Representing Orientation

- Representing Translations and Positions
  - to translate by 30 units in \( x \):
    - add together thirty 1 unit translations
  - arithmetic interpolation
    - divide the total translation by \( n \)

- Representing Rotations and Orientations
  - to rotate by 30 degrees:
    - \( R = R^T \)
      - where \( R \) is a 3x3 or 4x4 matrix that rotates by one degree
  - geometric interpolation
    - take the \( n \)th root of the desired final rotation matrix

Rotation in a 2D world

- SO(3) group in Lie algebra
  - four common alternative numerical representations:
    - 3x3 rotation matrix
    - Euler angles (fixed angles)
    - exponential map
    - unit quaternions

Rotation in a 3D world

- 3x3 Rotation Matrix
  - 9 elements
  - 3 orthogonality constraints
  - renormalization algorithms
  - extracting pure rotational component (polar decopm)
    
    \[
    \begin{bmatrix}
    m_1 & m_2 & m_3 \\
    m_2 & m_3 & m_1 \\
    m_3 & m_1 & m_2
    \end{bmatrix}
    \]
  
  - where
    
    \[ a \cdot b = 0 \]
    
    \[ b \times c = 0 \]
  
  - \( R = \begin{bmatrix} a & b & c \end{bmatrix} \)
  
  - and determinant = 1

3x3 Rotation Matrix

- Euler's Rotation Theorem
  - can always go from one orientation to another
  - with one rotation about a single axis

- Exponential Map
  - idea: encode amount of rotation into magnitude of \( k \)
    
    \[ k = \theta \]
    
    \[ Rot(k, \theta) \rightarrow SO(3) \]
  
  - axis definition undefined for no rotation
  - therefore define the zero vector to be the identity rotation
  - singularities for \( \theta = 2\pi \)

- Unit Quaternions
  - \( \begin{bmatrix} w \ + \ x \ + \ y \ + \ z \end{bmatrix} \)
    
    \[ \theta = \frac{\sqrt{2}}{2} \]
    
    \[ q = (\cos \frac{\theta}{2}, \sin \frac{\theta}{2}) \]
    
    \[ q = (x, y, z) \]
  
  - rotation of a vector, i.e., a point in a coord frame:
    
    \[ \vec{v}' = Rot(k, \theta) \vec{v} = q \vec{v} \cdot \vec{q} \]
    
    \[ \vec{v} = (0, 0, 1) \]
    
    \[ \vec{q} = (x, y, z) \]
  
  - two successive rotations
    
    \[ q_1(q_2, \vec{v}) \cdot \vec{q}_2 \]

Orientation Interpolation

- Linear interpolation of quaternions
  - note: \( q \) and \(-q\) represent the same orientation
    
    \[ q_1 \rightarrow q_2 \text{ or } q_1 \rightarrow -q_1 \]
  
  - choose shorter path, use dot product to compute
    
    \[ \cos \theta = q_1 \cdot q_2 = x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2 \]

- Orientation Interpolation
  - SLERP instead of LERP
    
    \[ slerp(q_1, q_2, \alpha) = \frac{\sin((1-\alpha)\theta)}{\sin\theta} q_1 + \frac{\sin(\alpha\theta)}{\sin\theta} q_2 \]

  - smooth interpolation of multiple orientations:
    - construct smooth curve on the 4D sphere

Other Rotation

- how many degrees of freedom in 3D ?
  
  - desired features of any representation
    - unique
    - continuous
    - compact
    - efficient to work with

Euler Angles

- choose 3 successive rotations about different axes
  
  - e.g., \( RPY \) : \( \psi, \theta, \phi \)
  
  \[ R_{xyz} = Rot(z, \alpha) \cdot Rot(y, \beta) \cdot Rot(x, \gamma) \]
  
  - common alternative: \( \alpha, \beta, \gamma \)
  
  - problem: "gimbal lock"
  
  - problem: non-uniqueness \( RPY(0,90,0) = RPY(90,90,90) \)