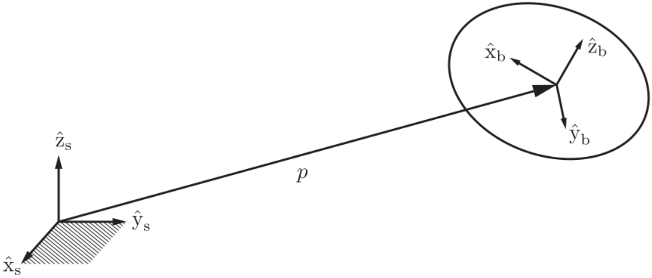
5ELEN018W - Robotic Principles
Lecture 3: Position and Orientation: Transformations

Dr Dimitris C. Dracopoulos

Pose

Pose is the *position* and *orientation* of one coordinate frame with respect to another reference coordinate frame.

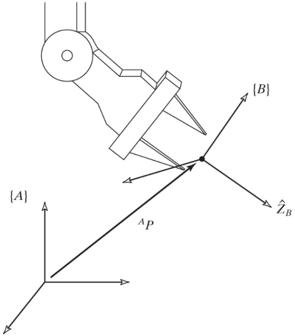
- Multiple coordinate frames are used in robotics to facilitate the computations for motion and different types of functionality.
- NASA is using them to simplify calculations!



Pose (cont'd)

The robotic hand needs to grasp something located in a specific point in space.

- The orientation of the hand needs to be described
- A coordinate frame is attached to the body (hand)
- The coordinate frame attached to the body needs to be described with respect to a reference coordinate frame (possible the world coordinate frame)

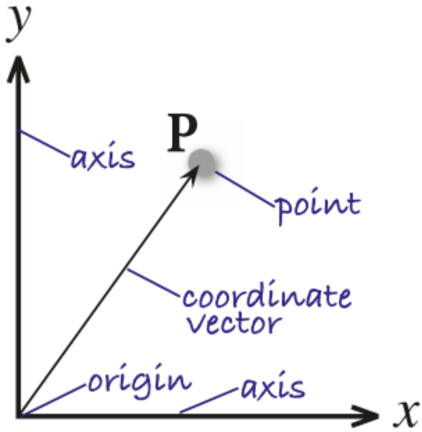


How to Specify Pose

Using transformations:

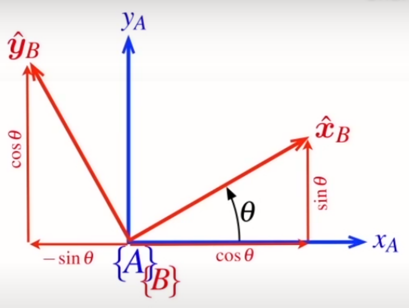
- Rotation
  - → represents orientation
  - → changes the reference frame in which a vector or frame is represented
  - → rotate a vector or a frame
- Translation (linear move along one of the axes)

Terminology of Coordinate Frames



Pose in the 2D Space

Rotation:



- A new coordinate frame $\{B\}$ with the same origin as $\{A\}$ but rotated counter-clockwise by angle θ
- Transforms vectors from new frame $\{B\}$ to the old frame $\{A\}$:

$$\begin{pmatrix} A_x \\ A_y \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} B_x \\ B_y \end{pmatrix} \quad (1)$$

Properties of the Rotation Matrix

$$\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

- The inverse matrix is the same as the Transpose! $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$

  $\rightarrow$ easy to compute
- The determinant is 1: $det(\boldsymbol{R}) = 1$

  $\rightarrow$ the length of a vector is unchanged after the rotation

How to Represent Translation

Just a vector with 2 elements corresponding to how much we move along the x and y axes.

$$V = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \tag{2}$$

Assuming P is the position of some object in a 2D space then we can apply transformation $\boldsymbol{T_V}$ by simply adding V to P:

$$\boldsymbol{T_V}(\boldsymbol{P}) = \boldsymbol{P} + \boldsymbol{V} \tag{3}$$

Homogeneous Form

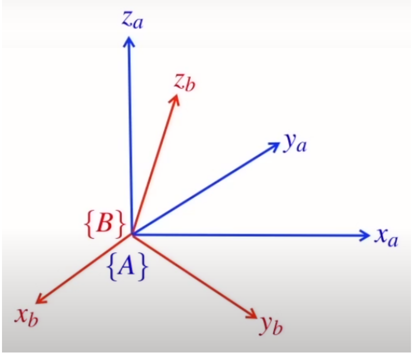
To represent both rotation and translation using a single matrix:

$$\begin{pmatrix} cos\theta & -sin\theta & V_x \\ sin\theta & cos\theta & V_y \\ 0 & 0 & 1 \end{pmatrix}$$

The left part is the rotation matrix and the right column is the translation vector!
A row $[0, 0, 0, 1]$ is appended in the end.

Pose in the 3D Space

Rotation:



- A new coordinate frame $\{B\}$ with the same origin as $\{A\}$ but rotated with respect to $\{A\}$
- Transforms vectors from new frame $\{B\}$ to the old frame $\{A\}$:

Elementary Rotation Matrices in 3D

Rotation about the x-axis:

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} \tag{4}$$

Rotation about the y-axis:

$$\boldsymbol{R}_y(\theta) = \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix} \tag{5}$$

Rotation about the z-axis:

$$\boldsymbol{R}_z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

Properties of the 3D Rotation Matrix

Similarly with the 2D case:

- The inverse matrix is the same as the Transpose! $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$
  - $\rightarrow$ easy to compute
- The determinant is 1: $det(\boldsymbol{R}) = 1$
  - $\rightarrow$ the length of a vector is unchanged after the rotation
- Rotations in 3D are not commutative (the order of rotation matters!)

Representation of Rotation in 3D as an Axis-Angle

Combining:

- a unit vector $\boldsymbol{e}$ indicating a single axis of rotation
- an angle θ describing the magnitude of the rotation about the axis

Example:

$$(axis, angle) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right) = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \frac{\pi}{2} \right) \tag{7}$$

a rotation of $90^\circ = \frac{\pi}{2}$ about the z-axis.
Reminder: $2\pi = 360^\circ \Rightarrow \pi = 180^\circ \Rightarrow \frac{\pi}{2} = 90^\circ$

- Use the Matlab `axang2rotm` to convert axis-angle rotation representation to rotation matrix and `rotm2axang` to convert rotation matrix to axis-angle representation!
- Matlab is using a row representation for this.

Matlab Example

```
>> a=[1   0         0
      0   cos(pi)   -sin(pi)
      0   sin(pi)   cos(pi)   ]

a =

    1.0000         0         0
         0   -1.0000   -0.0000
         0    0.0000   -1.0000

>> rotm2axang(a)

ans =

    1.0000         0         0    3.1416
```

How to Represent Translation in 3D

Just a vector with 3 elements corresponding to how much we move along the x, y and z axes.

$$V = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{8}$$

Assuming P is the position of some object then we can apply transformation $\boldsymbol{T_V}$ by simply adding V to P:

$$\boldsymbol{T_V}(\boldsymbol{P}) = \boldsymbol{P} + \boldsymbol{V} \tag{9}$$

Representing Pose in 3D

Different ways:

- Vector and 3 angles (roll, pitch, yaw)
- Homogeneous transformation (rotation and translation)
  - $\rightarrow$ advantage of transformations calculations using matrix multiplications!

Homogeneous Transformation in 3D

Construct a 4×4 array with the rotation matrix with 3 zeros (0) in the row below it, and the translation vector with an extra element of 1, as a column next to the rotation matrix:
e.g. rotation about x-axis with translation elements of v_x, x_y, v_z

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & cos\theta & -sin\theta & v_y \\ 0 & sin\theta & cos\theta & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10}$$

$\longrightarrow$ Remember, the matrix-based transformations allow to apply them (<u>or even to combine them!</u>) using **matrix multiplication**!

Homogeneous Transformation in 3D - Inverse Transformation

Although the inverse of the homogeneous transformation can be calculated as normally by computing the inverse of the original matrix (transformation), this can be done much faster.

- The homogeneous transformation matrix can be written as:

$$\begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}$$

  where R is the rotation matrix part and d is the translation vector part.

- then the inverse of the matrix (transformation) can be calculated as:

$$\begin{bmatrix} R' & -R' * d \\ 0 & 1 \end{bmatrix}$$