6ELEN018W - Applied Robotics
Lecture 2: Position and Orientation of a Robot

Dr Dimitris C. Dracopoulos

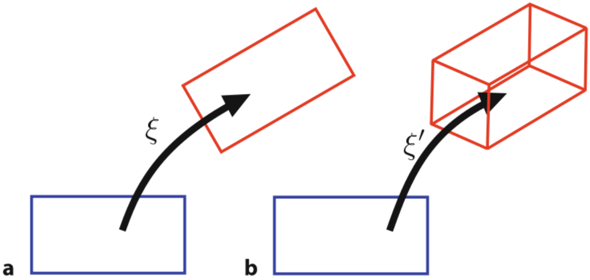
Why a Robot needs to know its Location?

A robot cannot perform any useful task (achieve its goal) if it is not able to detect its position and orientation within the environment.

- *A Robot is a goal oriented machine that can sense, plan and act.* (Peter Corke)

Pose of an Object

The position and orientation of an object (robot) is defined as its `pose`.

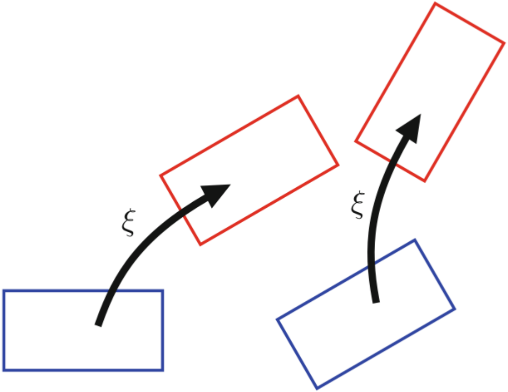


The motion ξ of a robot is defined with respect to its initial pose.

- ${}^{x}\xi_{y}$ denotes the motion from pose x to pose y.

Pose (cont'd)

Same motion ξ starting from 2 different initial poses:

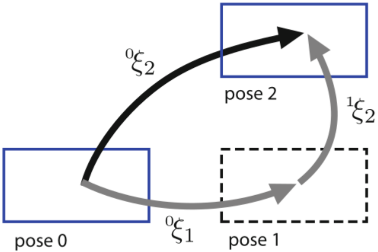


A pose of a robot can only be defined relatively to some other *reference* pose.

Pose (cont'd)

Composition of successive motions: ${}^0\xi_1$ followed by ${}^1\xi_2$:

$$
{}^0\xi_2 = {}^0\xi_1 \oplus {}^1\xi_2 \tag{1}
$$



- The order of motions matters! Composition of motions is not a commutative operation!

The inverse motion is denoted by $\ominus$:

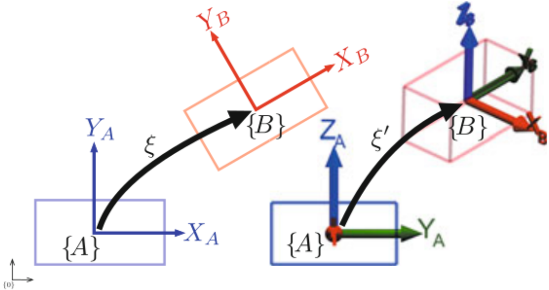
$$
{}^y\xi_x = \ominus {}^x\xi_y \tag{2}
$$

Coordinate Frames

To describe relative pose, 2 transformations are needed:

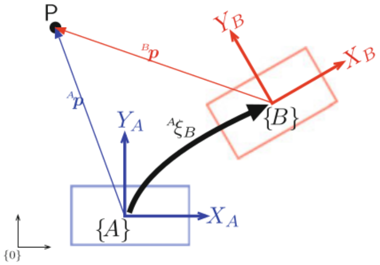
- translation
- rotation

To achieve this, a coordinate frame is attached to the body of a robot:



Location of a Point in Space

A point P can be described with respect to different coordinate vectors:



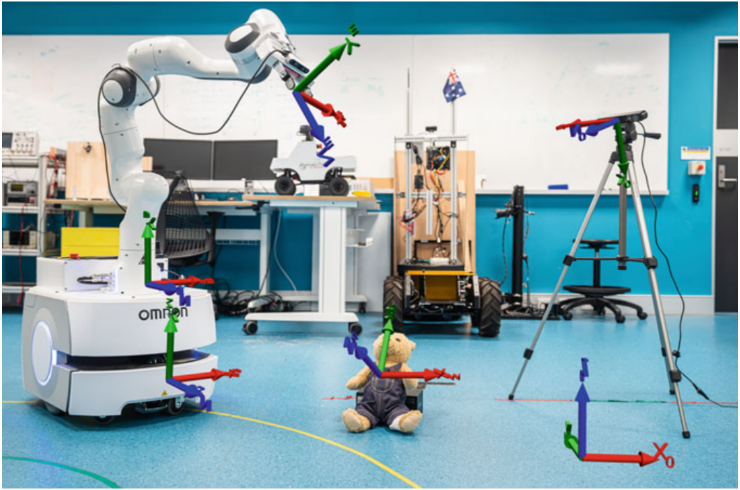
${}^{A}\boldsymbol{p}$ with respect to frame $\{A\}$ or ${}^{B}\boldsymbol{p}$ with respect to frame $\{B\}$.

$$ {}^{A}\boldsymbol{p} = A_B^{\xi} \cdot {}^{B}\boldsymbol{p} \tag{3} $$

where the $\cdot$ operator transforms the coordinate vector from one coordinate frame to another.

- $A_B^{\xi} \cdot {}^{B}\boldsymbol{p}$ is the motion from $\{A\}$ to $\{B\}$ and then to P.

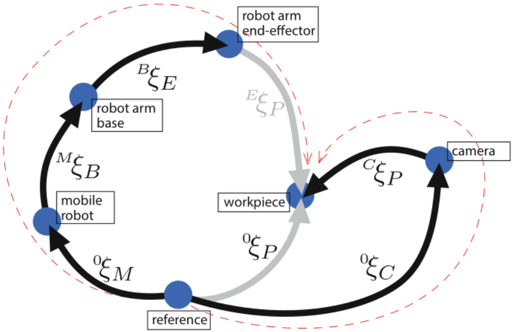
Reference Frames in Real World Robots



Pose Graphs

A *pose graph* is a directed graph which consists of:

- Vertices (poses)
- Edges with arrows (relative poses or motions)



Black arrows represent known relative poses, and the gray arrows are un- known relative poses that need to determined.

The Real World Robot

In order for the robot to grasp the workpiece, we need to know its pose relative to the robot's end effector: ${}^{E}\boldsymbol{\xi}_P$.
How to do this?

1. Look for 2 different equivalent paths which have the same start and end pose, one of the paths should include the unknown.
2. Solve for the unknown motion ${}^{E}\boldsymbol{\xi}_P$ (by inspecting the graph or using algebra).

Example: Choose the paths in red dashed lines:

$$ {}^{O}\boldsymbol{\xi}_M \oplus {}^{M}\boldsymbol{\xi}_B \oplus {}^{B}\boldsymbol{\xi}_E \oplus {}^{E}\boldsymbol{\xi}_P = {}^{O}\boldsymbol{\xi}_C \oplus {}^{C}\boldsymbol{\xi}_P \quad (4) $$

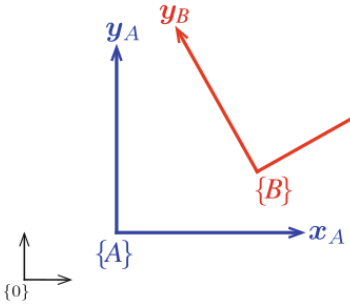
which can be rewritten for calculation of the unknown (desired) motion:

$$ {}^{E}\boldsymbol{\xi}_P = \ominus {}^{B}\boldsymbol{\xi}_E \ominus {}^{M}\boldsymbol{\xi}_B \ominus {}^{O}\boldsymbol{\xi}_M \oplus {}^{O}\boldsymbol{\xi}_C \oplus {}^{C}\boldsymbol{\xi}_P \quad (5) $$

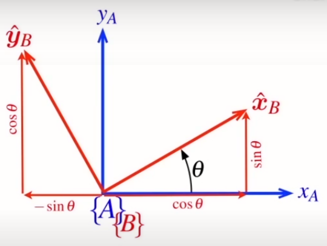
Pose in Two Dimensions (2D)

A point is represented using (x, y) coordinates or as a coordinate vector from the origin of the frame to the point:

$$\boldsymbol{p} = x\hat{\boldsymbol{x}} + y\hat{\boldsymbol{y}} \tag{6}$$



2D Rotation Matrix



$$(\hat{\boldsymbol{x}}_B \;\; \hat{\boldsymbol{y}}_B) = (\hat{\boldsymbol{x}}_A \;\; \hat{\boldsymbol{y}}_A) \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{7}$$

$$^{A}\boldsymbol{R}_B(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

is called the *rotation matrix* which transforms frame $\{A\}$ described by $(\hat{\boldsymbol{x}}_A \;\; \hat{\boldsymbol{y}}_A)$ into frame $\{B\}$ described by $(\hat{\boldsymbol{x}}_B \;\; \hat{\boldsymbol{y}}_B)$ (*positive values of θ are in the counter-clockwise direction*).

Transforming a Coordinate Vector

To transform a coordinate vector $({}^{B}p_x, {}^{B}p_y)$ with respect to frame $\{B\}$ to a vector in respect to frame $\{A\}$ the following form should be used:

$$\begin{pmatrix} {}^{A}p_x \\ {}^{A}p_y \end{pmatrix} = {}^{A}\boldsymbol{R}_B(\theta) \begin{pmatrix} {}^{B}p_x \\ {}^{B}p_y \end{pmatrix} \tag{8}$$

Properties of the Rotation Matrix

- The inverse matrix is the same as the Transpose! $\boldsymbol{R}^{-1} = \boldsymbol{R}^{T}$
  - easy to compute
- The determinant is 1: $det(\boldsymbol{R}) = 1$
  - the length of a vector is unchanged after the rotation (the same applies for the relative orientation of vectors)

Creating a rotation matrix in the Python Robotics Toolbox

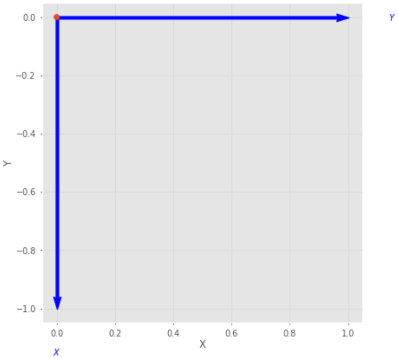
```
>>> R = rot2(math.pi/2)  # angle in radian by default
 array([[       0,        -1],
       [       1,         0]])
```

```
>>> 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 *

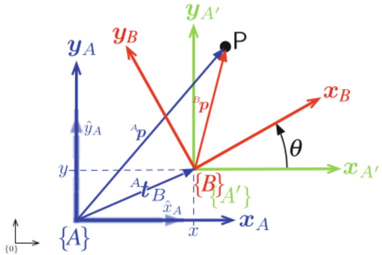
The toolbox also supports symbolic operations:

```
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
```

2D Homogeneous Transformation Matrix



To describe the relative pose of the frames below both a translation of the origin of frames as well as a rotation is needed:

1. A vector $^B\boldsymbol{p}$ with respect to frame $\{B\}$ is first transformed with respect to frame $\{A'\}$ which is a frame parallel to frame $\{A\}$. Use rotation.
2. A translation is then needed to transform the vector from frame $\{A'\}$ to frame $\{A\}$.

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

or equivalently:

$$
\begin{pmatrix} A_x \\ A_y \\ 1 \end{pmatrix} = \begin{pmatrix} {}^A\boldsymbol{R}_B(\theta) & {}^A\boldsymbol{t}_B \\ \mathbf{0}_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} {}^B x \\ {}^B y \\ 1 \end{pmatrix} \tag{9}
$$

- The homogeneous transformation can be considered as the relative pose (robot motion) which first translates the coordinate frame by $\boldsymbol{t}_B$ 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)
```

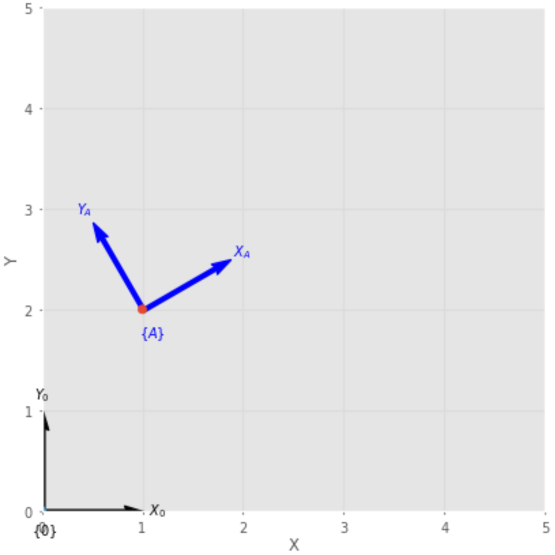
An example of a translation of (1, 2) followed by a rotation of 30 degrees:

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

A coordinate frame representing the above pose can be plotted:

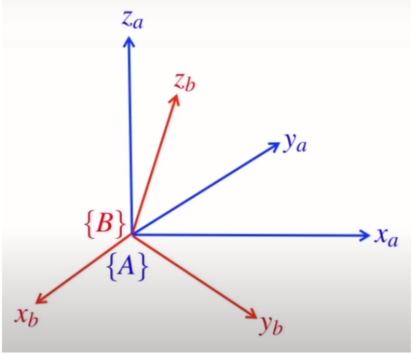
```
plotvol2([0, 5]);  # range of values in both axes is [0, 5]
trplot2(TA, frame="A", color="b");

# 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:

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

Rotation about the y-axis:

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

Rotation about the z-axis:

$$\boldsymbol{R}_z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{12}$$

Properties of the 3D Rotation Matrix

Similarly with the 2D case:

- The inverse matrix is the same as the Transpose! $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$
  - easy to compute
- The determinant is 1: $det(\boldsymbol{R}) = 1$
  - 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:

$$(axis, angle) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right) = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \frac{\pi}{2} \right) \tag{13}$$

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

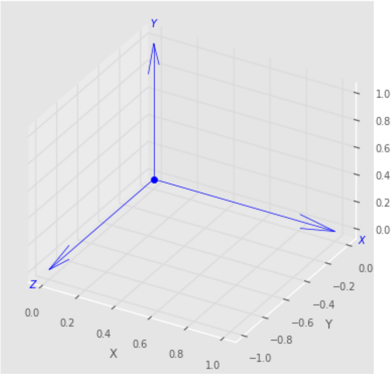
Python Toolbox Example

$\boldsymbol{R}_x(\frac{\pi}{2})$ 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.

$$V = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{14}$$

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

$$\boldsymbol{T_V}(\boldsymbol{P}) = \boldsymbol{P} + \boldsymbol{V} \tag{15}$$

Representing Pose in 3D

Different ways:

- Vector and 3 angles (roll, pitch, yaw)
- Homogeneous transformation (rotation and translation)
  - 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

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & cos\theta & -sin\theta & v_y \\ 0 & sin\theta & cos\theta & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

$\longrightarrow$ 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:

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