1st Homework IR

Ioannis Korogiannos

1100901

korogiannos.ioannis@ac.upatras.gr

github.com/korojohn/IntroRobotics

October 2025

Contents

1 Description

This report focuses on the determination of transformation matrices associated with the problem of object manipulation by a robotic system. The main objective is to analyze and define the mathematical relationships between different coordinate frames involved in the robot's operation—such as the base, joints, end-effector, and object frames.

2 Setup

For the solution, we need an appropriate setup. Firstly, we will use NumPy, a Python library that provides efficient tools for numerical computations and matrix operations. It will be used to define and manipulate transformation matrices, perform rotations and translations, and handle coordinate data efficiently.
We will also use Matplotlib, a visualization library that allows us to plot data and represent transformations graphically. Also, we can create 3D plots to visualize coordinate frames and object positions in space.
Additionally, we import patches, Rectangle, and Poly3DCollection from Matplotlib, which help us draw 2D and 3D geometric shapes, making the visualization of robotic systems and transformations clearer.
Here is the basic setup code, for the libraries.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import Rectangle
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
```

Code Block 1: "Libraries Setup"

3 Tasks

3.1 Task 1

In Task 1, I define the position and orientation of the World Frame.

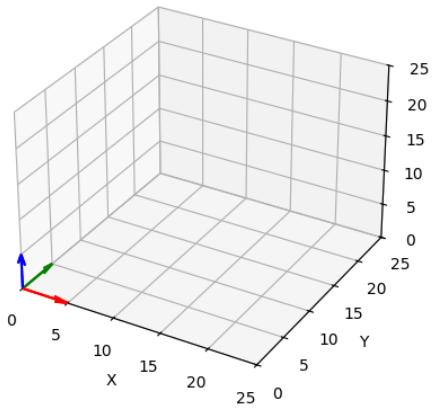

Figure 1: The World Frame $\{S\}$ at the origin $(0, 0, 0)$.

As shown in Figure 1, the world frame $\{S\}$ is placed at the origin of the Cartesian coordinate system, that is at point $(0, 0, 0)$. This choice simplifies the definition of all subsequent coordinate frames, since it serves as the global reference for both position and orientation in the workspace.

3.2 Task 2

In Task 2, we define a fixed Body Frame for each object, as also illustrated in Figure 2. Specifically, we define frame `Tbody1` (or $\{B1\}$) for the blue 'tool' cube and `Tbody2` (or $\{B2\}$) for the red 'target' box.

The Python code that implements these definitions is shown in Code Block 2. We use a helper function `homogeneous`, which combines a rotation matrix (e.g., from `RotZ`) and a translation vector into a 4×4 homogeneous transformation matrix.

```
# Tbody1: Frame for the blue box (center)
Tbody1 = homogeneous(RotZ(0), np.array([[17.5, 2.5, 2.5]]).T
    )

# Tbody2: Frame at the center of the red box (NOT rotated)
Tbody2_translation = np.array([[10., 12.5, 10]]).T
Tbody2 = homogeneous(RotY(0), Tbody2_translation)
```

Code Block 2: "Body Frame Definitions"

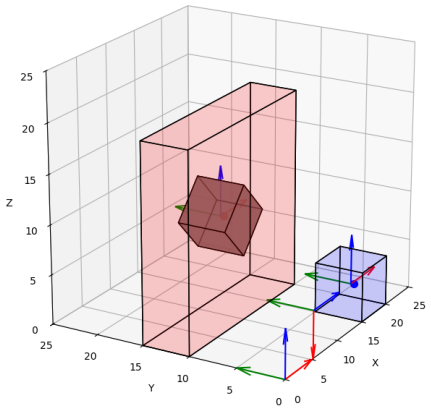

Figure 2: World Frame $\{S\}$ and Body Frames $\{B1\}$ (blue cube) and $\{B2\}$ (red box).

Analyzing the Code Block 2:

- **Tbody1 (Blue Box):** This frame is placed at the geometric center of the blue cube. Its origin's coordinates relative to the World Frame $\{S\}$ are **(17.5, 2.5, 2.5)**.
- **Tbody2 (Red Box):** This frame is placed at the geometric center of the red target box. Its origin's coordinates relative to the World Frame $\{S\}$ are **(10, 12.5, 10)**.

3.3 Task 3

Now, I will calculate the transformation matrix for each object, relative to the world frame $\{S\}$.

A homogeneous transformation matrix T combines rotation R (3×3) and translation p (3×1) into a single 4×4 matrix. This allows us to represent a complete change of frame (both position and orientation) with a single matrix multiplication.

The general form of a homogeneous transformation matrix T is:

$$T = \begin{bmatrix} R & p \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where R is the rotation matrix and p is the translation vector.

3.3.1 Object 1: Blue Cube (Tbody1)

As defined in Task 2, the blue cube's frame $\{B1\}$ is located at $(17.5, 2.5, 2.5)$ with zero rotation relative to the world frame.

The code to generate this matrix is (from Code Block 2):

```
Tbody1 = homogeneous(RotZ(0),np.array([[17.5, 2.5, 2.5]]).T)
```

Code Block 3: "Transformation for Blue Cube"

Here, the rotation matrix R is the identity matrix $I_{3\times3}$ (since $RotZ(0) = I$), and the translation vector is $p = [17.5, 2.5, 2.5]^T$.

The resulting transformation matrix, $T_{S\to B1}$, is:

$$Tbody1 = \begin{bmatrix} 1 & 0 & 0 & 17.5 \\ 0 & 1 & 0 & 2.5 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.3.2 Object 2: Red Box (Tbody2)

Similarly, the frame for the red box $\{B2\}$ is located at its geometric center $(10, 12.5, 10)$ and is also defined with zero rotation relative to the world frame.

The code snippet for this transformation is (from Code Block 2):

```
Tbody2_translation = np.array([[10., 12.5, 10]]).T
Tbody2 = homogeneous(RotY(0), Tbody2_translation)
```

Code Block 4: "Transformation for Red Box"

Again, the rotation matrix R is the identity matrix $I_{3\times3}$ (since $RotY(0) = I$), and the translation $p = [10, 12.5, 10]^T$.

The resulting transformation matrix, $T_{S\to B2}$, is:

$$Tbody2 = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 12.5 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.4 Task 4

In this task, we must select an appropriate transformation matrix for the base frame of the manipulator, let's call it $\{M\}$, with respect to the world frame $\{S\}$. The key requirement is that the manipulator must be able to perform the manipulation task, which involves accessing both the blue cube (at $\{B1\}$) and the red box (at $\{B2\}$).

3.4.1 Selection and Justification

For maximum simplicity and to ensure a clear, unobstructed workspace, we will select the most fundamental frame of reference as our manipulator's base: **the World Frame origin**.

This strategic choice means:

- The manipulator's base frame $\{M\}$ is perfectly aligned with the world frame $\{S\}$.
- The origin of the manipulator is at $(0, 0, 0)$.
- The manipulator's X, Y, and Z axes are collinear with the world's axes.

3.4.2 Transformation Matrix ($T_{S \to M}$)

To define this transformation in Python, we can simply use the `numpy.eye()` function, which generates an identity matrix.

```
# T_manipulator: Frame for the Manipulator Base {M}
# We place it at the world origin (0,0,0) with no rotation.
T_manipulator = np.eye(4)
```

Code Block 5: "Manipulator Base Frame Definition"

This corresponds to a transformation with:

- A rotation matrix $R = I_{3\times3}$ (the 3×3 identity matrix).
- A translation vector $p = [0, 0, 0]^T$.

The resulting homogeneous transformation matrix, $T_{S \to M}$, is the 4×4 identity matrix:

$$T_{S \to M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.5 Task 5

This task asks for the relative transformation matrix between the manipulator's end-effector, frame $\{E\}$, and the cube, frame $\{B1\}$, at the moment of grasp. This is the matrix $T_{E\to B1}$, which defines the pose of the cube *with respect to* the robot's gripper.

3.5.1 Defining the Grasp Frame

We will define the grasp based on the following assumptions:

- **Position:** The end-effector $\{E\}$ grasps the cube on its $x_S = 15$ face. Given that the cube's center $\{B1\}$ is at $x_S = 17.5$ (Task 3) and the cube has a side length of 5cm, this is the face at $x_{B1} = -2.5$ (relative to the cube's center).
- As requested, the grasp is centered in 'y and z', so the relative position of $\{E\}$ with respect to $\{B1\}$ is $p_{B1\to E} = [-2.5, 0, 0]^T$.
- **Orientation:** We assume the simplest possible grasp, where the end-effector's axes $\{E\}$ are aligned with the cube's axes $\{B1\}$. This means the relative rotation matrix $R_{B1\to E}$ is the 3×3 identity matrix I.

This assumption ($R_{B1\to E} = I$) is consistent with the description of the final motion, where the end-effector rotates "about the Y-axis (of the world frame) by 45 degrees", exactly like the cube (Task 6).

3.5.2 Visualizing the End-Effector Grasp

Here is how the end-effector is positioned when grasping the cube.

3.5.3 Calculation

First, we define the transformation from the cube $\{B1\}$ to the end-effector $\{E\}$:

$$T_{B1\to E} = \begin{bmatrix} R_{B1\to E} & p_{B1\to E} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -2.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrix we need is the inverse, $T_{E\to B1} = (T_{B1\to E})^{-1}$. Using the formula $T^{-1} = \begin{bmatrix} R^T & -R^T p \\ \mathbf{0} & 1 \end{bmatrix}$:

- $R^T = I^T = I$
- $-R^T p = -(I)p = -p = -[-2.5, 0, 0]^T = [2.5, 0, 0]^T$

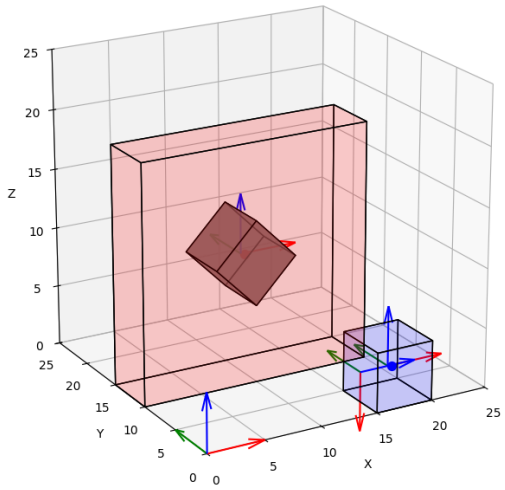

Figure 3: The end-effector $\{E\}$ grasping the blue cube. The end-effector's frame is aligned with the cube's frame, with its origin shifted to the grasp point on the cube's face.

The required relative transformation is:

$$T_{E\to B1} = \begin{bmatrix} 1 & 0 & 0 & 2.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This means that the cube's center $\{B1\}$ is always 2.5 cm in front of the end-effector's origin $\{E\}$, along its x_E axis.

3.6 Task 6

In this task, we calculate the cube's world-frame transformation matrix for two distinct poses:

1. The "pre-grasp" pose: Positioned right next to the hole, ready to be inserted.
2. The "final" pose: Positioned precisely inside the hole.

3.6.1 Part 1: Positioned Next to the Hole (Pre-Grasp Pose)

To be "right next to the hole," the cube must be aligned with the hole's center in the X and Z dimensions, and also share the same 45° rotation around the Y-axis. The hole itself is centered at $(10, 12.5, 10)$.

The red box's front face is at $Y = 10$. We therefore define our "pre-grasp" pose to be slightly in front of this face, at $Y = 7.5$, while matching the hole's other coordinates.

- **Translation**: $p = [10, 7.5, 10]^T$
- **Rotation**: $R = R_y(45°)$

This is exactly what is calculated in the provided Python code under the variable `Tbody1`:

```
Tbody1_translation = np.array([[10., 7.5, 10.]]).T
Tbody1_rotation = RotY(np.deg2rad(45))
Tbody1 = homogeneous(Tbody1_rotation, Tbody1_translation)
```

Code Block 6: "Pre-Grasp Pose (Tbody1)"

The resulting transformation matrix, $T_{pre-grasp}$ (which is `Tbody1`), is:

$$T_{pre-grasp} = \begin{bmatrix} \cos(45°) & 0 & \sin(45°) & 10 \\ 0 & 1 & 0 & 7.5 \\ -\sin(45°) & 0 & \cos(45°) & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 0.707 & 0 & 0.707 & 10 \\ 0 & 1 & 0 & 7.5 \\ -0.707 & 0 & 0.707 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

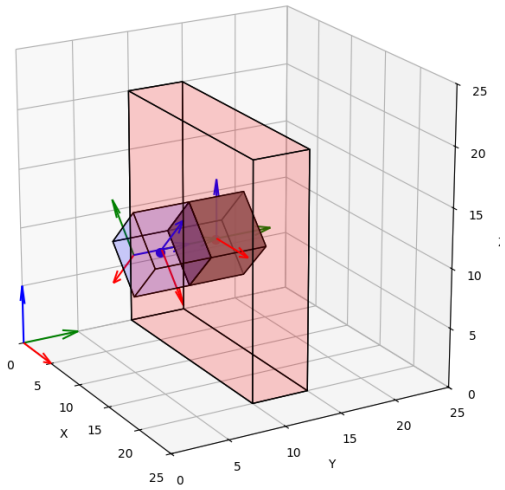

Figure 4: The blue cube positioned in its pre-grasp pose, right next to the red box's hole, ready for insertion.

3.6.2 Part 2: Positioned Inside the Box (Final Pose)

The second pose, "precisely positioned inside the box," is the final target pose for the cube. This pose is identical to the pose of the rhombus-shaped hole itself, which is centered at $(10, 12.5, 10)$ and also rotated by 45° around the Y-axis.

- **Translation**: $p = [10, 12.5, 10]^T$
- **Rotation**: $R = R_y(45°)$

This matrix is defined in the code as `T_rhombus`:

```
Tbody2_translation = np.array([[10., 12.5, 10]]).T
T_rhombus = homogeneous(RotY(np.deg2rad(45)),
    Tbody2_translation)
```

Code Block 7: "Final Pose (T_rhombus)"

The resulting transformation matrix, T_{final} (which is `T_rhombus`), is:

$$T_{final} = \begin{bmatrix} \cos(45°) & 0 & \sin(45°) & 10 \\ 0 & 1 & 0 & 12.5 \\ -\sin(45°) & 0 & \cos(45°) & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 0.707 & 0 & 0.707 & 10 \\ 0 & 1 & 0 & 12.5 \\ -0.707 & 0 & 0.707 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

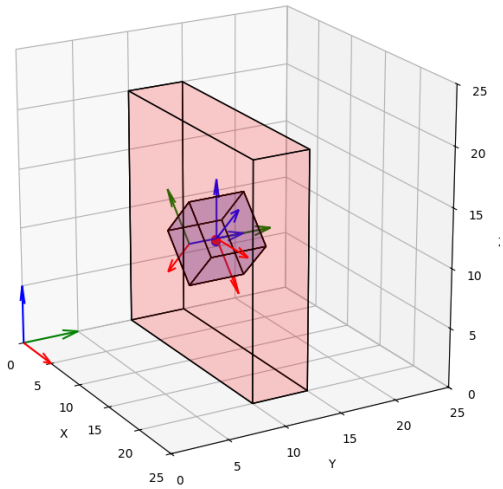

Figure 5: The blue cube precisely positioned inside the red box's rhombus-shaped hole, indicating the final successful placement.