# Summary

How it all fits together

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

Last update: January 16, 2020

# Agenda

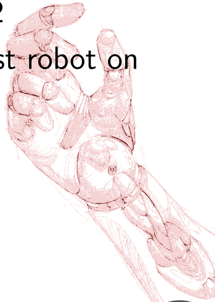
- Just a big summary!

# History and future

# Robots

## A brief history

- 1921: First use of the word *(Rossum's Universal Robots)*
- 1942: Asimov describes the three laws of robotics
- 1956: First commercial robot called Unimate
- 1965: Homogeneous transformations for geometric model description
- 1976: Robots go to space on board Viking 1 and 2
- 1997: PathFinder lands on Mars, becoming the first robot on another planet!
- 2017: Atlas performs its first backflip



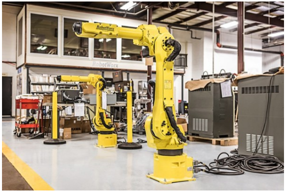
# Robots

## A brief history

- 1921: First use of the word *(Rossum's Universal Robots)*
- 1942: Asimov describes the three laws of robotics
- 1956: First commercial robot called Unimate
- 1965: Homogeneous transformations for geometric model description
- 1976: Robots go to space on board Viking 1 and 2
- 1997: PathFinder lands on Mars, becoming the first robot on another planet!
- 2017: Atlas performs its first backflip
- 2020: Your robot!

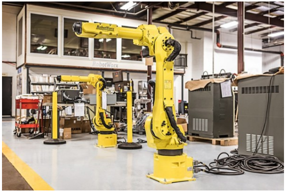
# Course overview

## Scope of this course



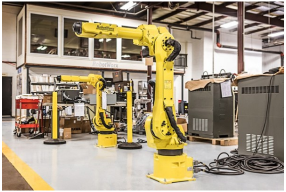
# Course overview

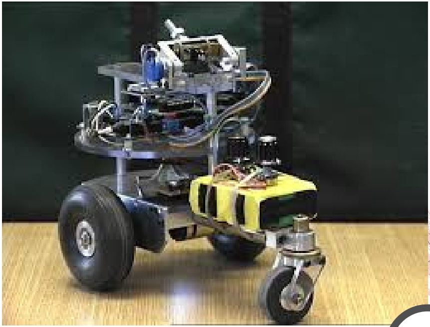
## Scope of this course

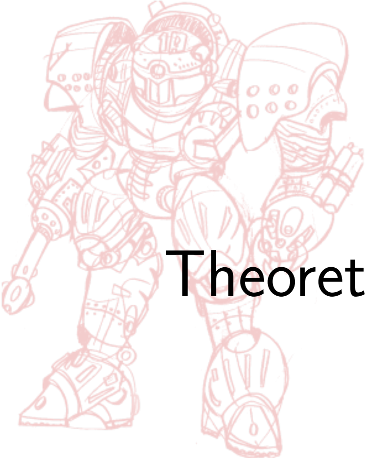


# Course overview

## Scope of this course



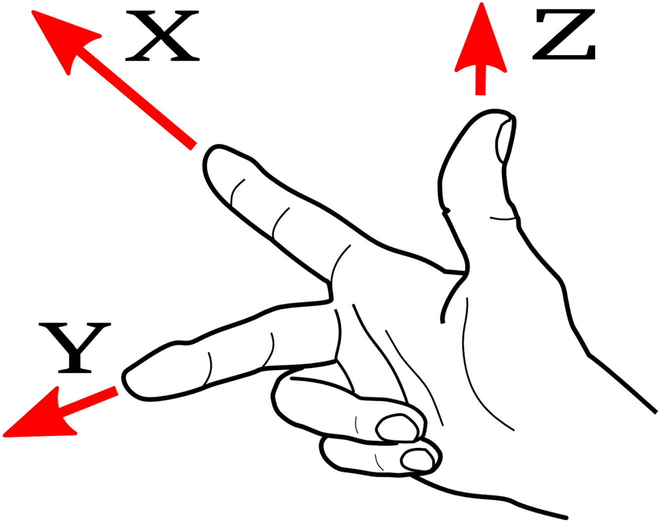




# Theoretical foundations

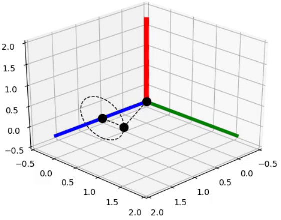
# Coordinate systems

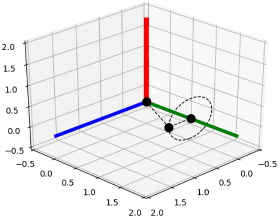
## The rule of the right hand

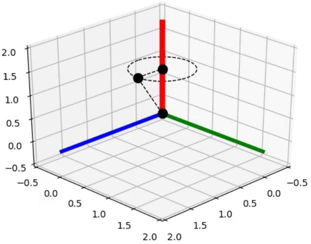


# Transformations

## Let's do it in 3D




# Transformations

## Homogenious translations

$$Trans(X,a) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans(Y,b) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans(Z,c) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Transformations

## Homogenious rotations

$$Rot(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

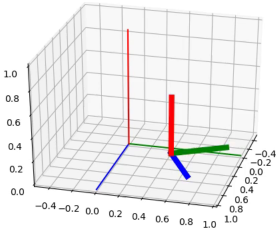
$$Rot(Y,\phi) = \begin{bmatrix} cos\phi & 0 & sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -sin\phi & 0 & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(Z,\omega) = \begin{bmatrix} cos\omega & -sin\omega & 0 & 0 \\ sin\omega & cos\omega & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

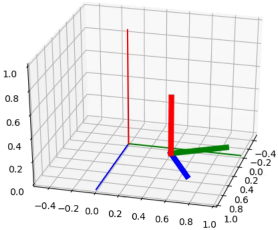
# Transformations

## Left and right multiplication

$$V_O' = Rot(X,\theta) * V_O$$




$$V_O' = V_O * Rot(X,\theta)$$

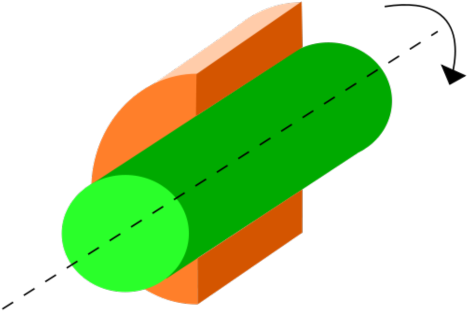


# Robot description

# Joints

## Revolute joints

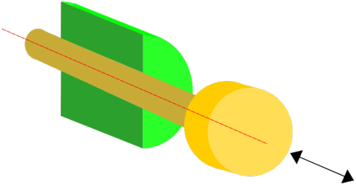
A revolute joint is a joint that allows motion that changes the orientation of a segment by rotating around a fixed axis. They can add one degree of freedom to a robot.



# Joints

## Prismatic joints

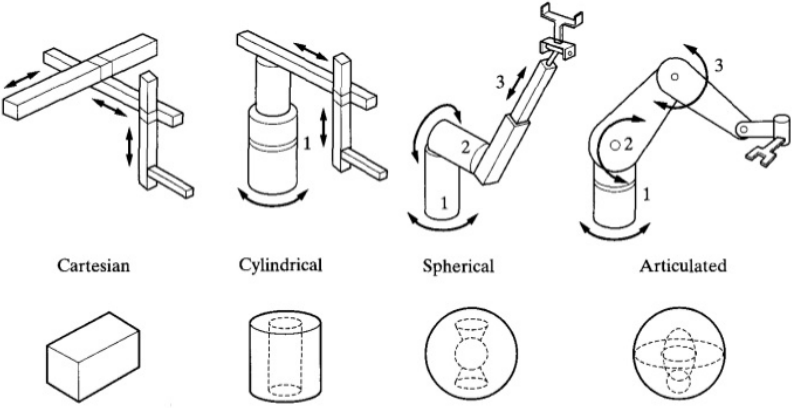
A prismatic joint is a joint that allows motion that changes the position of a segment by translating along a axis. They can add one degree of freedom to a robot.

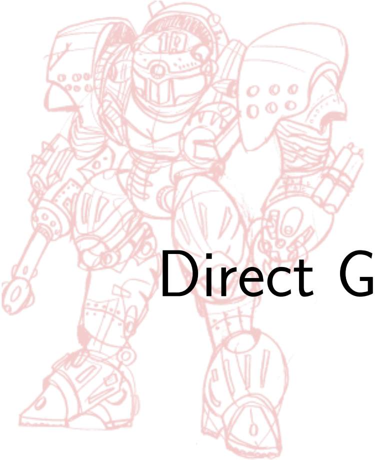



# Robotic modeling

## Work envelope

The combination of links and joints defines the degrees of freedom to a robot. Besides that, it also defines the work envelope of the robot.





# Direct Geometric Model

# Direct geometric model

## Dynamic calculation

$$R_0^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{1,2,3} & -s_{1,2,3} & l_3 c_{1,2,3} + l_2 c_{1,2} + l_1 c_1 \\ 0 & s_{1,2,3} & c_{1,2,3} & l_3 s_{1,2,3} + l_2 s_{1,2} + l_1 s_1 + 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Direct geometric model

The DGM is a transformation matrix, a function of the joint positions and link lengths. If we know these variables, we can calculate the position and orientation of the end effector (or any other point).

# Direct geometric model

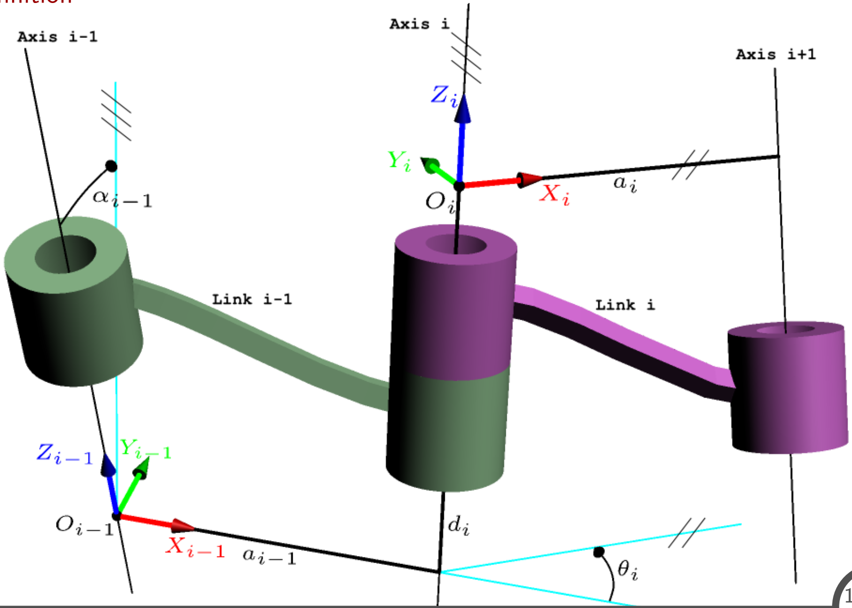
## Definition

### In simple words

How to I calculate the pose of the end-effector if I know the joint angles?

# Denavit-Hartenberg convention

## Definition



# Denavit-Hartenberg convention

## Calculating the parameters

To calculate the 4 parameters, we first construct coordinate frames (CF) on for each joint using the following procedure:

- We align the z-axis of each CF with the axis of rotation/translation of each joint

# Denavit-Hartenberg convention

## Calculating the parameters

To calculate the 4 parameters, we first construct coordinate frames (CF) on for each joint using the following procedure:

- We align the z-axis of each CF with the axis of rotation/translation of each joint
- We identify the common perpendicular between subsequent z-axes

# Denavit-Hartenberg convention

## Calculating the parameters

To calculate the 4 parameters, we first construct coordinate frames (CF) on for each joint using the following procedure:

- We align the z-axis of each CF with the axis of rotation/translation of each joint
- We identify the common perpendicular between subsequent z-axes
- We align $X_i$ with the common perpendiculars between $Z_i$ and $Z_{i+1}$

# Denavit-Hartenberg convention

## Calculating the parameters

To calculate the 4 parameters, we first construct coordinate frames (CF) on for each joint using the following procedure:

- We align the z-axis of each CF with the axis of rotation/translation of each joint
- We identify the common perpendicular between subsequent z-axes
- We align $X_i$ with the common perpendiculars between $Z_i$ and $Z_{i+1}$
- The positive direction for $X_i$ is from $Z_i$ to $Z_{i+1}$

# Denavit-Hartenberg convention

## Calculating the parameters

Once we have constructed the CFs, we identify the four parameters as following:

- $r_i$: distance between axes $Z_i$ and $Z_{i+1}$, measured on axis $X_i$

# Denavit-Hartenberg convention

## Calculating the parameters

Once we have constructed the CFs, we identify the four parameters as following:

- $r_i$: distance between axes $Z_i$ and $Z_{i+1}$, measured on axis $X_i$
- $\alpha_i$: angle between axes $Z_i$ and $Z_{i+1}$, measured around axis $X_i$

# Denavit-Hartenberg convention

## Calculating the parameters

Once we have constructed the CFs, we identify the four parameters as following:

- $r_i$: distance between axes $Z_i$ and $Z_{i+1}$, measured on axis $X_i$
- $\alpha_i$: angle between axes $Z_i$ and $Z_{i+1}$, measured around axis $X_i$
- $d_i$: distance between axes $X_i$ and $X_{i+1}$, measured on axis $Z_{i+1}$

# Denavit-Hartenberg convention

## Calculating the parameters

Once we have constructed the CFs, we identify the four parameters as following:

- $r_i$: distance between axes $Z_i$ and $Z_{i+1}$, measured on axis $X_i$
- $\alpha_i$: angle between axes $Z_i$ and $Z_{i+1}$, measured around axis $X_i$
- $d_i$: distance between axes $X_i$ and $X_{i+1}$, measured on axis $Z_{i+1}$
- $\theta_i$: angle between axes $X_i$ and $X_{i+1}$, measured around axis $Z_{i+1}$

# Denavit-Hartenberg convention

## Definition

### DH Parameters

We define four transformation matrices for the transformation from joint i to joint i+1. Two are rotation and two are translation matrices.

$$R_i^{i+1} = [X_i] * [Z_i]$$

# Denavit-Hartenberg convention

## Definition

### DH Parameters

We define four transformation matrices for the transformation from joint i to joint i+1. Two are rotation and two are translation matrices.

$$R_i^{i+1} = [X_i] * [Z_i]$$

where

$$[X_i] = R(x, \alpha_i) * T(x, r_i)$$

# Denavit-Hartenberg convention

## Definition

### DH Parameters

We define four transformation matrices for the transformation from joint i to joint i+1. Two are rotation and two are translation matrices.

$$R_i^{i+1} = [X_i] * [Z_i]$$

where

$$[X_i] = R(x, \alpha_i) * T(x, r_i)$$

and

$$[Z_i] = R(z, \theta_i) * T(z, d_i)$$

# Denavit-Hartenberg convention

## Definition

### DH Parameters

We define four transformation matrices for the transformation from joint i to joint i+1. Two are rotation and two are translation matrices.

$$R_i^{i+1} = [X_i] * [Z_i]$$

where

$$[X_i] = R(x, \alpha_i) * T(x, r_i)$$

and

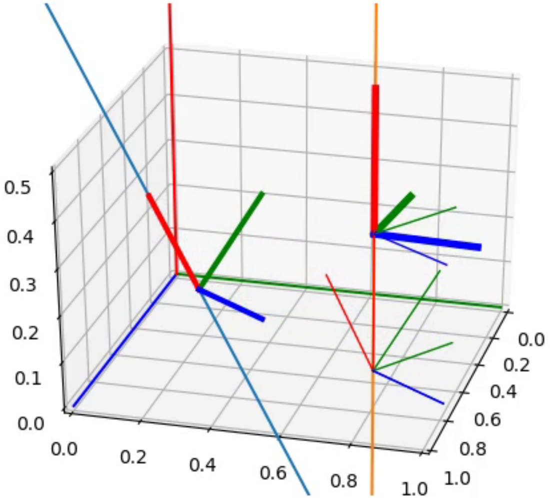
$$[Z_i] = R(z, \theta_i) * T(z, d_i)$$

therefore

$$R_i^{i+1} = R(x, \alpha_i) * T(x, r_i) * R(z, \theta_i) * T(z, d_i)$$

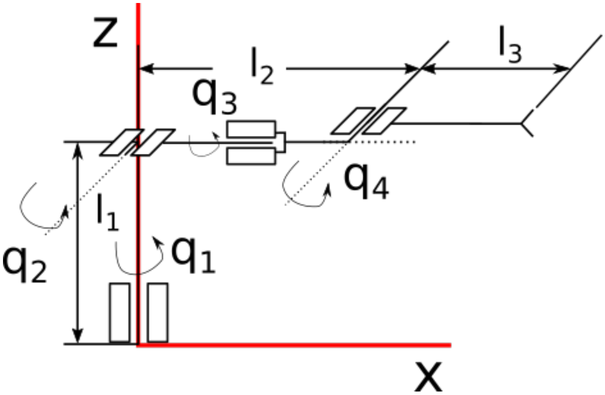
# Denavit-Hartenberg convention

## Visualising the angles



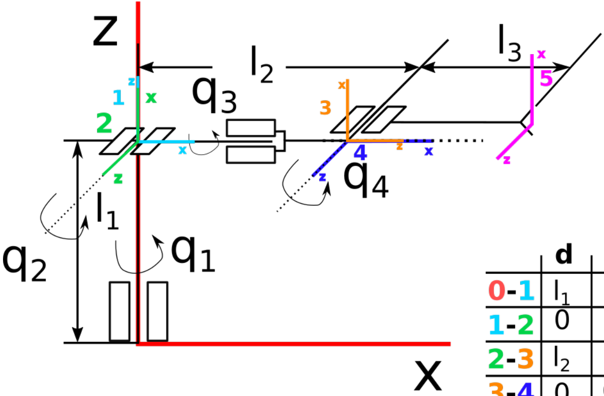
# Denavit-Hartenberg convention

## Examples



# Denavit-Hartenberg convention

## Examples




| | d | r | α | θ |
|---|---|---|---|---|
| 0-1 | $l_1$ | 0 | 0 | $q_1$ |
| 1-2 | 0 | 0 | $\pi/2$ | $q_2$ |
| 2-3 | $l_2$ | 0 | $\pi/2$ | $q_3$ |
| 3-4 | 0 | 0 | $-\pi/2$ | $q_4$ |
| 4-5 | 0 | $l_3$ | 0 | $\pi/2$ |

# Denavit-Hartenberg convention

## Parameters $\theta$ and $d$

**Revolute joints**

If $i$ is a revolute joint, parameter $\theta_i$ is always variable and relates to joint variable $q_i$

# Denavit-Hartenberg convention

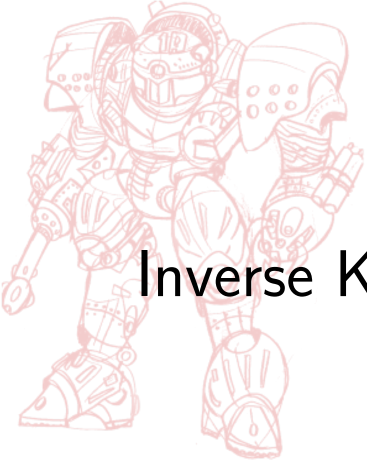
## Parameters $\theta$ and $d$

### Revolute joints

If $i$ is a revolute joint, parameter $\theta_i$ is always variable and relates to joint variable $q_i$

### Prismatic joints

If $i$ is a prismatic joint, parameter $d_i$ is always variable and relates to joint variable $q_i$



# Inverse Kinematics Model

# Inverse kinematics

## UpTown!

# Inverse and Direct models

## What is the difference?

### Direct geometric model

I want to know where will my end-effector be, if I position each joint to a specific position

# Inverse and Direct models

## What is the difference?

### Direct geometric model

I want to know where will my end-effector be, if I position each joint to a specific position

### Inverse geometric model

I want to know what should the joint coordinates be in order for my end-effector to achieve a specific position

# Inverse kinematics model

## Derivation

$$\begin{bmatrix} c_{1,2} & -s_{1,2} & 0 & l_2c_{1,2}+l_1c_1 \\ s_{1,2} & c_{1,2} & 0 & l_2s_{1,2}+l_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} X_X & Y_X & Z_X & P_x \\ X_Y & Y_Y & Z_Y & P_x \\ X_Z & Y_Z & Z_Z & P_x \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Inverse kinematics model

## Derivation

$$\begin{bmatrix} c_{1,2} & -s_{1,2} & 0 & l_2c_{1,2}+l_1c_1 \\ s_{1,2} & c_{1,2} & 0 & l_2s_{1,2}+l_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} X_X & Y_X & Z_X & P_x \\ X_Y & Y_Y & Z_Y & P_x \\ X_Z & Y_Z & Z_Z & P_x \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$cos(q_1+q_2) = X_x = Y_y$$

$$sin(q_1+q_2) = X_y = -Y_x$$

$$l_2cos(q_1+q_2) + l_1cosq_1 = P_x$$

$$l_2sin(q_1+q_2) + l_1sinq_1 = P_y$$

$$0 = P_z$$

We are looking for this

$$q_1 = f(P_x, P_y)$$

$$q_2 = g(P_x, P_y)$$

# Inverse kinematics model

## Analytical solutions

The idea is to try to find an equation for each joint variable $q_n$ that only depends on the pose or on other joint variables that have already been expressed in terms of the pose.

# Inverse kinematics model

## Analytical solutions

The idea is to try to find an equation for each joint variable $q_n$ that only depends on the pose or on other joint variables that have already been expressed in terms of the pose.

- We equate the DGM with the general homogeneous matrix

# Inverse kinematics model

## Analytical solutions

The idea is to try to find an equation for each joint variable $q_n$ that only depends on the pose or on other joint variables that have already been expressed in terms of the pose.

- We equate the DGM with the general homogeneous matrix
- We identify joint variables that can be isolated

# Inverse kinematics model

## Analytical solutions

The idea is to try to find an equation for each joint variable $q_n$ that only depends on the pose or on other joint variables that have already been expressed in terms of the pose.

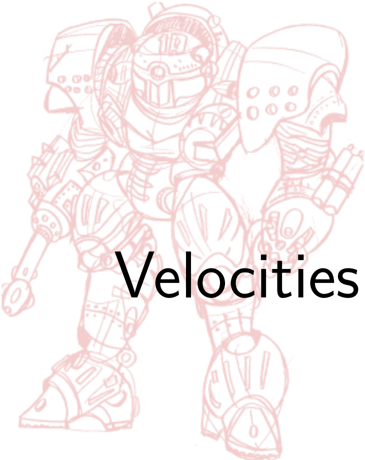
- We equate the DGM with the general homogeneous matrix
- We identify joint variables that can be isolated
- We identify pair of joint variables that can be simplified by division

# Inverse kinematics model

## Analytical solutions

The idea is to try to find an equation for each joint variable $q_n$ that only depends on the pose or on other joint variables that have already been expressed in terms of the pose.

- We equate the DGM with the general homogeneous matrix
- We identify joint variables that can be isolated
- We identify pair of joint variables that can be simplified by division
- We identify pair of joint variables that can be simplified by trigonometry



# Velocities and the Jacobian

# Robot velocity

## Background

We define a matrix called the 'Jacobian' that shows us how can we calculate the end-effector velocity if we know the join velocities

$$\xi = J\dot{q}$$

# Robot velocity

## Background

We define a matrix called the 'Jacobian' that shows us how can we calculate the end-effector velocity if we know the join velocities

$$\xi = J\dot{q}$$

What is the size of $\xi, \dot{q}$, and $J$?

# Defining the Jacobian

## Combining angular and linear velocities

We can calculate each column of the Jacobian matrix individually. Each column represents one joint. If joint $i$ is revolute, then:

$$J_i = \begin{bmatrix} z_i \times (o_n - o_i) \\ z_i \end{bmatrix}$$

If joint $i$ is prismatic, then:

$$J_i = \begin{bmatrix} z_i \\ 0 \end{bmatrix}$$

# Jacobian

## Inverse velocity

We now have a method to define the end-effector velocity (angular and linear) based on the joint velocities

$$\xi = J\dot{q}$$

# Jacobian

## Inverse velocity

We now have a method to define the end-effector velocity (angular and linear) based on the joint velocities

$$\xi = J\dot{q}$$

How do we do the opposite (i.e. define the joint velocities for specific end-effector velocity)?

# Jacobian

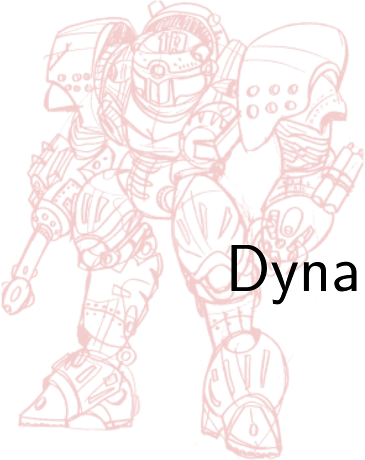
## Inverse velocity

We now have a method to define the end-effector velocity (angular and linear) based on the joint velocities

$$\xi = J\dot{q}$$

How do we do the opposite (i.e. define the joint velocities for specific end-effector velocity)?

$$J^{-1}\xi = \dot{q}$$



# Dynamic Modeling

# Dynamic modeling

## What is it all about?

**Kinematics**:

**Dynamics (Kinetics)**:

# Dynamic modeling

## What is it all about?

**Kinematics**: description of motion of bodies or system of bodies

**Dynamics (Kinetics)**:

# Dynamic modeling

## What is it all about?

**Kinematics**: description of motion of bodies or system of bodies

**Dynamics (Kinetics)**: desctiption of the causes resulting in those motions (i.e. forces and torques)

# Lagrangian mechanics

## A more sophisticated formulation of mechanics

Lagrange defined a basic quatity for any system of bodies as the difference between its kinetic and potential energy.

$$L = K - P$$

We call this quantity the Lagrangian of the system.

# Lagrangian mechanics

## A more sophisticated formulation of mechanics

Using this quantity, we can describe the evolution of any system of bodies under the influence of a set of external forces using the following equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau$$

Giving the same results as Newton!

# Definitions

## Potential and Kinetic energy

### Potential energy

The energy possessed by an object because of its position relative to other objects, stresses within itself, its electric charge, or other factors.

### Kinetic energy

The energy of an object that it possesses due to its motion.

# Definitions

## Potential energy

The most common source of potential energy in robotics is the gravitational field of Earth.

# Definitions

## Potential energy

The most common source of potential energy in robotics is the gravitational field of Earth.

$$P = mgh$$

# Definitions

## Potential energy

The most common source of potential energy in robotics is the gravitational field of Earth.

$$P = mgh$$

Where $m$ is the mass of the object, $g$ is the gravitational constant, and $h$ is the height of the object.

# Definitions

## Potential energy

The most common source of potential energy in robotics is the gravitational field of Earth.

$$P = mgh$$

Where $m$ is the mass of the object, $g$ is the gravitational constant, and $h$ is the height of the object.

But height from where? What is the reference?

# Definitions

## Kinetic energy

### Total kinetic energy

The total kinetic energy of an object is the sum of its linear and angular kinetic energy.

$$K_{total} = K_{linear} + K_{angular} = \frac{1}{2}(mu^2 + I\omega^2)$$

# Definitions

## Moment of inertia

### Moment of inertia

A tensor that determines the torque needed for a desired angular acceleration about a rotational axis.

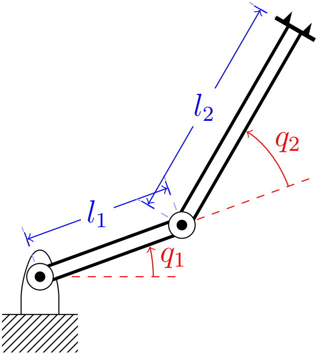
$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

The moment of inertia is the equivalent of mass, but for rotational movements.

# Lagrangian of a robot

## Dynamic energy

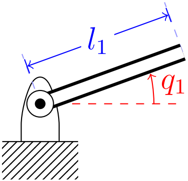
We need to calculate the total dynamic energy of the system.




The total dynamic energy is the sum of the dynamic energies of each segment. What is the dynamic energy of each segment?

# Lagrangian of a robot

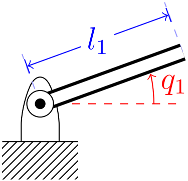
## Dynamic energy




We consider the mass of the link to be concentrated at its center of mass.

# Lagrangian of a robot

## Dynamic energy




We consider the mass of the link to be concentrated at its center of mass. Therefore:

$$P_1 = m_1 g \frac{l_1}{2} sin q_1$$

# Lagrangian of a robot

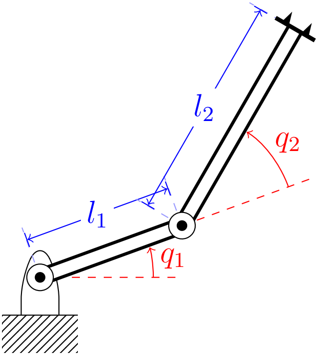
## Kinetic energy

Let's start with the linear kinetic energy first.

# Lagrangian of a robot

## Kinetic energy

Let's start with the linear kinetic energy first.

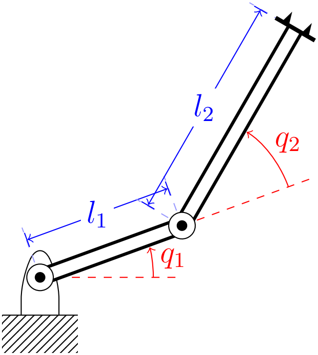



$$K_{lin} = \frac{1}{2} m u^2$$

# Lagrangian of a robot

## Kinetic energy

Let's start with the linear kinetic energy first.



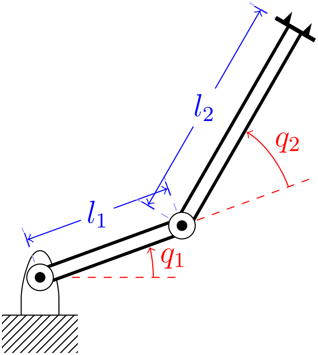

$$K_{lin} = \frac{1}{2} m u^2$$

$$u = J_u \dot{q} \mapsto u^2 = \dot{q}^T J_u^T J_u \dot{q}$$

# Lagrangian of a robot

## Kinetic energy

Let's start with the linear kinetic energy first.




$$K_{lin} = \frac{1}{2} m u^2$$

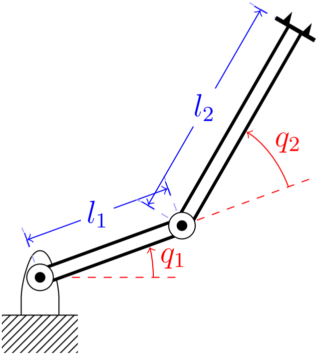
$$u = J_u \dot{q} \mapsto u^2 = \dot{q}^T J_u^T J_u \dot{q}$$

$$K_{lin} = \frac{1}{2} m \dot{q}^T J_u^T J_u \dot{q}$$

# Lagrangian of a robot

## Kinetic energy

Let's start with the linear kinetic energy first.




$$K_{lin} = \frac{1}{2}mu^2$$

$$u = J_u\dot{q} \mapsto u^2 = \dot{q}^T J_u^T J_u \dot{q}$$

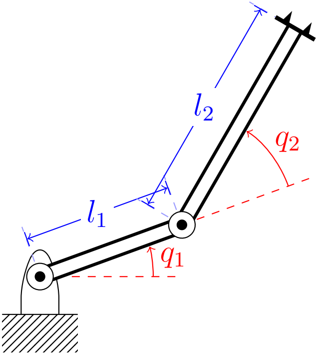
$$K_{lin} = \frac{1}{2}m\dot{q}^T J_u^T J_u \dot{q}$$

$$K_{lin,total} = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n}[m_i J_{ui}^T J_{ui}]\dot{q}$$

# Lagrangian of a robot

## Kinetic energy

... and then the angular kinetic energy:

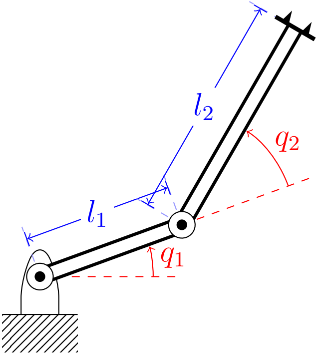



$$K_{lin} = \frac{1}{2} I\omega^2 = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} [J_{\omega i}^T R_i I_i R_i^T J_{\omega i}]\dot{q}$$

# Lagrangian of a robot

## Kinetic energy

Therefore, the total Kinetic energy of the robot is:




$$K = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} \left[ m_i J_{vi}^T J_{vi} + J_{\omega i} R_i I_i R_i^T J_{\omega i} \right] \dot{q}$$

# Lagrangian of a robot

## Condensed form

Eventually, after we do the derivation of the Lagrancian, we can write the dynamic model of the robot in this general form:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

# Lagrangian of a robot

## Condensed form

Eventually, after we do the derivation of the Lagrancian, we can write the dynamic model of the robot in this general form:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

The matrix $D$, contains information about the **inertia** of the system, therefore contains all the masses and moments of inertia.

# Lagrangian of a robot

## Condensed form

Eventually, after we do the derivation of the Lagrancian, we can write the dynamic model of the robot in this general form:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

The matrix $D$, contains information about the **inertia** of the system, therefore contains all the masses and moments of inertia.

The matrix $C$ has elements related to the **cetrifugal** and **Coriolis** terms

# Lagrangian of a robot

## Condensed form

Eventually, after we do the derivation of the Lagrancian, we can write the dynamic model of the robot in this general form:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

The matrix $D$, contains information about the **inertia** of the system, therefore contains all the masses and moments of inertia.

The matrix $C$ has elements related to the **cetrifugal** and **Coriolis** terms

Finally, the term $g$ contains the dependence of the potential energy from the position of the robot.

# Lagrangian of a robot

## Condensed form

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

We can show that:

# Lagrangian of a robot

## Condensed form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

We can show that:

$$D(q) = \sum_{i=1}^{n} \left[ m_i J_{vi}^T J_{vi} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i} \right]$$

# Lagrangian of a robot

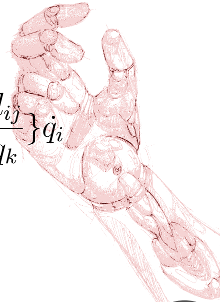
## Condensed form

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

We can show that:

$$D(q) = \sum_{i=1}^{n} \left[ m_i J_{vi}^T J_{vi} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i} \right]$$

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i = \sum_{i=1}^{n} \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i$$



# Lagrangian of a robot

## Condensed form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

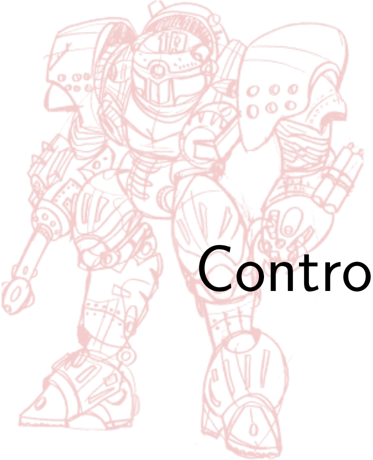
We can show that:

$$D(q) = \sum_{i=1}^{n} \left[ m_i J_{vi}^T J_{vi} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i} \right]$$

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i = \sum_{i=1}^{n} \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \dot{q}_i$$

and:

$$G(q) = \frac{\partial P}{\partial q}$$

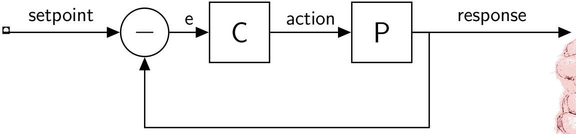


# Controlling the robot

# Control theory

## Feedback loops

The robot is a process, and if we want to accomplish some tasks, we need to be able to control its various aspects.

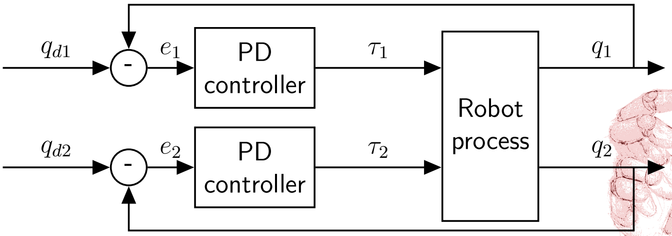



*A simple process with feedback*

# Robotic controllers

## Independent joint control

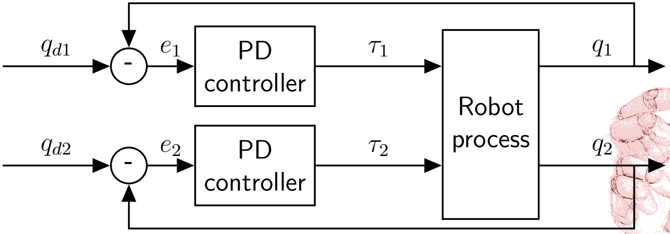
With this control strategy, we control each joint individually.



# Robotic controllers

## Independent joint control

With this control strategy, we control each joint individually.

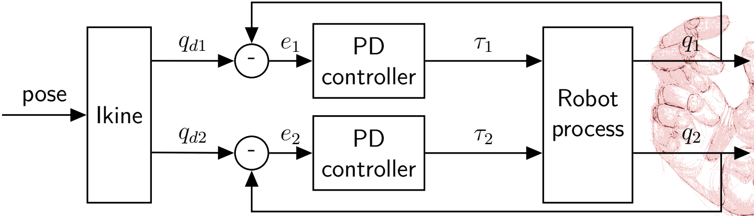



How can we control the end-effector pose?

# Robotic controllers

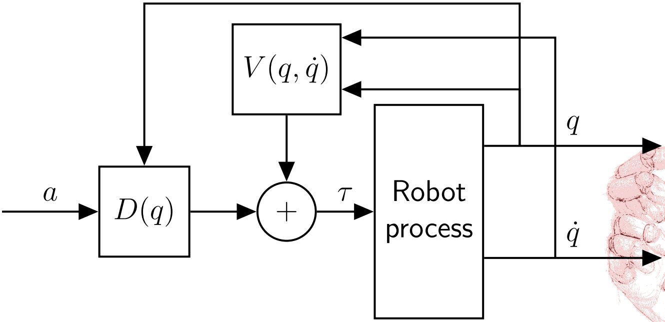
## Independent joint control

If we want to controle the pose of the end-effector, then we need to solve the inverse kinematics to define the joint coordinates for the specific pose.



# Control theory

## System linearization



# Other types of robots

# Quadrotor drones

## What is a quadrotor?

# Quadrotor drones

## What is a quadrotor?

Why four rotors?

# Quadrotor

## Achieving flight in X configuration

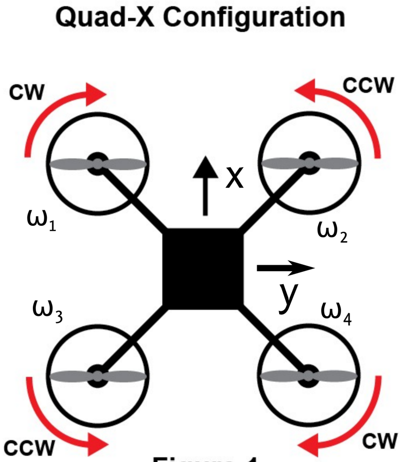
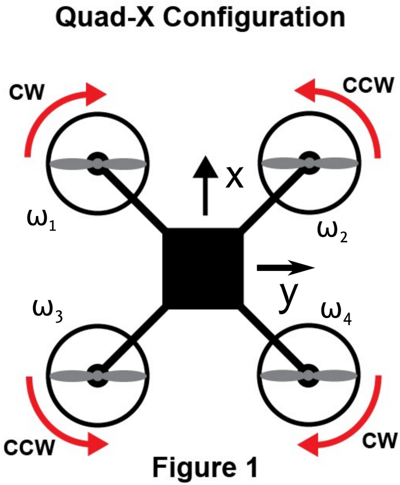


Figure 1

# Quadrotor

## Achieving flight in X configuration




- Translation on z:

  $\omega_1 = \omega_2 = \omega_3 = \omega_4$

# Quadrotor

## Achieving flight in X configuration

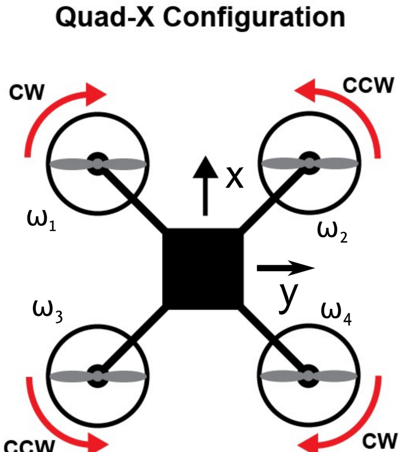
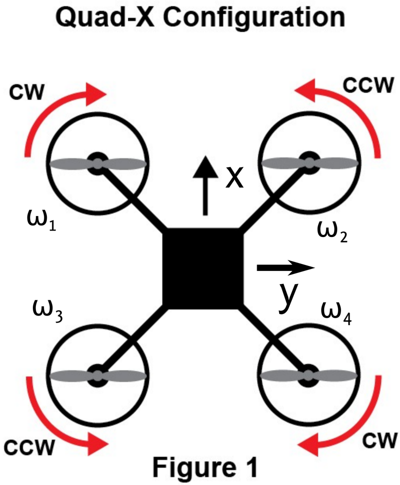


Figure 1

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$

# Quadrotor

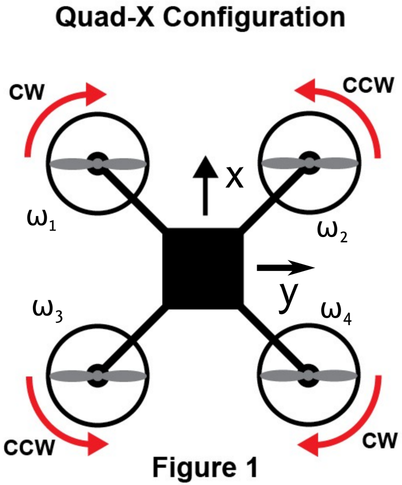
## Achieving flight in X configuration




- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$

# Quadrotor

## Achieving flight in X configuration




- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z:

# Quadrotor

## Achieving flight in X configuration

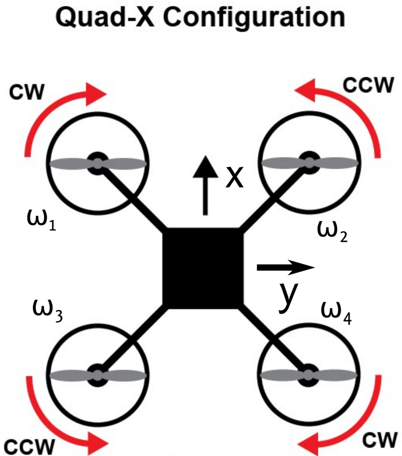


Figure 1

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$

# Quadrotor

## Achieving flight in X configuration

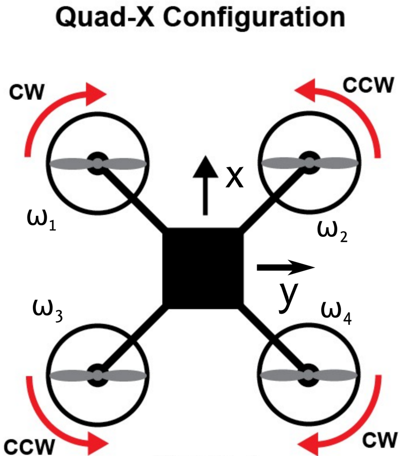



Figure 1

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y

# Quadrotor

## Achieving flight in X configuration

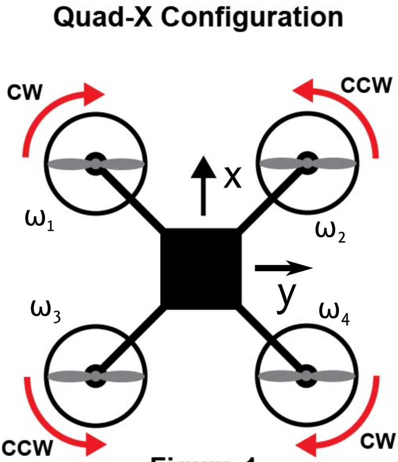


Figure 1

- Translation on z: $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x: $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y: $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z: $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y
- Translation on y: Rotation around x

# Quadrotor

## Achieving flight in X configuration

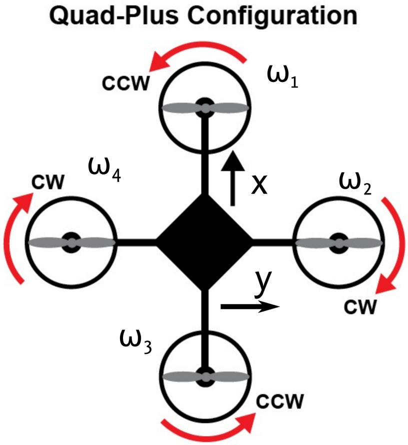


Figure 2

# Quadrotor

## Achieving flight in X configuration

- Translation on z:

  $\omega_1 = \omega_2 = \omega_3 = \omega_4$

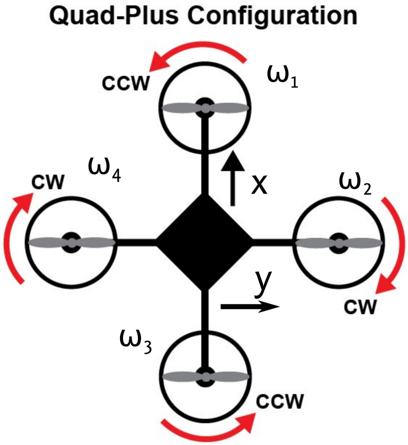


Figure 2

# Quadrotor

## Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$

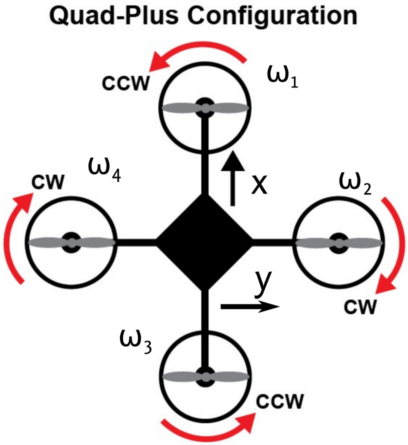


Figure 2

# Quadrotor

## Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$

Quad-Plus Configuration

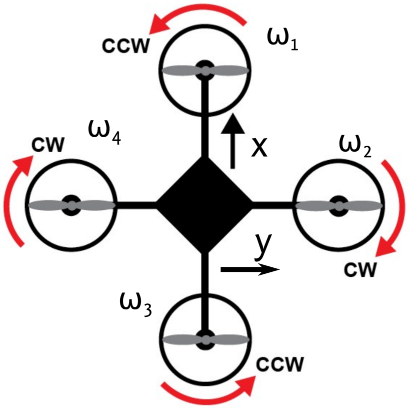



Figure 2

# Quadrotor

## Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$

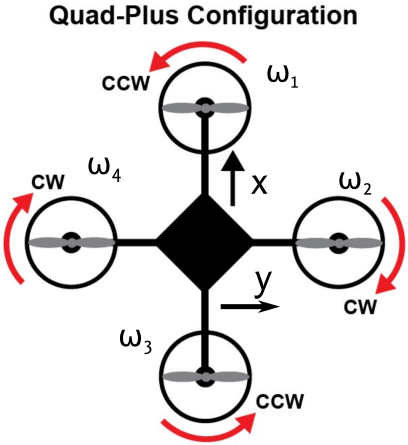


Figure 2

# Quadrotor

## Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y

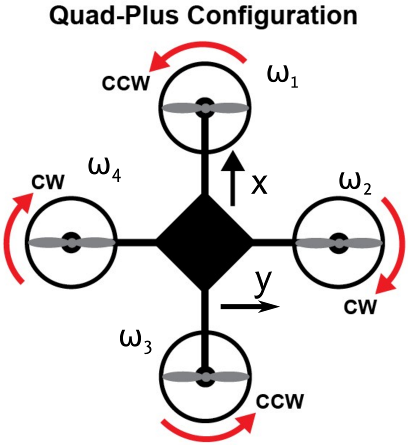


Figure 2

# Quadrotor

## Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y
- Translation on y: Rotation around x

Quad-Plus Configuration

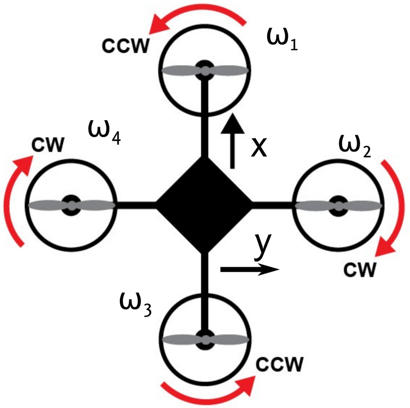
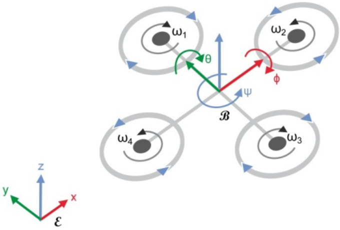



Figure 2

# Dynamic modeling

## The Lagrangian

Remember:



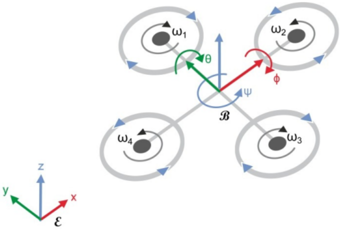

- $x, y, z$: translation along x, y, z axes of the fixed frame
- $\phi, \theta, \psi$: rotation around x, y, z axes of the fixed frame

$S = [\xi, \eta]^T$, where: $\xi = [x, y, z]^T$ and $\eta = [\phi, \theta, \psi]^T$

# Dynamic modeling

## The Lagrangian

Remember:




- $x, y, z$: translation along x, y, z axes of the fixed frame
- $\phi, \theta, \psi$: rotation around x, y, z axes of the fixed frame

$S = [\xi, \eta]^T$, where: $\xi = [x, y, z]^T$ and $\eta = [\phi, \theta, \psi]^T$
Therefore:

$$L(S, \dot{S}) = K_{lin} + K_{rot} - P = \frac{1}{2}\left(m\dot{\xi}^T\dot{\xi} + \dot{\eta}^T J(\eta)\dot{\eta}\right) - mgz$$

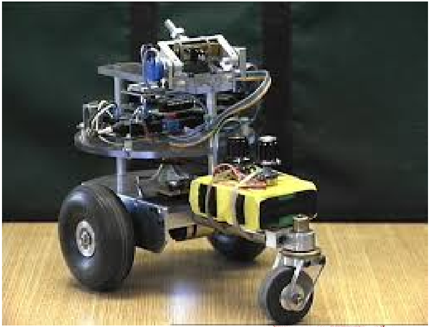
# Dynamic modeling

## Putting it all together

Therefore, if we plug our forces in the dynamic model equation, we have:

$$
\begin{gathered}
\ddot{x} = \left[c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi)\right]\frac{U_{coll}}{m} \\
\ddot{y} = \left[c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi)\right]\frac{U_{coll}}{m} \\
\ddot{z} = -g + c(\phi)c(\theta)\frac{U_{coll}}{m} \\
\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = J^{-1}(\eta)\left(\begin{bmatrix} U_\phi \\ U_\theta \\ U_\psi \end{bmatrix} - C(\eta, \dot{\eta})\eta\right)
\end{gathered}
$$

# Other types of robots



# Wheeled robots

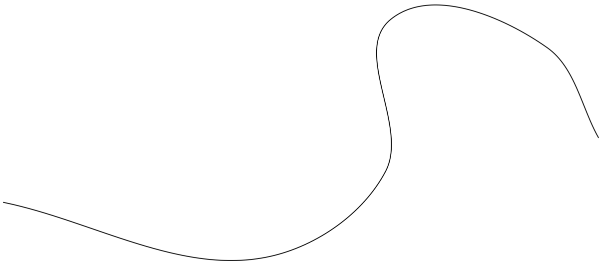
## Instantaneous center of rotation

Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.

# Wheeled robots

## Instantaneous center of rotation

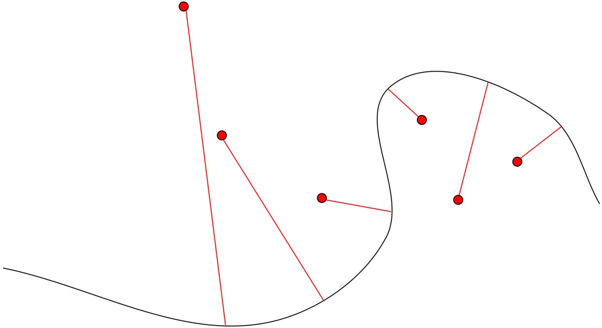
Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



# Wheeled robots

## Instantaneous center of rotation

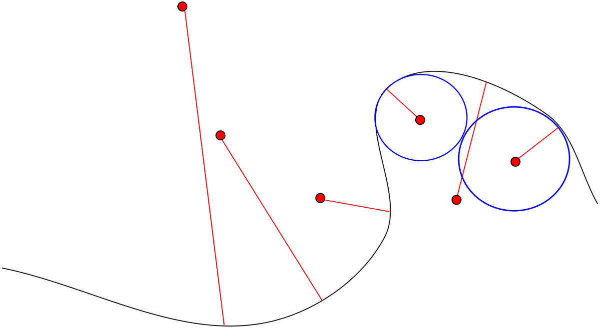
Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



# Wheeled robots

## Instantaneous center of rotation

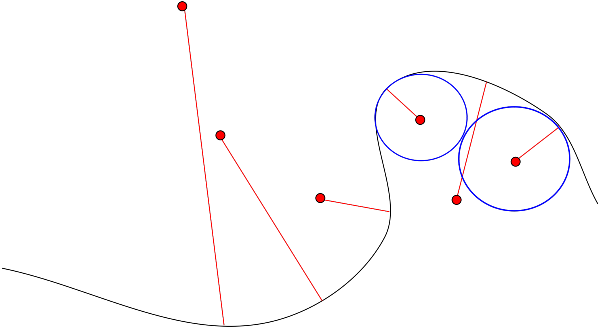
Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.



# Wheeled robots

## Instantaneous center of rotation

Every motion can be modeled as a rotation around a point. For a circular motion, this point is fixed, but for a more complex motion it is constantly moving.

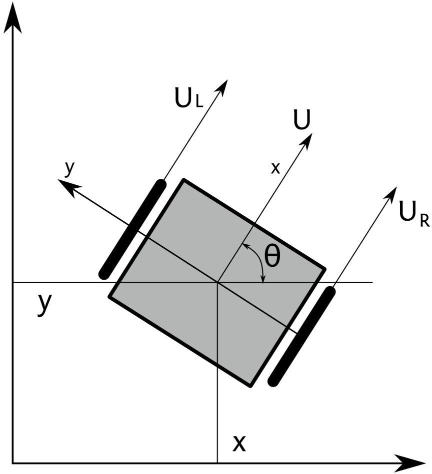


Where is the ICR for straight motion?

# Differential drive

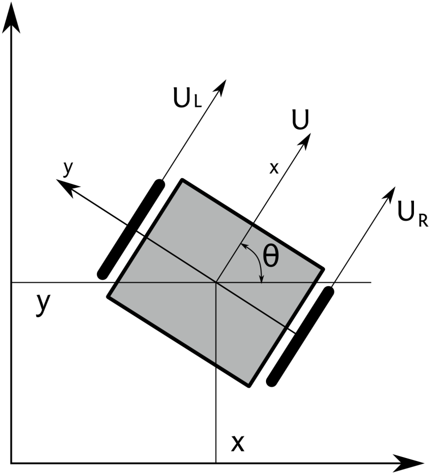
## Kinematics modeling

Kinematics model in the robot frame




$$\begin{bmatrix} u \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{-r}{L} & \frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}$$

Kinematics model in the world frame

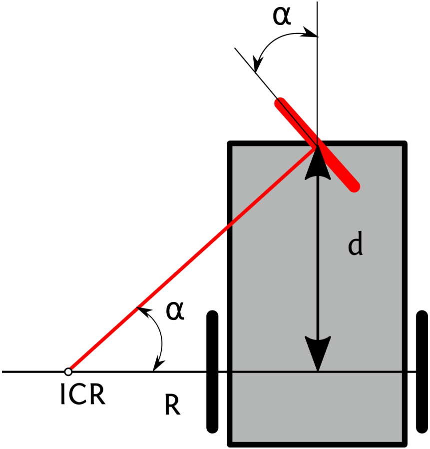



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix}$$

# Tricycle

## Kinematics model

Kinematics model in the robot body frame:

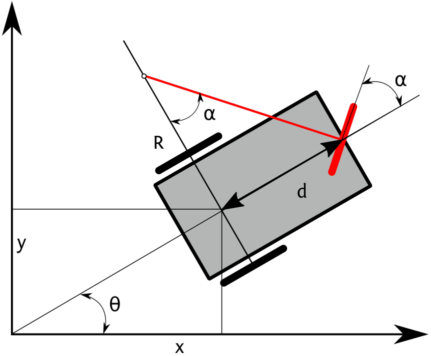



$$u(t) = u_s(t)cos(\alpha(t))$$
$$\omega(t) = \frac{u_s(t)}{d}sin(\alpha(t))$$

# Tricycle

## Kinematics model

Kinematics model in the world body frame:

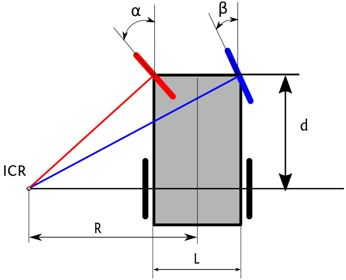



$$\dot{x} = u_s cos(\alpha(t)) \cos(\theta(t))$$
$$\dot{x} = u_s cos(\alpha(t)) \sin(\theta(t))$$
$$\dot{\theta}(t) = \frac{u_s(t)}{d} sin(\alpha(t))$$

# Four wheels

## Ackerman drive

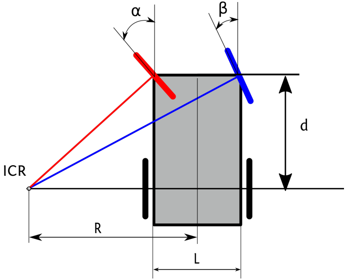
For this to work, the steering of the two wheels must be coordinated:




# Four wheels

## Ackerman drive

For this to work, the steering of the two wheels must be coordinated:



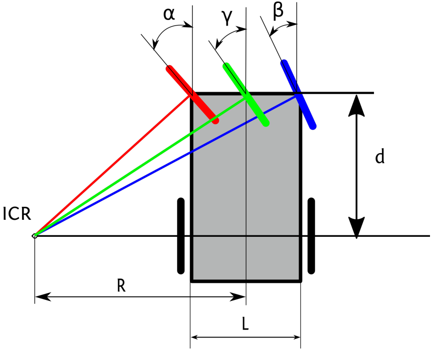

$\alpha > \beta$: when turning left

$\beta > \alpha$: when turning right

# Four wheels

## Ackerman drive

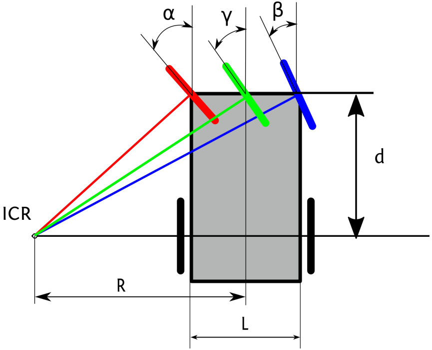
We can easily calculate the equivalent virtual angle $\gamma$



# Four wheels

## Ackerman drive

We can easily calculate the equivalent virtual angle $\gamma$



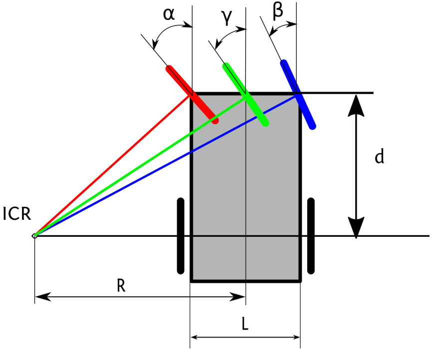

$$cot(\gamma) = cot(\alpha) + \frac{L}{2d}$$

$$cot(\gamma) = cot(\beta) - \frac{L}{2d}$$

# Four wheels

## Ackerman drive

We can easily calculate the equivalent virtual angle $\gamma$




$$cot(\gamma) = cot(\alpha) + \frac{L}{2d}$$

$$cot(\gamma) = cot(\beta) - \frac{L}{2d}$$

The kinematics models then are the same as for a tricycle with steering angle $\gamma$

# Mobile robots

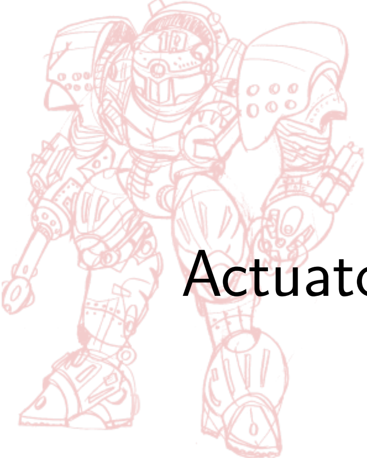
## Motion planning methods

**Roadmap approaches**:
Reduce all the possible paths to a subset of them

**Cell decomposition**:
Account for all of the free space

**Potential fields**:
Local control strategies, optimality

**Bug algorithms**:
Limited knowledge of environment



# Actuators and Sensors

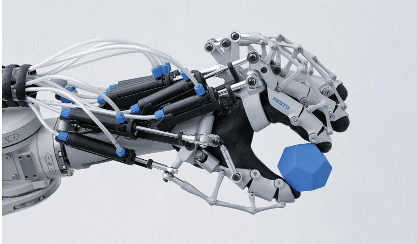
# Actuators

What type of actuator? Control signal?

# Actuators

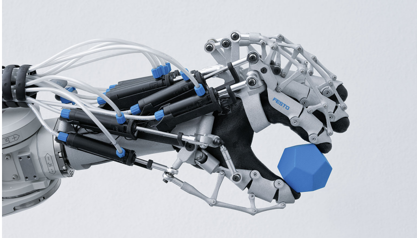
What type of actuator? Control signal?

# Actuators



What type of actuator? Control signal?

# Actuators




What type of actuator? Control signal?

# Sensors

## Many many types

There is practically a huge amount of sensors used in robots, depending on the application

# Sensors

## Many many types

There is practically a huge amount of sensors used in robots, depending on the application

# Sensors

## Many many types

There is practically a huge amount of sensors used in robots, depending on the application

# Sensors

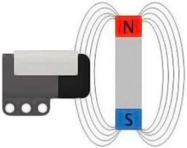
## Many many types

There is practically a huge amount of sensors used in robots, depending on the application

# Sensors

## Many many types

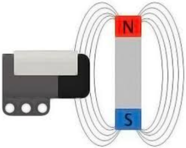
There is practically a huge amount of sensors used in robots, depending on the application



# Sensors

## Many many types

There is practically a huge amount of sensors used in robots, depending on the application




And many many more

# What's next?

Questions?