THE UNIVERSITY of EDINBURGH
informatics

Advanced Robotics

Rotations, placements, joint maps
Forward geometry

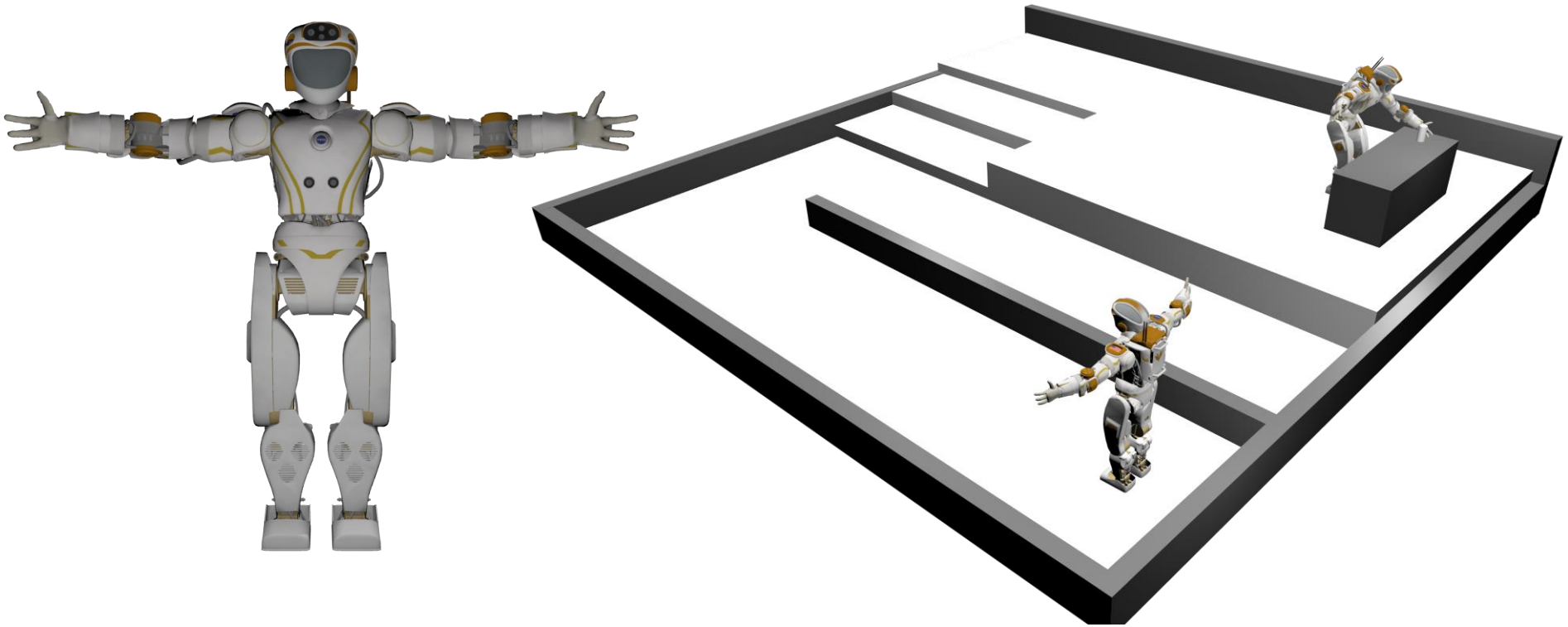
Steve Tonneau School of Informatics
University of Edinburgh

Reading for this week

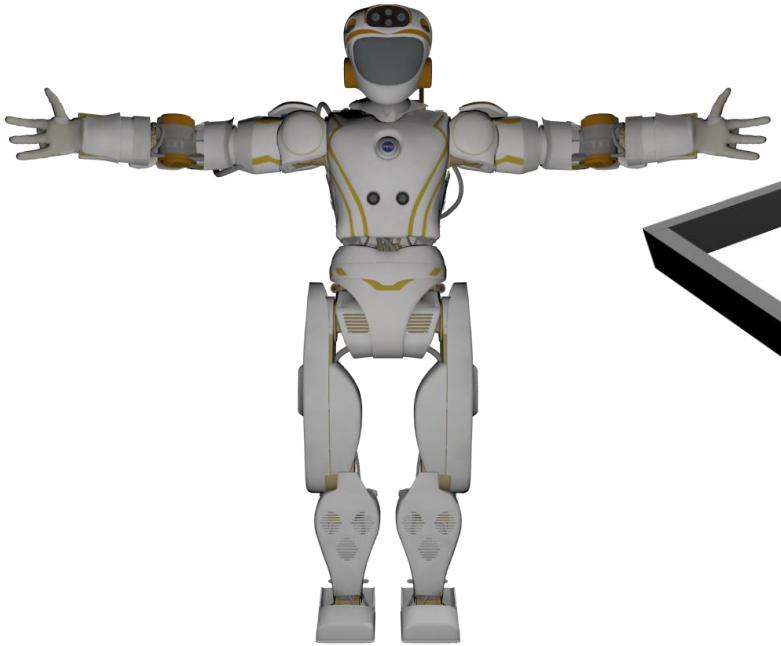
- Siciliano, B., et al., Robotics: Modelling, Planning and Control.

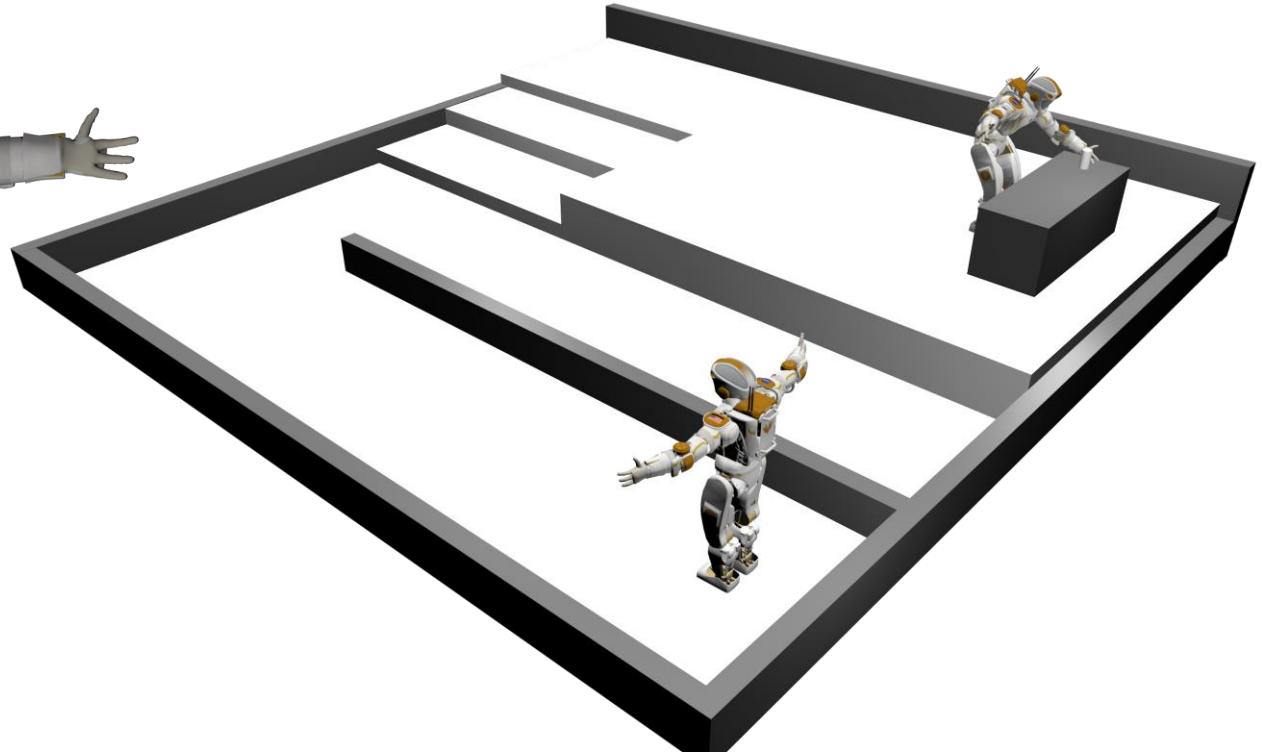
  Chapter 2.1 => 2.10 (note the slight difference in notation)

How to describe where a robot is in the world?

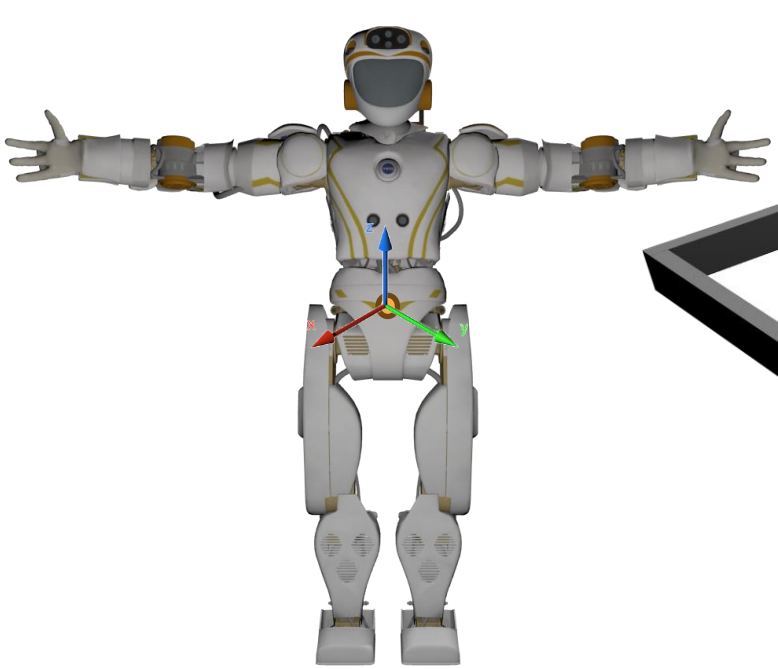


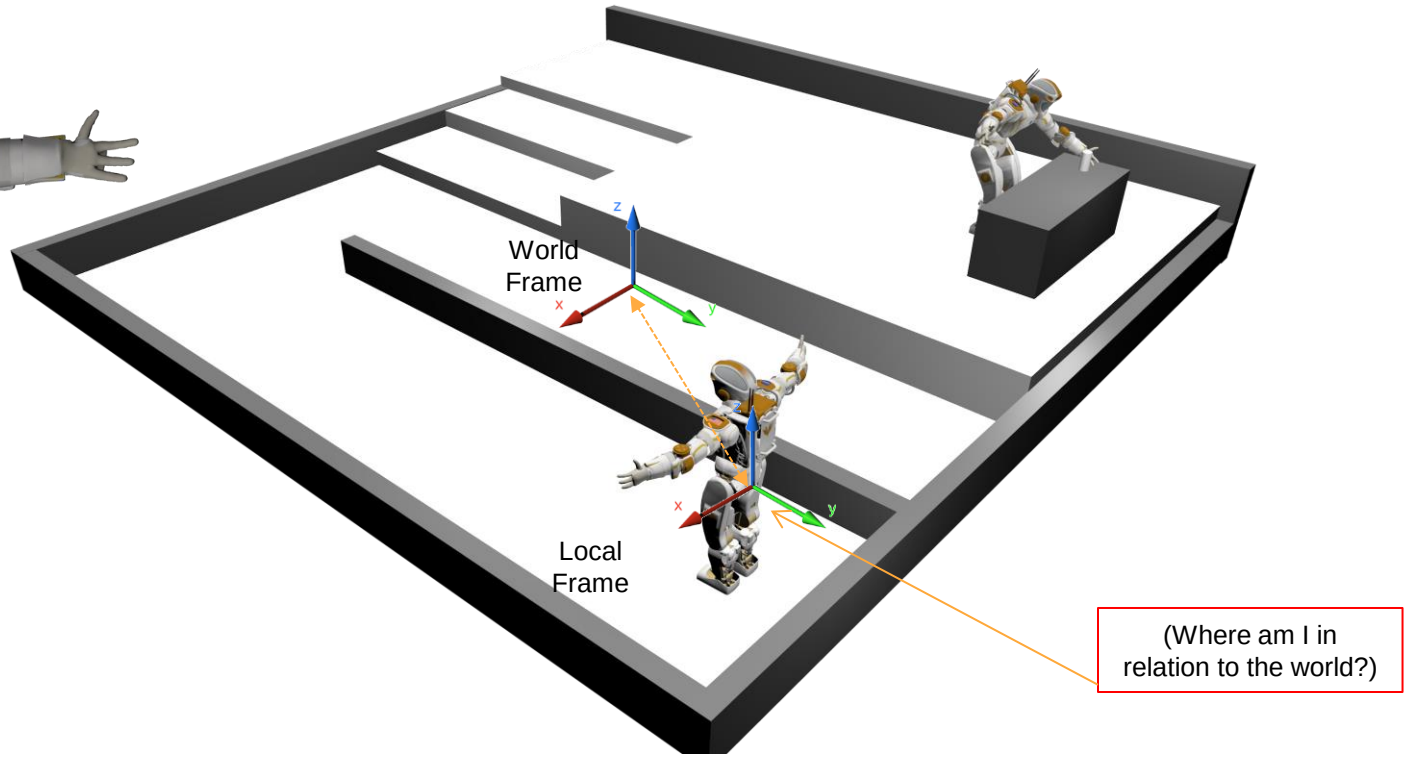
How to describe where a ~~robot is~~ you are in the world?



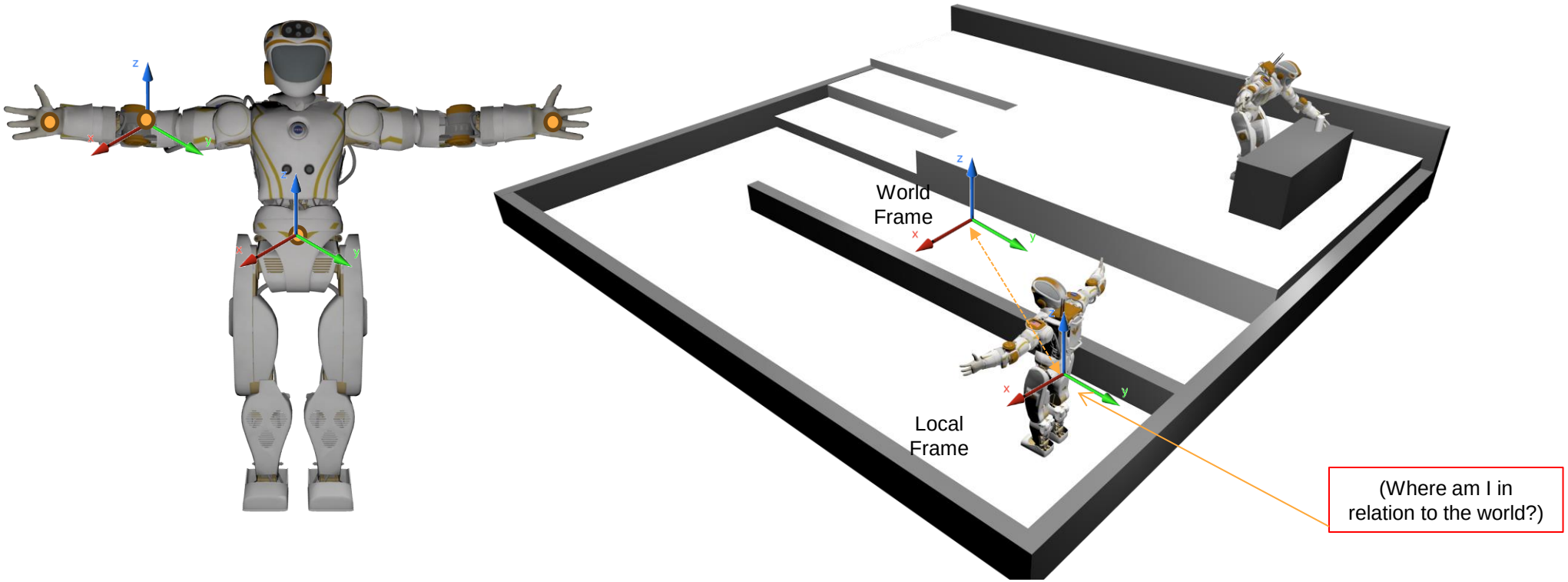


Recap: What Tools Do We Need?





Recap: What Tools Do We Need?



More generally, where is *any* point on the robot located in relation to the world or to each other?

Super fast recap: interrupt as needed

Coordinates: 2D Definition – Orthonormal basis

- The 2 dimensions are denoted using *x* and *y*

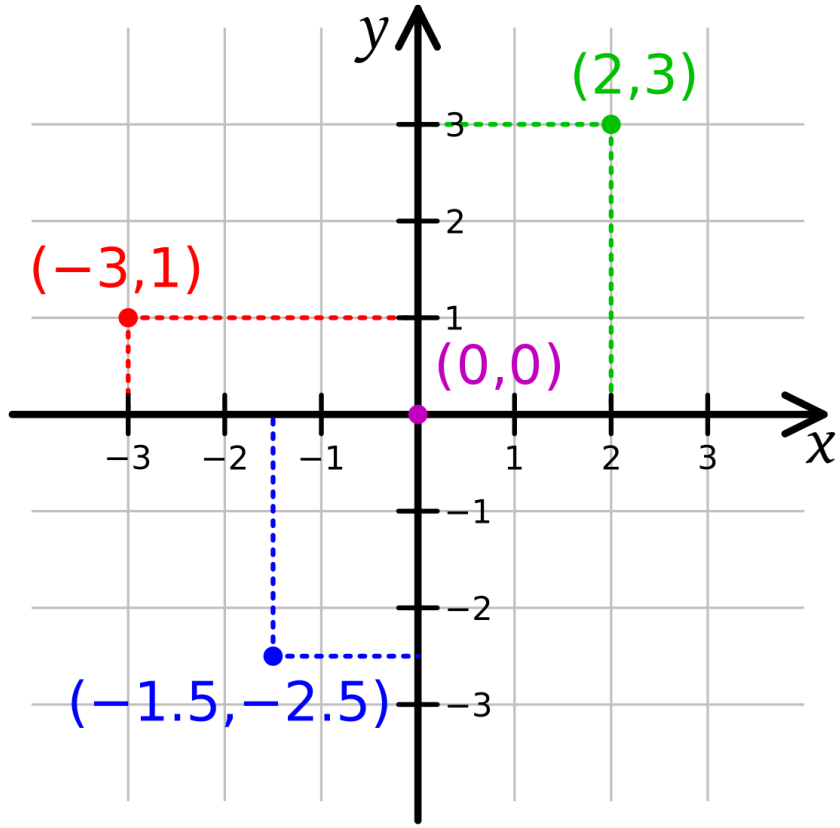
  X and Y unit, orthogonal unit vectors

- An origin is defined where *x* = 0 and *y* = 0

- Location on the plane **represented** as a vector:

$$^{O}\mathbf{p} = [x, y] \in \mathbb{R}^2$$

- Coordinates can be positive as well as negative

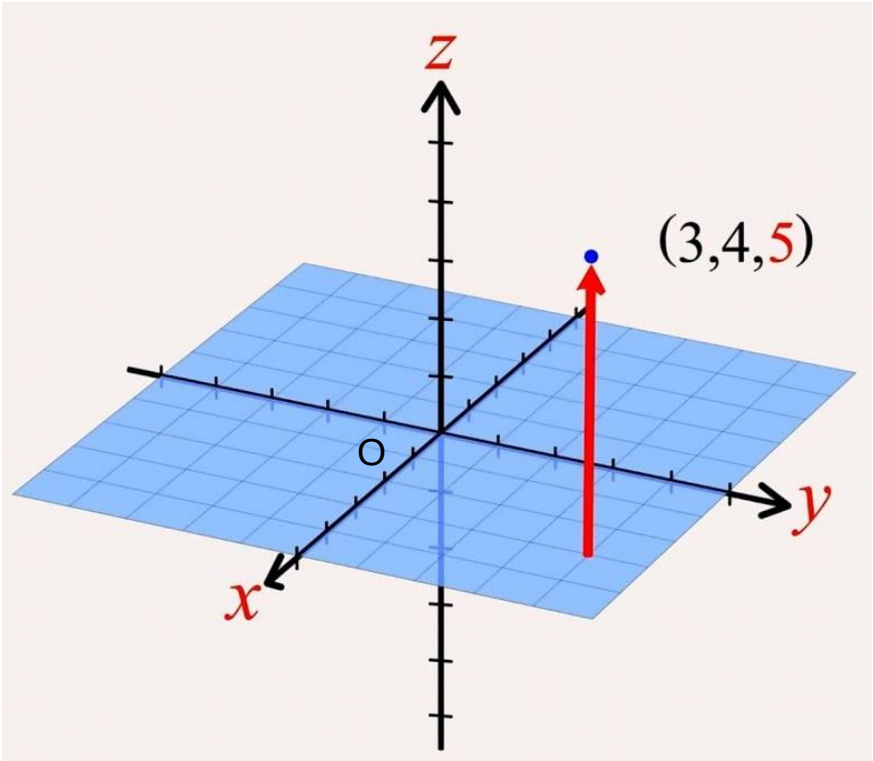


3D extension is straightforward

Points **represented** wrt an origin orthonormal frame O

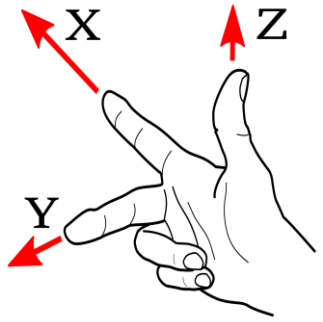
$$^{O}\mathbf{p} = [x, y, z] \in \mathbb{R}^3$$

$$\mathbf{O} = (0, 0, 0)$$



Convention for axis choice: right hand rule

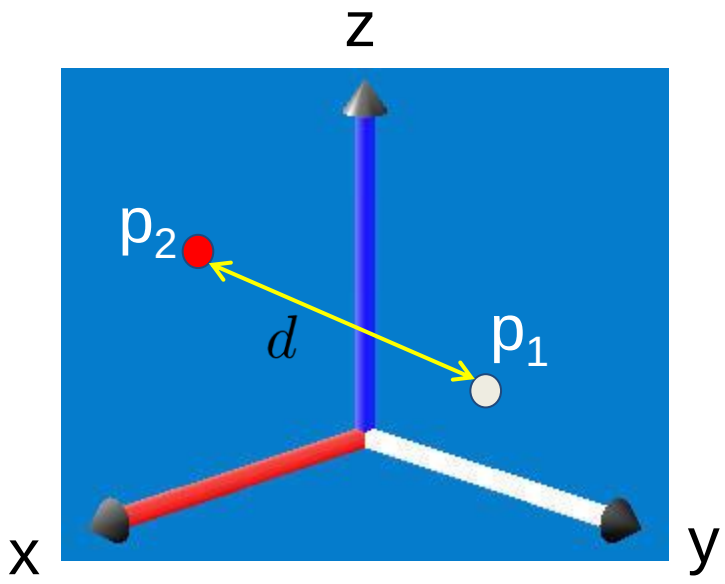
"X forward"



Euclidian distance

Given two points represented in the same frame:

$$^{O}\mathbf{p}_1 = [x_1, y_1, z_1] \qquad ^{O}\mathbf{p}_2 = [x_2, y_2, z_2]$$

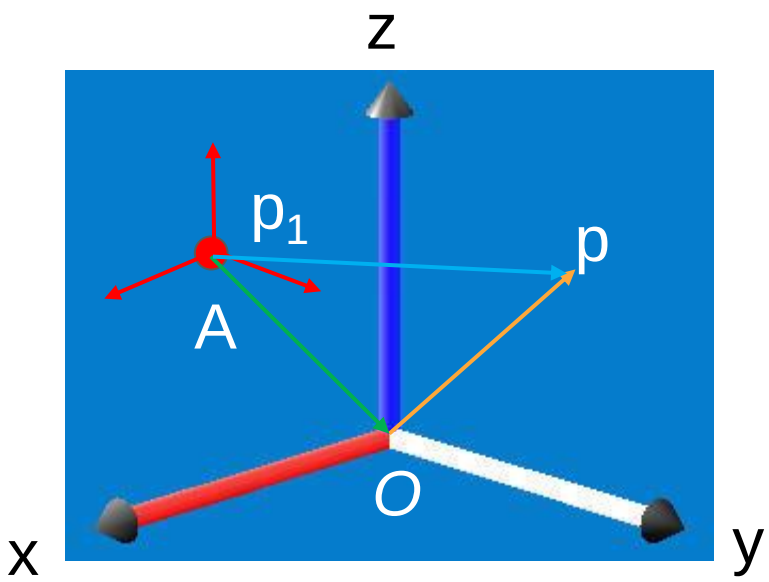


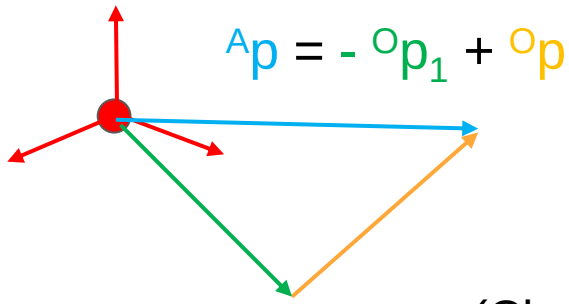
The Euclidian distance d between the points is :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Change of frame

A point p can be represented wrt to a **translated** frame using Chasles relation:





(Chasles relation)

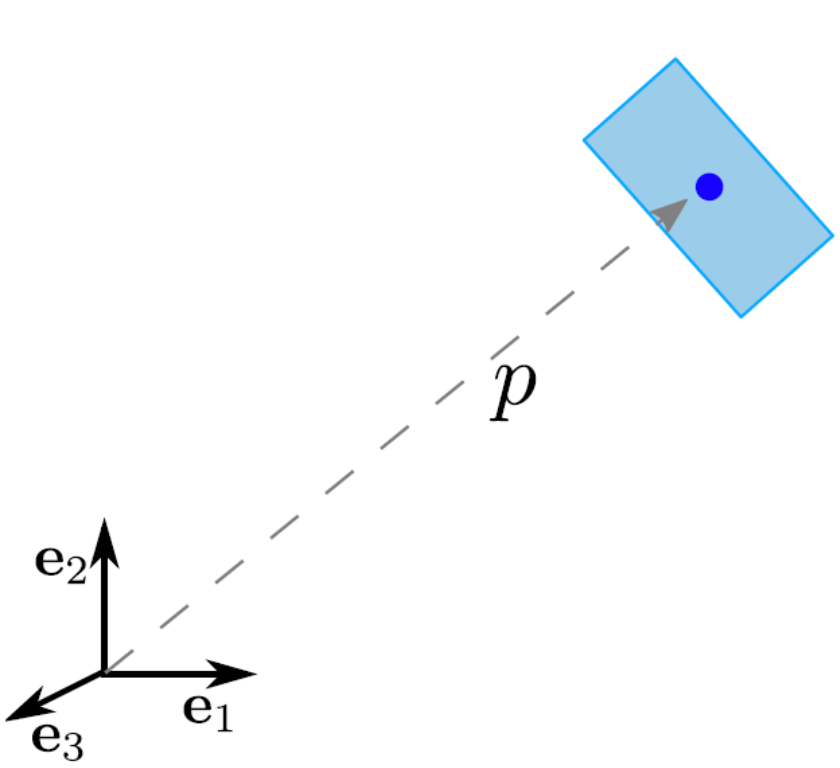
Representation of spatial relation

The coordinates in previous examples have all axes aligned.

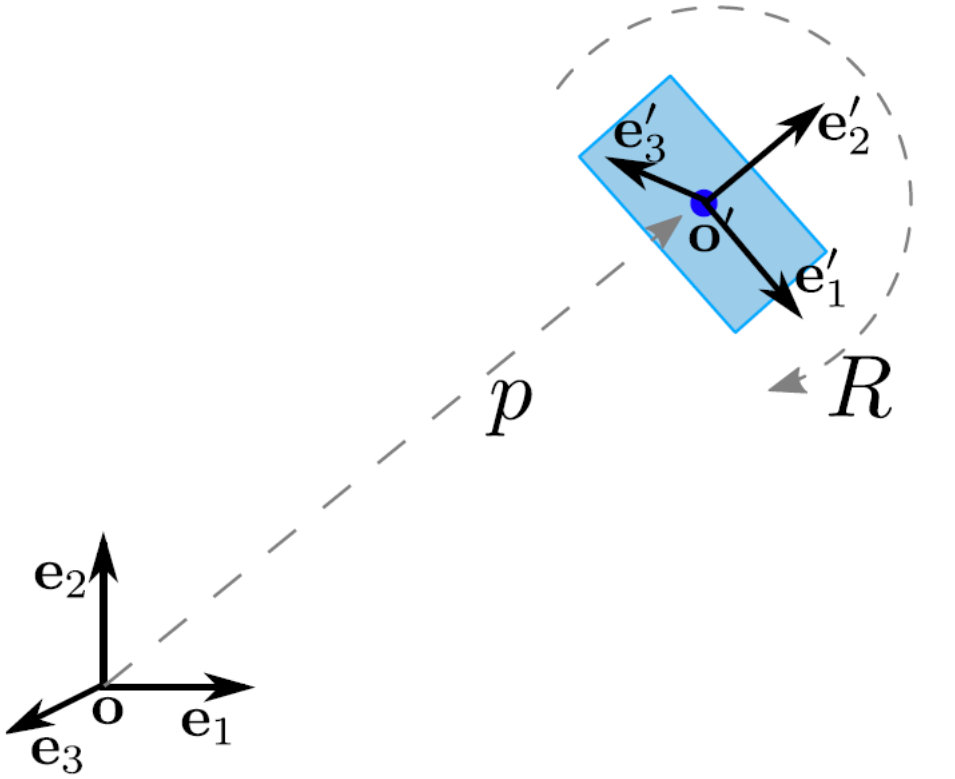
So what is still missing in a more general case?

Rotational representation

Rigid Body Position & Pose

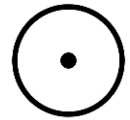


Position

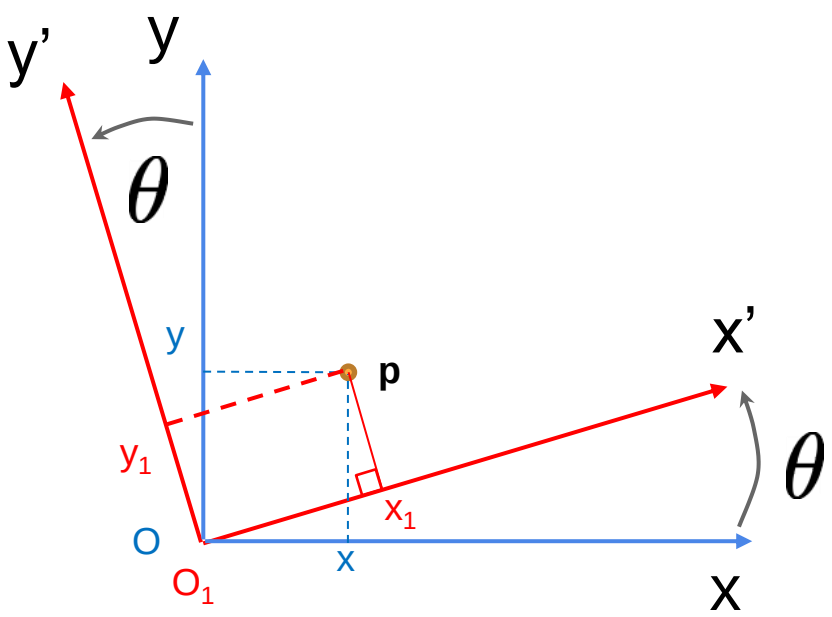


Pose = Position + Orientation

Example: a 2D Rotation

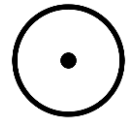
Transformation from O to O_1 : rotation of angle θ (around z axis) 

Given ${}^{O}\mathbf{p} = (x, y)$ vector in O frame, how to calculate ${}^{1}\mathbf{p}$?

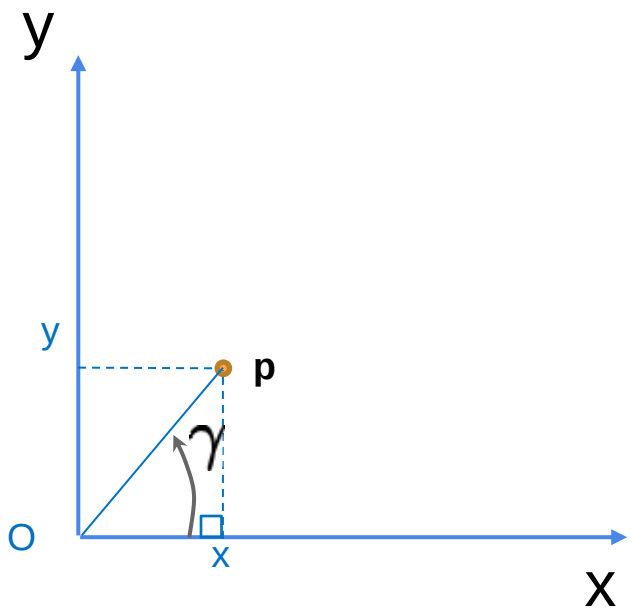


Example: a 2D Rotation

Transformation from O to O_1 : rotation of angle θ (around z axis)



Given ${}^{O}\mathbf{p} = (x, y)$ vector in O frame, how to calculate ${}^{1}\mathbf{p}$?

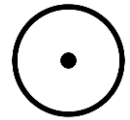


$x = |p|_2 * \cos(\gamma)$
$y = |p|_2 * \sin(\gamma)$

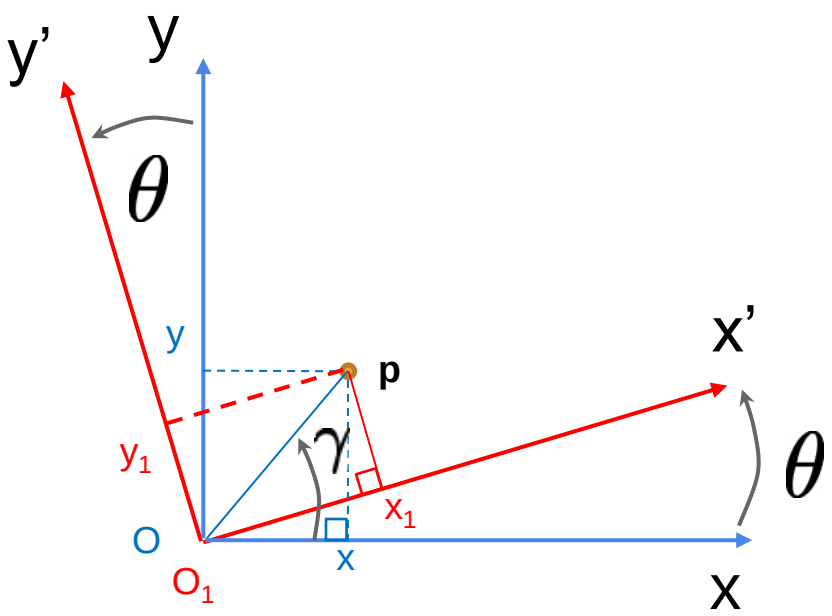
(SOH CAH TOA)

Example: a 2D Rotation

Transformation from O to O_1 : rotation of angle θ (around z axis)



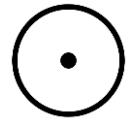
Given ${}^{O}\mathbf{p}$ = *(x, y)* vector in O frame, how to calculate ${}^{1}\mathbf{p}$?



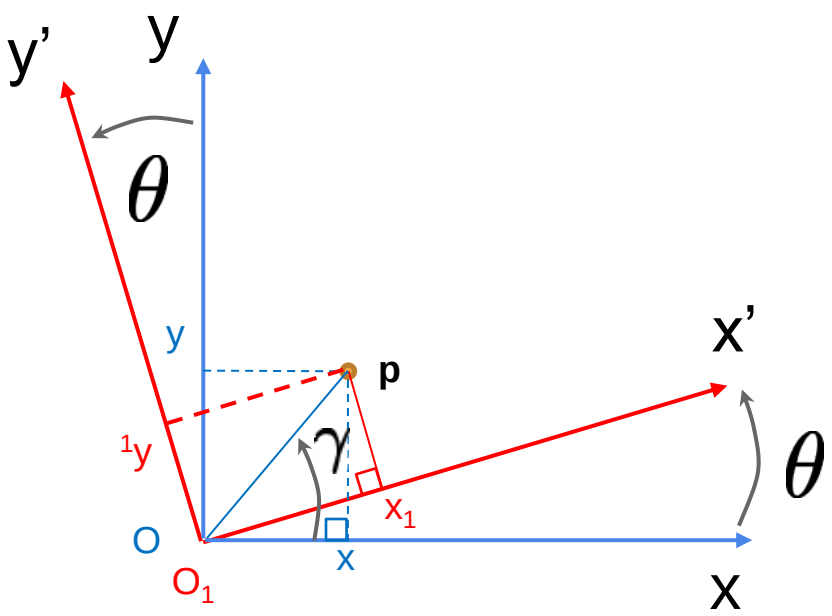
$x = |p|_2 * \cos(\gamma)$
$y = |p|_2 * \sin(\gamma)$

(SOH CAH TOA)

Example: a 2D Rotation

Transformation from O to O_1 : rotation of angle θ (around z axis) 

Given ${}^{O}\mathbf{p} = (x, y)$ vector in O frame, how to calculate ${}^{1}\mathbf{p}$?



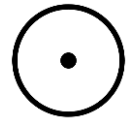
$x = |p|_2 * \cos(\gamma)$
$y = |p|_2 * \sin(\gamma)$

(SOH CAH TOA)

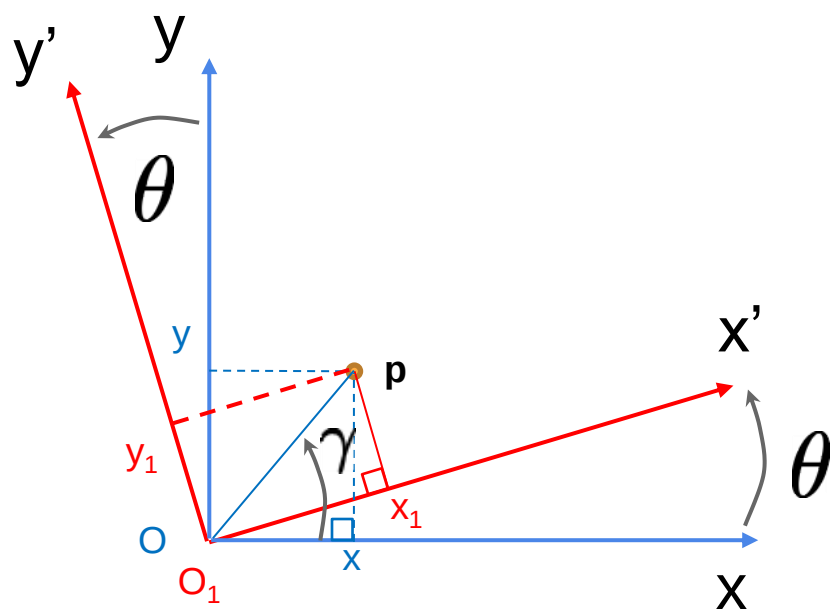
$x_1 = |p|_2 * \cos(\gamma - \theta)$
$y_1 = |p|_2 * \sin(\gamma - \theta)$

Example: a 2D Rotation

Transformation from O to O_1 : rotation of angle θ (around z axis)



Given ${}^{O}\mathbf{p}$ = *(x, y)* vector in O frame, how to calculate ${}^{1}\mathbf{p}$?



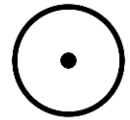
$x = |p|_2 * \cos(\gamma)$
$y = |p|_2 * \sin(\gamma)$

(SOH CAH TOA)

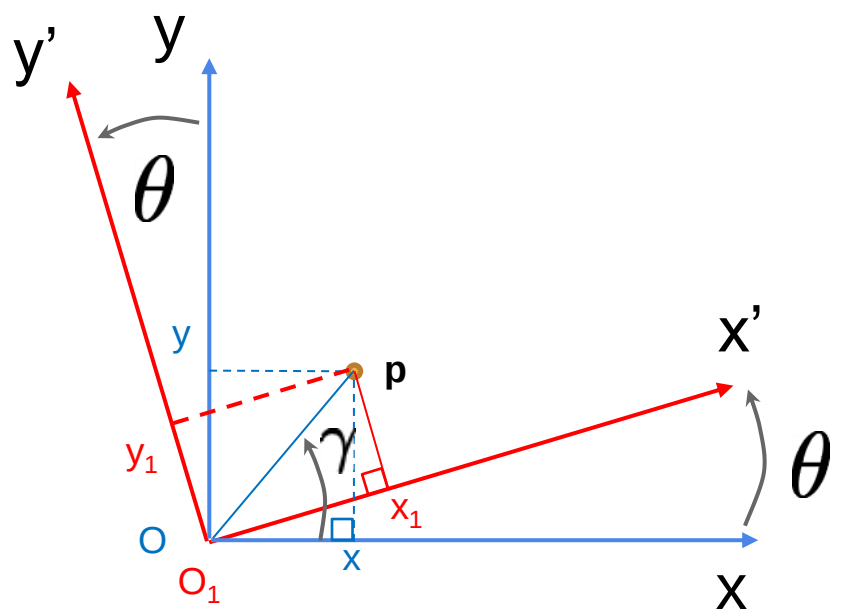
$x_1 = |p|_2 * \cos(\gamma - \theta) = |p|_2 [\cos(\gamma)\cos(\theta) + \sin(\gamma)\sin(\theta)]$
$y_1 = |p|_2 * \sin(\gamma - \theta) = |p|_2 [\sin(\gamma)\cos(\theta) - \cos(\gamma)\sin(\theta)]$

$$\sin(-a) = -\sin a$$
$$\cos(-a) = \cos a$$
$$\cos(a+b) = \cos a \cos b - \sin a \sin b$$
$$\sin(a+b) = \sin a \cos b + \cos a \sin b$$

Example: a 2D Rotation

Transformation from O to O_1 : rotation of angle θ (around z axis) 

Given ${}^{O}\mathbf{p}$ = *(x, y)* vector in O frame, how to calculate ${}^{1}\mathbf{p}$?



$x = |p|_2 * \cos(\gamma)$
$y = |p|_2 * \sin(\gamma)$

(SOH CAH TOA)

$x_1 = |p|_2 * \cos(\gamma - \theta) = |p|_2 [\cos(\gamma)\cos(\theta) + \sin(\gamma)\sin(\theta)]$
$y_1 = |p|_2 * \sin(\gamma - \theta) = |p|_2 [\sin(\gamma)\cos(\theta) - \cos(\gamma)\sin(\theta)]$

$x_1 = x\cos(\theta) + y\sin(\theta)$
$y_1 = y\cos(\theta) - x\sin(\theta)$

$$\sin(-a) = -\sin a$$
$$\cos(-a) = \cos a$$
$$\cos(a+b) = \cos a \cos b - \sin a \sin b$$
$$\sin(a+b) = \sin a \cos b + \cos a \sin b$$

Matrix form of the rotation

$x_1 = x \cos(\theta) + y \sin(\theta)$
$y_1 = y \cos(\theta) - x \sin(\theta)$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Counter-intuitive rotation by - θ

R is **orthogonal**, which means that its inverse is also its transpose, implying:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}^{-1}} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

rotation by θ

Verify as an exercise that $(R^{-1}) = R^T$

R is a rotation matrix

- Self corrective notation:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}}_{{}^{1}\mathbf{R}_{0}} \begin{bmatrix} x \\ y \end{bmatrix} \quad \Rightarrow \quad {}^{1}\mathbf{p} = {}^{1}\mathbf{R}_{\cancel{O}}\,{}^{\cancel{O}}\mathbf{p}$$

« from O representation into frame O_1 representation »

$$\begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{{}^{1}\mathbf{R}_{0}^{-1}} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \Rightarrow \quad {}^{0}\mathbf{p} = {}^{0}\mathbf{R}_{\cancel{1}}\,{}^{\cancel{1}}\mathbf{p}$$

« from O_1 representation into frame O representation»

$$\left({}^{1}\mathbf{R}_{0}\right)^{-1} = {}^{0}\mathbf{R}_{1}$$

Pure rotations in 2D and 3D – SO(2) and SO(3)

The sets of all matrices $\mathbf{R} \in \mathbb{R}^{n \times n}$ (with n = 2 or 3), such that

$$R^{-1} = R^{\top} \qquad \det(R) = 1$$

are the algebraic groups called SO(2) and SO(3) for Special Orthogonal groups. Any pure rotation can be **represented** with a matrix of SO(n), and the matrix multiplication is the group operator that applies the rotation.

$$\mathbf{R} = \mathbf{R}_2\mathbf{R}_1 \quad <=> \quad$$ « Apply rotation R1 to p, then apply rotation R2 to the result »

More on SO(3) and other representations tomorrow / at the end of the class

Simple Rotation Matrices

- 2D

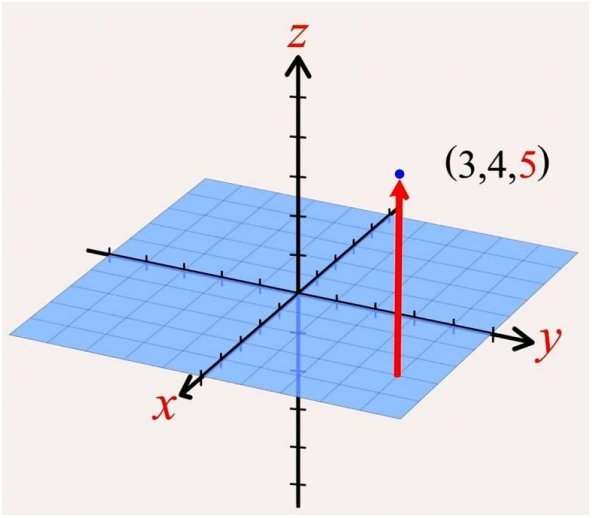
$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

- 3D – Any rotation obtained by composing:

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

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

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



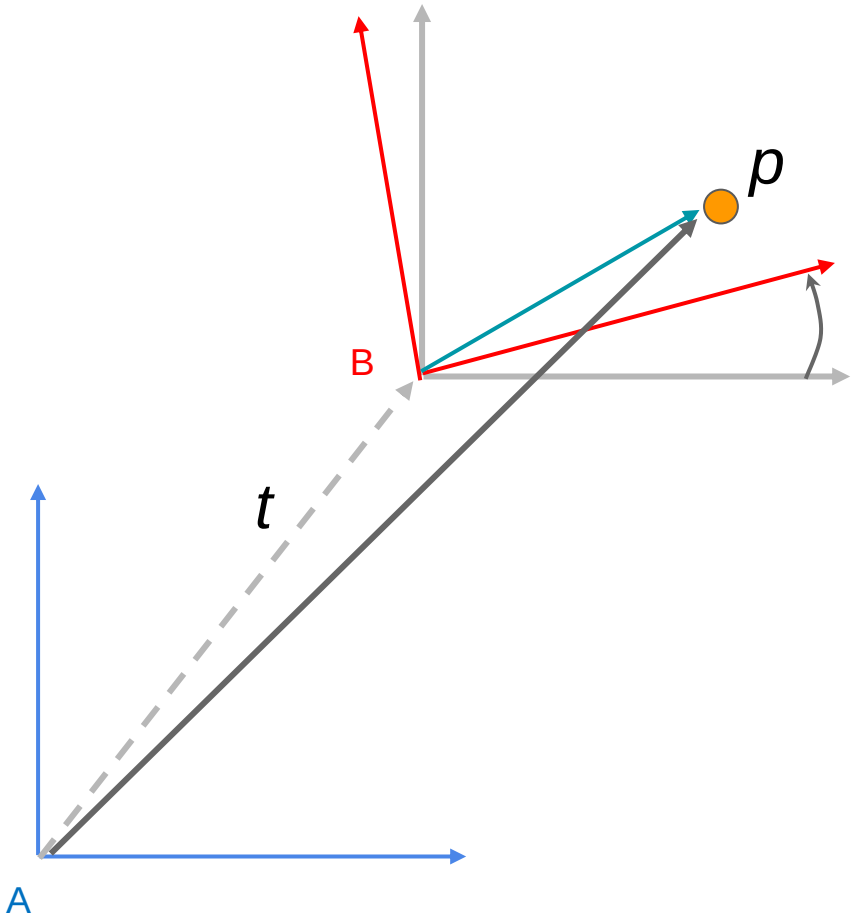
Composing rotations and translations

$$^{A}\mathbf{p} = {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{p} + \mathbf{t}$$

t vector and frame independent

The group of all transformations that consist in 3D rotations, translations, or arbitrary combinations of them is called the Special Euclidian group SE(3).

Again more tomorrow



Homogeneous Transformation Matrix trick

- ${}^{A}\mathbf{p} = {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{p} + \mathbf{t}$ => annoying to write (especially when composing)

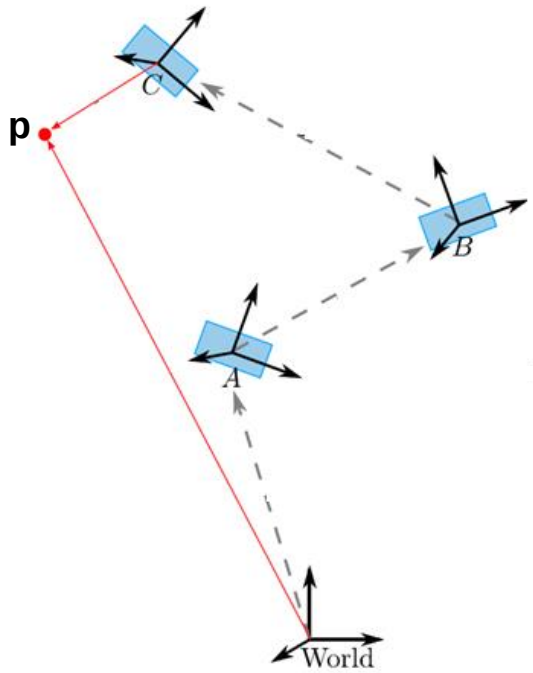
- This operation can be written in matrix form:

$$\begin{bmatrix} {}^{A}\mathbf{p} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{A}\mathbf{R}_{B} & t \\ \mathbf{0}_{3} & 1 \end{bmatrix}}_{{}^{A}\mathbf{M}_{B} \in \mathbb{R}^{4\times 4}} \begin{bmatrix} {}^{B}\mathbf{p} \\ 1 \end{bmatrix}$$

- With

$${}^{B}\mathbf{M}_{A} = ({}^{A}\mathbf{M}_{B})^{-1} = \begin{bmatrix} {}^{B}\mathbf{R}_{A} & -{}^{B}\mathbf{R}_{A}t \\ \mathbf{0}_{3} & 1 \end{bmatrix}$$

Composition of Transformations in SE(3)



$$^{W}\mathbf{p} = {}^{W}\mathbf{M}_{A}\,{}^{A}\mathbf{M}_{B}\,{}^{B}\mathbf{M}_{C}\,{}^{C}\mathbf{p}$$

Summary and todos

An element of the group….

Can be represented as …

$r \in SO(3)$ $\simeq$ $\mathbf{R} \in \mathbb{R}^{3\times3}$ Rotation matrix

rotation

$m \in SE(3)$ $\simeq$ $\mathbb{R}^3 \times SO(3)$ $\simeq$ $\mathbb{R}^3 \times \mathbb{R}^{3\times3} \simeq \mathbb{R}^{4\times4}$ Homogeneous matrix

displacement

translation rotation

Summary and todos

An element of the group….

Can be represented as …

$r \in SO(3)$ $\simeq$ $\mathbf{R} \in \mathbb{R}^{3\times3}$ Rotation matrix

rotation

$\mathbf{q} \in \mathbb{H} \simeq \mathbb{R}^4, \|\mathbf{q}\| = \mathbf{1}$ quaternion

$\omega \in so(3) \simeq \mathbb{R}^3$ A velocity ?

$m \in SE(3)$ $\simeq$ $\mathbb{R}^3 \times SO(3)$ $\simeq$ $\mathbb{R}^3 \times \mathbb{R}^{3\times3} \simeq \mathbb{R}^{4\times4}$ Homogeneous matrix

displacement

translation rotation

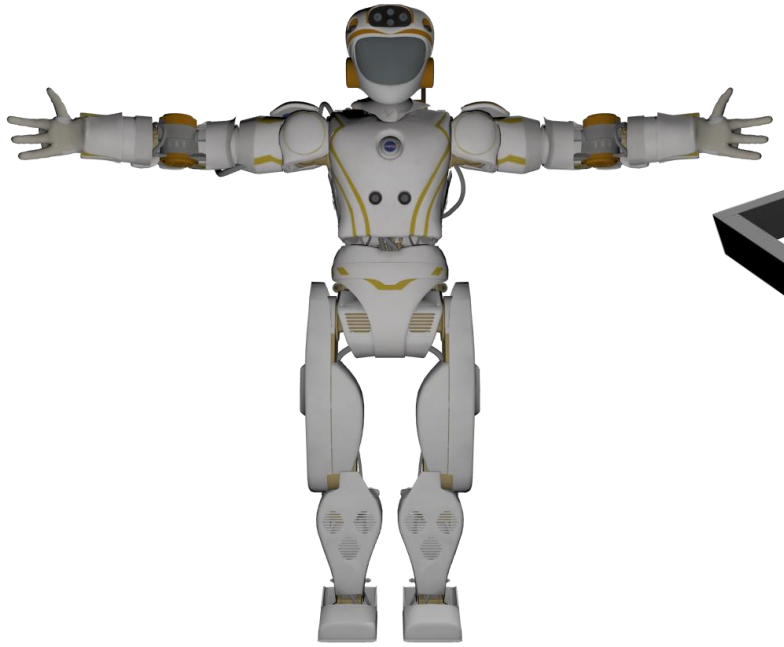
$\mathbb{R}^3 \times \mathbb{H} \simeq \mathbb{R}^7$

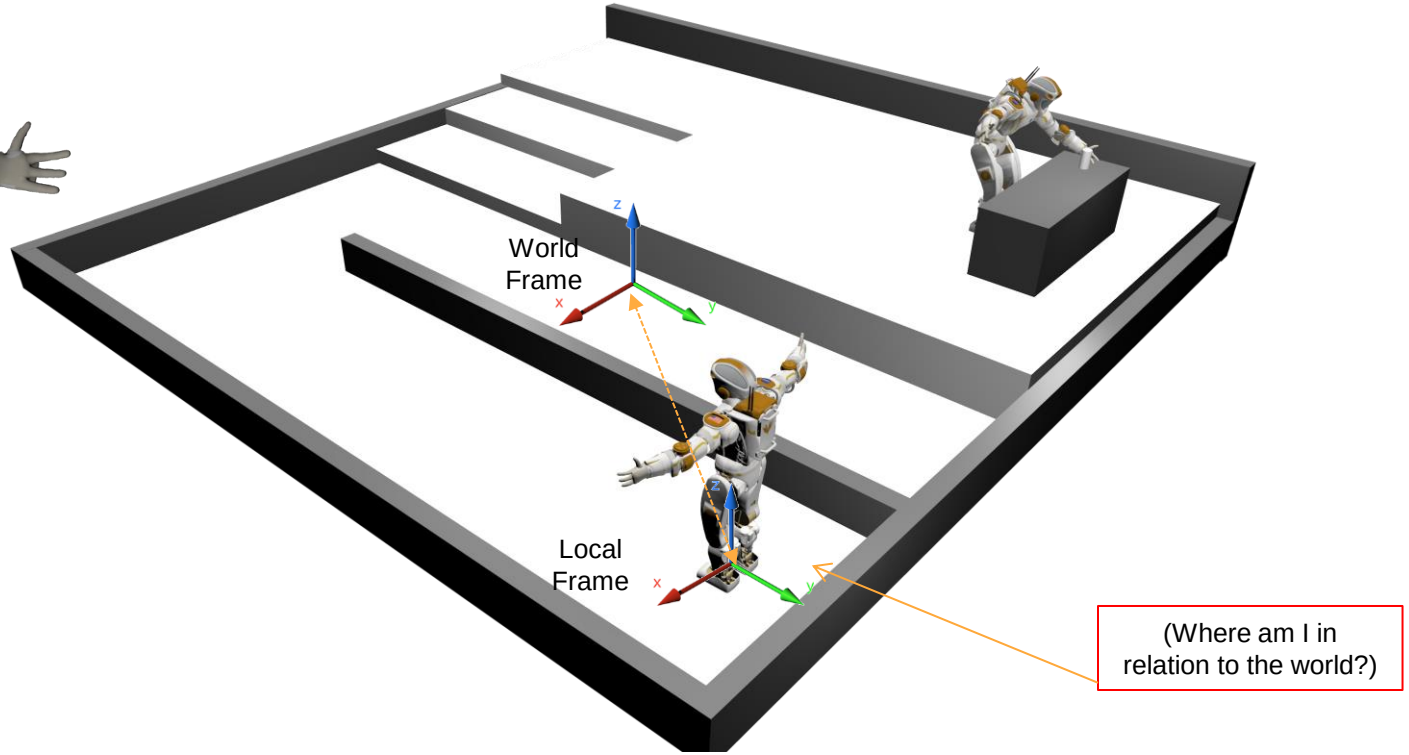
$\mathcal{V} \in se(3) = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \simeq \mathbb{R}^6$ A spatial velocity ?

Why is it so complicated to represent a rotation ?
Answer tomorrow

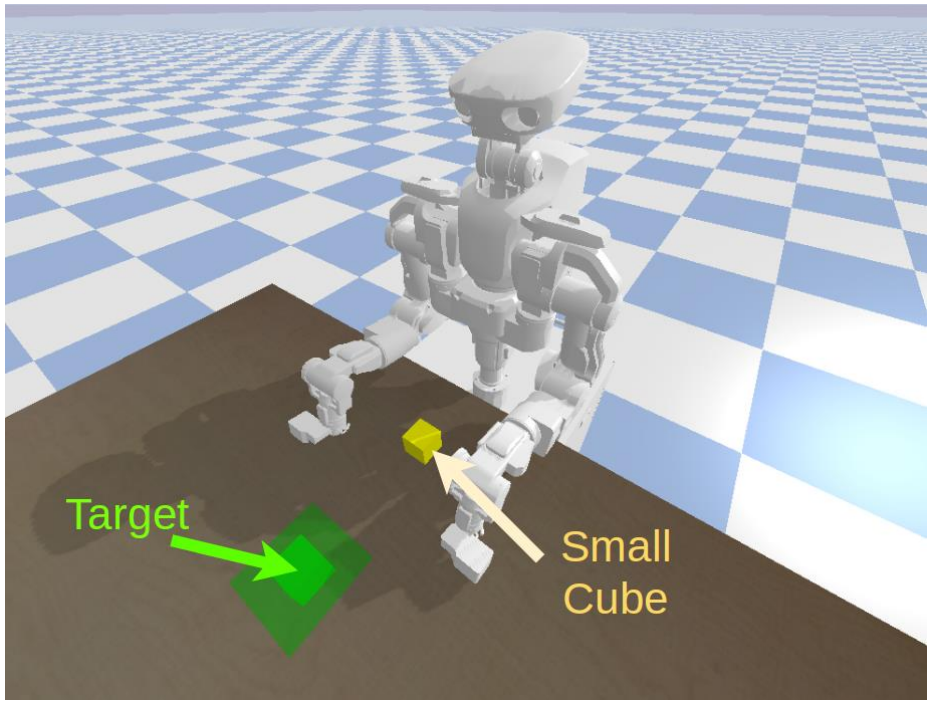
Joint maps and kinematic tree

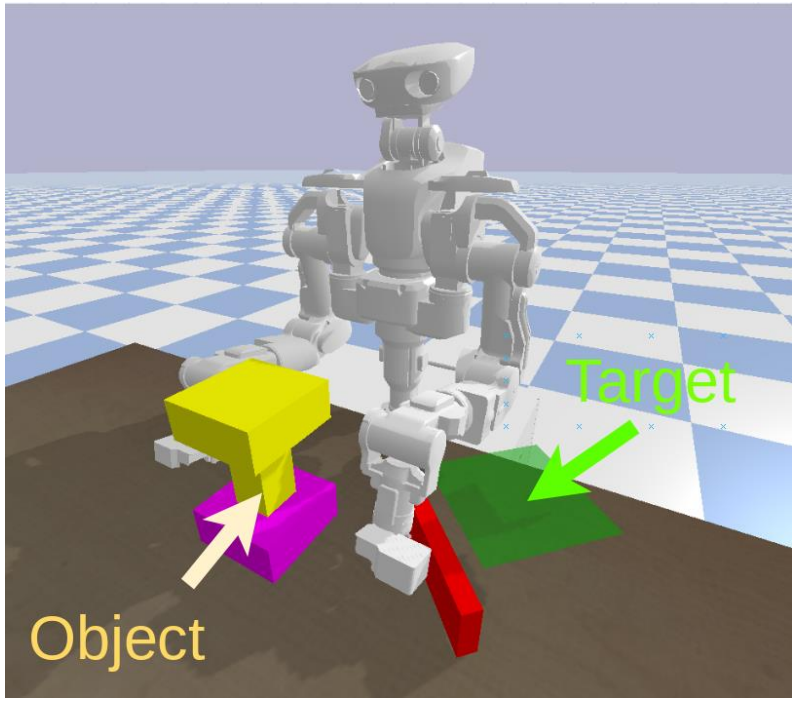
We know how to describe a robot position in the world





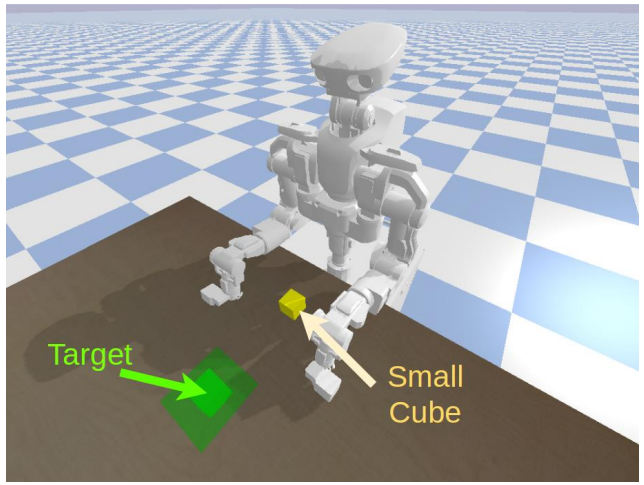
That does not tell us how to do this:

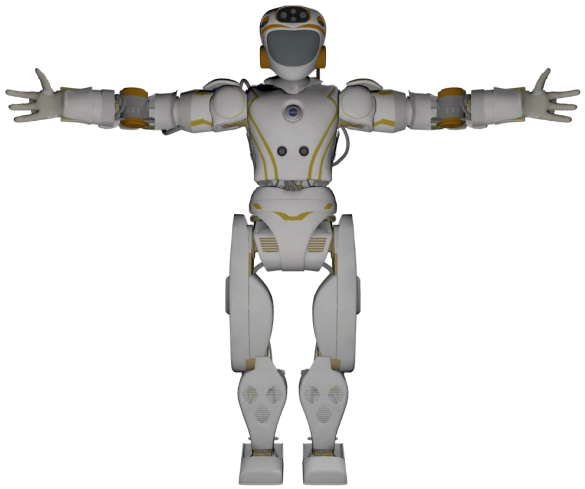




Today: How to describe the robot internal state ?

- Legged / manipulator robots are articulated
- We need to describe efficiently the robot posture, or configuration
- What do you suggest?





Kinematics?

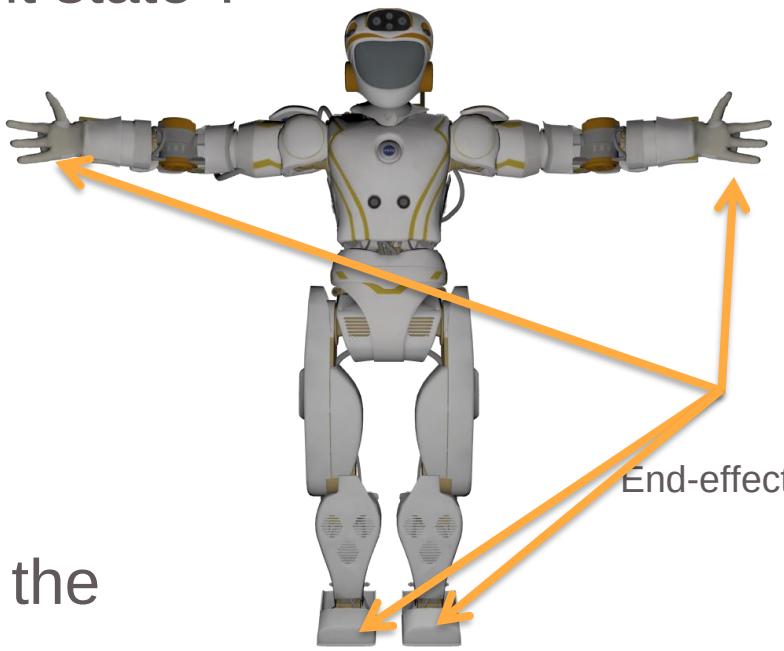
- Move all the joints (articulations) in a coordinated way such that the ***end-effector*** makes the desired movement

What relationship between the end-effector and the joint state ?

- when we know/set the joint state,
  where is the end-effector?

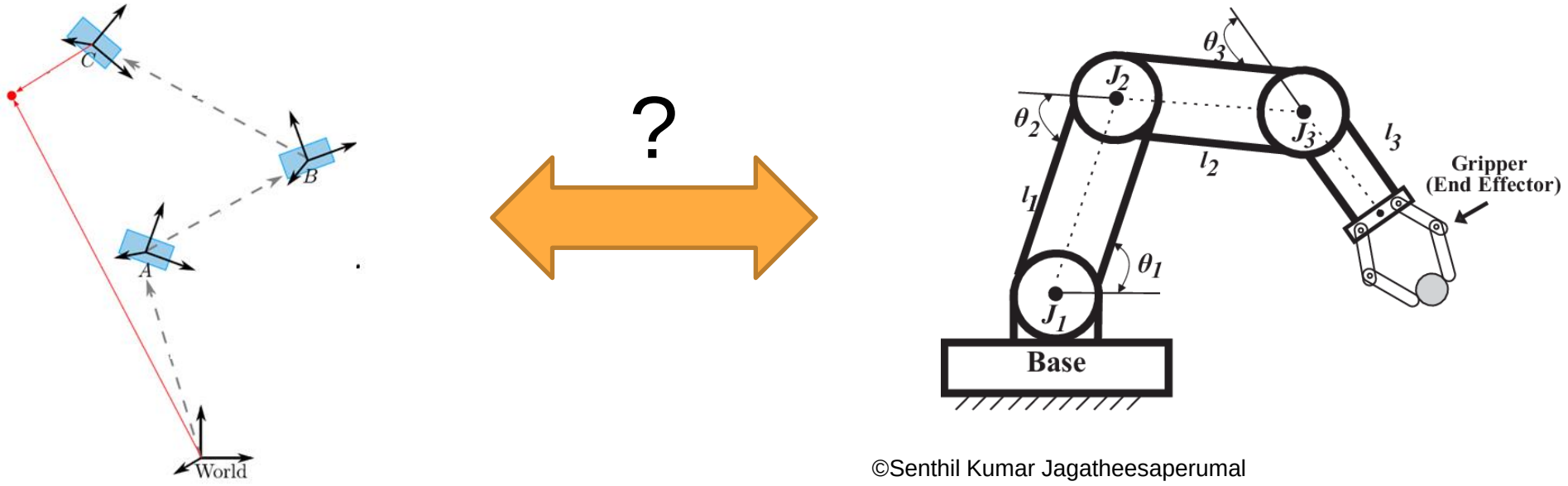
- when we change the joint state,
  how does the end-effector change position?

- Control: how to modify the joint state to reach a desired target ? (later, what command to send to the motors to achieve this?)



The kinematic tree

- Robot posture / configuration can be described by a set of frames

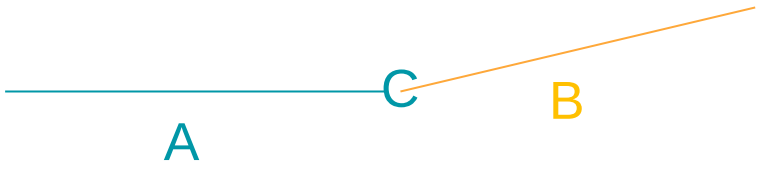


©Senthil Kumar Jagatheesaperumal

- Where do we set the frames? Is there a compact way to describe the posture? What is fixed / variable in the robot description ?

Joint definition

- A joint is a **mechanical constraint** between the placement of two rigid bodies A and B.

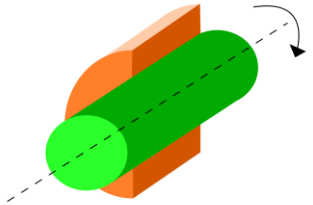


- It limits the placements of B with respect to A (his parent)

- Mathematically, this results on a placement ${}^{A}M_{B}$ to have a specific form

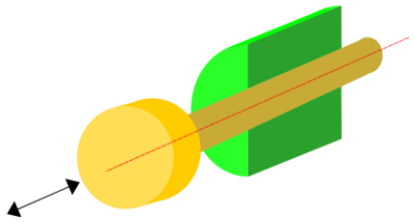
A subset of the possible joint Types

revolute joint: $\theta \in \mathbb{R}$ ⇔ rotation along an axis (e.g. X) ⇔



$$T_{A\to A'}(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

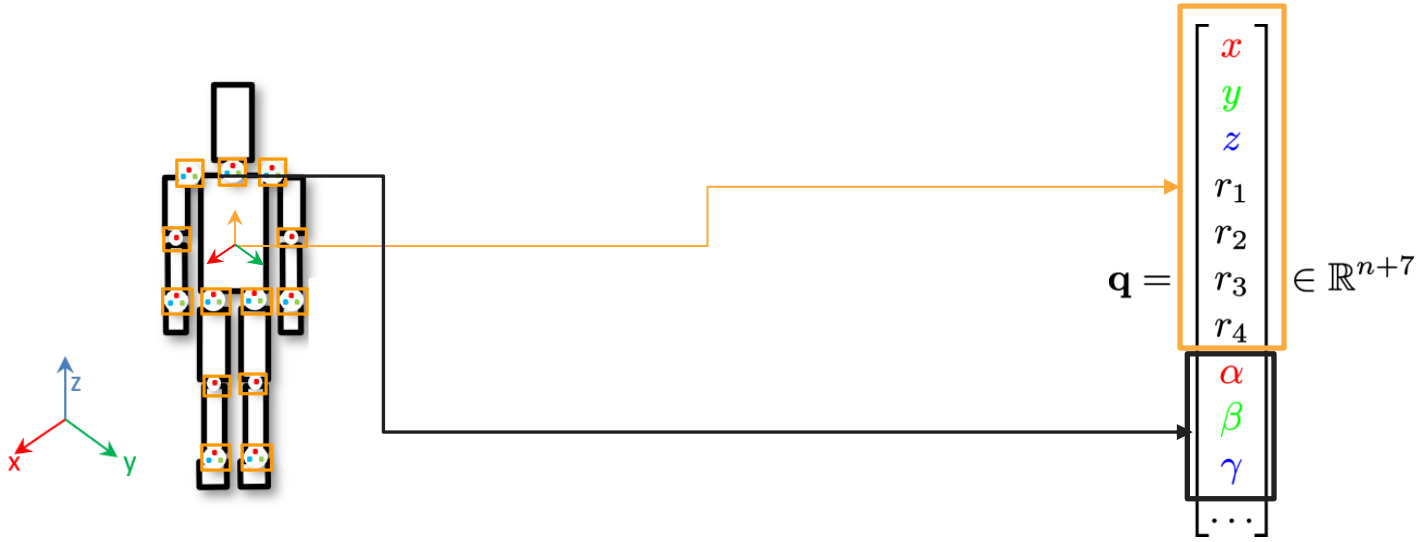
prismatic joint: $t \in \mathbb{R}$ ⇔ translation along an axis (eg X)



$$T_{A\to A'}(q) = \begin{bmatrix} 1 & 0 & 0 & t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The configuration space (Lozano-Peréz 83)

- Robot posture is a point **q** in the configuration space C, of dimension n, or n+6 if root is free (free-flyer joint), with n number of internal Degrees Of Freedom (dof)
  - Each internal dof represented by a joint parameter, subset of **q**
  - If using quaternions to represent free-flyer rotation, **q** is ***represented*** with n+7 != n+6 variables



q is used to describe both a quaternion and a configuration in the littérature …

3D environment W

Configuration space C

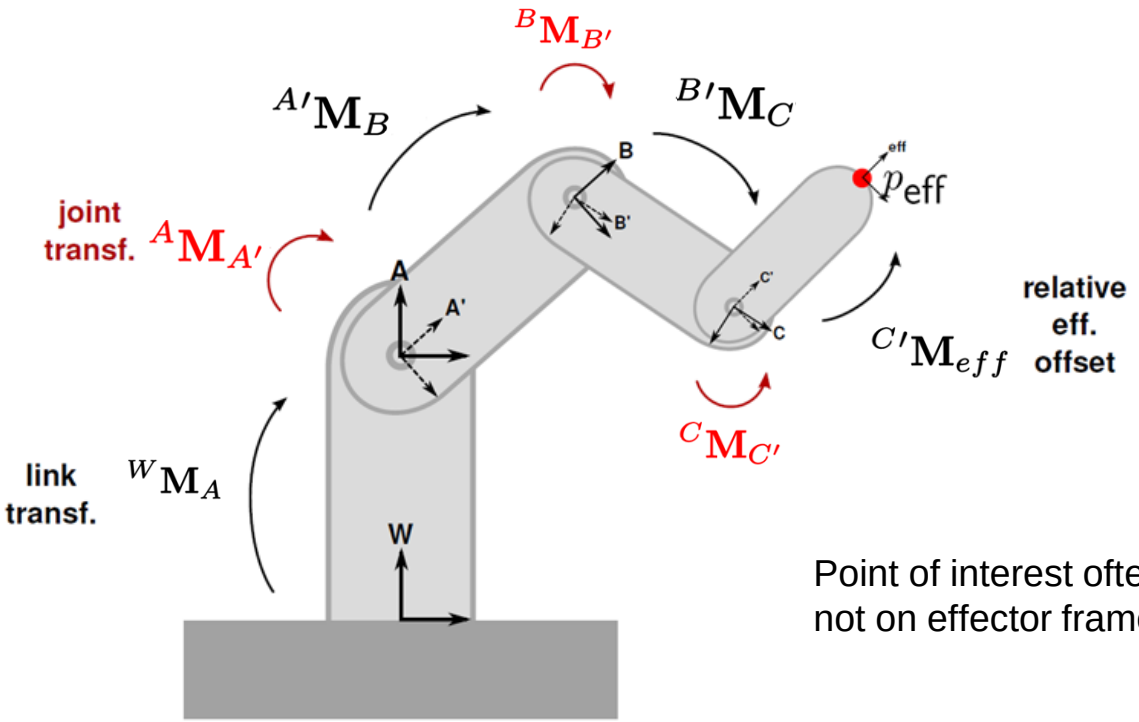
Kinematic chain and map

Generally, frames placed as follows to simplify the variable transformations

Denavit–Hartenberg (D-H) Convention

Black placements are constant

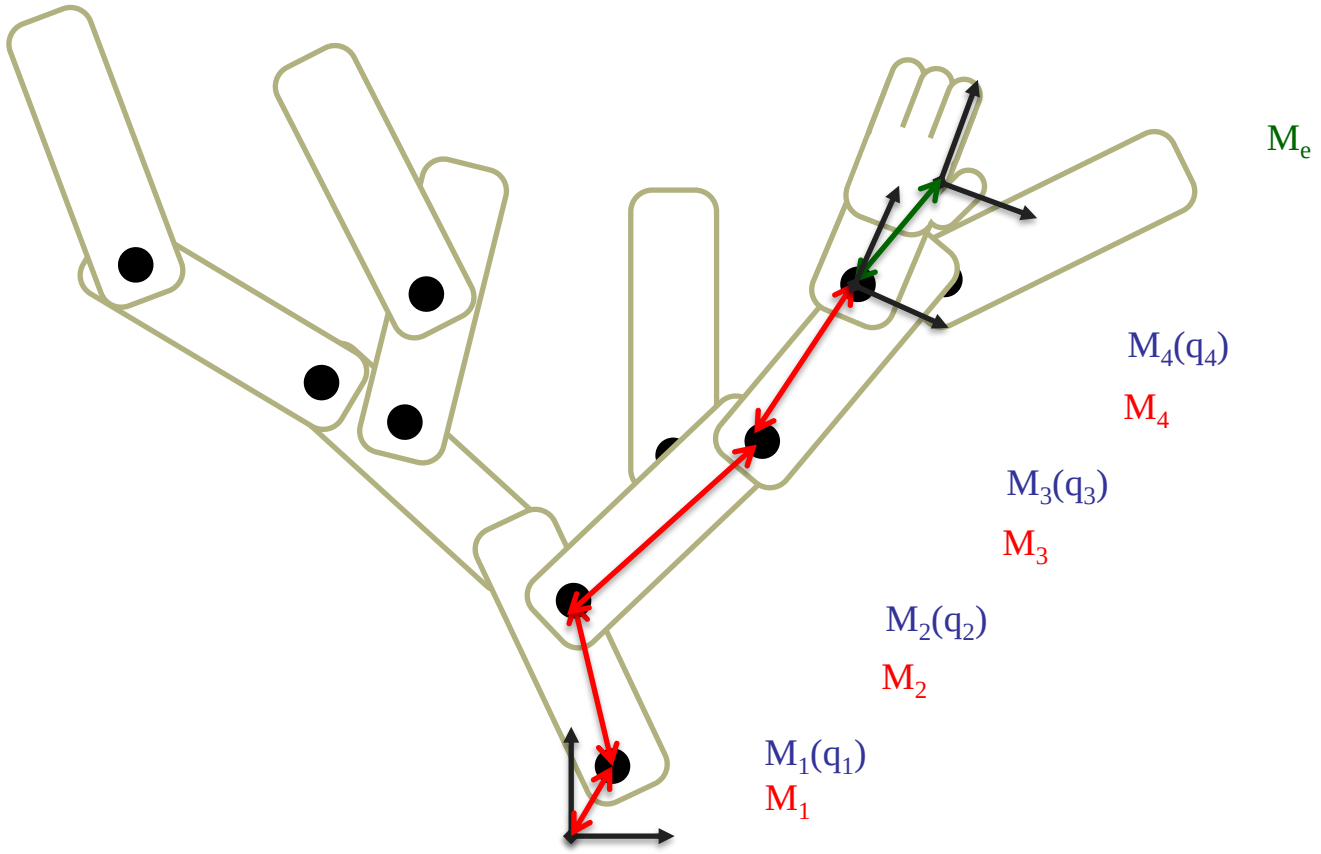
Red placement are functions of **q**



Point of interest often not on effector frame origin

$$ {}^{W}\mathbf{M}_{eff}(\mathbf{q}) = {}^{W}\mathbf{M}_{A}\,{}^{A}\mathbf{M}_{A'}(\mathbf{q})\,{}^{A'}\mathbf{M}_{B}\,{}^{B}\mathbf{M}_{B'}(\mathbf{q})\,{}^{B'}\mathbf{M}_{C}\,{}^{C}\mathbf{M}_{C'}(\mathbf{q})\,{}^{C'}\mathbf{M}_{eff} $$

Can also be a tree



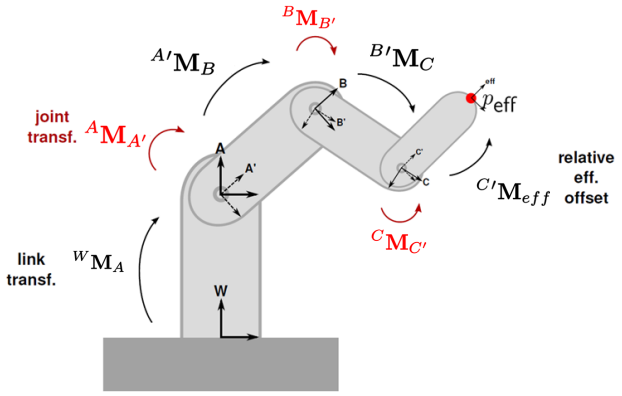
Forward and inverse geometry

- The computation of any map f(q) relates to **forward geometry**

FG: q ⟶ x = f(q) , x in SE(3)

- The inverse problem, called **inverse geometry**, consists, given a desired x*, in finding a q such that f(q) = x*

IG: x* ⟶ q = f^{-1}(x*)



$$^{W}\mathbf{M}_{eff}(\mathbf{q}) = {}^{W}\mathbf{M}_{A}\,{}^{A}\mathbf{M}_{A'}(\mathbf{q})\,{}^{A'}\mathbf{M}_{B}\,{}^{B}\mathbf{M}_{B'}(\mathbf{q})\,{}^{B'}\mathbf{M}_{C}\,{}^{C}\mathbf{M}_{C'}(\mathbf{q})\,{}^{C'}\mathbf{M}_{eff}$$

In the interesting cases, f is almost never invertible => numerical approaches through optimisation:

Search **q** such that f(q) = x* <=> min distance(f(q), x*) => end of tutorial1

Analytical Inverse Geometry + intro to the jacobian matrix

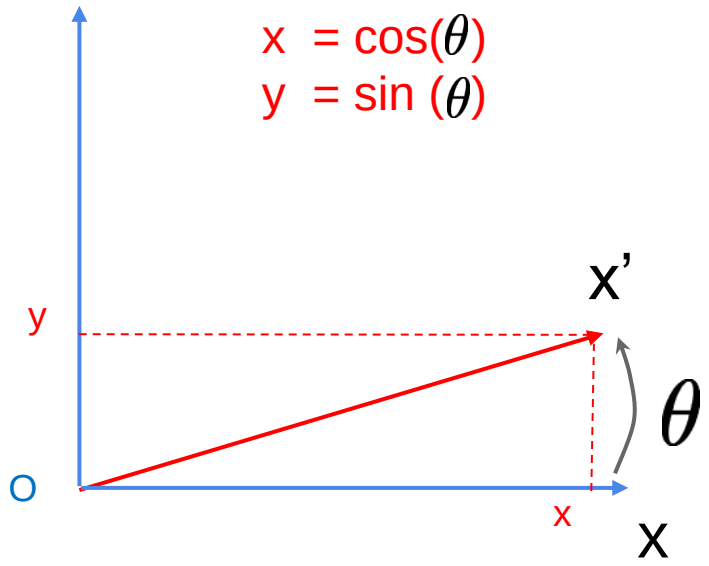
A really simple articulated robot

- 1 Dof, unit norm link (clock)
  - FG:

$$\phi(q) = \phi(\theta) = [cos(\theta), sin(\theta)]^T$$

  - IG:

$$\phi^{-1}([x, y]^T) = atan2(y, x)$$

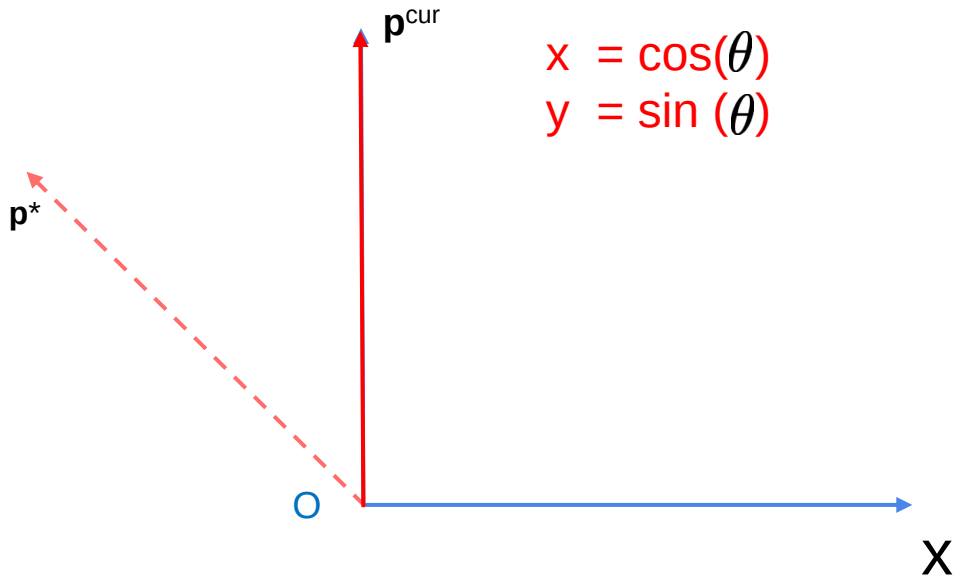


- This problem has an analytical solution. **When solution exists, analytical IG defined if num (dof) <= dim(task)** (necessary condition)

- What if we did not have an analytical solution?

A toy problem

- Current θ= pi/2 ⇔ $\mathbf{p}^{cur}$ = [0, 1]
- Target: $\mathbf{p}^*$ = [$-\sqrt{2}$ $\sqrt{2}$]

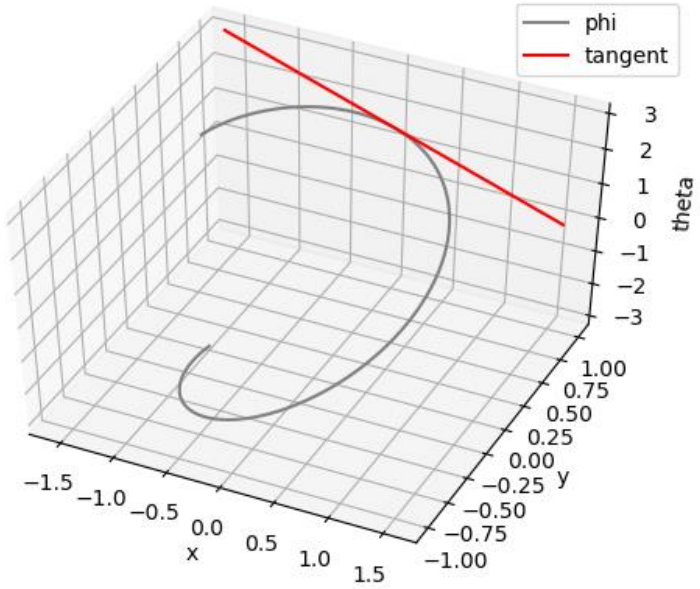


A toy problem

- Current θ= pi/2 ⇔ $\mathbf{p}^{cur}$ = [0, 1]
- Target: **p*** = [$-\sqrt{2}$ $\sqrt{2}$]
- The partial derivatives might give us information

$$\delta p = \frac{\partial}{\partial_q}\phi(q)\delta q$$

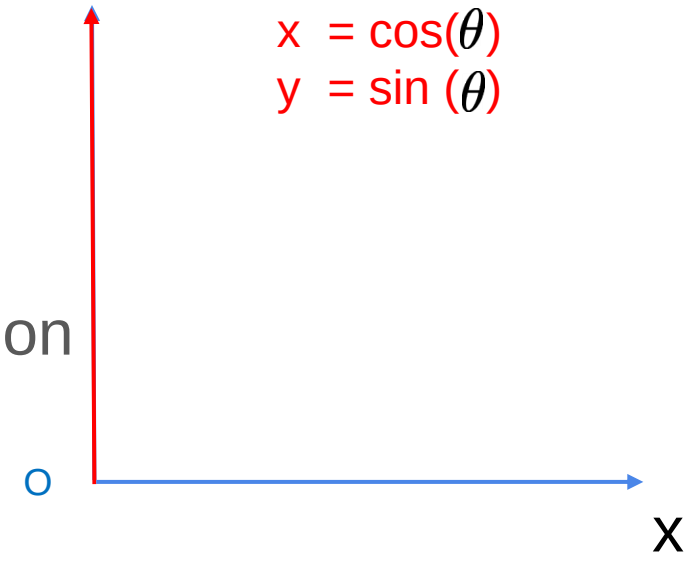
linearisation



Local derivative indicate how an infinitesimal change in configuration affects **p**

A toy problem

- Current θ= pi/2 $\Leftrightarrow$ $\mathbf{p}^{cur}$ = [0, 1]
- Target: $\mathbf{p^*}$ = [$\sqrt{2}$ $\sqrt{2}$]
- The partial derivatives might give us information



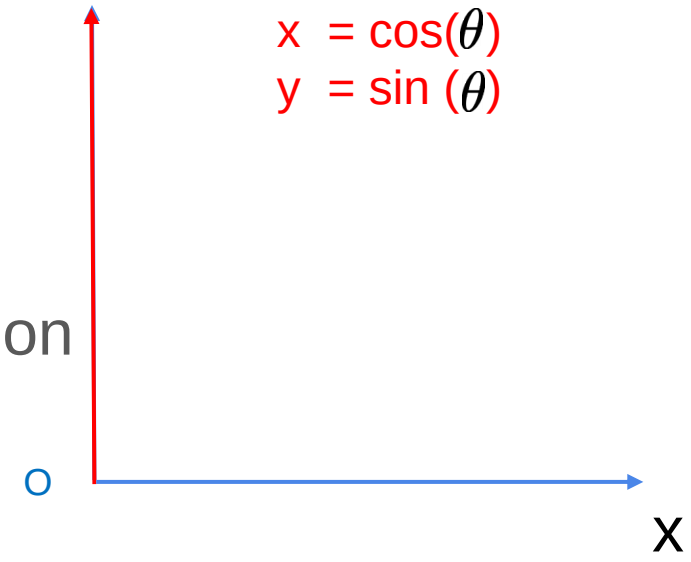
$$\frac{\partial}{\partial_\theta}\phi(\theta) = \begin{bmatrix} \frac{\partial_x}{\partial_\theta} \\ \frac{\partial_y}{\partial_\theta} \end{bmatrix}$$

$$\frac{\partial_x}{\partial_\theta} = -sin(\theta)$$

$$\frac{\partial_y}{\partial_\theta} = cos(\theta)$$

Toy problem

- Current θ= pi/2 $\Leftrightarrow$ $\mathbf{p}^{cur}$ = [0, 1]
- Target: $\mathbf{p^*}$ = [$\sqrt{2}$ $\sqrt{2}$]
- The partial derivatives might give us information



$$\frac{\partial}{\partial_\theta}\phi(\theta) = \begin{bmatrix} \frac{\partial_x}{\partial_\theta} \\ \frac{\partial_y}{\partial_\theta} \end{bmatrix}$$

$$\frac{\partial_x}{\partial_\theta} = -sin(\theta)$$

$$\frac{\partial_y}{\partial_\theta} = cos(\theta)$$

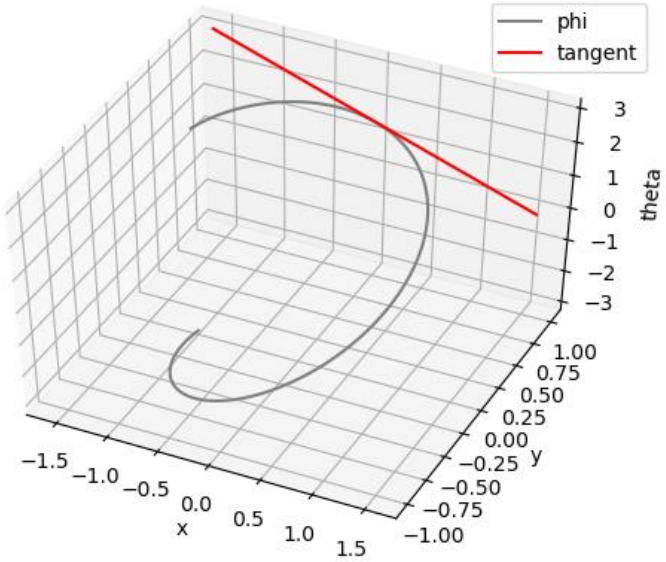
$\Leftrightarrow$

$$\frac{\partial}{\partial_\theta}\phi(\frac{\pi}{2}) = [-1, 0]^T$$

A gradient descent approach

$$\frac{\partial}{\delta_\theta}\phi(\frac{\pi}{2}) = [-1, 0]^T$$

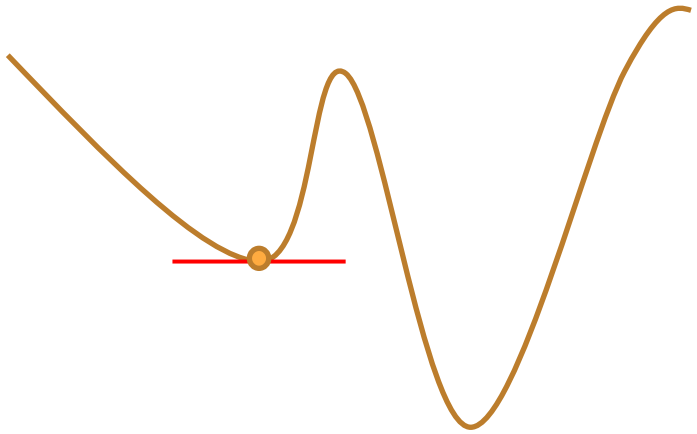
- Slope of the tangent at pi/2
- Thus, local linear approximation of Phi
  - Locally, if θ ↗, x ↘ (not true if we increase θ too much)
  - Nothing to say about y?



- To reach [-$\sqrt{2}$,$\sqrt{2}$], we have an idea of a baby step to make:
  - Increase θ a little. If target is reached, we won, otherwise…
  - …start again: compute the new partial derivatives, slope etc

Gradient descent => find the minimum of a function

- Here distance between current position / target can be used as such function
- Trying to find the **global minima** will not always work…



Exercise: compute partial derivatives for an arm with 2 unit length axes (2 dofs, ie 2 angle values)