Institute for Cognitive Systems
Technische Universität München
Prof. Gordon Cheng

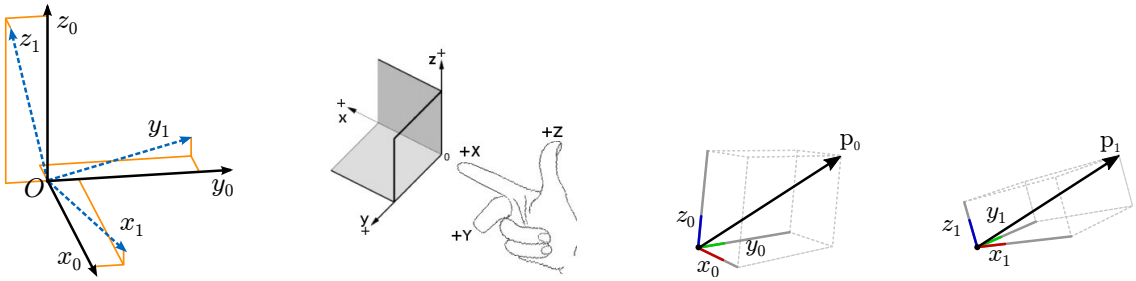
Multi-sensory Based Robot Dynamic Manipulation
Tutorial 1: Robot Spatial Descriptions

1 Mathematical toolbox for robot spacial description

a Rotations and Rotation Matrices

Let $(O_0x_0y_0z_0)$ and $(O_1x_1y_1z_1)$ be two different coordinate frames. The rotation matrix, mapping the coordinate frame $(O_1x_1y_1z_1)$ to the frame $(O_0x_0y_0z_0)$ is given by:

$$\mathbf{R}_0^1 = \begin{pmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{pmatrix} \tag{1}$$



(a) Translation vector

(b) Translation and Rotation of a coordinate frame

(c) The same vector **p** in two different coordinate frames

Figure 1: Rotation of $(O_1x_1y_1z_1)$ with respect to $(O_0x_0y_0z_0)$.

The columns of $\mathbf{R}_0^1$ represent the projection of the individual axis x_1, y_1, z_1 along each axis of the coordinate system $(O_0x_0y_0z_0)$ as depicted in figure 1a. The rotation matrix $\mathbf{R}_0^1$ can be used in order to represent a vector **p** in two different coordinate frames as exposed in Fig.1c:

$$\boxed{\mathbf{p}_0 = \mathbf{R}_0^1 \mathbf{p}_1} \tag{2}$$

All rotation matrices belongs to the special orthogonal matrix group "$\mathbf{SO}(3)$" (*with respect to matrix multiplication*). This group has the following properties:
$\forall \mathbf{R} \in \mathbf{SO}(3)$:

- $\det(\mathbf{R}) = 1$
- $\mathbf{R}^{-1} = \mathbf{R}^\top$
- $\mathbf{R}, \mathbf{R}^{-1} \in \mathbf{SO}(3) = \left\{ \mathbf{R} \in \mathbf{GL}(3) | \det(\mathbf{R}) = 1, \mathbf{R}\mathbf{R}^\top = \mathbf{I}) \right\}$
- The columns (and therefore the rows) of **R** are mutually orthogonal.
- Each column (and therefore each row) of **R** is a unit vector.

b Basic Rotation Matrices

Suppose the frame $(O_1x_1y_1z_1)$ is rotated through an angle φ about the z_0-axis, and it is desired to find the resulting transformation matrix $\mathbf{R}_{z,\varphi}$: Since the columns of a rotation matrix are the projection

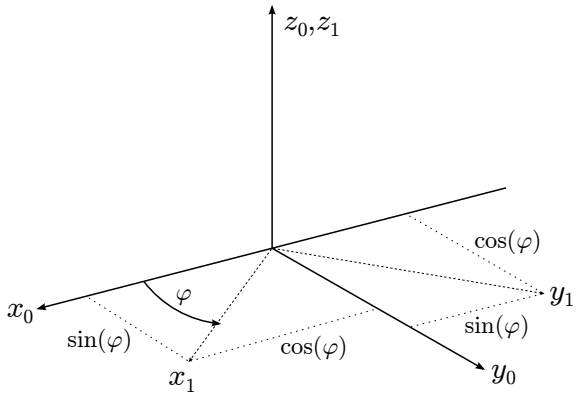


Figure 2: Rotation about z-axis

$$\mathbf{R}_{z,\varphi} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$\mathbf{R}_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix} \quad (4)$$

$$\mathbf{R}_{y,\varphi} = \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \quad (5)$$

Figure 3: Elementary rotation matrices about z, x and y-axis

of each individual axis x_1, y_1, z_1 of the current frame along the axes of the reference coordinate system $(O_0x_0y_0z_0)$ as exposed in eq.(1), we can easily build the corresponding matrix $\mathbf{R}_{z,\varphi}$ from the figure 2. Similar results can of course be obtained for the rotation matrices representing rotations about the x_0 and y_0-axes as exposed in figure 3.

c Composition of Rotations

The orientation of a coordinate frame with respect to a base frame can be obtained by a *sequence of consecutive rotations*. In this case, the resulting rotation matrix is computed as a *product of the corresponding elementary rotation matrices*. This is called composition of rotations. It is important to notice that such a product can be performed in *two different ways*[1], whether each elementary rotation has to be achieved with respect to the ***current coordinate frame*** or with respect to the ***reference coordinate frame***.

c.1 Rotation with respect to the current coordinate frame

We use the shorthand notation $c_\theta = \cos(\theta)$, $s_\theta = \sin(\theta)$ for trigonometric functions. Suppose a rotation matrix $\mathbf{R}$ represents a rotation of φ degrees about the current y-axis followed by a rotation of θ

[1] This is a direct consequence of the fact that rotation matrices are not commutative.

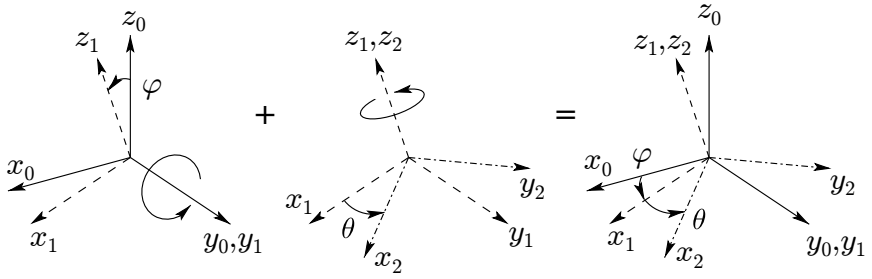


Figure 4: Composition of rotations about current axes.

degrees about the current z-axis. Refer to Figure 4. Then the matrix $\mathbf{R}_1$ is given by:

$$\mathbf{R}_1 = \mathbf{R}_{y_0,\varphi} \cdot \mathbf{R}_{z_1,\theta} \tag{6}$$

$$= \begin{bmatrix} c_\varphi & 0 & s_\varphi \\ 0 & 1 & 0 \\ -s_\varphi & 0 & c_\varphi \end{bmatrix} \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} c_\psi c_\theta & -c_\psi s_\theta & s_\psi \\ s_\theta & c_\theta & 0 \\ -s_\psi c_\theta & 0 & c_\psi \end{bmatrix}. \tag{8}$$

In this case the composite rotation performed relative to the **current frame** is obtained by **postmultiplying** the successive rotation matrices.

c.2 Rotation with respect to a fixed frame

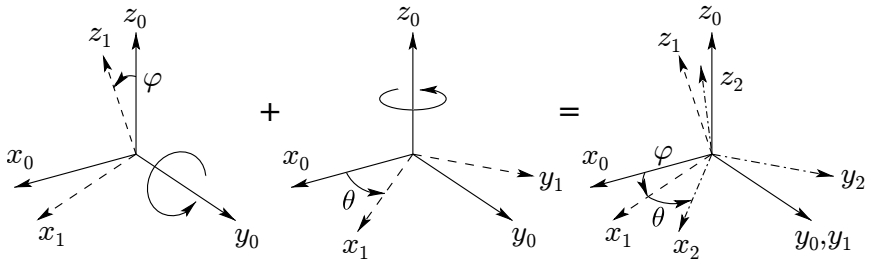


Figure 5: Composition of rotations about fixed axes.

In this case, we will refer to $(O_0x_0y_0z_0)$ as the **fixed frame**. It turns out that the correct composition law is simply to multiply the successive rotation matrices *in the reverse order* (**premultiply**) from that given by (6), and the new rotation $\mathbf{R}_2$ is shown as:

$$\mathbf{R}_2 = \mathbf{R}_{z_0,\theta} \cdot \mathbf{R}_{y_0,\varphi} \tag{9}$$

d Euler Angles

A rotation matrix **R** can also be described as a product of successive rotations about the principal coordinate axes x_0, y_0 and z_0 in a specific order. These rotations define the *yaw, pitch*, and *roll* angles (shown in Figure 6), which we shall also denote respectively ψ, θ, φ:

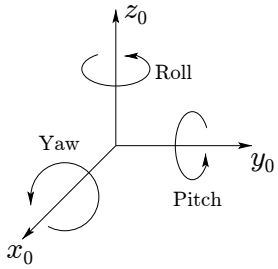


Figure 6: Roll, pitch, and yaw angles.

For this course, we specify the order of rotation as x-y-z, in other words, first a *yaw* about x_0 through an angle ψ, then *pitch* about the y_0 by an angle θ, and finally *roll* about the z_0 by an angle φ. Since the successive rotations are *relative to the fixed frame* $(O_0 x_0 y_0 z_0)$, the resulting transformation matrix is given by **premultiplying** the corresponding rotation matrices:

$$\mathbf{R}_1 = \mathbf{R}_{z,\varphi} \cdot \mathbf{R}_{y,\theta} \cdot \mathbf{R}_{x,\psi} \tag{10}$$

$$= \begin{bmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix} \tag{11}$$

$$= \begin{bmatrix} c_\varphi c_\theta & -s_\varphi c_\psi + c_\varphi s_\theta s_\psi & s_\varphi s_\psi + c_\varphi s_\theta c_\psi \\ s_\varphi c_\theta & c_\varphi c_\psi + s_\varphi s_\theta s_\psi & -c_\varphi s_\psi + s_\varphi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \tag{12}$$

This rotation matrix also can be represented as z-y-x order if the composition is given with respect to the current frame.

e Euler Angles from matrix R

Given a rotation matrix **R**, it is possible to obtain the set of euler angles ψ, θ, φ. For example, using trigonometric relationships:

$$d = \pm\sqrt{1 - \mathbf{R}(3,1)^2} \tag{13a}$$

$$\theta = arctan2(-\mathbf{R}(3,1), d) \tag{13b}$$

$$\varphi = arctan2(\mathbf{R}(2,1), \mathbf{R}(1,1)) \tag{13c}$$

$$\psi = arctan2(\mathbf{R}(3,2), \mathbf{R}(3,3)) \tag{13d}$$

Notice that this solution requires that $\mathbf{R}(3,1) \neq 1$, or $\mathbf{R}(1,1) \& \mathbf{R}(3,3) \neq 0$. A proper solution can be found analyzing these three cases individually, using eq.(12).

f Important note

There are **24 different possible representation** of a given rotation sequence:

- Post-multiplication (also called intrinsic rotations):
  - **Proper Euler angles** (z-x'-z", x-y'-x, y-z'-y", z-y'-z", x-z'-x", y-x'-y")
  - **Tait-Bryan angles** (x-y'-z", y-z-x", z-x'-y", x-z'-y", z-y'-x", y-x'-z") (used in EI7428 course)
- Pre-multiplication (also called extrinsic rotations):
  - **Proper Euler angles** (z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y)
  - **Tait-Bryan angles** (x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z) (used in EI7342 and EI7408 courses)

Tait-Bryan angles are sometimes called Euler angles as well.

g Translations

Let $(O_0x_0y_0z_0)$ and $(O_1x_1y_1z_1)$ be two frames with the same orientation but with different origins, as depicted on Figure 7. If we are given the coordinates $\mathbf{p}_1$ of a point $\mathbf{p}$ with respect to frame $(O_1x_1y_1z_1)$, and if we know the translation vector $\mathbf{p}_0^1$ between frames $(O_1x_1y_1z_1)$ and $(O_0x_0y_0z_0)$, it is then trivial to compute the coordinates $\mathbf{p}_0$ of the point $\mathbf{p}$ with respect to $(O_0x_0y_0z_0)$:

$$\boxed{\mathbf{p}_0 = \mathbf{p}_1 + \mathbf{p}_0^1} \tag{14}$$

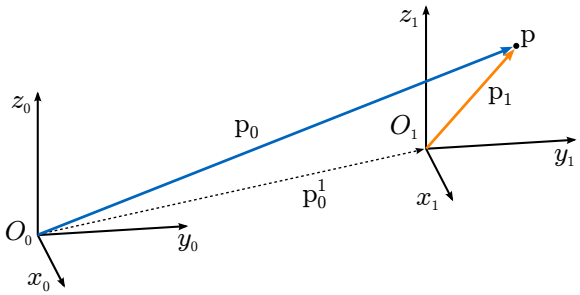


Figure 7: Translation vector

As previously, this result can be generalized to an arbitrary number of frames and translation vectors:

$$\boxed{\mathbf{p}_0 = \mathbf{p}_n + \mathbf{p}_{n-1}^n + \cdots + \mathbf{p}_0^1 = \mathbf{p}_n + \mathbf{p}_0^n} \tag{15}$$

h Combining Translations and Rotations

h.1 Problem statement

We now consider in Figure 8, the case of two frames $(O_0x_0y_0z_0)$ and $(O_1x_1y_1z_1)$ with both different origins and orientations. The coordinate frame $(O_1x_1y_1z_1)$ is here obtained from the frame $(O_0x_0y_0z_0)$ by first applying a rotation, specified by the matrix $\mathbf{R}_0^1$, and followed by a translation $\mathbf{p}_0^1$ with respect to $(O_0x_0y_0z_0)$:

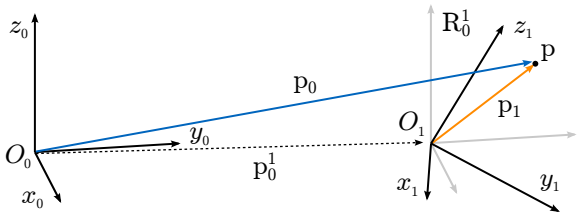


Figure 8: Combining Translations and Rotations

If we are given the coordinates $\mathbf{p}_1$ of a point $\mathbf{p}$ with respect to frame $(O_1x_1y_1z_1)$, and if we know the translation vector $\mathbf{p}_0^1$ and the rotation matrix $\mathbf{R}_0^1$ of the frame $(O_1x_1y_1z_1)$ with respect to the frame $(O_0x_0y_0z_0)$, it is then straightforward to compute the coordinates $\mathbf{p}_0$ of the point $\mathbf{p}$ with respect to $(O_0x_0y_0z_0)$ as an **Affine mapping**:

$$\boxed{\mathbf{p}_0 = \mathbf{R}_0^1\mathbf{p}_1 + \mathbf{p}_0^1} \tag{16}$$

Affine mappings are applications $\mathbf{T} : \mathbb{R}^n \to \mathbb{R}^n$, defined by a matrix $\mathbf{A} \in \mathbf{GL}(n)$ and a vector $\mathbf{b} \in \mathbb{R}^n$ such that:

$$\mathbf{T}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}, \ \forall \mathbf{x} \in \mathbb{R}^n$$

The set of all such affine transformations is called the affine group of dimension n, denoted by $\mathbf{A}(n)$. Since $\mathbf{T}$ is not a linear map, it cannot be easily generalized by simple matrix multiplication to the case of n distinct transformations. Let's consider for example the two-joint manipulator depicted in Figure 9a: the coordinates $\mathbf{p}_0$ of the point $\mathbf{p}$ with respect to $(O_0x_0y_0z_0)$ can be computed based on the coordinates of $\mathbf{p}_2$ of $\mathbf{p}$ with respect to $(O_2x_2y_2z_2)$ as:

$$\mathbf{p}_0 = \mathbf{R}_0^2\mathbf{p}_2 + \mathbf{p}_0^2 \tag{17}$$

The problem in the case of a robot, is that $\mathbf{p}_0^2$ **is not known** and must therefore be *computed iteratively, based on the known relative transformations between joints*:

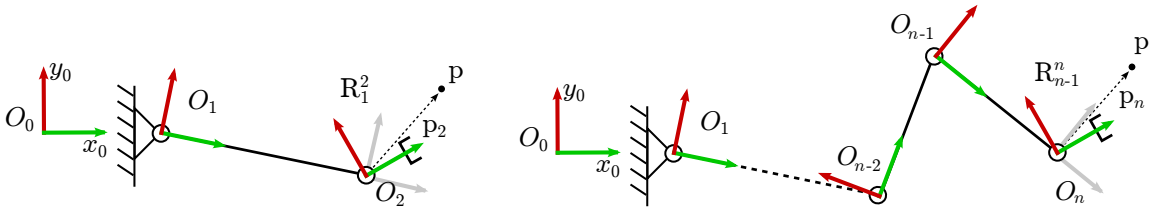
$$\begin{aligned} \mathbf{p}_0 &= \mathbf{R}_0^2\mathbf{p}_2 + \mathbf{p}_0^2 && (18)\\ &= \underbrace{\mathbf{R}_0^1\mathbf{R}_1^2\mathbf{p}_2}_{(a)} + \underbrace{\mathbf{R}_0^1\mathbf{p}_1^2}_{(b)} + \underbrace{\mathbf{p}_0^1}_{(c)} && (19) \end{aligned}$$

where:

- (a) is the coordinate of point $\mathbf{p}$ w.r.t frame 2 expressed in frame 0
- (b) is the Translation vector from frame 1 to frame 2 w.r.t frame 0
- (c) is the Translation vector from frame 0 to frame 1 w.r.t frame 0

This rapidly becomes intractable in the case of a n-link robot, such as the one of Figure 9b:

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{R}_0^n \mathbf{p}_n + \mathbf{p}_0^n & (20) \\ &= \mathbf{R}_0^1 \mathbf{R}_1^2 \cdots \mathbf{R}_{n-1}^n \mathbf{p}_n + \mathbf{R}_0^1 \mathbf{R}_1^2 \cdots \mathbf{R}_{n-2}^{n-1} \mathbf{p}_{n-1}^n + \cdots + \mathbf{R}_0^1 \mathbf{p}_1^2 + \mathbf{p}_0^1 & (21) \end{aligned}$$



(a) 2-joint planar robot

(b) *n*-joint planar robot

It is therefore necessary to find a simpler and more compact (i.e. linear) way to express these transformations.

h.2 Homogeneous Coordinates and Homogeneous Transformations

Obtaining simpler notations is possible by introducing so called **homogeneous coordinates**. The trick is here to add one dimension to the transformation matrix in order to account for both rotation and translation in a single linear transformation.

Using Homogeneous coordinates the vector $\mathbf{x} \in \mathbb{R}^n$ becomes $\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1}$. As a result, the affine mapping $\mathbf{T}$ becomes a linear mapping:

$$\begin{aligned} \mathbf{T} : \mathbb{R}^{n+1} &\rightarrow \mathbb{R}^{n+1} \\ \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} &\rightarrow \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0}_{1\times n} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \end{aligned}$$

where the matrix $\begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & 1 \end{pmatrix}$ with $\mathbf{A} \in \mathbf{GL}(n)$ and $\mathbf{b} \in \mathbb{R}^n$ is an element of $\mathbf{GL}(n+1)$. In our case, the matrix $\mathbf{A}$ is a rotation matrix and the vector $\mathbf{b}$ is a translation vector. As e result, the transformation of equation (16) can be reformulated as:

$$\begin{bmatrix} \mathbf{p}_0 \\ 1 \end{bmatrix} = \mathbf{H}_0^1 \begin{bmatrix} \mathbf{p}_1 \\ 1 \end{bmatrix} \tag{22}$$

where:

$$\mathbf{H}_0^1 = \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{p}_0^1 \\ 0 & 1 \end{bmatrix} \in \mathbf{SE}(3) \tag{23}$$

More interesting, the transformation of (21) can be reformulated as:

$$\begin{bmatrix} \mathbf{p}_0 \\ 1 \end{bmatrix} = \mathbf{H}_0^n \begin{bmatrix} \mathbf{p}_n \\ 1 \end{bmatrix} \tag{24}$$

Where:

$$\mathbf{H}_0^n = \begin{bmatrix} \mathbf{R}_0^n & \mathbf{p}_0^n \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \mathbf{H}_0^1 \cdot \mathbf{H}_1^2 \cdots \mathbf{H}_{n-1}^n \tag{25}$$

i Inverse mapping

The inverse of a homogeneous transformation is given by:

$$\boxed{\mathbf{H}_{i+1}^{i} = \left(\mathbf{H}_{i}^{i+1}\right)^{-1} = \begin{bmatrix} \left(\mathbf{R}_{i}^{i+1}\right)^{T} & -\left(\mathbf{R}_{i}^{i+1}\right)^{T} \mathbf{p}_{i}^{i+1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}} \tag{26}$$

Proof: Let $\left(\mathbf{H}_{i}^{i+1}\right)^{-1} = \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{p}} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$ denote the inverse of $\mathbf{H}_{i}^{i+1} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$. Therefore we have:

$$\mathbf{H}_{i}^{i+1}\left(\mathbf{H}_{i}^{i+1}\right)^{-1} = \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{p}} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}\hat{\mathbf{R}} & \mathbf{R}\hat{\mathbf{p}} + \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \mathbf{I}_{4\times 4} \tag{27}$$

Naturally leading to:

- $\mathbf{R}\hat{\mathbf{R}} = \mathbf{I}_{3\times 3} \Rightarrow \hat{\mathbf{R}} = \mathbf{R}^{-1} = \mathbf{R}^{\top}$
- $\mathbf{R}\hat{\mathbf{p}} + \mathbf{p} = \mathbf{0}_{3\times 1} \Rightarrow \hat{\mathbf{p}} = -\mathbf{R}^{\top}\mathbf{p}$

2 Homework

a Exercise 1: Homogeneous Transformations

In this exercise, you need to obtain the different Homogeneous Transformations to get the virtual world as showed in Fig. 9. Put your results in the matlab file "`Exercise_1.m`" provided in the template. **Please do not submit handwritten content !**

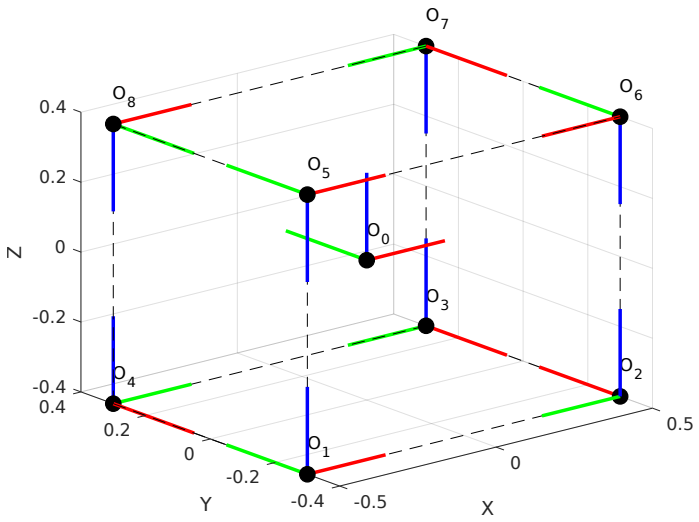


Figure 9: Expected coordinate frames after defining the Homogeneous Transformations.Red axis x, Blue axis z and Green axis y.

1. By simple inspection obtain manually the absolute homogeneous transformations $\mathbf{H}_0^i$ ($i = 1, 2, 3, ..., 8$), for each corner of the cage with respect to the coordinate frame O_0, see figure 9.
2. The rotation matrix of some frames in Fig. 9 can not be obtained using a sequence of basic rotations (Composition of rotations, see Fig. (4-5). Please explain why (elaborate your answer using for example the definition of **SE**(3)), and propose the proper coordinate frames to fix this problem.
3. Replace the coordinate frames found in the above question with the suitable coordinate frames, respecting the figure convention; then compute the associated absolute homogeneous transformations $\mathbf{H}_0^i$ ($i = 1, 2, 3, ..., 8$).
4. Extract the rotation matrices $\mathbf{R}_0^i$ and translation vectors $\mathbf{p}_0^i$ from each $\mathbf{H}_0^i$ ($i = 1, 2, 3, ..., 8$).
5. Represent each rotation matrix $\mathbf{R}_0^i$ as a set of Euler angles (*roll-pitch-yaw*) using the eq.(13).
6. Compute manually the relative homogeneous transformations $\mathbf{H}_{i-1}^i$ ($i = 1, 2, 3, ..., 8$), i.e. the relative transformations between the corners of the cage.
7. Compute again the absolute homogeneous transformations $\mathbf{H}_0^i$ of each coordinate frame using the relative transformations $\mathbf{H}_{i-1}^i$ and compare them with the absolute transformations $\mathbf{H}_0^i$, computed manually in the first item.

8. Plot each coordinate frame in matlab to visualize the cage. Use the "`plotCage`" function as a template. Three graphs are expected, for questions 1, 3, and 7 respectively.

b Exercise 2: Homogeneous Transformations – Relative vs Global

Consider the regular pentagon depicted in Fig. 10-(a). Each vertex $i = [1 \cdots 5]$ of this pentagon is associated with a specific coordinate frame, denoted as $(O_i x_i y_i z_i)$. The frame $(O_0 x_0 y_0 z_0)$ here denotes the base frame, and is located in the geometric center of the pentagon. For each question below, put your results in the matlab file "`Exercise_2.m`" provided in the template. **Please do not submit handwritten content !**

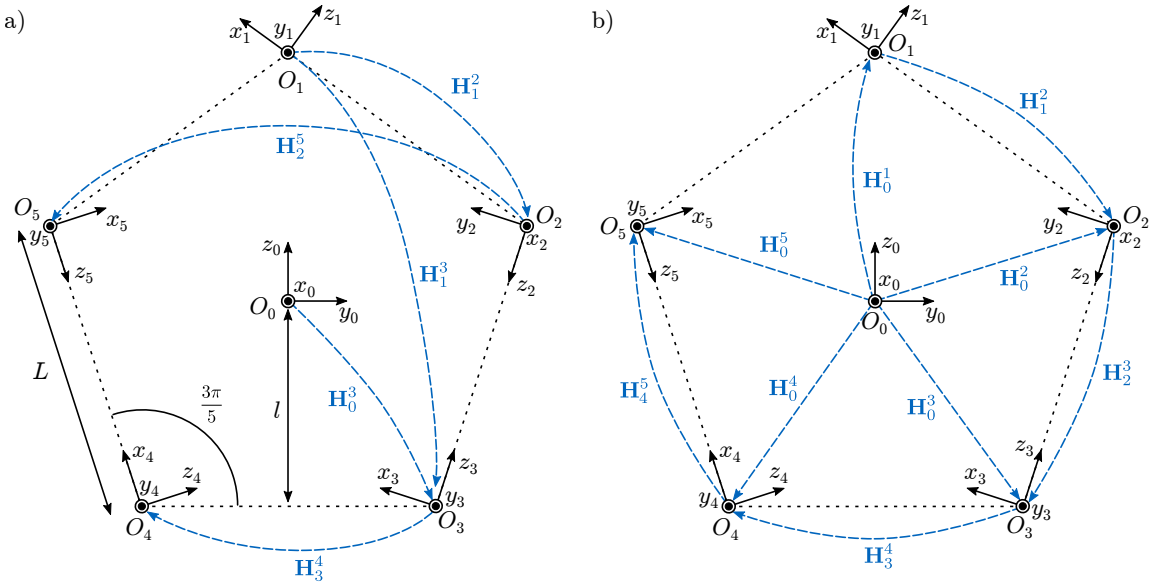


Figure 10: **(a)**: A Regular Pentagon. The transforms $\mathbf{H}_0^3$, $\mathbf{H}_3^4$, $\mathbf{H}_1^3$, $\mathbf{H}_2^5$ and $\mathbf{H}_1^2$ are already provided in the template. **(b)**: The homogeneous transformations depicted in this figure must be computed in this exercise.

1. Using the provided homogeneous transformations $\mathbf{H}_0^3$, $\mathbf{H}_3^4$, $\mathbf{H}_1^3$, $\mathbf{H}_2^5$ and $\mathbf{H}_1^2$, compute on matlab the absolute $\mathbf{H}_0^i$ associated with each frame $(O_i x_i y_i z_i)$ of the pentagon, see Fig. 10. Each $\mathbf{H}_0^i$ must be written as a sequence of the provided transformations. Write this sequence as a comment in your code. Plot each coordinate frame in matlab to visualize the pentagon as showed in figure 11a. Use the "`plotPent`" function as a template.
2. Compute the remaining relative homogeneous transformations $\mathbf{H}_4^5$ and $\mathbf{H}_2^3$ of the pentagon.
3. Compute $\mathbf{H}_1^5$, $\mathbf{H}_5^3$, and $\mathbf{H}_2^4$, using – for each – at least three different sequences of homogeneous transforms.
4. Transform the whole pentagon including all its coordinate frames, with respect to the frame $(O_0 x_0 y_0 z_0)$, following the next composite homogeneous transformation based on the provided sequences:

(a) Sequence 1: $\mathbf{H}_{S_1}$

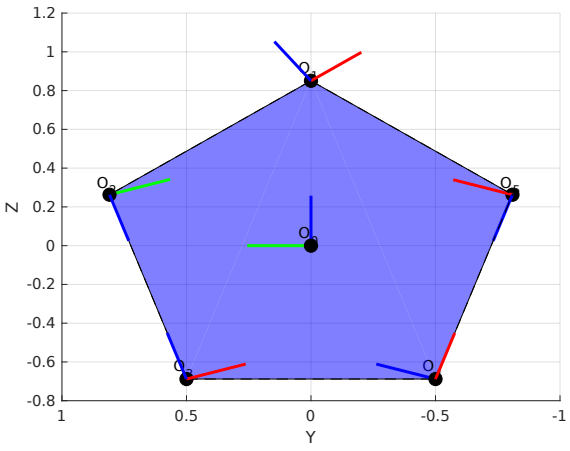
- Rotate about the x_0-axis by 90^o;
- Rotate about the new y_0'-axis by 45^o;
- Rotate about the new z_0''-axis by 30^o.
- Translate about the x_0-axis by $2m$.
- Translate about the y_0-axis by $0.75m$.
- Translate about the z_0-axis by $0.75m$.

(b) Sequence 2: $\mathbf{H}_{S_2}$

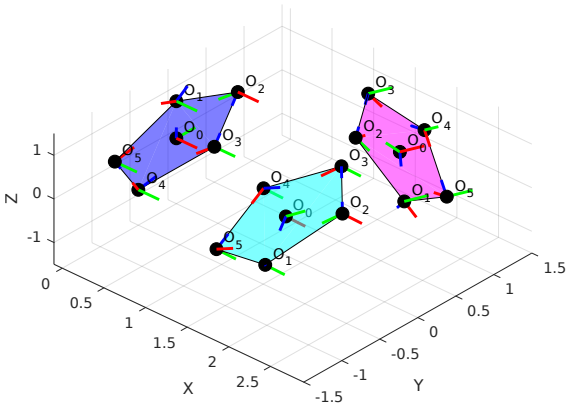
- Rotate about the x_0-axis by 90^o;
- Rotate about the y_0-axis by 45^o;
- Rotate about the z_0-axis by 30^o.
- Translate about the x_0-axis by $2m$.
- Translate about the y_0-axis by $-0.75m$.
- Translate about the z_0-axis by $0.75m$.

For each sequence, plot the corresponding coordinate frames in matlab, in order to visualize both the original and the transformed pentagons, as proposed in figure 11b. Explain the differences between the two sequences. Are the aspect ratio of the different pentagons preserved after applying the considered homogeneous transforms ?

5. Consider the rotation sequence 2 of question 4 ($\mathbf{H}_{S_2}$): what happens when you **replace** the last row of $\mathbf{H}_{S_2}$ by the vector $[a, b, c, d]$, where $a, b, c, d \in \mathbb{R}$? Use different values for a, b, c, d and plot the corresponding results for the pentagon in a set of new figures. In this case only plot the vertex without coordinate frames (i.e. only use the position vector of each vertex.). **Important note:** Remember that in order to plot 3d-points (vertex) using the transformation between them, these points must be homogeneous.



(a) Expected result for question 1: a regular pentagon.



(b) Expected result for question 4: in blue, the original pentagon, in magenta the pentagon after sequence 1 and in cyan after sequence 2.

Figure 11: Expected results