Autonomous Mobile Robots

Cheat Sheet by Janis Hutz
ETHZ, FS2026

1 Introduction

1.1 Probability

Def *(Sum rule)* $P(X) = \sum P(X,Y) = \sum P(X \cap Y)$

Def *(Prod)* $P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X)$

T *(Bayes)* $P(Y_i|X) = \dfrac{P(X|Y_i)P(Y_i)}{\sum_{j=1}^{n} P(X|Y_j)P(Y_j)}$

Def *(Cont. Var)* Sums become integrals
e.g. $\sum_X P(X) = 1$ becomes $\int p(x)\ \mathrm{d} = 1$

Def *(Indep.)* x, y indep. iff $p(x,y) = p(x)p(y)$

Def *(Cond. Indep.)* iff $p(x,y|z) = p(x|z)p(y|z)$

Def $E[x] = \int_{-\infty}^{\infty} \boldsymbol{x}p(\boldsymbol{x})\ \mathrm{d}x$, also for $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})$

Def $\mathrm{Cov}[x] = E[\boldsymbol{x}\boldsymbol{x}^\top] - E[\boldsymbol{x}]E[\boldsymbol{x}]^\top = \boldsymbol{\Sigma}$

Def *(Gauss. Dist.)* $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ($\boldsymbol{\mu}$ mean, $\boldsymbol{\Sigma}$ cov.),
PDF: $p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$

1.2 Measurement models

$\boldsymbol{z} = \boldsymbol{b}_C + s\boldsymbol{M}_S\boldsymbol{\omega} + \boldsymbol{b} + \boldsymbol{n} + \boldsymbol{o}$: $\boldsymbol{b}_C$ const bias, $\boldsymbol{b}$ time bias, $\boldsymbol{M}$ missal., $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ noise, $_S\omega$ corr. meas., $\boldsymbol{o}$ other infl.

1.3 Trigonometry

2 Locomotion & Kinematics

2.1 Positioning

Def *(Position Vector)* $_{\boxed{W}}\boldsymbol{t}_{\boxed{B}} = {}_{\boxed{W}}\boldsymbol{t}_{\boxed{W}\boxed{B}}$, Original Frame, End point, Target Frame, $\sin = s$, $\cos = c$

Def *(State vector)* x_R: x, v of rob in W, pos of sensors

Def *(Rot. Mat.)* $\boldsymbol{R}_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\boldsymbol{R}_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix}$; $\boldsymbol{R}_x(\varphi) \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\varphi) & -s(\varphi) \\ 0 & s(\varphi) & c(\varphi) \end{bmatrix}$

R Application: $_W\boldsymbol{a} = \boldsymbol{R}_{WB\,B}\boldsymbol{a}$

L $\boldsymbol{R}_{BW} = \boldsymbol{R}_{WB}^{-1} = \boldsymbol{R}_{WB}^\top$, $\det(\boldsymbol{R}_{WB}) = 1$ (orth.)

R Cols of $\boldsymbol{R}_{WB}$ are basis vec. of Frame $\underset{\rightarrow}{\mathcal{F}}_B$ in $\underset{\rightarrow}{\mathcal{F}}_W$

Def *(Euler Angles)* Yaw (z), Pitch (y), Roll (x), mult. rotation matrices, e.g. $\boldsymbol{R}_{EB} = \boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_y(\theta) \cdot \boldsymbol{R}_x(\varphi)$, bound..
$[\boldsymbol{n}]^\times = \boldsymbol{n}\boldsymbol{x}^\top$ (matrix from vec + arg $\boldsymbol{x}$)

Def *(Rot. Vec)* $\boldsymbol{\alpha} = \alpha\boldsymbol{n}$ ($\boldsymbol{n}$ normal)
$\boldsymbol{R}(\alpha, \boldsymbol{n}) = \boldsymbol{I}_3 + \sin(\alpha)[\boldsymbol{n}]^\times + (1 - \cos(\alpha))([\boldsymbol{n}]^\times)^2$

Def *(Quaternions)* $q = q_w + q_x i + q_y j + q_z k$ with $i^2 = j^2 = k^2 = -1$, ($ij = -ji = k$, same for jk and ki)

Def *(Transf. M)* $\boldsymbol{T}_{AB} = \begin{bmatrix} \boldsymbol{R}_{AB} & {}_A\boldsymbol{t}_B \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}$

$\boldsymbol{T}_{BA} = \boldsymbol{T}_{AB}^{-1} = \begin{bmatrix} \boldsymbol{R}_{AB}^\top & -\boldsymbol{R}_{AB}^\top {}_A\boldsymbol{t}_B \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}$ $\boldsymbol{T}_{AC} = \boldsymbol{T}_{AB}\boldsymbol{T}_{BC}$

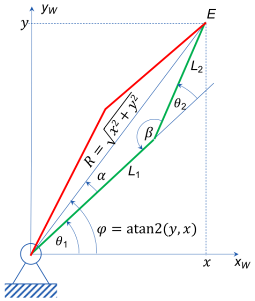
2.2 Forward Kinematics (FK)

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

For 2R system: $_W\boldsymbol{t}_{WE} = \begin{bmatrix} L_1\cos(\theta_1) + L_2\cos(\theta_1 + \theta_2) \\ L_1\sin(\theta_1) + L_2\sin(\theta_1 + \theta_2) \end{bmatrix}$

With workspace (pos) W for $\theta_1, \theta_2 \in [-\pi, \pi]$

2.3 Inverse Kinematics (IK)

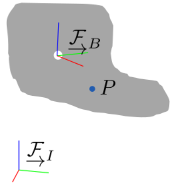


Option: Solve Forward Kinematics for angles.
Better: Law of cosine with polar coordinates. Compute angle using cosine rule,
$\theta_1 = \phi \pm \alpha$, $\theta_2 = \pm(\pi - \beta)$
(Positive for Elbow Down, Negative for Elbow Up)
Extension to 6R: 1. Waist: spherical coords (2 sol.)
2. 2 sols from 2R for shoulder + elbow
3. Solve for wrist joints (no influence on pos)

2.4 Temporal Models

For ***Cont-time n.-lin. system of ODE*** $\dot{\boldsymbol{x}} = \boldsymbol{f}_C(\boldsymbol{x}(t), \boldsymbol{u}(t))$, with measurements $\boldsymbol{z}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) + \boldsymbol{v}(t)$.
Need linearised (around $\boldsymbol{f}_C(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = 0$, at ***equilibrium***):
$\delta\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_C(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}}) + \boldsymbol{F}_C\delta\boldsymbol{x}(t) + \boldsymbol{G}_C\delta\boldsymbol{u}(t) + \boldsymbol{L}_C\boldsymbol{w}(t)$
$\delta\boldsymbol{z}(t) = \boldsymbol{H}\delta\boldsymbol{x}(t) + \boldsymbol{v}(t)$. Herein, $\boldsymbol{H}$ is measurements, $\boldsymbol{F}_C$ system, $\boldsymbol{G}$ input gain, $\boldsymbol{w}$ process noise, $\boldsymbol{v}$ measurement noise, both zero-mean ***Gaussian White Noise Process***.
For ***n-lin. cont-time system***: $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_C(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t))$
$\boldsymbol{z}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) = \boldsymbol{v})(t)$, linearization is the same
To ***discretize***, integrate from t_{k-1} to t_k:
$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{w}_k)$ $\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k$, ***linearised***:
$\delta\boldsymbol{x}_k = \boldsymbol{f}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{u}}) + \boldsymbol{F}\delta\boldsymbol{x}_{k-1} + \boldsymbol{G}_k\delta\boldsymbol{u}_k + \boldsymbol{L}_k\boldsymbol{w}_k$; $\delta\boldsymbol{z}_k = \boldsymbol{H}_k\delta\boldsymbol{x}_k$
Trapezoidal num. int $\Delta\boldsymbol{x}_1 = \Delta t\boldsymbol{f}_C(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, t_{k-1})$
$\Delta\boldsymbol{x}_2 = \Delta t\boldsymbol{f}_C(\boldsymbol{x}_{k-1} + \Delta\boldsymbol{x}_1, \boldsymbol{u}_k, t_k)$, then:
$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + 0.5 \cdot (\Delta\boldsymbol{x}_1 + \Delta\boldsymbol{x}_2)$

2.5 Rigid body & IMU kinematics



Velocity $_I\boldsymbol{v}_{IB} = \frac{\mathrm{d}}{\mathrm{d}t}({}_I\boldsymbol{t}_B)$
Rot. Velocity $_I\boldsymbol{\omega}_{IB} = \frac{\mathrm{d}}{\mathrm{d}t}(\alpha)\ {}_I\boldsymbol{t}$
Velocity point P $_B\boldsymbol{v}_{IP} = {}_B\boldsymbol{v}_{IB} + {}_B\boldsymbol{\omega}_{IB} \times {}_B\boldsymbol{t}_P$
Rotation Matrices

- For left pertubing $\dot{\boldsymbol{R}}_{IB} = [{}_I\omega_{IB}]^\times \boldsymbol{R}_{IB}$
- For right pertubing $\dot{\boldsymbol{R}}_{IB} = \boldsymbol{R}_{IB}[{}_I\omega_{IB}]^\times$
- Constant angular velocity ($\exp[\Delta\alpha]^\times = \delta\boldsymbol{R}(\Delta\alpha)$) $\boldsymbol{R}_{IB}(t + \Delta t) = \exp[\Delta\alpha]^\times \boldsymbol{R}_{IB}(t)$

Quaternions

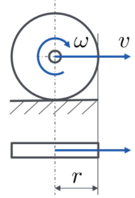
- For left pertubing $\dot{\boldsymbol{q}}_{IB} = \frac{1}{2}\begin{bmatrix} {}_I\boldsymbol{\omega}_{IB} \\ 0 \end{bmatrix} \otimes \boldsymbol{q}_{IB}$
- For right pertubing $\dot{\boldsymbol{q}}_{IB} = \frac{1}{2}\boldsymbol{q}_{IB} \otimes \begin{bmatrix} {}_B\boldsymbol{\omega}_{IB} \\ 0 \end{bmatrix}$

IMU (Outputs $_S\tilde{\boldsymbol{a}}$ (accel.), $_S\tilde{\boldsymbol{\omega}}$ (rot. accel.))
$_W\dot{\boldsymbol{t}}_S = {}_W\boldsymbol{v}$, $\dot{\boldsymbol{q}}_{WS} = \frac{1}{2}\boldsymbol{q}_{WS} \otimes \begin{bmatrix} {}_S\tilde{\boldsymbol{\omega}} + \boldsymbol{w}_g - \boldsymbol{b}_g \\ 0 \end{bmatrix}$
$_W\dot{\boldsymbol{v}} = \boldsymbol{R}_{WS}\ ({}_S\tilde{\boldsymbol{a}} + \boldsymbol{w}_a - \boldsymbol{b}_a) + {}_W\boldsymbol{g}$ where gray parts only IRL (in theor. models, leave out), with $\dot{\boldsymbol{b}}_g = \boldsymbol{w}_{b_g}$ and $\dot{\boldsymbol{b}}_a = \boldsymbol{w}_{b_a}$
IMU Sensor Model: $\tilde{\boldsymbol{z}} = \boldsymbol{b}_C + s\boldsymbol{M}\boldsymbol{z} + \boldsymbol{b} + \boldsymbol{n} + \boldsymbol{o}$ where bias $\boldsymbol{b}$ and scale s often modelled time-varying $\dot{b}(t) = \sigma_C n(t)$. $\boldsymbol{b}_C$ const. calib; $\boldsymbol{M}$ Misalignment; $\boldsymbol{n}$ noise; $\boldsymbol{o}$ other infl.

2.6 Rigid Body Dynamics

Def *(Newton II)* For fin. body w/ mass m and inertia mat. I, with force $\boldsymbol{F}$ and torque $\boldsymbol{T}$ on ***Centre of Mass*** (CoM), expressed in body frame:
$_B\boldsymbol{F} = \sum {}_B\boldsymbol{F}_i = m({}_B\dot{\boldsymbol{v}}_{CoM}) + m_B\boldsymbol{\omega} \times {}_B\boldsymbol{v}_{CoM}$
$_B\boldsymbol{T} = \sum {}_B\boldsymbol{T}_i = \boldsymbol{I}({}_B\dot{\boldsymbol{\omega}}) + {}_B\boldsymbol{\omega} \times \boldsymbol{I}_B\boldsymbol{\omega}$
$_B\boldsymbol{v}_{CoM}$ vel. of CoM, $_B\omega$ rot. speed; both w.r.t. world frame

2.7 Wheeled robot Kinematics



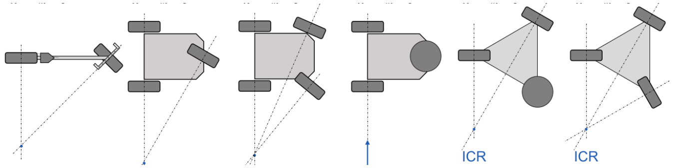
Non-holonomic systems **not integrable**, no inst. move in every direct.
Wheel constraints $v_i = \omega_i r_i$

- *Driving straight* all $\boldsymbol{v}$ equal
- *Turning* Wheel axis must intersect the ***Instant Centre of Rotation*** (ICR), speeds: $v_i \div R_i = \Omega$ (R_i dist. wheel-ICR, Ω, vehicle body rotation rate)

Maneuverability

- Deg. of Mobility: $\delta_m = 3 - \#$constrained directions
- Deg. of Steerability: $\delta_s = \#$steerable wheels
- Deg. of Maneuverability: $\delta_M = \delta_m + \delta_s$

Wheel Configurations



Bicyle	Tricycle	Ackermann	Diff. Drive	Two-Steer	Three-Steer
$\delta_m = 1$	$\delta_m = 1$	$\delta_m = 0$	$\delta_m = 1$	$\delta_m = 1$	$\delta_m = 0$
$\delta_s = 1$	$\delta_s = 1$	$\delta_s = 2$	$\delta_s = 0$	$\delta_s = 2$	$\delta_s = 3$
$\delta_M = 2$	$\delta_M = 2$	$\delta_M = 2$	$\delta_M = 2$	$\delta_M = 3$	$\delta_M = 3$

Differential Drive Kinematics

State vec $\boldsymbol{x} = [x_1, x_2, \theta]^\top$, ***Inputs*** $\boldsymbol{u} = [\omega_l, \omega_r]^\top$, r_r radius of right wheel, w width of robot

Gen. eq. of Motion $\dot{x}_1 = v\cos(\theta)$, $\dot{x}_2 = v\sin(\theta)$, $\dot{\theta} = \Omega$, with $v = 0.5 \cdot (\omega_l r_l + \omega_r + r_r)$, $\Omega = \frac{\omega_r r_r - \omega_l r_l}{w}$

Straight: $v = \omega_l r_l = \omega_r r_r$, $\Omega = 0$, $D = v\Delta t$.

$$\boldsymbol{b}_s = \begin{bmatrix} D\cos(\theta) \\ D\sin(\theta) \\ 0 \end{bmatrix} \quad \boldsymbol{b}_t = \begin{bmatrix} R(\sin(\Delta\theta + \theta) - \sin(\theta)) \\ -R(\cos(\Delta\theta + \theta) - \cos(\theta)) \\ \Delta\theta \end{bmatrix}$$

Turning: $\Omega = (\omega_l r_l)/R_l = (\omega_r r_r)/R_r$, $R = v/\Omega$, $\Delta\theta = \Omega\Delta t$

Discretized: $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} b_i$ with $i \in \{s, t\}$. ($\int \dots \, \mathrm{d}\Delta t$)

3 Sensors & Actuators

Meas. Model: $\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{v} + \boldsymbol{o}$, with $\boldsymbol{h}(\boldsymbol{x})$ deterministic mean, $\boldsymbol{v}$ zero-mean noise, $\boldsymbol{o}$ unmodelled effects, $\boldsymbol{x}$ true state

Motor encoders Typ. 64-2048 incrm. per rev; Estim. rot

Rolling-Shutter Most CMOS sensors don't take full image at once, need time stamp for each row

3.1 GNSS

Need ultra-precise time sync ($c \approx 0.3\ m/ns$). ***Errors***

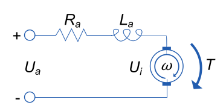
- Multipath problem (signal bounce) (0.5 - 100m)
- Ionosphere delays (10m)
- Satellite pos. err, trop. delay (1m)

3.2 Actuators

Hydraulic acc., easy control, power; maint., speed, price

Pneumatic price, shock abs., speed; acc., loud, maint.

3.2.1 DC Motor



(Kirchoff) $U_a = L_a \dot{I}_a + R_a I_a + U_i$

(Torque, Lorenz Force) $T = k_T I_a$

(Induced V, Faraday) $U_i = k_i \omega$

(Mech. pow. eq. el. pow)

$U_i I_a = k_i \omega I_a = T_\omega = k_T I_a \omega \Rightarrow k_i = k_T =: k$

3.3 Cameras

Def *(Pinhole projection)* $\begin{bmatrix} u & v \end{bmatrix}^\top = \frac{f}{z}\begin{bmatrix} x & y \end{bmatrix}^\top$ with f the distance to the lens and z the full distance

$u = c_u + f \cdot x'$ and $v = c_v + f \cdot y'$ where $x' = t_x \div t_z$ and $y' = t_y \div t_z$ where u, v are the pixel x, y coords, $\boldsymbol{c} = [c_u, c_v]^\top$ is optical centre of cam in pixel coords, f scale factor, and ${}_{\boldsymbol{C}}\boldsymbol{t}_{\boldsymbol{P}} = [t_x, t_y, t_z]^\top$

$$\text{The full proj: } \boldsymbol{u} = \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \boldsymbol{K}\ {}_C\boldsymbol{t}_P$$

If p. in diff frame ${}_W\boldsymbol{t}_P$, then $\boldsymbol{u} = \boldsymbol{K}[\boldsymbol{R}_{CW}\ {}_C\boldsymbol{t}_{CW}] = {}_W\boldsymbol{t}_P$

3.3.1 Pinhole Camera Projection with distortion

Def Model: $\boldsymbol{u} = \boldsymbol{k}(\boldsymbol{d}(\boldsymbol{p}({}_C\boldsymbol{t}_P)))$, with:

(Projection) $\boldsymbol{x'} = \boldsymbol{p}({}_C\boldsymbol{t}_P) = t_z^{-1} \cdot [t_x, t_y]^\top$

(Distortion model, $r^2 = x'^2 + y'^2$, $\boldsymbol{x''} = \boldsymbol{d}(\boldsymbol{x'})$)

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

(Scale and Centre) $\boldsymbol{u} = \boldsymbol{k}(\boldsymbol{x''}) = \mathrm{diag}([f_u, f_v]) \cdot \boldsymbol{x''} + \boldsymbol{x}$

All with k_i radial distortion params, optional for $i > 2$, p_i tang. dist. param, f_u, f_v focal length in pixels

Inverse ${}_C\boldsymbol{r} = [\boldsymbol{d}^{-1}(\boldsymbol{k^{-1}}(\boldsymbol{u})), 1]^\top$

(To unit plane) $\boldsymbol{x''} = \boldsymbol{k}^{-1}(\boldsymbol{u}) = [f_u^{-1}, f_v^{-1}]^\top(\boldsymbol{u} - \boldsymbol{c})$

(Un-distort) $\boldsymbol{x'} = \boldsymbol{d}^{-1}(\boldsymbol{x''})$ (usually comp. numerically)

(Compute ray) ${}_C\boldsymbol{r} = [\boldsymbol{x'}, 1]^\top$

3.3.2 Undestorting a whole image

$\boldsymbol{u}_i = \boldsymbol{k}(\boldsymbol{d}(\boldsymbol{k}_{\text{new}}^{-1}(\boldsymbol{u}_{i,\text{new}})))$ where $\boldsymbol{u}_{i,\text{new}}$ is the place of pixel in output, $\boldsymbol{u}_i$ is the input

Omnidir. Cam undistortion model with $f(u, v) = \sum_{i=0}^{N} a_i \rho^i$ with $\rho = \sqrt{(u - c_u)^2 + (v - c_v)^2}$, $N = 4$ accurately describes it for most fisheye and catadioptric cameras

3.4 Depth and Range sensing

3.4.1 Triangulation-based

Struct. Light Single cam, single projector: Spatial acc, no worky in bright light, interference with other IR depth cams

Active Stereo 2 cams, 1 proj: worky in bright light, need stereo matching, less accurate, error grows with distance

3.4.2 Classic Stereo

Both images: same plane, focal length, centre, x-axis. Given corresponding pixels $[u_l, v]$ and $[u_r, v]$, $z = \frac{b \cdot f}{u_r - u_l}$ with $u_l = f \cdot \frac{x}{z} + c_u$ and $u_r = f \cdot \frac{x - b}{z} + c_u$

3.4.3 Time of Flight, Projection

No occlusions/shadows, Interference with other dev, multipath leading to larger distances sensed

$$\textit{\textbf{Proj.}}\ \boldsymbol{z} = \begin{bmatrix} \boldsymbol{u} \\ d \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{d}(\boldsymbol{p}({}_C\boldsymbol{t}_P))) \\ [0, 0, 1]_c\boldsymbol{t}_P \end{bmatrix} \quad \textit{\textbf{Back}}\text{: } {}_C\boldsymbol{t}_P = \begin{bmatrix} d\boldsymbol{x'} \\ d \end{bmatrix}$$

3.4.4 Range Sensors

Ultrasonic Typ. freq: 40kHz - 180kHz, Range: 12cm - 5m, Acc: $\approx$ 2cm, rel error $\approx$ 2% meas. for transp. surf., cheap(ish), Cone wider, reflect. angle dep, wind / currents

LiDAR Time-of-Flight-based, Accuracy, range, works in bright light, Complex, expensive, one timestamp per measurement. Typical Ranges: up to 100m

4 Multi-Sensor Estimation

4.1 Linearization

$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\overline{\boldsymbol{x}}) + \boldsymbol{J_f}\big|_{x = \overline{x}}(\boldsymbol{x} - \overline{\boldsymbol{x}})$, f', no vec in 1D; $\overline{\boldsymbol{x}}$ lin. p.

Def *(Jac.)* $\boldsymbol{J_f}$ rows for eq of $\boldsymbol{f}$ cols for vars of each eq.

Approximation using finite differences $\frac{f(\overline{x} + h) - f(\overline{x})}{h}$,

or central differences (vector of $\frac{\boldsymbol{f}(\overline{\boldsymbol{x}}) + h_i \boldsymbol{e_i}}{h_i}$, with $\boldsymbol{e_i}$ unit vec)

4.2 Linear Least Squares

Goal: $\underset{x \in \mathbb{R}^n}{\text{argmin}} ||\boldsymbol{A}\boldsymbol{x} - b||_2^2$, $\boldsymbol{A}$: rows i-th datap. col c: t_i^{c-1}.

Man. sol.: comp. $M = A^\top A$, $b' = A^\top b$, then $Mx = b'$.

Prob. sol.: $\text{argmax} p(\boldsymbol{x} \mid \boldsymbol{z})$ with

- ***Max. Like*** $p(\boldsymbol{x}|\boldsymbol{z}) \propto p(\boldsymbol{z}|\boldsymbol{x}) = \prod_{i=1}^{N} p(\boldsymbol{z}_i|\boldsymbol{x})$
- ***M a Post*** $p(\boldsymbol{x}|\boldsymbol{z}) \propto p(\boldsymbol{z}|\boldsymbol{x}) p(\boldsymbol{x}) = p(\boldsymbol{x}) \prod_{i=1}^{N} p(\boldsymbol{z}_i|\boldsymbol{x})$

4.3 Non-Linear Least Squares

Find $\boldsymbol{x}^* = \text{argmax} p(\boldsymbol{x}|\boldsymbol{z}) = \text{argmin}(-\log(p(\boldsymbol{x}|\boldsymbol{z})))$

Gauss-Newton

Levenberg-Marquardt

Local Param.

4.4 Bayes Filter

$\boldsymbol{x}_k^R$ state at time k, $\boldsymbol{z}_k^p$ dist. meas., $\boldsymbol{u}_k^p$ wheel odometry (= meas.). Typ. care ab. curr. state: altern. pred. & update.

4.5 Particle Filter

Is a bayes filter approximating the state distribution with a set of random samples. Update step:

- Apply Bayes rule $w'_{k,s} = \mathbb{P}[\boldsymbol{z}_i \mid \boldsymbol{x}_{k,s}] w_{k-1,s}$
- Renormalize: $w_{k,s} = w'_{k,s} \div \sum_s w'_{k,s}$
- Resample: rand. sel. S particles acc. to weights and $w_{k,s} = S^{-1}$

4.6 Kalman Filtear (KF)

Bayes Filter for Gauss. dist of R.V. & linear meas. model. Initial state $\boldsymbol{x}_0 \sim \mathcal{N}(\widehat{\boldsymbol{x}}, \boldsymbol{P}_0)$, $\boldsymbol{P}_0$ previous covariance;

Prediction With linear state transition model ($\boldsymbol{u}_k$ odometry, $\boldsymbol{w}_k$ noise (covariance $\boldsymbol{Q}_k$)):

$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{w}_k$ with $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k)$:

- ***Mean*** $\widehat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\widehat{\boldsymbol{x}}_{k-1} + \boldsymbol{G}\boldsymbol{u}_k$
- ***Covariance*** $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^\top + \boldsymbol{L}\boldsymbol{Q}_k\boldsymbol{L}^\top$

Update Lin. meas.: $\widetilde{\boldsymbol{z}}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k$ with $\boldsymbol{v_k} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$:

- ***Meas. residual***: $\boldsymbol{y}_k = \widetilde{\boldsymbol{z}}_k - \boldsymbol{H}\widehat{\boldsymbol{x}}_{k|k-1}$
- ***Resid. Cov***: $\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^\top \boldsymbol{S}_k^{-1}$
- ***Kalman gain***: $\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^\top \boldsymbol{S}_k^{-1}$
- ***Updated mean***: $\widehat{\boldsymbol{x}}_{k|k} = \widehat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \boldsymbol{y}_k$
- ***Updated Cov.***: $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_{k|k-1}$

4.7 Extended Kalman Filater (EKF)

Non-l. state trans. model $\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{w}_k)$ as above:

- ***Mean***: $\widehat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{f}(\widehat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k)$
- ***Cov.***: $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_k^\top + \boldsymbol{L}_k\boldsymbol{Q}_k\boldsymbol{L}_k^\top$
  With $\boldsymbol{F}_k$ linearisation $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ and $\boldsymbol{L}_k$ lin. $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}$

Update N-Lin. meas.: $\widetilde{\boldsymbol{z}}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k$:

- ***Meas. residual***: $\boldsymbol{y}_k = \widetilde{\boldsymbol{z}}_k - \boldsymbol{h}(\widehat{\boldsymbol{x}}_{k|k-1})$

Difference to above: $\boldsymbol{H}$ becomes $\boldsymbol{H}_k$, and $\boldsymbol{H}^\top$ is $\boldsymbol{H}_k^\top$

5 SLAM to Spatial AI

5.1 Keypoints

Corner det. $SSD(\Delta_x, \Delta_y) \approx [\Delta_x \ \Delta_y]\boldsymbol{M}[\Delta_x \ \Delta_y]^\top$ with $\boldsymbol{M} = \boldsymbol{R}^\top \text{diag}(\lambda_1, \lambda_2)\boldsymbol{R}$; λ_i E.V. of M; $\boldsymbol{R} = \det(M) - \kappa \cdot \text{trace}(M)^2 = \lambda_1\lambda_2 - \kappa(\lambda_1+\lambda_2)^2$; $M = \sum_{x,y\in P} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$

Blob Detection (I is the image)
Laplacian of Gaussian (LoG): $L = g(x,y,t)\cdot I(x,y)$. Then apply Laplacian Operator $\nabla^2_{\text{norm}} L = t\left(\frac{\partial^2 L}{\partial x^2} + \frac{\partial L}{\partial y^2}\right)$
Diff. of Gaussians (DoG): $\Delta L = L(x,y,t) - L(x,y,kt)$
SIFT Detector (1) Subsample + Blur ***(2)*** DoG on each res. image ***(3)*** Keypoints extrema in DoG pyramid

5.2 Bootstrapping

PnP Problem Persp. n-P. Find sol. for camera pose *directly*
RANSAC RANdom SAmpling Consensus for find. outliers & correct
Stereo Triang. Given two rays (known poses for points in 2D). Find good point in 3D. Fast sol: ***Midpoint Method***:
1 Find p. along ray w/ min. dist (Lin. Least Squares)
$\boldsymbol{\lambda} = [\lambda_1 \ \lambda_2]^\top = \text{argmin}||({}_W\boldsymbol{t}_{C_2} + \lambda_2 {}_W\boldsymbol{e}_2) - ({}_W\boldsymbol{t}_{C_1} + \lambda_1 {}_W\boldsymbol{e}_2)||^2$
2 Solve normal equation $\boldsymbol{A\lambda} = \boldsymbol{b}$ with $\boldsymbol{q} = -{}_W\boldsymbol{e}_1^\top {}_W\boldsymbol{e}_2$:

$$\boldsymbol{A} = \begin{bmatrix} 1 & \boldsymbol{q} \\ \boldsymbol{q} & 1 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{e}_1^\top \cdot ({}_W\boldsymbol{t}_{C_2} - {}_W\boldsymbol{t}_{C_1}) \\ -\boldsymbol{e}_2^\top \cdot ({}_W\boldsymbol{t}_{C_2} - {}_W\boldsymbol{t}_{C_1}) \end{bmatrix}$$

3 Pick midp. ${}_W\boldsymbol{t}_P = 0.5(\tau_1 + \tau_2)$; $\tau_n = {}_W\boldsymbol{t}_{C_n} + \lambda_n {}_W\boldsymbol{e}_n)$

5.3 Place Recognition

Idea: Build vocab from "visual words" (in Training). ***Runtime*** Detect keypoints; Extract descriptors; Build histogram; Query DB for similarity; If no match, insert into DB, else use to compute with e.g. RANSAC

5.4 Mapping

Problem Formulations *Localisation* (always static given map):
$\boldsymbol{x}^*_{R,k} = \text{argmax}\ \mathbb{P}(x_{R,k} \mid \boldsymbol{x}_M, \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k})$ (recursive)
$\boldsymbol{x}^*_{R,1:k} = \text{argmax}\ \mathbb{P}(x_{R,1:k} \mid \boldsymbol{x}_M, \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k})$ (batch)
SLAM $\{\boldsymbol{x}^*_{R,k}, \boldsymbol{x}^*_M\} = \text{argmax}\ \mathbb{P}(\boldsymbol{x}_R, \boldsymbol{x}_M \mid \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k})$ (as ↑)
Mapp.: $\boldsymbol{x}^*_M = \text{argmax}\ \mathbb{P}(\boldsymbol{x}_M \mid \boldsymbol{x}^*_{R,1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k})$ with given poses $\boldsymbol{x}^*_{R,1:k}$. Thus temp. model $\boldsymbol{u}_{1:k}$ doesn't matter.
Prob. Occ. Grid $\mathbb{P}(f_j) = \mathbb{P}(\neg o_j) = 1 - \mathbb{P}(o_j)$, with $\mathbb{P}(o_j)$ prob. cell j occupied; pairwise independent.
Occ. Map. w/ depth sensor $\mathbb{P}(o_j \mid \boldsymbol{x}_{R,1:k}) =$

$$\frac{\mathbb{P}(o_j \mid \boldsymbol{x}_{R,k}, \boldsymbol{z}_k)\mathbb{P}(\boldsymbol{z}_k \mid \boldsymbol{x}_{R,k})\mathbb{P}(o_j \mid \boldsymbol{x}_{R,1:k-1}, \boldsymbol{z}_{1:k-1})}{\mathbb{P}(o_j)\mathbb{P}(\boldsymbol{z}_k \mid \boldsymbol{x}_{R,1:k}, \boldsymbol{z}_{1:k-1})}$$

map prior; Prev. occ. est.; Occ. based on curr. range meas.;
Update function: $l(a) := \log(\text{Odds}(a))$ with $\text{Odds}(a) = \frac{\mathbb{P}(a)}{1-\mathbb{P}(a)}$ (inv. sensor model): $l(o_j|\boldsymbol{x}_{R,1:k}, \boldsymbol{z}_{1:k}) =$

$$l(o_j|\boldsymbol{x}_{R,k}, \boldsymbol{z}_k) + l(o_j|\boldsymbol{x}_{R,1:k-1}, \boldsymbol{z}_{1:k-1}) - l(o_j)$$

In 3D 3D voxel j as signed dist. s and weight w, update:

$$s_k = \frac{w_{k-1}s_{k-1} + \widetilde{s}_k}{w_{k-1}+1} \text{ with } w_k = \min(w_{\max}, w_{k-1}+1)$$

Impl. Using HashTables or octree

5.5 Dense Tracking and Loop Closure

Idea: Min. sum of sq. errors over all pixels w/ Gauss-Newton.