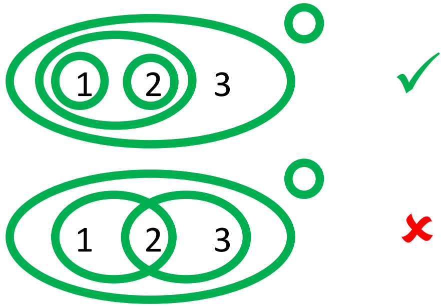
- ⇨A **rigid body** generally means a one-piece robot
- ⇨A **link** is a rigid piece, often a part of a multi-piece robot
- ⇨The **links** of a multi-piece robot are joined with **joints (connectors)**
- ⇨This course mostly work with a single rigid body



The Configuration Space

Recall concepts of topological spaces and (topological) manifolds

- ⇨A topological space is a pair (X, Γ)
  - ⇨ X is a set, Γ is a collection of **open** subsets of X,
  - ⇨ $\emptyset \in \Gamma$ and $X \in \Gamma$
  - ⇨ Arbitrary union of elements of Γ is still in Γ
  - ⇨ Finite intersection of elements of Γ is still in Γ



- ⇨(Topological) manifolds M of dimension n are topological spaces such that every local neighborhood is homeomorphic to $\mathbb{R}^n$

Manifolds nicely capture the **configurations** of robots

- ⇨A configuration is a unique position of a robot (where it is?)
- ⇨The space of configurations is the **configuration space**, or C-space
- ⇨The **dimension** of this space is often the same as the degrees-of-freedom (dof) of the robot
- ⇨E.g., for a car, three dimensions x, y, θ

Why the Configuration Space?

A powerful abstraction for solving **motion planning** problems

- ⇨Motion planning is to find feasible motions for robots to go from x_I to x_G
- ⇨This is non-trivial, e.g., how to plan for parallel parking a car?

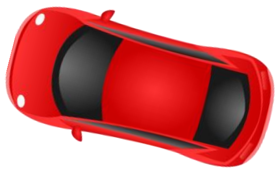
  - ⇨ A hard problem for many drivers!
  - ⇨ And this is just a problem in 2D/3D!
- ⇨Obviously, the position and the orientation must be changed together
- ⇨With C-space, this becomes **searching for a path** in the joint space of 2D position $(x, y) \in \mathbb{R}^2$ and rotation $\theta \in S^1$
- ⇨As a mathematical problem
  - ⇨ You only need an arbitrarily small amount of wiggle room to park your car (STLC)
  - ⇨ So knowing this, when I was in grad school, I sometimes did this…

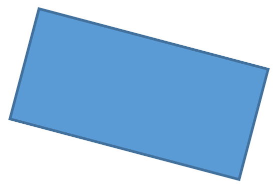
Modeling Robot as Linked Rigid Bodies

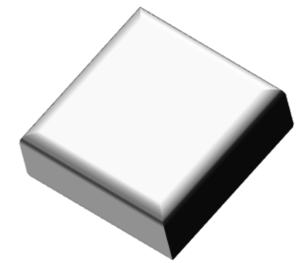
Common robot models

⇨A single point (point robot)

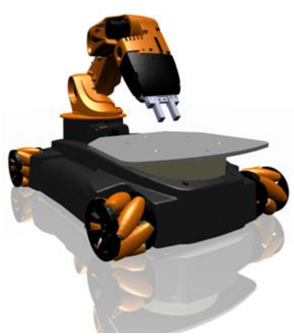
⇨A single rigid body







⇨Multiple rigid bodies (**links**) joined with **joints**



DOF and Types of Joints

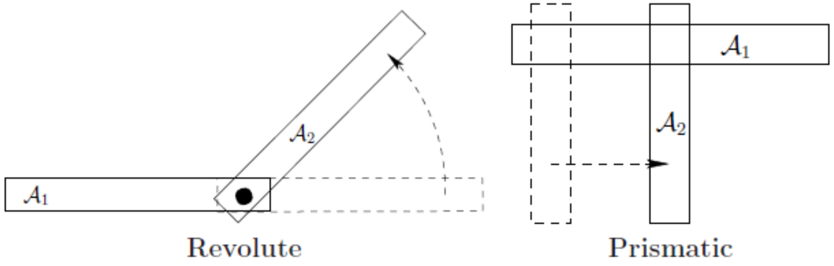
Configuration: specification of where all pieces of a robot are

Degrees of freedom (dof): the smallest number of real-valued (i.e., continuous) coordinates to fully describe configurations of a robot

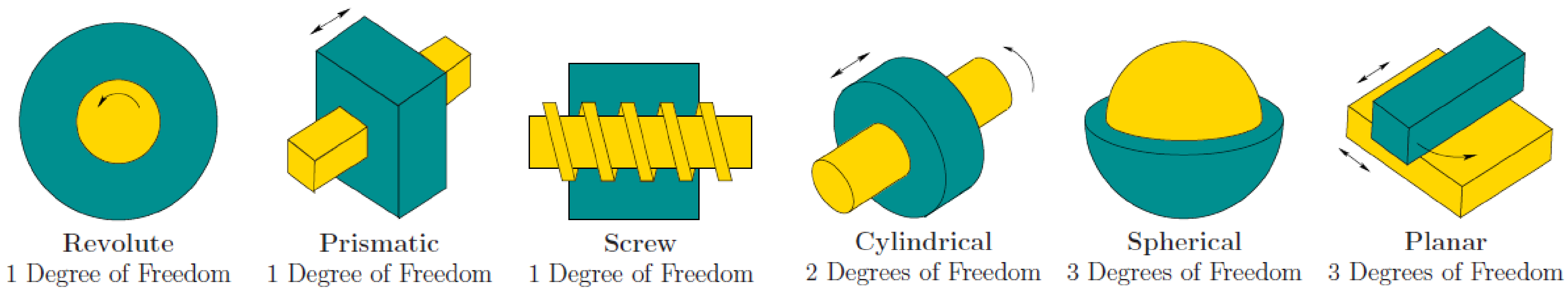
⇨More on this later

Types of joints

⇨2D



⇨3D



Robots generally are viewed as rigid bodies joined by joints

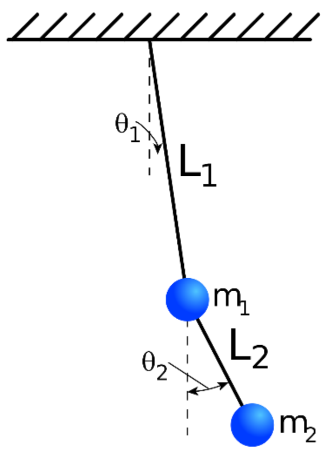
Image source: Planning Algorithms

Examples

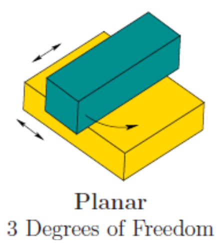
Train

A fan blade

Door



Double pendulum

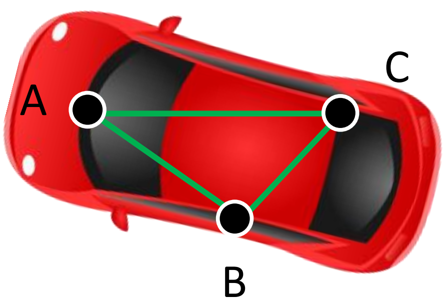


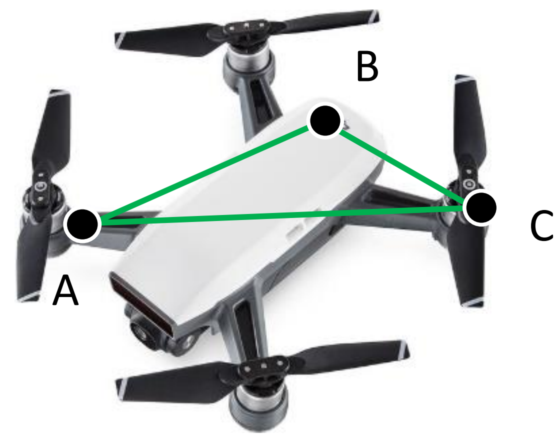
Coin lying flat on a table

Coin on edge

DOF for a Single Rigid Body

The position is fully determined by three fixed points on the body





General formula: DOF = total DOF of points - # of constraints

⇨Car: 2 x 3 – 3 = 3

⇨Quadcopter: 3 x 3 – 3 = 6

Alternatively, can do this incrementally

⇨For the car, A has 2 dofs

⇨Once A is fixed, because d_{AB} is fixed, B has 1 extra dof

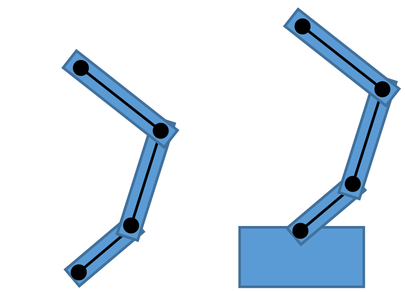
⇨For fixed AB, C is fixed, so 0 extra dof

⇨What about a quadcopter?

Determining the DOF for General Robots

2D chains

- ⇨Base link is 3D ($\mathbb{R}^2 \times S^1$)
- ⇨If fixed, then often 1D
- ⇨Adding joints generally adds one more dimension



3D chains

- ⇨Base link is 6D ($\mathbb{R}^3 \times SO(3)$)
- ⇨If fixed, depending on the joint
- ⇨Then add the DOF of each additional joint

Closed chains

$$\boxed{DOF = N(k-1) - \sum_{i=1}^{n}(N - f_i) = N(k - n - 1) + \sum_{i=1}^{n} f_i}$$

- ⇨We have a formula!
- ⇨N: 6 for 3D, 3 for 2D
- ⇨k: # of links (including the ground link)
- ⇨n: the number of joints
- ⇨f_i: DOF of the joint
- ⇨Examples
  - ⇨ 2D, 3 links
  - ⇨ 2D, 4 links
  - ⇨ 2D, 6 links

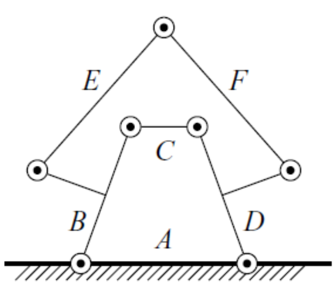


Image source: Principles of Robot Motion

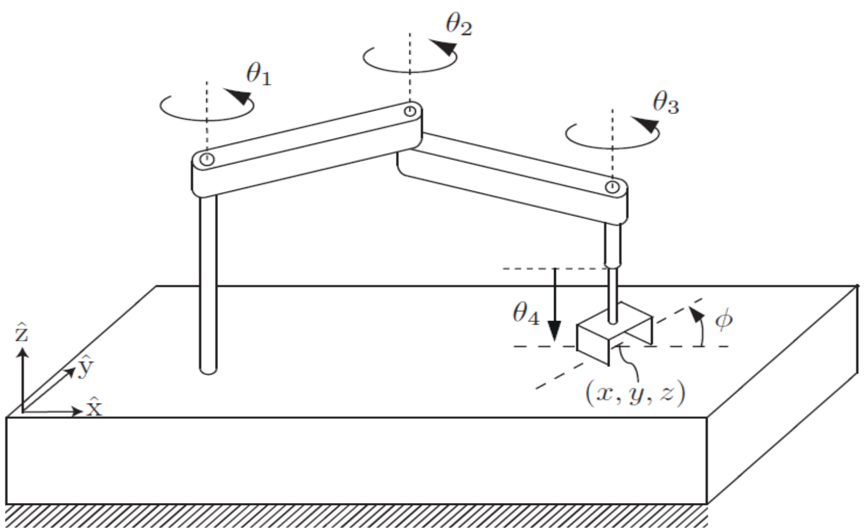
Task Space and Workspace

Task space: a space where the robot's task can be naturally expressed

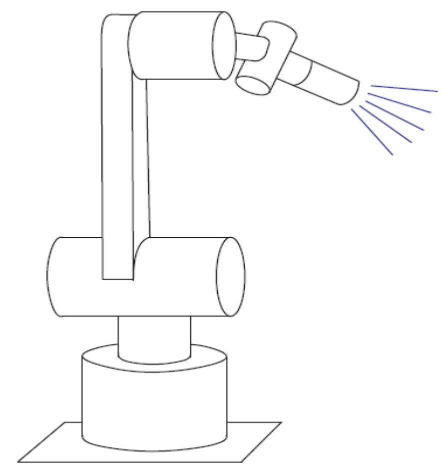
Workspace: captures the "reachable" space of the end-effector

Both involve some user choice and often are different from C-space

Examples



SCARA robot
Task space: $\mathbb{R}^3 \times S^1$
Workspace: reachable points in $\mathbb{R}^3$



Spray paint arm
Task space: $\mathbb{R}^3 \times S^2$
Workspace: reachable points in $\mathbb{R}^3 \times S^2$

Image source: Modern Robotics, http://modernrobotics.org

Coordinate Frames

We use two types of coordinate frames (or simply frames)

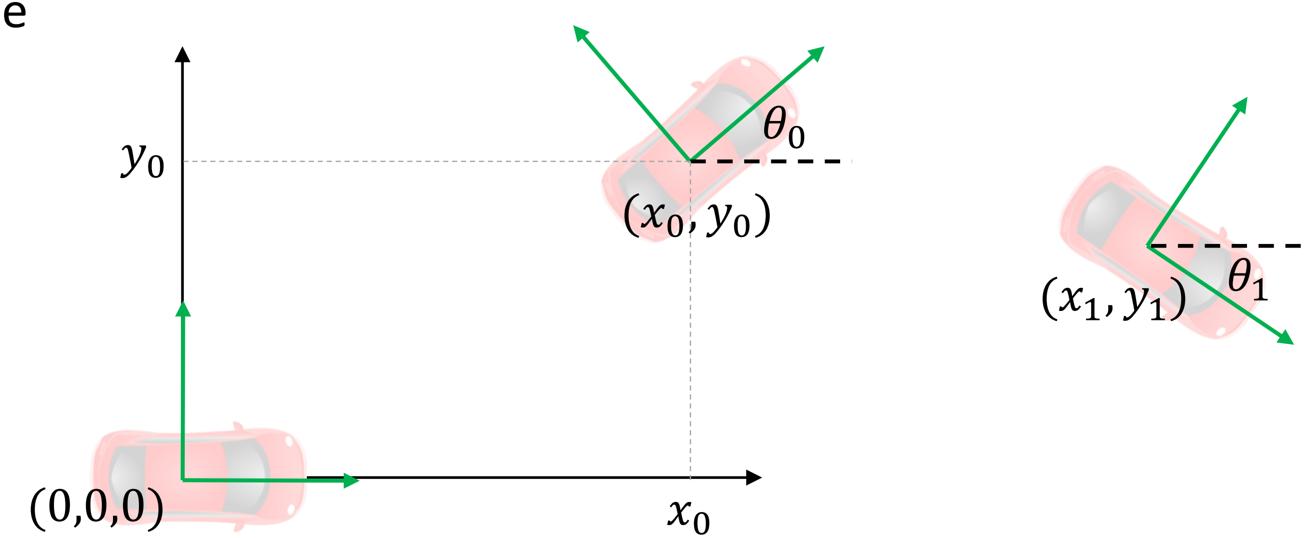
⇨A **global frame**: a "world" coordinate frame

⇨A **local (body) frame**: a coordinate frame "fixed" on the robot

⇨A configuration can be represented as a matrix, e.g., in 2D

$$(x_0, y_0, \theta_0) \rightarrow P_0 = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & x_0 \\ \sin\theta_0 & \cos\theta_0 & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

⇨Rigid body transformation: moving the local frame with respect to the global frame



Rigid Body Transformations in 2D

Given (x_0, y_0, θ_0) and $(\Delta x, \Delta y, \Delta\theta)$, how to compute (x_1, y_1, θ_1)?

⇨ $(\Delta x, \Delta y, \Delta\theta)$ here means "rotate by $\Delta\theta$ and then translate by $(\Delta x, \Delta y)$"

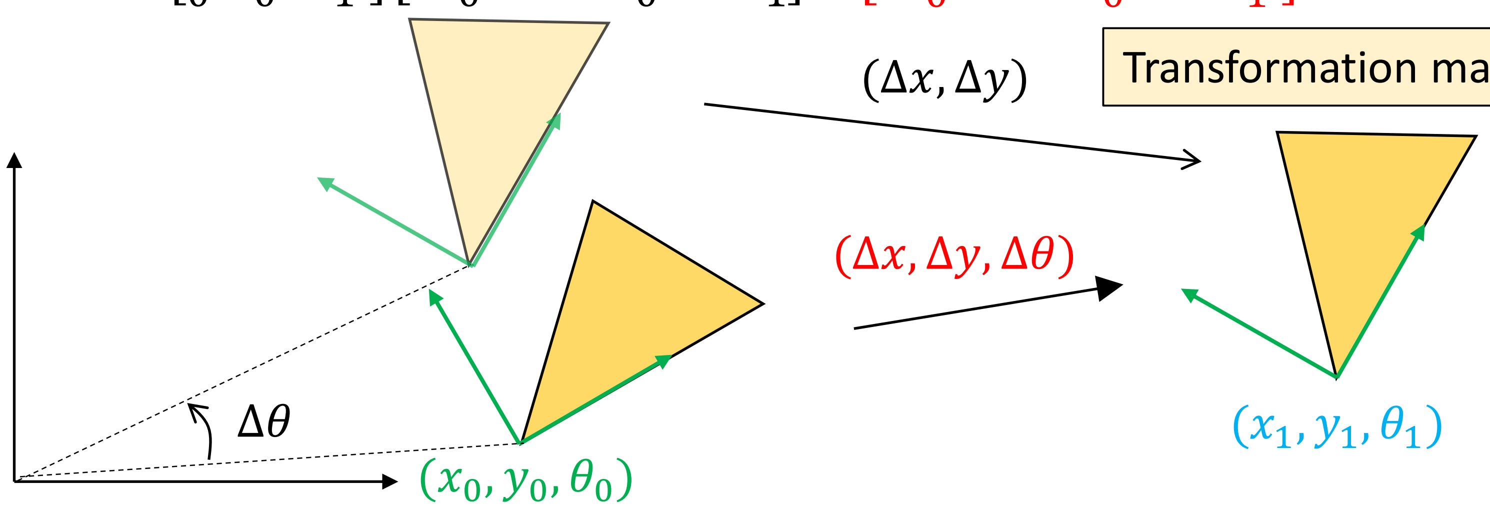
⇨ First, represent $(\Delta x, \Delta y, \Delta\theta)$ also as a matrix

⇨ A **rotational** component $R(\theta) = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & 0 \\ \sin\Delta\theta & \cos\Delta\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

⇨ **Followed** by a **translational** component $r(x, y) = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$

⇨ Together, $\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & 0 \\ \sin\Delta\theta & \cos\Delta\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & \Delta x \\ \sin\Delta\theta & \cos\Delta\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} = T$

Transformation matrix



Rigid Body Transformations in 2D, Continued

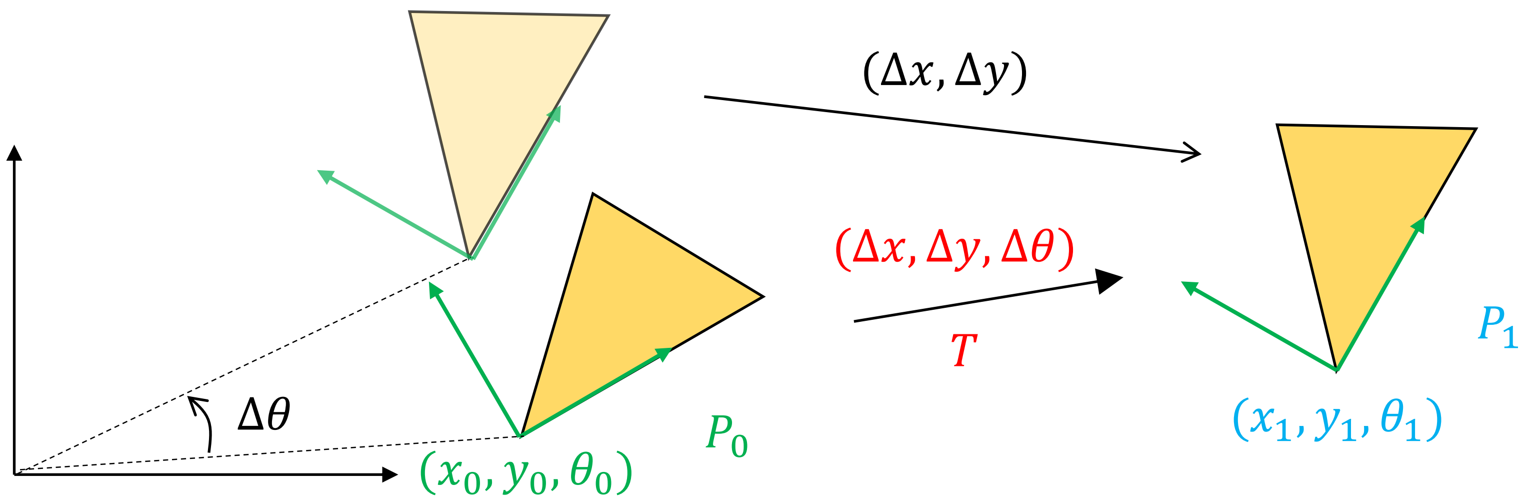
Given (x_0, y_0, θ_0) and $(\Delta x, \Delta y, \Delta\theta)$, how to compute (x_1, y_1, θ_1)?

⇨Use matrix multiplication!

⇨Represent $(\Delta x, \Delta y, \Delta\theta)$ as a matrix $T = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & \Delta x \\ \sin\Delta\theta & \cos\Delta\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$

⇨The operation is "simple" (simple for computers) multiplication

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & x_1 \\ \sin\theta_1 & \cos\theta_1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} = P_1 = TP_0 = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & \Delta x \\ \sin\Delta\theta & \cos\Delta\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & x_0 \\ \sin\theta_0 & \cos\theta_0 & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$



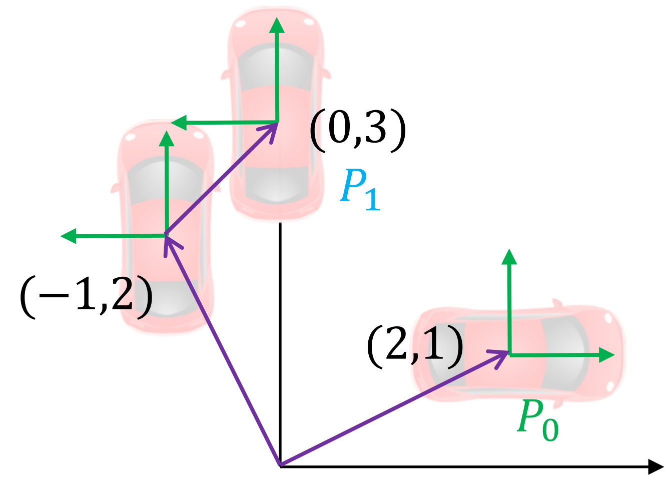
Example

A 2D transformation example

$\theta_0 = 0$

⇨ $P_0 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

$(x_0 = 2, y_0 = 1)$



⇨ Rotate 90 degrees counterclockwise and then translate by (1,1)

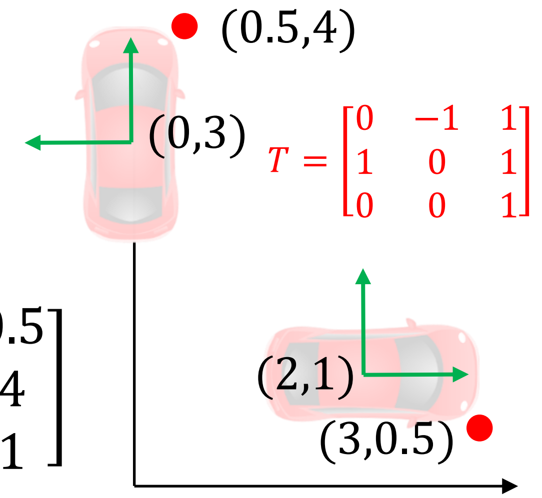
$$\Rightarrow T = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & \Delta x \\ \sin\Delta\theta & \cos\Delta\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 90 & -\sin 90 & 1 \\ \sin 90 & \cos 90 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

⇨ Apply the transformation

$$P_1 = TP_0 = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

⇨ Let’s check…

Why Matrix Multiplication?



It applies to all points on the rigid body

⇨E.g., $P_0' = (3,0.5)$

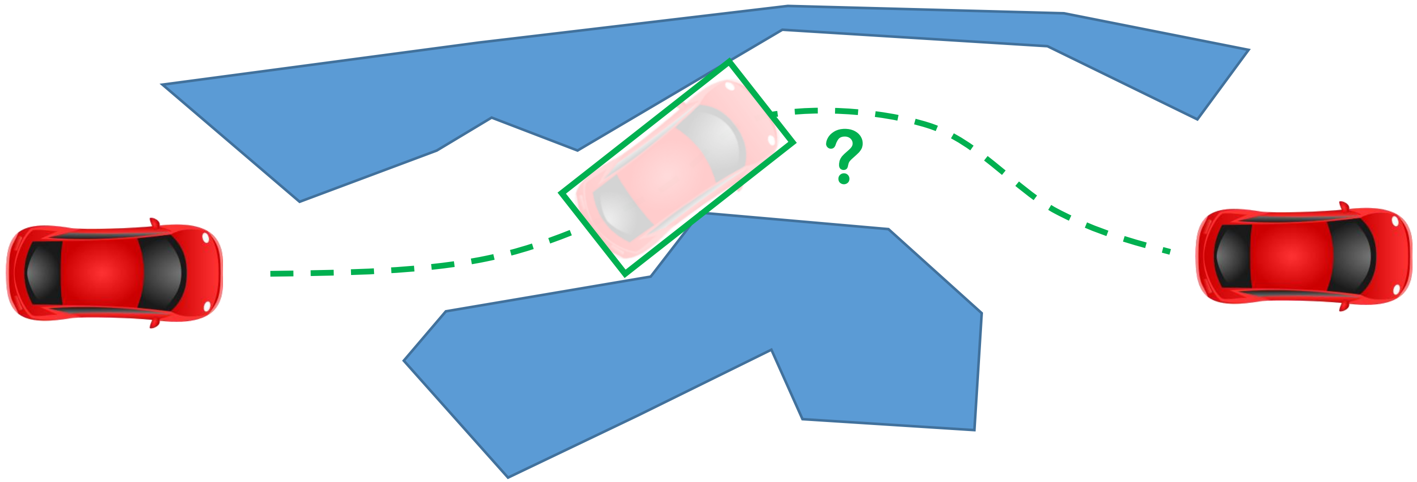
⇨$P_1{}' = TP_0{}' = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5 \\ 1 & 0 & 4 \\ 0 & 0 & 1 \end{bmatrix}$

⇨Can be easily chained, i.e.

$$P_n = T_n \ldots T_1 P_0$$

This is not easily doable with other approaches (e.g., additions)

Essential for things like collision checking



Change Global Frame

Changing the global coordinate frame can also be useful sometimes

⇨E.g., a drone is protecting one base and then a different base

⇨Can also be done using a transformation matrix

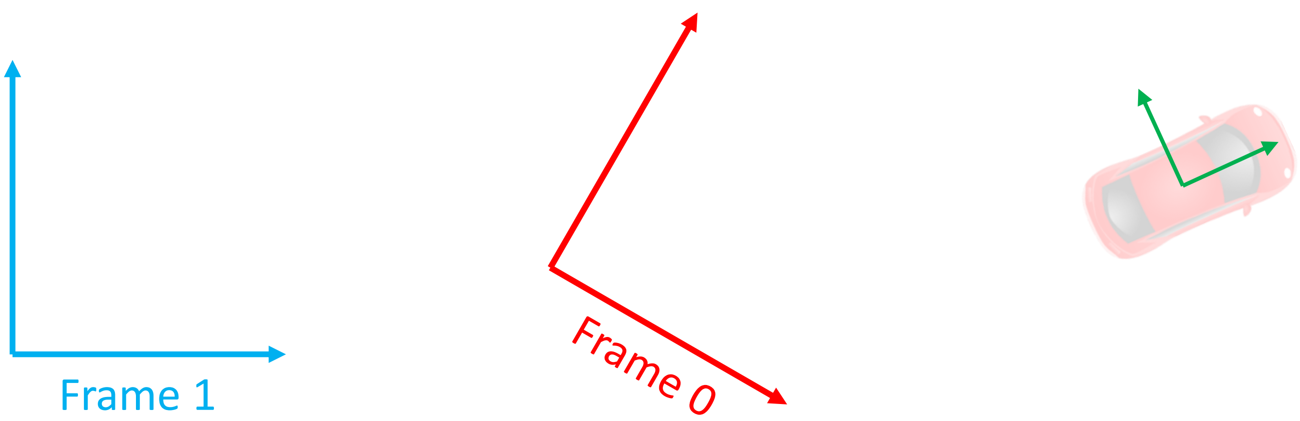
Going from frame 0 to frame 1

⇨Let the P^0 be the configuration of the local frame in frame 0

⇨Let T be the configuration of frame 0 in frame 1

⇨Then the configuration of the local frame in frame 1 is simply

$$P^1 = TP^0$$



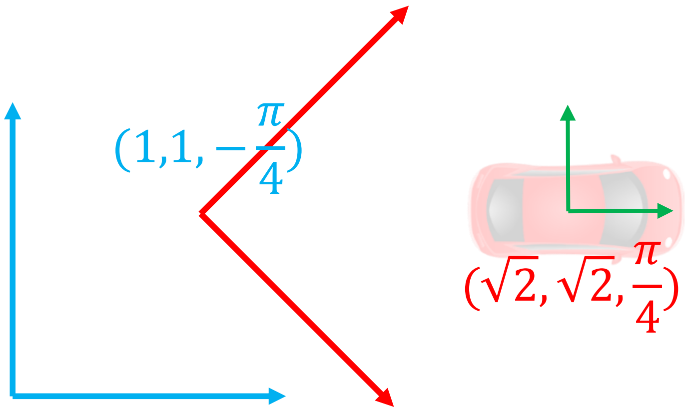
Change Global Frame: Example

The local frame has a configuration $(\sqrt{2},\sqrt{2},\frac{\pi}{4})$ in the red global frame

$$\Rightarrow \text{Write as } P^0 = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \sqrt{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

The red global frame has a configuration $(1,1,-\frac{\pi}{4})$ in the blue global frame

$$\Rightarrow \text{Written as } T = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



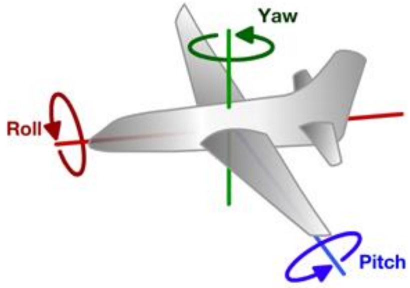
Going from frame 0 to frame 1

$$\Rightarrow P^1 = TP^0 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \sqrt{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Rigid Body Transformations in 3D

Homogeneous transformation generalizes to higher dimensions

In 3D, each transformation has 4 components



- ⇨Yaw: counterclockwise rotation of α along the z axis
- ⇨Pitch: counterclockwise rotation of β along the y axis
- ⇨Roll: counterclockwise rotation of γ along the x axis
- ⇨Translation (x_t, y_t, z_t) in $\mathbb{R}^3$
- ⇨Using homogeneous transformation

$$T = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & x_t \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & y_t \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- ⇨Remember the order!
  - ⇨ Roll by γ
  - ⇨ Pitch by β
  - ⇨ Yaw by α
  - ⇨ Translate by (x_t, y_t, z_t)
- ⇨Of course, other transformations can also be done

Special Euclidean Group $SE(3)$

Special Euclidean group $SE(3) = \mathbb{R}^3 \times SO(3)$

The name is similar to how $SE(2)$ is named

$SO(3)$ however is very interesting…

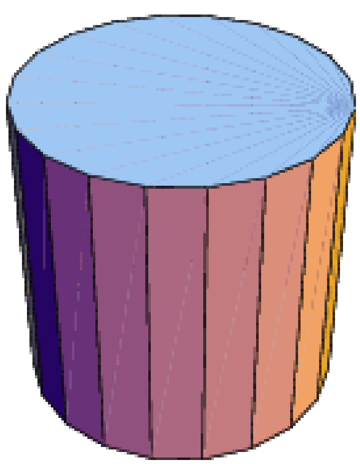
- ⇨These are all possible 3D rotations
- ⇨A 3D rotation can be represented as a rotation of θ along a 3D vector v
- ⇨But this is not unique!

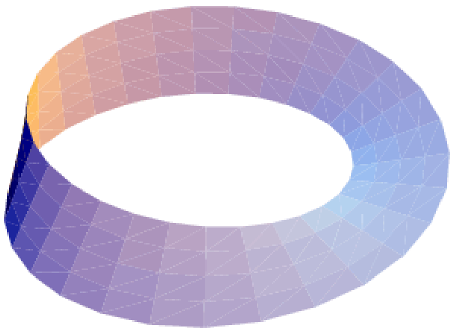
- ⇨It turns out that $SO(3) \cong \mathbb{R}P^3$ (real projective 3-space)
- ⇨Important: $SO(3)$ is not the same as S^3 (surface of a 4D ball)

C-Space Topology, Revisited

The topology of C-space is its most important property

- ⇨ E.g., $SE(2) = \mathbb{R}^2 \times S^1 \neq \mathbb{R}^3$
  - ⇨ A car in 2D rotating clockwise in place will repeat a configuration periodically
  - ⇨ A point in 3D moving along z-axis will never repeat a configuration
- ⇨ Similarly, $SE(2) \neq SO(3)$
- ⇨ Similarly, cylinder $\mathbb{R} \times S^1 \neq$ Mobius band
  - ⇨ A robot traveling continuously on a cylinder can never change side
  - ⇨ A robot traveling continuously on a Mobius band can reach both sides



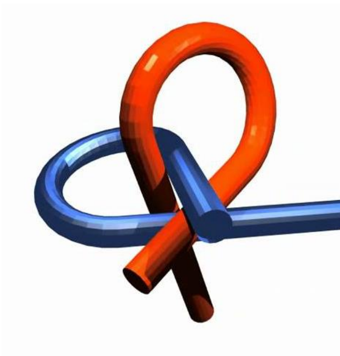


Obstacles and Free Configuration Space

Planning in C-space is trivial without obstacles

- ⇨Why?
- ⇨To go from x_I to x_G, simply draw a straight line between them!

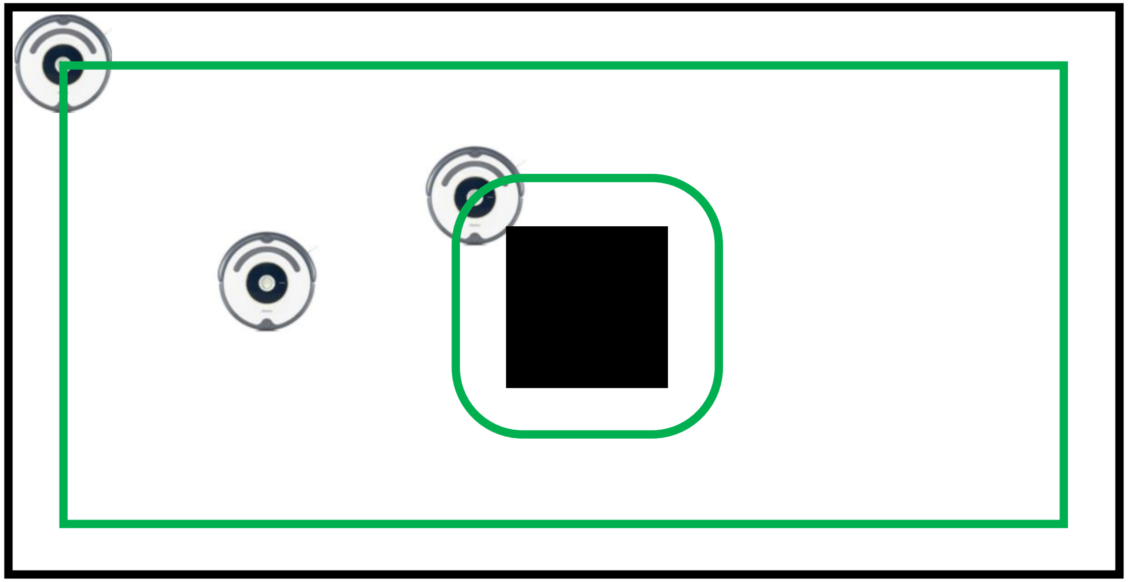
However, obstacles make things more interesting



- ⇨Let q be a robot configuration
- ⇨C-space obstacle C_{obs}: all q that are in collision with an obstacle
  - ⇨ The obstacle could be the robot itself
- ⇨Free C-space: $C_{free} = C \backslash C_{obs}$

A 2D example

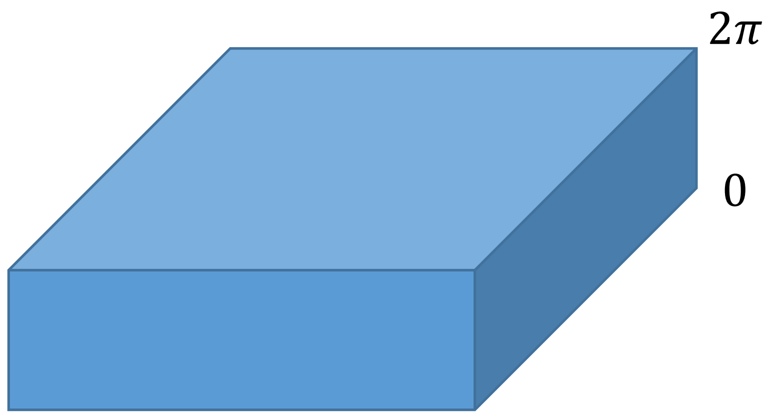
- ⇨Ignore rotation for now
- ⇨More on this later



How Does a Configuration Space Look Like?

Rigid body transformations $SE(2)$

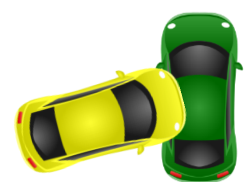
⇨ When there are no obstacles, $\mathbb{R}^2 \times [0, 2\pi)$ with 0 and 2π identified



⇨ It can be more complex with obstacles

- ⇨ E.g. parallel parking a car

Not in collision

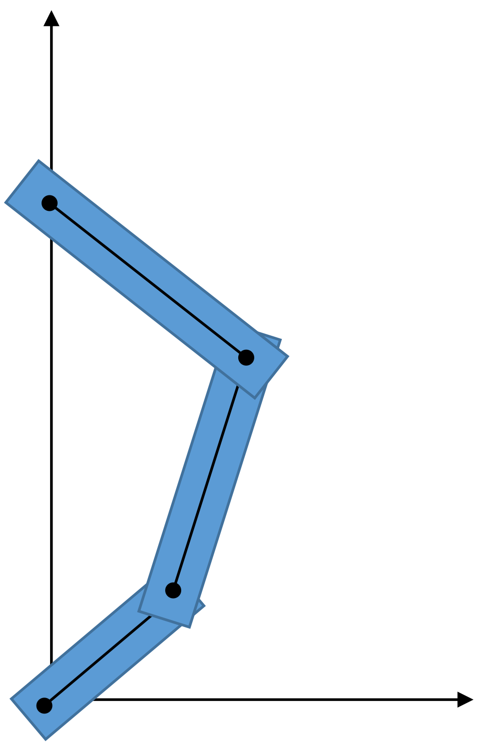


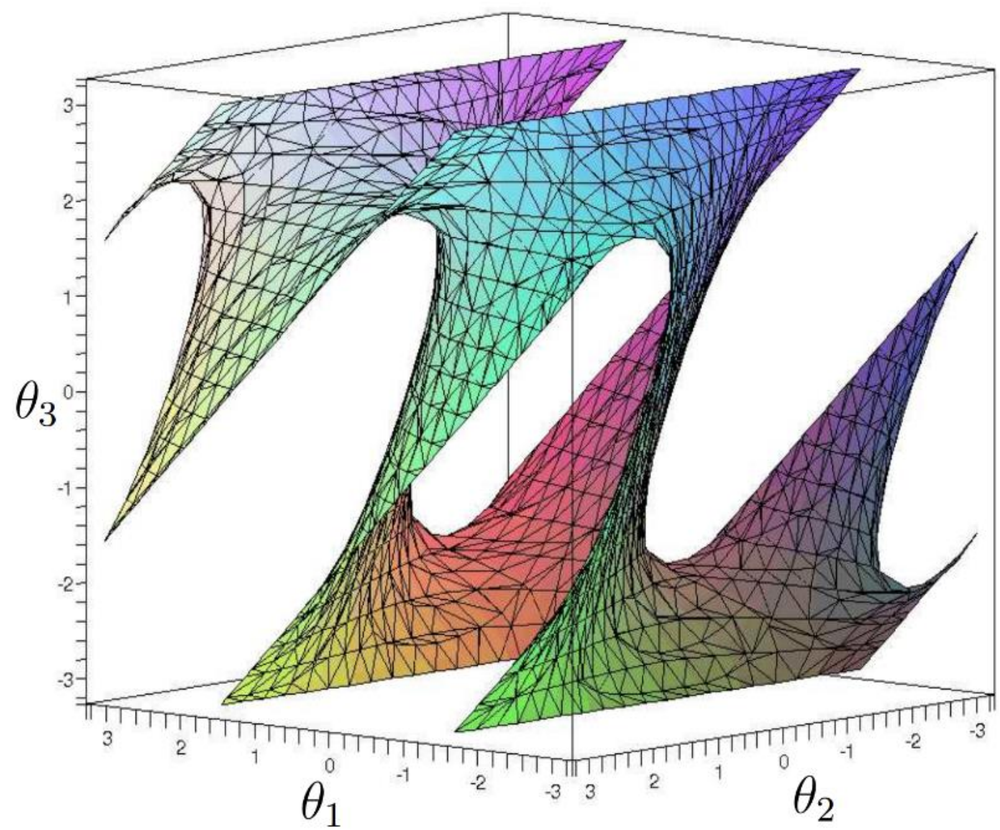
Same (x, y), different θ, in collision

- ⇨ Part of the space is “carved” out

How Does a Configuration Space Look Like?

A 3-chain line in 2D with one end on the origin and the other on y axis





Visualizing a 2-link arm https://www.cs.unc.edu/~jeffi/c-space/robot.xhtml

Image sources: http://planning.cs.uiuc.edu/ch4.pdf

Computing the Free Configuration Spaces

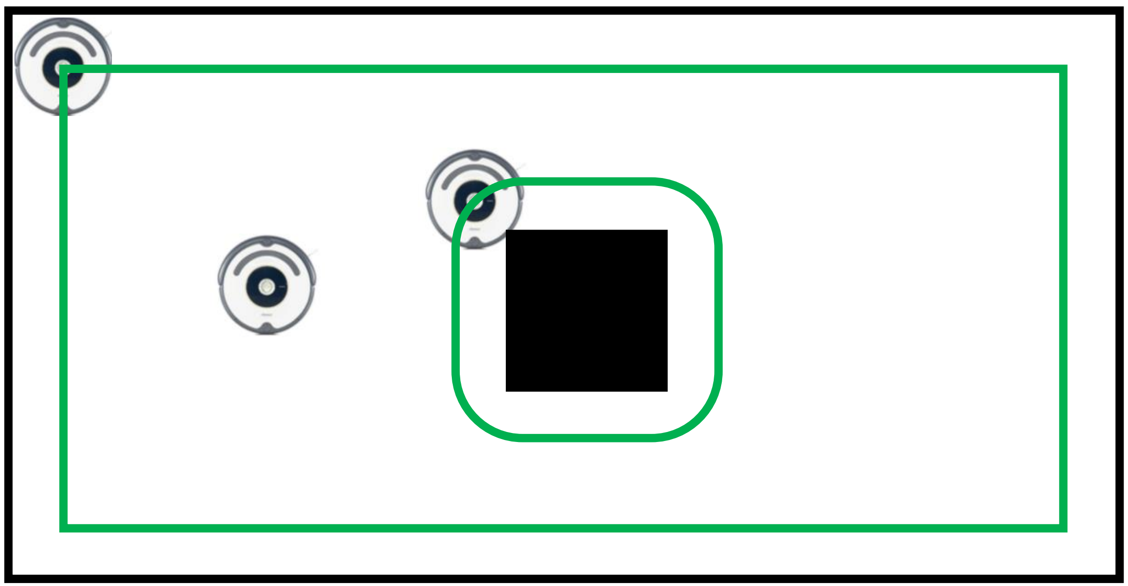
⇨The computation can be extremely challenging

⇨For easy cases, we can use Minkowski sum, **defined** as

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

⇨Example: disc robot in 2D (rotation invariant)

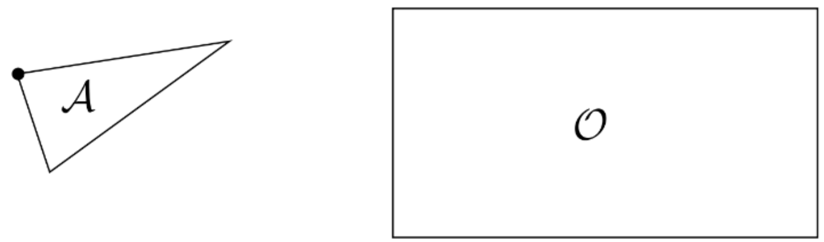
⇨A: an obstacle, $B = \{(x, y) \mid x^2 + y^2 \leq r^2\}$



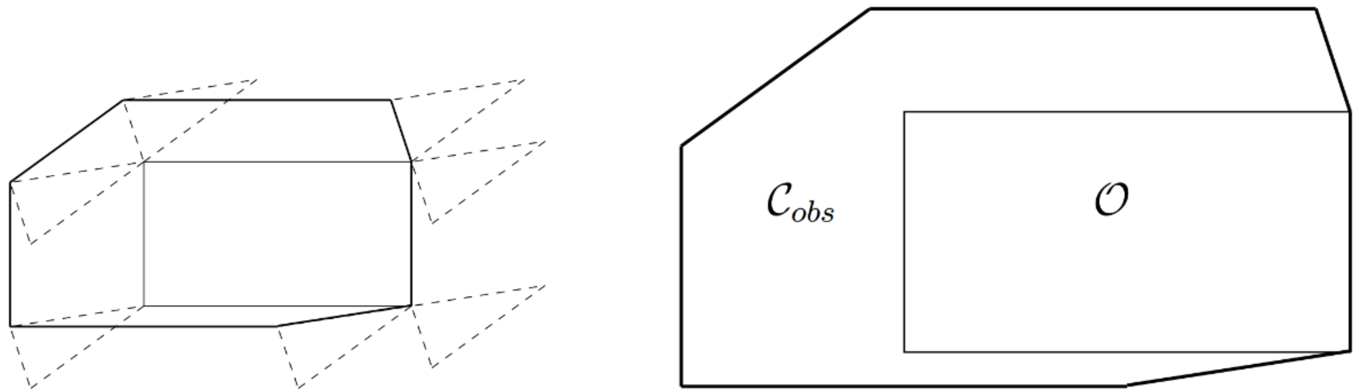
⇨The robot is now shrunk into a point!

A Slightly More Complex Example

⇨What about this case (A only translates but does not rotate)?



⇨We can do the same, or simply slide



⇨Rotation makes the computation much more complex (recall 3-link example)

Image sources: http://planning.cs.uiuc.edu/ch4.pdf