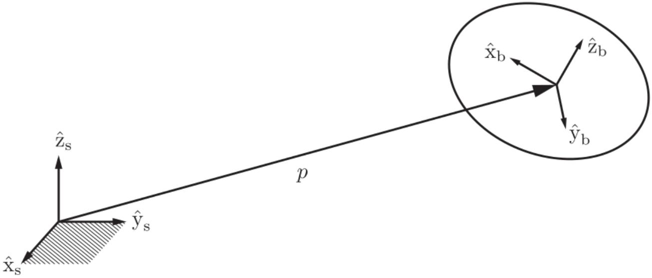
5ELEN018W - Robotic Principles
Lecture 3: Position and Orientation: Transformations

Dr Dimitris C. Dracopoulos

Pose

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

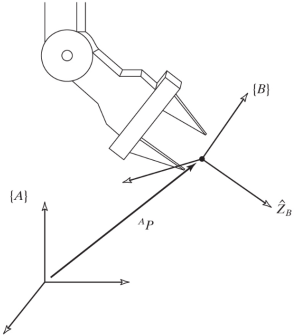
- Multiple coordinate frames are used in robotics to facilitate the computations for motion and different types of functionality.
- NASA is using them to simplify calculations!



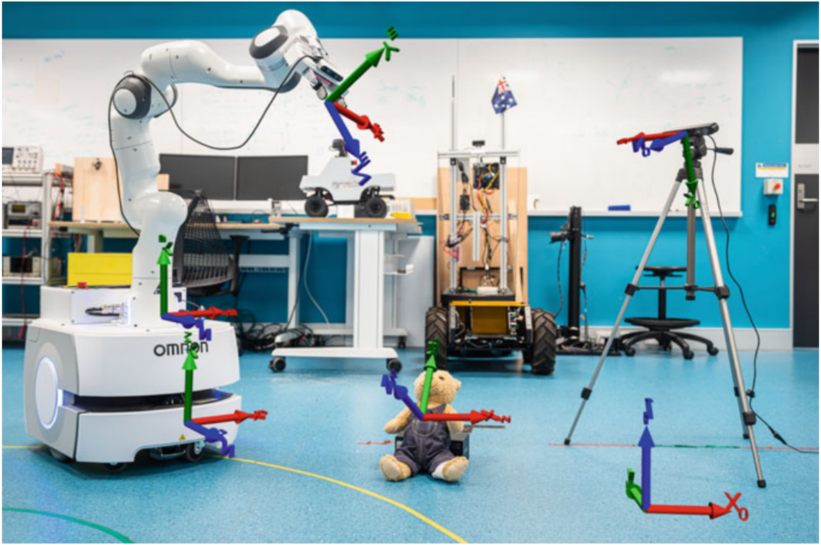
Pose (cont'd)

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)



Reference Frames in Real World Robots

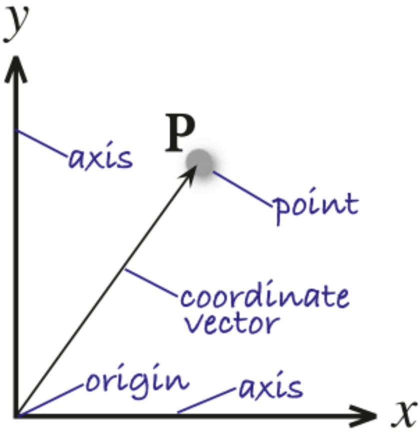


How to Specify Pose

Using transformations:

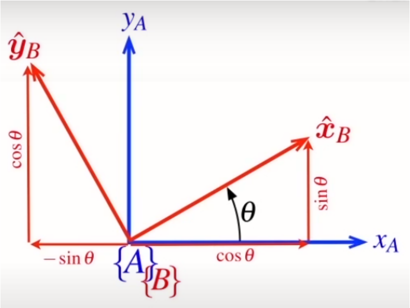
- Rotation
  - $\rightarrow$ represents orientation
  - $\rightarrow$ changes the reference frame in which a vector or frame is represented
  - $\rightarrow$ rotate a vector or a frame
- Translation (linear move along one of the axes)

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 (positive angle)
- Transforms vectors (their coordinates) from new frame $\{B\}$ to the old frame $\{A\}$:

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

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

Creating a rotation matrix in the Python Robotics Toolbox

Python Robotics Toolbox:
https://github.com/petercorke/RVC3-python

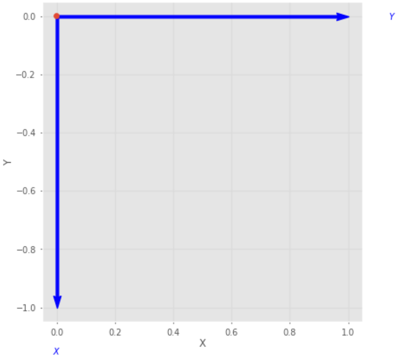
```
>>> from spatialmath.base import *
>>> R = rot2(math.pi/2)  # angle in radians by default
 array([[       0,        -1],
        [       1,         0]])
```

```
>>> rot2(90, 'deg')  # angle in degrees
 array([[      -1,         0],
        [       0,        -1]])
```

Visualising Rotation

The orientation represented by a rotation matrix can be visualised as a coordinate frame:

```
R2 = rot2(-math.pi/2)

trplot2(R2)
```



Operations for Matrix Rotations

The product of two rotation matrices is also a rotation matrix:

```
R2=rot2(-math.pi/2)
R=rot2(math.pi/2)

R@R2
```

- @ must be used for multiplication of NumPy arrays! Do not use *

The toolbox also supports symbolic operations:

```
from sympy import *
theta = Symbol('theta')
R = Matrix(rot2(theta))  # convert to SymPy matrix
```

Operations for Matrix Rotations (cont'd)

```
>>> R*R
Matrix([
[-sin(theta)**2 + cos(theta)**2,       -2*sin(theta)*cos(theta)]
[ 2*sin(theta)*cos(theta),       -sin(theta)**2 + cos(theta)**2]]

>>> simplify(R*R)
Matrix([
[cos(2*theta), -sin(2*theta)],
[sin(2*theta),  cos(2*theta)]])


>>> R.det()
sin(theta)**2 + cos(theta)**2

>>> R.det().simplify()
1
```

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:

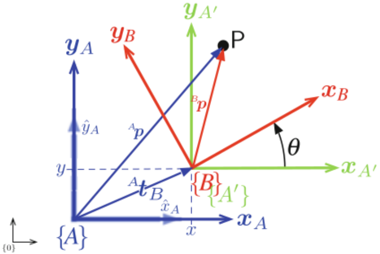
$$\begin{pmatrix} cos\theta & -sin\theta & V_x \\ sin\theta & cos\theta & V_y \\ 0 & 0 & 1 \end{pmatrix}$$

The left part is the rotation matrix and the right column is the translation vector!

A row $[0, 0, 1]$ is appended in the end.

- The above represents first a translation (V_x, V_y) followed by a rotation with angle θ.

Derivation of the Homogeneous Form



Derivation of the Homogeneous Form (cont'd)

$$
\begin{aligned}
\begin{pmatrix} A_x \\ A_y \end{pmatrix} &= \begin{pmatrix} A'_x \\ A'_y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
&= \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\
&= \begin{pmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_x \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix}
\end{aligned}
$$

or equivalently:

$$
\begin{pmatrix} A_x \\ A_y \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{A}\boldsymbol{R}_B(\theta) & {}^{A}\boldsymbol{t}_B \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix} \tag{4}
$$

- The homogeneous transformation can be considered as the relative pose which first translates the coordinate frame by ${}^{A}\boldsymbol{t}_B$ with respect to frame $\{A\}$ and then is rotated by ${}^{A}\boldsymbol{R}_B(\theta)$

Working with the Toolbox for Homogeneous Transformations

```
>>> trot2(0.3)  # translation of 0 and rotation by 0.3 radians.
```

which is equivalent to the composition of a translation of 0 followed by a rotation of 0.3 radians:

```
>>> transl2(0, 0) @ trot2(0.3)
```

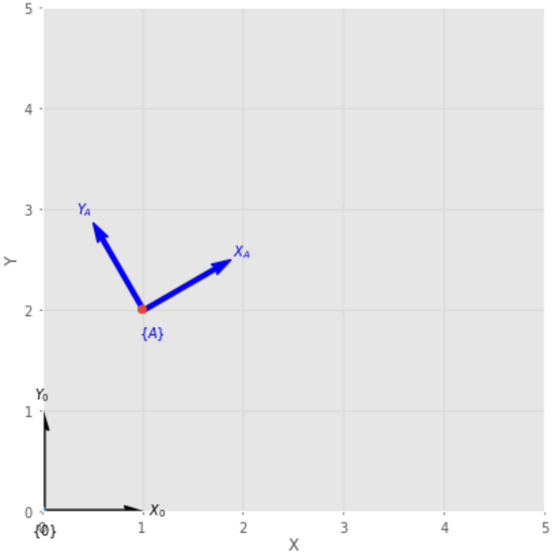
An example of a translation of (1, 2) followed by a rotation of 30 degrees:

```
>>> TA = transl2(1,2) @ trot2(30, "deg")
```

A coordinate frame representing the above pose can be plotted:

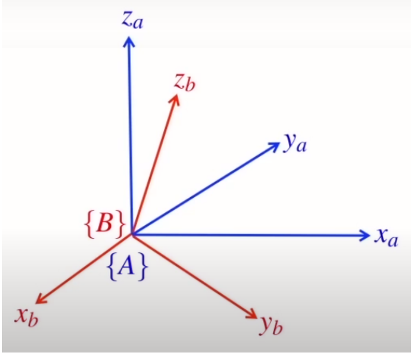
```
plotvol2([0, 5]);  # range of values in both axes is [0, 5]
trplot2(TA, frame="A", color="b");

# add the reference frame to the plot
T0 = transl2(0, 0);
trplot2(T0, frame="0", color="k");
```

Working with the Toolbox for Homogeneous Transformations (cont'd)



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:

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

Rotation about the y-axis:

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

Rotation about the z-axis:

$$\boldsymbol{R}_z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

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 <u>single</u> axis of rotation
- an angle θ describing the magnitude of the rotation about the axis

Example:

$$(axis, angle) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right) = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \frac{\pi}{2} \right) \tag{8}$$

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

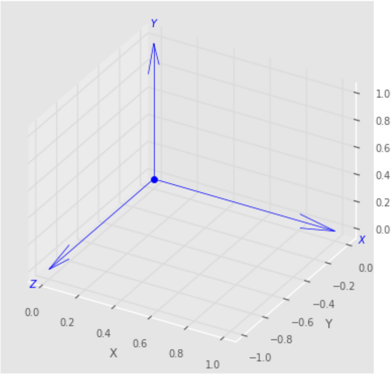
Python Toolbox Example

$\boldsymbol{R}_x(\frac{\pi}{2})$ can be represented as:

```
>>> R = rotx(math.pi / 2)
```

The orientation represented by a rotation matrix can be visualized as a coordinate frame rotated with respect to the reference coordinate frame:

```
trplot(R)
```



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.

$$V = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{9}$$

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

$$\boldsymbol{T_V}(\boldsymbol{P}) = \boldsymbol{P} + \boldsymbol{V} \tag{10}$$

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

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & cos\theta & -sin\theta & v_y \\ 0 & sin\theta & cos\theta & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

$\longrightarrow$ 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:

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