Robotics

Nicolas Marchand

Nicolas.Marchand@gipsa-lab.fr

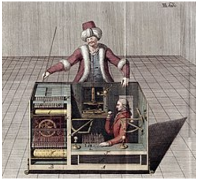
COPERNIC Team, Data Science Pole, **gipsa**-lab

Grenoble, France

ENSIT

Introduction

- Historical perspective
  - A great interest in "automatic/robotic" systems even in mythology:
    - Talos, created by Hephaistos and offered to the king Minos
    - In Iliad, Hephaistosis referred as creator of technical artificial creatures
    - 384-322 BJC, Aristote is speaking about machines doing human work

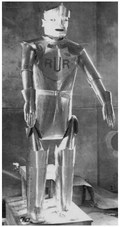


- First realizations
  - In Egypt: moving jaw of Anubis mask, moving arm of Amon's statue to designate the new pharaon
  - In Alexandria: fountains with moving birds (hydraulic systems)
  - Early mechanical automatons in the 9th/10th century (mainly in the Arabic world)
  - Clocks and automatons in the 13th/14th century (Europe)
  - Industrial automatons: 18th/19th century (e.g. Vaucanson, Jacquard)

Introduction

- Historical perspective
  - First use of the word Robot (means forced labor or serf in Czech) in the play R.U.R. (Rossum's Universal Robots) by Karel Capek (1890-1938) in January 1921.

  In R.U.R., Capek poses a paradise, where the machines initially bring so many benefits but in the end bring an equal amount of blight in the form of unemployment and social unrest



RUR

- Science fiction
  - Often a bad image: men against robots, dystopic society, etc. More and more a good image.

Formal definition (Robot Institute of America)

A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks

Robots and their image

- Robots have a bad image (1930-1960)
  - Robots take human works
  - Robots are dangerous since potentially independent and more intelligent than we are
- Robots have a better image (1960-today)
  - Robots can make things that human can not do (space, etc.)
  - Human can do things that robots can not do (we still are clever)
  - Robots can be games
  - Robots can be good or bad

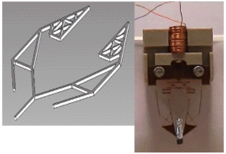
Robotics industry (1/many)

- Where are the robots ?
  - France:
    - 61% in automotive industry
    - 14% in chemical industry
    - ...
    - 4% in electricity industry
    - 3% in food industry

- What kind of robots ?
  - Industry: ground fixed robots: manipulators, arm robots, ...
  - Private individuals: mobile robots: service, games, ...
- Future of robots:
  - Industrial mobile robotics
  - Medical robotics
  - Service robots (growing field)

Robotics industry... today (2/many)

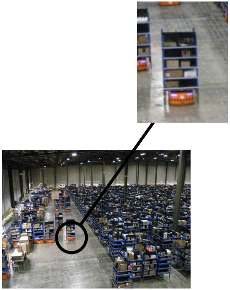
Vacuum cleaner



Micromanipulator

Surgical robot

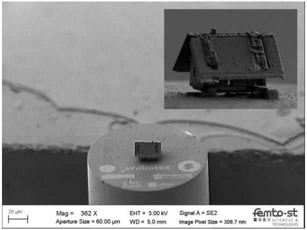
Forest robot



Kuka robot for automotive industry

Hollywood robots

… AND IN THE FUTURE ? (2 BIS/MANY)

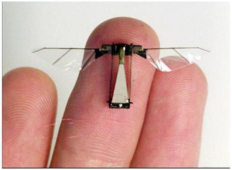


Nanorobotics (FEMTO-ST, Fr)

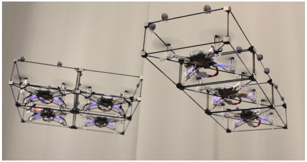
Military robots (Black Hornet, FLIR Systems, No)
2015 : Autonomous weapons: an open letter from AI & Robotics researchers [link]

Bio-inspired robots
Spot from Boston Dynamics

Exoskeleton



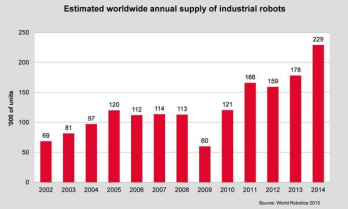
Microrobotics (Harvard, USA)



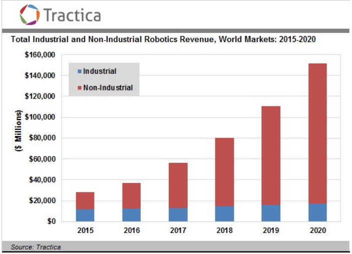
Modular robots

Robotics industry (3/many)

- Past

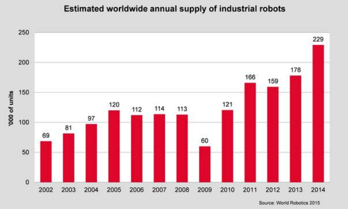


- Future

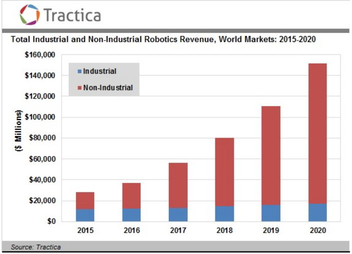


Robotics industry (3/many)

- Past



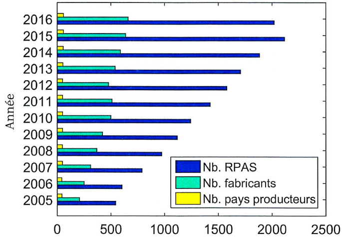
- Future



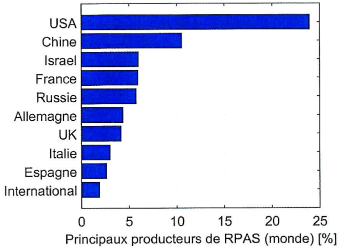
- Enova Robotics, Sousse, Tunisia

Robotics industry: UAVs (4/many)

- UAV's Manufacturer

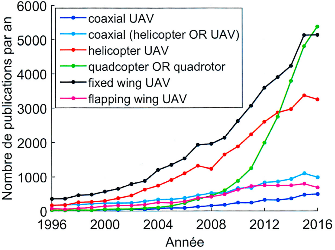


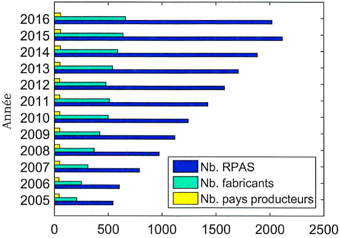
- UAVs by countries



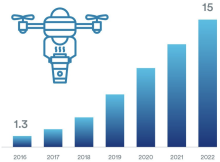
Robotics industry: UAVs (5/many)

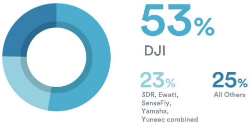
- Development of the drone's industry:





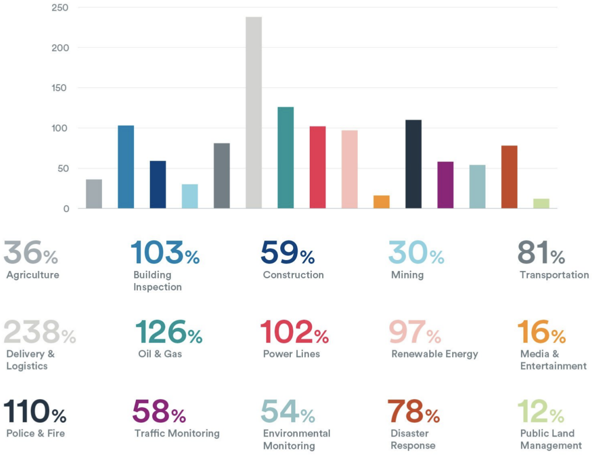
- Foundation of DJI: 2006





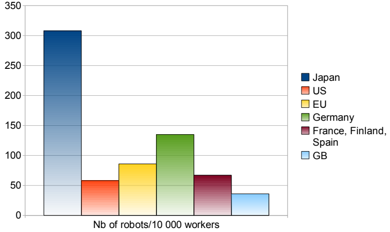
- Very competitive market with a high technological level of intergration
- Commercial margin of 10% to 15% (more than 50% on iPhone)

Robotics industry: UAVs (6/many)



Robolution (1/many)

- Number of robots for every 10 000 workers:



- 70% of robots in companies with more than 1000 employees
- 17% of robots in companies with less than 300 employees
- In 2002, 95% of robots > 30k€ and 32% of robots > 60k€
- 79% of decrease of the mean price between 1990 and 2002
- Big robots manufacturers: ABB (S), KUKA (G), Fanuc (JP), etc.

Robolution (2/many)

- Robotics enables 90% of cost reduction (60% for delocation)
- Each new robot destroys 6.2 jobs [MIT/Boston 1990-2007, 2017]
- 47% of jobs in the US, 50% of jobs in Europe have a high risk of being replaced by robots in the next 20 years [Oxford, 2013] ... but only 9% according [OCDE, 2016]
- Poor countries are more vulnerable, especially world factories (85% of the jobs in Ethiopia, 77% in Chine [World Bank])
- Sectors with high impact: Administration et Production
- Winner sectors: Finance, Maths/Sciences, Education
- No link between unemployment and robots
- Helps to relocate jobs in countries where the consumers are
- Very few studies on created jobs (compared to destroyed jobs)
- 800 000 direct jobs in robotics in 2020 and more than 2 millions in connected domains (electronic, energy, agriculture, etc.)

Robolution (3/many)

- What about the previous industrial revolution ?
  - Machines have created more jobs than they have replaced in the last 140 years
  - Working is getting less and less exhausting
  - Increase of new jobs (+580% éducation)
  - But we had fears, as in any big change periods:
    - 1675 : Destruction of machines by weavers (England), 1788 : 2000 workers break weaving machines (France), 1811-1812 : Luddism (Angleterre)
    - 1858 : Karl Marx is prophesies the replacement of the humans by machines
    - 1930 : John Maynard Keynes invents the term "technological unemployment"

Outline

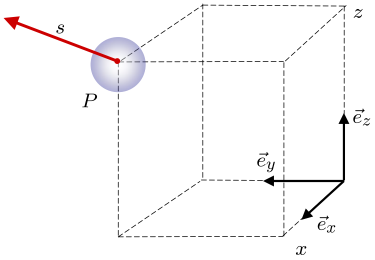
- Basic mechanics for robotics
  - Space representation
    frames, coordinate transformation, etc.
  - Force and torques
- Modelisation
- Control for robots
  - All potential problems:
    Oscillations, dry friction, saturations, etc.
  - Linear approaches
  - Nonlinear approaches

Outline

Outline

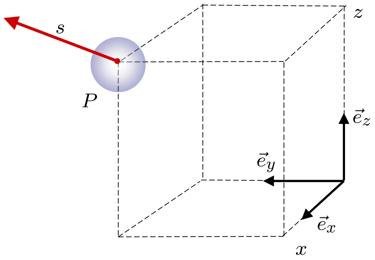
Position and speed

- The **position** of some point P in the **fixed** frame $\mathcal{F}(o, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ is the vector $\vec{p} = (x, y, z)^T$



Position and speed

- The **position** of some point P in the **fixed** frame $\mathcal{F}(o, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ is the vector $\vec{p} = (x, y, z)^T$
- The **speed** of P in $\mathcal{F}$ is the vector $\vec{s} = \dot{\vec{p}} = (\dot{x}, \dot{y}, \dot{z})^T$



Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:
  - axis $\vec{e}_x$ is given by:

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

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:
  - axis $\vec{e}_x$ is given by:

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

  - axis $\vec{e}_y$ is given by:

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

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:
  - axis $\vec{e}_x$ is given by:

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

  - axis $\vec{e}_y$ is given by:

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

  - axis $\vec{e}_z$ is given by:

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

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:
  - axis $\vec{e}_x$ is given by:

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

  - axis $\vec{e}_y$ is given by:

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

  - axis $\vec{e}_z$ is given by:

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

  - a unit vector $\vec{u} = (u_x, u_y, u_z)^T$:

$$\begin{pmatrix} u_x^2 + (1-u_x^2)c_\theta & u_x u_y(1-c_\theta) - u_z s_\theta & u_x u_z(1-c_\theta) + u_y s_\theta \\ u_x u_y(1-c_\theta) + u_z s_\theta & u_y^2 + (1-u_y^2)c_\theta & u_y u_z(1-c_\theta) - u_x s_\theta \\ u_x u_z(1-c_\theta) - u_y s_\theta & u_y u_z(1-c_\theta) + u_x s_\theta & u_z^2 + (1-u_z^2)c_\theta \end{pmatrix}$$

with $c_. = \cos(\cdot)$ and $s_. = \sin(\cdot)$ (and later on $t_. = \tan(\cdot)$)

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:
  - axis $\vec{e}_x$ is given by:

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

  - axis $\vec{e}_y$ is given by:

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

  - axis $\vec{e}_z$ is given by:

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

  - a unit vector $\vec{u} = (u_x, u_y, u_z)^T$:

$$\begin{pmatrix} u_x^2 + (1-u_x^2)c_\theta & u_x u_y(1-c_\theta) - u_z s_\theta & u_x u_z(1-c_\theta) + u_y s_\theta \\ u_x u_y(1-c_\theta) + u_z s_\theta & u_y^2 + (1-u_y^2)c_\theta & u_y u_z(1-c_\theta) - u_x s_\theta \\ u_x u_z(1-c_\theta) - u_y s_\theta & u_y u_z(1-c_\theta) + u_x s_\theta & u_z^2 + (1-u_z^2)c_\theta \end{pmatrix}$$

  with $c_. = \cos(\cdot)$ and $s_. = \sin(\cdot)$ (and later on $t_. = \tan(\cdot)$)
- The coordinates q of point Q obtained by rotating P with rotation R is $q = Rp$

Rotations

- A rotation is represented by a 3×3 matrix R such that $R^T = R^{-1}$ and $\det R = 1$
- A **rotation** of angle θ around:
  - axis $\vec{e}_x$ is given by:

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

  - axis $\vec{e}_y$ is given by:

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

  - axis $\vec{e}_z$ is given by:

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

  - a unit vector $\vec{u} = (u_x, u_y, u_z)^T$:

$$\begin{pmatrix} u_x^2 + (1-u_x^2)c_\theta & u_x u_y(1-c_\theta) - u_z s_\theta & u_x u_z(1-c_\theta) + u_y s_\theta \\ u_x u_y(1-c_\theta) + u_z s_\theta & u_y^2 + (1-u_y^2)c_\theta & u_y u_z(1-c_\theta) - u_x s_\theta \\ u_x u_z(1-c_\theta) - u_y s_\theta & u_y u_z(1-c_\theta) + u_x s_\theta & u_z^2 + (1-u_z^2)c_\theta \end{pmatrix}$$

  with $c_\cdot = \cos(\cdot)$ and $s_\cdot = \sin(\cdot)$ (and later on $t_\cdot = \tan(\cdot)$)

- The coordinates q of point Q obtained by rotating P with rotation R is $q = Rp$
- The rotation resulting from 2 successive rotations R_1 and then R_2 is $R_2 R_1$

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 > := v_1^T v_2 \in \mathbb{R}$

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 >:= v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y} v_{2z} - v_{1z} v_{2y} \\ v_{1z} v_{2x} - v_{1x} v_{2z} \\ v_{1x} v_{2y} - v_{1y} v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 > := v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y} v_{2z} - v_{1z} v_{2y} \\ v_{1z} v_{2x} - v_{1x} v_{2z} \\ v_{1x} v_{2y} - v_{1y} v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

- The **skew-symmetric matrix** associated to a vector $p = (x, y, z)^T$ is:

$$p^\times := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 > := v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y} v_{2z} - v_{1z} v_{2y} \\ v_{1z} v_{2x} - v_{1x} v_{2z} \\ v_{1x} v_{2y} - v_{1y} v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

- The **skew-symmetric matrix** associated to a vector $p = (x, y, z)^T$ is:

$$p^\times := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

- The set of skew-symmetric matrix with the brackett $[M_1, M_2] = M_1 M_2 - M_2 M_1$ is called $SO(3)$ and forms an algebra

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 > := v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y} v_{2z} - v_{1z} v_{2y} \\ v_{1z} v_{2x} - v_{1x} v_{2z} \\ v_{1x} v_{2y} - v_{1y} v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

- The **skew-symmetric matrix** associated to a vector $p = (x, y, z)^T$ is:

$$p^\times := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

- The set of skew-symmetric matrix with the brackett $[M_1, M_2] = M_1 M_2 - M_2 M_1$ is called $SO(3)$ and forms an algebra
- Skew-symmetric matrices and cross product:

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 >:= v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y}v_{2z} - v_{1z}v_{2y} \\ v_{1z}v_{2x} - v_{1x}v_{2z} \\ v_{1x}v_{2y} - v_{1y}v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

- The **skew-symmetric matrix** associated to a vector $p = (x, y, z)^T$ is:

$$p^\times := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

- The set of skew-symmetric matrix with the brackett $[M_1, M_2] = M_1M_2 - M_2M_1$ is called $SO(3)$ and forms an algebra
- Skew-symmetric matrices and cross product:

$$v^\times u = v \times u$$

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 > := v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y}v_{2z} - v_{1z}v_{2y} \\ v_{1z}v_{2x} - v_{1x}v_{2z} \\ v_{1x}v_{2y} - v_{1y}v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

- The **skew-symmetric matrix** associated to a vector $p = (x, y, z)^T$ is:

$$p^\times := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

- The set of skew-symmetric matrix with the brackett $[M_1, M_2] = M_1M_2 - M_2M_1$ is called $SO(3)$ and forms an algebra
- Skew-symmetric matrices and cross product:

$$v^\times u = v \times u$$

- Skew-symmetric matrices and rotations

Products and associated tools

- The **scalar product** $< v_1, v_2 >$ is defined by: $< v_1, v_2 >:= v_1^T v_2 \in \mathbb{R}$
- The **cross product** $v_1 \times v_2$ is defined by:

$$v_1 \times v_2 := \begin{pmatrix} v_{1y}v_{2z} - v_{1z}v_{2y} \\ v_{1z}v_{2x} - v_{1x}v_{2z} \\ v_{1x}v_{2y} - v_{1y}v_{2x} \end{pmatrix} \in \mathbb{R}^3$$

- The **skew-symmetric matrix** associated to a vector $p = (x, y, z)^T$ is:

$$p^\times := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

- The set of skew-symmetric matrix with the brackett $[M_1, M_2] = M_1M_2 - M_2M_1$ is called $SO(3)$ and forms an algebra
- Skew-symmetric matrices and cross product:

$$v^\times u = v \times u$$

- Skew-symmetric matrices and rotations

$$u^\times \sin\theta + (I - uu^T)\cos\theta + uu^T = \exp((u\theta)^\times)$$

is the rotation of angle θ leaving axis u fixed

Attitude representation: angles

- Attitude:

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix

Attitude representation: angles

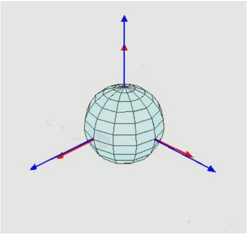
- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, ...

Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, ...

L. Euler (1707-1783)

- **Euler angles**: 3 angles, 27 possible rotations

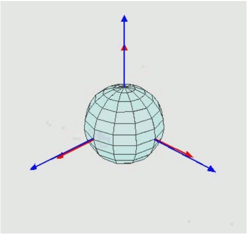


Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, ...

L. Euler (1707-1783)

- **Euler angles**: 3 angles, 27 possible rotations
- Engineering and robotics communities typically use 3-1-3 Euler angles

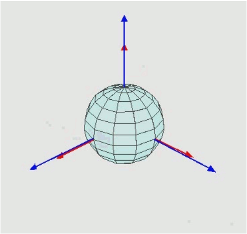


Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, . . .

L. Euler (1707-1783)

- **Euler angles**: 3 angles, 27 possible rotations
- Engineering and robotics communities typically use 3-1-3 Euler angles
- Representations with singularities



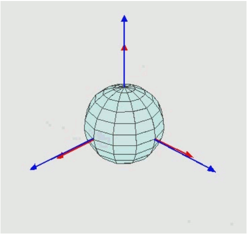
Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, . . .

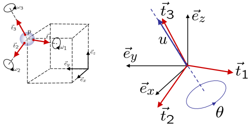
L. Euler (1707-1783)

W.R. Hamilton (1805-1865)

- **Euler angles**: 3 angles, 27 possible rotations
- Engineering and robotics communities typically use 3-1-3 Euler angles
- Representations with singularities



- **Quaternions**



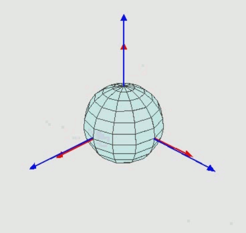
Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, ...

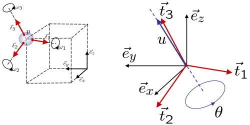
L. Euler (1707-1783)

W.R. Hamilton (1805-1865)

- **Euler angles**: 3 angles, 27 possible rotations
- Engineering and robotics communities typically use 3-1-3 Euler angles
- Representations with singularities



- **Quaternions**



- u fixed by rotation of angle θ
- the quaternion is:

$$q = \begin{pmatrix} u_x \sin\theta/2 \\ u_y \sin\theta/2 \\ u_z \sin\theta/2 \\ \cos\theta/2 \end{pmatrix} = \begin{pmatrix} \vec{q} \\ q_0 \end{pmatrix}$$

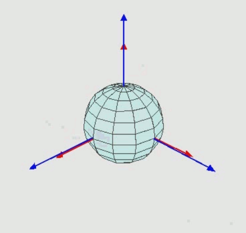
Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, ...

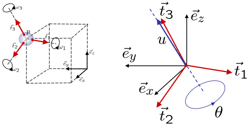
L. Euler (1707-1783)

W.R. Hamilton (1805-1865)

- **Euler angles**: 3 angles, 27 possible rotations
- Engineering and robotics communities typically use 3-1-3 Euler angles
- Representations with singularities



- **Quaternions**



- *u* fixed by rotation of angle θ
- the quaternion is:

$$q = \begin{pmatrix} u_x \sin\theta/2 \\ u_y \sin\theta/2 \\ u_z \sin\theta/2 \\ \cos\theta/2 \end{pmatrix} = \begin{pmatrix} \vec{q} \\ q_0 \end{pmatrix}$$

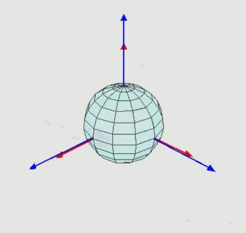
Attitude representation: angles

- Attitude:
  - equivalent of position for angles: what is the orientation of an object w.r.t. the ground ?
  - gives the **rotation that transforms the reference frame into the body frame**
- Many attitude representations
  - Euler angles
  - Quaternions
  - Rotation matrix
  - Tait-Bryan angles, Fick angles, Helmholtz angles, dip-slip-rake, azimuth-elevation-skew, . . .

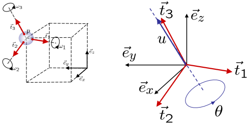
L. Euler (1707-1783)

W.R. Hamilton (1805-1865)

- **Euler angles**: 3 angles, 27 possible rotations
- Engineering and robotics communities typically use 3-1-3 Euler angles
- Representations with singularities



- **Quaternions**

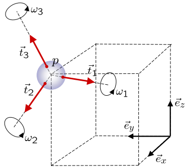


- u fixed by rotation of angle θ
- the quaternion is:

$$q = \begin{pmatrix} u_x \sin\theta/2 \\ u_y \sin\theta/2 \\ u_z \sin\theta/2 \\ \cos\theta/2 \end{pmatrix} = \begin{pmatrix} \vec{q} \\ q_0 \end{pmatrix}$$

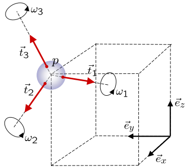
Attitude representation : angular velocities

- The angular velocity $\omega = (\omega_1, \omega_2, \omega_3)^T$ represents the rotation speed w.r.t. each axis of the body frame



Attitude representation : angular velocities

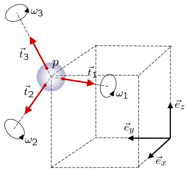
- The angular velocity $\omega = (\omega_1, \omega_2, \omega_3)^T$ represents the rotation speed w.r.t. each axis of the body frame



- **Caution:** *Angular velocities <u>are not</u> the time derivatives of Euler angles*

Attitude representation : angular velocities

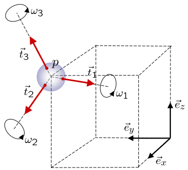
- The angular velocity $\omega = (\omega_1, \omega_2, \omega_3)^T$ represents the rotation speed w.r.t. each axis of the body frame



- **Caution:** *Angular velocities <u>are not</u> the time derivatives of Euler angles*
- Angular velocities are given by:

Attitude representation : angular velocities

- The angular velocity $\omega = (\omega_1, \omega_2, \omega_3)^T$ represents the rotation speed w.r.t. each axis of the body frame

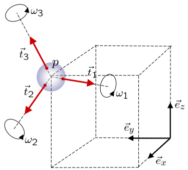


- **Caution:** *Angular velocities are not the time derivatives of Euler angles*
- Angular velocities are given by:
  - Rotation matrix:

$$\dot{R} = R\omega^\times$$

Attitude representation : angular velocities

- The angular velocity $\omega = (\omega_1, \omega_2, \omega_3)^T$ represents the rotation speed w.r.t. each axis of the body frame



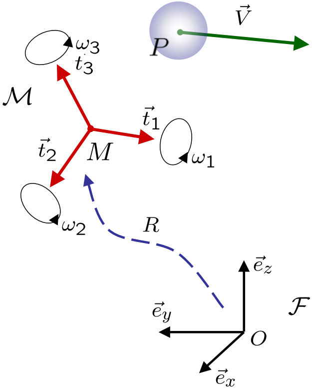
- **Caution:** *Angular velocities are not the time derivatives of Euler angles*
- Angular velocities are given by:
  - Rotation matrix:

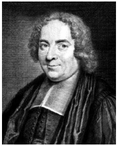
$$\dot{R} = R\omega^{\times}$$

  - Quaternions :

$$\begin{aligned} \dot{\vec{q}} &= \frac{1}{2}\Omega(\vec{\omega})q \\ &= \frac{1}{2}\Xi(q)\vec{\omega} \end{aligned} \quad \text{with} \quad \begin{cases} \Omega(\vec{\omega}) = \begin{pmatrix} 0 & -\vec{\omega}^T \\ \vec{\omega} & -\vec{\omega}^{\times} \end{pmatrix} \\ \Xi(q) = \begin{pmatrix} -\vec{q}^T \\ I_{3\times 3}q_0 + \vec{q}^{\times} \end{pmatrix} \end{cases}$$

Moving frames





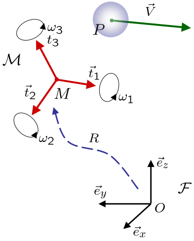
P. Varignon (1654-1722)

Varignon's formula

$$\frac{d\vec{U}^{\mathcal{M}}}{dt} = \frac{d\vec{U}^{\mathcal{F}}}{dt} + \Omega^{\mathcal{F}/\mathcal{M}} \times \vec{U}^{\mathcal{F}}$$

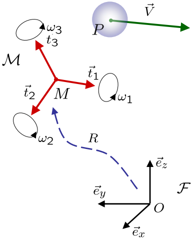
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed inertial frame



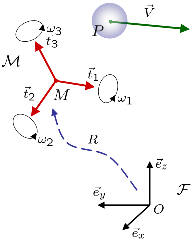
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed inertial frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame



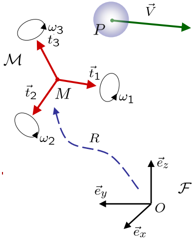
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed inertial frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame
- R: rotation matrix s.t. $\mathcal{M} = R\mathcal{F}$



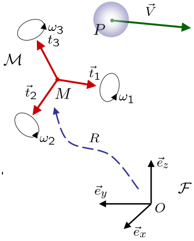
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed inertial frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame
- R: rotation matrix s.t. $\mathcal{M} = R\mathcal{F}$
- $\Omega^{\mathcal{M}/\mathcal{F}}$: angular velocity matrix of $\mathcal{M}$ w.r.t.



Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed inertial frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame
- R: rotation matrix s.t. $\mathcal{M} = R\mathcal{F}$
- $\Omega^{\mathcal{M}/\mathcal{F}}$: angular velocity matrix of $\mathcal{M}$ w.r.t.



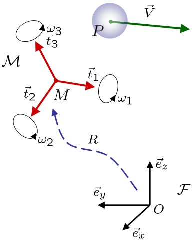
- **Velocities**:
  - Absolute velocity

$$\left.\frac{d\vec{OP}}{dt}\right|^{\mathcal{F}} = \left.\frac{d\vec{OM}}{dt}\right|^{\mathcal{F}} + \left.\frac{d\vec{MP}}{dt}\right|^{\mathcal{M}} + \Omega^{\mathcal{M}/\mathcal{F}} \times \vec{MP}$$

  - Speed of $\mathcal{M}$ w.r.t $\mathcal{F}$
  - Relative velocity
  - Due to the rotation of $\mathcal{M}$ w.r.t. $\mathcal{F}$

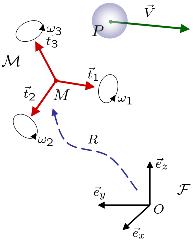
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed frame



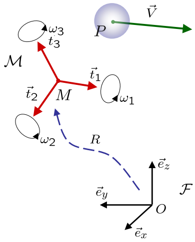
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame



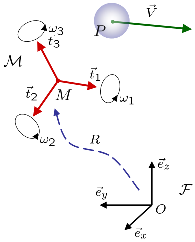
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame
- R: rotation matrix s.t. $\mathcal{M} = R\mathcal{F}$



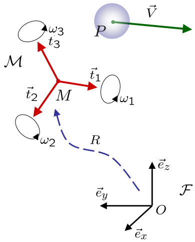
Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame
- R: rotation matrix s.t. $\mathcal{M} = R\mathcal{F}$
- $\Omega^{\mathcal{M}/\mathcal{F}}$: angular velocity matrix of $\mathcal{M}$ w.r.t. $\mathcal{F}$



Moving frames

- $\mathcal{F} := (O, \vec{e}_x, \vec{e}_y, \vec{e}_z)$ fixed frame
- $\mathcal{M} := (M, \vec{t}_1, \vec{t}_2, \vec{t}_3)$: mobile frame
- R: rotation matrix s.t. $\mathcal{M} = R\mathcal{F}$
- $\Omega^{\mathcal{M}/\mathcal{F}}$: angular velocity matrix of $\mathcal{M}$ w.r.t. $\mathcal{F}$
- **Acceleration**:



$$\ddot{P}^{\mathcal{F}} := \left(\frac{d\dot{P}^{\mathcal{F}}}{dt}\right)^{\mathcal{F}} = \frac{d\dot{P}^{\mathcal{M}}}{dt}^{\mathcal{F}} + \frac{d\Omega^{\mathcal{M}/\mathcal{F}} \times P^{\mathcal{F}}}{dt}$$

- $\dfrac{d\dot{P}^{\mathcal{M}}}{dt}^{\mathcal{F}} = \ddot{P}^{\mathcal{M}} + \Omega^{\mathcal{M}/\mathcal{F}} \times \dot{P}^{\mathcal{M}}$ (Varignon's formula)
- $\dfrac{d\Omega^{\mathcal{M}/\mathcal{F}} \times P^{\mathcal{F}}}{dt} = \dot{\Omega}^{\mathcal{M}/\mathcal{F}} \times P^{\mathcal{F}} + \Omega^{\mathcal{M}/\mathcal{F}} \times \dot{P}^{\mathcal{F}}$

$$= \dot{\Omega}^{\mathcal{M}/\mathcal{F}} \times P^{\mathcal{F}} + \Omega^{\mathcal{M}/\mathcal{F}} \times \dot{P}^{\mathcal{M}} + \Omega^{\mathcal{M}/\mathcal{F}} \times (\Omega^{\mathcal{M}/\mathcal{F}} \times P^{\mathcal{F}})$$

all together:

$$\ddot{P}^{\mathcal{M}} = \ddot{P}^{\mathcal{F}} - 2\Omega \times \dot{P}^{\mathcal{M}} - \dot{\Omega} \times P^{\mathcal{F}} - \Omega \times (\Omega \times P^{\mathcal{F}})$$

- Coriolis effect
- Euler effect (tangent acceleration)
- Centrifugal effect

Newton's laws

Consider:

- an inertial frame $\mathcal{F}$
- a body of mass $m := \sum_i m_i$ composed of elements located in $\vec{p}_i$ with speed $\vec{v}_i$ in $\mathcal{F}$
- *or* a body of mass $m := \int_{\text{body}} dm$ composed of elementary part located in $\vec{p}_{dm}$ with speed $\vec{v}_{dm}$ in $\mathcal{F}$
- $\vec{p} := \dfrac{\sum_i m_i \vec{p}_i}{m}$ defines the position of its center of mass G in $\mathcal{F}$
- *or* $\vec{p} := \dfrac{\int_{\text{body}} dm \vec{p}_{dm}}{m}$ defines the position of its center of mass G in $\mathcal{F}$
- $\vec{v} := \dot{\vec{p}}$ defines speed of the center of mass
- $\vec{r}_i := (\vec{p}_i - \vec{p})$ (resp. $\vec{r}_{dm} := (\vec{p}_{dm} - \vec{p})$)

Linear Momentum

$$\vec{P} := \sum_i m_i \vec{v}_i = m\vec{v} \in \mathbb{R}^3$$

$$\vec{P} := \int_{\text{body}} \vec{v}_{dm} dm \in \mathbb{R}^3$$

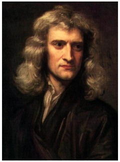
Angular Momentum

$$\vec{L} := \sum_i m_i (\vec{p}_i - \vec{p}) \times \vec{v}_i$$

$$\vec{L} := \int_{\text{body}} (\vec{p}_{dm} - \vec{p}) \times \vec{v}_{dm} dm$$

$$= \underbrace{\int_{\text{body}} ||\vec{r}_{dm}||^2 \, dm}_{J:\ \text{moment of inertia}} \quad \vec{\omega}$$

Newton's laws


I. Newton (1643-1727)


J. L. Lagrange (1736-1813)

Consider:

- a rigid body
- an inertial frame $\mathcal{F}$
- a moving frame $\mathcal{M}$ centered in the center of mass and aligned with the main axis of the rigid body
- Let $\vec{F}_i$'s be forces applying on the body with moment arm $\vec{a}_i$

Newton's second law

$$\sum \vec{F} = \frac{d\vec{P}^{\mathcal{F}}}{dt}$$

Conservation of the angular momentum

$$\sum \vec{\tau} = \frac{d\vec{L}^{\mathcal{F}}}{dt}$$

- In a moving frame (Varignon's formula):

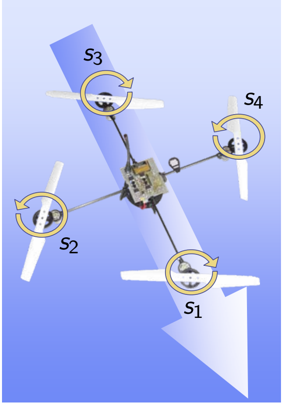
$$\frac{d\vec{L}^{\mathcal{F}}}{dt} = \frac{d\vec{L}^{\mathcal{M}}}{dt} + \Omega \times \vec{L}$$

Menu

How it works

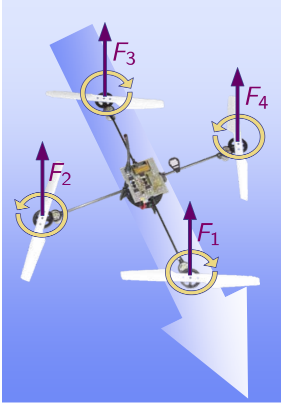
How it works

- 4 fixed rotors with controlled rotation speed s_i



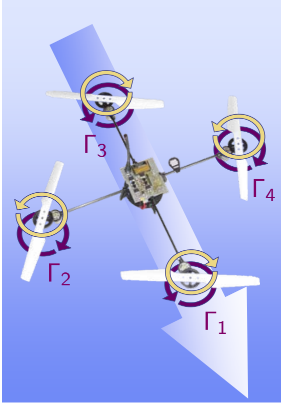
How it works

- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i



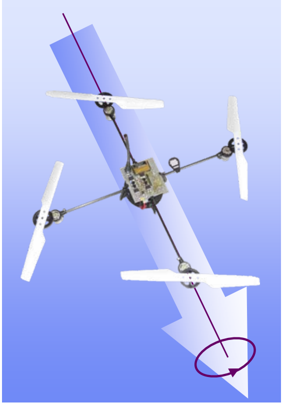
How it works

- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i



How it works

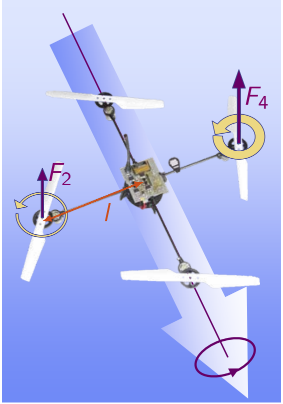
- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i
- **Roll movement**



How it works

- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i
- **Roll movement** generated with a dissymmetry between left and right forces:

$$\Gamma_r = l(F_4 - F_2)$$

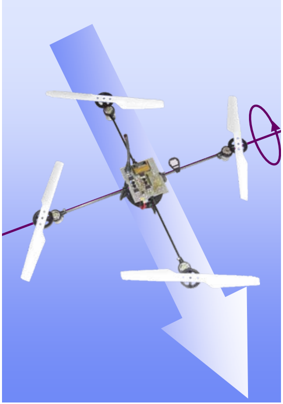


How it works

- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i
- **Roll movement** generated with a dissymmetry between left and right forces:

$$\Gamma_r = l(F_4 - F_2)$$

- **Pitch movement**



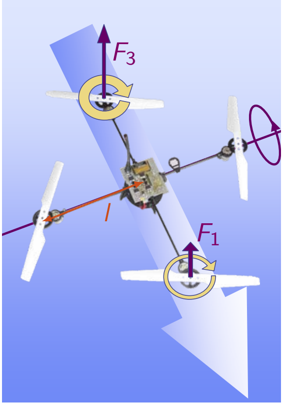
How it works

- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i
- **Roll movement** generated with a dissymmetry between left and right forces:

$$\Gamma_r = l(F_4 - F_2)$$

- **Pitch movement** generated with a dissymmetry between front and rear forces:

$$\Gamma_p = l(F_1 - F_3)$$



How it works

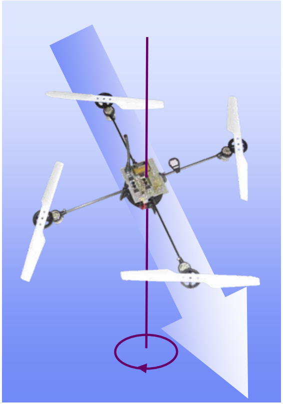
- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i
- **Roll movement** generated with a dissymmetry between left and right forces:

$$\Gamma_r = l(F_4 - F_2)$$

- **Pitch movement** generated with a dissymmetry between front and rear forces:

$$\Gamma_p = l(F_1 - F_3)$$

- **Yaw movement**



How it works

- 4 fixed rotors with controlled rotation speed s_i
- 4 generated forces F_i
- 4 counter-rotating torques Γ_i
- **Roll movement** generated with a dissymmetry between left and right forces:

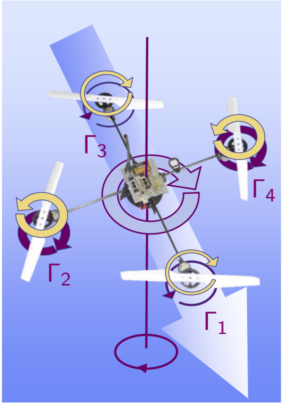
$$\Gamma_r = l(F_4 - F_2)$$

- **Pitch movement** generated with a dissymmetry between front and rear forces:

$$\Gamma_p = l(F_1 - F_3)$$

- **Yaw movement** generated with a dissymmetry between front/rear and left/right torques:

$$\Gamma_y = \Gamma_1 + \Gamma_3 - \Gamma_2 - \Gamma_4$$



Modeling the quadrotor

Actuation and aerodynamics

- **Electrical motor**: A 2^{nd} order system with friction and saturation

Actuation and aerodynamics

- **Electrical motor**: A 2nd order system with friction and saturation usually *approximated* by a 1rst order system:

$$\dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\mathrm{sat}_{\bar{U}_i}(U_i) \quad i \in \{1,2,3,4\}$$

s_i: rotation speed

U_i: voltage applied to the motor; real control variable

τ_{load}: motor load: $\tau_{\text{load}} = k_{gearbox} c_D |s_i| s_i$ with c_D drag coefficient

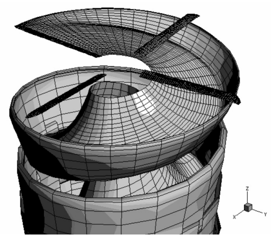
Actuation and aerodynamics

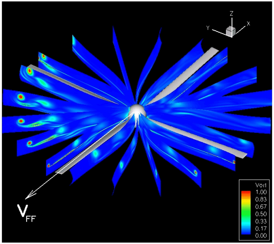
- **Electrical motor**: A 2^{nd} order system with friction and saturation usually *approximated* by a 1^{rst} order system:

$$\dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\text{sat}_{\bar{U}_i}(U_i) \quad i \in \{1,2,3,4\}$$

s_i: rotation speed
U_i: voltage applied to the motor; real control variable
τ_{load}: motor load: $\tau_{\text{load}} = k_{gearbox} c_D |s_i| s_i$ with c_D drag coefficient

- **Aerodynamical forces and torques**: Very complex models exist





Actuation and aerodynamics

- **Electrical motor**: A 2nd order system with friction and saturation usually *approximated* by a 1rst order system:

$$\dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\text{sat}_{\bar{U}_i}(U_i) \quad i \in \{1,2,3,4\}$$

s_i: rotation speed
U_i: voltage applied to the motor; real control variable
τ_{load}: motor load: $\tau_{\text{load}} = k_{gearbox} c_D \left|s_i\right| s_i$ with c_D drag coefficient

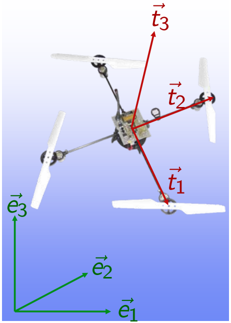
- **Aerodynamical forces and torques**: Very complex models exist but overcomplicated for control, better use the *simplified* model:

$$\begin{aligned} F_i &= c_T s_i^2 \\ \Gamma_r &= l c_T (s_4^2 - s_2^2) \\ \Gamma_p &= l c_T (s_1^2 - s_3^2) \\ \Gamma_y &= l c_D (s_1^2 + s_3^2 - s_2^2 - s_4^2) \end{aligned} \qquad i \in \{1,2,3,4\}$$

c_T: thrust coefficient, c_D: drag coefficient

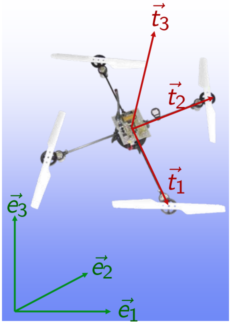
Frames and variables

- Two frames
  - a fixed frame $\mathcal{E}(\vec{e}_1, \vec{e}_2, \vec{e}_3)$
  - a frame attached to the X4 $\mathcal{T}(\vec{t}_1, \vec{t}_2, \vec{t}_3)$



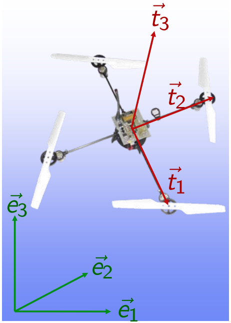
Frames and variables

- Two frames
  - a fixed frame $\mathcal{E}(\vec{e}_1, \vec{e}_2, \vec{e}_3)$
  - a frame attached to the X4 $\mathcal{T}(\vec{t}_1, \vec{t}_2, \vec{t}_3)$
- Frame change
  - a rotation matrix R from $\mathcal{T}$ to $\mathcal{E}$



Frames and variables

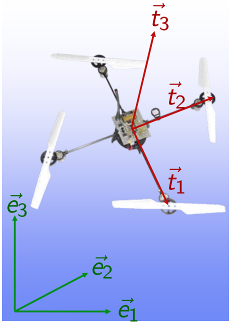
- Two frames
  - a fixed frame $\mathcal{E}(\vec{e}_1, \vec{e}_2, \vec{e}_3)$
  - a frame attached to the X4 $\mathcal{T}(\vec{t}_1, \vec{t}_2, \vec{t}_3)$
- Frame change
  - a rotation matrix R from $\mathcal{T}$ to $\mathcal{E}$



- State variables:

Frames and variables

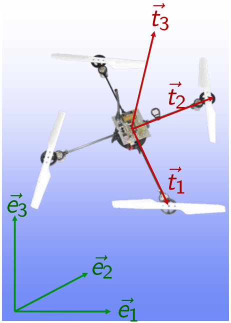
- Two frames
  - a fixed frame $\mathcal{E}(\vec{e}_1, \vec{e}_2, \vec{e}_3)$
  - a frame attached to the X4 $\mathcal{T}(\vec{t}_1, \vec{t}_2, \vec{t}_3)$
- Frame change
  - a rotation matrix R from $\mathcal{T}$ to $\mathcal{E}$



- State variables:
  - Cartesian coordinates (in $\mathcal{E}$)
    - position $\vec{p}$
    - velocity $\vec{v}$

Frames and variables

- Two frames
  - a fixed frame $\mathcal{E}(\vec{e}_1, \vec{e}_2, \vec{e}_3)$
  - a frame attached to the X4 $\mathcal{T}(\vec{t}_1, \vec{t}_2, \vec{t}_3)$
- Frame change
  - a rotation matrix R from $\mathcal{T}$ to $\mathcal{E}$



- State variables:
  - Cartesian coordinates (in $\mathcal{E}$)
    - position $\vec{p}$
    - velocity $\vec{v}$
  - Attitude coordinates:
    - angular velocity $\vec{\omega}$ in the moving frame $\mathcal{T}$
    - *either:* Euler angles three successive rotations about $\vec{t}_3$, $\vec{t}_1$ and $\vec{t}_3$ of angles angles ϕ, θ and ψ giving R
    - *or:* Quaternion representation $(q_0, \vec{q}) = (\cos\beta/2, \vec{u}\sin\beta/2)$ represent a rotation of angle β about $\vec{u}$

Newton's laws

- **Cartesian coordinates**:

$$\begin{cases} \dot{\vec{p}} = \vec{v} \\ m\dot{\vec{v}} = -mg\vec{e}_3 + R \cdot \underbrace{\sum_i F_i(s_i)\vec{t}_3}_{\vec{T}\ :\ \text{control thrust}} + \vec{F}_{ext} \end{cases}$$

Newton's laws

- **Cartesian coordinates**:

$$\begin{cases} \dot{\vec{p}} = \vec{v} \\ m\dot{\vec{v}} = -mg\vec{e}_3 + R \cdot \underbrace{\sum_i F_i(s_i)\vec{t}_3}_{\vec{T}\ :\ \text{control thrust}} + \vec{F}_{ext} \end{cases}$$

- **Attitude**:

Newton's laws

- **Cartesian coordinates**:

$$\left\{\begin{array}{rcl} \dot{\vec{p}} & = & \vec{v} \\ m\dot{\vec{v}} & = & -mg\vec{e}_3 + R \cdot \underbrace{\sum_i F_i(s_i)\vec{t}_3}_{\vec{T}\ :\ \text{control thrust}} + \vec{F}_{ext} \end{array}\right.$$

- **Attitude**:
  - Rotation matrix formalism:

$$\left\{\begin{array}{rcl} \dot{R} & = & R\vec{\omega}^\times \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^\times J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{array}\right. \quad \text{with } \vec{\omega}^\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

$\vec{\omega}^\times$ is the skew symmetric tensor associated to $\vec{\omega}$

Newton's laws

- **Cartesian coordinates**:

$$\begin{cases} \dot{\vec{p}} = \vec{v} \\ m\dot{\vec{v}} = -mg\vec{e}_3 + R \cdot \underbrace{\sum_i F_i(s_i)\vec{t}_3}_{\vec{T}\ :\ \text{control thrust}} + \vec{F}_{ext} \end{cases}$$

- **Attitude**:
  - Rotation matrix formalism:

$$\begin{cases} \dot{R} = R\vec{\omega}^\times \\ J\dot{\vec{\omega}} = -\vec{\omega}^\times J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases} \quad \text{with } \vec{\omega}^\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

  $\vec{\omega}^\times$ is the skew symmetric tensor associated to $\vec{\omega}$
  - Quaternion formalism:

$$\begin{cases} \dot{\vec{q}} = \frac{1}{2}\Omega(\vec{\omega})\vec{q} \\ \quad = \frac{1}{2}\Xi(q)\vec{\omega} \\ J\dot{\vec{\omega}} = -\vec{\omega}^\times J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases} \quad \text{with } \begin{cases} \Omega(\vec{\omega}) = \begin{pmatrix} 0 & -\vec{\omega}^T \\ \vec{\omega} & -\vec{\omega}^\times \end{pmatrix} \\ \Xi(q) = \begin{pmatrix} -\vec{q}^T \\ I_{3\times 3}q_0 + \vec{q}^\times \end{pmatrix} \end{cases}$$

Newton's laws

- **Cartesian coordinates**:

$$\begin{cases} \dot{\vec{p}} &= \vec{v} \\ m\dot{\vec{v}} &= -mg\vec{e}_3 + R \cdot \underbrace{\sum_i F_i(s_i)\vec{t}_3}_{\vec{T}\ :\ \text{control thrust}} + \vec{F}_{ext} \end{cases}$$

- **Attitude**:
  - Rotation matrix formalism:

$$\begin{cases} \dot{R} &= R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} &= -\vec{\omega}^{\times} J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases} \quad \text{with } \vec{\omega}^{\times} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

    $\vec{\omega}^{\times}$ is the skew symmetric tensor associated to $\vec{\omega}$
  - Quaternion formalism:

$$\begin{cases} \dot{\vec{q}} &= \frac{1}{2}\Omega(\vec{\omega})\vec{q} \\ &= \frac{1}{2}\Xi(q)\vec{\omega} \\ J\dot{\vec{\omega}} &= -\vec{\omega}^{\times} J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases} \quad \text{with } \begin{cases} \Omega(\vec{\omega}) = \begin{pmatrix} 0 & -\vec{\omega}^T \\ \vec{\omega} & -\vec{\omega}^{\times} \end{pmatrix} \\ \Xi(q) = \begin{pmatrix} -\vec{q}^T \\ I_{3\times 3}q_0 + \vec{q}^{\times} \end{pmatrix} \end{cases}$$

where $\vec{\Gamma}_c = \begin{pmatrix} \Gamma_r(s_2, s_4) \\ \Gamma_p(s_1, s_3) \\ \Gamma_y(s_1, s_2, s_3, s_4) \end{pmatrix}$ are the control torques

The wronskian matrix

- Consider the 1-2-3 Euler angles (ϕ, θ, ψ)
- The rotation matrix is given by:

$$R = R_z R_y R_x = \begin{pmatrix} c_\theta c_\phi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\phi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}$$

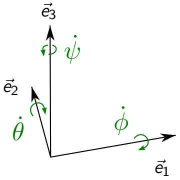
The wronskian matrix

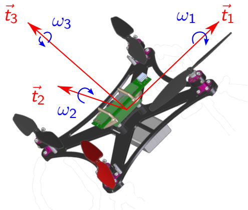
- Consider the 1-2-3 Euler angles (ϕ, θ, ψ)
- The rotation matrix is given by:

$$R = R_z R_y R_x = \begin{pmatrix} c_\theta c_\phi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\phi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}$$

- The relation between the time derivative of the Euler angles and the angular velocity is:

$$\vec{\omega} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R_z \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + R_z R_y \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} = W^{-1} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}$$





The wronskian matrix

- Consider the 1-2-3 Euler angles (ϕ, θ, ψ)
- The rotation matrix is given by:

$$R = R_z R_y R_x = \begin{pmatrix} c_\theta c_\phi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\phi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}$$

- The relation between the time derivative of the Euler angles and the angular velocity is:

$$\vec{\omega} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R_z \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + R_z R_y \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} = W^{-1} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}$$

- W is called the **wronskian matrix** given by (for 1-2-3 Euler angles):

$$W = \begin{pmatrix} 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \\ 0 & c_\phi & -s_\phi \\ 1 & s_\phi t_\theta & c_\phi t_\theta \end{pmatrix}$$

- This matrix is singular for $\theta = \pi/2 + k\pi$

A first model: review of nonlinearities

$$
\left\{
\begin{array}{rcl}
\dot{s}_i & = & -\dfrac{k_m^2}{J_r R} s_i - \dfrac{k_{gearbox} c_D}{J_r} |s_i| \, s_i + \dfrac{k_m}{J_r R} \operatorname{sat}_{\bar{U}_i}(U_i) \\
\dot{\vec{p}} & = & \vec{v} \\
m\dot{\vec{v}} & = & -mg\vec{e}_3 + R \begin{pmatrix} 0 \\ 0 \\ \displaystyle\sum_i F_i(s_i) \end{pmatrix} \\
\dot{R} & = & R\vec{\omega}^{\times} \\
J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times} J\vec{\omega} + \begin{pmatrix} \Gamma_r(s_2, s_4) \\ \Gamma_p(s_1, s_3) \\ \Gamma_y(s_1, s_2, s_3, s_4) \end{pmatrix}
\end{array}
\right.
$$

A first model: review of nonlinearities

$$\begin{cases} \dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{k_{gearbox} c_D}{J_r} |s_i| \, s_i + \frac{k_m}{J_r R} \operatorname{sat}_{\bar{U}_i}(U_i) \\ \dot{\vec{p}} = \vec{v} \\ m\dot{\vec{v}} = -mg\vec{e}_3 + R \begin{pmatrix} 0 \\ 0 \\ \sum_i F_i(s_i) \end{pmatrix} \\ \dot{R} = R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} = -\vec{\omega}^{\times} J\vec{\omega} + \begin{pmatrix} \Gamma_r(s_2, s_4) \\ \Gamma_p(s_1, s_3) \\ \Gamma_y(s_1, s_2, s_3, s_4) \end{pmatrix} \end{cases}$$

In red: nonlinearities

In blue: where the control variables act

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

- **Mechanical parameters**:

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

- **Mechanical parameters**:

  l length of an arm of the helicopter, easy to measure

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

- **Mechanical parameters**:

l length of an arm of the helicopter, easy to measure

m total mass of the helicopter, easy to measure

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

- **Mechanical parameters**:

l length of an arm of the helicopter, easy to measure

m total mass of the helicopter, easy to measure

J body inertia, hard to have precisely

Parameter identification

- **Electrical motor**:
  - For small input steps, the system behaves very close to a **linear** first order system
  - Hence, use linear identification tools
  - $\bar{U}_i$ is found on the data-sheet of the motor (damage avoidance)
- **Aerodynamical parameters**: b and c_D

b and c_D measured with specific test beds, depends upon temperature, distance from ground, etc.

- **Mechanical parameters**:

l length of an arm of the helicopter, easy to measure

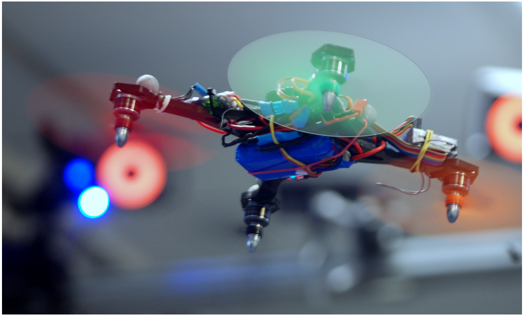
m total mass of the helicopter, easy to measure

J body inertia, hard to have precisely

I_r rotor inertia, hard to have precisely

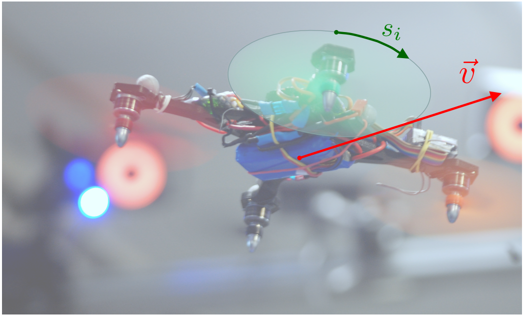
The flapping effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, that is colinear to $\vec{t}_3$
- It has been proved to be false because it neglects the effect of the apparent wind speed, this is the **flapping effect**



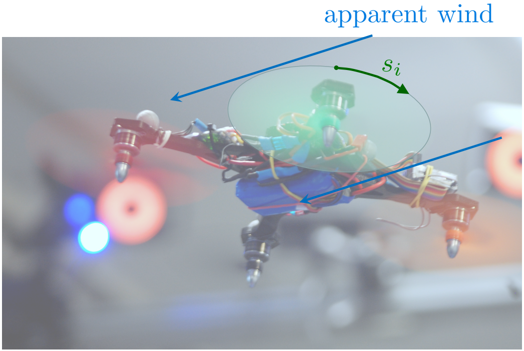
The flapping effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, that is colinear to $\vec{t}_3$
- It has been proved to be false because it neglects the effect of the apparent wind speed, this is the **flapping effect**



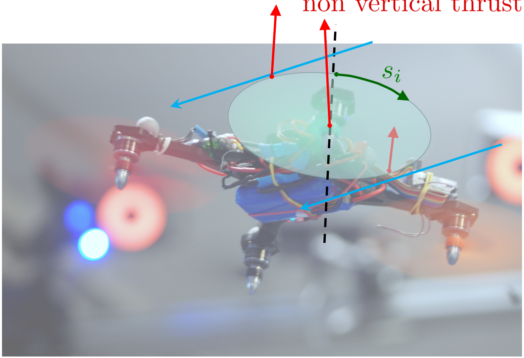
The flapping effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, that is colinear to $\vec{t}_3$
- It has been proved to be false because it neglects the effect of the apparent wind speed, this is the **flapping effect**



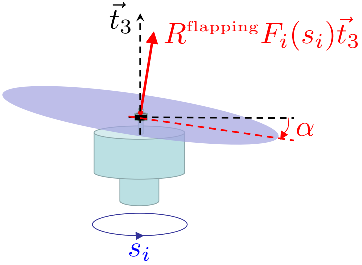
The flapping effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, that is colinear to $\vec{t}_3$
- It has been proved to be false because it neglects the effect of the apparent wind speed, this is the **flapping effect**
- Higher thrust on one side of the blades
- The thrust becomes $\sum_i R_i^{\text{flapping}} F_i(s_i)\vec{t}_3$, torques are also modified



The flapping effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, that is colinear to $\vec{t}_3$
- It has been proved to be false because it neglects the effect of the apparent wind speed, this is the **flapping effect**
- Higher thrust on one side of the blades
- The thrust becomes $\sum_i R_i^{\text{flapping}} F_i(s_i)\vec{t}_3$, torques are also modified



Modeling more into details: the flapping effect

- The flapping matrix takes can be decomposed :

$$
\begin{aligned}
R^{\text{flapping}} &= R_x^{\text{flapping}} \cdot R_y^{\text{flapping}} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c(\beta) & -s(\beta) \\ 0 & s(\beta) & c(\beta) \end{pmatrix} \cdot \begin{pmatrix} c(\alpha) & 0 & s(\alpha) \\ 0 & 1 & 0 \\ -s(\alpha) & 0 & c(\alpha) \end{pmatrix}
\end{aligned}
$$

Modeling more into details: the flapping effect

- The flapping matrix takes can be decomposed :

$$\begin{aligned} R^{\text{flapping}} &= R_x^{\text{flapping}} \cdot R_y^{\text{flapping}} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c(\beta) & -s(\beta) \\ 0 & s(\beta) & c(\beta) \end{pmatrix} \cdot \begin{pmatrix} c(\alpha) & 0 & s(\alpha) \\ 0 & 1 & 0 \\ -s(\alpha) & 0 & c(\alpha) \end{pmatrix} \end{aligned}$$

- α and β can be composed as follows :

$$\begin{aligned} \alpha &= \alpha_v + \alpha_\omega \\ \beta &= \beta_v + \beta_\omega \end{aligned}$$

Modeling more into details: the flapping effect

- The flapping matrix takes can be decomposed :

$$
\begin{aligned}
R^{\text{flapping}} &= R_x^{\text{flapping}} \cdot R_y^{\text{flapping}} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c(\beta) & -s(\beta) \\ 0 & s(\beta) & c(\beta) \end{pmatrix} \cdot \begin{pmatrix} c(\alpha) & 0 & s(\alpha) \\ 0 & 1 & 0 \\ -s(\alpha) & 0 & c(\alpha) \end{pmatrix}
\end{aligned}
$$

- α and β can be composed as follows :

$$
\begin{aligned}
\alpha &= \alpha_v + \alpha_\omega \\
\beta &= \beta_v + \beta_\omega
\end{aligned}
$$

- α_v and β_v represent the contribution of the linear speed of the body to the flapping effect

Modeling more into details: the flapping effect

- The flapping matrix takes can be decomposed :

$$\begin{aligned} R^{\text{flapping}} &= R_x^{\text{flapping}} \cdot R_y^{\text{flapping}} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & c(\beta) & -s(\beta) \\ 0 & s(\beta) & c(\beta) \end{pmatrix} \cdot \begin{pmatrix} c(\alpha) & 0 & s(\alpha) \\ 0 & 1 & 0 \\ -s(\alpha) & 0 & c(\alpha) \end{pmatrix} \end{aligned}$$

- α and β can be composed as follows :

$$\begin{aligned} \alpha &= \alpha_v + \alpha_\omega \\ \beta &= \beta_v + \beta_\omega \end{aligned}$$

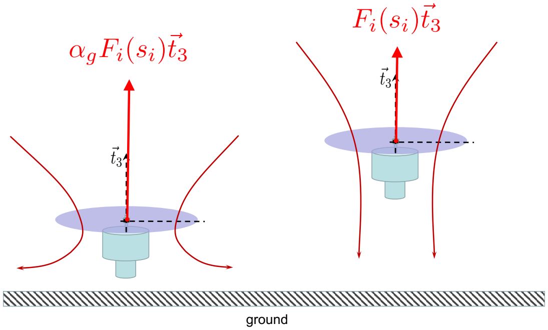
- α_v and β_v represent the contribution of the linear speed of the body to the flapping effect
- a_ω and b_ω represent the contribution of the rotational speed of the body to the flapping effect

The ground effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, with $F_i(s_i) = c_T s_i^2$

The ground effect

- The thrust was assumed to be $\sum_i F_i(s_i)\vec{t}_3$, with $F_i(s_i) = c_T s_i^2$
- Unfortunately, c_T is not constant but depends upon
  - the density of the air, therefore of the temperature
  - the ground distance : it is the ground effect, $\alpha_g(z) \geq 1$



Rotors effects

- Each rotor may be thought of as a rigid disc rotating around the vertical axis the body frame, with angular velocity s_i. The rotor's axis of rotation is itself moving with the angular velocity of the frame. This leads to the following **gyroscopic torque** :

$$\vec{\Gamma}_{\text{gyro}} = I_r \vec{\omega} \times \vec{t}_3 \sum_i (-1) i \, |s_i|$$

  - I_r is the inertia matrix of a rotor

Rotors effects

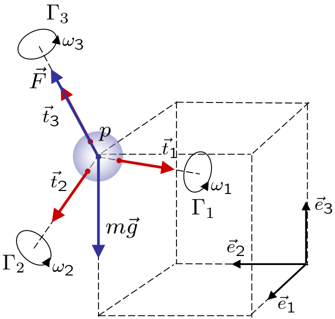
- Each rotor may be thought of as a rigid disc rotating around the vertical axis the body frame, with angular velocity s_i. The rotor's axis of rotation is itself moving with the angular velocity of the frame. This leads to the following **gyroscopic torque** :

$$\vec{\Gamma}_{\text{gyro}} = I_r \vec{\omega} \times \vec{t}_3 \sum_i (-1) i \, |s_i|$$

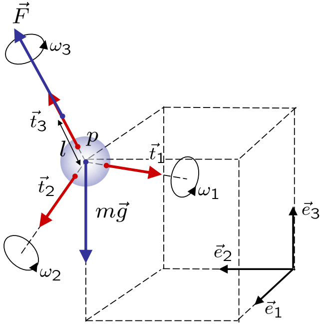
  - I_r is the inertia matrix of a rotor
- Each rotor produces a counter rotating torque that can be expressed as:

$$\begin{aligned} s_{res} &:= \sum_i (-1)^i \, |s_i| \\ \vec{\Gamma}_I &= I_r \dot{s}_{res} \vec{t}_3 \end{aligned}$$

Other effects



Other effects



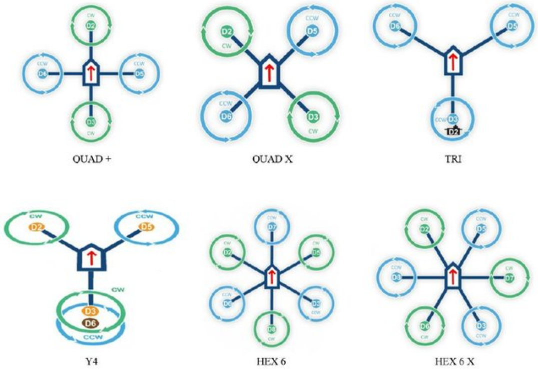
- Superposition of thrust center and mass center
- External forces
- Air friction: $-K_v \, ||\vec{v}|| \, \vec{v}$

The mixing matrix

- The **mixing matrix** M_x links the torques and thrust force to the rotational speed of the rotors

The mixing matrix

- The **mixing matrix** M_x links the torques and thrust force to the rotational speed of the rotors
- Depends on the considered configuration (not the same for $+$ or x configuration)



The mixing matrix

- The **mixing matrix** M_x links the torques and thrust force to the rotational speed of the rotors
- Depends on the considered configuration (not the same for + or x configuration)
- For the + configuration presented before, we have:

$$\begin{pmatrix} T \\ \Gamma_r \\ \Gamma_p \\ \Gamma_y \end{pmatrix} = \underbrace{\begin{pmatrix} c_T & c_T & c_T & c_T \\ 0 & -lc_T & 0 & lc_T \\ lc_T & 0 & -lc_T & 0 \\ lc_D & -lc_D & lc_D & -lc_D \end{pmatrix}}_{M_x} \begin{pmatrix} s_1^2 \\ s_2^2 \\ s_3^2 \\ s_4^2 \end{pmatrix}$$

The mixing matrix

- The **mixing matrix** M_x links the torques and thrust force to the rotational speed of the rotors
- Depends on the considered configuration (not the same for + or x configuration)
- For the + configuration presented before, we have:

$$\begin{pmatrix} T \\ \Gamma_r \\ \Gamma_p \\ \Gamma_y \end{pmatrix} = \underbrace{\begin{pmatrix} c_T & c_T & c_T & c_T \\ 0 & -lc_T & 0 & lc_T \\ lc_T & 0 & -lc_T & 0 \\ lc_D & -lc_D & lc_D & -lc_D \end{pmatrix}}_{M_x} \begin{pmatrix} s_1^2 \\ s_2^2 \\ s_3^2 \\ s_4^2 \end{pmatrix}$$

- Flapping and other effect renders the relation between the rotor's speeds and control thrust and torques complex

The mixing matrix

- The **mixing matrix** M_x links the torques and thrust force to the rotational speed of the rotors
- Depends on the considered configuration (not the same for + or x configuration)
- For the + configuration presented before, we have:

$$\begin{pmatrix} T \\ \Gamma_r \\ \Gamma_p \\ \Gamma_y \end{pmatrix} = \underbrace{\begin{pmatrix} c_T & c_T & c_T & c_T \\ 0 & -lc_T & 0 & lc_T \\ lc_T & 0 & -lc_T & 0 \\ lc_D & -lc_D & lc_D & -lc_D \end{pmatrix}}_{M_x} \begin{pmatrix} s_1^2 \\ s_2^2 \\ s_3^2 \\ s_4^2 \end{pmatrix}$$

- Flapping and other effect renders the relation between the rotor's speeds and control thrust and torques complex
- With flapping appears coupling phenomenon: the thrust affects the yaw movement and the drag affects thrust/roll/pitch movements

Complete model

- **Actuation**: depends upon the type of electrical drive you use

Complete model

- **Actuation**: depends upon the type of electrical drive you use
- **Body**:

$$\begin{cases} \dot{\vec{p}} &= \vec{v} \\ m\dot{\vec{v}} &= -mg\vec{e}_3 - K_v \|\vec{v}\| \vec{v} + R\vec{T} + \vec{F}_{ext} \\ \dot{R} &= R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} &= -\vec{\omega}^{\times} J\vec{\omega} + I_r \dot{s}_{res} \vec{t}_3 + I_r \vec{\omega} \times \vec{t}_3 \sum_i (-1)i\,|s_i| + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases}$$

Complete model

- **Actuation**: depends upon the type of electrical drive you use
- **Body**:

$$\begin{cases} \dot{\vec{p}} &= \vec{v} \\ m\dot{\vec{v}} &= -mg\,\vec{e}_3 - K_v\,||\vec{v}||\,\vec{v} + R\,\vec{T} + \vec{F}_{ext} \\ \dot{R} &= R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} &= -\vec{\omega}^{\times} J\vec{\omega} + I_r \dot{s}_{res}\vec{t}_3 + I_r\vec{\omega} \times \vec{t}_3 \sum_i (-1)i\,|s_i| + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases}$$

- **Thrust**:

$$\vec{T} = \sum_i R_i^{\text{flapping}} \alpha_g c_T s_i^2 \vec{t}_3$$

Complete model

Robotics

N. Marchand

Introduction

Modeling

Cartesian coordinates

Orientation

Frames

Newton

X4

Kinematics

Arm robots

Inner-loop

Geometrical model

Kinematic model

Dynamical model

Conclusion

Path planning

Workspace and obstacles

path planning

Mobile robotics

Visual servoing

- **Actuation**: depends upon the type of electrical drive you use
- **Body**:

$$\begin{cases} \dot{\vec{p}} &= \vec{v} \\ m\dot{\vec{v}} &= -mg\,\vec{e}_3 - K_v\,||\vec{v}||\,\vec{v} + R\,\vec{T} + \vec{F}_{ext} \\ \dot{R} &= R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} &= -\vec{\omega}^{\times} J\vec{\omega} + I_r \dot{s}_{res}\vec{t}_3 + I_r\vec{\omega} \times \vec{t}_3 \sum_i (-1)i\,|s_i| + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases}$$

- **Thrust**:

$$\vec{T} = \sum_i R_i^{\text{flapping}} \alpha_g c_T s_i^2 \vec{t}_3$$

- **Torques**:

$$\vec{\Gamma}_c = \sum_i R_i^{\text{flapping}} \alpha_g c_T s_i^2 \vec{t}_3 \times p_{rotor_i}^{\mathcal{T}} + \sum_i (-1)^{i+1} c_D s_i^2 \vec{t}_3$$

Outline

Jointed-arm robots

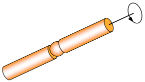
- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:

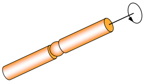
Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm

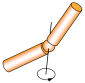


Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm

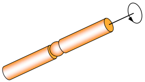


  - rotary perpendicular to the arm

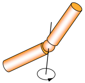


Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



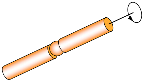
  - rotary perpendicular to the arm



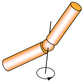
- Each possible movement is called a **degree of freedom (dof)**

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



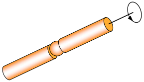
  - rotary perpendicular to the arm



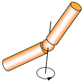
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



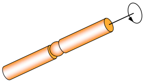
  - rotary perpendicular to the arm



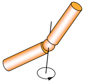
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A “universal” robot has 12 dof:

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



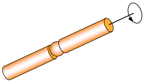
  - rotary perpendicular to the arm



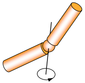
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A "universal" robot has 12 dof:
  - 6 for spatial position (vehicle)

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



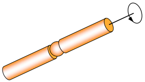
  - rotary perpendicular to the arm



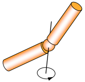
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A “universal” robot has 12 dof:
  - 6 for spatial position (vehicle)
  - 3 for the arm

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



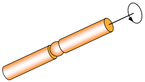
  - rotary perpendicular to the arm



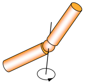
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A "universal" robot has 12 dof:
  - 6 for spatial position (vehicle)
  - 3 for the arm
  - 3 for the terminal tool

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



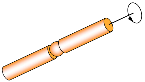
  - rotary perpendicular to the arm



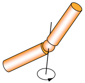
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A "universal" robot has 12 dof:
  - 6 for spatial position (vehicle)
  - 3 for the arm
  - 3 for the terminal tool
- In the industrial context, a polyvalent robot will have 6 dof

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



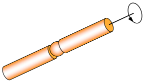
  - rotary perpendicular to the arm



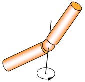
- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A "universal" robot has 12 dof:
  - 6 for spatial position (vehicle)
  - 3 for the arm
  - 3 for the terminal tool
- In the industrial context, a polyvalent robot will have 6 dof
- 6 dof are sufficient for any position and orientation of the terminal tool in the *reachable space*

Jointed-arm robots

- Jointed-arm robot: A robot whose arm is constructed of rigid members connected by rotary joints
- Two possible rotary joints:
  - rotary around the arm



  - rotary perpendicular to the arm



- Each possible movement is called a **degree of freedom (dof)**
- Sometimes movements are coupled (more than 1 dof/articulation)
- A "universal" robot has 12 dof:
  - 6 for spatial position (vehicle)
  - 3 for the arm
  - 3 for the terminal tool
- In the industrial context, a polyvalent robot will have 6 dof
- 6 dof are sufficient for any position and orientation of the terminal tool in the *reachable space*
- Many tasks can be performed with less than 6 dof: "pick and place" needs only 4 dof

Jointed-arm robots

- Characteristic variables:

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i
  - Actuator torques C_i of the joint i

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i
  - Actuator torques C_i of the joint i
  - Angles θ_i of the joint

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i
  - Actuator torques C_i of the joint i
  - Angles θ_i of the joint
  - Spatial position X_i of the extremity of the joint

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i
  - Actuator torques C_i of the joint i
  - Angles θ_i of the joint
  - Spatial position X_i of the extremity of the joint
- Controlling a robot is equivalent to mastering the relation

$$u_i \rightleftarrows C_i \rightleftarrows \theta_i \rightleftarrows X_i$$

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i
  - Actuator torques C_i of the joint i
  - Angles θ_i of the joint
  - Spatial position X_i of the extremity of the joint
- Controlling a robot is equivalent to mastering the relation

$$u_i \rightleftarrows C_i \rightleftarrows \theta_i \rightleftarrows X_i$$

  - Actuator's dynamics

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i of the joint i
  - Actuator torques C_i of the joint i
  - Angles θ_i of the joint
  - Spatial position X_i of the extremity of the joint
- Controlling a robot is equivalent to mastering the relation

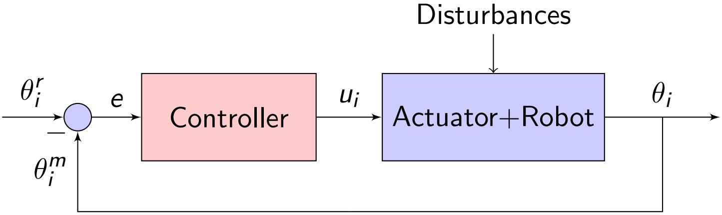
$$u_i \rightleftarrows C_i \rightleftarrows \theta_i \rightleftarrows X_i$$

- Actuator's dynamics
- Robot's dynamics

Outline

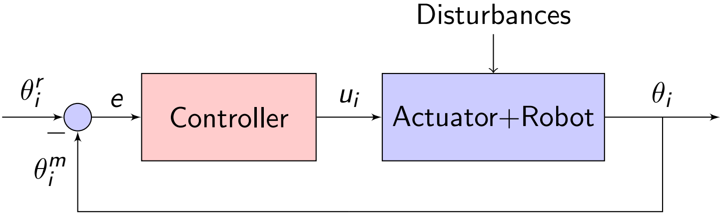
Inner control loop

- Inner control loop:



Inner control loop

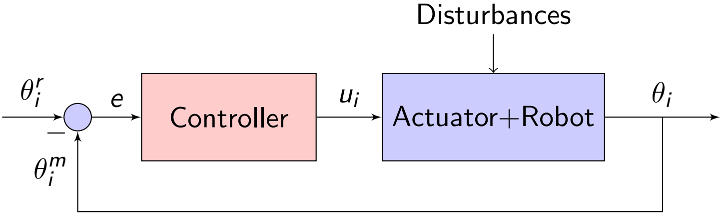
- Inner control loop:



- Enables to force θ to follow the reference θ_r

Inner control loop

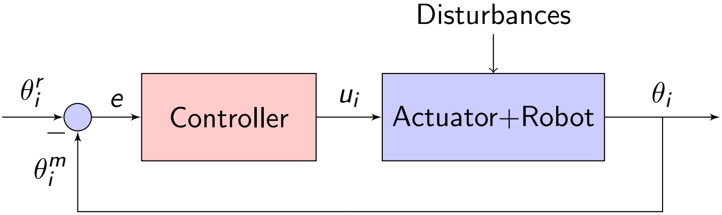
- Inner control loop:



- Enables to force θ to follow the reference θ_r
- The actuator is usually a first (electric) or second order system (pneumatic)

Inner control loop

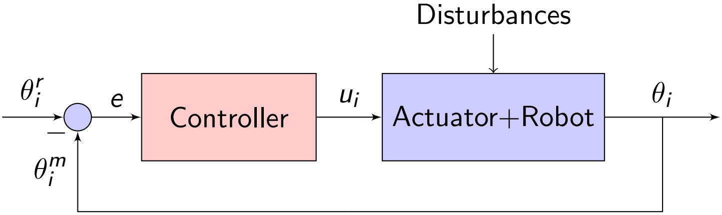
- Inner control loop:



- Enables to force θ to follow the reference θ_r
- The actuator is usually a first (electric) or second order system (pneumatic)
- Usually controlled with a PID controller with

Inner control loop

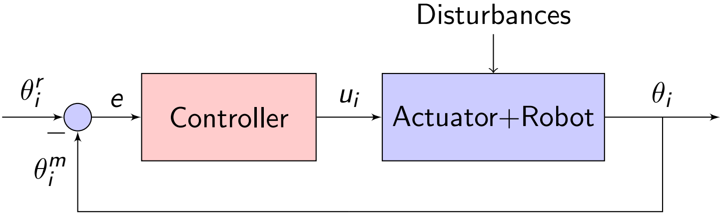
- Inner control loop:



- Enables to force θ to follow the reference θ_r
- The actuator is usually a first (electric) or second order system (pneumatic)
- Usually controlled with a PID controller with
  - filtered derivative action

Inner control loop

- Inner control loop:



- Enables to force θ to follow the reference θ_r
- The actuator is usually a first (electric) or second order system (pneumatic)
- Usually controlled with a PID controller with
  - filtered derivative action
  - anti-windup to tackle saturations

Inner control loop

Anti-windup PID

- We go back to the **X4** example and focus on the **rotors**:

$$\begin{cases} \dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r} \tau_{\text{load}} + \frac{k_m}{J_r R} \operatorname{sat}_{\bar{U}_i}(U_i) \end{cases}$$

Inner control loop

Anti-windup PID

- We go back to the **X4** example and focus on the **rotors**:

$$\left\{ \dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\mathrm{sat}_{\bar{U}_i}(U_i) \right.$$

- If one wants to **act on the X4 with desired forces** F_i^d, it is necessary to be able to **set the rotors speeds** s_i **to** s_i^d with

$$s_i^d = \sqrt{\frac{1}{b} F_i^d}$$

Inner control loop

Anti-windup PID

- We go back to the **X4** example and focus on the **rotors**:

$$\begin{cases} \dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\mathrm{sat}_{\bar{U}_i}(U_i) \end{cases}$$

- If one wants to **act on the X4 with desired forces** F_i^d, it is necessary to be able to **set the rotors speeds** s_i **to** s_i^d with

$$s_i^d = \sqrt{\frac{1}{b}F_i^d}$$

- A usual way to control the electrical motor consist in

Inner control loop

Anti-windup PID

- We go back to the **X4** example and focus on the **rotors**:

$$\left\{ \dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\mathrm{sat}_{\bar{U}_i}(U_i) \right.$$

- If one wants to **act on the X4 with desired forces F_i^d**, it is necessary to be able to **set the rotors speeds s_i to s_i^d** with

$$s_i^d = \sqrt{\frac{1}{b} F_i^d}$$

- A usual way to control the electrical motor consist in
  - taking τ_{load} **as un unknown load**

Inner control loop

Anti-windup PID

- We go back to the **X4** example and focus on the **rotors**:

$$\left\{ \dot{s}_i = -\frac{k_m^2}{J_r R} s_i - \frac{1}{J_r}\tau_{\text{load}} + \frac{k_m}{J_r R}\,\mathrm{sat}_{\bar{U}_i}(U_i) \right.$$

- If one wants to **act on the X4 with desired forces** F_i^d, it is necessary to be able to **set the rotors speeds s_i to s_i^d** with

$$s_i^d = \sqrt{\frac{1}{b} F_i^d}$$

- A usual way to control the electrical motor consist in
  - taking τ_{load} **as un unknown load**
  - **neglecting the voltage limitations** $\bar{U}_i$

Inner control loop

Anti-windup PID

- The so obtained **system is linear**

$$\frac{s_i(s)}{U_i(s)} = \frac{\frac{1}{k_m}}{1+\frac{J_r R}{k_m^2}s} = \frac{G}{1+\tau s}$$

Inner control loop

Anti-windup PID

- The so obtained **system is linear**

$$\frac{s_i(s)}{U_i(s)} = \frac{\frac{1}{k_m}}{1+\frac{J_r R}{k_m^2}s} = \frac{G}{1+\tau s}$$

- Define a **PI controller** for it:

$$C(s) = K_p + \frac{K_i}{s}$$

Inner control loop

Anti-windup PID

- The so obtained **system is linear**

$$\frac{s_i(s)}{U_i(s)} = \frac{\frac{1}{k_m}}{1 + \frac{J_r R}{k_m^2} s} = \frac{G}{1 + \tau s}$$

- Define a **PI controller** for it:

$$C(s) = K_p + \frac{K_i}{s}$$

- Taking $K_i = \dfrac{1}{\tau_{CL} G}$ and $K_p = \tau K_i$, the closed loop system is:

$$\frac{s_i(s)}{U_i(s)} = \frac{1}{1 + \tau_{CL} s}$$

Inner control loop

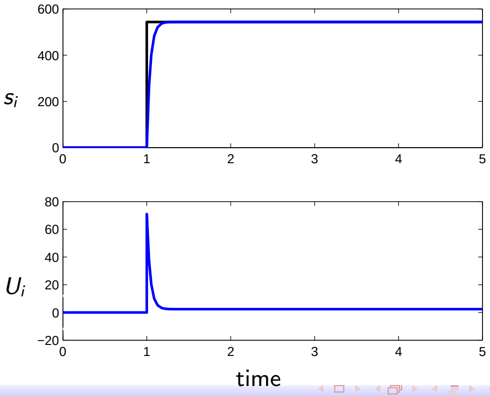
Anti-windup PID

- Make a step that **compensates the weight**, that is such that $s_i^d = \sqrt{\frac{mg}{4b}}$ so that $\sum_i F_i^d = mg$

Inner control loop

Anti-windup PID

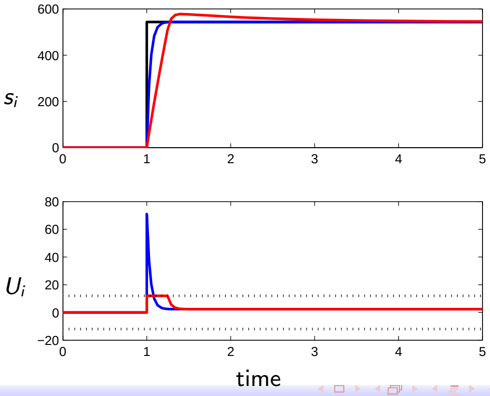
- Make a step that **compensates the weight**, that is such that $s_i^d = \sqrt{\dfrac{mg}{4b}}$ so that $\sum_i F_i^d = mg$
- Taking $\tau_{CL} = 50$ **ms**, one gets **without saturations**



Inner control loop

Anti-windup PID

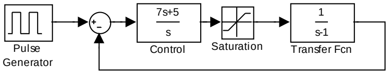
- Make a step that **compensates the weight**, that is such that $s_i^d = \sqrt{\dfrac{mg}{4b}}$ so that $\sum_i F_i^d = mg$
- Taking $\tau_{CL} = 50$ **ms**, one gets **with saturations**



Inner control loop

Anti-windup PID

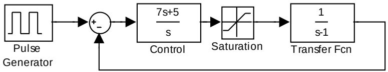
- The result could be worse:



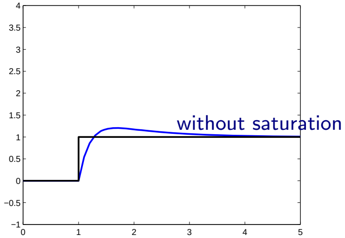
Inner control loop

Anti-windup PID

- The result could be worse:



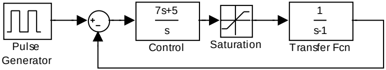
- For $u \in [-1.2, 1.2]$, the closed-loop behavior is:



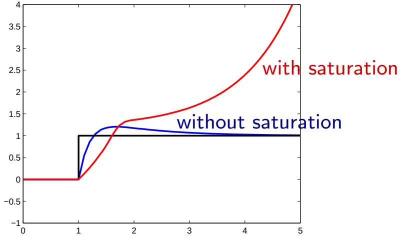
Inner control loop

Anti-windup PID

- The result could be worse:



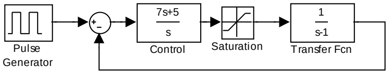
- For $u \in [-1.2, 1.2]$, the closed-loop behavior is:



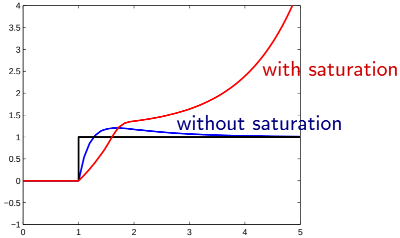
Inner control loop

Anti-windup PID

- The result could be worse:



- For $u \in [-1.2, 1.2]$, the closed-loop behavior is:

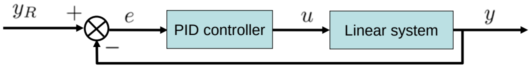


- **Saturations may lead to instability** especially in the presence of integrators in the loop

Inner control loop

Anti-windup PID

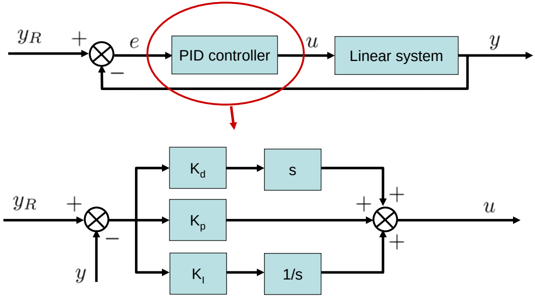
- Consider a **linear system with a PID** controller:



Inner control loop

Anti-windup PID

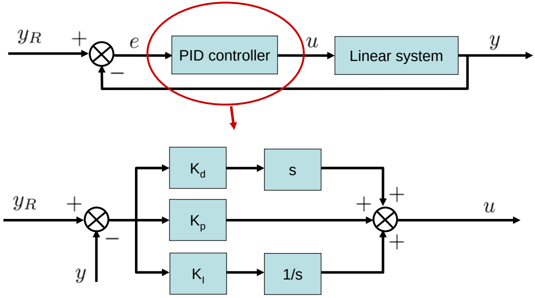
- Consider a **linear system with a PID** controller:



Inner control loop

Anti-windup PID

- Consider a **linear system with a PID** controller:

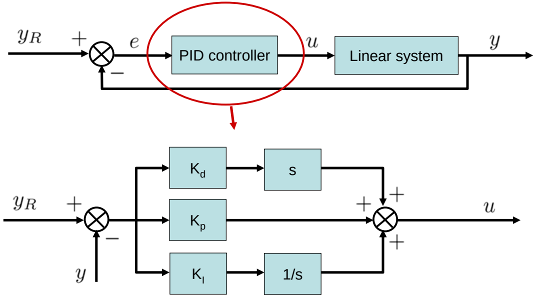


- The instability comes from the **integration** of the error

Inner control loop

Anti-windup PID

- Consider a **linear system with a PID** controller:

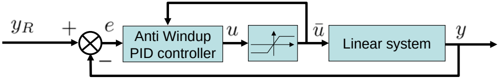


- The instability comes from the **integration** of the error
- **Key idea:** soften the integral effect when the control is saturated

Inner control loop

Anti-windup PID

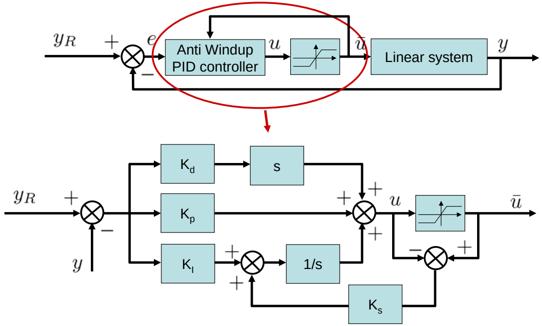
- Structure of the **PID controller with anti-windup**:



Inner control loop

Anti-windup PID

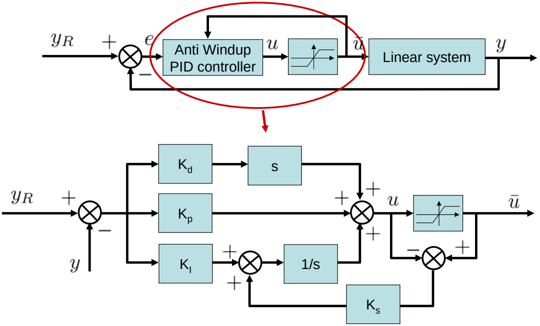
- Structure of the **PID controller with anti-windup**:



Inner control loop

Anti-windup PID

- Structure of the **PID controller with anti-windup**:

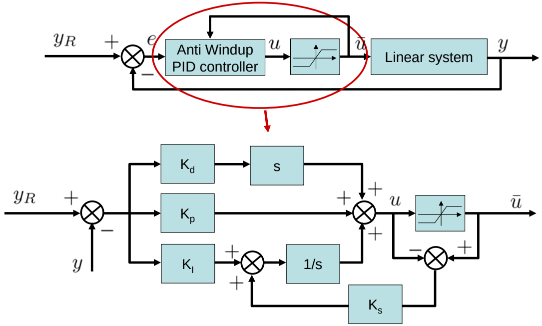


- If $u = \bar{u}$, that is if u is not saturated, **then the PID controller with anti-windup is identical to the classical PID controller**

Inner control loop

Anti-windup PID

- Structure of the **PID controller with anti-windup**:



- If $u = \bar{u}$, that is if u is not saturated, **then the PID controller with anti-windup is identical to the classical PID controller**
- **If u is saturated** $(u \neq \bar{u})$, K_s **tunes the reduction of the integral effect** of the PID

Inner control loop

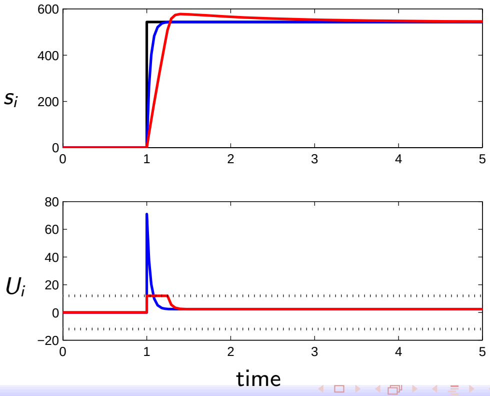
Anti-windup PID

- Make a step that **compensates the weight**, that is such that $s_i^d = \sqrt{\frac{mg}{4b}}$ so that $\sum_i F_i^d = mg$

Inner control loop

Anti-windup PID

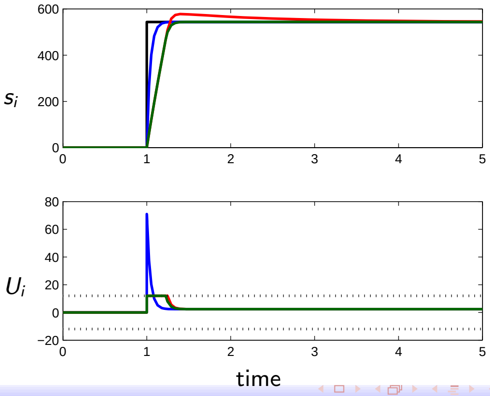
- Make a step that **compensates the weight**, that is such that $s_i^d = \sqrt{\frac{mg}{4b}}$ so that $\sum_i F_i^d = mg$
- Taking $\tau_{CL} = 50$ **ms**, one gets **without anti-windup**



Inner control loop

Anti-windup PID

- Make a step that **compensates the weight**, that is such that $s_i^d = \sqrt{\frac{mg}{4b}}$ so that $\sum_i F_i^d = mg$
- Taking $\tau_{CL} = 50$ **ms**, one gets **with anti-windup**



Inner control loop

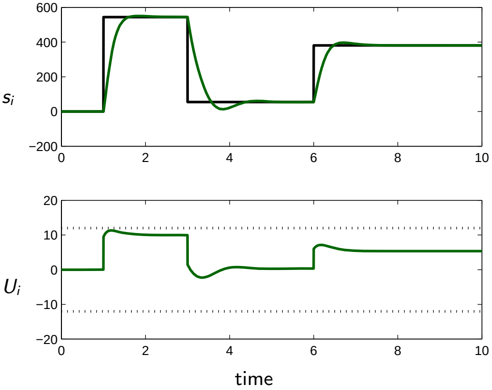
Towards gain scheduling

- Take again $\tau_{CL} = 50$ **ms** and a **PI controller tuned at** s_i^d

Inner control loop

Towards gain scheduling

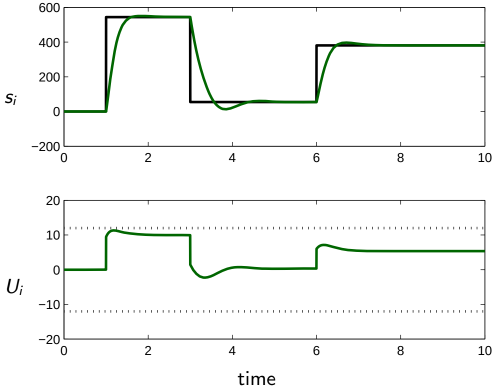
- Take again $\tau_{CL} = 50$ **ms** and a **PI controller tuned at** s_i^d
- Make speed steps of different level



Inner control loop

Towards gain scheduling

- Take again $\tau_{CL} = 50$ **ms** and a **PI controller tuned at s_i^d**
- Make speed steps of different level



- The controller is well tuned near s_i^d but **not very good a large range of use**

Inner control loop

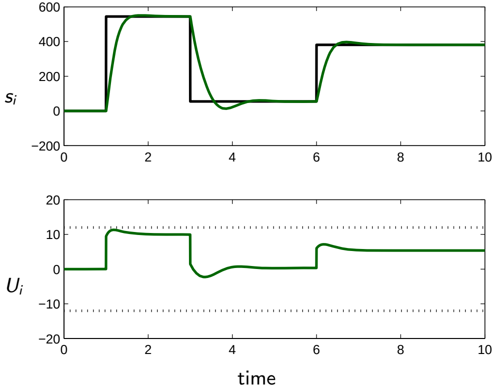
Towards gain scheduling

- Take again $\tau_{CL} = 50$ **ms** and a **PI controller tuned at the current** s_i

Inner control loop

Towards gain scheduling

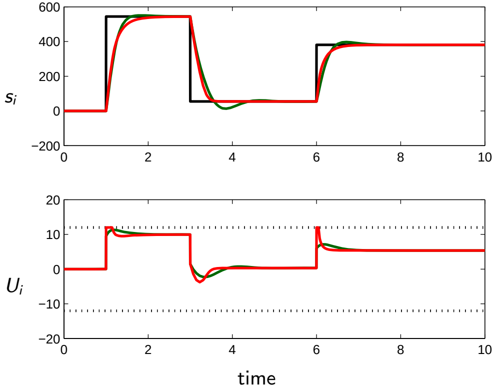
- Take again $\tau_{CL} = 50$ **ms** and a **PI controller tuned at the current s_i**
- Make speed steps of different level



Inner control loop

Towards gain scheduling

- Take again $\tau_{CL} = 50$ **ms** and a **PI controller tuned at the current s_i**
- Make speed steps of different level



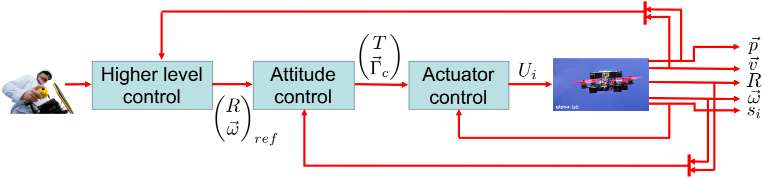
- **The rotors are now well controlled...almost**

Assumptions and principles

- We assume we can measure every thing (thanks to people like Hassen) !

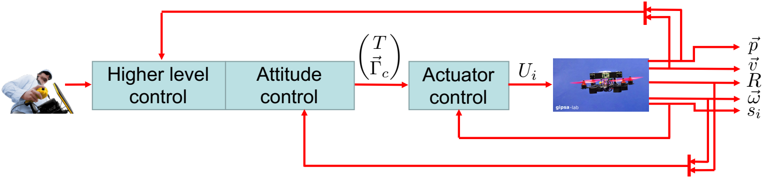
Assumptions and principles

- We assume we can measure every thing (thanks to people like Hassen) !
- Three embedded loops in a control strategy as follows



Assumptions and principles

- We assume we can measure every thing (thanks to people like Hassen) !
- Three embedded loops in a control strategy as follows



Actuator control

- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system

Actuator control

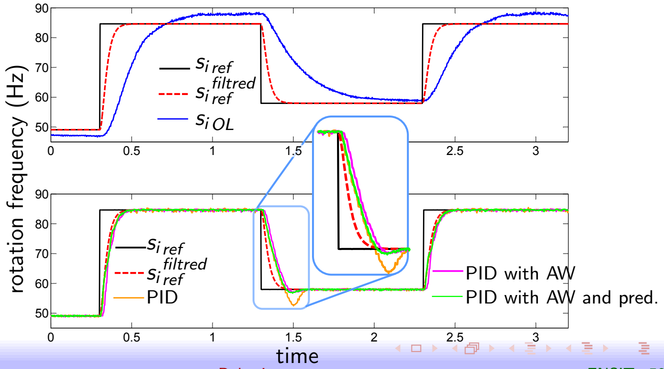
- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system
- A speed measure is required, usually optic or magnetic

Actuator control

- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system
- A speed measure is required, usually optic or magnetic
- Two usual approaches: sliding mode and PI controller with anti-windup

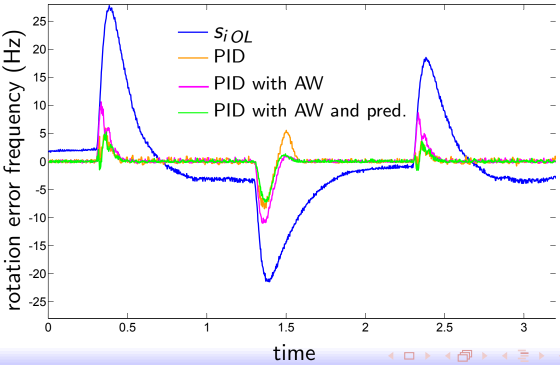
Actuator control

- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system
- A speed measure is required, usually optic or magnetic
- Two usual approaches: sliding mode and PI controller with anti-windup



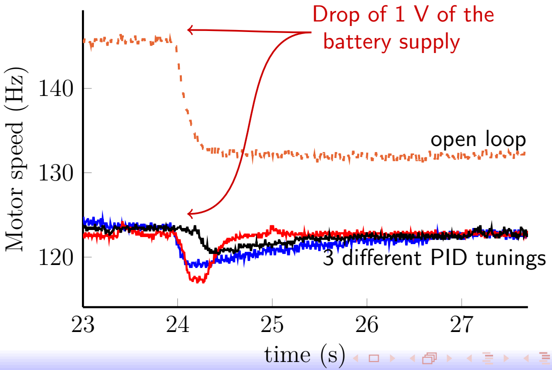
Actuator control

- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system
- A speed measure is required, usually optic or magnetic
- Two usual approaches: sliding mode and PI controller with anti-windup



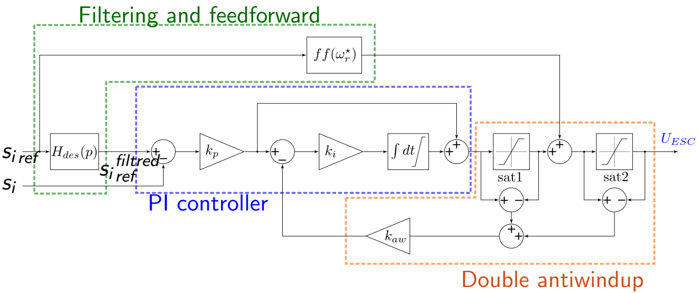
Actuator control

- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system
- A speed measure is required, usually optic or magnetic
- Two usual approaches: sliding mode and PI controller with anti-windup



Actuator control

- Closing the actuator loop will
  - increase the precision of the forces and torques generated
  - allow non identical actuators
  - render the system more reactive
  - face battery drop problem and the non constant gain of the open loop system
- A speed measure is required, usually optic or magnetic
- Two usual approaches: sliding mode and PI controller with anti-windup



Attitude control

- Required by most of the higher level control strategies
- Basic for stable remote piloting
- Embedded in all commercial platforms

Attitude control

- Required by most of the higher level control strategies
- Basic for stable remote piloting
- Embedded in all commercial platforms
- Consist in controlling only the part of the model corresponding to the angular motion:

$$\left\{ \begin{array}{rcl} \dot{R} & = & R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times} J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{array} \right.$$

$$\left\{ \begin{array}{rcl} \dot{\vec{q}} & = & \frac{1}{2}\Omega(\vec{\omega})\vec{q} \\ & = & \frac{1}{2}\Xi(q)\vec{\omega} \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times} J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{array} \right.$$

Attitude control

- Required by most of the higher level control strategies
- Basic for stable remote piloting
- Embedded in all commercial platforms
- Consist in controlling only the part of the model corresponding to the angular motion:

$$\left\{\begin{array}{rcl} \dot{R} & = & R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times}J\vec{\omega}+\vec{\Gamma}_c+\vec{\Gamma}_{ext} \end{array}\right. \qquad \left\{\begin{array}{rcl} \dot{\vec{q}} & = & \frac{1}{2}\Omega(\vec{\omega})\vec{q} \\ & = & \frac{1}{2}\Xi(q)\vec{\omega} \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times}J\vec{\omega}+\vec{\Gamma}_c+\vec{\Gamma}_{ext} \end{array}\right.$$

- Linearization of the rotational dynamics gives three second order integrators
- Most of the applied strategies are PID controllers based on the linearization. Some sliding mode approaches. Few nonlinear approaches.

Attitude control

- Required by most of the higher level control strategies
- Basic for stable remote piloting
- Embedded in all commercial platforms
- Consist in controlling only the part of the model corresponding to the angular motion:

$$\begin{cases} \dot{R} = R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} = -\vec{\omega}^{\times}J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases} \qquad \begin{cases} \dot{\vec{q}} = \frac{1}{2}\Omega(\vec{\omega})\vec{q} \\ \quad = \frac{1}{2}\Xi(q)\vec{\omega} \\ J\dot{\vec{\omega}} = -\vec{\omega}^{\times}J\vec{\omega} + \vec{\Gamma}_c + \vec{\Gamma}_{ext} \end{cases}$$

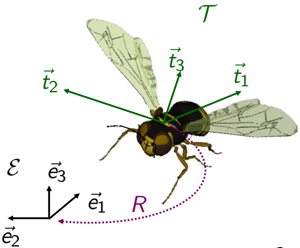
- Linearization of the rotational dynamics gives three second order integrators
- Most of the applied strategies are PID controllers based on the linearization. Some sliding mode approaches. Few nonlinear approaches.
- Valid only *around* zero angles position, but
  - robust
  - easy to tune
  - can handle saturation
  - can be adaptive
  - ...

A saturation based attitude control

Body's dynamics

$$
\begin{aligned}
\dot{\vec{p}} &= \vec{v} \\
\dot{\vec{v}} &= -g\vec{e}_3 + \frac{1}{m}R\vec{f} - c\vec{v} \\
\begin{pmatrix} \dot{q}_0 \\ \dot{q}_v \end{pmatrix} &= \frac{1}{2}\begin{pmatrix} -q_v^T \\ I_3 q_0 - q_v^\times \end{pmatrix}\omega \\
\dot{\omega} &= J^{-1}(\vec{\tau} - \omega \times J\omega)
\end{aligned}
$$

R : Rotation matrix from $\mathcal{T}$ to $\mathcal{E}$



$\mathcal{E}$: Fixed frame

$\mathcal{T}$: Mobile frame

A saturation based attitude control

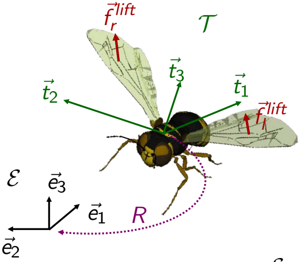
Body's dynamics

$$\dot{\vec{p}} = \vec{v}$$

$$\dot{\vec{v}} = -g\vec{e}_3 + \frac{1}{m}R\vec{f} - c\vec{v}$$

$$\begin{pmatrix}\dot{q}_0 \\ \dot{q}_v\end{pmatrix} = \frac{1}{2}\begin{pmatrix} -q_v^T \\ I_3 q_0 - q_v^{\times}\end{pmatrix}\omega$$

$$\dot{\omega} = J^{-1}(\vec{\tau} - \omega \times J\omega)$$



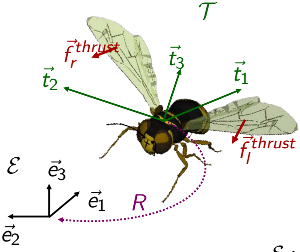
R : Rotation matrix from $\mathcal{T}$ to $\mathcal{E}$
$\vec{f}, \vec{\tau}$: Aerodynamic force and torque in $\mathcal{T}$
$\vec{p}$, $\vec{v}$: Linear position and velocity in $\mathcal{E}$

$\mathcal{E}$: Fixed frame
$\mathcal{T}$: Mobile frame
m : Body's mass
c : Viscosity coefficient
g : Gravity

A saturation based attitude control

Body's dynamics

$$
\begin{aligned}
\dot{\vec{p}} &= \vec{v} \\
\dot{\vec{v}} &= -g\vec{e}_3 + \frac{1}{m} R\vec{f} - c\vec{v} \\
\begin{pmatrix} \dot{q}_0 \\ \dot{q}_v \end{pmatrix} &= \frac{1}{2} \begin{pmatrix} -q_v^T \\ I_3 q_0 - q_v^\times \end{pmatrix} \omega \\
\dot{\omega} &= J^{-1}(\vec{\tau} - \omega \times J\omega)
\end{aligned}
$$



R : Rotation matrix from $\mathcal{T}$ to $\mathcal{E}$

$\vec{f}, \vec{\tau}$: Aerodynamic force and torque in $\mathcal{T}$

$\vec{p}$, $\vec{v}$: Linear position and velocity in $\mathcal{E}$

$q = [q_0 \ q_v]^T$: Quaternion

$q_v^\times$: Skew symmetric matrix associated to q_v

ω : Angular velocity in $\mathcal{T}$

$\mathcal{E}$: Fixed frame

$\mathcal{T}$: Mobile frame

m : Body's mass

c : Viscosity coefficient

g : Gravity

I_3 : Identity matrix

J : Inertia matrix

A saturation based attitude control

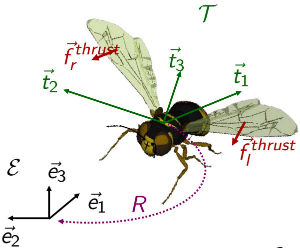
Body's dynamics

$$\dot{\vec{p}} = \vec{v}$$

$$\dot{\vec{v}} = -g\vec{e}_3 + \frac{1}{m}R\vec{f} - c\vec{v}$$

$$\begin{pmatrix}\dot{q}_0 \\ \dot{q}_v\end{pmatrix} = \frac{1}{2}\begin{pmatrix}-q_v^T \\ I_3 q_0 - q_v^\times\end{pmatrix}\omega$$

$$\dot{\omega} = J^{-1}(\underline{\vec{\tau}} - \omega \times J\omega)$$



R : Rotation matrix from $\mathcal{T}$ to $\mathcal{E}$
$\vec{f}, \vec{\tau}$: Aerodynamic force and torque in $\mathcal{T}$
$\vec{p}$, $\vec{v}$: Linear position and velocity in $\mathcal{E}$
$q = [q_0 \ q_v]^T$: Quaternion
$q_v^\times$: Skew symmetric matrix associated to q_v
ω : Angular velocity in $\mathcal{T}$

$\mathcal{E}$: Fixed frame
$\mathcal{T}$: Mobile frame
m : Body's mass
c : Viscosity coefficient
g : Gravity
I_3 : Identity matrix
J : Inertia matrix

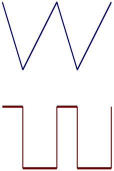
A saturation based attitude control

$\phi^{wing}(\phi_0^{wing}, t)$

$\psi^{wing}(\psi_0^{wing}, t)$

θ^{wing} NUL

A saturation based attitude control



$\phi^{wing}(\phi_0^{wing}, t)$

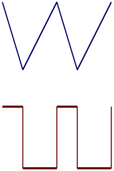
$\psi^{wing}(\psi_0^{wing}, t)$

→ aerodynamic forces and torques

$(\vec{f}, \vec{\tau})$

θ^{wing} NUL

A saturation based attitude control



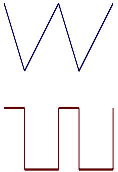
$\phi^{wing}(\phi_0^{wing}, t)$

$\psi^{wing}(\psi_0^{wing}, t)$ → aerodynamic forces and torques

$(\vec{f}, \vec{\tau})$

θ^{wing} NUL

A saturation based attitude control



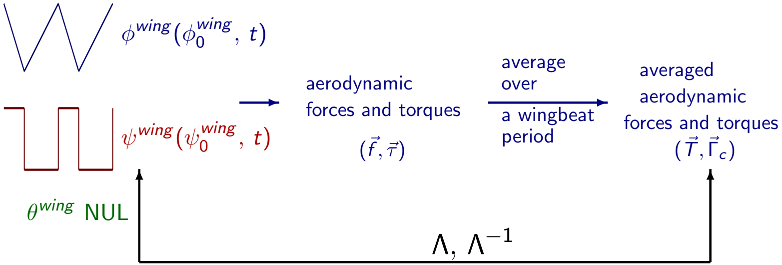
$\phi^{wing}(\phi_0^{wing}, t)$

$\psi^{wing}(\psi_0^{wing}, t)$

θ^{wing} NUL

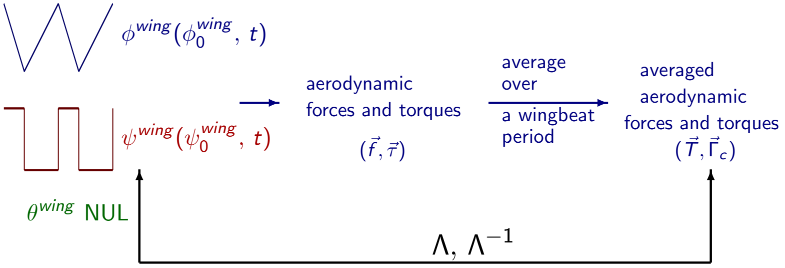
→ aerodynamic forces and torques $(\vec{f}, \vec{\tau})$ —average over a wingbeat period→ averaged aerodynamic forces and torques $(\vec{T}, \vec{\Gamma}_c)$

A saturation based attitude control



$$(\overline{\vec{T}}, \overline{\vec{\Gamma}_c}) = \Lambda(\phi_0^{wing}, \psi_0^{wing})^{d,g}$$

A saturation based attitude control



$$(\overline{\vec{T}}, \overline{\vec{\Gamma}}_c) = \Lambda(\phi_0^{wing}, \psi_0^{wing})^{d,g}$$

Find $\vec{T} = \vec{T}(\overline{p}, \overline{v}, \overline{q}, \overline{\omega})$ and $\vec{\Gamma}_c = \vec{\Gamma}_c(\overline{p}, \overline{v}, \overline{q}, \overline{\omega})$ such that the system has the desired behavior

A saturation based attitude control

Wings angles amplitudes $(\phi_0^{wing}, \psi_0^{wing})^{l,r}$ $\xrightarrow{\Lambda}$ Averaged forces and torques $(\vec{T}, \vec{\Gamma}_c)$

A saturation based attitude control

Wings angles amplitudes $(\phi_0^{wing}, \psi_0^{wing})^{l,r}$ $\xrightarrow{\Lambda}$ Averaged forces and torques $(\vec{T}, \vec{\Gamma}_c)$

↑

Constraints

$$0 \leq \phi_0^{wing} \leq \tilde{\phi}_0^{wing}$$
$$-\tilde{\psi}_0^{wing} \leq \psi_0^{wing} \leq \tilde{\psi}_0^{wing}$$

A saturation based attitude control

Wings angles amplitudes $(\phi_0^{wing}, \psi_0^{wing})^{l,r}$ $\xrightarrow{\Lambda}$ Averaged forces and torques $(\vec{T}, \vec{\Gamma}_c)$

↑ ↕

Constraints $\xrightarrow{?}$ Constraints

$$0 \leq \phi_0^{wing} \leq \tilde{\phi}_0^{wing}$$

$$-\tilde{\psi}_0^{wing} \leq \psi_0^{wing} \leq \tilde{\psi}_0^{wing}$$

Saturations

- Physical limitations: wings angles saturation

$$\begin{array}{rcccl} 0 & \leq & (\phi_0^{wing})^{l,r} & \leq & \tilde{\phi}_0^{wing} \\ -\tilde{\psi}_0^{wing} & \leq & (\psi_0^{wing})^{l,r} & \leq & \tilde{\psi}_0^{wing} \end{array}$$

$$\begin{array}{rcl} \beta x \left[\phi_0^{wing,r^2} \cos \psi_0^d - \phi_0^{wing,l^2} \cos \psi_0^g \right] & = & \Gamma_r^{max} \\ \alpha x \left[\phi_0^{wing,r^2} \sin \psi_0^d - \phi_0^{wing,l^2} \sin \psi_0^g \right] & = & \Gamma_y^{max} \end{array}$$

Saturations

- Physical limitations: wings angles saturation

$$\begin{array}{rcccl} 0 & \leq & (\phi_0^{wing})^{l,r} & \leq & \tilde{\phi}_0^{wing} \\ -\tilde{\psi}_0^{wing} & \leq & (\psi_0^{wing})^{l,r} & \leq & \tilde{\psi}_0^{wing} \end{array}$$

$$\begin{array}{rcl} \beta x\left[\phi_0^{wing,r^2}\cos\psi_0^d - \phi_0^{wing,l^2}\cos\psi_0^g\right] & = & \Gamma_r^{max} \\ \alpha x\left[\phi_0^{wing,r^2}\sin\psi_0^d - \phi_0^{wing,l^2}\sin\psi_0^g\right] & = & \Gamma_y^{max} \end{array}$$

- Coupled saturations of the control torques

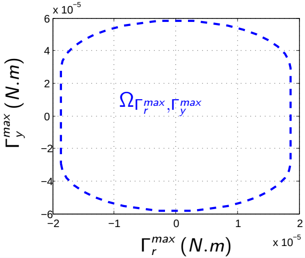
Saturations

- Physical limitations: wings angles saturation

$$\begin{array}{rcccl} 0 & \leq & (\phi_0^{wing})^{l,r} & \leq & \tilde{\phi}_0^{wing} \\ -\tilde{\psi}_0^{wing} & \leq & (\psi_0^{wing})^{l,r} & \leq & \tilde{\psi}_0^{wing} \end{array}$$

$$\begin{array}{rcl} \beta x\left[\phi_0^{wing,r^2}\cos\psi_0^d - \phi_0^{wing,l^2}\cos\psi_0^g\right] & = & \Gamma_r^{max} \\ \alpha x\left[\phi_0^{wing,r^2}\sin\psi_0^d - \phi_0^{wing,l^2}\sin\psi_0^g\right] & = & \Gamma_y^{max} \end{array}$$

- Coupled saturations of the control torques



Admissible saturation set

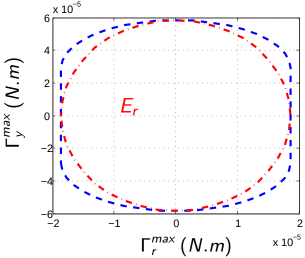
Saturations

- Physical limitations: wings angles saturation

$$\begin{array}{ccccc} 0 & \leq & (\phi_0^{wing})^{l,r} & \leq & \tilde{\phi}_0^{wing} \\ -\tilde{\psi}_0^{wing} & \leq & (\psi_0^{wing})^{l,r} & \leq & \tilde{\psi}_0^{wing} \end{array}$$

$$\begin{array}{ccc} \beta x \left[\phi_0^{wing,r^2} \cos\psi_0^d - \phi_0^{wing,l^2} \cos\psi_0^g\right] & = & \Gamma_r^{max} \\ \alpha x \left[\phi_0^{wing,r^2} \sin\psi_0^d - \phi_0^{wing,l^2} \sin\psi_0^g\right] & = & \Gamma_y^{max} \end{array}$$

- Coupled saturations of the control torques



Admissible saturation set

Reduced to an ellipse in order to simplify the computations

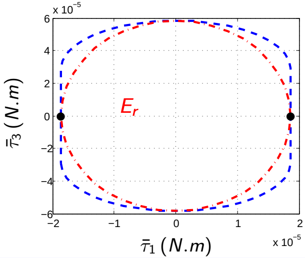
Saturations

- Physical limitations: wings angles saturation

$$
\begin{array}{ccccc}
0 & \leq & (\phi_0^{wing})^{l,r} & \leq & \tilde{\phi}_0^{wing} \\
-\tilde{\psi}_0^{wing} & \leq & (\psi_0^{wing})^{l,r} & \leq & \tilde{\psi}_0^{wing}
\end{array}
$$

$$
\begin{array}{ccc}
\beta x \left[\phi_0^{wing,r^2} \cos \psi_0^d - \phi_0^{wing,l^2} \cos \psi_0^g \right] & = & \Gamma_r^{max} \\
\alpha x \left[\phi_0^{wing,r^2} \sin \psi_0^d - \phi_0^{wing,l^2} \sin \psi_0^g \right] & = & \Gamma_y^{max}
\end{array}
$$

- Coupled saturations of the control torques



Admissible saturation set

Reduced to an ellipse in order to simplify the computations

Nul yaw torque for a maximal roll torque

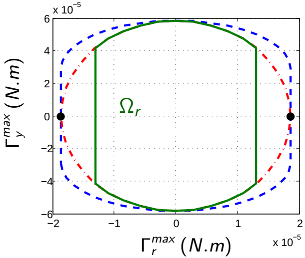
Saturations

- Physical limitations: wings angles saturation

$$\begin{array}{ccccc} 0 & \leq & (\phi_0^{wing})^{l,r} & \leq & \tilde{\phi}_0^{wing} \\ -\tilde{\psi}_0^{wing} & \leq & (\psi_0^{wing})^{l,r} & \leq & \tilde{\psi}_0^{wing} \end{array}$$

$$\begin{array}{rcl} \beta x\left[\phi_0^{wing,r^2}\cos\psi_0^d - \phi_0^{wing,l^2}\cos\psi_0^g\right] & = & \Gamma_r^{max} \\ \alpha x\left[\phi_0^{wing,r^2}\sin\psi_0^d - \phi_0^{wing,l^2}\sin\psi_0^g\right] & = & \Gamma_y^{max} \end{array}$$

- Coupled saturations of the control torques



Admissible saturation set

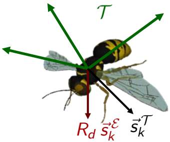
Reduced to an ellipse in order to simplify the computations

Nul yaw torque for a maximal roll torque

Roll stabilization is preferred in order to bring the body to a horizontal position

Attitude stabilization

- Attitude with reference vectors:

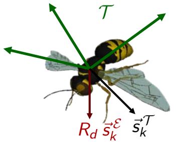


Current vector: $s_k^{\mathcal{T}} = R\, s_k^{\mathcal{E}}$

Desired vector: $s_{k_d}^{\mathcal{T}} = R_d\, s_k^{\mathcal{E}}$

Attitude stabilization

- Attitude with reference vectors:

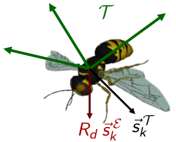


Current vector: $s_k^{\mathcal{T}} = R\, s_k^{\mathcal{E}}$
Desired vector: $s_{k_d}^{\mathcal{T}} = R_d\, s_k^{\mathcal{E}}$

  - Yaw control is impossible
  - Add another sensor giving a non collinear measurement (magnetometer for example)

Attitude stabilization

- Attitude with reference vectors:

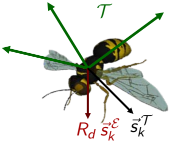


Current vector: $s_k^{\mathcal{T}} = R\,s_k^{\mathcal{E}}$
Desired vector: $s_{k_d}^{\mathcal{T}} = R_d\,s_k^{\mathcal{E}}$

- Yaw control is impossible
- Add another sensor giving a non collinear measurement (magnetometer for example)
- Number of non collinear sensors $n \geq 2$

Attitude stabilization

- Attitude with reference vectors:



Current vector: $s_k^{\mathcal{T}} = R\, s_k^{\mathcal{E}}$

Desired vector: $s_{k_d}^{\mathcal{T}} = R_d\, s_k^{\mathcal{E}}$

- Attitude error:

$$\vec{\zeta} = \frac{\Delta^{-1}}{n} \sum_{k=1}^{n} \vec{s}_k^{\mathcal{T}} \times R_d \vec{s}_k^{\mathcal{E}}$$

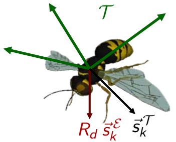
Δ : positive diagonal matrix

n : sensors number

R_d : desired orientation of $\mathcal{T}$ relatively to $\mathcal{E}$)

Attitude stabilization

- Attitude with reference vectors:



Current vector: $s_k^{\mathcal{T}} = R\, s_k^{\mathcal{E}}$
Desired vector: $s_{k_d}^{\mathcal{T}} = R_d\, s_k^{\mathcal{E}}$

- Attitude error:

$$\vec{\zeta} = \frac{\Delta^{-1}}{n} \sum_{k=1}^{n} \vec{s}_k^{\mathcal{T}} \times R_d \vec{s}_k^{\mathcal{E}}$$

Δ : positive diagonal matrix
n : sensors number
R_d : desired orientation of $\mathcal{T}$ relatively to $\mathcal{E}$)

- If $\vec{s}_k^{\mathcal{T}}$ and $R_d\vec{s}_k^{\mathcal{E}}$ are collinear, then $\vec{\zeta} = 0$

A saturation based attitude control

- Sensors used: Rate gyros and reference sensors

A saturation based attitude control

- Sensors used: Rate gyros and reference sensors
- **Control torques**

$$\bar{\tau}_j = -\operatorname{sat}_{\tilde{\tau}_j}(\lambda_j[\bar{\omega}_{G_j} + \rho_j \bar{\zeta}_j)]) \qquad j = \{1,2,3\}$$

λ_j, ρ_j : positive tuning parameters

$\operatorname{sat}_{\tilde{\tau}_j}$: saturation function

$\bar{\omega}_{G_j}$: averaged angular velocity (measured by the rate gyros)

$\bar{\zeta}_j$: averaged attitude error

A saturation based attitude control

- Sensors used: Rate gyros and reference sensors
- **Control torques**

$$\bar{\tau}_j = -\operatorname{sat}_{\tilde{\tau}_j}(\lambda_j[\bar{\omega}_{G_j} + \rho_j \bar{\zeta}_j)]) \qquad j = \{1, 2, 3\}$$

λ_j, ρ_j	:	positive tuning parameters
$\operatorname{sat}_{\tilde{\tau}_j}$	:	saturation function
$\bar{\omega}_{G_j}$	:	averaged angular velocity (measured by the rate gyros)
$\bar{\zeta}_j$	:	averaged attitude error

- Stability proved (rigid body) using Lyapunov function:

$$V = \frac{1}{2}\vec{\omega}^T J \vec{\omega} + \frac{1}{n}\sum_{k=1}^{n}(1 - \vec{s}_k^{\,m\,T} R_d\, \vec{s}_k^{\,f})$$

A saturation based attitude control

- Sensors used: Rate gyros and reference sensors
- **Control torques**

$$\bar{\tau}_j = -\operatorname{sat}_{\tilde{\tau}_j}(\lambda_j[\bar{\omega}_{G_j} + \rho_j\bar{\zeta}_j)]) \qquad j = \{1,2,3\}$$

λ_j, ρ_j	:	positive tuning parameters
$\operatorname{sat}_{\tilde{\tau}_j}$	:	saturation function
$\bar{\omega}_{G_j}$	:	averaged angular velocity (measured by the rate gyros)
$\bar{\zeta}_j$	:	averaged attitude error

- Stability proved (rigid body) using Lyapunov function:

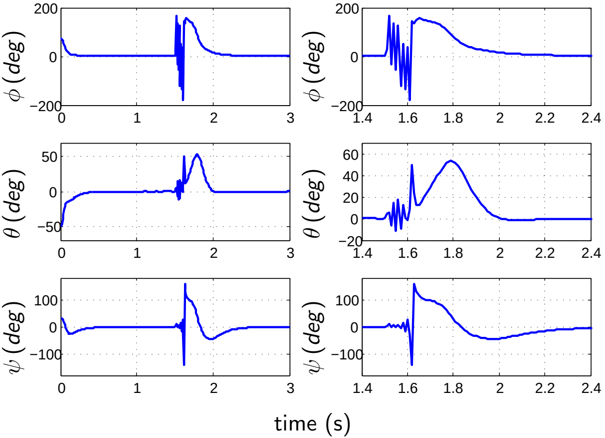
$$V = \frac{1}{2}\vec{\omega}^T J\vec{\omega} + \frac{1}{n}\sum_{k=1}^{n}(1 - \vec{s}_k^{\,m\,T} R_d\, \vec{s}_k^{\,f})$$

- **Generalized PID controller**, *almost* global stability, simpler version using quaternion exists, stability independent from the knowledge of J, robust to velocity sensor saturation.

A saturation based attitude control

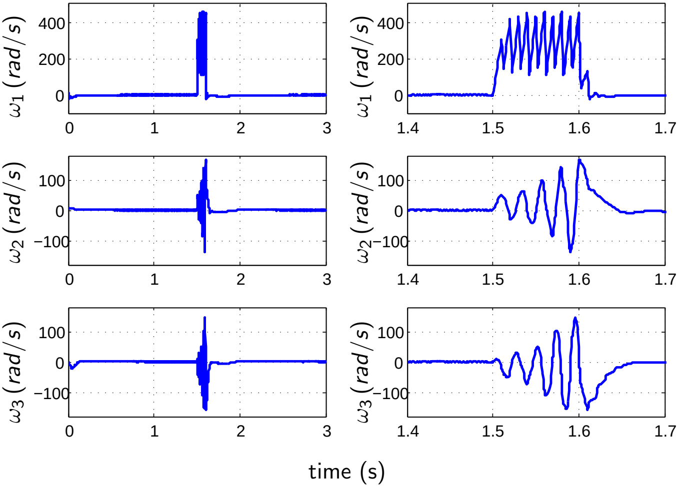
Initial orientation: $(\phi, \theta, \psi) = (70, -50, 30)^\circ$

Angles (Roll, pitch, yaw)



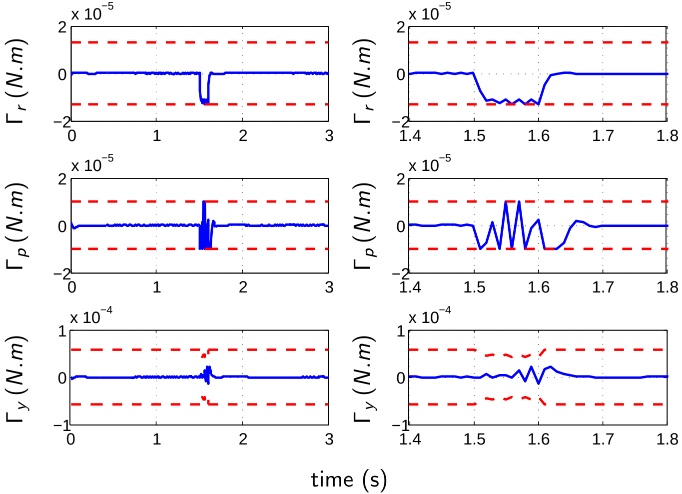
A saturation based attitude control

Angular velocities



A saturation based attitude control

Control torques



Position control

Linearization

- First thing people want to try ?
- Many possible approaches
- Taking

$$x := (\phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, p^T, v^T)^T$$

the linearization of the linear and angular dynamics around some reference x^r of the form $(0, 0, \psi^r, 0, 0, 0, {p^r}^T, 0)^T$ is given by :

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}$$

with the following matrices A and B:

$$A := \begin{pmatrix} 0_{3\times 3} & I_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & I_{3\times 3} \\ \begin{matrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \end{pmatrix}, \quad B := \begin{pmatrix} 0_{3\times 3} & 0_{3\times 1} \\ J^{-1} & 0_{3\times 1} \\ 0_{3\times 3} & 0_{3\times 1} \\ 0_{3\times 3} & \begin{matrix} 0 \\ 0 \\ \dfrac{1}{m} \end{matrix} \end{pmatrix}$$

- Linear control is always possible but not very suitable

Position control

- **A robotic oriented nonlinear approach**
  - Based on fast attitude and actuator loops
  - Angle tracking is assumed to be perfect
  - The aim is to bring the UAV to $\vec{p}^\star = \begin{pmatrix} p_1^\star & p_2^\star & p_3^\star \end{pmatrix}^T$
  - Filter the position target $\vec{p}_f^\star = \dfrac{\vec{p}^\star}{(\tau_f s + 1)^3}$, $\vec{p}_f^\star$ must be C^3
  - Let the tilde denote the error, for instance $\tilde{x} = x_f^\star - x$
  - With PIDs controllers, define an acceleration target on the two first direction ($i = 1, 2$):

$$[\ddot{p}_{i_f}]^\star = k_P \tilde{p}_i + k_I \int_0^t \tilde{p}_i dt + k_D \left(\dot{p}_{i_f}^\star - v_i\right) + \ddot{p}_{i_f}^\star$$

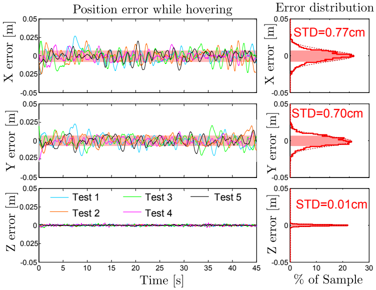
  - With a PID controller, compute the thrust control:

$$T^\star = \frac{k_P \tilde{p}_3 + k_I \int_0^t \tilde{p}_3 dt + k_D \left(\dot{p}_{3_f}^\star - v_3\right)}{c_\phi c_\theta} + \frac{m}{c_\phi c_\theta}\left(g + \ddot{p}_{3_f}^\star\right)$$

  - Yaw angle ψ can be stabilized to any direction independently
  - Compute the roll and pitch control:

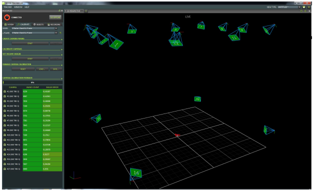
$$\begin{aligned} \phi^\star &= \sin^{-1}\left(\frac{m}{T^\star}([\ddot{p}_{1_f}]^\star s_\psi - [\ddot{p}_{2_f}]^\star c_\psi)\right) \\ \theta^\star &= \sin^{-1}\left(\frac{m}{T^\star}([\ddot{p}_{1_f}]^\star c_\psi + [\ddot{p}_{2_f}]^\star s_\psi)\right) \end{aligned}$$

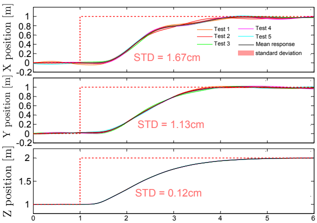
Position control

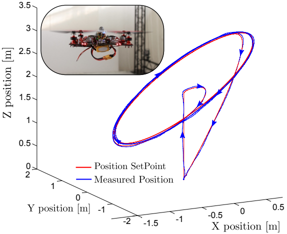


x, y error: $\pm 1.5cm$

z error: milimetric







Outline

Jointed-arm robots

- Characteristic variables:

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i
  - Actuator torques C_i

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i
  - Actuator torques C_i
  - Angles θ_i

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i
  - Actuator torques C_i
  - Angles θ_i
  - Spatial position X_i

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i
  - Actuator torques C_i
  - Angles θ_i
  - Spatial position X_i
- Controlling a robot is equivalent to mastering the relation

$$u_i \rightleftarrows C_i \rightleftarrows \theta_i \rightleftarrows X_i$$

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i
  - Actuator torques C_i
  - Angles θ_i
  - Spatial position X_i
- Controlling a robot is equivalent to mastering the relation

$$u_i \rightleftarrows C_i \rightleftarrows \theta_i \rightleftarrows X_i$$

  - Actuator dynamics

Jointed-arm robots

- Characteristic variables:
  - Actuator control u_i
  - Actuator torques C_i
  - Angles θ_i
  - Spatial position X_i
- Controlling a robot is equivalent to mastering the relation

$$u_i \rightleftarrows C_i \rightleftarrows \theta_i \rightleftarrows X_i$$

- Actuator dynamics
- Robot dynamics

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...
  - The dynamical phenomena must be negligible

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called “forward kinematics”
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...
  - The dynamical phenomena must be negligible
    - mass effect fully compensated by the inner-loop

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...
  - The dynamical phenomena must be negligible
    - mass effect fully compensated by the inner-loop
    - few flexibility of the arms (not for spatial robotic arms !)

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...
  - The dynamical phenomena must be negligible
    - mass effect fully compensated by the inner-loop
    - few flexibility of the arms (not for spatial robotic arms !)
  - Sufficiently simple model to be online inverted

Geometrical model of robots

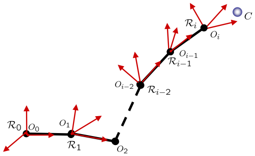
- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...
  - The dynamical phenomena must be negligible
    - mass effect fully compensated by the inner-loop
    - few flexibility of the arms (not for spatial robotic arms !)
  - Sufficiently simple model to be online inverted
  - The model must be invertible

Geometrical model of robots

- Consist in finding the relations $X_i = f_i(\theta_i)$
- Sometimes called "forward kinematics"
- That gives $X_n = f(\theta_i, \ldots, \theta_n)$, the position of the extremity of the arm as a functions of the control angles (and of the robot parameters)
- The aim is then to deduce the θ_i^r's using f^{-1} (inversion)
- Assumptions:
  - The model must be quite precise
    - no friction, no drift, no backlash, no dead zone, ...
  - The dynamical phenomena must be negligible
    - mass effect fully compensated by the inner-loop
    - few flexibility of the arms (not for spatial robotic arms !)
  - Sufficiently simple model to be online inverted
  - The model must be invertible
- Despite the limitations, this approach is widely used (oversized robots)

Computation of the geometrical model

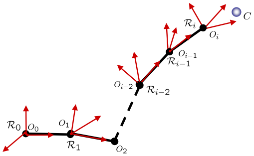
Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)

Computation of the geometrical model

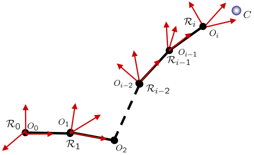
Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$

Computation of the geometrical model

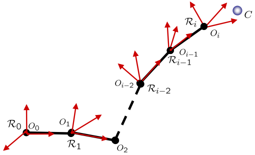
Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R^i_{i-1}\vec{v}(\mathcal{R}_{i-1})$

Computation of the geometrical model

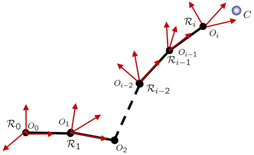
Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R^i_{i-1}\vec{v}(\mathcal{R}_{i-1})$
  - $\vec{v}(\mathcal{R}_i) = \prod_{k=1}^{i} R^k_{k-1}\vec{v}(\mathcal{R}_0) = R^i_0\vec{v}(\mathcal{R}_0)$

Computation of the geometrical model

Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R_{i-1}^{i}\vec{v}(\mathcal{R}_{i-1})$
  - $\vec{v}(\mathcal{R}_i) = \prod_{k=1}^{i} R_{k-1}^{k}\vec{v}(\mathcal{R}_0) = R_0^i\vec{v}(\mathcal{R}_0)$
- **Position**: for any point C

Computation of the geometrical model

Combination of rotations and translations

Robotics

N. Marchand

Introduction

Modeling
Cartesian coordinates
Orientation
Frames
Newton

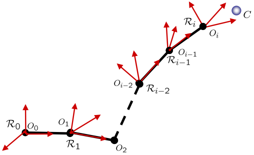
X4

Kinematics
Arm robots
Inner-loop
Geometrical model
Kinematic model
Dynamical model
Conclusion

Path planning
Workspace and obstacles
path planning

Mobile robotics

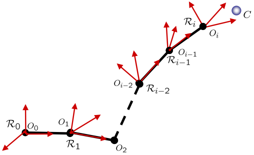
Visual servoing



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R_{i-1}^{i}\vec{v}(\mathcal{R}_{i-1})$
  - $\vec{v}(\mathcal{R}_i) = \prod_{k=1}^{i} R_{k-1}^{k}\vec{v}(\mathcal{R}_0) = R_0^i\vec{v}(\mathcal{R}_0)$
- **Position**: for any point C
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_OO_i}(\mathcal{R}_0) + \overrightarrow{O_iC}(\mathcal{R}_0) = \overrightarrow{O_OO_i}(\mathcal{R}_0) + R_i^0\overrightarrow{O_iC}(\mathcal{R}_i)$

Computation of the geometrical model

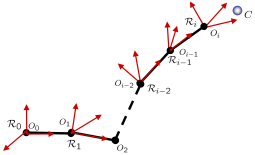
Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R_{i-1}^{i}\vec{v}(\mathcal{R}_{i-1})$
  - $\vec{v}(\mathcal{R}_i) = \prod_{k=1}^{i} R_{k-1}^{k}\vec{v}(\mathcal{R}_0) = R_0^i\vec{v}(\mathcal{R}_0)$
- **Position**: for any point C
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_OO_i}(\mathcal{R}_0) + \overrightarrow{O_iC}(\mathcal{R}_0) = \overrightarrow{O_OO_i}(\mathcal{R}_0) + R_i^0\overrightarrow{O_iC}(\mathcal{R}_i)$
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_OO_1}(\mathcal{R}_0) + R_1^0\overrightarrow{O_1O_2}(\mathcal{R}_1) + \cdots + R_{i-1}^0\overrightarrow{O_{i-1}O_i}(\mathcal{R}_{i-1}) + R_i^0\overrightarrow{O_iC}(\mathcal{R}_i)$

Computation of the geometrical model

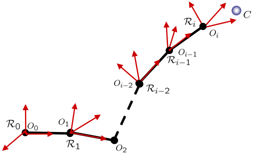
Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R^i_{i-1}\vec{v}(\mathcal{R}_{i-1})$
  - $\vec{v}(\mathcal{R}_i) = \prod_{k=1}^{i} R^k_{k-1}\vec{v}(\mathcal{R}_0) = R^i_0\vec{v}(\mathcal{R}_0)$
- **Position**: for any point C
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_0O_i}(\mathcal{R}_0) + \overrightarrow{O_iC}(\mathcal{R}_0) = \overrightarrow{O_0O_i}(\mathcal{R}_0) + R^0_i\overrightarrow{O_iC}(\mathcal{R}_i)$
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_0O_1}(\mathcal{R}_0) + R^0_1\overrightarrow{O_1O_2}(\mathcal{R}_1) + \cdots + R^0_{i-1}\overrightarrow{O_{i-1}O_i}(\mathcal{R}_{i-1}) + R^0_i\overrightarrow{O_iC}(\mathcal{R}_i)$
- where R^{i+1}_i is the rotation matrix from $\mathcal{R}_i$ to $\mathcal{R}_{i+1}$:

Computation of the geometrical model

Combination of rotations and translations



- Let X be the orientation and position of the last segment in $\mathcal{R}_0$ (usually variable to control)
- **Orientation**: for any $\vec{v}$
  - $\vec{v}(\mathcal{R}_i) = R_{i-1}^i \vec{v}(\mathcal{R}_{i-1})$
  - $\vec{v}(\mathcal{R}_i) = \prod_{k=1}^{i} R_{k-1}^k \vec{v}(\mathcal{R}_0) = R_0^i \vec{v}(\mathcal{R}_0)$
- **Position**: for any point C
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_0O_i}(\mathcal{R}_0) + \overrightarrow{O_iC}(\mathcal{R}_0) = \overrightarrow{O_0O_i}(\mathcal{R}_0) + R_i^0 \overrightarrow{O_iC}(\mathcal{R}_i)$
  - $\overrightarrow{O_0C}(\mathcal{R}_0) = \overrightarrow{O_0O_1}(\mathcal{R}_0) + R_1^0 \overrightarrow{O_1O_2}(\mathcal{R}_1) + \cdots + R_{i-1}^0 \overrightarrow{O_{i-1}O_i}(\mathcal{R}_{i-1}) + R_i^0 \overrightarrow{O_iC}(\mathcal{R}_i)$
- where R_i^{i+1} is the rotation matrix from $\mathcal{R}_i$ to $\mathcal{R}_{i+1}$:
  - $R_i^{i+1} = {R_{i+1}^i}^T$, $\det R_i^{i+1} = 1$

Computation of the geometrical model

Combination of rotations and translations

- Easy way to compute the geometrical model: **homogeneous coordinates**

Computation of the geometrical model

Combination of rotations and translations

- Easy way to compute the geometrical model: **homogeneous coordinates**
- Let $\vec{v} := \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}$, then it is equivalent to the 4-dimension vector $\vec{V}$ with $\omega = 1$:

$$V = \begin{pmatrix} v_1\omega \\ v_2\omega \\ v_3\omega \\ \omega \end{pmatrix}$$

Computation of the geometrical model

Combination of rotations and translations

- Easy way to compute the geometrical model: **homogeneous coordinates**
- Let $\vec{v} := \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}$, then it is equivalent to the 4-dimension vector $\vec{V}$ with $\omega = 1$:

$$V = \begin{pmatrix} v_1\omega \\ v_2\omega \\ v_3\omega \\ \omega \end{pmatrix}$$

- **Translation**: a translation of vector $\begin{pmatrix} a & b & c \end{pmatrix}$ is given by:

$$\text{Trans} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Computation of the geometrical model

Combination of rotations and translations

- Easy way to compute the geometrical model: **homogeneous coordinates**
- Let $\vec{v} := \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}$, then it is equivalent to the 4-dimension vector $\vec{V}$ with $\omega = 1$:

$$V = \begin{pmatrix} v_1\omega \\ v_2\omega \\ v_3\omega \\ \omega \end{pmatrix}$$

- **Translation**: a translation of vector $\begin{pmatrix} a & b & c \end{pmatrix}$ is given by:

$$\text{Trans} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

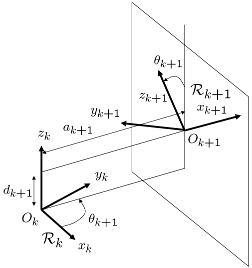
- **Rotation**: a rotation of matrix R is given by:

$$\text{Rot} = \begin{pmatrix} R & 0_{3\times 1} \\ 0_{1\times 3} & 1 \end{pmatrix}$$

Note that still $R^{-1} = R^T$ and $det(R) = 1$

Computation of the geometrical model

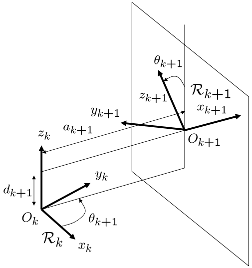
Denavit-Hartenberg's convention



- Consider two successive articulations

Computation of the geometrical model

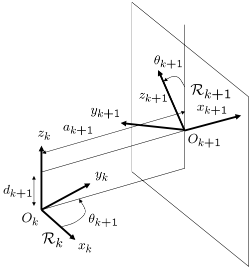
Denavit-Hartenberg's convention



- Consider two successive articulations
- Then, to go from O_k to O_{k+1} and from $\mathcal{R}_k$ to $\mathcal{R}_{k+1}$, one does successively:

Computation of the geometrical model

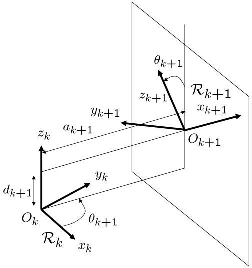
Denavit-Hartenberg's convention



- Consider two successive articulations
- Then, to go from O_k to O_{k+1} and from $\mathcal{R}_k$ to $\mathcal{R}_{k+1}$, one does successively:
  - One rotation around z_k of angle θ_{k+1}

Computation of the geometrical model

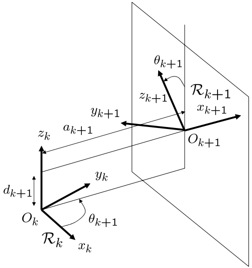
Denavit-Hartenberg's convention



- Consider two successive articulations
- Then, to go from O_k to O_{k+1} and from $\mathcal{R}_k$ to $\mathcal{R}_{k+1}$, one does successively:
  - One rotation around z_k of angle θ_{k+1}
  - One translation along z_k of distance d_{k+1}

Computation of the geometrical model

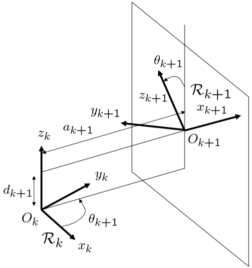
Denavit-Hartenberg's convention



- Consider two successive articulations
- Then, to go from O_k to O_{k+1} and from $\mathcal{R}_k$ to $\mathcal{R}_{k+1}$, one does successively:
  - One rotation around z_k of angle θ_{k+1}
  - One translation along z_k of distance d_{k+1}
  - One translation along x_{k+1} of distance a_{k+1}

Computation of the geometrical model

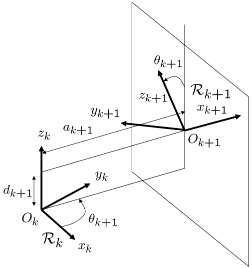
Denavit-Hartenberg's convention



- Consider two successive articulations
- Then, to go from O_k to O_{k+1} and from $\mathcal{R}_k$ to $\mathcal{R}_{k+1}$, one does successively:
  - One rotation around z_k of angle θ_{k+1}
  - One translation along z_k of distance d_{k+1}
  - One translation along x_{k+1} of distance a_{k+1}
  - One rotation around x_{k+1} of angle α_{k+1}

Computation of the geometrical model

Denavit-Hartenberg's convention



- Consider two successive articulations
- Then, to go from O_k to O_{k+1} and from $\mathcal{R}_k$ to $\mathcal{R}_{k+1}$, one does successively:
  - One rotation around z_k of angle θ_{k+1}
  - One translation along z_k of distance d_{k+1}
  - One translation along x_{k+1} of distance a_{k+1}
  - One rotation around x_{k+1} of angle α_{k+1}
- The DH parametrization **always exists and is unique**

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
    - The model must be invertible (for any X^r, there is some θ_i^r)
    - We talk about *resolvable robots*
    - Can be inverted using a optimization procedure

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r
- **Drawbacks:** the actuators are in closed loop but the robot is in open-loop

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r
- **Drawbacks:** the actuators are in closed loop but the robot is in open-loop
  - what about the speed ?

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r
- **Drawbacks:** the actuators are in closed loop but the robot is in open-loop
  - what about the speed ?
  - the trajectory is not well defined (obstacle avoidance, etc.)

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r
- **Drawbacks:** the actuators are in closed loop but the robot is in open-loop
  - what about the speed ?
  - the trajectory is not well defined (obstacle avoidance, etc.)
  - dry friction if multiple X^d

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r
- **Drawbacks:** the actuators are in closed loop but the robot is in open-loop
  - what about the speed ?
  - the trajectory is not well defined (obstacle avoidance, etc.)
  - dry friction if multiple X^d
  - what about the influence of the weight (that depends upon the configuration)

Control with the geometrical model

- Compute the set of θ_i^r corresponding to the reference X^r
- θ_i as a function of X^r is often called "inverse kinematics"
  - The model must be invertible (for any X^r, there is some θ_i^r)
  - We talk about *resolvable robots*
  - Can be inverted using a optimization procedure
- Make a step in the inner control loop to go from θ_i^0 to θ_i^r
- **Drawbacks:** the actuators are in closed loop but the robot is in open-loop
  - what about the speed ?
  - the trajectory is not well defined (obstacle avoidance, etc.)
  - dry friction if multiple X^d
  - what about the influence of the weight (that depends upon the configuration)
  - inertia may cause overshoot or oscillations

Exercise

- Compute the matrix transformation of the Denavit-Hartenberg's convention

Exercise

- Compute the matrix transformation of the Denavit-Hartenberg's convention
  - One rotation around z_k of angle θ_{k+1}:

$$R_1 = \begin{pmatrix} c\theta_{k+1} & -s\theta_{k+1} & 0 & 0 \\ s\theta_{k+1} & c\theta_{k+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Exercise

- Compute the matrix transformation of the Denavit-Hartenberg's convention
  - One rotation around z_k of angle θ_{k+1}:

$$R_1 = \begin{pmatrix} c\theta_{k+1} & -s\theta_{k+1} & 0 & 0 \\ s\theta_{k+1} & c\theta_{k+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - One translation along z_k of distance d_{k+1}

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{k+1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Exercise

- Compute the matrix transformation of the Denavit-Hartenberg's convention
  - One rotation around z_k of angle θ_{k+1}:
$$R_1 = \begin{pmatrix} c\theta_{k+1} & -s\theta_{k+1} & 0 & 0 \\ s\theta_{k+1} & c\theta_{k+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - One translation along z_k of distance d_{k+1}
$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{k+1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - One translation along x_{k+1} of distance a_{k+1}
$$T_2 = \begin{pmatrix} 1 & 0 & 0 & a_{k+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Exercise

- Compute the matrix transformation of the Denavit-Hartenberg's convention
  - One rotation around z_k of angle θ_{k+1}:

$$R_1 = \begin{pmatrix} c\theta_{k+1} & -s\theta_{k+1} & 0 & 0 \\ s\theta_{k+1} & c\theta_{k+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - One translation along z_k of distance d_{k+1}

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{k+1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - One translation along x_{k+1} of distance a_{k+1}

$$T_2 = \begin{pmatrix} 1 & 0 & 0 & a_{k+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - One rotation around x_{k+1} of angle α_{k+1}

$$R_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{k+1} & -s\alpha_{k+1} & 0 \\ 0 & s\alpha_{k+1} & c\alpha_{k+1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Exercise

- Compute the matrix transformation of the Denavit-Hartenberg's convention
  - One rotation around z_k of angle θ_{k+1}:
$$R_1 = \begin{pmatrix} c\theta_{k+1} & -s\theta_{k+1} & 0 & 0 \\ s\theta_{k+1} & c\theta_{k+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - One translation along z_k of distance d_{k+1}
$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{k+1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - One translation along x_{k+1} of distance a_{k+1}
$$T_2 = \begin{pmatrix} 1 & 0 & 0 & a_{k+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - One rotation around x_{k+1} of angle α_{k+1}
$$R_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{k+1} & -s\alpha_{k+1} & 0 \\ 0 & s\alpha_{k+1} & c\alpha_{k+1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
  - The matrix transformation of the Denavit-Hartenberg's convention is: $R_2 \cdot T_2 \cdot T_1 \cdot R_1$

Outline

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\dfrac{d\theta}{dt}$
- Sometimes called "velocity kinematics"

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables
- The kinematic model is "simply" the derivation of the geometric model $X = f(\theta_0, \theta_1, \cdots, \theta_n)$:

$$\dot{X} = \frac{\partial f}{\partial \theta}\dot{\theta}$$

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables
- The kinematic model is "simply" the derivation of the geometric model $X = f(\theta_0, \theta_1, \cdots, \theta_n)$:

$$\dot{X} = \frac{\partial f}{\partial \theta}\dot{\theta}$$

- $J := \frac{\partial f}{\partial \theta}$ is called the *Jacobian* of the robot

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables
- The kinematic model is "simply" the derivation of the geometric model $X = f(\theta_0, \theta_1, \cdots, \theta_n)$:

$$\dot{X} = \frac{\partial f}{\partial \theta}\dot{\theta}$$

- $J := \frac{\partial f}{\partial \theta}$ is called the *Jacobian* of the robot
- J represents the instantaneous transformation between a vector of joint velocities and the linear and angular velocities of the end-effector

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables
- The kinematic model is "simply" the derivation of the geometric model $X = f(\theta_0, \theta_1, \cdots, \theta_n)$:

$$\dot{X} = \frac{\partial f}{\partial \theta}\dot{\theta}$$

- $J := \frac{\partial f}{\partial \theta}$ is called the *Jacobian* of the robot
- J represents the instantaneous transformation between a vector of joint velocities and the linear and angular velocities of the end-effector
- J can be decomposed into J_v and J_ω so that:

$$\begin{aligned} \dot{x}_n^{\mathcal{R}_f} &= J_v\dot{\theta} \\ \omega_n^{\mathcal{R}_f} &= J_\omega\dot{\theta} \end{aligned}$$

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables
- The kinematic model is "simply" the derivation of the geometric model $X = f(\theta_0, \theta_1, \cdots, \theta_n)$:

$$\dot{X} = \frac{\partial f}{\partial \theta}\dot{\theta}$$

- $J := \frac{\partial f}{\partial \theta}$ is called the *Jacobian* of the robot
- J represents the instantaneous transformation between a vector of joint velocities and the linear and angular velocities of the end-effector
- J can be decomposed into J_v and J_ω so that:

$$\begin{aligned} \dot{x}_n^{\mathcal{R}_f} &= J_v\dot{\theta} \\ \omega_n^{\mathcal{R}_f} &= J_\omega\dot{\theta} \end{aligned}$$

- The kinematic model can also be obtained using the composition of speed and decomposing the Denavit-Hartenberg's parametrization:

$$R(z,\theta)T(z,d)T(x^+,a)R(x^+,\alpha)$$

Kinematic model of robots

- Express the infinitesimal mouvement dX as a function of speed of the actuators $\frac{d\theta}{dt}$
- Sometimes called "velocity kinematics"
- Assumes that, thanks to inner-loops, actuators speeds can be assumed to be control variables
- The kinematic model is "simply" the derivation of the geometric model $X = f(\theta_0, \theta_1, \cdots, \theta_n)$:

$$\dot{X} = \frac{\partial f}{\partial \theta}\dot{\theta}$$

- $J := \frac{\partial f}{\partial \theta}$ is called the *Jacobian* of the robot
- J represents the instantaneous transformation between a vector of joint velocities and the linear and angular velocities of the end-effector
- J can be decomposed into J_v and J_ω so that:

$$\begin{aligned} \dot{x}_n^{\mathcal{R}_f} &= J_v\dot{\theta} \\ \omega_n^{\mathcal{R}_f} &= J_\omega\dot{\theta} \end{aligned}$$

- The kinematic model can also be obtained using the composition of speed and decomposing the Denavit-Hartenberg's parametrization:

$$R(z,\theta)T(z,d)T(x^+,a)R(x^+,\alpha)$$

- Fastidious in many cases but systematic ! See books for that

Kinematic model of robots

- Kinematic model can be used if "it can be stopped quasi instantaneously" (quickly w.r.t. the tasks to be done)

Kinematic model of robots

- Kinematic model can be used if “it can be stopped quasi instantaneously” (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected

Kinematic model of robots

- Kinematic model can be used if “it can be stopped quasi instantaneously” (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected
- Many cases can happen:

Kinematic model of robots

- Kinematic model can be used if "it can be stopped quasi instantaneously" (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected
- Many cases can happen:
  - J is square and full rank: miracle !

Kinematic model of robots

- Kinematic model can be used if "it can be stopped quasi instantaneously" (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected
- Many cases can happen:
  - J is square and full rank: miracle !
  - J is square but for some articulation position, $\det J = 0$ (singularities), the singularities are usually avoided

Kinematic model of robots

- Kinematic model can be used if "it can be stopped quasi instantaneously" (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected
- Many cases can happen:
  - J is square and full rank: miracle !
  - J is square but for some articulation position, $\det J = 0$ (singularities), the singularities are usually avoided
  - J has more columns than rows: add a criterium to find the optimal path

Kinematic model of robots

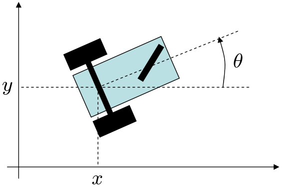
- Kinematic model can be used if "it can be stopped quasi instantaneously" (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected
- Many cases can happen:
  - J is square and full rank: miracle !
  - J is square but for some articulation position, $\det J = 0$ (singularities), the singularities are usually avoided
  - J has more columns than rows: add a criterium to find the optimal path
  - J has more rows than columns: impossible configurations of nonholonomic constraints, nonlinear control theory to solve this problem

Kinematic model of robots

- Kinematic model can be used if "it can be stopped quasi instantaneously" (quickly w.r.t. the tasks to be done)
- As for geometrical model, the dynamics has to be neglected
- Many cases can happen:
  - J is square and full rank: miracle !
  - J is square but for some articulation position, $\det J = 0$ (singularities), the singularities are usually avoided
  - J has more columns than rows: add a criterium to find the optimal path
  - J has more rows than columns: impossible configurations of nonholonomic constraints, nonlinear control theory to solve this problem
- **The kinematic model is a state space representation of a controlled system**

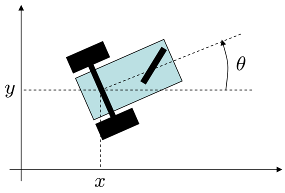
Example of kinematic model

- Example: the car in the plane



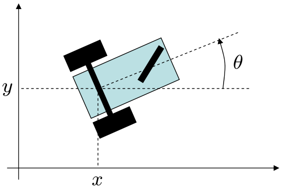
Example of kinematic model

- Example: the car in the plane
  - Characterizing variables (state variables): x, y and θ



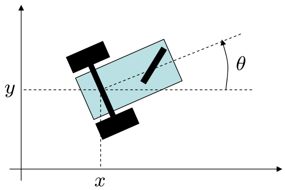
Example of kinematic model

- Example: the car in the plane
  - Characterizing variables (state variables): x, y and θ
  - Control variables: speed of each wheels V_r and V_l



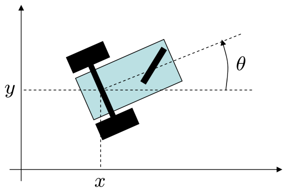
Example of kinematic model

- Example: the car in the plane
  - Characterizing variables (state variables): x, y and θ
  - Control variables: speed of each wheels V_r and V_l
  - The kinematic model is given by the relation between $\dot{x}$, $\dot{y}$, $\dot{\theta}$ and the controls V_r and V_l



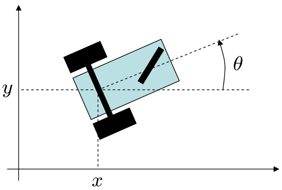
Example of kinematic model

- Example: the car in the plane
  - Characterizing variables (state variables): x, y and θ
  - Control variables: speed of each wheels V_r and V_l
  - The kinematic model is given by the relation between $\dot{x}$, $\dot{y}$, $\dot{\theta}$ and the controls V_r and V_l
  - What is the kinematic model of the car ?



Example of kinematic model

- Example: the car in the plane
  - Characterizing variables (state variables): x, y and θ
  - Control variables: speed of each wheels V_r and V_l
  - The kinematic model is given by the relation between $\dot{x}$, $\dot{y}$, $\dot{\theta}$ and the controls V_r and V_l
  - What is the kinematic model of the car ?
  - What is the expression of the Jacobian of this robot ?



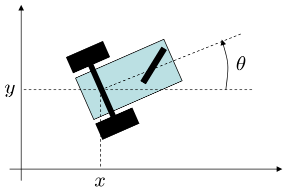
$$\dot{x} = \frac{V_l + V_r}{2} \cos\theta$$

$$\dot{y} = \frac{V_l + V_r}{2} \sin\theta$$

$$\dot{\theta} = \frac{V_r - V_l}{d}$$

Example of kinematic model

- Example: the car in the plane
  - Characterizing variables (state variables): x, y and θ
  - Control variables: speed of each wheels V_r and V_l
  - The kinematic model is given by the relation between $\dot{x}$, $\dot{y}$, $\dot{\theta}$ and the controls V_r and V_l
  - What is the kinematic model of the car ?
  - What is the expression of the Jacobian of this robot ?
  - Is this system underactuated or overactuated ? Explain why



$$\begin{aligned}\dot{x} &= \frac{V_l + V_r}{2}\cos\theta\\ \dot{y} &= \frac{V_l + V_r}{2}\sin\theta\\ \dot{\theta} &= \frac{V_r - V_l}{d}\end{aligned}$$

$$J = \frac{1}{2}\begin{pmatrix}\cos\theta & \cos\theta\\ \sin\theta & \sin\theta\\ -\frac{2}{d} & \frac{2}{d}\end{pmatrix}$$

Relation between workspace forces and joint torques

- The workspace forces and joint torques are linked with the relation:

$$\tau = J_v^T F$$

Relation between workspace forces and joint torques

- The workspace forces and joint torques are linked with the relation:

$$\tau = J_v^T F$$

- the Jacobian must be derived at each origin O_i of each link frame

Kinematic redundancy

When a robot is given by its kinematic model $\dot{X} = J\dot{\theta}$

- J is usually $n \times p$ with $X \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^p$
- $r = p - n$ is called the **kinematic redundancy number**
- When $r < 0$, the robot is underactuated, usually the case with mobile robots $\Rightarrow$ advanced control
- When $r > 0$, the robot is overactuated. It has redundancy.

For a robot with redundancy, one can write:

- $J = \begin{pmatrix} J_n & J_{p-n} \end{pmatrix}$ with J_n invertible

Control through the kinematic equation

Control with J^t Take a robot given by its kinematic model $\dot{X} = J\dot{\theta}$

- Control with J^t
  - Apply a fictive force $F = K(X - X_d)$ with K positive and symmetric
  - Take $\dot{\theta} = J^t F = J^t K(X - X_d) = J^t K e$
  - Then the elastic potential $\Phi(e) = \frac{1}{2} e^t K e$ is such that

$$\dot{\Phi}(e) = -e^t K J J^t K e < 0$$

  - $e \rightharpoonup 0$, $X \rightharpoonup X_d$
  - Automatically handles redundancy

Control through the kinematic equation

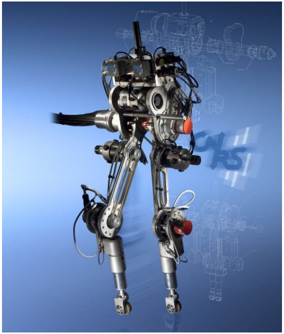
Control with J^+ Take a robot given by its kinematic model $\dot{X} = J\dot{\theta}$

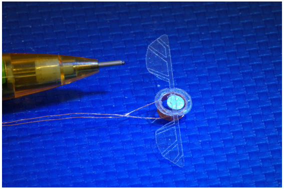
- Control with $J^+ := J^t(JJ^t)^{-1}$
  - J^+ is the Moore-Penrose pseudo-inverse (`pinv` in Matlab)
  - Can be obtained through SVD decomposition. $J = U\Delta V^t$, Δ diagonal $\Longrightarrow J^+ = V\Delta^+ U^t$, Δ^+ is the inverse of the nonzero coefficient of Δ
  - Taking $\dot{\theta} = J^+\dot{X}$ minimizes the energy $\dot{\theta}^t\dot{\theta}$
  - Taking $\dot{\theta} = J_M^+\dot{X}$ with $J_M^+ := M^{-1}J^t(JM^{-1}J^t)^{-1}$ minimizes the kinetic energy $T = \frac{1}{2}\dot{\theta}^t M(\theta)\dot{\theta}$

Outline

Dynamical model of robots

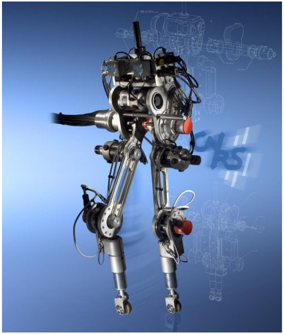
- Express the accelerations of movement as a function of the actuation variables

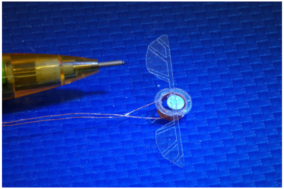




Dynamical model of robots

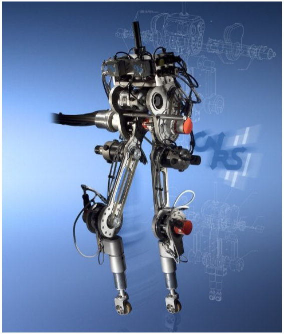
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)

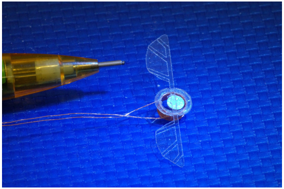




Dynamical model of robots

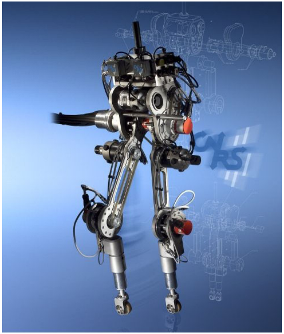
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)

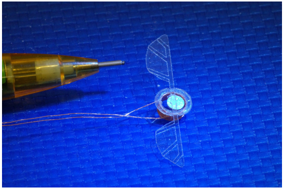




Dynamical model of robots

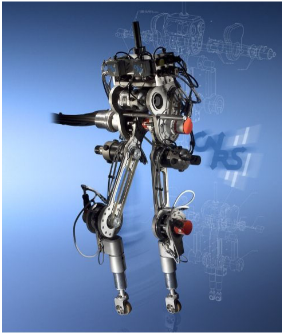
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)
- Very complex and most of the time impossible to control (too complex to design a control)

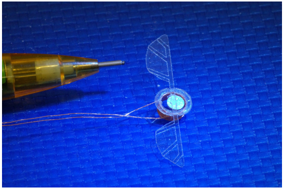




Dynamical model of robots

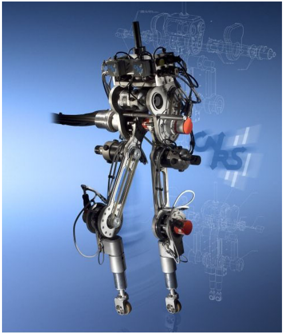
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)
- Very complex and most of the time impossible to control (too complex to design a control)
- simplifications are required:

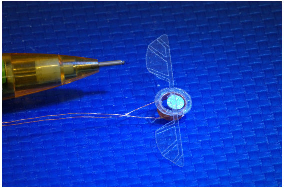




Dynamical model of robots

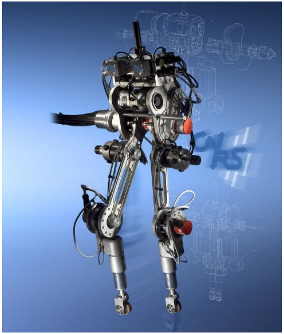
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)
- Very complex and most of the time impossible to control (too complex to design a control)
- simplifications are required:
    - based on relative speed of the $\neq$ parts of the robot

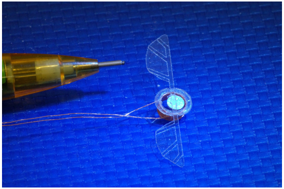




Dynamical model of robots

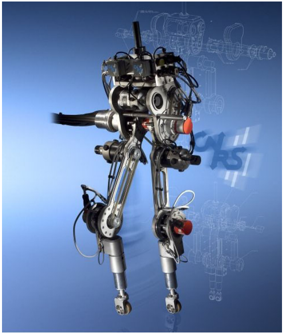
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)
- Very complex and most of the time impossible to control (too complex to design a control)
- simplifications are required:
  - based on relative speed of the $\neq$ parts of the robot
  - thanks to inner-loops that can render parts instantaneous w.r.t. other parts of the robot

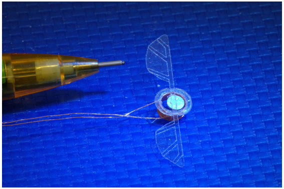




Dynamical model of robots

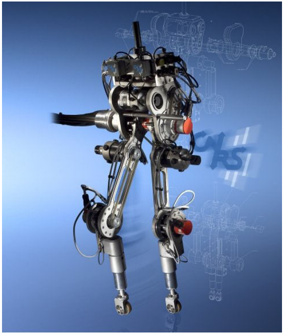
- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)
- Very complex and most of the time impossible to control (too complex to design a control)
- simplifications are required:
  - based on relative speed of the $\neq$ parts of the robot
  - thanks to inner-loops that can render parts instantaneous w.r.t. other parts of the robot
- Almost never used for arm-robots

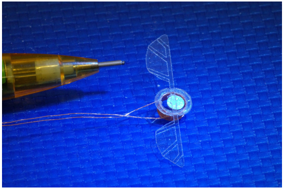




Dynamical model of robots

- Express the accelerations of movement as a function of the actuation variables
- The dynamical model is obtained writing the mechanical equations of the system (conservation of momentum)
- Sometimes also includes the actuators dynamics (mainly electrical or pneumatical)
- Very complex and most of the time impossible to control (too complex to design a control)
- simplifications are required:
  - based on relative speed of the $\neq$ parts of the robot
  - thanks to inner-loops that can render parts instantaneous w.r.t. other parts of the robot
- Almost never used for arm-robots
- Widely used for flying or diving robots (UAVs, AUVs, etc.) or walking robots





Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

  - Obtained thanks to the Euler-Lagrange formalism

Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

  - Obtained thanks to the Euler-Lagrange formalism
  - q are the generalized coordinates

Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

  - Obtained thanks to the Euler-Lagrange formalism
  - q are the generalized coordinates
  - $C(q,\dot{q})\dot{q} = \sum_i \sum_j c_{ij}(q)\dot{q}_i\dot{q}_j$

Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

  - Obtained thanks to the Euler-Lagrange formalism
  - q are the generalized coordinates
  - $C(q,\dot{q})\dot{q} = \sum_i \sum_j c_{ij}(q)\dot{q}_i\dot{q}_j$
    - Centrifugal effect when $i = j$ (term in $\dot{q}_i^2$)

Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

  - Obtained thanks to the Euler-Lagrange formalism
  - q are the generalized coordinates
  - $C(q,\dot{q})\dot{q} = \sum_i \sum_j c_{ij}(q)\dot{q}_i\dot{q}_j$
    - Centrifugal effect when $i = j$ (term in $\dot{q}_i^2$)
    - Coriolis effect when $i \neq j$ (terms in $\dot{q}_i\dot{q}_j$)

Dynamical models of robots

n-link manipulator

- The dynamical equations are of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = r$$

  - Obtained thanks to the Euler-Lagrange formalism
  - q are the generalized coordinates
  - $C(q,\dot{q})\dot{q} = \sum_i \sum_j c_{ij}(q)\dot{q}_i\dot{q}_j$
    - Centrifugal effect when $i = j$ (term in $\dot{q}_i^2$)
    - Coriolis effect when $i \neq j$ (terms in $\dot{q}_i\dot{q}_j$)
  - An important literature on the control of this type of systems can be found

Dynamical models of robots

flying and diving robots

- The dynamical equations are of the form:

$$\begin{cases} \dot{\vec{p}} & = & \vec{v} \\ m\dot{\vec{v}} & = & -mg\vec{e}_3 + R\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} \\ \dot{R} & = & R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times} J\vec{\omega} + \begin{pmatrix} \Gamma_r \\ \Gamma_p \\ \Gamma_y \end{pmatrix} \end{cases}$$

Dynamical models of robots

flying and diving robots

- The dynamical equations are of the form:

$$\left\{\begin{array}{rcl}\dot{\vec{p}} & = & \vec{v} \\ m\dot{\vec{v}} & = & -mg\vec{e}_3 + R\begin{pmatrix}F_x \\ F_y \\ F_z\end{pmatrix} \\ \dot{R} & = & R\vec{\omega}^{\times} \\ J\dot{\vec{\omega}} & = & -\vec{\omega}^{\times}J\vec{\omega} + \begin{pmatrix}\Gamma_r \\ \Gamma_p \\ \Gamma_y\end{pmatrix}\end{array}\right.$$

- The number of available controls depends upon the system

Outline

Different models of robots

- **Geometrical model** (or forward kinematic model):

$$\text{Position of the robot} = f(\text{position of the actuators})$$

Different models of robots

- **Geometrical model** (or forward kinematic model):

$$\text{Position of the robot} = f(\text{position of the actuators})$$

- **Inverse geometrical model** (or inverse kinematic model):

$$\text{Position of the actuators} = f(\text{position of the robot})$$

Different models of robots

- **Geometrical model** (or forward kinematic model):

$$\text{Position of the robot} = f(\text{position of the actuators})$$

- **Inverse geometrical model** (or inverse kinematic model):

$$\text{Position of the actuators} = f(\text{position of the robot})$$

- **Kinematic model (state space representation)** (or velocity kinematic model):

$$\text{Speed of the robot} = f(\text{position,actuation speed})$$

Different models of robots

- **Geometrical model** (or forward kinematic model):

$$\text{Position of the robot} = f(\text{position of the actuators})$$

- **Inverse geometrical model** (or inverse kinematic model):

$$\text{Position of the actuators} = f(\text{position of the robot})$$

- **Kinematic model (state space representation)** (or velocity kinematic model):

$$\text{Speed of the robot} = f(\text{position,actuation speed})$$

- **Dynamical model (state space representation)**:

$$\text{Robot acceleration} = f(\text{position and speed,forces/torques})$$

Outline

Path planning

- Need to choose a path for the end effector that avoids

Path planning

- Need to choose a path for the end effector that avoids
  - collisions

Path planning

- Need to choose a path for the end effector that avoids
  - collisions
  - singularities of the robot

Path planning

- Need to choose a path for the end effector that avoids
  - collisions
  - singularities of the robot
- Collision are easy to characterize in the workspace but may need to be transformed in the configuration space

Path planning

- Need to choose a path for the end effector that avoids
  - collisions
  - singularities of the robot
- Collision are easy to characterize in the workspace but may need to be transformed in the configuration space
- The complexity of obstacle avoidance grows exponentially with the number of DOF

Path planning

- Need to choose a path for the end effector that avoids
  - collisions
  - singularities of the robot
- Collision are easy to characterize in the workspace but may need to be transformed in the configuration space
- The complexity of obstacle avoidance grows exponentially with the number of DOF
- The method used are (usually):

Path planning

- Need to choose a path for the end effector that avoids
  - collisions
  - singularities of the robot
- Collision are easy to characterize in the workspace but may need to be transformed in the configuration space
- The complexity of obstacle avoidance grows exponentially with the number of DOF
- The method used are (usually):
  - Potential field: renders the obstacle repulsive

Path planning

- Need to choose a path for the end effector that avoids
  - collisions
  - singularities of the robot
- Collision are easy to characterize in the workspace but may need to be transformed in the configuration space
- The complexity of obstacle avoidance grows exponentially with the number of DOF
- The method used are (usually):
  - Potential field: renders the obstacle repulsive
  - Gradient descent or Probabilistic roadmaps to generate the path

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable
  - Dexterous

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable
  - Dexterous
- The "configuration" is the "location" of all points of the robot

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable
  - Dexterous
- The "configuration" is the "location" of all points of the robot
  - Configuration answers the question: where is the robot

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable
  - Dexterous
- The "configuration" is the "location" of all points of the robot
  - Configuration answers the question: where is the robot
  - The configuration can be adapted to the problem: from the set of all points of the robot to the sole the effector

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable
  - Dexterous
- The "configuration" is the "location" of all points of the robot
  - Configuration answers the question: where is the robot
  - The configuration can be adapted to the problem: from the set of all points of the robot to the sole the effector
  - The θ_i's are sufficient to characterize the configuration of an arm robot for arm robots

Workspace and obstacles

- The workspace is the volume W the end effector can reach. Usually divided into:
  - Reachable
  - Dexterous
- The "configuration" is the "location" of all points of the robot
  - Configuration answers the question: where is the robot
  - The configuration can be adapted to the problem: from the set of all points of the robot to the sole the effector
  - The θ_i's are sufficient to characterize the configuration of an arm robot for arm robots
- The set of θ_i's corresponding to a possible configuration is noted Q

Workspace and obstacles

- Obstacles are denotes O_i and the set of obstacle is $O = \cup O_i$

Workspace and obstacles

- Obstacles are denotes O_i and the set of obstacle is $O = \cup O_i$
- Let $\theta \in Q$ and $C(\theta)$ denote the corresponding configuration

Workspace and obstacles

- Obstacles are denotes O_i and the set of obstacle is $O = \cup O_i$
- Let $\theta \in Q$ and $C(\theta)$ denote the corresponding configuration
- Then the workspace can be divided into:

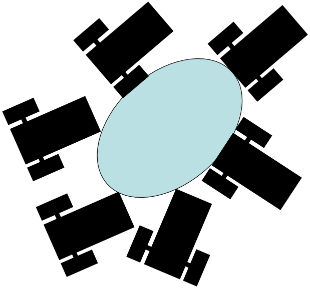
Workspace and obstacles

- Obstacles are denotes O_i and the set of obstacle is $O = \cup O_i$
- Let $\theta \in Q$ and $C(\theta)$ denote the corresponding configuration
- Then the workspace can be divided into:
  - the collision-free configuration subspace $Q_f = \{\theta \in Q | C(\theta) \cap O = \emptyset\}$

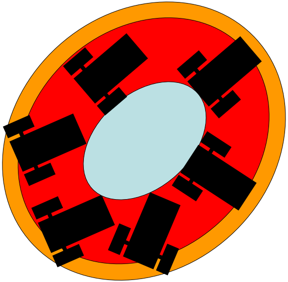
Workspace and obstacles

- Obstacles are denotes O_i and the set of obstacle is $O = \cup O_i$
- Let $\theta \in Q$ and $C(\theta)$ denote the corresponding configuration
- Then the workspace can be divided into:
  - the collision-free configuration subspace $Q_f = \{\theta \in Q | C(\theta) \cap O = \emptyset\}$
  - the collision configuration subspace $Q_c = \{\theta \in Q | C(\theta) \cap O \neq \emptyset\}$

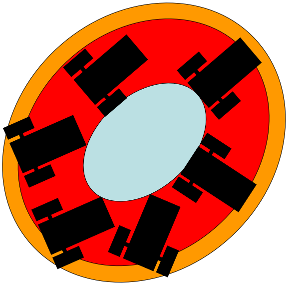
Example: the car



Example: the car

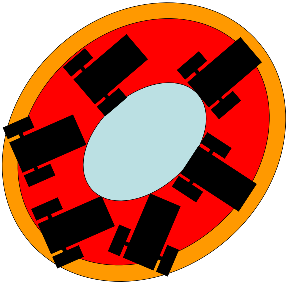


Example: the car



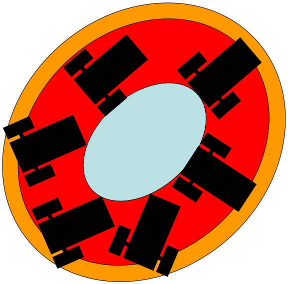
- The collision configuration subspace is the convex hull in which the robot and an obstacle make vertex to vertex contact

Example: the car



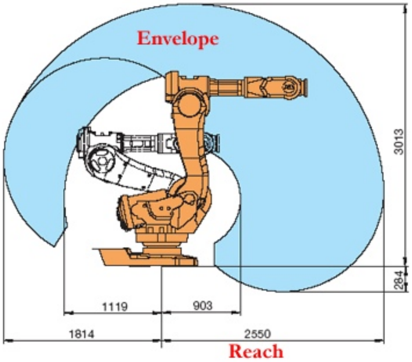
- The collision configuration subspace is the convex hull in which the robot and an obstacle make vertex to vertex contact
- Can be much more complicate to obtain

Example: the car



- The collision configuration subspace is the convex hull in which the robot and an obstacle make vertex to vertex contact
- Can be much more complicate to obtain
- Numerical simulation can easily solve this problem (systematic simulation)

Example: arm robot



Outline

A recall on gradient descent

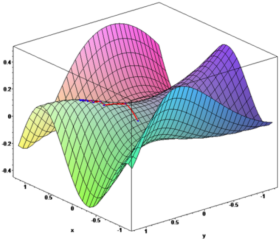
- $F(x, y) = \sin(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3)\cos(2x + 1 - e^y)$

A recall on gradient descent

- $F(x, y) = \sin(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3)\cos(2x + 1 - e^y)$
- $z := (x, y),\ F(x, y) = F(z)$

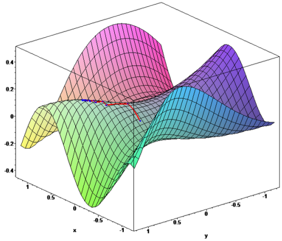
A recall on gradient descent

- $F(x,y) = \sin(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3)\cos(2x + 1 - e^y)$
- $z := (x,y)$, $F(x,y) = F(z)$
- **Aim**: finding $z^\star$ such that $F(z^\star)$ is minimum



A recall on gradient descent

- $F(x,y) = \sin(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3)\cos(2x + 1 - e^y)$
- $z := (x,y)$, $F(x,y) = F(z)$
- Aim: finding $z^\star$ such that $F(z^\star)$ is minimum

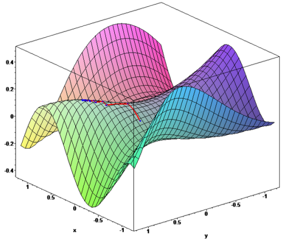


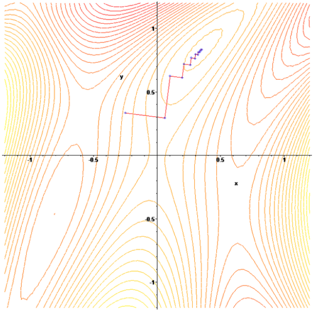
- Maximum/minimum obtained iteratively by :

$$z_{k+1} = z_k - \gamma \nabla F(z_k)$$

A recall on gradient descent

- $F(x,y) = \sin(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3)\cos(2x + 1 - e^y)$
- $z := (x,y)$, $F(x,y) = F(z)$
- Aim: finding $z^\star$ such that $F(z^\star)$ is minimum





- Maximum/minimum obtained iteratively by :

$$z_{k+1} = z_k - \gamma \nabla F(z_k)$$

A recall on gradient descent

About the stop criteria

- Many solutions to stop the iteration

$$z_{k+1} = z_k - \gamma \nabla F(z_k)$$

A recall on gradient descent

About the stop criteria

- Many solutions to stop the iteration

$$z_{k+1} = z_k - \gamma \nabla F(z_k)$$

- Better from the criteria point of view:

$$\text{stops if } F(z_{k+1}) > F(z_k)$$

A recall on gradient descent

About the stop criteria

- Many solutions to stop the iteration

$$z_{k+1} = z_k - \gamma \nabla F(z_k)$$

- Better from the criteria point of view:
  stops if $F(z_{k+1}) > F(z_k)$
- No more improvement in the criteria:
  stops if $|F(z_{k+1}) - F(z_k)| < \varepsilon$

A recall on gradient descent

About the stop criteria

- Many solutions to stop the iteration

$$z_{k+1} = z_k - \gamma \nabla F(z_k)$$

- Better from the criteria point of view:
  stops if $F(z_{k+1}) > F(z_k)$
- No more improvement in the criteria:
  stops if $|F(z_{k+1}) - F(z_k)| < \varepsilon$
- No more slope (almost the same as previous condition)
  stops if $||\nabla F(z_k)|| < \varepsilon$

A recall on gradient descent

About the step size γ

- On the step size γ

A recall on gradient descent

About the step size γ

- On the step size γ
- Newton-Euler method: H, Hessian of F

$$z_{k+1} = z_k - \nabla F(z_k) H(x_k)^{-1}$$

A recall on gradient descent

About the step size γ

- On the step size γ
- Newton-Euler method: H, Hessian of F

$$z_{k+1} = z_k - \nabla F(z_k) H(x_k)^{-1}$$

- Quasi-Newton method:

$$z_{k+1} = z_k - \rho_k B_k \nabla F(z_k)$$

B_k: approximation of the Hessian

http://en.wikipedia.org/wiki/Quasi-Newton_method)

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0, 1] \rightarrow Q_f$ such that

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0,1] \rightarrow Q_f$ such that
  - $\gamma(0) = \theta_0$ and $\gamma(1) = \theta_f$

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0,1] \rightarrow Q_f$ such that
  - $\gamma(0) = \theta_0$ and $\gamma(1) = \theta_f$
- γ will represent a configuration between the initial configuration and the final

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0,1] \to Q_f$ such that
  - $\gamma(0) = \theta_0$ and $\gamma(1) = \theta_f$
- γ will represent a configuration between the initial configuration and the final
- The aim will be to fin successive γ that remain in Q_f:

$\tau \to \gamma(\tau)$ is a path from θ_0 to θ_f

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0,1] \rightarrow Q_f$ such that
  - $\gamma(0) = \theta_0$ and $\gamma(1) = \theta_f$
- γ will represent a configuration between the initial configuration and the final
- The aim will be to fin successive γ that remain in Q_f:

$\tau \rightarrow \gamma(\tau)$ is a path from θ_0 to θ_f

- We define a potential field (criterium):

$$U(\theta) = U_{att}(\theta) + U_{rep}(\theta)$$

The aim will be to minimize the criterium

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0,1] \to Q_f$ such that
  - $\gamma(0) = \theta_0$ and $\gamma(1) = \theta_f$
- γ will represent a configuration between the initial configuration and the final
- The aim will be to fin successive γ that remain in Q_f:

$$\tau \to \gamma(\tau) \text{ is a path from } \theta_0 \text{ to } \theta_f$$

- We define a potential field (criterium):

$$U(\theta) = U_{att}(\theta) + U_{rep}(\theta)$$

  - $U_{att}(\theta)$ will attract γ to θ_f: the goal configuration

The aim will be to minimize the criterium

Path planning problem formulation

- Want to go from one configuration θ_0 (position) to another one θ_f
- We define a continuous function $\gamma : [0,1] \to Q_f$ such that
  - $\gamma(0) = \theta_0$ and $\gamma(1) = \theta_f$
- γ will represent a configuration between the initial configuration and the final
- The aim will be to fin successive γ that remain in Q_f:

  $\tau \to \gamma(\tau)$ is a path from θ_0 to θ_f
- We define a potential field (criterium):

$$U(\theta) = U_{att}(\theta) + U_{rep}(\theta)$$

  - $U_{att}(\theta)$ will attract γ to θ_f: the goal configuration
  - $U_{rep}(\theta)$ will repulse the system away from obstacle

The aim will be to minimize the criterium

Simple exemple of objective function

- Take $U_{att}(\theta) = ||\theta - \theta_f||$: U_{att} is the distance to the final destination

Simple exemple of objective function

- Take $U_{att}(\theta) = ||\theta - \theta_f||$: U_{att} is the distance to the final destination
- Take $U_{rep}(\theta) = \dfrac{1}{d(\theta, Q_c)}$: U_{rep} is infinite if there is a risk of obstacle

Attractive/Repulsive fields

- Trying to minimize or maximize the distance is not necessary appropriate

Attractive/Repulsive fields

- Trying to minimize or maximize the distance is not necessary appropriate
- Inappropriate criterium may:

Attractive/Repulsive fields

- Trying to minimize or maximize the distance is not necessary appropriate
- Inappropriate criterium may:
  - generate local minima

Attractive/Repulsive fields

- Trying to minimize or maximize the distance is not necessary appropriate
- Inappropriate criterium may:
  - generate local minima
  - be delicate to minimize

Attractive/Repulsive fields

- Trying to minimize or maximize the distance is not necessary appropriate
- Inappropriate criterium may:
  - generate local minima
  - be delicate to minimize
  - have singularities

Attractive/Repulsive fields

- Trying to minimize or maximize the distance is not necessary appropriate
- Inappropriate criterium may:
  - generate local minima
  - be delicate to minimize
  - have singularities
- **The main problem consist in finding a criterium that will be convex** (or close to)

Manipulator robots

- We define a potential field for each articulation

Manipulator robots

- We define a potential field for each articulation
- The attractive field is a monotonically increasing function of the distance of the i^{th} frame to the goal position

Manipulator robots

- We define a potential field for each articulation
- The attractive field is a monotonically increasing function of the distance of the i^{th} frame to the goal position
- The attractive field applies a fictitious force that push the manipulator into its goal position

Manipulator robots

- We define a potential field for each articulation
- The attractive field is a monotonically increasing function of the distance of the i^{th} frame to the goal position
- The attractive field applies a fictitious force that push the manipulator into its goal position
- The repulsive field will create a fictitious force that will prevent collisions by repelling the robot from the obstacles

Attractive fields

- Simple potential field: *conic well potential*

$$U_{att_i}(\theta) = \zeta_i \, ||O_i(\theta) - O_i(\theta_f)||$$

Attractive fields

- Simple potential field: *conic well potential*

$$U_{att_i}(\theta) = \zeta_i \left|\left| O_i(\theta) - O_i(\theta_f) \right|\right|$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\zeta_i \nabla \left|\left| O_i(\theta) - O_i(\theta_f) \right|\right| = -\zeta_i \frac{O_i(\theta) - O_i(\theta_f)}{\left|\left| O_i(\theta) - O_i(\theta_f) \right|\right|}$$

Attractive fields

- Simple potential field: *conic well potential*

$$U_{att_i}(\theta) = \zeta_i \, ||O_i(\theta) - O_i(\theta_f)||$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\zeta_i \nabla \, ||O_i(\theta) - O_i(\theta_f)|| = -\zeta_i \frac{O_i(\theta) - O_i(\theta_f)}{||O_i(\theta) - O_i(\theta_f)||}$$

  - it is a ζ_i-norm vector pointing to the objective

Attractive fields

- Simple potential field: *conic well potential*

$$U_{att_i}(\theta) = \zeta_i \,||O_i(\theta) - O_i(\theta_f)||$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\zeta_i \nabla \,||O_i(\theta) - O_i(\theta_f)|| = -\zeta_i \frac{O_i(\theta) - O_i(\theta_f)}{||O_i(\theta) - O_i(\theta_f)||}$$

  - it is a ζ_i-norm vector pointing to the objective
  - has a singularity at the objective

Attractive fields

- Simple potential field: *conic well potential*

$$U_{att_i}(\theta) = \zeta_i \left|\left| O_i(\theta) - O_i(\theta_f) \right|\right|$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\zeta_i \nabla \left|\left| O_i(\theta) - O_i(\theta_f) \right|\right| = -\zeta_i \frac{O_i(\theta) - O_i(\theta_f)}{\left|\left| O_i(\theta) - O_i(\theta_f) \right|\right|}$$

  - it is a ζ_i-norm vector pointing to the objective
  - has a singularity at the objective
  - ζ_i is a ponderation between articulations

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \,||O_i(\theta) - O_i(\theta_f)||^2$$

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere
- Or the hybrid potential:

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere
- Or the hybrid potential:
  - $U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere
- Or the hybrid potential:
  - $U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
  - $U_{att_i}(\theta) = -d\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| - \frac{1}{2}\zeta_i d^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere
- Or the hybrid potential:
  - $U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
  - $U_{att_i}(\theta) = -d\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| - \frac{1}{2}\zeta_i d^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
- The corresponding force is:

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere
- Or the hybrid potential:
  - $U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
  - $U_{att_i}(\theta) = -d\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| - \frac{1}{2}\zeta_i d^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
- The corresponding force is:
  - $F_{att_i}(\theta) = -\zeta_i(O_i(\theta) - O_i(\theta_f))$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$

Attractive fields

- Instead we use: *parabolic well potential*

$$U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$$

- The corresponding force is:

$$F_{att_i}(\theta) = -\nabla \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| = -\zeta_i(O_i(\theta) - O_i(\theta_f))$$

  - this force is defined everywhere
- Or the hybrid potential:
  - $U_{att_i}(\theta) = \frac{1}{2}\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
  - $U_{att_i}(\theta) = -d\zeta_i \left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| - \frac{1}{2}\zeta_i d^2$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
- The corresponding force is:
  - $F_{att_i}(\theta) = -\zeta_i(O_i(\theta) - O_i(\theta_f))$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$
  - $F_{att_i}(\theta) = -d\zeta_i \frac{O_i(\theta) - O_i(\theta_f)}{\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right|}$ if $\left|\left|O_i(\theta) - O_i(\theta_f)\right|\right| \leq d$

Repulsive fields

- Again, one repulsive field by articulation is given

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$
- Define the repulsive field:

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$
- Define the repulsive field:
  - $U_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$
- Define the repulsive field:
  - $U_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$
  - $U_{rep_i}(\theta) = \frac{\zeta_i}{2}\left(\frac{1}{d(\theta, O)} - \frac{1}{\rho_0}\right)^2$ if $d(\theta, O) \leq \rho_i$

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$
- Define the repulsive field:
  - $U_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$
  - $U_{rep_i}(\theta) = \frac{\zeta_i}{2}\left(\frac{1}{d(\theta, O)} - \frac{1}{\rho_0}\right)^2$ if $d(\theta, O) \leq \rho_i$
- The corresponding fictive force is:

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$
- Define the repulsive field:
  - $U_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$
  - $U_{rep_i}(\theta) = \frac{\zeta_i}{2}\left(\frac{1}{d(\theta, O)} - \frac{1}{\rho_0}\right)^2$ if $d(\theta, O) \leq \rho_i$
- The corresponding fictive force is:
  - $F_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$

Repulsive fields

- Again, one repulsive field by articulation is given
- Should *strongly* repel the robot close to obstacles
- Usually, should not have any influence far from the obstacle
- First define a radius of influence $\rho_i > \rho_0$
- Define the repulsive field:
  - $U_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$
  - $U_{rep_i}(\theta) = \frac{\zeta_i}{2}\left(\frac{1}{d(\theta, O)} - \frac{1}{\rho_0}\right)^2$ if $d(\theta, O) \leq \rho_i$
- The corresponding fictive force is:
  - $F_{rep_i}(\theta) = 0$ if $d(\theta, O) > \rho_i$
  - $F_{rep_i}(\theta) = -\zeta_i\left(\frac{1}{d(\theta, O)} - \frac{1}{\rho_0}\right) d(\theta, O)^{-2}\nabla d(\theta, O)$ if $d(\theta, O) \leq \rho_i$

From attractive/repulsive forces to actuator torques

- The total joint torques acting on a robot is the sum of the torques from all attractive and repulsive potentials:

$$\tau(\theta) = \sum_i J_{O_i}^T(\theta)\left(F_{att_i}(\theta) + F_{rep_i}(\theta)\right)$$

Gradient descent

- Now that we can formulate the total torques acting on the joints in the configuration space due to the artificial potentials, we can formulate a path planning algorithm

Gradient descent

- Now that we can formulate the total torques acting on the joints in the configuration space due to the artificial potentials, we can formulate a path planning algorithm
  1. First, determine your initial configuration

Gradient descent

- Now that we can formulate the total torques acting on the joints in the configuration space due to the artificial potentials, we can formulate a path planning algorithm
  1. First, determine your initial configuration
  2. Second, given a desired point in the workspace, calculate the final configuration using the inverse kinematics: Use this to create an attractive potential field

Gradient descent

- Now that we can formulate the total torques acting on the joints in the configuration space due to the artificial potentials, we can formulate a path planning algorithm
  1. First, determine your initial configuration
  2. Second, given a desired point in the workspace, calculate the final configuration using the inverse kinematics: Use this to create an attractive potential field
  3. Locate obstacles in the workspace: Create a repulsive potential field

Gradient descent

- Now that we can formulate the total torques acting on the joints in the configuration space due to the artificial potentials, we can formulate a path planning algorithm
  1. First, determine your initial configuration
  2. Second, given a desired point in the workspace, calculate the final configuration using the inverse kinematics: Use this to create an attractive potential field
  3. Locate obstacles in the workspace: Create a repulsive potential field
  4. Sum the joint torques in the configuration space

Gradient descent

- Now that we can formulate the total torques acting on the joints in the configuration space due to the artificial potentials, we can formulate a path planning algorithm
  1. First, determine your initial configuration
  2. Second, given a desired point in the workspace, calculate the final configuration using the inverse kinematics: Use this to create an attractive potential field
  3. Locate obstacles in the workspace: Create a repulsive potential field
  4. Sum the joint torques in the configuration space
  5. Use gradient descent to reach your target configuration

Gradient descent

1. $i = 0,\ \theta[0] = \theta_0$
2. if $||\theta[i] - \theta_f|| > \varepsilon$, then:
   - $\theta[i+1] = \theta[i] + \alpha[i] \dfrac{\tau(\theta[i])}{||\tau(\theta[i])||}$
   - $i = i + 1$
   - goto 2

   else:
   - return $\theta[0], \dots, \theta[i]$

- Many other algorithm are possible
  - steepest descent (gradient) (Euler)
  - Newton
  - ... see optimization books
- the $\theta[0], \dots, \theta[i]$ are the successive configuration to track = path
- It is possible to add random to escape local minima

Probabilistic roadmap

- Randomly sample the configuration space

Probabilistic roadmap

- Randomly sample the configuration space
- Enables to roughly separate Q_f from O

Probabilistic roadmap

- Randomly sample the configuration space
- Enables to roughly separate Q_f from O
- Discards the points "too close" from O

Probabilistic roadmap

- Randomly sample the configuration space
- Enables to roughly separate Q_f from O
- Discards the points "too close" from O
- Connect using straight line segments that do not intersect obstacles

Probabilistic roadmap

- Randomly sample the configuration space
- Enables to roughly separate Q_f from O
- Discards the points "too close" from O
- Connect using straight line segments that do not intersect obstacles
- Eventually resample until Q_f is sufficiently covered

Probabilistic roadmap

- Randomly sample the configuration space
- Enables to roughly separate Q_f from O
- Discards the points "too close" from O
- Connect using straight line segments that do not intersect obstacles
- Eventually resample until Q_f is sufficiently covered
- Chose the path in the connected space

Some final remarks

- All the previous methods assume an a priori knowledge of the environnement

Some final remarks

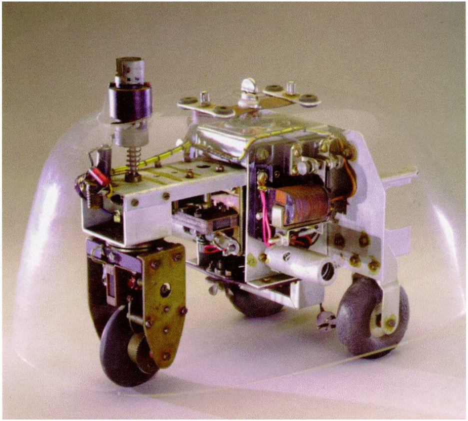
- All the previous methods assume an a priori knowledge of the environnement
- Predictive control can also be used to handle constraints "on line"

Some final remarks

- All the previous methods assume an a priori knowledge of the environnement
- Predictive control can also be used to handle constraints "on line"
- Adding fictive force is a very power tool also widely used in formation control or robotics with communication constraints (mainly range)

Mobile robotics

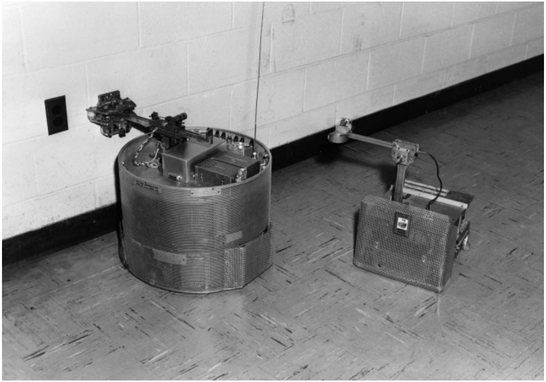
- Born in the 50s, aiming to *autonomously moving* robots



Grey Walter's "Turtle" (machina speculatrix): attracted by light

Mobile robotics

- Born in the 50s, aiming to autonomous mobile robots



John Hopkins Univ. "Beast" robot: first use of transistor based sensing (ultrasound and photodiodes)

Mobile robotics

- Born in the 50s, aiming to autonomous mobile robots

Shakey robot from Stanford Univ.
Platform used to show first results on AI (1969)

Mobile robotics

- Bio inspired locomotion: first biped robot

Honda E0 first biped robot (1986)

Mobile robotics

- Bio inspired locomotion: first biped walk

Rabbit robot CNRS-Grenoble (2004)

Mobile robotics

- Bio inspired locomotion: more about mobility

Boston Dynamics (SoftBank)

Mobile robotics

- SLAM: Simultaneous localization and mapping

https://github.com/erik-nelson/blam

Mobile robotics

- SLAM: Simultaneous localization and mapping

Mobile robotics

- Aerial robotics

Mobile robotics

- Bionics

Basis of navigation

Navigation gathers different problems

Approach of a given visible target, going to the target. Each new sensing produces an action. Typically what some insects do.
Usually based on a gradient approach

Guidance Ability to go to some position characterized by a visible environnement.
Usually based on a gradient approach

To goal navigation In that case, the target don't need to be visible but the robot has a representation of the world.
Graph or gradient approach

Topological navigation Same as previous one with a memory of the possible the spatial relationship between positions: the robot can go back)
Graph or gradient approach

Metric navigation Same as above but the robot is capable to memorize the metric positions: the robot can go back to a point without taking the same path.

- The 3 first strategies: reactiv navigation
- The 2 last enable trajectory planification also called path planning

Basis of navigation

Three key words of navigation

- Navigation relies on

Perception	where am i ?
Planification	where should i go ?
Action	how can i move ?

- The order of Perception/Planification/Action is not trivial
- Sometimes it may be necessary to move to see where to go: perception depends upon control
- Sometimes it may be necessary see to know where to go: navigation depends upon perception

Basis of navigation

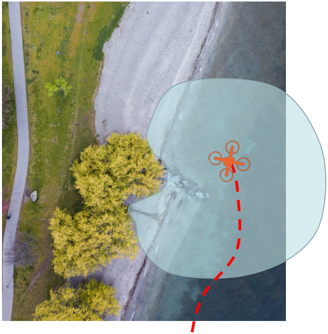
the perception

- Two kind of perceptions:
  - **Proprioceptive information** Everything that the robot can measure independently from the environnement, typically the rotation of its wheels, accelerometers, gyrometers, etc.
  - **Exterioceptive information** Everything that the robot sense in the outside world, typically distance to obstacles. Sensors are cameras, infrared/laser/ultra sound sensors, etc.
- Two type of problems
  - **Perception variability** The perception of the same place can vary (e.g. because of the sun)
  - **Perceptual aliasing** The same perception signals can correspond to 2 different places
- Perception is merged via a fusion step

Basis of navigation

the perception

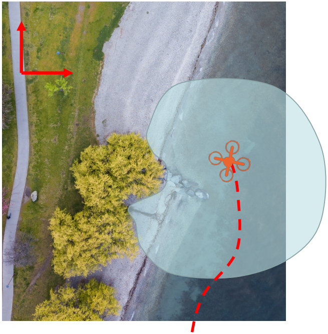
- Different kind of usage of perception information
  - Direct



Basis of navigation

the perception

- Different kind of usage of perception information
  - Direct
  - To built a metric map



Basis of navigation

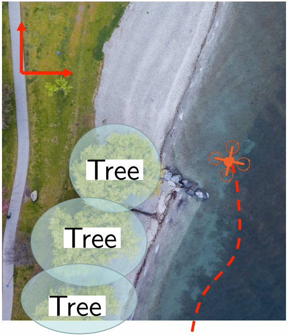
the perception

- Different kind of usage of perception information
  - Direct
  - To built a metric map
  - To built a metric map with objects

Basis of navigation

the perception

- Different kind of usage of perception information
  - Direct
  - To built a metric map
  - To built a metric map with objects
  - To built a metric map with objects of known typology



Basis of navigation

Navigation key words of navigation

- Navigation relies on

Perception	where am i ?
Planification	where should i go ?
Action	how can i move ?

- The order of Perception/Planification/Action is not trivial
- Sometimes it may be necessary to move to see where to go: perception depends upon control
- Sometimes it may be necessary see to know where to go: navigation depends upon perception

Introduction to visual servoing

- An arm robot equipped with a camera

Introduction to visual servoing

- An arm robot equipped with a camera
- Aim: bring the final effector to a given predefined configuration

Introduction to visual servoing

- An arm robot equipped with a camera
- Aim: bring the final effector to a given predefined configuration
- The configuration is defined by a *final* image feature to reach

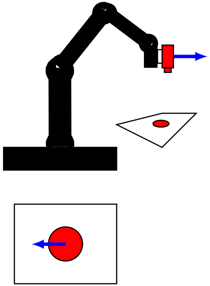
Introduction to visual servoing

- An arm robot equipped with a camera
- Aim: bring the final effector to a given predefined configuration
- The configuration is defined by a *final* image feature to reach
- Two possible configurations

Introduction to visual servoing

- An arm robot equipped with a camera
- Aim: bring the final effector to a given predefined configuration
- The configuration is defined by a *final* image feature to reach
- Two possible configurations

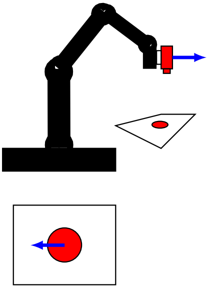
- Eye in hand configuration



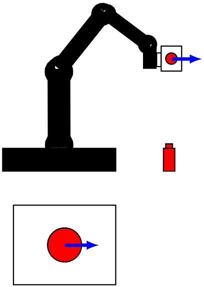
Introduction to visual servoing

- An arm robot equipped with a camera
- Aim: bring the final effector to a given predefined configuration
- The configuration is defined by a *final* image feature to reach
- Two possible configurations

- Eye in hand configuration



- Eye to hand configuration



Introduction to visual servoing

The key points

- Being able to extract feature from the image: "recognize" points of the object
- Being able to characterize the relation between the robot movement and the image changes

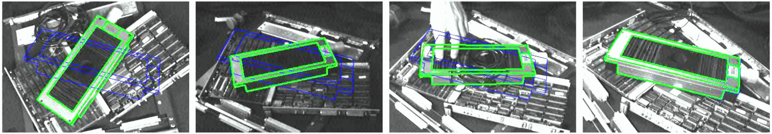
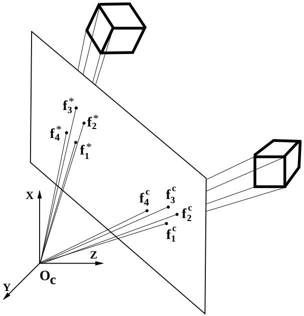


Image based visual servoing

The interaction matrix



The interaction matrix links the mouvement of O_c (lateral and rotational) to the movement of the feature points (f_i^c)

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^\star$$

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^{\star}$$

  - s denotes the current feature depending upon

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^\star$$

  - s denotes the current feature depending upon
    - the robot configuration $q(t)$

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^\star$$

  - s denotes the current feature depending upon
    - the robot configuration $q(t)$
    - a set of parameters a gathering all additional information (coarse camera intrinsic parameters, three-dimensional model of objects, etc.)

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^{\star}$$

  - s denotes the current feature depending upon
    - the robot configuration $q(t)$
    - a set of parameters a gathering all additional information (coarse camera intrinsic parameters, three-dimensional model of objects, etc.)
  - $s^{\star}$ denotes the target feature

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^\star$$

  - s denotes the current feature depending upon
    - the robot configuration $q(t)$
    - a set of parameters a gathering all additional information (coarse camera intrinsic parameters, three-dimensional model of objects, etc.)
  - $s^\star$ denotes the target feature
- The relation between the image and the real world is given by the interaction matrix:

$$\dot{s} = L_s \nu_c$$

where

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^\star$$

  - s denotes the current feature depending upon
    - the robot configuration $q(t)$
    - a set of parameters a gathering all additional information (coarse camera intrinsic parameters, three-dimensional model of objects, etc.)
  - $s^\star$ denotes the target feature
- The relation between the image and the real world is given by the interaction matrix:

$$\dot{s} = L_s \nu_c$$

where

  - $\nu_c := (v_c, \omega_c) = (\text{linear veloc}_{\text{cam frame}}, \text{angular veloc}_{\text{cam frame}})$

Introduction to visual servoing

A short mathematical background

- Positioning error:

$$e(t) = s(q(t), a) - s^\star$$

  - s denotes the current feature depending upon
    - the robot configuration $q(t)$
    - a set of parameters a gathering all additional information (coarse camera intrinsic parameters, three-dimensional model of objects, etc.)
  - $s^\star$ denotes the target feature
- The relation between the image and the real world is given by the interaction matrix:

$$\dot{s} = L_s \nu_c$$

where

  - $\nu_c := (v_c, \omega_c) = (\text{linear veloc}_{\text{cam frame}}, \text{angular veloc}_{\text{cam frame}})$
  - $L_s \in \mathbb{R}^{k\times 6}$: interaction matrix (Jacobian)

Control in visual servoing

A simple control approach

- Coupling the error and the interaction relation, one gets:

$$\dot{e} = L_s \nu_c$$

Control in visual servoing

A simple control approach

- Coupling the error and the interaction relation, one gets:

$$\dot{e} = L_s \nu_c$$

- Take the linear velocities and angular velocities as control variable

Control in visual servoing

A simple control approach

- Coupling the error and the interaction relation, one gets:

$$\dot{e} = L_s \nu_c$$

- Take the linear velocities and angular velocities as control variable
- Let $L_s^+ := (L_s^T L_s)^{-1} L_s^T$ be the Moore–Penrose pseudo-inverse of L_s

Control in visual servoing

A simple control approach

- Coupling the error and the interaction relation, one gets:

$$\dot{e} = L_s \nu_c$$

- Take the linear velocities and angular velocities as control variable
- Let $L_s^+ := (L_s^T L_s)^{-1} L_s^T$ be the Moore–Penrose pseudo-inverse of L_s
- To force an exponential decrease of the error:

$$\dot{e} = -\lambda e$$

we must chose

$$\nu_c := -\lambda L_s^+ e$$

Control in visual servoing

A simple control approach

- Coupling the error and the interaction relation, one gets:

$$\dot{e} = L_s \nu_c$$

- Take the linear velocities and angular velocities as control variable
- Let $L_s^+ := (L_s^T L_s)^{-1} L_s^T$ be the Moore–Penrose pseudo-inverse of L_s
- To force an exponential decrease of the error:

$$\dot{e} = -\lambda e$$

we must chose

$$\nu_c := -\lambda L_s^+ e$$

- Practically, L_s is never known perfectly and we use an approximation

Image-based visual servoing

- Take a 3D point of coordinates $P = (X, Y, Z)$ in the camera frame

Image-based visual servoing

- Take a 3D point of coordinates $P = (X, Y, Z)$ in the camera frame
- Its coordinates in the image will be $p = (x, y)$:

$$\begin{aligned} x &= X/Z = (u - c_u)/f\alpha \\ y &= Y/Z = (v - c_v)/f \end{aligned}$$

where f is the focal length, α is the ratio of the pixel dimensions, c_u and c_v are the coordinates of the principal point.

Image-based visual servoing

- Take a 3D point of coordinates $P = (X, Y, Z)$ in the camera frame
- Its coordinates in the image will be $p = (x, y)$:

$$
\begin{aligned}
x &= X/Z = (u - c_u)/f\alpha \\
y &= Y/Z = (v - c_v)/f
\end{aligned}
$$

  where f is the focal length, α is the ratio of the pixel dimensions, c_u and c_v are the coordinates of the principal point.
- Derivating, we get

$$
\begin{aligned}
\dot{x} &= \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \\
\dot{y} &= \dot{Y}/Z - Y\dot{Z}/Z^2 = (\dot{Y} - y\dot{Z})/Z
\end{aligned}
$$

Image-based visual servoing

- Using the Varignon's formula

$$\dot{X} = -v_c - \omega_c^{\times} X$$

Image-based visual servoing

- Using the Varignon's formula

$$\dot{X} = -v_c - \omega_c^{\times} X$$

- Mixing the two last equation, we get the interaction matrix form P

$$\dot{p} = L_p \nu_c$$

with

$$L_p = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{pmatrix}$$

Image-based visual servoing

- Using the Varignon's formula

$$\dot{X} = -v_c - \omega_c^{\times} X$$

- Mixing the two last equation, we get the interaction matrix form P

$$\dot{p} = L_p \nu_c$$

with

$$L_p = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{pmatrix}$$

- Z is the depth and is usually not known

Image-based visual servoing

- Using the Varignon's formula

$$\dot{X} = -v_c - \omega_c^{\times} X$$

- Mixing the two last equation, we get the interaction matrix form P

$$\dot{p} = L_p \nu_c$$

with

$$L_p = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{pmatrix}$$

- Z is the depth and is usually not known
- To control six degrees of freedom, at least three points are required (p_1, p_2, p_3)

Image-based visual servoing

- Using the Varignon's formula

$$\dot{X} = -v_c - \omega_c^\times X$$

- Mixing the two last equation, we get the interaction matrix form P

$$\dot{p} = L_p \nu_c$$

with

$$L_p = \begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{pmatrix}$$

- Z is the depth and is usually not known
- To control six degrees of freedom, at least three points are required (p_1, p_2, p_3)
- Camera parameters can be obtained by calibration

Image-based visual stereo servoing

- We assume now that we have two cameras