Imperial College London

SMART ROBOTICS LAB

01/02/2021

Lecture 3: Kinematics and Temporal Models

70003 Advanced Robotics

Dr Stefan Leutenegger

Teaching Assistants: Dr Masha Popovic, Sotiris Papatheodorou, Binbin Xu, and Nils Funk

Representing Orientation (SO3): Rotation Matrix

Orientation in 3D Euclidean space (SO3) has 3 Degrees of Freedom (DoF).

You can express it using a rotation matrix, e.g. $\mathbf{C}_{WS}$:

- The rotation matrix changes the coordinate representation of a vector as
  $${}_W\mathbf{a} = \mathbf{C}_{WS}\ {}_S\mathbf{a}.$$
- The rotation matrix is orthonormal. Consequently:
  $$\mathbf{C}_{SW} = \mathbf{C}_{WS}^{-1} = \mathbf{C}_{WS}^{T},$$
  $$\det\mathbf{C}_{SW} = 1.$$
- The columns of $\mathbf{C}_{WS}$ are the basis vectors of $\underrightarrow{\mathcal{F}}_S$ expressed in $\underrightarrow{\mathcal{F}}_W$. (This is handy to know for visualisations.)

Advantages of this parameterisation:

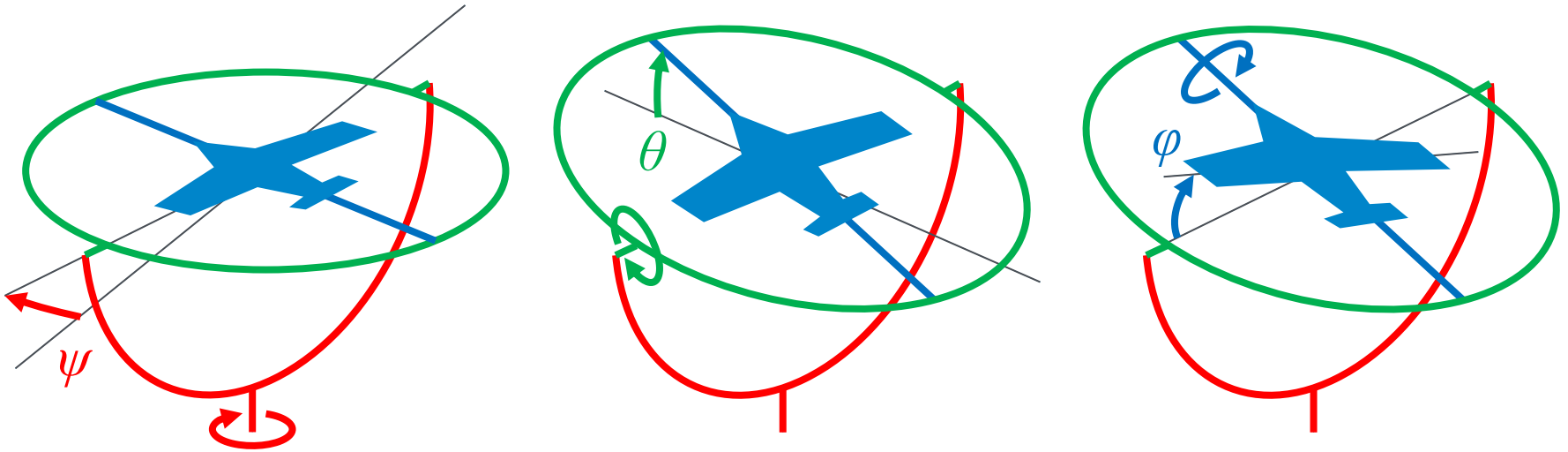
- No singularities

Drawbacks of this parameterisation:

- 9 parameters to represent 3 DoF (overkill).
- Must enforce the above orthonormality constraints.

Representing Orientation (SO3): Euler Angles (Tait-Brian)

Rotation Matrix (e.g. $\mathbf{C}_{EB}$) is parametrized with **3 successive rotations** using the **zyx Tait-Brian Angles** (specific kind of Euler Angles):



1 **Yaw:** ψ

2 **Pitch:** θ

3 **Roll:** φ

Euler Angles (Tait-Brian): Gimbal Lock

What happens at pitch angle $\theta = \pm\pi/2$?

- Yaw and roll axes align…
- 1 DoF is thus lost at this point
- It is a singularity!
- **Note:** we can still describe this orientation (just not uniquely – set yaw or roll to zero)

Advantages of Tait-Bryan parameterisation:

- Minimal representation, thus no constraints to enforce.
- Reasonably easy to interpret, e.g. in a plot against time.

Drawbacks of this parameterisation:

- **"Gimbal Lock singularity"**
  (all Euler angle representations will have a singularity)

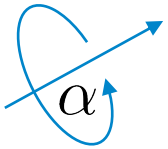
Quaternions of Orientation [9]

Hamiltonian Quaternion:

A general quaternion is a complex number (real part and a 3D imaginary part):
$q = q = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$, with $\mathbf{i}^2=\mathbf{j}^2=\mathbf{k}^2=-1$, $\mathbf{ij}=-\mathbf{ji}=\mathbf{k}$, $\mathbf{jk}=-\mathbf{kj}=\mathbf{i}$, $\mathbf{ki}=-\mathbf{ik}=\mathbf{j}$.

Unit Quaternion (Q. of Orientation):

Again: think of orientation as rotation around a normal $\mathbf{n}$ by an angle α:



$$\mathbf{v} = \sin\frac{\alpha}{2}\mathbf{n}, \quad a = \cos\frac{\alpha}{2}$$

$$\mathbf{q} = [q_x, q_y, q_z, q_w]^T = [\mathbf{v}(\mathbf{q})^T, a(\mathbf{q})]^T$$

I.e. we use 4 parameters for 3 DoF.

Unit length constraint: $\|\mathbf{q}\| = 1.$

Neutral element: $\boldsymbol{\iota} = [0, 0, 0, 1]^T.$

Inverse: $\mathbf{q}^{-1} = [-\mathbf{v}(\mathbf{q})^T, a(\mathbf{q})]^T.$

[9] kindr cheatsheet, available at https://github.com/ethz-asl/kindr/blob/master/doc/cheatsheet/cheatsheet_latest.pdf, March 2015.

SE3 and Homogenous Transformation Matrices

A 6D pose consists of position and orientation.
We can write the respective frame transformation of a position vector $\mathbf{r}_P$ as:

$$ {}_A\mathbf{r}_P = \mathbf{C}_{AB}\, {}_B\mathbf{r}_P + {}_A\mathbf{r}_B. $$

Or, we can use the homogeneous transformation matrix:

$$ \boldsymbol{T}_{AB} = \begin{bmatrix} \mathbf{C}_{AB} & {}_A\mathbf{r}_B \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}. $$

Now: $ {}_A\boldsymbol{r}_P := \begin{bmatrix} {}_A\mathbf{r}_P \\ 1 \end{bmatrix} = \boldsymbol{T}_{AB} \begin{bmatrix} {}_B\mathbf{r}_P \\ 1 \end{bmatrix}. $

Notes

As inverse transform, we obtain $ \boldsymbol{T}_{BA} = \boldsymbol{T}_{AB}^{-1} = \begin{bmatrix} \mathbf{C}_{AB}^T & -\mathbf{C}_{AB}^T\, {}_A\mathbf{r}_B \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}. $

We can stack transformations as $ \boldsymbol{T}_{AC} = \boldsymbol{T}_{AB}\, \boldsymbol{T}_{BC}. $

The Multivariate Gaussian Distribution

We write $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

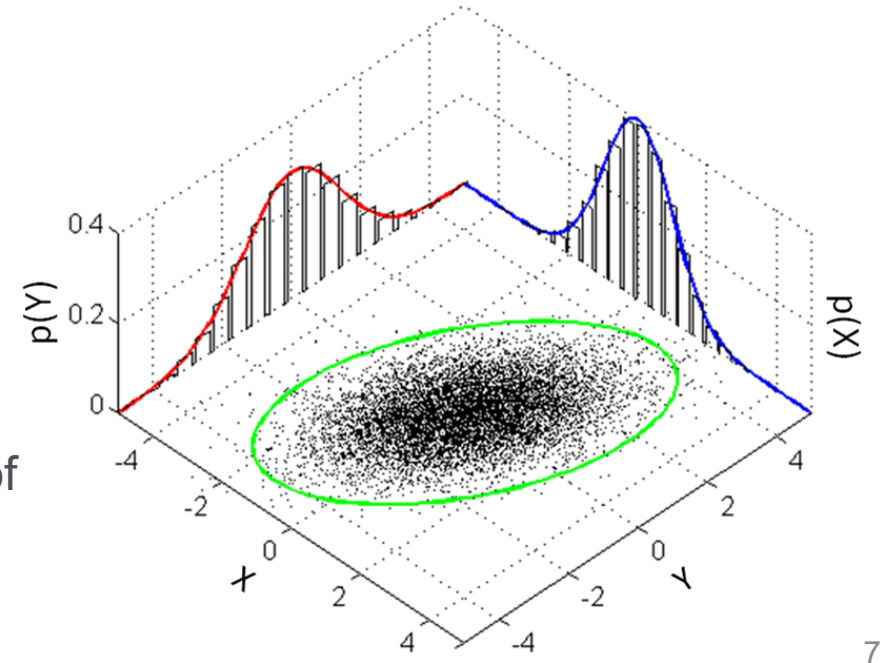
Probability density function:

$$p(\mathbf{x}) = p(x_1, \ldots, x_k)$$

$$= \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right).$$

$\boldsymbol{\mu}$: Mean

$\boldsymbol{\Sigma}$: Covariance matrix



Right: Samples from Gaussian distribution of two variables X and Y.

Measurement Models: IMU and in General

A MEMS IMU measures **rotation rates** ${}_S\tilde{\boldsymbol{\omega}}$ and **acceleration** ${}_S\tilde{\mathbf{a}}$.

We model a (discrete-time) measurement as:

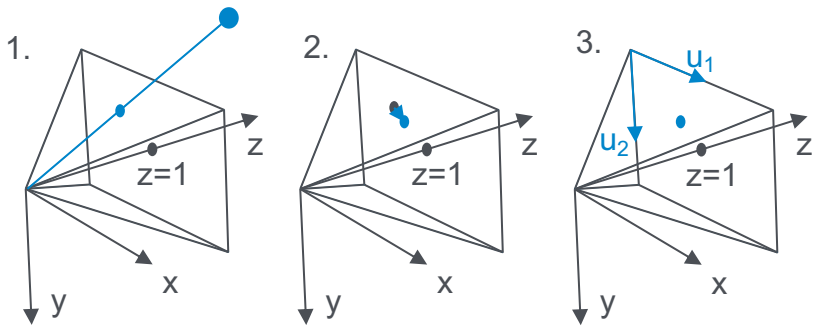
$$\tilde{\mathbf{z}} = \underbrace{\mathbf{b}_C}_{\text{Constant, calibrate.}} + \underbrace{s\mathbf{M}\mathbf{z}}_{\text{Often constant, calibrate.}} + \underbrace{\mathbf{b}}_{\text{Model \& estimate!}} + \underbrace{\mathbf{n}}_{\text{Model!}} + \mathbf{o}.$$

- $\mathbf{z}$: Correct measurement
- $\mathbf{b}_C$: Long-term constant bias
- s : Scaling
- $\mathbf{M}$: Misalignment
- $\mathbf{b}$: Time-varying bias (often also a function of temperature)
- $\mathbf{n}$: Noise, typically modelled as $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$
- $\mathbf{o}$: Other (un-modelled) influences

Note: We will model many sensors in a similar way…

Pinhole Camera Projection with Distortion[1] (World2Cam)

Complete Model: $\mathbf{u} = \mathbf{k}(\mathbf{d}(\mathbf{p}({}_C\mathbf{r}_P)))$



1. Project point to unit plane:

$$\mathbf{x}' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \mathbf{p}\left(\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \right) = \frac{1}{r_3} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}.$$

2. Apply distortion model – e.g. Radial-Tangential (with $r^2 = x_1'^2 + x_2'^2$):

$$\mathbf{x}'' = \mathbf{d}(\mathbf{x}') = \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} + \begin{bmatrix} 2p_1 x_1' x_2' + p_2(r^2 + 2x_1'^2) \\ p_1(r^2 + 2x_2'^2) + 2p_2 x_1' x_2' \end{bmatrix}.$$

3. Scale and centre: $\mathbf{u} = \mathbf{k}(\mathbf{x}'') = \begin{bmatrix} f_1 & 0 \\ 0 & f_2 \end{bmatrix} \mathbf{x}'' + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}.$

k_1, k_2 : Radial distortion parameters.
k_3-k_6 : (Optional) radial distortion parameters.
p_1, p_2 : Tangential distortion parameters.

f_1, f_2 : x/y focal lengths in pixels.
c_1, c_2 : Image centre in pixels.

[1] Following OpenCV, http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html .

About Un-Distorting a Whole Image

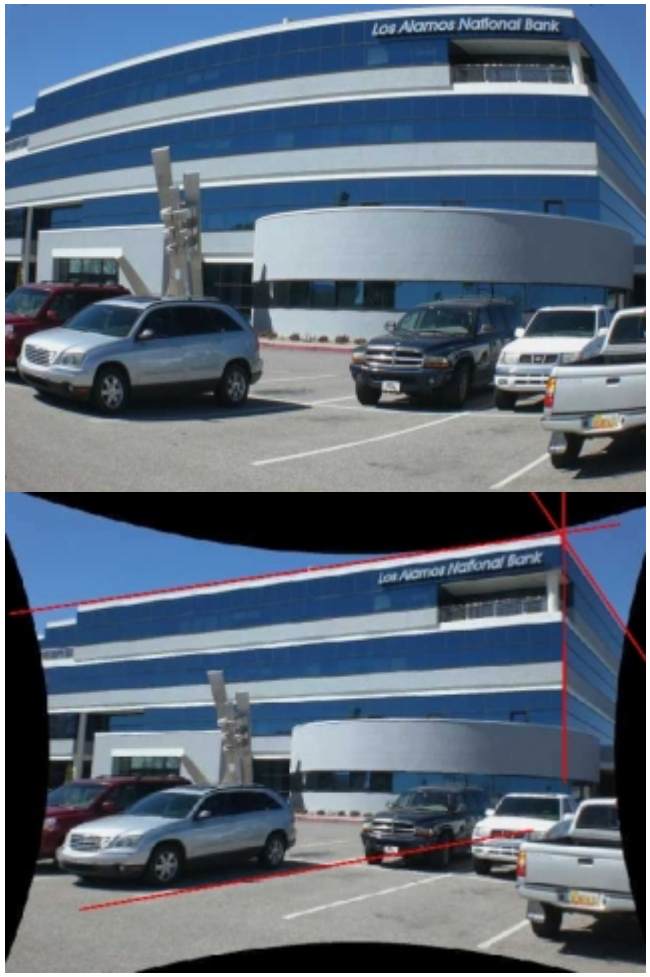


Image source: http://www.edwardrosten.com/work/ .

We can do this (maybe surprisingly) using the **distortion function** in pixel space:

1. For all pixel locations $\mathbf{u}_{i,\text{new}}$ of the desired output image, compute their distorted locations
   $$\mathbf{u}_i = \mathbf{k}(\mathbf{d}(\mathbf{k}_{\text{new}}{}^{-1}(\mathbf{u}_{i,\text{new}}))).$$
   Choose $\mathbf{k}_{\text{new}}$ with desired new focal lengths and image centre.
2. Re-sample the colours at these locations $\mathbf{u}_i$ (using some interpolation scheme between the neighbouring pixels, e.g. bi-linear).

Notes

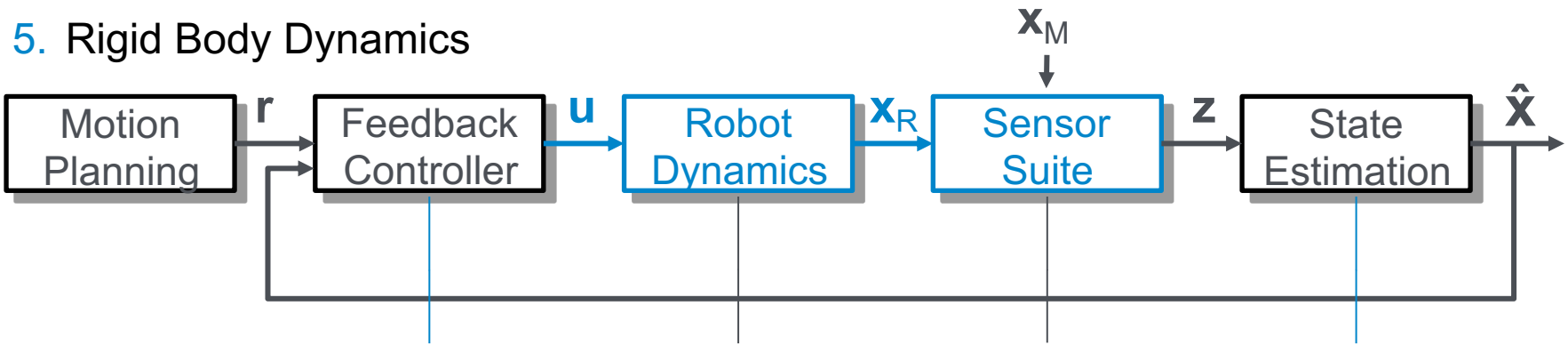
- Resampling always loses image quality.
- Depending on choices in Step 1, you may get samples from outside the original image (set e.g. black).

Schedule: Lectures

What	Date	Topic
Lecture 1	18/01/21	Introduction, Problem Formulation, and Examples
Lecture 2	25/01/21	Representations and Sensors
Lecture 3	**01/02/21**	**Kinematics and Temporal Models**
Lecture 4	08/02/21	The Extended Kalman Filter
Lecture 5	15/02/21	Feedback Control
Lecture 6	22/02/21	Nonlinear Least Squares
Lecture 7	01/03/21	Vision-Based Simultaneous Localisation and Mapping
Lecture 8	08/03/21	Revision, Q&A

Today

1. Linear vs. Nonlinear Functions and Systems
2. Continuous-Time and Discrete-Time Dynamic Systems
3. Sensor Models
4. Kinematics Examples
5. Rigid Body Dynamics



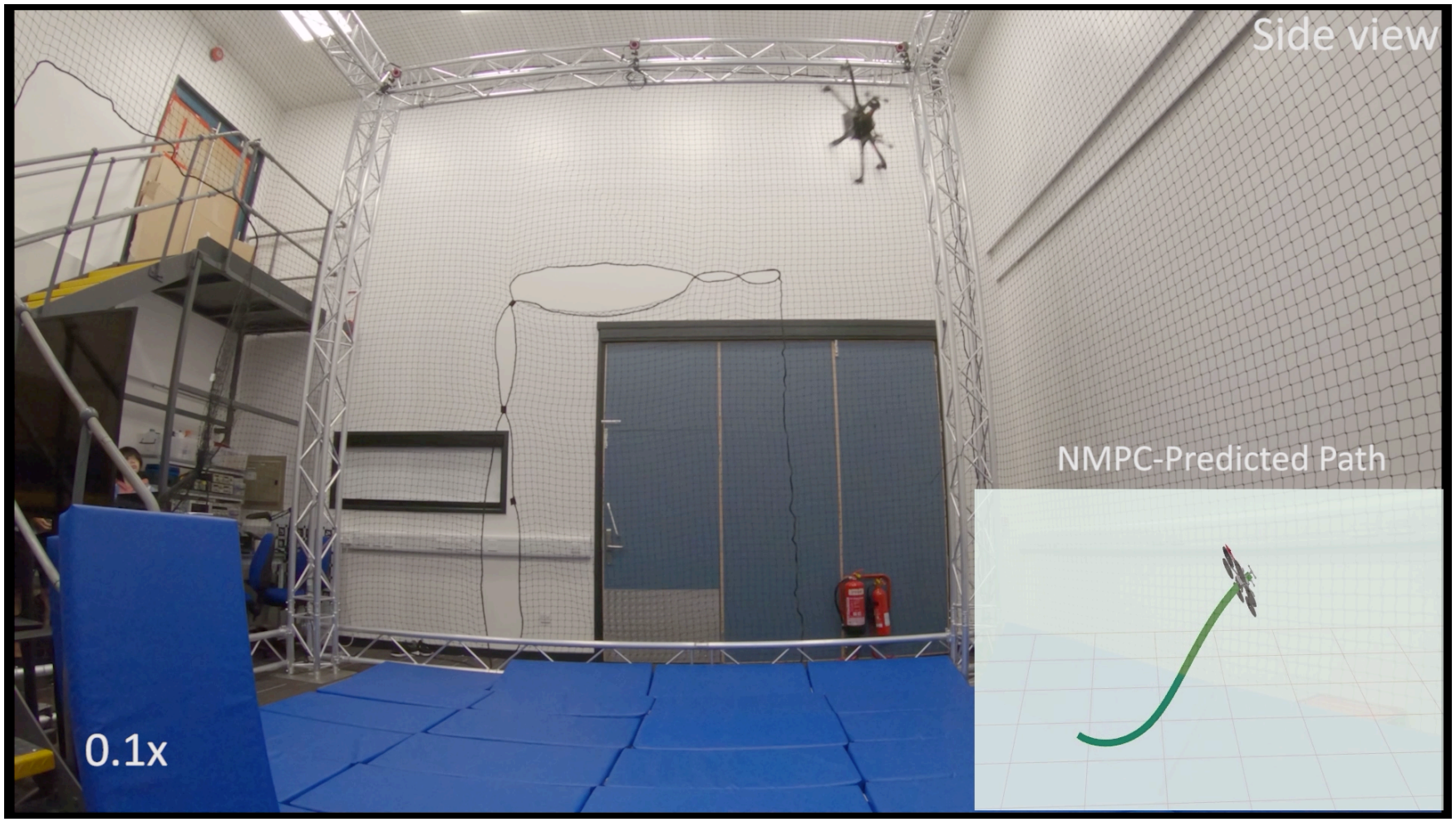
What commands should we send to the robot so it follows the reference trajectory?

How does the robot react to inputs given initial states?

How are measure-ments are generated given the robot states?

What is the best explan-ation for the robot states given the measure-ments?

Model-Based Estimation and Control Example



Starting Point: Continuous-Time Nonlinear System

u: input vector
$\mathbf{x}_R$: state vector

We often start by modelling system (robot dynamics) as a **continuous-time nonlinear** system of ordinary differential equations

$$\frac{\partial \mathbf{x}(t)}{\partial t} =: \dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}(t), \mathbf{u}(t)),$$

of which there may be measurements available of the form

$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t)).$$

But often, we will need to have a **linearised** version for system analysis and estimator formulations.

Linearising Vector-valued Functions

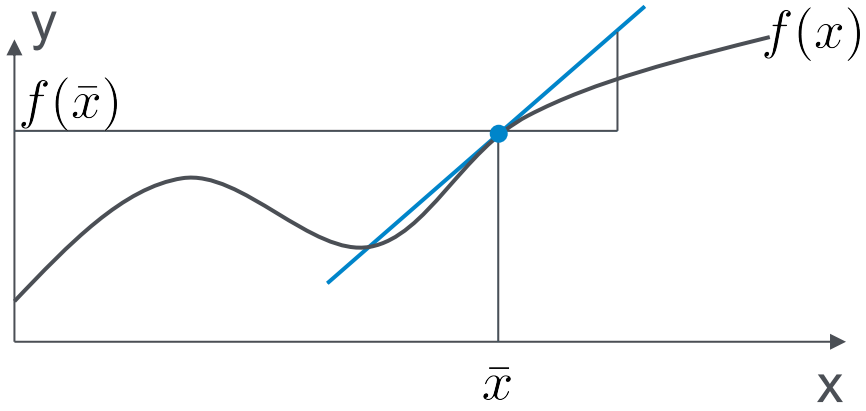
What does linearization mean?

1D case:

We approximate a function $f(x)$ around a **linearisation point** $\bar{x}$ with

$$f(x) \approx f(\bar{x}) + \left.\frac{df}{dx}\right|_{x=\bar{x}} (x - \bar{x}).$$

(First order Taylor approximation)

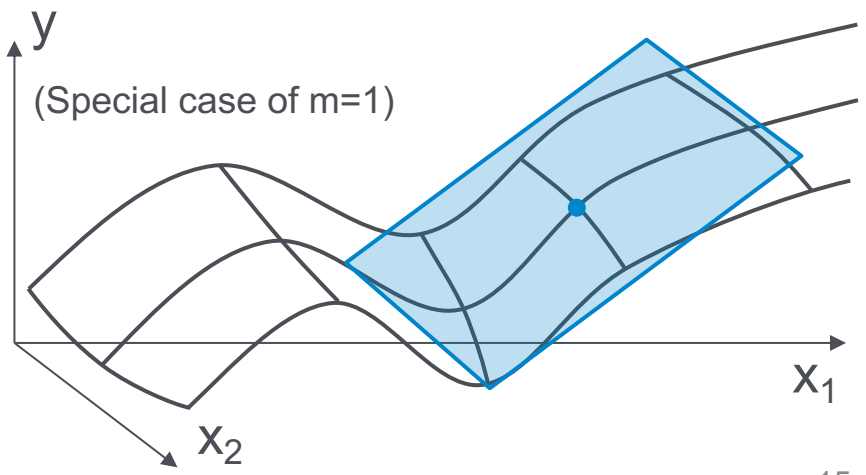


Multivariate case:

$$\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^n,$$

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\bar{\mathbf{x}}) + \mathbf{F}|_{\mathbf{x}=\bar{\mathbf{x}}} (\mathbf{x} - \bar{\mathbf{x}}).$$

Where $\mathbf{F}|_{\mathbf{x}=\bar{\mathbf{x}}} \in \mathbb{R}^{m \times n}$ is called the **Jacobian Matrix**.



Imperial College London

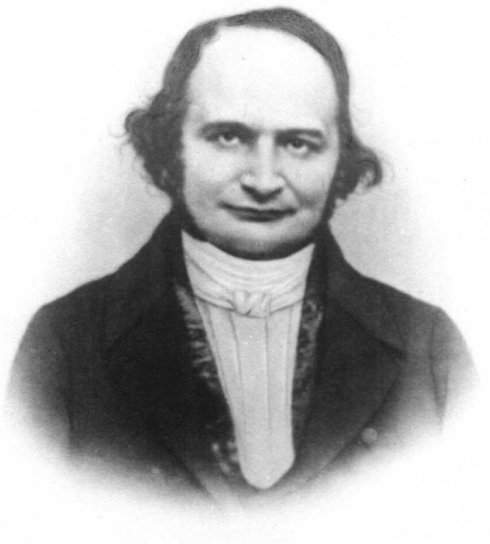
SMART ROBOTICS LAB

The Jacobian Matrix

Here is how the Jacobian is defined:

$$\mathbf{F} = \frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

Note: for the case of a scalar field, i.e. $f(\mathbf{x}) \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$, this corresponds to the gradient vector.



Carl Gustav Jacobi Jacobi
1804-1851
en.wikipedia.org/wiki/Carl_Gustav_Jacob_Jacobi

Example with m=2, n=3:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 4x_1 - 3x_2^2 \\ x_2 + 2x_3^3 \end{bmatrix}.$$

$$\mathbf{F}(\mathbf{x}) =$$

Numeric Differences

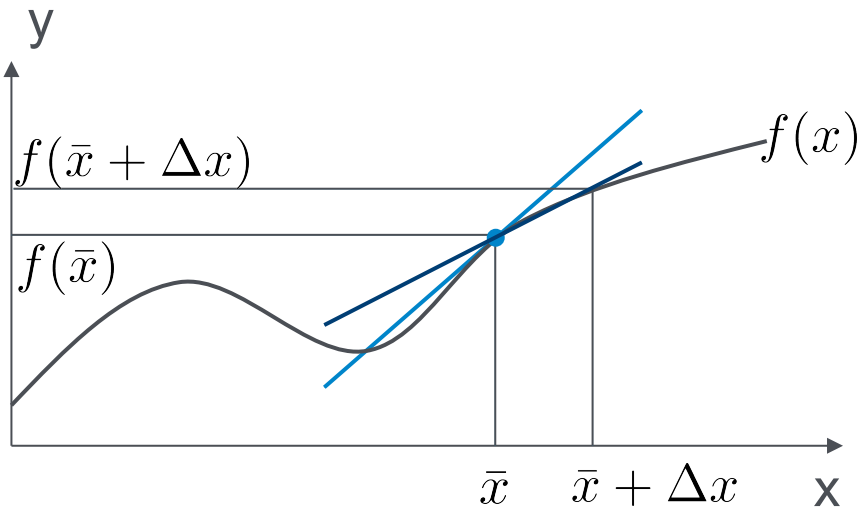
In 1D, we know the derivative (if it exists)

$$\left.\frac{df}{dx}\right|_{x=\bar{x}} = \lim_{\Delta x \to 0} \frac{f(\bar{x}+\Delta x) - f(\bar{x})}{\Delta x}.$$

Approximate this with **finite differences**:

$$\left.\frac{df}{dx}\right|_{x=\bar{x}} \approx \frac{f(\bar{x}+\Delta x) - f(\bar{x})}{\Delta x},$$

with sufficiently small Δx .



Similarly, for the multidimensional case (**forward differences**):

$$\frac{d\mathbf{f}}{d\mathbf{x}}\Big|_{\mathbf{x}=\bar{\mathbf{x}}} \approx \left[\frac{\mathbf{f}(\bar{\mathbf{x}}+\Delta x_1\mathbf{e}_1) - \mathbf{f}(\bar{\mathbf{x}})}{\Delta x_1} \quad \cdots \quad \frac{\mathbf{f}(\bar{\mathbf{x}}+\Delta x_n\mathbf{e}_n) - \mathbf{f}(\bar{\mathbf{x}})}{\Delta x_n}\right],$$

or numerically preferred with **central differences**:

$$\frac{d\mathbf{f}}{d\mathbf{x}}\Big|_{\mathbf{x}=\bar{\mathbf{x}}} \approx \left[\frac{\mathbf{f}(\bar{\mathbf{x}}+\Delta x_1\mathbf{e}_1) - \mathbf{f}(\bar{\mathbf{x}}-\Delta x_1\mathbf{e}_1)}{2\Delta x_1} \quad \cdots \quad \frac{\mathbf{f}(\bar{\mathbf{x}}+\Delta x_n\mathbf{e}_n) - \mathbf{f}(\bar{\mathbf{x}}-\Delta x_n\mathbf{e}_n)}{2\Delta x_n}\right],$$

where $\mathbf{e}_i$ denotes the unit vector in the ith dimension.

We can use this scheme to **verify the correctness of analytical Jacobians**!

Continuous-Time Dynamic Systems

Typically, we will model as a **nonlinear continuous-time system** of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)).$$
$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t)) + \mathbf{v}(t).$$

Linearise: around $\overline{\mathbf{x}}, \overline{\mathbf{u}}$

The **linearised continuous-time system** can be handy for analysis:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}_c\delta\mathbf{x}(t) + \mathbf{G}_c\delta\mathbf{u}(t) + \mathbf{L}_c\mathbf{w}(t).$$
$$\delta\mathbf{z}(t) = \mathbf{H}\delta\mathbf{x}(t) + \mathbf{v}(t).$$

Linear Time-Invariant Dynamic Systems

The evolution of states $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is described by a system of linear differential equations, where $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is a control input:

$$\dot{\mathbf{x}}(t) = \mathbf{F}_c\mathbf{x}(t) + \mathbf{G}_c\mathbf{u}(t).$$

$\mathbf{F}_c$: system matrix $\mathbf{G}_c$: input gain

Furthermore, we may consider a measurable output $\mathbf{z}(t) \in \mathbb{R}^{n_z}$:

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t).$$

$\mathbf{H}$: measurement matrix

We may want to deal with a **stochastic** system:

$$\dot{\mathbf{x}}(t) = \mathbf{F}_c\mathbf{x}(t) + \mathbf{G}_c\mathbf{u}(t) + \mathbf{L}_c\mathbf{w}(t).$$

$\mathbf{w}(t) \in \mathbb{R}^{n_w}$: process noise

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{v}(t).$$

$\mathbf{v}(t)$: measurement noise

Typically, $\mathbf{w}(t)$ will be assumed to be a zero-mean, mutually independent **Gaussian White Noise Process**.

$\mathbf{v}(t)$ will be assumed to be a zero-mean **Gaussian-Distributed** random variable.

Example Nonlinear and Linearised Dynamics

Consider the nonlinear system

$$\begin{aligned}\dot{x}_1 &= x_2, \\ \dot{x}_2 &= k_1 \sin x_2 + k_2 u_1.\end{aligned}$$

What does the linearisation look like?

$\mathbf{F}_c =$

$\mathbf{G}_c =$

Nonlinear Dynamic Systems

Typically, we will model as a **nonlinear continuous-time system** of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)).$$
$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t)) + \mathbf{v}(t).$$

Linearise: around $\overline{\mathbf{x}}, \overline{\mathbf{u}}$

The **linearised continuous-time system** can be handy for analysis:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}_c\delta\mathbf{x}(t) + \mathbf{G}_c\delta\mathbf{u}(t) + \mathbf{L}_c\mathbf{w}(t).$$
$$\delta\mathbf{z}(t) = \mathbf{H}\delta\mathbf{x}(t) + \mathbf{v}(t).$$

Discretise: integrate from t_{k-1} to t_k .

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k).$$
$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k.$$

Linearise: around $\overline{\mathbf{x}}, \overline{\mathbf{u}}$

$$\delta\mathbf{x}_k = \mathbf{F}_k\delta\mathbf{x}_{k-1} + \mathbf{G}_k\delta\mathbf{u}_k + \mathbf{L}_k\mathbf{w}_k.$$
$$\delta\mathbf{z}_k = \mathbf{H}_k\delta\mathbf{x}_k + \mathbf{v}_k.$$

The **discrete-time nonlinear system** is used in our estimator implementations

Also the **linearised discrete-time system** is needed in the implementation

Numeric Integration – Simple Methods

- We would like to obtain the state transition function for a time interval Δt.
- Nonlinear differential equations typically can't be integrated analytically.
- Thus: resort to numeric integration schemes!

Euler Forward

A simple first order method:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta t \mathbf{f}_c(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, t_{k-1}).$$

This is an instructional example – you would typically not do this.

Trapezoidal

Slightly modified Euler forward, still simple, but more accurate (second order):

$$\Delta \mathbf{x}_1 = \Delta t \mathbf{f}_c(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, t_{k-1}),$$ **evaluate at step k-1**

$$\Delta \mathbf{x}_2 = \Delta t \mathbf{f}_c(\mathbf{x}_{k-1} + \Delta \mathbf{x}_1, \mathbf{u}_k, t_k),$$ **eval. at k using Euler Fwd state prediction**

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \frac{1}{2}(\Delta \mathbf{x}_1 + \Delta \mathbf{x}_2).$$ **average the two**

Typical Sensor-Intrinsic States and Their Models

Recall the IMU sensor model: $\tilde{\mathbf{z}} = \underbrace{\mathbf{b}_C}_{\text{Constant, calibrate.}} + \underbrace{s\mathbf{M}\mathbf{z}}_{\text{Often constant, calibrate.}} + \underbrace{\mathbf{b}}_{\text{Model \& estimate!}} + \underbrace{\mathbf{n}}_{\text{Model!}} + \mathbf{o}.$

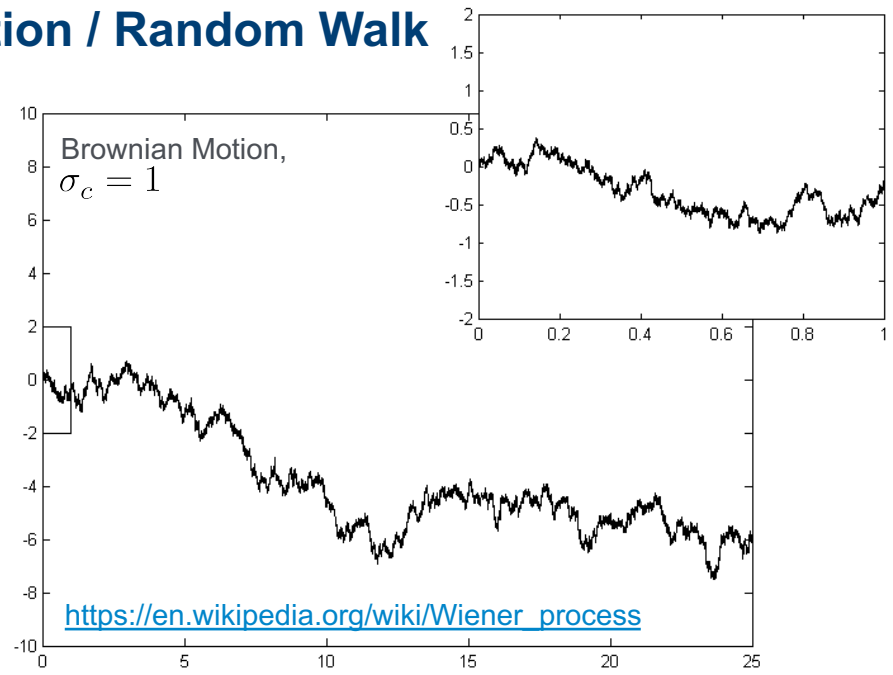
Bias $\mathbf{b}$ (and scale s, plus other params) often modelled as a time-varying state:

Scaled Wiener Process / Brownian Motion / Random Walk

$\dot{b}(t) = \sigma_c n(t).$

$n(t)$: Zero-mean Gaussian White Noise Process.

Note: $E[b(t)] = 0, \mathrm{var}[b(t)] = \sigma_c^2 t$, thus not bounded as time progresses.



Kinematics vs. Dynamics

Kinematics

- Analyses the **motion** of objects **not** considering **causes**.
- Use of rigid body (coordinate frame) **transformations** (pose parameters) and their **derivatives**.

Dynamics

- Considers effects of **forces** and **torques** on **motion**.
- Use of **Newton's Second Law** (or Lagrangian Dynamics, not covered in this course).

Control

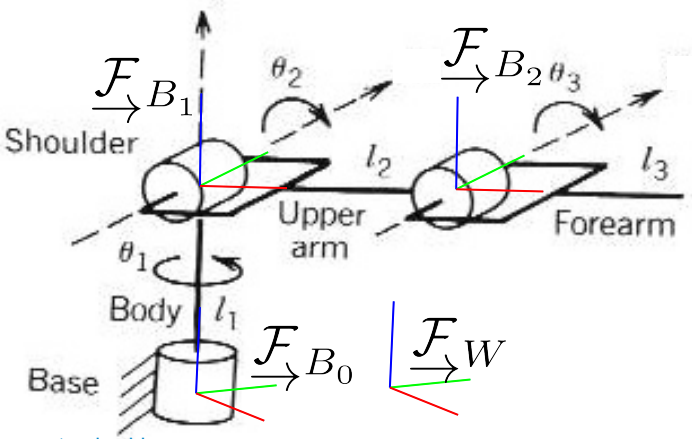
- Physically meaningful models: torques/forces inputs (use dynamics).
- But sometimes, taking pose parameters and/or their derivatives as inputs may be a good enough approximation (use kinematics).

Estimation

- We may have access to either pose parameters / their derivatives or torques (use kinematics or dynamics).

Manipulator Kinematics

Joint angles/positions uniquely define the configuration: **not a temporal model**!



www.tareksobh.org

www.directindustry.com

www.photonics.com

Forward Kinematics $\boldsymbol{T}_{WB_n}(\boldsymbol{\theta}) = ?$

Attach a frame to each rigid body (origin through joint axis) and parameterise the relative transformations $\boldsymbol{T}_{B_{i-1}B_i}(\theta_i)$:

$$\boldsymbol{T}_{WB_n}(\boldsymbol{\theta}) = \boldsymbol{T}_{WB_0}\boldsymbol{T}_{B_0B_1}(\theta_1)\ldots\boldsymbol{T}_{B_{n-1}B_n}(\theta_n).$$

Inverse Kinematics $\boldsymbol{\theta}(\boldsymbol{T}_{WB_n}) = ?$

- Often nonlinear, without analytic solution.
- Depending on the configuration (importantly no. joints): under- or over-determined.
- Use numeric (iterative) solving…

Closed Kinematic Chains / Parallel Robots

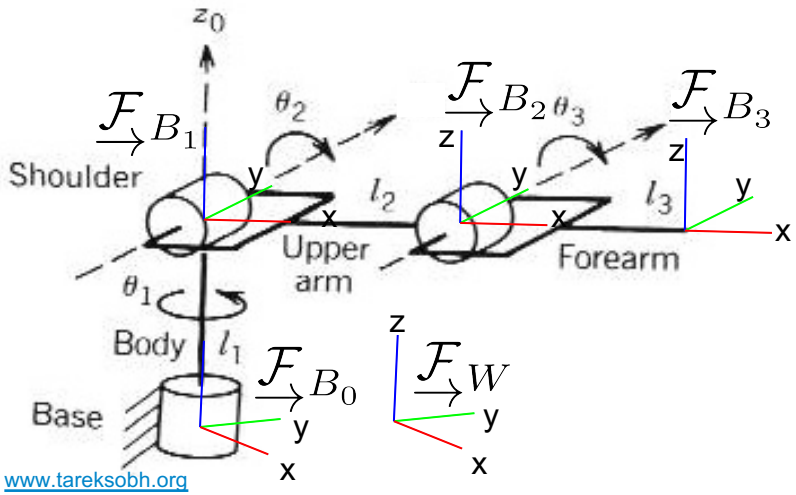
Parallel kinematics chains, joined by passive links (yielding additional constraints)

Imperial College London

SMART ROBOTICS LAB

Manipulator Forward Kinematics Example

$$\boldsymbol{T}_{WB_n}(\boldsymbol{\theta}) = \boldsymbol{T}_{WB_0}\boldsymbol{T}_{B_0B_1}(\theta_1)\boldsymbol{T}_{B_1B_2}(\theta_2)\boldsymbol{T}_{B_2B_3}(\theta_3).$$



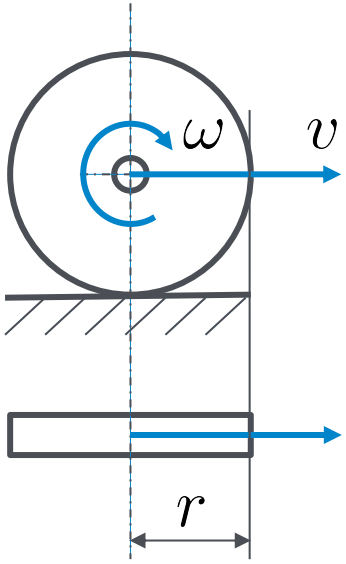
$$\boldsymbol{T}_{WB_0} =$$

$$\boldsymbol{T}_{B_0B_1}(\theta_1) =$$

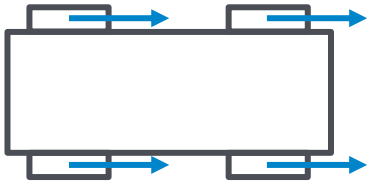
$$\boldsymbol{T}_{B_1B_2}(\theta_2) =$$

$$\boldsymbol{T}_{B_2B_3}(\theta_3) =$$

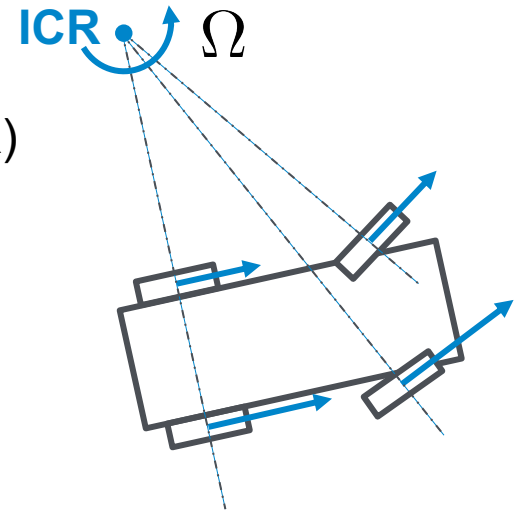
Wheel Odometry – Some Basics



- Vehicle speeds at wheel axes mid-wheel: $v_i = \omega_i r_i$
- Those velocity vectors are:
  - parallel to the ground where the wheel touches it,
  - perpendicular to the wheel rotation axis.
- For the vehicle to undergo a rigid body motion, the following has to hold.
  - When driving **straight**: all velocity vectors must be equal (i.e. direction and magnitude).



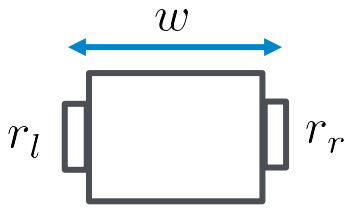
  - When **turning**: all wheel axes must intersect the vehicle Instant Centre of Rotation (ICR) axis perpendicularly and the speeds must be $v_i / R_i = \Omega$ (R_i: distance wheel-ICR, Ω vehicle body rotation rate)

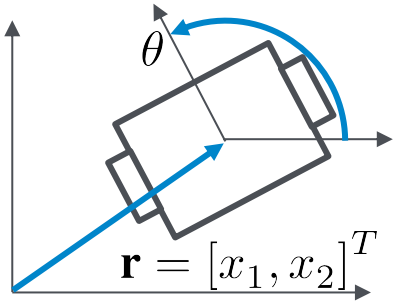


Stanley self-driving car, DARPA Grand Challenge 2005

Differential Drive Example Re-Visited

Consider the following diff-drive robot with state vector $\mathbf{x} = [x_1, x_2, \theta]^T$ and inputs $\mathbf{u} = [\omega_l, \omega_r]^T$.





General equation of motion:

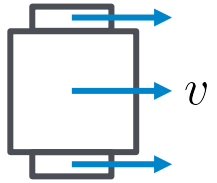
$$\dot{x}_1 = v\cos(\theta),$$
$$\dot{x}_2 = v\sin(\theta),$$
$$\dot{\theta} = \Omega.$$

With:

$$v = \frac{\omega_l r_l + \omega_r r_r}{2},$$
$$\Omega = \frac{\omega_r r_r - \omega_l r_l}{w}.$$

- When moving **straight forward**: $v = \omega_l r_l = \omega_r r_r, \quad \Omega = 0$

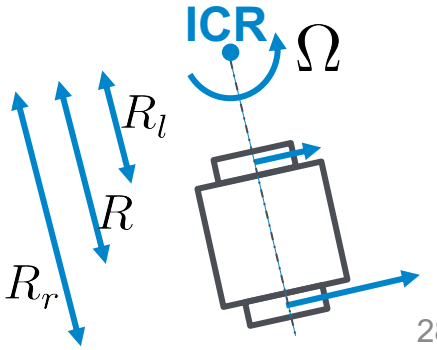
  Discretised*: $\mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} D\cos\theta \\ D\sin\theta \\ 0 \end{bmatrix}$ with $D = v\Delta t$.



- When **turning**: $\Omega = (\omega_l r_l)/R_l = (\omega_r r_r)/R_r$

  $R = v/\Omega, \Delta\theta = \Omega\Delta t$

  Discretised*: $\mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} R(\sin(\Delta\theta + \theta) - \sin\theta) \\ -R(\cos(\Delta\theta + \theta) - \cos\theta) \\ \Delta\theta \end{bmatrix}$



*) These are analytic integrals (we were lucky) over a time step Δt (with wheel velocities held constant).

IMU Kinematics (Naively)

Remember what a MEMS IMU will output:

So we can formulate the kinematics with these inputs (**indirect** formulation):

$$
\begin{aligned}
{}_W\dot{\mathbf{r}}_S &= {}_W\mathbf{v}\,, \\
\dot{\mathbf{q}}_{WS} &= \frac{1}{2}\mathbf{q}_{WS} \otimes \begin{bmatrix} {}_S\tilde{\boldsymbol{\omega}} \\ 0 \end{bmatrix}, \\
{}_W\dot{\mathbf{v}} &= \mathbf{C}_{WS}\,{}_S\tilde{\mathbf{a}} + {}_W\mathbf{g}.
\end{aligned}
$$

IMU Kinematics with Sensor Error Models

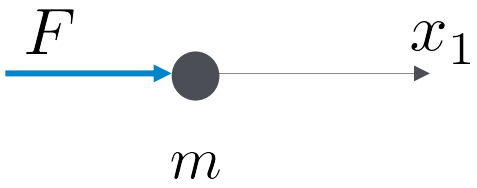
In reality, the measurements will be far from perfect. We can include the sensor error model as e.g.:

$$
\begin{aligned}
{}_W\dot{\mathbf{r}}_S &= {}_W\mathbf{v}\,, \\
\dot{\mathbf{q}}_{WS} &= \frac{1}{2}\mathbf{q}_{WS} \otimes \begin{bmatrix} {}_S\tilde{\boldsymbol{\omega}} + \mathbf{w}_{\mathrm{g}} - \mathbf{b}_{\mathrm{g}} \\ 0 \end{bmatrix}, \\
{}_W\dot{\mathbf{v}} &= \mathbf{C}_{WS}\left({}_S\tilde{\mathbf{a}} + \mathbf{w}_{\mathrm{a}} - \mathbf{b}_{\mathrm{a}}\right) + {}_W\mathbf{g}, \\
\dot{\mathbf{b}}_{\mathrm{g}} &= \mathbf{w}_{b_{\mathrm{g}}}, \\
\dot{\mathbf{b}}_{\mathrm{a}} &= \mathbf{w}_{b_{\mathrm{a}}}.
\end{aligned}
$$

IMU biases: Random Walk

Newton's Second Law: 1D Example

Assume a point mass m is pushed by a force F:



$$F = ma.$$

So, we can write the equations of motion as:

$$\dot{x}_1 = x_2,$$

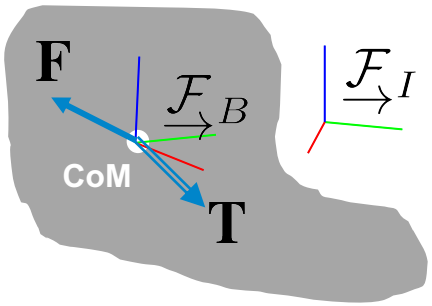
$$\dot{x}_2 = u_1,$$

where we were using an input $u_1 = \dfrac{F}{m}$.

Newton's 2nd Law (Rigid Bodies): Euler's Laws of Motion

Now we consider a body of finite extent, mass m and inertia matrix $\mathbf{I}$ affected by a (total) force $\mathbf{F}$ and a (total) torque $\mathbf{T}$ acting on its **Centre of Mass** (CoM).

The law of motion expressed **in body frame** $\underrightarrow{\mathcal{F}}_B$ is stated as:



$$
\begin{aligned}
{}_B\mathbf{F} &= \sum {}_B\mathbf{F}_i = m({}_B\dot{\mathbf{v}}_{\mathrm{CoM}}) + m\,{}_B\boldsymbol{\omega} \times {}_B\mathbf{v}_{\mathrm{CoM}}, \\
{}_B\mathbf{T} &= \sum {}_B\mathbf{T}_i = \mathbf{I}({}_B\dot{\boldsymbol{\omega}}) + {}_B\boldsymbol{\omega} \times \mathbf{I}\,{}_B\boldsymbol{\omega}.
\end{aligned}
$$

With:

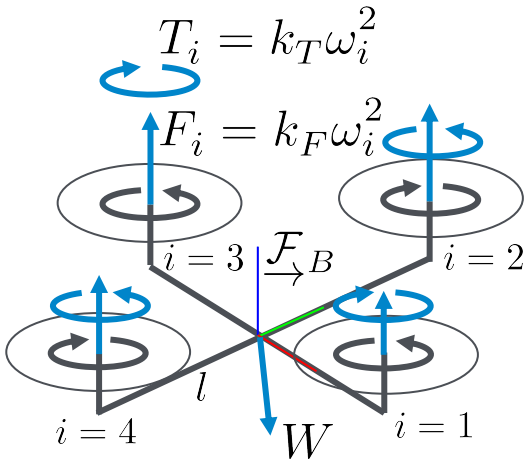
${}_B\mathbf{v}_{\mathrm{CoM}}$: velocity of CoM w.r.t. inertial frame $\underrightarrow{\mathcal{F}}_I$ expressed in body frame.

${}_B\boldsymbol{\omega}$: rotation speed of body w.r.t. inertial frame $\underrightarrow{\mathcal{F}}_I$ expressed in body frame.

Note: for the complete set of equations of motion that contain position and orientation, you will want to add the rigid body kinematics from before…

Dynamics Example: Simplified Quadrotor Model

www.fsd.mw.tum.de, AscTech Hummingbird



Inertial frame: $\mathcal{F}_W$

All we have to do:

1. **Define control inputs.**
   Rotor speeds $\mathbf{u} = [\omega_1, \omega_2, \omega_3, \omega_4]^T$.
2. **Make assumptions and simplifications about the states and inputs.**
   Near hover.
3. **List all the forces and torques we think is sensible to consider.**
   Thrust forces F_i, friction moments T_i and weight W.
   (Disregarding aerodynamic drag and rotor gyroscopic effects since near hover assumed)
4. **Express them as function of states / inputs.**
   $F_i = k_F \omega_i^2, \quad T_i = k_T \omega_i^2, \quad W = mg.$
5. **Reduce them to the CoM.**
   $${}_B\mathbf{F} = [0, 0, F_1 + F_2 + F_3 + F_4]^T - \mathbf{C}_{BW}\,[0, 0, mg]^T,$$
   $${}_B\mathbf{M} = [F_2 l - F_4 l, -F_1 l + F_3 l, -T_1 + T_2 - T_3 + T_4]^T.$$
6. **Plug them into the Newton-Euler equations combined with rigid body kinematics.**

A Note on Parameter Identification

Options to identify parameters:

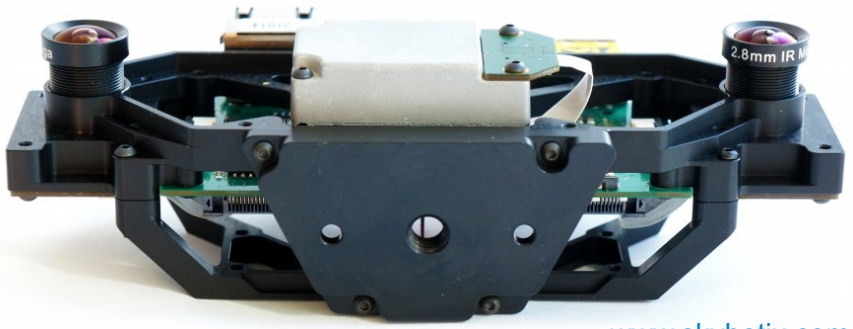
- Measure directly (e.g. mass)
- Obtain through computation/simulation (e.g. inertia matrix)
- Calibrate / identify, i.e. estimate
- Noise parameters: from datasheets and/or statistics

Calibration

- Use as much ground truth information available as possible!
- Try and separate the problem into independent estimation parts (i.e. estimate one set of parameters that is not influenced by an unknown different set)

Example: stereo camera and IMU rig

1. Get noise parameters from datasheets
2. Calibrate intrinsics of all cameras individually
3. Calibrate extrinsics of all cameras w.r.t. the IMU jointly
4. Optional: allow for online re-calibration…



www.skybotix.com

Any Questions?

See you on Thursday at 1pm for the practical.