Mathematical Principles in Vision and Graphics:
Rigid Transformations

Ass.Prof. Friedrich Fraundorfer

SS 2019
Learning goals

- Understand the problems of dealing with rotations
- Understand how to represent rotations
- Understand the terms SO(3) etc.
- Understand the use of the tangent space
- Understand Euler angles, Axis-Angle, and quaternions
- Understand how to interpolate, filter and optimize rotations
Outline

- Rigid transformations
- Problems with rotation matrices
- Properties of rotation matrices
- Matrix groups $SO(3)$, $SE(3)$
- Manifolds
- Tangent space
- Skew-symmetric matrices
- Exponential map
- Euler angles, angle-axis, quaternions
- Interpolation
- Filtering
- Optimization
Motivation: 3D Viewer
Rigid transformations

- Coordinates are related by:
  \[ X_c = RX_w + T \]
  \[ \begin{bmatrix} X_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ 1 \end{bmatrix} \]

- Rigid transformation belong to the matrix group SE(3)
- What does this mean?
Properties of rotation matrices

Rotation matrix:

\[ R = [r_1, r_2, r_3] \in \mathbb{R}^{3 \times 3} \]

\[ R^T R = I, \det(R) = +1 \]

Coordinates are related by: \[ X_c = RX_w \]

- Rotation matrices belong to the matrix group \( \text{SO}(3) \)
- What does this mean?
Problems with rotation matrices

- Optimization of rotations (bundle adjustment)
  - Newton’s method $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

- Linear interpolation

- Filtering and averaging
  - E.g. averaging rotation from IMU or camera pose tracker for AR/VR glasses
Matrix groups

- The set of all the n×n invertible matrices is a group w.r.t. the matrix multiplication:

\[ GL(n) = (\{M \in \mathbb{R}^{n \times n} | \det(M) \neq 0\}, \times) \]

General linear group

- Reminder: A group is an algebraic structure consisting of a set of elements equipped with an operation that combines any two elements to form a third element.
- The operation satisfies four conditions called the group axioms, namely closure, associativity, identity and invertibility.
- Closure means for all \( a, b \) in \( G \), the result of the operation, \( a \cdot b \), is also in \( G \).
Matrix groups

- The set of all the nxn orthogonal matrices is a group w.r.t. the matrix multiplication:

\[ O(n) = \left(\{A \in GL(n) | A^{-1} = A^T\}, \times\right) \]

Orthogonal group

\[ A \in O(n) \Rightarrow \det(A) = \pm 1 \]
Matrix groups

- The set of all the nxn orthogonal matrices with determinant equal to +1 is a group w.r.t. the matrix multiplication:

\[ SO(n) = (\{A \in O(n) | \det(A) = +1 \}, \times) \]

Special orthogonal group

- SO(3) … group of orthogonal 3x3 matrices with \( \det=+1 \) …. “rotation matrices”

- \( R_3 = R_1 \times R_2 \) … R3 is still an SO(3) element

- \( R_3 = R_1 + R_2 \) … R3 is NOT an SO(3) element. Not a rotation matrix anymore.
Matrix groups

- The set of all the rigid transformations in $\mathbb{R}^n$ is a group (not commutative) with the composition operation

$$\left( \left\{ F : \mathbb{R}^n \rightarrow \mathbb{R}^n \mid \text{F rigid} \right\}, \circ \right)$$

- The set is isomorphic to the special Euclidean group $\text{SE}(n)$

- The mathematical properties of a “rigid transformation” are specified by the special Euclidean group $\text{SE}(n)$

$$RT = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$
Matrix groups: Summary

Vector space of all the $n \times n$ matrices

$GL(n) = \{ M \in \mathbb{R}^{n \times n} | \det(M) \neq 0 \}, \times$

General linear group

$O(n) = \{ A \in GL(n) | A^{-1} = A^T \}, \times$

Orthogonal group

$SO(n) = \{ A \in O(n) | \det(A) = +1 \}, \times$

Special orthogonal group

$O(n)/SO(n) = \{ A \in O(n) | \det(A) = -1 \}$

Set of orthogonal matrices which do not preserve orientation (not a group)

GL(n), O(n), SO(n) and SE(n) are all smooth manifolds (e.g. surfaces, curves, solids immersed in some big vector space)
The special Euclidean group is constructed by the Cartesian product (a composition operation) from SO(n)xR^n.

\[ SE(n) = (SO(n) \times \mathbb{R}^n, \times) \]

\[ (M, t) \times (S, q) = (MS, Mq + t) \]

The Cartesian product defines where the values of SO(n) and R^n (rotation and translation) go to form the transformation matrix

\[ RT = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \]

\[ SE(3) = \begin{bmatrix} SO(3) & \mathbb{R}^3 \\ 0 & 1 \end{bmatrix} \]
Manifolds

- Non-mathematical definition: Manifolds are shapes like curves, surfaces, and volumes in Euclidean space (i.e. a vector space).
- A sphere is an example of a manifold of dimension 2.
- The matrix groups SO(3), SE(3) are manifolds.
Shapes of SO(2) and SO(3)

SO(2) … 1-manifold

SO(3) … 3-manifold (3-sphere)
A solid ball in $\mathbb{R}^3$
The tangent space of the manifold $M$ in $p$ (every point $p$ on the manifold has a different tangent space) is isomorphic to a subspace of $V$. 
Tangent space of a manifold

- $\text{TM}(p)$ is a vector space (subspace of $V$) and has dimension $k$. 

- 1-manifold (curves) $\Rightarrow$ 1 dim TM (lines)
- 2-manifold (surface) $\Rightarrow$ 2 dim TM (planes)
- 3-manifold (volumes, e.g. 3-sphere) $\Rightarrow$ 3 dim TM (full volumes)
Tangent space of SO(2) and SO(3)

**TSO(2)** is a vector space with dimension 1
subspace of $\mathbb{R}^{2 \times 2}$
subspaces are defined by matrices

**TSO(3)** is a vector space with dimension 3
subspace of $\mathbb{R}^{3 \times 3}$
Skew-symmetric matrix

- M is skew-symmetric iff $M^T = -M$

$$M^T = -M \begin{bmatrix} 0 & 3 & 6 \\ -3 & 0 & -1 \\ -6 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 4 \\ -4 & 0 \end{bmatrix}$$

$$so(n) = \{M \in \mathbb{R}^{n \times n} | M^T = -M \}, +, []$$

Special orthogonal Lie algebra

- The term “algebra” means that so(n) is a vector space (more specific than group)
- We have addition in the vector space now (in SO(n) it was only multiplication)
Skew-symmetric matrices

- The special orthogonal Lie algebra is the tangent space of SO(n) at identity.
- The tangent space of SO(n) in any other point R is a rotated version of so(n)
- $T_{SO(n)}(R) = R \times so(n)$ is not a skew-symmetric matrix anymore, but a rotated one
so(2) and so(3)

- so(3) is a vector space of dimension 3
- so(2) is a vector space of dimension 1

\[
\begin{bmatrix}
0 & 3 & 6 \\
-3 & 0 & -1 \\
-6 & 1 & 0
\end{bmatrix}
\quad \begin{bmatrix}
0 & 4 \\
-4 & 0
\end{bmatrix}
\]
The hat operator

- The hat operator is used to form skew-symmetric matrices
- for so(3):
  \[ \hat{\mathbf{v}} : \mathbb{R}^3 \rightarrow so(3) \]
  \[ (x, y, z) \rightarrow \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \]
- for so(2):
  \[ \hat{\mathbf{x}} : \mathbb{R} \rightarrow so(2) \]
  \[ \hat{x} \rightarrow \begin{bmatrix} 0 & -x \\ x & 0 \end{bmatrix} \]
- Different notation

\[ [t]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \]
The hat operator

- The hat operator is used to define the cross-product in matrix form

\[
a \times b = \hat{a}b \quad \forall a, b \in \mathbb{R}^3
\]
Lie groups

- GL(n), O(n), SO(n) and SE(n) are all Lie groups (groups which are a smooth manifold where the operation is a differentiable function between manifolds)
- Also we have seen that the group of skew-symmetric matrices is called Lie algebra so(n) and is the tangent space of the special orthonormal group SO(n)
- But how to compute an element of the tangent space so(n) from SO(n) or vice versa?
- The exponential map!
Exponential map

- Given a Lie group $G$, with its related Lie algebra $g=T_G(I)$, there always exists a smooth map from Lie algebra $g$ to the Lie group $G$ called exponential map

\[ \exp : g \rightarrow G \]
Angle-axis representation for rotations:

- \( \omega \in \mathbb{R}^k \) is the angle-axis representation (\( \mathbb{R}^3 \))
- \( \exp(\omega) \) is the 3x3 rotation matrix (element of \( \text{SO}(3) \))
Euler angles

- Euler’s Theorem for rotations: Any element in SO(3) can be described as a sequence of three rotations around the canonical axes, where no successive rotations are about the same axis.

\[
R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}
\]

\[
R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}
\]

\[
R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

For any \( R \in SO(3) \) there \( \exists \alpha, \beta, \gamma \mid R = R_x(\alpha)R_y(\beta)R_z(\gamma) \)

- \( \alpha, \beta, \gamma \) are called Euler angles of \( R \) according to the XYZ representation (3 DOF/parameters)
Euler angles

- Given $M$ (element of $SO(3)$) there are 12 possible ways to represent it:

  $M \in SO(3)$ there $\exists \alpha, \beta, \gamma \mid M = R_x(\alpha)R_y(\beta)R_z(\gamma)$

  $M \in SO(3)$ there $\exists \alpha, \beta, \gamma \mid M = R_x(\alpha)R_z(\gamma)R_y(\beta)$

  $M \in SO(3)$ there $\exists \alpha, \beta, \gamma \mid M = R_x(\alpha)R_z(\gamma)R_x(\beta)$

  $M \in SO(3)$ there $\exists \alpha, \beta, \gamma \mid M = R_z(\alpha)R_x(\gamma)R_z(\beta)$

  ...

- A common convention is ZYX corresponding to a rotation first around the x-axis (roll), then the y-axis (pitch) and finally around the z-axis (yaw)

[https://en.wikipedia.org/wiki/Aircraft_principal_axes, CC BY-SA 3.0]
Euler angles

- The parameterization has singularities, called gimbal lock
- A gimbal lock happens when after a rotation around an axis, two axes align, resulting in a loss of one degree of freedom

![Gimbal Lock Diagram]

cannot move (gimbal lock)
Angle-Axis

- Euler’s rotation theorem also states that any rotation can be expressed as a single rotation about some axis.

- The axis can be represented as a three-dimensional unit vector, and the angle by a scalar.

- 3 DOF/parameters

- Angle-axis defines a unique mapping and does not have gimbal lock
Angle-Axis

- The operation to compute the rotation matrix $SO(3)$ from the angle-axis parameters is by using the exponential map!

$$\omega \in \mathbb{R}^k \rightarrow \exp(\hat{\omega})$$

- The exponential map can be computed in closed form using the Rodrigues formula

$$R = I + (\sin \Theta)K + (1 - \cos \Theta)K^2$$

$$K = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

- There also exists the inverse
Quaternions are extensions of complex numbers, with 3 imaginary values instead of 1:

\[ a + ib + jc + kd \]

Like the imaginary component of complex numbers, squaring the components gives:

\[ i^2 = j^2 = k^2 = -1 \]

One way to express a quaternion is as a pair consisting of the real value and the 3D vector consisting of the imaginary components:

\[ q = (a, w) \text{ with } w = (b, c, d) \]

It is basically a 4-vector
Quaternions

- If \( q = a + ib + jc + kd \) is a unit quaternion (\( ||q||=1 \)), then \( q \) corresponds to a rotation:

\[
R(q) = \begin{bmatrix}
1 - 2c^2 - 2d^2 & 2bc - 2ad & 2bd + 2ac \\
2bc + 2ad & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\
2bd - 2ac & 2cd + 2ab & 1 - 2b^2 - 2c^2 \\
\end{bmatrix}
\]

- Because \( q \) is a unit quaternion, we can write \( q \) as:

\[
q = \left( \cos \left( \frac{a}{2} \right), \sin \left( \frac{a}{2} \right) w \right), \quad ||w|| = 1
\]

- It turns out the \( q \) corresponds to the rotation whose:
  - Axis of rotation is \( w \) and
  - Angle of the rotation is \( a \)
Interpolation in SO(3)

- Given two rotation matrices $R_1, R_2$ one would like to find a smooth path in $SO(3)$ connecting these two matrices

\[ R(\lambda) \in SO(3), \lambda \in [0,1] \]

$R(\lambda)$ smooth

$R(0) = R_1$

$R(1) = R_2$
Interpolation in SO(3)

- Approach 1: Linearly interpolate $R_1$ and $R_2$ as matrices (naive approach)

\[
R(\lambda) = \pi_{SO(3)}(\lambda R_1 + (1 - \lambda)R_2)
\]

Projection onto sphere (not accurate)

\[
\pi_{SO(3)}(M) = \arg\min_{R \in SO(3)} \|M - R\|_F^2
\]

Not an element of SO(3), not a rotation matrix at all
Interpolation in SO(3)

- Approach 2: Linearily interpolate $R_1$ and $R_2$ using Euler angles

- Each axis is interpolated independently
- If $R_1$ and $R_2$ are too far apart -> not intuitive motion
Interpolation in SO(3)

- Approach 3: Linearily interpolate $R_1$ and $R_2$ using angle-axis

$$\omega(\lambda) = \lambda \omega_1 + (1 - \lambda) \omega_2$$

- Interpolation happens in tangent space (vector space) and is then projected using the exponential map onto the manifold
- A constant speed in the tangent space leads to an increasing speed in the manifold
Interpolation in SO(3)

- Approach 4: Linearily interpolate $R_1$ and $R_2$ using the geodesic distance

- Interpolation happens on manifold, geodesic can be computed using the exponential map
- Method is called SLERP (spherical linear interpolation)
Filtering in SO(3)

- Given $n$ different noisy measurements for the rotation of an object

$$R_1, ..., R_n$$

- What is the filtered average of it?
Filtering in SO(3)

- Possible approaches:

  - Average the rotation matrices $R_i$
    \[ \frac{1}{n} \sum_{i=1}^{n} R_i \] (not rotation)

  - Average the Euler angles of each $R_i$
    \[ \left( \frac{1}{n} \sum_{i=1}^{n} \alpha_i, \frac{1}{n} \sum_{i=1}^{n} \beta_i, \frac{1}{n} \sum_{i=1}^{n} \gamma_i \right) \] (is rotation)

  - Average the angle-axis of each $R_i$
    \[ \frac{1}{n} \sum_{i=1}^{n} \omega_i \]

  - Average the quaternions of each $R_i$
    \[ \frac{1}{n} \sum_{i=1}^{n} q_i \]

- All equally problematic and do not accurately respect the noise model
Optimization in SO(3)

- Newton-Method

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \]

- Naive approach:
  - \( X_n \) are elements of rotation matrix. Then the update step (addition) would not result in a rotation matrix.

- \( X_n \) are Euler angles:
  - To evaluate \( f(X_n) \) the rotation matrix has to be created from the Euler angles. Could lead to gimbal lock.
  - Derivatives of Euler angle construction has to be computed.

- \( X_n \) are elements of the tangent space so(3)
  - Represents angle-axis notation
  - No gimbal lock
  - Minimal representation of 3 parameters
Learning goals - Recap

- Understand the problems of dealing with rotations
- Understand how to represent rotations
- Understand the terms SO(3) etc.
- Understand the use of the tangent space
- Understand Euler angles, Axis-Angle, and quaternions
- Understand how to interpolate, filter and optimize rotations