Autonomous Navigation for Flying Robots

Lecture 3.1:
3D Geometry

Jürgen Sturm
Technische Universität München
Points in 3D

- 3D point
  \[ \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \]

- Augmented vector
  \[ \tilde{\mathbf{x}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4 \]

- Homogeneous coordinates
  \[ \check{\mathbf{x}} = \begin{pmatrix} \check{x} \\ \check{y} \\ \check{z} \\ \check{w} \end{pmatrix} \in \mathbb{P}^3 \]
Geometric Primitives in 3D

- 3D line $\mathbf{r} = (1 - \lambda)\mathbf{p} + \lambda\mathbf{q}$ through points $\mathbf{p}, \mathbf{q}$
- Infinite line: $\lambda \in \mathbb{R}$
- Line segment joining $\mathbf{p}, \mathbf{q}$: $0 \leq \lambda \leq 1$
Geometric Primitives in 3D

- 3D plane
- 3D plane equation
- Normalized plane

with unit normal vector $\|\hat{\mathbf{n}}\| = 1$ and distance $d$

$\tilde{\mathbf{m}} = (a, b, c, d)^\top$

$\tilde{x} \cdot \tilde{\mathbf{m}} = ax + by + cz + d = 0$

$\mathbf{m} = (\hat{n}_x, \hat{n}_y, \hat{n}_z, d)^\top = (\hat{n}, d)$
3D Transformations

- Translation
  \[
  \tilde{x}' = \begin{pmatrix}
  I & t \\
  0^\top & 1
  \end{pmatrix}
  \tilde{x}
  \]

- Euclidean transform (translation + rotation), (also called the Special Euclidean group SE(3))
  \[
  \tilde{x}' = \begin{pmatrix}
  R & t \\
  0^\top & 1
  \end{pmatrix}
  \tilde{x}
  \]

- Scaled rotation, affine transform, projective transform…
3D Euclidean Transformations

- Translation $\mathbf{t} \in \mathbb{R}^3$ has 3 degrees of freedom
- Rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ has 3 degrees of freedom

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
3D Rotations

- A rotation matrix is a 3x3 orthogonal matrix

\[
R = \begin{pmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33}
\end{pmatrix}
\]

- Also called the special orientation group SO(3)
- Column vectors correspond to coordinate axes
3D Rotations

- What operations do we typically do with rotation matrices?
  - Invert, concatenate
  - Estimate/optimize
- How easy are these operations on matrices?

\[
R = \begin{pmatrix}
r_{11} & r_{12} & r_{13} \\
r_{21} & r_{22} & r_{23} \\
r_{31} & r_{32} & r_{33}
\end{pmatrix}
\]
3D Rotations

- Advantage:
  Can be easily concatenated and inverted (how?)

- Disadvantage:
  Over-parameterized (9 parameters instead of 3)

\[
R = \begin{pmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33}
\end{pmatrix}
\]
Euler Angles

- Product of 3 consecutive rotations (e.g., around X-Y-Z axes)
- Roll-pitch-yaw convention is very common in aerial navigation (DIN 9300)

Roll-Pitch-Yaw Convention

- Roll $\phi$, Pitch $\theta$, Yaw $\psi$
- Conversion to 3x3 rotation matrix:

$$R = R_Z(\psi)R_Y(\theta)R_X(\phi)$$

$$= \begin{pmatrix}
\cos \psi & -\sin \psi & 0 \\
\sin \psi & \cos \psi & 0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi & \cos \phi
\end{pmatrix}$$

$$= \begin{pmatrix}
\cos \psi \cos \theta & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi & \cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi \\
\sin \psi \cos \theta & \sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi \\
-\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi
\end{pmatrix}$$
Roll-Pitch-Yaw Convention

- Roll $\phi$, Pitch $\theta$, Yaw $\psi$
- Conversion from 3x3 rotation matrix:

$$
\phi = \text{Atan2}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2})
$$

$$
\psi = -\text{Atan2}\left(\frac{r_{21}}{\cos(\phi)}, \frac{r_{11}}{\cos(\phi)}\right)
$$

$$
\theta = \text{Atan2}\left(\frac{r_{32}}{\cos(\phi)}, \frac{r_{33}}{\cos(\phi)}\right)
$$
Euler Angles

- Advantage:
  - Minimal representation (3 parameters)
  - Easy interpretation

- Disadvantages:
  - Many “alternative” Euler representations exist (XYZ, ZXZ, ZYX, …)
  - Difficult to concatenate
  - Singularities (gimbal lock)
Gimbal Lock

- When the axes align, one degree-of-freedom (DOF) is lost:

http://commons.wikimedia.org/wiki/File:Rotating_gimbal-xyz.gif
Axis/Angle

- Represent rotation by
  - rotation axis \( \mathbf{n} \) and
  - rotation angle \( \theta \)
- 4 parameters
- 3 parameters
  - length is rotation angle
  - also called the angular velocity
  - minimal but not unique (why?)
Conversion

- Rodriguez’ formula

\[ R(\hat{n}, \theta) = I + \sin \theta [\hat{n}]_\times + (1 - \cos \theta) [\hat{n}]_\times^2 \]

- Inverse

\[ \theta = \cos^{-1} \left( \frac{\text{trace}(R) - 1}{2} \right), \quad \hat{n} = \frac{1}{2 \sin \theta} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \]

see: An Invitation to 3D Vision (Ma, Soatto, Kosecka, Sastry), Chapter 2
Axis/Angle

- Also called twist coordinates
- Advantages:
  - Minimal representation
  - Simple derivations
- Disadvantage:
  - Difficult to concatenate
  - Slow conversion
Quaternions

- Quaternion \( q = (q_w, q_x, q_y, q_z)^\top \in \mathbb{R}^4 \)
- Real and vector part
  \( q = (r, v), \ r \in \mathbb{R}, \ v \in \mathbb{R}^3 \)
- Unit quaternions have \( \|q\| = 1 \)
- Opposite sign quaternions represent the same rotation
- Otherwise unique
Quaternions

- Advantage: multiplication, inversion and rotations are very efficient
- Concatenation
  \[(r_1, v_1)(r_2, v_2) = (r_1 r_2 - v_1 \cdot v_2, r_1 v_2 + r_2 v_1 + v_1 \times v_2)\]
- Inverse (=flip signs of real or imaginary part)
  \[(r, v)^{-1} = (r, v)^* \equiv (-r, v) \equiv (r, -v)\]
- Rotate 3D vector \(p \in \mathbb{R}^3\) using a quaternion:
  \[(r, v)(0, p)(r, v)^*\]
Quaternions

- Rotate 3D vector $\mathbf{p} \in \mathbb{R}^3$ using a quaternion:
  
  $$(r, \mathbf{v})(0, \mathbf{p})(r, \mathbf{v})^*$$

- Relation to axis/angle representation

  $$q = (r, \mathbf{v}) = (\cos \frac{\theta}{2}, \sin \frac{\theta}{2} \hat{n})$$
3D Orientations

- **Note:** In general, it is very hard to “read” 3D orientations/rotations, no matter in what representation.

- **Observation:** They are usually easy to visualize and can then be intuitively interpreted.

- **Advice:** Use 3D visualization tools for debugging (RVIZ, libqglviewer, ...)

Jürgen Sturm

Autonomous Navigation for Flying Robots
3D to 2D Perspective Projections

Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/

Jürgen Sturm
Autonomous Navigation for Flying Robots
3D to 2D Perspective Projections

$x \in \mathbb{R}^2$

$p \in \mathbb{R}^3$

Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/
3D to 2D Perspective Projections

- Pin-hole camera model

\[
\tilde{x} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{p}
\]

- Note: $\tilde{x}$ is homogeneous, needs to be normalized

\[
\tilde{x} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad \Rightarrow \quad x = \begin{pmatrix} \tilde{x}/\tilde{z} \\ \tilde{y}/\tilde{z} \end{pmatrix}
\]
Camera Intrinsics

- So far, 2D point is given in meters on image plane
- But: we want 2D point be measured in pixels (as the sensor does)

Richard Szeliski, Computer Vision: Algorithms and Applications
http://szeliski.org/Book/
Camera Intrinsic

- Need to apply some scaling/offset

\[ \tilde{x} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{p} \]

\[ \text{intrinsic } K \text{ projection} \]

- Focal length $f_x, f_y$
- Camera center $c_x, c_y$
- Skew $s$
Camera Extrinsics

- Assume \( \tilde{p}_w \) is given in world coordinates
- Transform from world to camera (also called the camera extrinsics)

\[
\tilde{p} = \begin{pmatrix} R & t \\ 0^\top & 1 \end{pmatrix} \tilde{p}_w
\]

- Projection of 3D world points to 2D pixel coordinates:

\[
\tilde{x} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & t \end{pmatrix} \tilde{p}_w
\]
Lessons Learned

- 3D points, lines, planes
- 3D transformations
- Different representations for 3D orientations
  - Choice depends on application
  - Which representations do you remember?
- 3D to 2D perspective projections
- You really have to know 2D/3D transformations by heart (for more info, read Szeliski, Chapter 2, available online)