Cheatsheet MEAM 620
fclad@seas.upenn.edu, Pag. 1 of 2

1 Kinematics Equations

Rotation Matrix

$${}^{A}\mathbf{R}_{B} = \begin{bmatrix} \mathbf{a}_1\cdot\mathbf{b}_1 & \mathbf{a}_1\cdot\mathbf{b}_2 & \mathbf{a}_1\cdot\mathbf{b}_3 \\ \mathbf{a}_2\cdot\mathbf{b}_1 & \mathbf{a}_2\cdot\mathbf{b}_2 & \mathbf{a}_2\cdot\mathbf{b}_3 \\ \mathbf{a}_3\cdot\mathbf{b}_1 & \mathbf{a}_3\cdot\mathbf{b}_2 & \mathbf{a}_3\cdot\mathbf{b}_3 \end{bmatrix}$$

$${}^{A}\mathbf{R}_{B} = \left[\, {}^{A}[\mathbf{b}_1] \quad {}^{A}[\mathbf{b}_2] \quad {}^{A}[\mathbf{b}_3] \right]$$

${}^{A}R_B$ are the basis vectors of B represented in the coordinates of A.

Skew-symmetric matrix

$$[a]_\times = \hat{a} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

Angular velocities

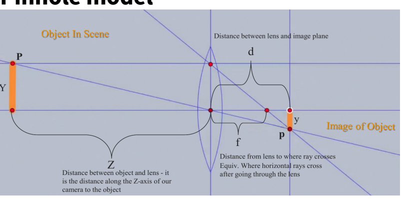
$\hat{\omega}_b = R^T\dot{R}$

$\hat{\omega}_s = \dot{R}R^T$

where R has to be in inertial frame (${}^{W}R_C$).

2 CV basics

Pinhole model



$1/f = 1/Z + 1/d$

$Y/Z = y/d \Rightarrow y = dY/Z$

Problem: d changes depending where the object is in the scene. Hence: $y \approx fY/Z$

Only valid if $Z >> d$

Calibration Procedure

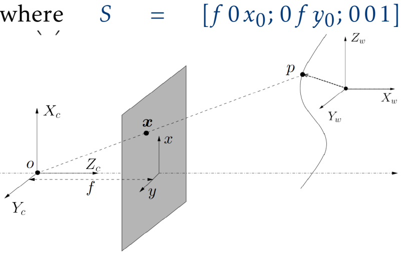
Finds f and Radial distortion:

$r = norm(x,y), x' = x(1 + k_1 t + k_2 r^2 + k_3 r^3$ (and similar for y)

Projection Equation

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim S\underbrace{\begin{pmatrix} r_1 & r_2 & r_3 & t \end{pmatrix}}_{{}^{C}R_W|{}^{C}T_W}\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

where $S = [f\,0\,x_0; 0\,f\,y_0; 0\,0\,1]$



t is the distance from the camera to the world coordinate system in camera coordinates

Projective Geometry

- Points at the infinity (do not intersect the 2D plane): $\begin{pmatrix} x & y & 0 \end{pmatrix}^T$
- Points in the image (intersect the 2D plane): $\begin{pmatrix} x & y & 1 \end{pmatrix}^T$

Lines $\rightarrow$ represented by normal vectors $l^T p = 0$

- Line that goes trough two points: $l \sim p \times q$
- intersection of two lines: $p \sim l \times l'$
- if three points are collinear: $p^T(q \times r) = 0$
- if three lines are concurrent: $l^T(l' \times l'') = 0$
- Lines **in the image** are represented by **points in the infinity**
- Points **in the image** represent **lines in the infinity**.

3 Homographies

Properties

- H has 8 DOF, it can be determined up to a scale.
- H has to be invertible, $det(H) \neq 0$.
- It maps lines to lines: $l^T p = 0, p' \sim Hp \Rightarrow l^T H^{-1} p' \Rightarrow l' \rightarrow H^{-T} l$
- Preserves incidence (three collinear points remain collinear): $p^T(q \times r) = 0 \Rightarrow (Hp)^T(Hq \times Hr) \Rightarrow det(H)p^T(q \times r) = 0$
  Property
  $(Ma) \times (Mb) = det(M)M^{-T}(a \times b)$
- The determinant of a Rot. + Transl. is $det(r_1\, r_2\, T) = T^T(r_1 \times r_2)$, which vanishes if the camera is in the Z plane.

Computing homographies

- Four point: Each point gives 2 equations. $Ah = 0 \Rightarrow USV^T = A$
  $h = V(9)$
- 2 Points infinity, origin, $(1\,1\,1)$. Projections A, B, C, D.
  $H \sim (\alpha A\, \beta B\, \gamma C)$
  $(\alpha\, \beta\, \gamma)^T = (A\,B\,C)^{-1} D$

Constant plane in X, Y, Z

Eg. for $X = h$: $H = \begin{pmatrix} r_2 & r_3 & hr_1 + t \end{pmatrix}$

Plane constraint

Eg: $AX_w + BY_w + CZ_w = 1$

- Substitute the 1 in the last position. Expand.
- Replace $Z_w = 1/C(1 - AX_w - BY_w)$

Pose estimation

$$\begin{pmatrix} \hat{R}_1 & \hat{R}_2 & \hat{T} \end{pmatrix} = \underbrace{\begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1}\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_{K^{-1}H}$$

To solve:

$argmin_{R \in SO(3)}$

$||R - \begin{pmatrix} \hat{R}_1 & \hat{R}_2 & \hat{R}_1 \times \hat{R}_2 \end{pmatrix}|^2$

Solve with SVD

$\begin{pmatrix} \hat{R}_1 & \hat{R}_2 & \hat{R}_1 \times \hat{R}_2 \end{pmatrix} = USV^T$

Finally:

$$R = U\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^T) \end{pmatrix}V^T$$

We do this trick to force the determinant of R is 1

To get the scale here, we use the average

$T = \hat{T}/.5 \cdot (|\hat{R}_1| + |\hat{R}_2|)$

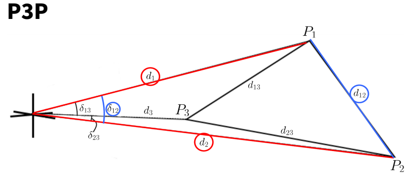
4 PnP

- **Got:** 2D-3D correspondences
- **Want:** T, R between camera and world coordinate system.

Solutions

- Linear hack (hw 3)
  $$\lambda_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = R\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + T$$
  Expand the matrices. Solve for λ_i, and get two equations per point. **We need at least 6 points.**
- Non-linear optimization
- P3P

P3P



Simple formulation:

$d_i^2 + d_j^2 - 2d_i d_j \cos(\delta_{ij}) = d_{ij}^2$

Use this for simple problems!

Complex formulation: we use $d_2 = ud_1$, $d_3 = vd_1$, then:

$d_{13}^2(u^2 + v^2 - 2uv\cos(\delta_{23})$

$= d_{23}^2(1 + v^2 - 2v\cos(\delta_{13}))$

$d_{12}^2(1 + v^2 - 2v\cos(\delta_{13}))$

$= d_{13}^2(1 + u^2 - 2u\cos(\delta_{12}))$

1. Solve u^2 in 1.
2. Insert u^2 back into 2.
3. Solve u, leaving terms in v and v^2.
4. Insert u back into 1. Quartic polynomial in v. **At most 4 solutions.**

To find the angles, we can use the dot product of the rays. Each pixel on the camera image denotes a direction. Using camera K matrix, you can find the vector in the camera frame. Using inner products give the angle cosines. After solving P3P, we have to use Procrustes to find R and T.

5 Procrustes (3D-3D registration)

- **Got:** two set of 3D points
- **Want:** R, T such that
  $\min_{R,T} \sum |A_i - RB_i + T|^2$

Procrustes solution:

We need at least three point correspondences.

1. $\overline{P} = \frac{1}{N}\sum P_i$, $barP' = \frac{1}{N}\sum P_i'$
2. $Z = \sum(P_i - \overline{P})(P_i' - \overline{P'})^T = USV^T$
3. Rectify R using the same trick we use in pose estimation.
4. $T = \overline{P'} - R\overline{P}$

6 Optical Flow

$$\mathbf{p} = \frac{1}{Z}A(\mathbf{p})V + B(\mathbf{p})\Omega$$

where

$$A(\mathbf{p}) = \begin{pmatrix} -1 & 0 & p_x \\ 0 & -1 & p_y \end{pmatrix}$$

$$B(\mathbf{p}) = \begin{pmatrix} -p_x p_y & -(p_x^2 + 1) & p_y \\ 1 + p_y^2 & -p_x p_y & -p_x \end{pmatrix}$$

V: velocity in **inertial frame**

Ω: ang velocity in **inertial frame**

$\mathbf{p}$: 2D point in the image

$\dot{\mathbf{p}}$: 2D velocity in the image

How to find this?

$\mathbf{p} = \mathbf{P}/Z$

$\dot{\mathbf{p}} = \dot{\mathbf{P}}/Z - \dot{Z}/Z\mathbf{p}$

$\dot{Z} = e_3^T\dot{\mathbf{P}}$

$\dot{\mathbf{P}} = -\mathbf{V} - \mathbf{\Omega} \times \mathbf{P}$

$= 1/Z(\mathbf{p}e_3^T - I)\mathbf{V} + (I - \mathbf{p}e_3^T)[p]_\times\mathbf{\Omega}$

Cases

- Known depth:
  $\mathbf{V}, \mathbf{\Omega} = argmin_{\mathbf{V},\mathbf{\Omega}}$
  $$\sum \left| \begin{pmatrix} \frac{1}{Z_i}A(\mathbf{p}_i) & B(\mathbf{p}_i) \end{pmatrix}\begin{pmatrix} \mathbf{V} \\ \mathbf{\Omega} \end{pmatrix} - \dot{\mathbf{p}}_i \right|^2$$
- No translational vel:
  Useful when the drone is flying very high, or if we want to track the stars.
  $\Sigma^* = argmin_\Sigma \sum |B(\mathbf{p}_i\Sigma - \dot{\mathbf{p}}_i|^2$
- No angular vel:
  We use the cross product trick to eliminate Z_i $\dot{\mathbf{p}} = \frac{1}{Z_i}A(\mathbf{p})\mathbf{V}$
  $[\dot{\mathbf{p}}]_\times A(\mathbf{p})\mathbf{V} = 0$
  And we can do SVD here.
- Everything unknown. Difficult problem.

Optical flow as a local search

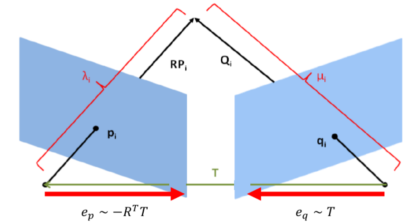
Assumptions:

- Brightness consistency
- Minimal geometric deformations
- Minimal patch displacement
- Patch is sufficiently interesting
- Wall is not "white".

Barber poll issue: we cannot see if it moves in a specific direction.

Finding features(SURF) -> **Track them** (KLT)

7 Structure from Motion (SFM)



- **Got:** 2D-2D correspondences between two views p, q.
- **Want:** T, R between two views.

$\mu q = R\lambda p + T$

T is here in the coordinates of q.

$DOF = 6$ (3 translation, 3 rotation), but we can only find the translation up to a scale. Finally, $DOF = 5$.

Epipolar constraint

From the image, we see that the vectors μq, T and λRp are coplanar. We can write the triple product:

$\mu q^T(T \times \lambda Rp) = 0$

$q^T(T \times Rp) = 0$

$q^T\hat{T}Rp = 0$

$q^T Ep = 0$, where $E = \hat{T}R$ essential matrix.

The planes spanned by T, λq and μp is called epipolar plane (cross product vanishes).

Why we cannot recover the scale?

Scaling q or p will not violate the constraint. Therefore, we can obtain the translation up to a scale.

Fundamental Matrix

If our points are not calibrated, we have to calibrate them:

$F = K^{-T}\hat{T}RK'^{-}1$

Epipolar Line

In p-plane, line with coefficients $E^T q$. All epipolar lines go through the epipole e_p:

$Ee_p = \hat{T}R(-R^T T) = \hat{T}T = T \times T = 0$

Essential Matrix Calculation

8-point algorithm: because we are finding a linear solution

$E = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix}$

$$q^T\begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix}\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = 0$$

$$= \begin{pmatrix} p_x q^T & p_y q^T & p_z q^T \end{pmatrix}\begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

$$\mathbf{a} = \begin{pmatrix} p_x q^T & p_y q^T & p_z q^T \end{pmatrix}\begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{pmatrix}E' = 0$$

and we do SVD. Practically:

$$E = U\mathrm{diag}\left(\frac{\sigma_1 + \sigma_2}{2}, \frac{\sigma_1 + \sigma_2}{2}, 0\right)V^T\mathrm{j}$$

Other options for estimation

- 5 point algorithm. Quite complex. Minimum number.
- 7 point algorithm. Easier, don't need to do full estimation.

Always RANSAC when estimating E, we can have bad points matches.

Essential matrix properties

$E=\hat{T}R \Rightarrow EE^T = \hat{T}\hat{T}^T = TT^T - T^T T I$

$= \begin{pmatrix} t_x^2 & t_xt_y & t_xt_z\\ t_xt_y & t_y^2 & t_yt_z \\ t_xt_z & t_yt_z & t_z^2\end{pmatrix} - |T|^2 I$

If we solve $det(EE^T - \lambda I)=0$, we find two eigenvalues $|T|^2$.
To be essential, E should have $\sigma_1 = \sigma_2 > 0$ and $\sigma_3 = 0$

Essential matrix decomposition

Useful properties:

$\widehat{Qa} = Q\hat{a}Q^T$

$\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix} = \hat{T}_z^T R_{z,\pi/2}$

Where

$T_{z,\pi/2} = \begin{pmatrix}0&1&0\\-1&0&0\\0&0&0\end{pmatrix}$

Therefore we can write $E = \sigma U \begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix} V^T = \sigma \underbrace{U\hat{T}_z^T}_{\text{antisymmetric}}\ \underbrace{R_z V^T}_{\text{orthogonal}}$

Two R:
- $R_1 = UR^T_{z,\pi/2}V^T$
- $R_2 = UR^T_{z,-\pi/2}V^T$

Two $\hat{T}$:
- $T_1 = UR^T_{z,\pi/2}\Sigma U^T$
- $T_2 = UR^T_{z,-\pi/2}\Sigma U^T$

Finally, disambiguate with $\lambda q = \mu R p + T$ such that $\lambda, \mu > 0$

Triangulation
- **Got:** T, R and 2D correspondences in two images.
- **Want:** Depth of points in each camera μ_i, λ_i.

We set the translation to $|T| = 1$

$\underbrace{(q_i \quad -Rp_i)}_{3\times2}\underbrace{\begin{pmatrix}\mu_i\\\lambda_i\end{pmatrix}}_{2\times1} = \underbrace{T}_{3\times1}$

3 eqs with 2 unknowns, solve with pseudo inverse

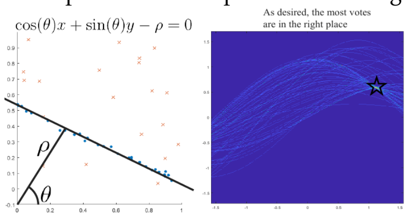
Tips and Tricks
- $R = I$, the epipolar constraint is $q^T(T\times p)=0$ or $(q\times p)^T T=0$. We need 2 points to solve the problem (up to a scale).

- Pure translation essential matrix: we can recognize if it is skew symmetric. If E is antisymmetric, that means that $R = I$!

8 RANSAC

Hough transform

Parametrize objects that you are trying to fit. For each point, iterate through all the objects that fit with this point. Plot in parameter space→ voting.



Issues: Memory grows exponentially in parameter space. Need bounds.

Sample consensus

RANSAC with all the points, count the one that has the most inliers. **Problem**: a lot of combinations.

RANSAC inliers

$k = \dfrac{\log(1-p_{success})}{\log(1-\epsilon^M)}$

k: iterations, $p_{success}$: target success probability, ϵ: inlier proportion in set, M: min. number of points for model.

9 VO

- **SFM**: 3D reconstruction and pose estimation from image sets.
- **VO**: focus on estimation and local consistency.
- **SLAM**: focus on global consistence. VSLAM = visual odometry + loop closure + graph optimization.

Why VO? No wheel slip. More accurate trajectory estimates. GPS-denied environments.
Why not? Low illumination. A lot of moving objects. Not texture.
Solution: complementary sensor suite. Camera + IMU.

10 Kalman Filter

Assumptions
- $p(x_0)\sim N(\mu_0,\Sigma_0)$
- $p(x_t|x_{t-1},u_t)$ linear, AGWN.
  - $x_t = A_tx_{t-1}+B_tu_t+n_t$
  - $n_t\sim N(0,Q_t)$
  - $x_t,n_t\in R^n$, $u_t\in R^m$, $A_t,Q_t\in R^{n\times n}$, $B_t\in R^{n\times m}$
- $p(z_t|x_t)$ linear, AGWN
  - $z_t = C_tx_t+v_t$
  - $v_t\sim N(0,R_t)$
  - $z_t,v_t\in R^p$, $C_t\in R^{p\times n}$, $R_t\in R^{p\times p}$

Equations

Prediction: uses input u_t and Q_t:

$\overline{\mu_t} = A\mu_{t-1}+Bu_t$

$\overline{\Sigma}_t = A\Sigma_{t-1}A^T+Q$

Where does this come?
- Sum of Gaussians: $z=x+y$ is also a Gaussian with $\mu_z=\mu_x+\mu_y$, $\Sigma_z=\Sigma_x+\Sigma_y$.
- Affine transformations: $X\sim N(\mu_X,\Sigma_X)$, $Y=AX+b$, then $Y\sim N(\mu_Y,\Sigma_Y)$, $\mu_Y=A\mu_X+b$, $\Sigma_Y=A\Sigma_XA^T$.

Update: uses measurement z_t and R_t:

$K_t = \overline{\Sigma}_t C^T(C\overline{\Sigma}_tC^T+R)^{-1}$

$\mu_t = \overline{\mu}_t + K_t(\underbrace{z_t - C\overline{\mu}_t}_{\text{Innovation}})$

$\Sigma_t = \overline{\Sigma}_t - K_tC\overline{\Sigma}_t$

Where does this come?
- $Y=[X\,Z]^T$ multivariate Gaussian, $\mu=[\mu_X\,\mu_Z]^T$, $\Sigma=[\Sigma_{XX}\,\Sigma_{XZ};\Sigma_{ZX}\,\Sigma_{ZZ}]$ $p(X|Z)=P(X,Z)/P(Z)$ has $\mu_{X|Z}=\mu_X+\Sigma_{XZ}\Sigma_{ZZ}^{-1}(X-\mu_Z)$ $\Sigma_{X|Z}=\Sigma_{XX}-\Sigma_{XZ}\Sigma_{ZZ}^{-1}\Sigma_{ZX}$
- The best update without a measurement is $x_t=\overline{x}_t$. Then $\begin{pmatrix}x_t\\z_t\end{pmatrix}=\begin{pmatrix}I&0\\C&I\end{pmatrix}\begin{pmatrix}\overline{x}_t\\v_t\end{pmatrix}$ With mean $[\overline{\mu}_t\ C\overline{\mu}_t]^T$ and $\Sigma=\begin{pmatrix}I&0\\C&I\end{pmatrix}\begin{pmatrix}\overline{\Sigma}_t&0\\0&R\end{pmatrix}\begin{pmatrix}I&C^T\\0&I\end{pmatrix}$ $= \begin{pmatrix}\overline{\Sigma}_t & \overline{\Sigma}_tC^T\\ C\overline{\Sigma}_t & C\overline{\Sigma}_tC^T+R\end{pmatrix}$

Kalman Gain

Degree to which the measurement is incorporated ("trusted")
- Perfect sensor: $R=0$ $K_t=C^{-1}$, $\mu_t=C^{-1}z_t$, $\Sigma_t=0$
- Horrible sensor: $R\to\infty$ $K_t\to0$ $\mu_t\to\overline{\mu}_t$, $\Sigma_t\to\overline{\Sigma}_t$

Kalman Facts
- If dist. not Gaussian, Kalman filter is the minimum variance linear estimator (noise must be uncorrelated with initial state x_0).
- **Variance never increases due to receiving a measurement.**
- **Variance update independent of the measurement realization.**
- The Kalman filter permits individual update steps for each sensor as data becomes available.

11 Extended Kalman Filter

- $p(x_0)\sim N(\mu_0,\Sigma_0)$
- $\dot{x}_t = f(x_t,u_t,n_t)$, $n_t\sim N(0,Q_t)$
- $z=h(x,v)$, $v_t\sim N(0,R_t)$

We use one-step Euler integration to discretize the system in the interval $\tau=[t',t)$.

Prediction Linearization

Linearize dynamics around

$x=\mu_{t-1}$, $u=u_t$, $n=0$

$f(x_t,u,n)\approx f(\mu_{t-1},u_t,0)+\underbrace{\frac{\partial f}{\partial x}\Big|_{\mu_{t-1},u_t,0}}_{A_t}(x-\mu_{t-1})+\underbrace{\frac{\partial f}{\partial u}\Big|_{\mu_{t-1},u_t,0}}_{B_t}(u-u_t)+\underbrace{\frac{\partial f}{\partial n}\Big|_{\mu_{t-1},u_t,0}}_{U_t}(n-0)$

One-step Euler integration

$x_t\approx x_{t-1}+f(x_{t-1},u_t,n_t)\delta t$

$x_t\approx x_{t-1}+(f(\mu_{t-1},u_t,0)+A_t(x_{t-1}-\mu_{t-1})+U_tn)\delta t$

$x_t\approx \underbrace{(I+A_t\delta t)}_{F_t}x_{t-1}+\underbrace{(U_t\delta_t)}_{V_t}n_t+\underbrace{(f(\mu_{t-1},u_t,0)-A_t\mu_{t-1})\delta t}_{b_t}$

- $\overline{\mu}_t = F_t\mu_{t-1}+b_t = \mu_{t-1}+\delta t f(\mu_{t-1},u_t,0)$
- $\overline{\Sigma}_t = F_t\Sigma_{t-1}F_t^T+V_tQ_tV_t^T$

Update Linearization

Linearize observation model around

$x=\overline{\mu}_t$, $v=0$

$h(x,v)\approx h(\overline{\mu}_t,0)+\underbrace{\frac{\partial h}{\partial x}\Big|_{\overline{\mu}_t,0}}_{C_t}(x-\overline{\mu}_t)+\underbrace{\frac{\partial h}{\partial v}\Big|_{\overline{\mu}_t,0}}_{W_t}(v-0)$

$z_t\approx h(\overline{\mu_t},0)+C_t(x_t-\overline{\mu}_t)+W_tv_t$

We define the matrix

$\begin{pmatrix}x_t\\z_t\end{pmatrix}=\begin{pmatrix}I&0\\C_t&W_t\end{pmatrix}\begin{pmatrix}\overline{x}_t\\z_t\end{pmatrix}+\begin{pmatrix}0\\h(\overline{\mu}_t,0)-C_t\overline{\mu}_t\end{pmatrix}$

$\Sigma=\begin{pmatrix}\overline{\Sigma}_t & \overline{\Sigma}_tC_t^T\\ C_t\overline{\Sigma}_t & C_t\overline{\Sigma}_tC_t^T+W_tR_tW_t^T\end{pmatrix}$

- $\mu_t=\overline{\mu}_t+K_T(z_t-h(\overline{\mu}_t,0))$
- $\Sigma_t=\overline{\Sigma}_t-K_tC_t\overline{\Sigma}_t$
- $K_t=\overline{\Sigma}_tC_t^T(C_t\overline{\Sigma}_tC_t^T+W_tR_tW_t^T)^{-1}$

Project Implementations

Why choose sensors as inputs instead of observations?
- Keeps state space and dimension of belief small.
- We might have very high confidence in the sensors (and very low confidence in our aerodynamical model).

Model

$\omega=(p\quad q\quad r)^T = \begin{pmatrix}c\theta&0&-c\phi s\theta\\0&1&s\phi\\s\theta&0&c\phi c\theta\end{pmatrix}(\dot\phi\quad \dot\theta\quad \dot\psi)=T(\mathbf{q})\dot{\mathbf{q}}$

First Implementation: gyro + VICON (linear vel)

$\mathbf{x}=(\mathbf{p}\quad \mathbf{q}\quad \mathbf{b}_g)^T$ ($\mathbf{b}_g$ bias gyro)

$\mathbf{u}=(\mathbf{v}_m\quad \omega_m)^T$

$\mathbf{v}_m=\dot{\mathbf{p}}_m+\mathbf{n}_v$

$\omega_m=\omega+\mathbf{b}_g+\mathbf{n}_g$

$\dot{\mathbf{b}}_g=\mathbf{n}_{bg}\sim N(0,Q_g)$ (bias gyro drift)

$\dot{\mathbf{x}}=f(\mathbf{x},\mathbf{u},\mathbf{n})=\begin{pmatrix}\mathbf{v}_m-\mathbf{n}_v\\T(\mathbf{q})^{-1}(\omega_m-\mathbf{b}_g-\mathbf{n}_g)\\\mathbf{n}_{bg}\end{pmatrix}$

$\mathbf{n}=(\mathbf{n}_v\quad \mathbf{n}_g\quad \mathbf{n}_{bg})$

$\mathbf{z}=\begin{pmatrix}I&0&0\\0&I&0\end{pmatrix}\mathbf{x}+\mathbf{v}$

Second implementation: gyro + accel

$\mathbf{x}=(\mathbf{p}\quad \mathbf{q}\quad \mathbf{v}\quad \mathbf{b}_g\quad \mathbf{b}_a)^T$

$\mathbf{u}=(\mathbf{a}_m\quad \omega_m)^T$

($\mathbf{b}_g$ bias gyro, $\mathbf{b}_a$ bias accel)

$\omega_m=\omega+\mathbf{b}_g+\mathbf{n}_g$

$\dot{\mathbf{b}}_g=\mathbf{n}_{bg}\sim N(0,Q_g)$ (bias gyro drift)

$\mathbf{a}_m = R(\mathbf{q})^T(\ddot{\mathbf{p}}-\mathbf{g})+\mathbf{b}_a+\mathbf{n}_a$

$\dot{\mathbf{b}}_a=\mathbf{n}_{ba}\sim N(0,Q_a)$ (bias accel drift)

$\dot{\mathbf{x}}=f(\mathbf{x},\mathbf{u},\mathbf{n})=\begin{pmatrix}\mathbf{v}\\T(\mathbf{1})^{-1}(\omega_m-\mathbf{b}_g-\mathbf{n}_g)\\\mathbf{g}+R(\mathbf{q})(\mathbf{a}_m-\mathbf{b}_a-\mathbf{n}_a)\\\mathbf{n}_{bg}\\\mathbf{n}_{ba}\end{pmatrix}$

$\mathbf{n}=(\mathbf{a}_v\quad \mathbf{n}_g\quad \mathbf{n}_{bg}\quad \mathbf{n}_{ba})$

$\mathbf{z}=\begin{pmatrix}I&0&0&0&0\\0&I&0&0&0\\0&0&I&0&0\end{pmatrix}\mathbf{x}+\mathbf{v}$

12 Good things to know

Trigonometric identities

Sum of angles

$\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$

$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sin\alpha\sin\beta$

Double angles

$\sin(2\alpha)=2\sin\alpha\cos\alpha$

$\cos(2\alpha)=\cos^2\alpha-\sin^2\alpha$

Jacobian

$\dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}=\begin{pmatrix}\frac{\partial f_1}{\partial x_1}&\cdots&\frac{\partial f_1}{\partial x_n}\\\vdots&&\vdots\\\frac{\partial f_m}{\partial x_1}&\cdots&\frac{\partial f_m}{\partial x_n}\end{pmatrix}$