# 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** $E[x] = \int_{-\infty}^{\infty} \boldsymbol{x} p(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x}$, 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.)* $\boldsymbol{R}_z = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

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

**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 $\underrightarrow{\mathcal{F}}_B$ in $\underrightarrow{\mathcal{F}}_W$

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

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

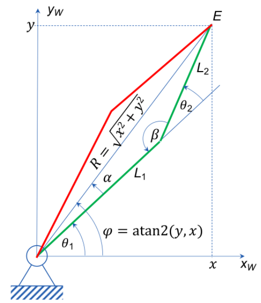
## 2.2 Forward Kinematics (FK)

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

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)

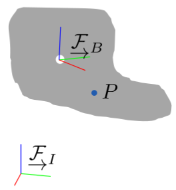



***Option***: Solve Forward Kinematics for angles.
***Better***: Law of cosine with polar coordinates. Compute angle using cosine rule,
$\theta_1 = \phi \pm \alpha$, $\theta_2 = \pm(\pi - \beta)$
(Positive for Elbow Down, Negative for Elbow Up)
***Extension to 6R***: 1. Waist: spherical coords (2 sol.)
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.))
${}_W\dot{\boldsymbol{t}}_S = {}_W\boldsymbol{v}$, $\dot{\boldsymbol{q}}_{WS} = \frac{1}{2}\boldsymbol{q}_{WS} \otimes \begin{bmatrix} {}_S\tilde{\boldsymbol{\omega}} + w_g - b_g \\ 0 \end{bmatrix}$
${}_W\dot{\boldsymbol{v}} = \boldsymbol{R}_{WS}\ ({}_S\tilde{\boldsymbol{a}} + \boldsymbol{w}_a - b_a) + {}_W\boldsymbol{g}$ 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

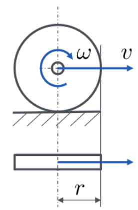
**Def** *(Newton II)* For fin. body w/ mass $m$ and inertia mat. $I$, with force $\boldsymbol{F}$ and torque $\boldsymbol{T}$ on ***Centre of Mass*** (CoM), expressed in body frame:

$$ {}_B\boldsymbol{F} = \sum {}_B\boldsymbol{F}_i = m({}_B\dot{\boldsymbol{v}}_{CoM}) + m_B\boldsymbol{\omega} \times {}_B\boldsymbol{v}_{CoM} $$

$$ {}_B\boldsymbol{T} = \sum {}_B\boldsymbol{T}_i = \boldsymbol{I}({}_B\dot{\boldsymbol{\omega}}) + {}_B\boldsymbol{\omega} \times \boldsymbol{I}_B\boldsymbol{\omega} $$

${}_B\boldsymbol{v}_{CoM}$ vel. of CoM, ${}_B\omega$ rot. speed; both w.r.t. world frame

## 2.7 Wheeled robot Kinematics



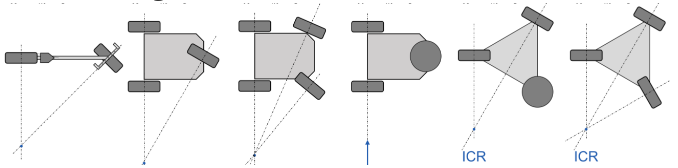

***Non-holonomic*** systems **not integrable**, no inst. move in every direct.
***Wheel constraints*** $v_i = \omega_i r_i$

- *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, $\Omega$, 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 |
|---|---|---|---|---|---|
| $\delta_m = 1$ | $\delta_m = 1$ | $\delta_m = 0$ | $\delta_m = 2$ | $\delta_m = 1$ | $\delta_m = 0$ |
| $\delta_s = 1$ | $\delta_s = 1$ | $\delta_s = 2$ | $\delta_s = 0$ | $\delta_s = 2$ | $\delta_s = 3$ |
| $\delta_M = 2$ | $\delta_M = 2$ | $\delta_M = 2$ | $\delta_M = 2$ | $\delta_M = 3$ | $\delta_M = 3$ |

***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

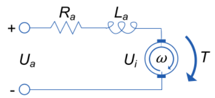
(Kirchoff) $U_a = L_a\dot{I}_a + R_a I_a + U_i$

(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 ${}_C\boldsymbol{t}_P = [t_x, t_y, t_z]^\top$

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

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

### 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