Lab 1 – Kamerageometri med Eigen

26.01.2017

Del 1: Introduksjon til Eigen

Eigen 3

- C++ bibliotek for lineær algebra
  - http://eigen.tuxfamily.org/
- «Template bibliotek» – «Header only»
  - Flerplatform, Ingen linking!
- Godt dokumentert!
  - https://eigen.tuxfamily.org/dox/
  - https://eigen.tuxfamily.org/dox/AsciiQuickReference.txt
  - https://eigen.tuxfamily.org/dox/group__TutorialMatrixClass.html

Bli kjent med Eigen

- Lag deg noen vektorer og matriser
  - $\boldsymbol{t} = \begin{bmatrix} 1.0 \\ 0.0 \\ 3.0 \end{bmatrix}$
  - $A = \begin{bmatrix} 1.0 & 0.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{bmatrix}$
  - $I = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
  - $T = \begin{bmatrix} A & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}$
  - $B = A^T$

Bli kjent med Eigen

- Lek litt med koeffisientindeksering
  - $\boldsymbol{t} = \begin{bmatrix} 1.0 \\ 0.0 \\ 3.0 \end{bmatrix}$ ← Sett til 2.0
  - $A = \begin{bmatrix} 1.0 & 0.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{bmatrix}$

Bli kjent med Eigen

- Blokkoperasjoner
  - Lag en vektor fra en rad i $A = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{bmatrix}$
  - Lag en vektor fra en kolonne i $A = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{bmatrix}$
  - Lag en matrise fra den midterste 2x2 submatrisen i T
  - Hva skjer med A og T om dere endrer disse vektorene/matrisene?

Bli kjent med Eigen

- Matrise- og vektoraritmetikk
  - Legg to vektorer/matriser sammen
  - Multipliser to matriser sammen
  - Ta prikkproduktet av to vektorer
  - Ta kryssproduktet av to vektorer
  - Ta koeffisientvis multiplikasjon mellom to matriser

Bli kjent med Eigen

- Reduksjonsoperasjoner
  - Ta summen av alle koeffisientene i en matrise
  - Beregn minimumsverdien i en matrise
    - Finn posisjonen til denne koeffisienten
  - Lag en vektor som gir minimumsverdien i hver kolonne i en matrise
  - Finn L1- og L2-normen til en vektor
  - Finn antall koeffisienter i en matrise som er større enn en gitt verdi

Bli kjent med Eigen

- Mer avansert bruk
  - Ta en titt på Map
    - https://eigen.tuxfamily.org/dox/group__TutorialMapClass.html
    - Lag en std::vector med tall, og bruk dette minnet som en Eigen matrise
  - Ta en titt på select()
    - https://eigen.tuxfamily.org/dox/classEigen_1_1DenseBase.html#aaec49d21e1bf32797180e5d21a489f98
  - Lek litt med lineær algebra
    - https://eigen.tuxfamily.org/dox/group__DenseLinearSolvers__chapter.html

Del 2 – Kamerageometri

Pose

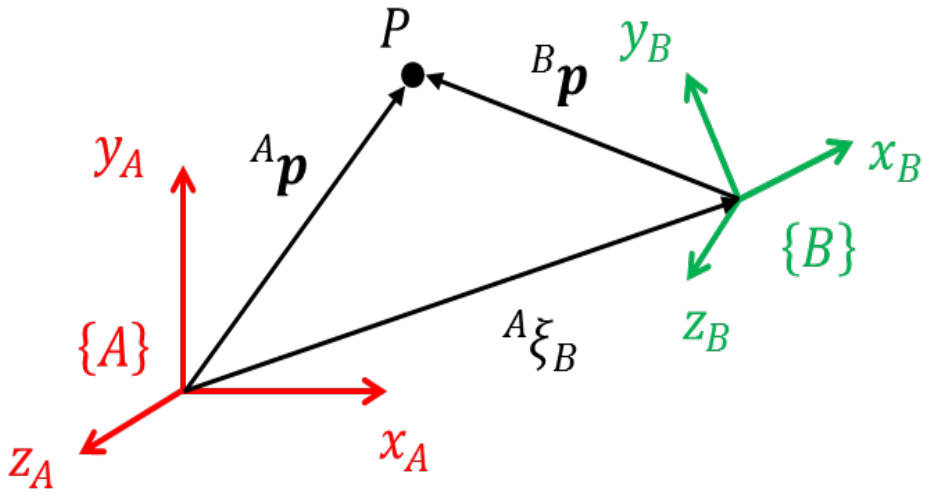
- Posen til $\{B\}$ relativt til $\{A\}$ kan representeres med en projektiv transformatsjon ${}^{A}T_{B} \in SE(3)$

$$ {}^{A}\xi_{B} \quad \mapsto \quad {}^{A}T_{B} = \begin{bmatrix} {}^{A}R_{B} & {}^{A}\boldsymbol{t}_{B} \\ \boldsymbol{0} & 1 \end{bmatrix} $$

- Egenskaper

$$ \begin{aligned} {}^{A}\boldsymbol{p} = {}^{A}\xi_{B} \cdot {}^{B}\boldsymbol{p} \quad &\mapsto \quad {}^{A}\tilde{\boldsymbol{p}} = {}^{A}T_{B}\, {}^{B}\tilde{\boldsymbol{p}} \\ {}^{A}\xi_{C} = {}^{A}\xi_{B} \oplus {}^{B}\xi_{C} \quad &\mapsto \quad {}^{A}T_{C} = {}^{A}T_{B}\, {}^{B}T_{C} \\ \ominus {}^{A}\xi_{B} \quad &\mapsto \quad {}^{A}T_{B}^{-1} \end{aligned} $$

Pose



- En av egenskapene til pose er å transformere punkter (husk homogen representasjon)

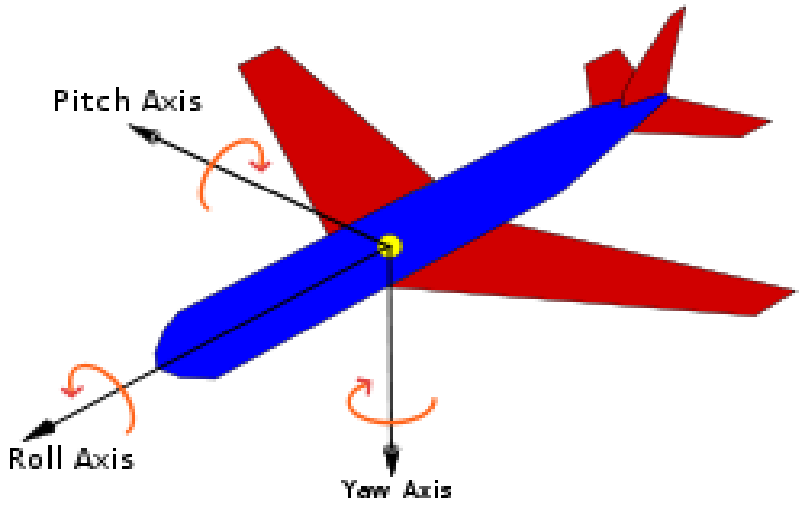
$$ {}^{A}\tilde{\boldsymbol{p}} = {}^{A}T_{B}\,{}^{B}\tilde{\boldsymbol{p}} $$

$$ \begin{bmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}R_{B} & {}^{A}\boldsymbol{t}_{B} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{bmatrix} $$

Euler vinklene yaw, pitch, roll

- Euler vinkler brukes gjerne til å beskrive hvordan et referansesystem er orientert relativt til et annet

- Mange alternativer f.eks yaw-pitch-roll



From http://en.wikipedia.org/wiki/Euler_angles

Euler vinklene yaw, pitch, roll

- Yaw = Rotasjon mot klokka en vinkel α om z-aksen
- Pitch = Rotasjon mot klokka en vinkel β om y-aksen
- Roll = Rotasjon mot klokka en vinkel γ om z-aksen

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

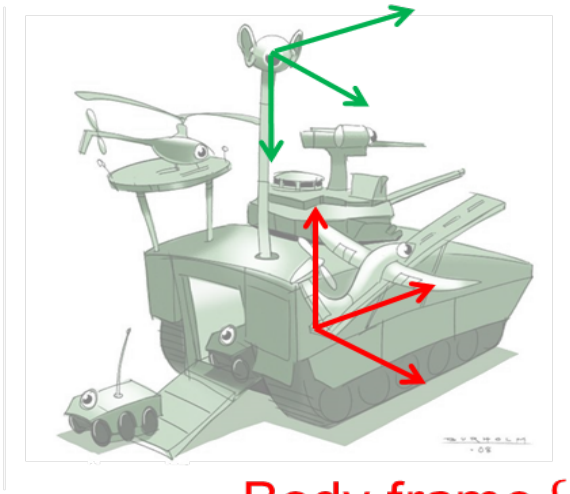
- Orienteringen til body-frame $\{B\}$ relativt til en referanse-frame $\{W\}$

$$\begin{aligned} {}^{W}R_B(\alpha,\beta,\gamma) &= R_z(\alpha)R_y(\beta)R_x(\gamma) \\ &= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \end{aligned}$$

Problem

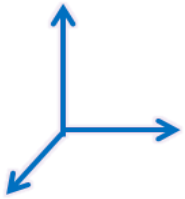
P

Camera frame $\{C\}$



Body frame $\{B\}$

Hvor i bildet observerer kameraet punktet P?



World frame $\{W\}$

Det vi vet om $\{B\}$

- Kjøretøyet er 6m langt, 3m bredt og 2m høyt.
- Body frame $\{B\}$ er plassert i sentrum av kjøretøyet med
  - Body x-aksen peker fremover
  - Body y-aksen peker til venstr
  - Body z-aksen peker oppover
- Pose til $\{B\}$ relative til $\{W\}$ er gitt ved
  - $x = 15.0m$
  - $y = 10.0m$
  - $z = 1.5m$
  - $yaw = 210.0°$
  - $roll = 0.0°$
  - $pitch = 0.0°$

$$\Big\} = {}^{W}\xi_{B}$$

Det vi vet om $\{C\}$

- Kameraet står fast på kjøretøyet
- Kamera framen $\{C\}$ er som vanlig
  - Kamera x-aksen peker til høyre
  - Kamera y-aksen peker nedover
  - Kamera z-aksen peker fremover
- Pose til $\{C\}$ relativt til $\{B\}$ er gitt ved
  - $x = \text{Half of vehicle length}$
  - $y = 0.0$
  - $z = \text{Half of vehicle height}$
  - $yaw = -90.0°$
  - $pitch = 0.0°$
  - $roll = -90.0°$

$$\Big\} = {}^{B}\xi_{C}$$

- The camera captures images with
  - $rows = 120$
  - $cols = 160$
- The camera intrinsics are known
  - $f_u = 80$
  - $f_v = f_u$
  - $s = 0$
  - $c_u = cols/2$
  - $c_v = rows/2$

Hvordan løse problemet

- Bruk perspektiv kamera modellen til å projisere punktene inn i bildet

$$\tilde{\boldsymbol{u}} = P\,{}^{W}\tilde{\boldsymbol{X}}$$
$$\tilde{\boldsymbol{u}} = K[R \quad \boldsymbol{t}]\,{}^{W}\tilde{\boldsymbol{X}}$$
$$\tilde{\boldsymbol{u}} = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}[R \quad \boldsymbol{t}]\,{}^{W}\tilde{\boldsymbol{X}}$$

- Husk at

$$\begin{bmatrix} R & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} = {}^{C}T_{W} = {}^{W}T_{C}{}^{-1}$$

- Bestem ${}^{W}T_{C}$ ut fra de angitte posene

$${}^{W}T_{C} = {}^{W}T_{B}\,{}^{B}T_{C}$$

- Bestem ${}^{W}T_{B}$ ut fra beskrivelsen av pose til til $\{B\}$ relative til $\{W\}$
- Bestem ${}^{B}T_{C}$ ut fra beskrivelsen av pose til til $\{C\}$ relative til $\{B\}$
- Beregn kameramatrisen $\mathrm{P} = K[R \quad \boldsymbol{t}]$ og bruk den til å projisere verdenspunkter inn i bildet

Hvordan løse problemet

- Tips
  - Rotasjonsmatrisen $R_x(\gamma)$ kan genereres ved
    ```
    Eigen::AngleAxisd( γ * M_PI / 180, Eigen::Vector3d::UnitX())
    ```
  - Rotasjonsmatrisen $R_y(\beta)$ kan genereres ved
    ```
    Eigen::AngleAxisd( β * M_PI / 180, Eigen::Vector3d::UnitY())
    ```
  - Rotasjonsmatrisen $R_z(\alpha)$ kan genereres ved
    ```
    Eigen::AngleAxisd( α * M_PI / 180, Eigen::Vector3d::UnitZ())
    ```

- Represent the pose of $\{C\}$ relative to $\{B\}$ by a Euclidean projective transformation ${}^{B}T_{C}$
- Compute the pose of $\{C\}$ relative to $\{W\}$ as ${}^{W}T_{C}$
- Compute the camera matrix $\mathrm{P} = K[R \quad \boldsymbol{t}]$ and use it to project points into the image