Imperial College London

SMART ROBOTICS LAB

08/02/2021

Lecture 4:
The Extended Kalman Filter

70003 Advanced Robotics

Dr Stefan Leutenegger

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

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

IMU Kinematics with Sensor Error Models

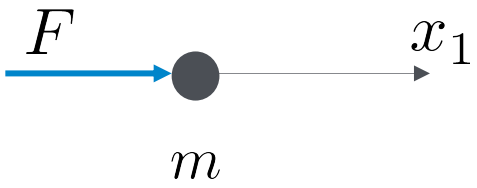
In reality, the measurements will be far from perfect. We can include the sensor error model as e.g.:

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

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:

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

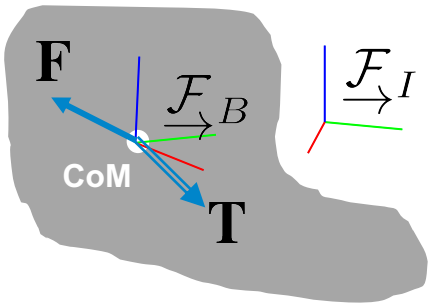
$$\dot{x}_2 = u_1,$$

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

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:

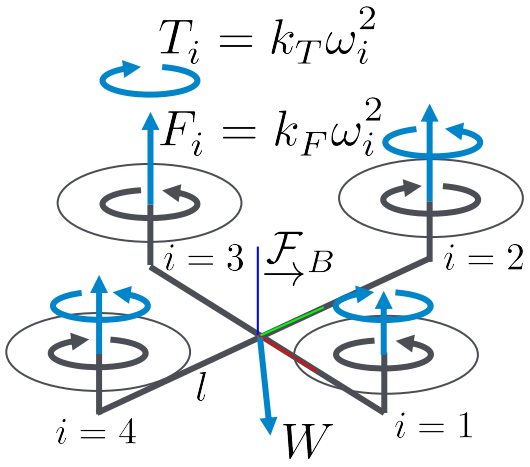
${}_B\mathbf{v}_{\mathrm{CoM}}$: velocity of CoM w.r.t. inertial frame $\underrightarrow{\mathcal{F}}_I$ expressed in body frame.

${}_B\boldsymbol{\omega}$: 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:

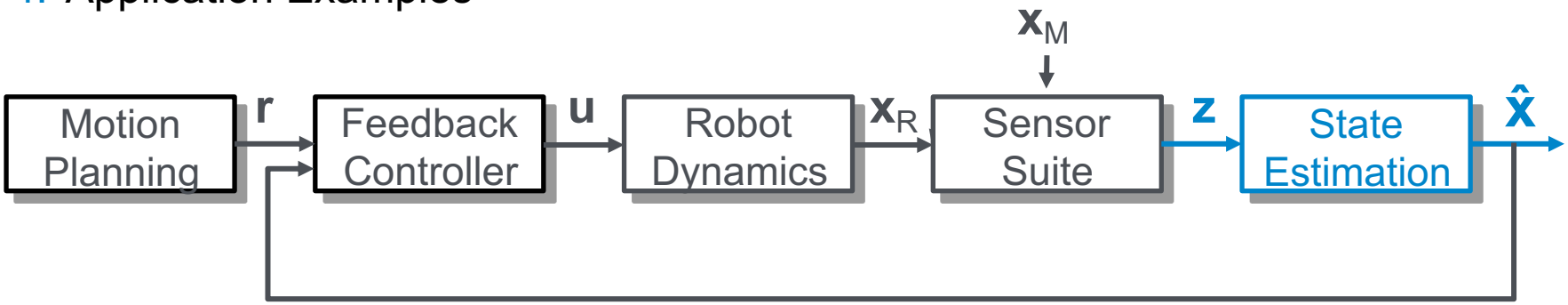
1. **Define control inputs.**
   Rotor speeds $\mathbf{u} = [\omega_1, \omega_2, \omega_3, \omega_4]^T$.
2. **Make assumptions and simplifications about the states and inputs.**
   Near hover.
3. **List all the forces and torques we think is sensible to consider.**
   Thrust forces F_i, friction moments 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.**

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. The Concept of Recursive Estimation
2. The Kalman Filter
3. The Extended Kalman Filter (EKF)
4. Application Examples



Probabilistic Estimation

We use Bayesian Statistics:

Measurements $\mathbf{z}$ are modeled as samples from a **distribution** $p(\mathbf{z}|\mathbf{x})$ given the variables $\mathbf{x}$ (here: robot states plus possibly the map).

- **Maximum Likelihood (ML) Estimation:**
  What values for variables best explain all the measurements?

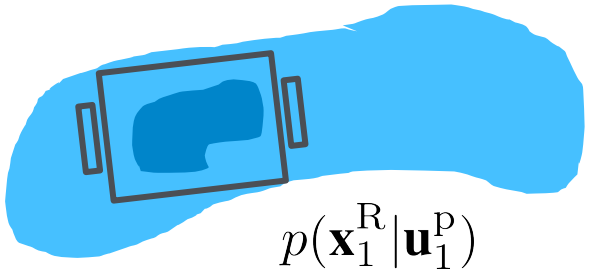
- **Maximum a Posteriori (MAP) Estimation:**
  What values for variables best explain all the measurements and a prior belief $p(\mathbf{x})$ about those variables?

Now we will talk about how to compute these values.

2D Robot Localisation Example

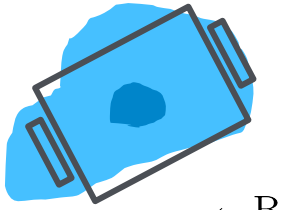
Has wheel odometry

State: $\mathbf{x}_k^{\mathrm{R}} = [x_1, x_2, \theta]^T$



$p(\mathbf{x}_1^{\mathrm{R}} | \mathbf{u}_1^{\mathrm{p}})$

wheel odometry measurements/inputs $\mathbf{u}_1^{\mathrm{p}}$

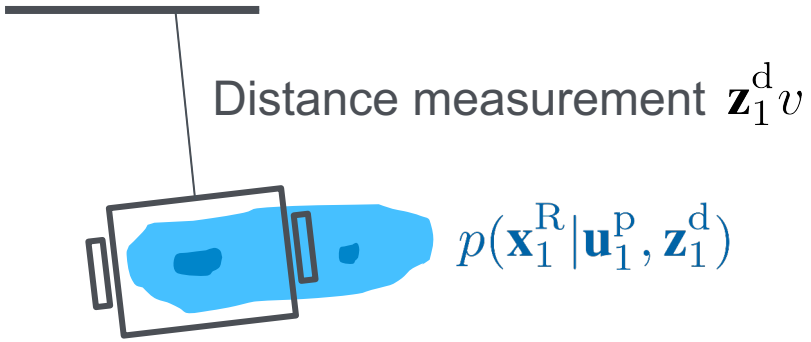


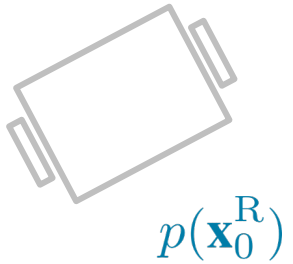
$p(\mathbf{x}_0^{\mathrm{R}})$

2D Robot Localisation Example

Has wheel odometry and makes (forward) distance measurements

State: $\mathbf{x}_k^{\mathrm{R}} = [x_1, x_2, \theta]^T$





2D Robot Localisation: Uncertainty Representation

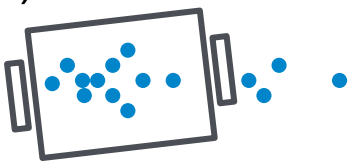
Has wheel odometry and makes (forward) distance measurements

State: $\mathbf{x}_k = [x_1, x_2, \theta]^T$

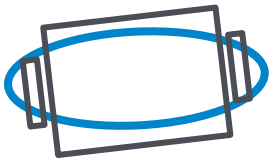
The "true" distribution
(we'll never know it)

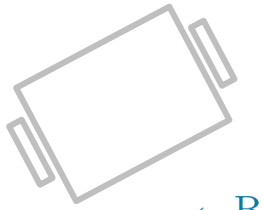
$p(\mathbf{x}_1^{\mathrm{R}} | \mathbf{u}_1^{\mathrm{p}}, \mathbf{z}_1^{\mathrm{d}})$

Particle distribution
(used in a particle filter)

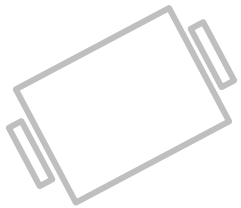


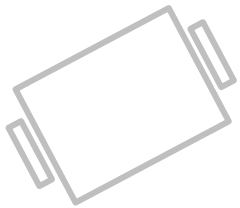
Normal Distribution
(used in an EKF)





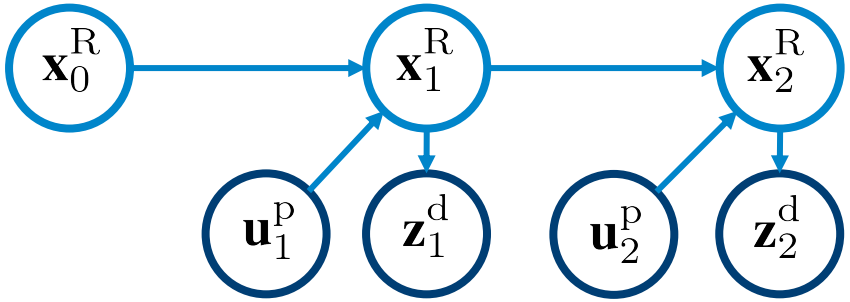
$p(\mathbf{x}_0^{\mathrm{R}})$





Differential Drive Robot Example

Bayesian Network (a Directed Acyclic Graph) of the diff drive robot: the generative model



$\mathbf{x}_k^{\mathrm{R}}$: the robot state at time step k.
$\mathbf{z}_k^{\mathrm{d}}$: a distance measurement.
$\mathbf{u}_k^{\mathrm{p}}$: wheel odometry (measurement).

$$p(\mathbf{x}_0^{\mathrm{R}}, \ldots, \mathbf{x}_N^{\mathrm{R}}, \mathbf{z}_1^{\mathrm{d}}, \ldots, \mathbf{z}_N^{\mathrm{d}}, \mathbf{u}_1^{\mathrm{p}}, \ldots, \mathbf{u}_N^{\mathrm{p}}) =$$

$$\underbrace{p(\mathbf{x}_0^{\mathrm{R}})}_{\text{Prior}} \underbrace{p(\mathbf{x}_1^{\mathrm{R}}|\mathbf{u}_1^{\mathrm{p}}, \mathbf{x}_0^{\mathrm{R}})}_{\text{State transition model}} p(\mathbf{u}_1^{\mathrm{p}}) \underbrace{p(\mathbf{z}_1^{\mathrm{d}}|\mathbf{x}_1^{\mathrm{R}})}_{\text{Dist. meas. likelihood}} \ldots p(\mathbf{x}_N^{\mathrm{R}}|\mathbf{u}_N^{\mathrm{p}}, \mathbf{x}_{N-1}^{\mathrm{R}}) p(\mathbf{u}_N^{\mathrm{p}}) p(\mathbf{z}_N^{\mathrm{d}}|\mathbf{x}_N^{\mathrm{R}}).$$

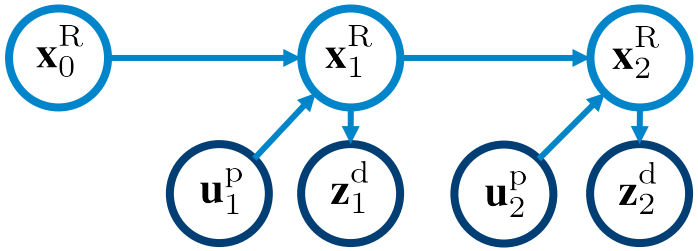
From here, we could try MAP estimation of the entire trajectory $\mathbf{x}$ with given $\mathbf{z}$ and $\mathbf{u}$

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{z}, \mathbf{u}).$$

This would lead us into nonlinear optimisation (see Lecture 6)…

Recursive Estimation and the Bayes Filter

- So far we have looked at batch estimation, i.e. estimating every variable at once.
- Recall, however, the typical structure of a robot state estimation problem:



$\mathbf{x}_k^{\mathrm{R}}$: the robot state at time step k.
$\mathbf{z}_k^{\mathrm{d}}$: a distance measurement.
$\mathbf{u}_k^{\mathrm{p}}$: wheel odometry (measurement).

$$p(\mathbf{x}_0^{\mathrm{R}}, \ldots, \mathbf{x}_N^{\mathrm{R}}, \mathbf{z}_1^{\mathrm{d}}, \ldots, \mathbf{z}_N^{\mathrm{d}}, \mathbf{z}_1^{\mathrm{p}}, \ldots, \mathbf{z}_N^{\mathrm{p}}) = p(\mathbf{x}_0^{\mathrm{R}}) p(\mathbf{x}_1^{\mathrm{R}}|\mathbf{u}_1^{\mathrm{p}}, \mathbf{x}_0^{\mathrm{R}}) p(\mathbf{u}_1^{\mathrm{p}}) p(\mathbf{z}_1^{\mathrm{d}}|\mathbf{x}_1^{\mathrm{R}}) \ldots p(\mathbf{x}_N^{\mathrm{R}}|\mathbf{u}_N^{\mathrm{p}}, \mathbf{x}_{N-1}^{\mathrm{R}}) p(\mathbf{u}_N^{\mathrm{p}}) p(\mathbf{z}_N^{\mathrm{d}}|\mathbf{x}_N^{\mathrm{R}}).$$

Often, we are only interested in the **current state** (e.g. to control the robot) and not the whole (growing!) history of robot states (trajectory).

This can be done by alternating **prediction** (using a temporal model) and **update**:

Bayes Filter – recursive MAP estimation:

- Initialise the prior distribution $p(\mathbf{x}_0^{\mathrm{R}})$.
- At every k[th] iteration do the following two steps:

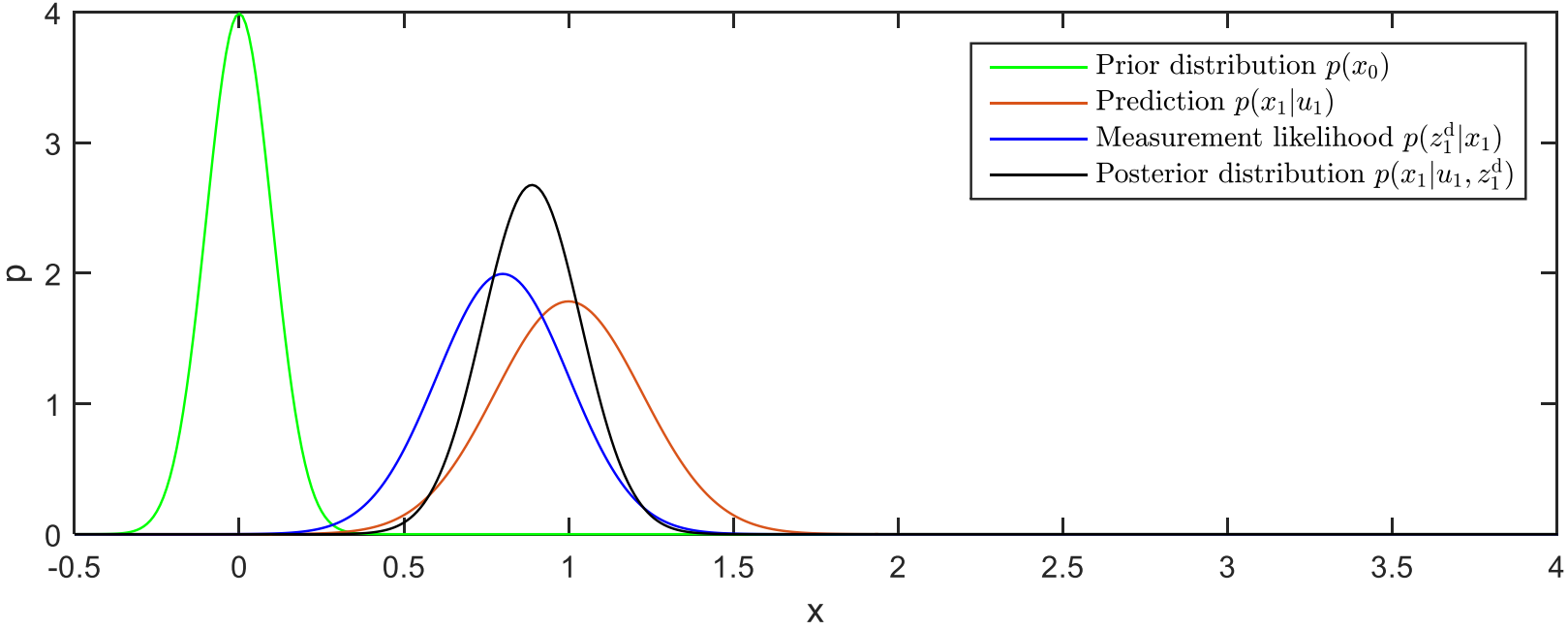
1. **Predict**: $p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_{1:k}^{\mathrm{p}}, \mathbf{z}_{1:k-1}^{\mathrm{d}}) = \int p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_k^{\mathrm{p}}, \mathbf{x}_{k-1}^{\mathrm{R}}) p(\mathbf{x}_{k-1}^{\mathrm{R}}|\mathbf{u}_{1:k-1}^{\mathrm{p}}, \mathbf{z}_{1:k-1}^{\mathrm{d}}) d\mathbf{x}_{k-1}^{\mathrm{R}}$ **Prod./sum rule**
2. **Update**: $p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_{1:k}^{\mathrm{p}}, \mathbf{z}_{1:k}^{\mathrm{d}}) = \eta p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_{1:k}^{\mathrm{p}}, \mathbf{z}_{1:k-1}^{\mathrm{d}}) p(\mathbf{z}_k^{\mathrm{d}}|\mathbf{x}_k^{\mathrm{R}})$ **Bayes' theorem**

1D Kalman Filter (KF) Example

KF: Bayes Filter assuming Gaussian distributions for all random variables as well as linear measurement and state transition models!

Example: robot moving along the x-axis…

- State transition model: $x_k = x_{k-1} + u + w, \quad w \sim \mathcal{N}(0, q)$, where u^{p} is an odometry reading.
- Measurement updates (absolute position): $z^{\mathrm{d}} = x, \quad z^{\mathrm{d}} \sim \mathcal{N}(0, r).$



Kalman Filter (KF) Problem Statement

Given:

- Gaussian distributed initial state $\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_0)$.
- Linear state transition model $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{G}\mathbf{u}_k + \mathbf{L}\mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$
- Linear measurements $\tilde{\mathbf{z}}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$

Compute:

1. The **predicted** distribution after state transition
   $\mathbf{x}_{k|k-1} \sim p(\mathbf{x}_k) = \mathcal{N}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$.
2. The posterior distribution after measurement **update**
   $\mathbf{x}_{k|k} \sim p(\mathbf{x}_k|\mathbf{z}_k) = \mathcal{N}(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$.

Kalman Filter (KF) Derivation

1. **Prediction:** $p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_{1:k}^{\mathrm{p}}, \mathbf{z}_{1:k-1}^{\mathrm{d}}) = \int \underbrace{p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_k^{\mathrm{p}}, \mathbf{x}_{k-1}^{\mathrm{R}})}_{\mathcal{N}(\mathbf{F}\mathbf{x}_{k-1}+\mathbf{G}\mathbf{u}_k, \mathbf{Q}_k)} \underbrace{p(\mathbf{x}_{k-1}^{\mathrm{R}}|\mathbf{u}_{1:k-1}^{\mathrm{p}}, \mathbf{z}_{1:k-1}^{\mathrm{d}})}_{\mathcal{N}(\hat{\mathbf{x}}_{k-1|k}, \mathbf{P}_{k-1|k-1})} d\mathbf{x}_{k-1}^{\mathrm{R}}$

   The evaluation of this product / integral is very technical. Instead, we derive the result here using the linear error propagation:

Kalman Filter (KF) Derivation

1. **Update:** $p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_{1:k}^{\mathrm{p}}, \mathbf{z}_{1:k}^{\mathrm{d}}) = \eta \underbrace{p(\mathbf{x}_k^{\mathrm{R}}|\mathbf{u}_{1:k}^{\mathrm{p}}, \mathbf{z}_{1:k-1}^{\mathrm{d}})}_{\mathcal{N}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})} \underbrace{p(\mathbf{z}_k^{\mathrm{d}}|\mathbf{x}_k^{\mathrm{R}})}_{\mathcal{N}(\mathbf{H}\mathbf{x}_k, \mathbf{R}_k)}$

Kalman Filter (KF) Equations

KF: Bayes Filter assuming Gaussian distributions for all random variables!

Initialisation:

Decide on starting point for states $\mathbf{x}_0$ and their prior covariance $\mathbf{P}_0$.

Prediction:

Odometry/control — Noise, with covariance $\mathbf{Q}_k$

With linear state transition model $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{G}\mathbf{u}_k + \mathbf{L}\mathbf{w}_k$

Mean: $\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{G}\mathbf{u}_k$

Covariance: $\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\top + \mathbf{L}\mathbf{Q}_k\mathbf{L}^\top$

Measurement update:

Noise, with covariance $\mathbf{R}_k$

With measurement $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$

Measurement residual: $\mathbf{y}_k = \tilde{\mathbf{z}}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$

Residual covariance: $\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R}_k$

Kalman gain: $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\top\mathbf{S}_k^{-1}$

Updated state (mean): $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k$

Updated covariance: $\mathbf{P}_{k|k} = (\mathbf{1} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}$

Extended Kalman Filter (EKF): Nonlinear Version

EKF: Bayes Filter (up to linearisation) assuming Gaussian distributions for all random variables!

Initialisation:

Decide on starting point for states $\mathbf{x}_0$ and their prior covariance $\mathbf{P}_0$.

Prediction:

Odometry/control $\mathbf{u}_k$, Noise, with covariance $\mathbf{Q}_k$

With nonlinear state transition model $\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k)$.

Mean: $\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$

Covariance: $\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{L}_k \mathbf{Q}_k \mathbf{L}_k^\top$

$\mathbf{F}_k$: Linearisation $\partial \mathbf{f}/\partial \mathbf{x}$; $\mathbf{L}_k$: Linearisation $\partial \mathbf{f}/\partial \mathbf{w}$

Measurement update:

With measurement $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k$ — Noise, with covariance $\mathbf{R}_k$

Measurement residual: $\mathbf{y}_k = \tilde{\mathbf{z}}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$

Residual covariance: $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k$

$\mathbf{H}$: Linearisation $\delta \mathbf{h}/\partial \mathbf{x}$

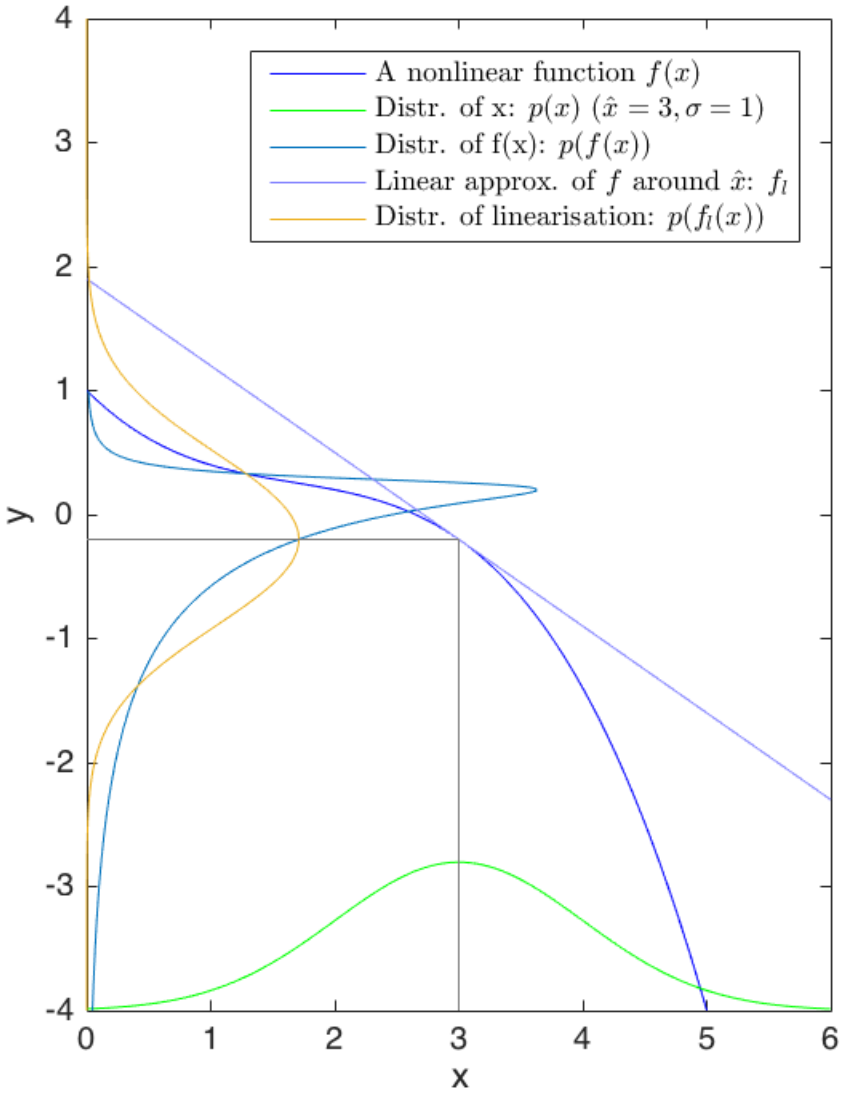
Kalman gain: $\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$

Updated state (mean): $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k$

Updated covariance: $\mathbf{P}_{k|k} = (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$

Problem with Linearisation in EKF – 1D Example

- The linear approximation might not be a good one for the whole distribution.
- Error propagation through the linearised approximation may e.g.
  - shift peak (**biased solution**) or
  - Lead to higher peak (**overconfidence**)
- Whether or not this causes trouble depends on the problem we are trying to solve:
  - intrinsic nonlinearities in propagation and measurement
  - Noise levels (process noise and measurement noise)

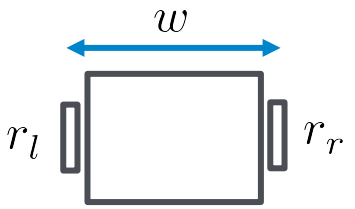


Potential EKF Problems and Mitigation

Potential problems with the EKF:

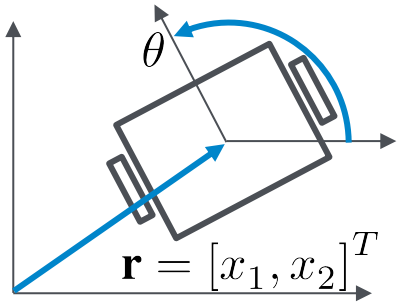
1. The **nonlinearities** may be significant w.r.t. uncertainty of involved variables. To address this, a number of alternatives were proposed:
   - Sigma-point filters (e.g. UKF) pass characteristic points around the mean through the nonlinear function rather than using the Jacobian.
   - Iterated EKF (IEKF).
2. Measurements **may not** be approximated well with **Gaussian** likelihood and come with outliers.
   - Check the consistency of the residual (e.g. $\mathbf{y}^T\mathbf{S}^{-1}\mathbf{y}$) to reject outliers.

Example: EKF Localisation for Diff-Drive Lawn-Mover



Given:

- Differential-drive robot measuring left and right wheel angles $\Delta\varphi_l, \Delta\varphi_r$.
- GPS position measurements.
- Yaw angle measurements by compass.



Find:

- Suitable definition of the robot state.
- (Nonlinear) state transition and measurement equations (plus linearisation).

Solution:

State: $\mathbf{x}_k = [x_1, x_2, \theta]^T$.

State transition (constant wheel velocities during step – see Robotics 333):

$r_r\Delta\varphi_r \neq r_l\Delta\varphi_l : \mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} R(\sin(\Delta\theta+\theta) - \sin\theta) \\ -R(\cos(\Delta\theta+\theta) - \cos\theta) \\ \Delta\theta \end{bmatrix}, \mathbf{F}_k =$
(on an arc)

$r_r\Delta\varphi_r = r_l\Delta\varphi_l : \mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} D\cos\theta \\ D\sin\theta \\ \Delta\theta \end{bmatrix}, \mathbf{F}_k =$
(straight forward)

$$\Delta\theta = \frac{r_r\Delta\varphi_r - r_l\Delta\varphi_l}{w},$$
$$R = w\frac{r_l\Delta\varphi_l + r_r\Delta\varphi_r}{r_r\Delta\varphi_r - r_l\Delta\varphi_l},$$
$$D = \frac{r_l\Delta\varphi_l + r_r\Delta\varphi_r}{2}.$$

Position measurement: $\mathbf{z}^{\mathrm{p}} =$, $\mathbf{H}^{\mathrm{p}} =$, (coincidently linear).

Compass measurement: $z^{\mathrm{c}} =$, $\mathbf{H}^{\mathrm{c}} =$, (coincidently linear).

Local Parameterisation of Orientation Again…

A quaternion can be interpreted as defined in a 3-dimensional manifold.
A manifold is a topological space that locally resembles Euclidean space near each point [12]. We can express e.g. uncertainty in this Euclidean space (tangent space).

Minimal local parameterisation:

$\delta\boldsymbol{\alpha} \in \mathbb{R}^3$: (small) rotation vector

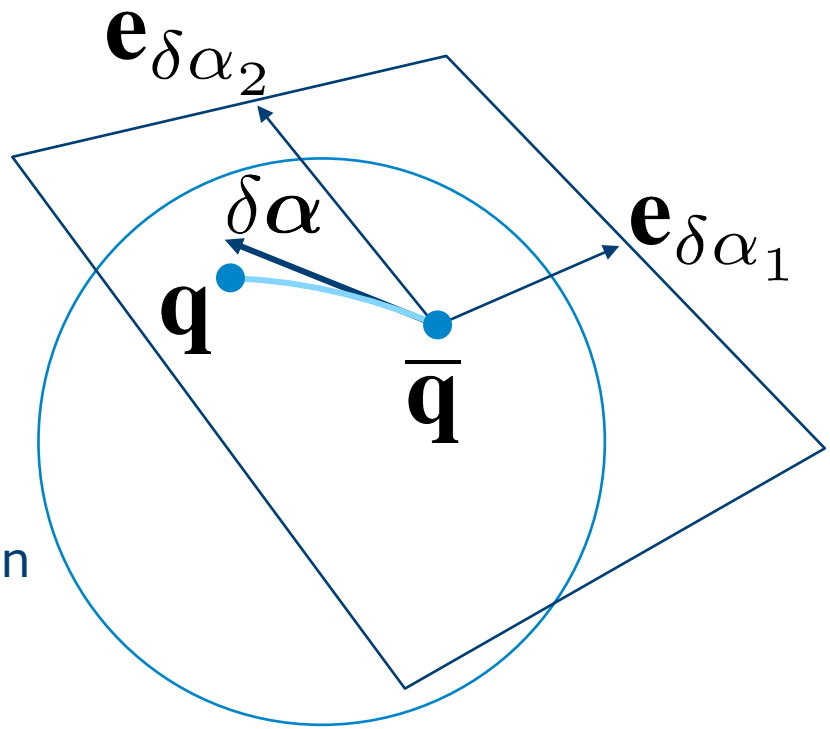
Can be converted into quaternion:

$$\delta\mathbf{q} := \exp(\delta\boldsymbol{\alpha}) = \begin{bmatrix} \operatorname{sinc}\left\|\frac{\delta\boldsymbol{\alpha}}{2}\right\| \frac{\delta\boldsymbol{\alpha}}{2} \\ \cos\left\|\frac{\delta\boldsymbol{\alpha}}{2}\right\| \end{bmatrix}.$$

Now we write:

$$\mathbf{q}_{WS} = \delta\mathbf{q}(\delta\boldsymbol{\alpha}) \otimes \overline{\mathbf{q}}_{WS}.$$

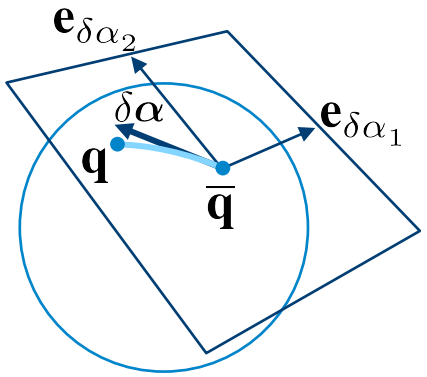
Linearisation point



Now, we **compute Jacobians** w.r.t. $\delta\boldsymbol{\alpha} \in \mathbb{R}^3$ instead of the quaternion $\mathbf{q} \in S^4$.

[12] https://en.wikipedia.org/wiki/Manifold

Local Parameterisation in Practice



We **compute Jacobians** w.r.t. $\delta\boldsymbol{\alpha} \in \mathbb{R}^3$ instead of $\mathbf{q} \in S^4$.

In practice, we will need only the following Jacobians, assuming we use $\mathbf{q}_{AB} = \delta\mathbf{q} \otimes \overline{\mathbf{q}}_{AB}$:

$$\frac{\partial}{\partial \delta\boldsymbol{\alpha}} \mathbf{C}_{AB}\,{}_{B}\mathbf{a} = -[\overline{\mathbf{C}}_{AB}\,{}_{B}\mathbf{a}]^{\times}. \text{ and } \frac{\partial}{\partial \delta\boldsymbol{\alpha}} \mathbf{C}_{BA}\,{}_{A}\mathbf{a} = \overline{\mathbf{C}}_{BA}\,[{}_{A}\mathbf{a}]^{\times}.$$

Proof (writing out with Lecture 2, Slide 21):

$$\mathbf{C}_{AB}\,{}_{B}\mathbf{a} = \mathbf{C}(\delta\mathbf{q})\overline{\mathbf{C}}_{AB}\,{}_{B}\mathbf{a} = \left(\mathbf{1}_3 + 2a(\delta\mathbf{q})\,[\mathbf{v}(\delta\mathbf{q})]^{\times} + 2\left([\mathbf{v}(\delta\mathbf{q})]^{\times}\right)^2\right)\overline{\mathbf{C}}_{AB}\,{}_{B}\mathbf{a}$$

$$\approx \left(\mathbf{1} + 2a(\delta\mathbf{q})\,[\mathbf{v}(\delta\mathbf{q})]^{\times}\right)\overline{\mathbf{C}}_{AB}\,{}_{B}\mathbf{a} \approx \left(\mathbf{1} + [\delta\boldsymbol{\alpha}]^{\times}\right)\overline{\mathbf{C}}_{AB}\,{}_{B}\mathbf{a}$$

Neglecting square of small imaginary part

$$\delta\mathbf{q} \approx \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\alpha} \\ 1 \end{bmatrix}$$

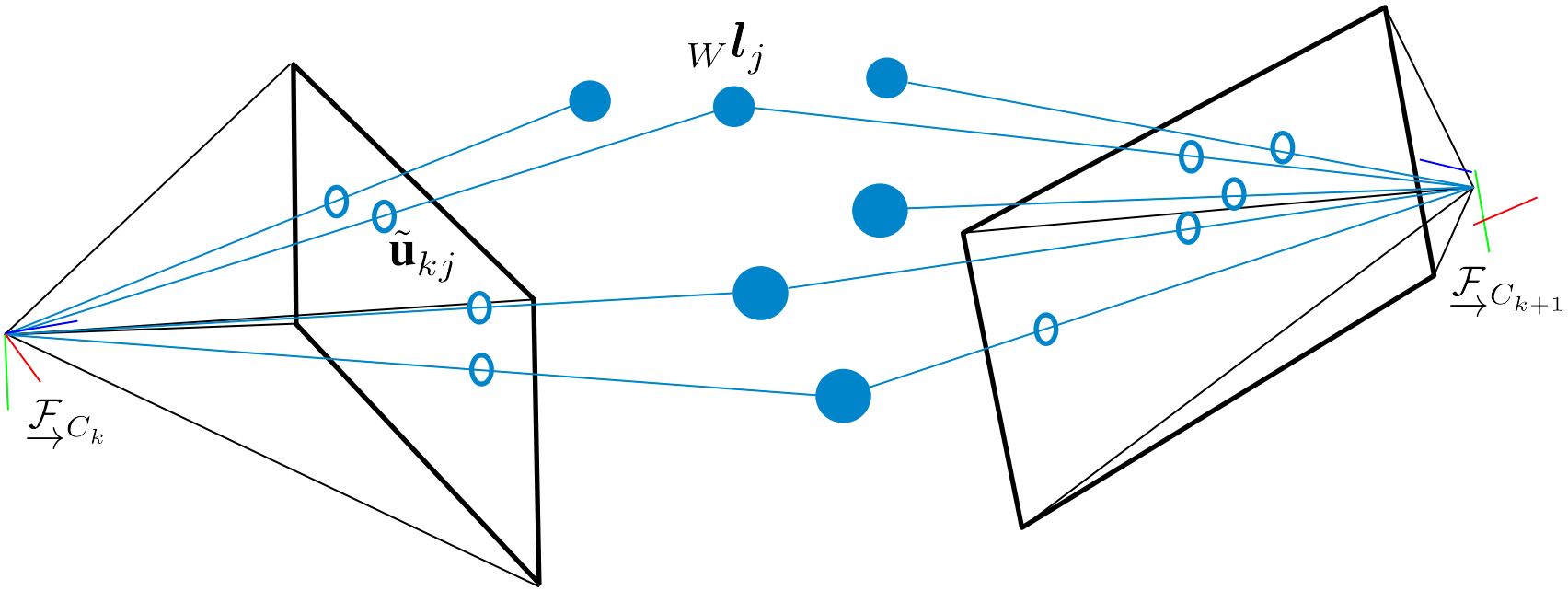
EKF Update

In the EKF, $\Delta\boldsymbol{\chi} = \mathbf{K}_k\mathbf{y}_k$ we will now contain a component $\Delta\boldsymbol{\alpha}$ to be applied as $\mathbf{q}_{k|k} = \delta\mathbf{q}(\Delta\boldsymbol{\alpha}) \otimes \mathbf{q}_{k|k-1}$.

Example: Visual-Inertial Localisation

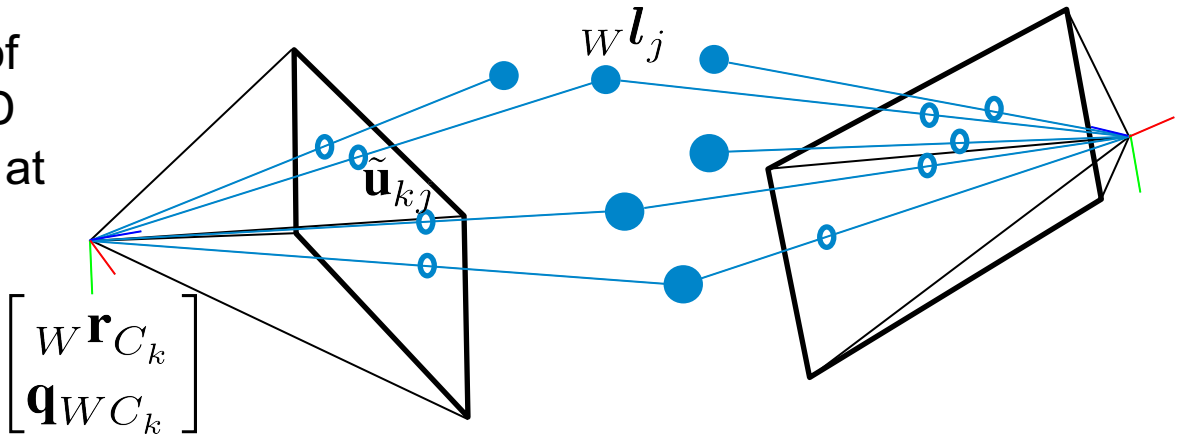
Now we will work towards the case of visual-inertial localisation/SLAM with a **general (6D) motion**.

First, we will consider **visual measurements:**



Example: Keypoint Measurements

Given: 2D detections $\tilde{\mathbf{u}}_{kj}$ of corresponding unknown 3D landmarks ${}_W\boldsymbol{l}_j$ in images at time step k .



Find: Measurement model $\mathbf{h}(\mathbf{x}) = \mathbf{h}({}_W\mathbf{r}_{C_k}, \mathbf{q}_{WC_k}, {}_W\boldsymbol{l}_j)$ and linearisation $\mathbf{H}$.

Solution:

$$\mathbf{h}({}_W\mathbf{r}_{C_k}, \mathbf{q}_{WC_k}, {}_W\boldsymbol{l}_j) = \mathbf{u}(\boldsymbol{T}_{C_kW}\ {}_W\boldsymbol{l}_j),$$

with $\mathbf{u} = \mathbf{k}(\mathbf{d}(\mathbf{p}({}_C\mathbf{l}_j)))$ (see Lecture 2),

1. project
2. distort
3. scale and centre

Example: Keypoint Measurement Linearisation

We use local perturbations $\delta\boldsymbol{\chi} = [\delta\mathbf{r}, \delta\boldsymbol{\alpha}, \delta\mathbf{l}]^T$ around $\bar{\mathbf{x}} = [{}_W\bar{\mathbf{r}}_C, \bar{\mathbf{q}}_{WC}, {}_W\bar{\mathbf{l}}_j]^T$.

We would like to compute $\frac{\partial \mathbf{h}}{\partial {}_W\mathbf{r}_C}$, $\frac{\partial \mathbf{h}}{\partial \delta\boldsymbol{\alpha}_k}$ and $\frac{\partial \mathbf{h}}{\partial {}_W\mathbf{l}_j}$,

of $\mathbf{h}({}_W\mathbf{r}_C, \mathbf{q}_{WC}, {}_W\boldsymbol{l}_j) = \mathbf{u}({}_C\mathbf{l}_j) = \mathbf{k}(\mathbf{d}(\mathbf{p}({}_C\mathbf{l}_j)))$.

With chain rule $\frac{\partial \mathbf{h}}{\partial {}_W\mathbf{r}_C} =$, $\frac{\partial \mathbf{h}}{\partial \delta\boldsymbol{\alpha}} =$, $\frac{\partial \mathbf{h}}{\partial {}_W\mathbf{l}_j} =$.

Jacobian of world2cam: $\mathbf{U}({}_C\bar{\mathbf{l}}_j) := \frac{\partial}{\partial {}_C\mathbf{l}_j}\mathbf{u}({}_C\mathbf{l}_j) =$.

Now, we know (Lecture 2): ${}_C\mathbf{l}_j = \mathbf{C}_{WC}^T\, {}_W\mathbf{l}_j - \mathbf{C}_{WC}^T\, {}_W\mathbf{r}_C$, thus

$\frac{\partial {}_C\mathbf{l}_j}{\partial {}_W\mathbf{r}_C} =$

$\frac{\partial {}_C\mathbf{l}_j}{\partial \delta\boldsymbol{\alpha}} =$

$\frac{\partial {}_C\mathbf{l}_j}{\partial {}_W\mathbf{l}_j} =$

Example: IMU Kinematics With Sensor Intrinsics

In Lecture 3, we derived the **continuous-time nonlinear** model

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

IMU biases

We want to find the dynamics of infinitesimal perturbations $\delta\boldsymbol{\chi}$ around the linearisation point $\bar{\mathbf{x}}$ as

$$\delta\dot{\boldsymbol{\chi}}(t) = \mathbf{F}_c(\bar{\mathbf{x}}, t)\delta\boldsymbol{\chi}(t) + \mathbf{L}_c(\bar{\mathbf{x}}, t)\mathbf{w}(t),$$

with the noise terms $\delta\mathbf{w} = [\mathbf{w}_g, \mathbf{w}_a, \mathbf{w}_{b_g}, \mathbf{w}_{b_a}]^T$,

$\delta\boldsymbol{\chi} = [\delta\mathbf{r}, \delta\boldsymbol{\alpha}, \delta\mathbf{v}, \delta\mathbf{b}_\mathrm{g}, \delta\mathbf{b}_\mathrm{a}]^T$, $\bar{\mathbf{x}} = [{}_W\bar{\mathbf{r}}_S, \bar{\mathbf{q}}_{WS}, {}_W\bar{\mathbf{v}}_{WS}, \bar{\mathbf{b}}_\mathrm{g}, \bar{\mathbf{b}}_\mathrm{a}]^T$, and

$\mathbf{x} = [{}_W\bar{\mathbf{r}}_S + \delta\mathbf{r}, \delta\mathbf{q} \otimes \bar{\mathbf{q}}_{WS}, {}_W\bar{\mathbf{v}}_{WS} + \delta\mathbf{v}, \bar{\mathbf{b}}_\mathrm{g} + \delta\mathbf{b}_\mathrm{g}, \bar{\mathbf{b}}_\mathrm{a} + \delta\mathbf{b}_\mathrm{a}]^T$.

Linearised IMU Kinematics and Sensor Intrinsics

Omitting the derivation, we obtain:

$$
\begin{aligned}
\delta\dot{\mathbf{r}} &= {}_W\mathbf{v}\,, \\
\delta\dot{\boldsymbol{\alpha}} &= (\mathbf{1} + \delta\boldsymbol{\alpha}^{\times})\overline{\mathbf{C}}_{WS}\,(\mathbf{w}_{\mathrm{g}} - \delta\mathbf{b}_{\mathrm{g}}), \\
\delta\dot{\mathbf{v}} &= \mathbf{C}_{WS}\,({}_S\tilde{\mathbf{a}} + \mathbf{w}_{\mathrm{a}} - \mathbf{b}_{\mathrm{a}}) + {}_W\mathbf{g}, \\
\delta\dot{\mathbf{b}}_{\mathrm{g}} &= \mathbf{w}_{b_{\mathrm{g}}}\,, \\
\delta\dot{\mathbf{b}}_{\mathrm{a}} &= \mathbf{w}_{b_{\mathrm{a}}}\,.
\end{aligned}
$$

Now, for the overall linearised dynamics $\delta\dot{\boldsymbol{\chi}}(t) = \mathbf{F}_c(\overline{\mathbf{x}}, t)\delta\boldsymbol{\chi}(t) + \mathbf{L}_c(\overline{\mathbf{x}}, t)\mathbf{w}(t)$, we finally obtain finally:

$$
\begin{bmatrix} \delta\dot{\mathbf{r}} \\ \delta\dot{\boldsymbol{\alpha}} \\ \delta\dot{\mathbf{v}} \\ \delta\dot{\mathbf{b}}_{\mathrm{g}} \\ \delta\dot{\mathbf{b}}_{\mathrm{a}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{1}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\overline{\mathbf{C}}_{WS} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -[\overline{\mathbf{C}}_{WS}\,({}_S\tilde{\mathbf{a}} - \overline{\mathbf{b}}_{\mathrm{a}})]^{\times} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\overline{\mathbf{C}}_{WS} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}}_{\mathbf{F}_c} \begin{bmatrix} \delta\mathbf{r} \\ \delta\boldsymbol{\alpha} \\ \delta\mathbf{v} \\ \delta\mathbf{b}_{\mathrm{g}} \\ \delta\mathbf{b}_{\mathrm{a}} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{0}_{3\times1} \\ \overline{\mathbf{C}}_{WS}\,\mathbf{w}_{\mathrm{g}} \\ \overline{\mathbf{C}}_{WS}\,\mathbf{w}_{\mathrm{a}} \\ \mathbf{w}_{b_{\mathrm{g}}} \\ \mathbf{w}_{b_{\mathrm{a}}} \end{bmatrix}}_{\mathbf{L}_c\mathbf{w}}\,.
$$

Discrete-Time IMU Kinematics / Sensor Intrinsics

Discretisation (using Trapezoidal Integration):

$$\Delta\mathbf{x}_1 = \Delta t \mathbf{f}_c(\mathbf{x}_{k-1}, t_{k-1}),$$
$$\Delta\mathbf{x}_2 = \Delta t \mathbf{f}_c(\mathbf{x}_{k-1} + \Delta\mathbf{x}_1, t_k)),$$
$$\mathbf{x}_k = \mathbf{x}_{k-1} + \frac{1}{2}(\Delta\mathbf{x}_1 + \Delta\mathbf{x}_2).$$

Make sure to **re-normalise the quaternions** whenever modifying!

Linearised Discrete Error Dynamics (using Trapezoidal Integration):

We apply the chain-rule to the discretisation scheme…

$$\mathbf{F}_k = \mathbf{1} + \frac{\Delta t}{2}\mathbf{F}_c(\mathbf{x}_{k-1}, t_{k-1}) + \frac{\Delta t}{2}(\mathbf{F}_c(\mathbf{x}_{k-1} + \Delta\mathbf{x}_1, t_k)(\mathbf{1} + \Delta t \mathbf{F}_c(\mathbf{x}_{k-1}, t_{k-1}))).$$
$$\mathbf{L}_k = \frac{1}{2}(\mathbf{L}_c(\mathbf{x}_{k-1}, t_{k-1}) + \mathbf{L}_c(\mathbf{x}_{k-1} + \Delta\mathbf{x}_1, t_k)).$$

Covariance Propagation: $\mathbf{P}_k = \mathbf{F}_k\mathbf{P}_{k-1}\mathbf{F}_k^\top + \mathbf{L}_k\mathbf{Q}_k\mathbf{L}_k^\top$.

With: $\mathbf{Q}_k = \text{diag}[\sigma_{c,g}\mathbf{1}_3, \sigma_{c,a}\mathbf{1}_3, \sigma_{c,w_g}\mathbf{1}_3, \sigma_{c,w_a}\mathbf{1}_3]\Delta t.$

The sigmas denote continuous-time noise densities (e.g. from IMU datasheets).

Any Questions?

See you on Thursday at 1pm for the practical!