Robotics Foundations

Henrik I Christensen

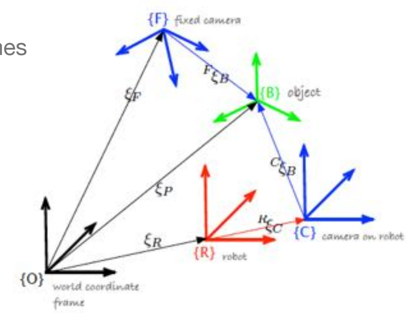
Outline

- Representation of position and orientation
  - Reference Frames
  - Representation of position and orientation in 2D and 3D
  - Models for transformations between reference frames
  - Homogenous coordinates
- Representations for time and motions
  - Trajectory models
  - Time varying reference frames
- Summary

Source: Many Illustrations / basic material adopted from P. Corke, Robotics, Vision and Control, STAR Vol. 73, Springer Verlag, 2011

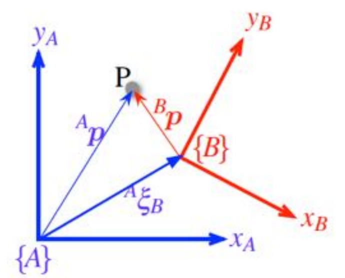
Reference Frames

- Robotics is all about management of reference frames
- Perception is about estimation of reference frames
- Planning is how to move reference frames
- Control is the implementation of trajectories for reference frames
- The relation between references frames is essential to a successful system



Representation of position and orientation

- We will make reference to different frames such as A and B
- The transformation between frames will be denoted by the the symbol xi (ξ)
- If we have a point P then we will use ^AP to denote that P is represented in reference frame A
- For transformations between reference frames we will use the notation $^A\xi_B$

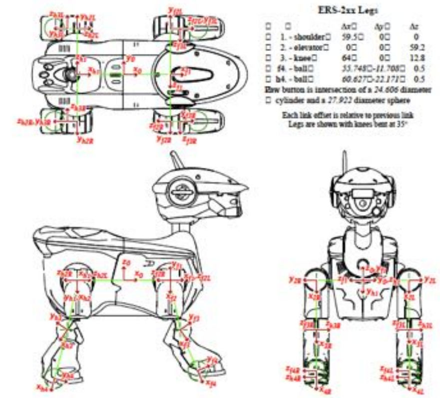


Use of reference frames - example

- Easy to consider

(c) Henrik I Christensen

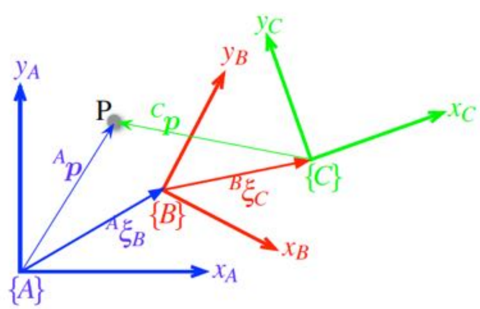
Use of reference frames - example



(c) Henrik I Christensen

Transformations

- We can compose transformation
  - ${}^{A}\xi_{C} = {}^{A}\xi_{B} \oplus {}^{B}\xi_{C}$
- We can represent a point ${}^{C}P$ in the frame A through the transformation
  - ${}^{A}P = ({}^{A}\xi_{B} \oplus {}^{B}\xi_{C})\ {}^{C}P$



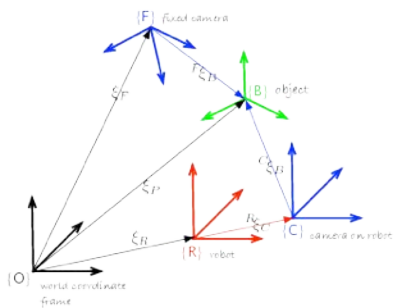
(c) Henrik I Christensen

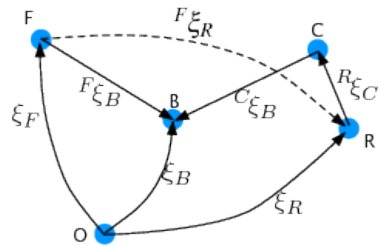
Basics concepts:

- A point P is described by a vector that specify the displacement from the origin of the reference frame.
- A set of points that represent a rigid object can be described by a single coordinate frame, and its constituent points are described by displacements from that coordinate frame
- The position and orientation of an object's coordinate frame is referred to as its pose
- A relative pose describes the pose of one coordinate frame with respect to another and is denoted by an algebraic variable ξ
- A coordinate vector describing a point can be represented with respect to a different coordinate frame by applying the relative pose to the vector using the · operator
- We can perform algebraic manipulation of expressions written in terms of relative poses

(c) Henrik I Christensen

Pose Algebra

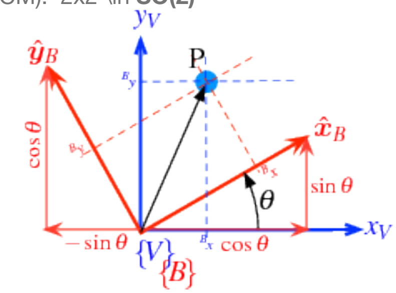




(c) Henrik I Christensen

Rotation in 2D

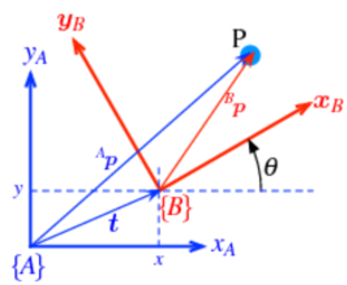
- Rotation Matrices (DCM): 2x2 \in **SO(2)**



- Rotation Matrices (DCM), 2x2 ∈ **SO(2)**
- Columns of **vRb**: axes of B in V. In MATLAB notation: [c -s; s c]
- 4 numbers, but only 1D **Manifold**

(c) Henrik I Christensen

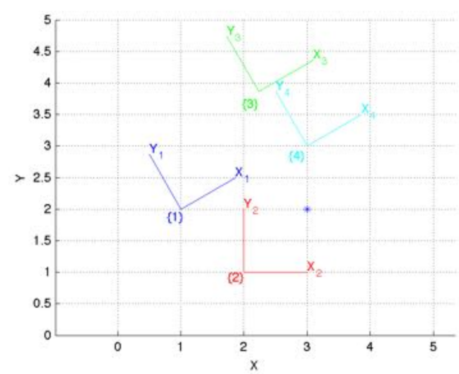
Poses in 2D



- (x,y,θ) or (R,t)
- Better: SE(2), next slide

(c) Henrik I Christensen

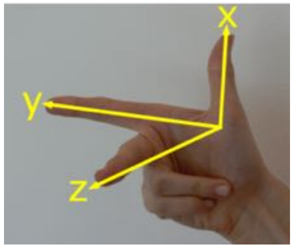
SE(2)

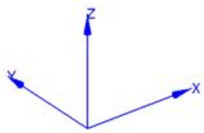


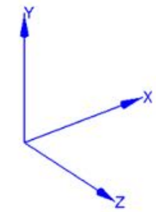
- [R t; 0 0 1] = 3x3 matrix ∈ **SE(2)**

(c) Henrik I Christensen

Rotations in 3D





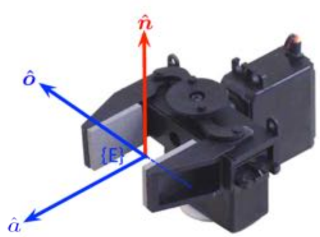


- Rotation Matrices (DCM), 3x3 $\in$ **SO(3)**
- Columns of $^{a}\mathbf{R}_{b}$: axes of B in A. In MATLAB notation: [Xb Yb Zb]
- 9 numbers, but 3D **Manifold**

Two vector representations

- In some applications it is convenient to talk about direction and orientation.
- Can you think of any?

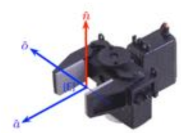
Representing end-effector Pose



- Z_E = approach vector a
- Y_E = orientation vector o
- X_E = o x a

Two vector representations

- In some applications it is convenient to talk about direction and orientation
  - Grasping
  - Docking



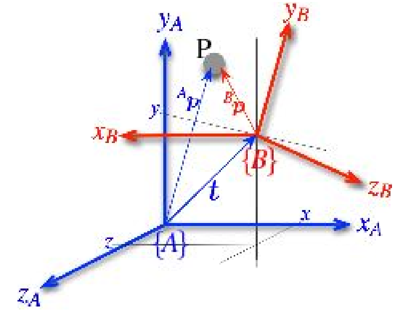
- We can define orientation (**o**) and direction (**a**) as two vectors. They complete define the rotation matrix R the third column can defined as the orthogonal vector **n**=**o** x **a**
- The resulting rotation matrix is then

$$\boldsymbol{R} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix}$$

2D Pose Representation

- In 2D we will represent the pose of an object by (x, y, θ)
- To represent translations we can use simple vector addition
  - $P_{new} = P_{old} + T$
- For rotations we can do simple operations
  - $P_{new} = R\ P_{old}$
- To make operations manageable it is easier to use homogenous transformations
- Representing the vector with an additional element
  - $P = (x, y, \theta, 1)^T$
- Or without orientation
  - $P = (x, y, 1)^T$
- We can consider also the 3 dimensional case

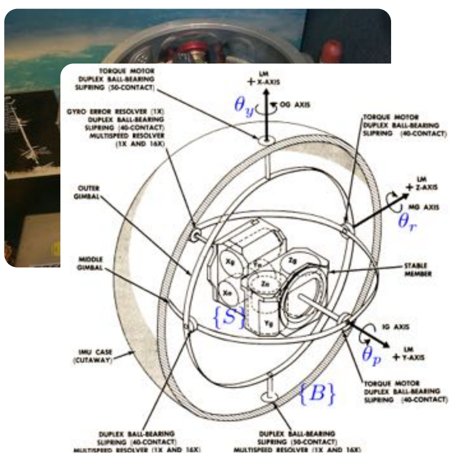
Poses in 3D



- [R t; 0 0 0 1], 4x4 matrix in **SE(3)**

Representing 3D Rotations

- Rotation Matrices (DCM)
- Euler Angles
  - Eulerian
  - Cardanian
  - Gimbal Lock
- Axis-Angle
- Unit Quaternions



3D Representation

- Basic model - $P = (x, y, z, 1)^T$
- Structure of a transformation matrix

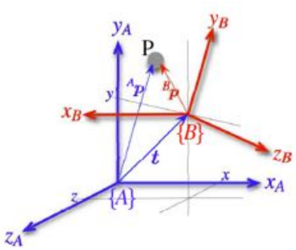
$$\xi = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- The relation between reference frames is exactly the same as before



Representing rotations

- There are two commonly used rotation representations
- Euler rotations are represented by rotations about axes
  - ZYZ, XYX, YXY, YZY, ZXZ
- Cardanian type rotations are about all three axes
  - XYZ, XZY, YZX, YXZ, ZXY, ZYX
- The Euler representation is widely used in Aerospace and Mechanical Dynamics. The most common model is ZYZ, i.e.

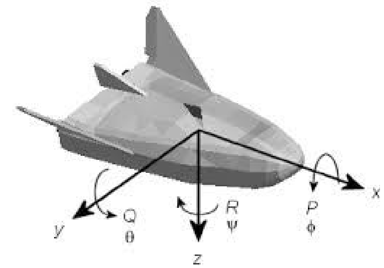
$$\boldsymbol{R} = \boldsymbol{R}_z(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$$

- Represented by the three angles of rotation.
- Another commonly used representation is pitch-roll-yaw

$$\boldsymbol{R} = \boldsymbol{R}_x(\theta_r)\boldsymbol{R}_y(\theta_p)\boldsymbol{R}_z(\theta_y)$$

- Often seen in maritime applications. Typically x direction is forward, y is to the right and z is downward.

Roll-Pitch-Yaw



$$\mathcal{R}_v^b(\phi,\theta,\psi) = \mathcal{R}_{v2}^b(\phi)\mathcal{R}_{v1}^{v2}(\theta)\mathcal{R}_v^{v1}(\psi) \quad (2.4)$$

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

$$= \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{pmatrix}, \quad (2.5)$$

Beard, 2011, Small Unmanned Aircraft

UNIT Quaternions

- 2D:
  - θ
  - [cθ -sθ; sθ cθ]
  - z = (cθ, sθ)
- 3D:
  - θ_r, θ_p, θ_y
  - [r_{11} r_{12} r_{13};r_{21} r_{22} r_{23};r_{31} r_{32} r_{33}]
  - q = c(θ/2) <s(θ/2) ***v***>

For the case of quaternions our generalized pose is $\xi \sim \mathring{q} \in \mathbb{Q}$ and

$$\mathring{q}_1 \oplus \mathring{q}_2 \mapsto s_1 s_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2, < s_1 \boldsymbol{v}_2 + s_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2 >$$

which is known as the quaternion or Hamilton product, and

$$\ominus \mathring{q} \mapsto \mathring{q}^{-1} = s, < -\boldsymbol{v} >$$

which is the quaternion conjugate. The zero pose $0 \mapsto 1 <0,0,0>$ which is the identity quaternion. A vector $\boldsymbol{v} \in \mathbb{R}^3$ is rotated $\mathring{q} \cdot \boldsymbol{v} \mapsto \mathring{q}\mathring{q}(\boldsymbol{v})\mathring{q}^{-1}$ where $\mathring{q}(\boldsymbol{v}) = 0, <\boldsymbol{v}>$ is known as a pure quaternion.

Representations of Pose - Overview

representation	$\oplus$	$\ominus$	rotn.	transl.	dim	MATLAB
$(x,y,\theta) \in \mathbb{R}^2 \times \mathbb{S}$			✓	✓	2D	
$\boldsymbol{T} \in SE(2)$	$\boldsymbol{T}_1\boldsymbol{T}_2$	$\boldsymbol{T}^{-1}$	✓	×	2D	`se2(x, y)`
$\boldsymbol{R} \in SO(2)$	$\boldsymbol{R}_1\boldsymbol{R}_2$	$\boldsymbol{R}^T$	×	✓	2D	`se2(0, 0, th)`
$\boldsymbol{T} \in SE(2)$	$\boldsymbol{T}_1\boldsymbol{T}_2$	$\boldsymbol{T}^{-1}$	✓	✓	2D	`se2(x, y, th)`
$(x,y,z,\Gamma) \in \mathbb{R}^3 \times \mathbb{S}^3$			✓	✓	3D	
$\boldsymbol{R} \in SO(3)$	$\boldsymbol{R}_1\boldsymbol{R}_2$	$\boldsymbol{R}^T$	×	✓	3D	`rotx`, `roty`, ...
$\Gamma \in \mathbb{S}^3$	×			✓	3D	`tr2eul`, `eul2tr`
$\Gamma \in \mathbb{S}^3$	×			✓	3D	`tr2rpyl`, `rpy2tr`
$\boldsymbol{T} \in SE(3)$	$\boldsymbol{T}_1\boldsymbol{T}_2$	$\boldsymbol{T}^{-1}$	✓	✓	3D	`transl(x,y,z)`
$\mathring{q} \in \mathbb{Q}$	$\mathring{q}_1\mathring{q}_2$	$\mathring{q}^{-1}$	×	✓	3D	`quaternion`, ...

Time and motion

- Robotics is often about movement from a position A to B ($P_A \rightarrow P_B$)
- Typically we have additional constraints
  - Smooth trajectory
  - Time constraints
  - Maximum speed
  - Acceleration
  - Jerk
- How can we formulate trajectories that satisfy these constraints?

Paths (Locus) vs. Trajectories (Locus + Time)

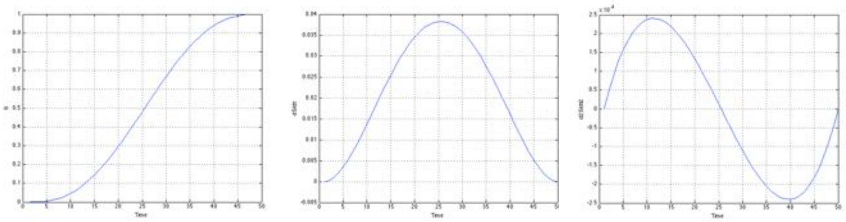
Smooth trajectories in 1-D

- For representation of trajectories we can use polynomials
- A typical example is a quintic polynomial
  - $S(t) = A\,t^5 + B\,t^4 + C\,t^3 + D\,t^2 + E\,t + F$ with $t \in [0, T]$
- It will have smooth first and second order derivatives
- An example of such models is the clothoid model for roads in Europe

- An alternative is use of linear segments with parabolic blend
  - Smooth trajectories, with linear velocity variations
  - The motion has step changes in acceleration
  - Where do we see this type of trajectories?

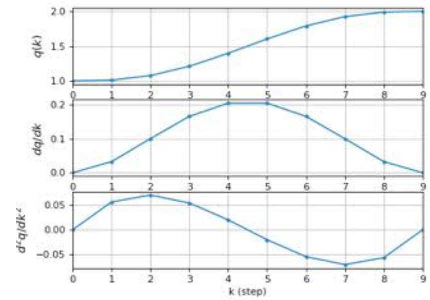
- Lets look at a couple of examples

5th order polynomial trajectory

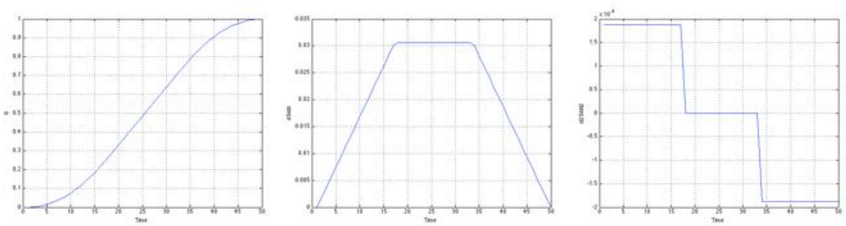


Example code from the RTB

```
from roboticstoolbox import quintic
tg = quintic(1, 2, 10)
tg
# Trajectory created by quintic: 10 time steps x 1 axes
len(tg)
# 10
tg.q
# array([1.    , 1.0115, 1.0764, 1.2099, 1.3967, 1.6033, 1.7901,
# 1.9236, 1.9885, 2.    ])
tg.plot()
```



Linear trajectory with parabolic blends

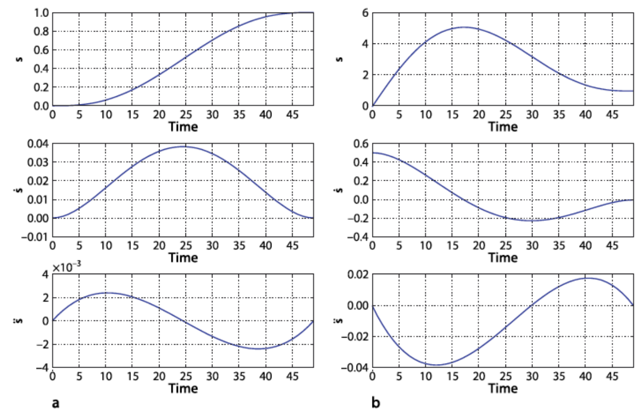


Quintics

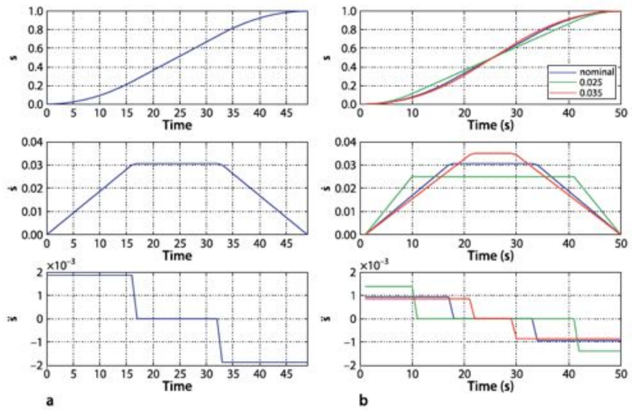
$$S(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

$$\dot{S}(t) = 5At^4 + 4Bt^3 + 3Ct^2 + 2Dt + E$$

$$\ddot{S}(t) = 20At^3 + 12Bt^2 + 6Ct + 2D$$

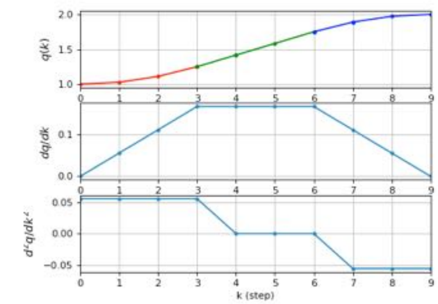


Linear Segment with Parabolic Blend

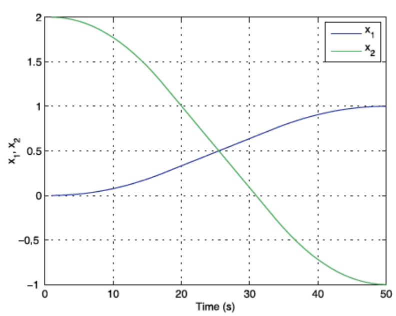


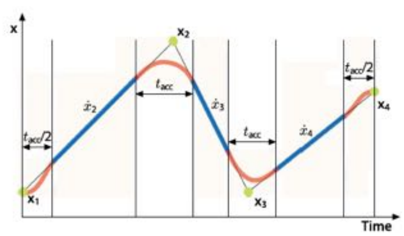
Pure trapezoidal

```
from roboticstoolbox import trapezoidal
tg = trapezoidal(1, 2, 10)
tg
# Trajectory created by trapezoidal: 10 time steps x 1 axes
len(tg)
# 10
tg.q
# array([1.    , 1.0278, 1.1111, 1.25  , 1.4167, 1.5833, 1.75  , 1.8889, 1.9722, 2.    ])
```



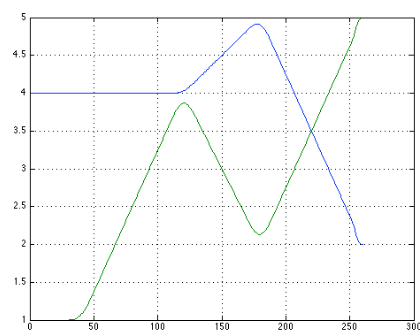
Multiple dimensions, Multi-SEGMENT

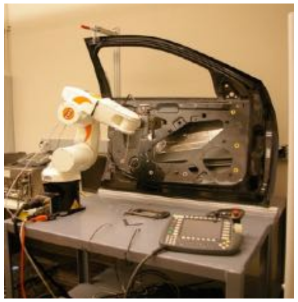




Multi-segment trajectories

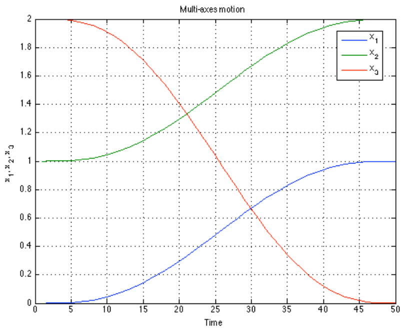
- Frequently the motion is not simply a motion from A to B, but a more complex motion to cover a set of trajectories.
- We can define a set of via points.
- We can “blend” at the via points to achieve acceleration around those areas, but closers to constant velocity elsewhere





Multi-dimensional case

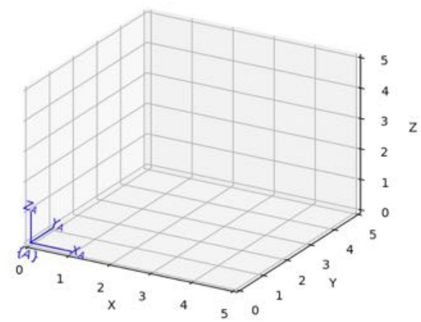
- The single dimensional case generalizes directly to multiple dimensions / multiple axes.
- Most robot controllers will directly generate trajectories given a specification of given constraints
- Example:



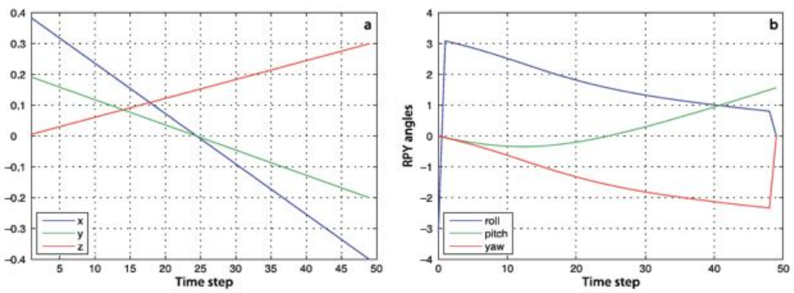
Cartesian motion

- We could do the same in Cartesian space
- We can define points and do linear interpolation between them

- Animate1 - step change
- Animate2 - smooth motion



Cartesian Motion



Examples of use of moving reference frames

- Most common example
  - Inertial Navigation System (INS)
    - Inertial Measurement Unit (IMU)
    - Estimation of ego-motion
  - Estimation of vehicle motion from GPS + INS/IMU
  - Estimation of end-effector motion
  - Estimation of motion of people (and gestures)

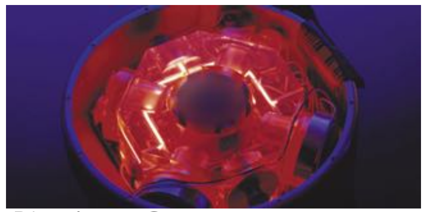
Inertial Navigation

- Measure angular velocity with gyroscope, acceleration with accelerometer
- Integrate over time:

$$\boldsymbol{R}\langle k+1\rangle = \delta_t \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}\langle k\rangle + \boldsymbol{R}\langle k\rangle$$

$${}^{0}\boldsymbol{a} = {}^{0}\boldsymbol{R}_B\, {}^{B}\boldsymbol{a}$$

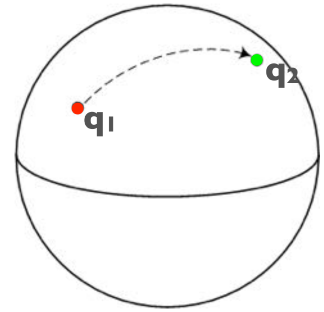
USS Alabama



Ring-laser Gyro

Interpolation in 3D

- Suppose $\delta = c(\theta/2) <s(\theta/2)\ \boldsymbol{v}>$ is rotation between q_1 and q_2
- Slerp(q_1, q_2, t) =
  $q_1 \oplus c(t\theta/2) <s(t\theta/2)\ \boldsymbol{v}>$



Time-varying coordinate frames

- Handling rotation of coordinate frames is a little more complex
- We can write the derivative of the rotation matrix

$$\dot{R}(s) = S(\omega)R(t)$$

- Where

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

- where omega describes the rotation around each axis
- We can do the transformation directly in homogenous coordinates, i.e.

$$T = \begin{bmatrix} S(\delta_\theta) & \delta_d \\ 0 & 0 \end{bmatrix} + I_{4x4}$$

Summary

- A few foundational concepts on position, orientation, time and trajectories
- Many different models for representation
- Position is relative easy
- Rotation has a number of issues to consider
- Using homogenous transformations often pays off
- We will use reference frames extensively.
- Robotics is about estimation, control and planning across references frames
- Many more foundational concepts covered in the:
  - Handbook of Robotics, B. *Siciliano & O. Khatib*, Springer Verlag, 2010
  - Robotics, Vision and Control, *P. Corke*, Springer Verlag, STAR Volume 73, Oct 2011 (incl. MATLAB toolbox)