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.

Pose

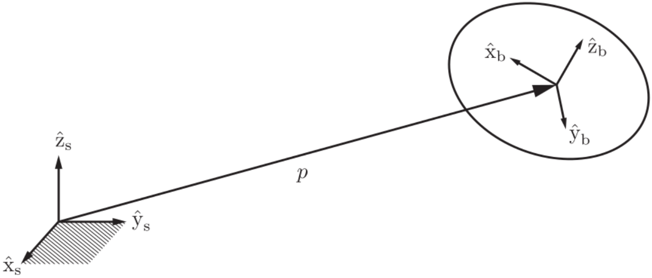
Pose is the *position* and *orientation* of one coordinate frame with respect to another reference coordinate frame.

- Multiple coordinate frames are used in robotics to facilitate the computations for motion and different types of functionality.

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.

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

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)

Pose (cont'd)

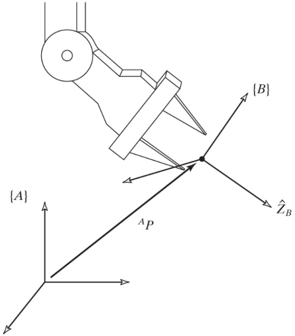
The robotic hand needs to grasp something located in a specific point in space.

- 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)

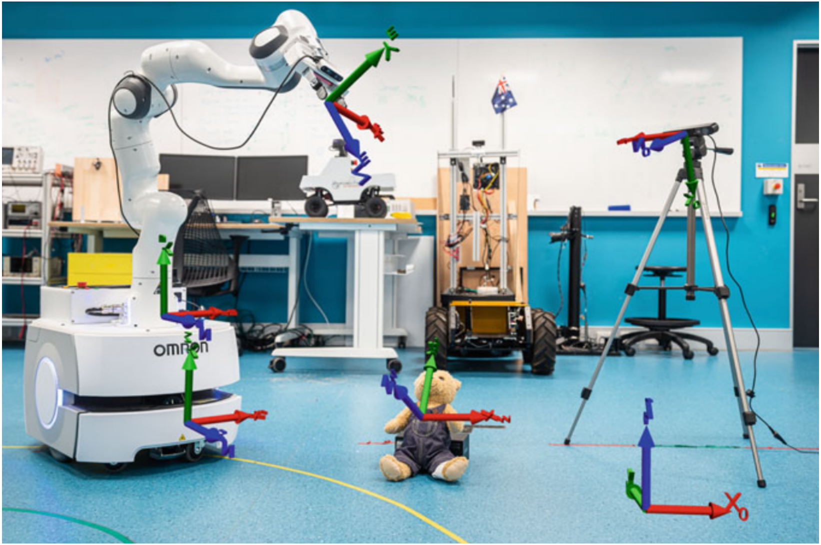
Pose (cont'd)

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:

How to Specify Pose

Using transformations:

- Rotation

How to Specify Pose

Using transformations:

- Rotation

  $\rightarrow$ represents orientation

How to Specify Pose

Using transformations:

- Rotation
  - → represents orientation
  - → changes the reference frame in which a vector or frame is represented

How to Specify Pose

Using transformations:

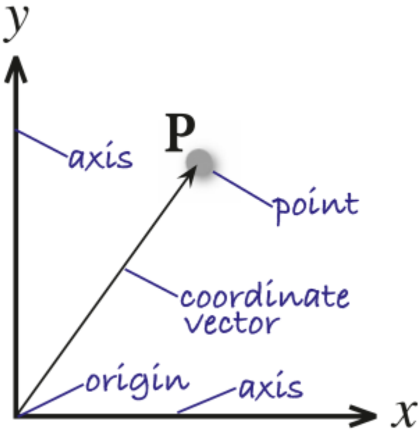
- Rotation
  - → represents orientation
  - → changes the reference frame in which a vector or frame is represented
  - → rotate a vector or a frame

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



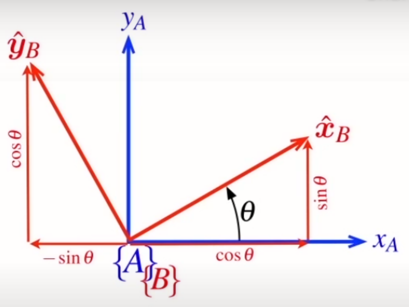
Pose in the 2D Space

Pose in the 2D Space

Rotation:

Pose in the 2D Space

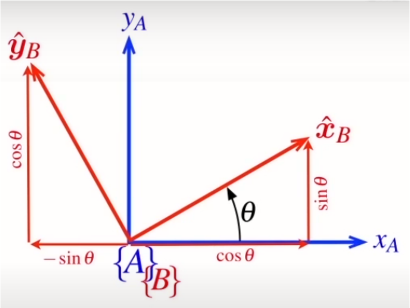
Rotation:



- A new coordinate frame $\{B\}$ with the same origin as $\{A\}$ but rotated counter-clockwise by angle θ (positive angle)

Pose in the 2D Space

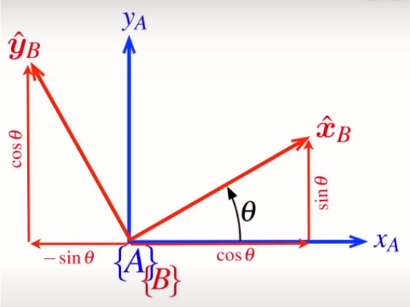
Rotation:



- 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 $\{A\}$:

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 $\{A\}$:

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

Properties of the Rotation Matrix

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

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! $\boldsymbol{R}^{-1} = \boldsymbol{R}^{T}$

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: $det(\boldsymbol{R}) = 1$

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

Working with Matlab - Rotations

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

```
>> syms theta
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

```
>> syms theta
>> R = rotm2d(theta)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

```
>> syms theta
>> R = rotm2d(theta)
>> R*R
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

```
>> syms theta
>> R = rotm2d(theta)
>> R*R
>> simplify(R*R)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

```
>> syms theta
>> R = rotm2d(theta)
>> R*R
>> simplify(R*R)
>> det(R)
```

Working with Matlab - Rotations

```
>> R=rotm2d(pi/2)
```

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

```
>> plottform2d(R)
```

Checking properties:

```
>> det(R)
>> det(R*R)
>> isequal(inv(R), R')
```

Symbolic mathematics can also be used:

```
>> syms theta
>> R = rotm2d(theta)
>> R*R
>> simplify(R*R)
>> det(R)
>> simplify(det(R))
```

How to Install the Additional (non-Mathworks) Matlab Robotics Toolbox

Download from
`https://github.com/petercorke/RVC3-MATLAB` the zip file and unzip it in a location where you can use it (in the university labs this should be done in your `H:` drive).
The `toolbox` directory in the directory you unzipped the above should be included in the Matlab path:

- In the university labs and from inside Matlab execute `path(path,'H:\RVC3-MATLAB\toolbox')` assuming that `H:\RVC3-MATLAB` contains the extracted contents of the zip file. You have to execute this every time you restart Matlab in the labs.
- In your personal computer where you have admin rights, you can use the `pathtool` command (from inside Matlab) to add the additional directory to the Matlab path permanently.

How to Represent Translation

How to Represent Translation

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

How to Represent Translation

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

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

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:

Homogeneous Form

To represent both rotation and translation using a single matrix:

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

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!

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.

Derivation of the Homogeneous Form

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

Derivation of the Homogeneous Form

$$\begin{pmatrix} A_x \\ A_y \end{pmatrix} = \begin{pmatrix} A'_x \\ A'_y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$= \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} {}^Bx \\ {}^By \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$= \begin{pmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_x \end{pmatrix} \begin{pmatrix} {}^Bx \\ {}^By \\ 1 \end{pmatrix}$$

or equivalently:

$$\begin{pmatrix} A_x \\ A_y \\ 1 \end{pmatrix} = \begin{pmatrix} {}^A\boldsymbol{R}_B(\theta) & {}^A\boldsymbol{t}_B \\ \mathbf{0}_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} {}^Bx \\ {}^By \\ 1 \end{pmatrix} \qquad (4)$$

Derivation of the Homogeneous Form

$$\begin{pmatrix} A_x \\ A_y \end{pmatrix} = \begin{pmatrix} A'_x \\ A'_y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$= \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$= \begin{pmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_x \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix}$$

or equivalently:

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

- 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

```
>> tformr2d(pi/2)    homogeneous transformation - rotation only
```

Working with the Toolbox for Homogeneous Transformations

```
>> tform2d(pi/2)    homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)   homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)   homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)    homogeneous transformation - rotation only
```

```
 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)   homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
>> T0 = trvec2tform([0 0])
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)    homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
>> T0 = trvec2tform([0 0])
>> plottform2d(T0, frame="0", color="k")    reference frame
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)    homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
>> T0 = trvec2tform([0 0])
>> plottform2d(T0, frame="0", color="k")    reference frame

>> TB = trvec2tform([2 1])
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)   homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
>> T0 = trvec2tform([0 0])
>> plottform2d(T0, frame="0", color="k")   reference frame

>> TB = trvec2tform([2 1])
>> plottform2d(TB,frame="B",color="r");
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)    homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
>> T0 = trvec2tform([0 0])
>> plottform2d(T0, frame="0", color="k")    reference frame

>> TB = trvec2tform([2 1])
>> plottform2d(TB,frame="B",color="r");
>> TAB = TA*TB
```

Working with the Toolbox for Homogeneous Transformations

```
>> tformr2d(pi/2)   homogeneous transformation - rotation only

 homogeneous transformation - translation only by x=1, y=2
>> trvec2tform([1 2])
```

Translation of (1, 2) followed by a rotation of $\frac{\pi}{4}$ radians:

```
>> trvec2tform([1 2])*tformr2d(pi/4)
```

or in a single step:

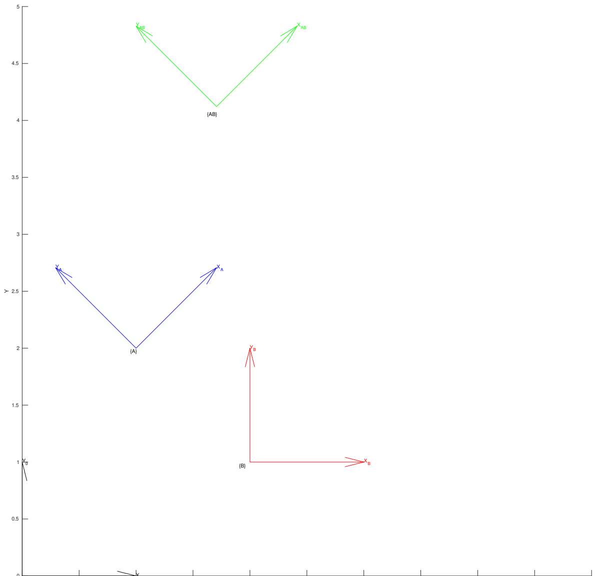
```
>> TA = tform2d(1, 2, pi/4)
```

A coordinate frame representing the above pose can be plotted:

```
>> axis([0 5 0 5]) range of values in both axes is [0, 5]
>> plottform2d(TA)
>> T0 = trvec2tform([0 0])
>> plottform2d(T0, frame="0", color="k")   reference frame

>> TB = trvec2tform([2 1])
>> plottform2d(TB,frame="B",color="r");
>> TAB = TA*TB
>> plottform2d(TAB,frame="AB",color="g");
```

Working with Matlab (cont'd)



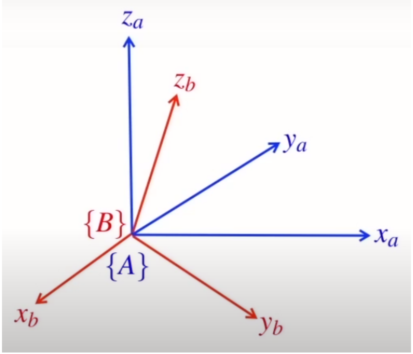
Pose in the 3D Space

Pose in the 3D Space

Rotation:

Pose in the 3D Space

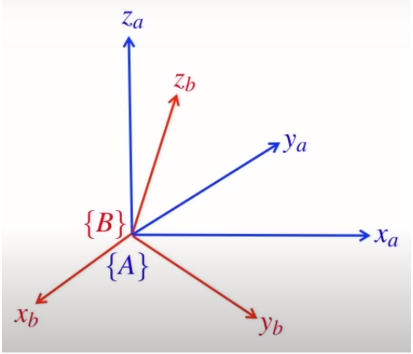
Rotation:



- A new coordinate frame $\{B\}$ with the same origin as $\{A\}$ but rotated with respect to $\{A\}$

Pose in the 3D Space

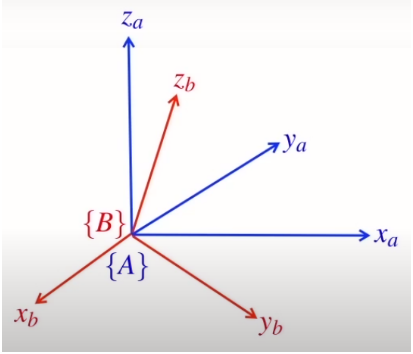
Rotation:



- 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 $\{A\}$:

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

Elementary Rotation Matrices in 3D

Rotation about the x-axis:

Elementary Rotation Matrices in 3D

Rotation about the x-axis:

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

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:

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:

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

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:

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

Properties of the 3D Rotation Matrix

Similarly with the 2D case:

Properties of the 3D Rotation Matrix

Similarly with the 2D case:

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

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: $det(\boldsymbol{R}) = 1$

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

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:

Representation of Rotation in 3D as an Axis-Angle

Combining:

- a unit vector $\boldsymbol{e}$ indicating a <u>single</u> axis of rotation

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

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:

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

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:

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

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

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:

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

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

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:

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

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

- Use the Matlab `axang2rotm` to convert axis-angle rotation representation to rotation matrix and `rotm2axang` to convert rotation matrix to axis-angle representation!

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:

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

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

- 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

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
```

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
```

Robotics Matlab Toolbox Example

```
>> R = rotmx(pi/2)
```

Robotics Matlab Toolbox Example

```
>> R = rotmx(pi/2)
```

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

Robotics Matlab Toolbox Example

```
>> R = rotmx(pi/2)
```

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

```
plottform(R, linewidth=2)
```

Robotics Matlab Toolbox Example

```
>> R = rotmx(pi/2)
```

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

```
plottform(R, linewidth=2)
```

Robotics Matlab Toolbox Example

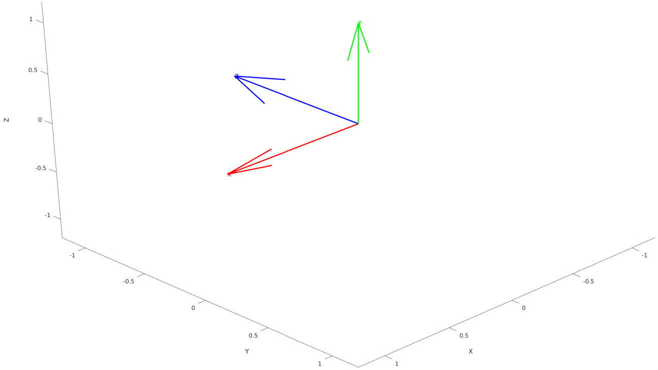
```
>> R = rotmx(pi/2)
```

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

```
plottform(R, linewidth=2)
```

To animate the reference frame moving to the specified relative pose:

```
>> animtform(R)
```


Z
1
0.5
0
-0.5
-1
Y
-1
-0.5
0
0.5
1
X
-1
-0.5
0
0.5
1

How to Represent Translation in 3D

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.

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.

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

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:

Representing Pose in 3D

Different ways:

- Vector and 3 angles (roll, pitch, yaw)

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

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:

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

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

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

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.

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:

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