2nd Homework IR

Ioannis Korogiannos
1100901
korogiannos.ioannis@ac.upatras.gr
github.com/korojohn/IntroRobotics

December 2025

Contents

1 Task 1: World Frame and Body Frames

1.1 World Frame {S}

The world frame is defined at the global origin:

$$\text{Origin}_S = (0, 0, 0)$$

1.2 Body Frames

We define fixed body frames for the objects relative to {S}.

Blue Cube (Frame {B}) The frame is located at the center of the blue cube:

$$p_{S\to B} = [17.5,\ 2.5,\ 2.5]^T$$

$$T_{S\to B} = \begin{bmatrix} 1 & 0 & 0 & 17.5 \\ 0 & 1 & 0 & 2.5 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Red Box (Frame {R}) The frame is located at the center of the red box:

$$p_{S\to R} = [10,\ 12.5,\ 10]^T$$

$$T_{S\to R} = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 12.5 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2 Task 2: Manipulator Base Frame Selection

To simplify the workspace analysis, the manipulator base frame {M} is chosen to coincide with the World Frame {S}.

$$T_{S\to M} = I_{4\times 4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3 Task 3: End-Effector Grasp Strategy & Matrix

3.1 Grasp Position Strategy (Top Grasp)

To successfully grasp the cube, we define the target position for the end-effector (p_E) at the **center of the top surface** of the blue cube. Given that the cube's center is at $x = 17.5$,at $y = 2.5$ cm and at $z = 5.0$ cm.

$$p_E = [17.5,\ 2.5,\ 5]^T$$

We select this top-down approach position because, with this placement, we do not exceed our Joint Limits (shown in Table 1) and the Newton-Raphson inverse kinematics algorithm converges efficiently in just 8 iterations(shown in Figure 2.

Joints	**Min angle (°)**	**Max angle (°)**
Joint 1	-160	160
Joint 2	-70	115
Joint 3	-170	170
Joint 4	-113	75
Joint 5	-170	170
Joint 6	-115	115
Joint 7	-180	180

Table 1: Joint limits (in degrees) for the Elephant myArm manipulator.

3.2 Relative Transformation Matrix ($T_{E\to B}$)

Based on the top grasp strategy defined above, we determine the relative transformation matrix between the end-effector {E} and the cube {B}.

Since the gripper is positioned at the top surface ($z = 5$) and the cube center is at ($z = 2.5$), the cube center is located 2.5 cm **below** the gripper (along the gripper's Z-axis).

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

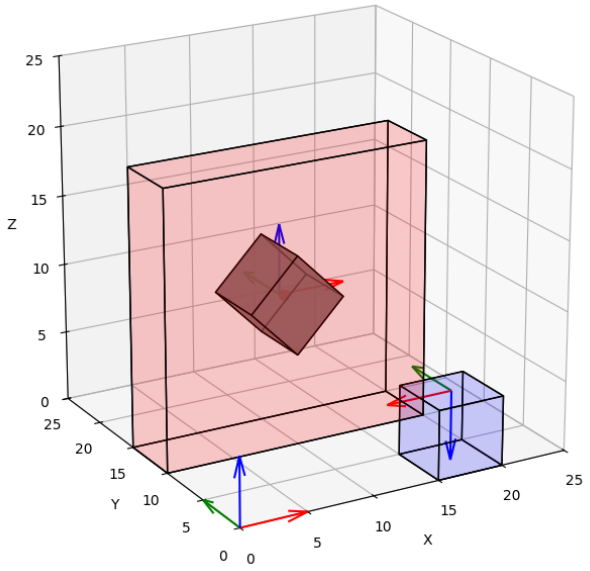

Figure 1: End-Effector Visual - Top Grasp Position

4 Task 4: Target Poses (Pre-Grasp & Final)

We calculate the cube's world-frame transformation matrix for two scenarios involving the rhombus-shaped hole in the red box. The hole is rotated by 45° around the Y-axis.

4.1 Part 1: Pre-Grasp Pose (Next to Hole)

The cube is aligned with the hole but positioned slightly in front of it (at $Y = 7.5$) to prepare for insertion.

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

$$T_{\text{pre-grasp}} \approx \begin{bmatrix} 0.707 & 0 & 0.707 & 10 \\ 0 & 1 & 0 & 7.5 \\ -0.707 & 0 & 0.707 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2 Part 2: Final Pose (Inside the Hole)

The cube is precisely positioned inside the box.

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

$$T_{\text{final}} \approx \begin{bmatrix} 0.707 & 0 & 0.707 & 10 \\ 0 & 1 & 0 & 12.5 \\ -0.707 & 0 & 0.707 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5 Task 5: Inverse Kinematics for Grasping

To calculate the joint positions required to grasp the cube, we employed the **Newton-Raphson** numerical inverse kinematics algorithm.

5.1 Kinematic Model Parameters

Based on the kinematic structure of the Elephant myArm 300 Pi, we defined the geometric parameters required for the Product of Exponentials (PoE) formula. The dimensions were converted from millimeters to meters.

Link Lengths (Size) The vertical offsets between the joints are defined as:

$$d_1 = 0.1695\,\text{m}$$
$$d_3 = 0.1155\,\text{m}$$
$$d_5 = 0.1278\,\text{m}$$
$$d_7 = 0.0660\,\text{m}$$

Screw Axes (Twist Configuration) For each joint $i = 1 \dots 7$, we define the rotation axis unit vector ω_i and a point q_i on that axis relative to the base frame $\{S\}$ in the zero configuration:

- **Joint 1:** Vertical rotation at base.

$$\omega_1 = [0, 0, 1]^T, \quad q_1 = [0, 0, 0]^T$$

- **Joint 2:** Horizontal rotation (shoulder).

$$\omega_2 = [0, 1, 0]^T, \quad q_2 = [0, 0, d_1]^T$$

- **Joint 3:** Vertical rotation.

$$\omega_3 = [0, 0, 1]^T, \quad q_3 = [0, 0, d_1 + d_3]^T$$

- **Joint 4:** Horizontal rotation (elbow).

$$\omega_4 = [1, 0, 0]^T, \quad q_4 = [0, 0, d_1 + d_3]^T$$

- **Joint 5:** Vertical rotation.

$$\omega_5 = [0, 0, 1]^T, \quad q_5 = [0, 0, d_1 + d_3 + d_5]^T$$

- **Joint 6:** Horizontal rotation (wrist).

$$\omega_6 = [1, 0, 0]^T, \quad q_6 = [0, 0, d_1 + d_3 + d_5]^T$$

- **Joint 7:** Vertical rotation (end-effector flange).

$$\omega_7 = [0, 0, 1]^T, \quad q_7 = [0, 0, d_1 + d_3 + d_5 + d_7]^T$$

5.1.1 Matrix Exponential

The term $e^{[\mathcal{S}]\theta}$ is calculated using the Rodrigues' formula extended to $SE(3)$. For a screw axis $\mathcal{S} = (\omega, v)$ and angle θ:

1. **Rotation** ($R \in SO(3)$):

$$R = I + \sin\theta[\omega] + (1 - \cos\theta)[\omega]^2$$

2. **Translation** ($p \in \mathbb{R}^3$):

$$p = \left(I\theta + (1 - \cos\theta)[\omega] + (\theta - \sin\theta)[\omega]^2\right) v$$

The resulting transformation matrix is:

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

5.1.2 Space Jacobian Calculation

The Space Jacobian $J_s(\theta) \in \mathbb{R}^{6\times 7}$ relates joint velocity to end-effector twist. It is computed column-by-column using the Adjoint mapping of the cumulative transformation:

$$J_s(\theta) = [\mathcal{S}_1 \mid \mathrm{Ad}_{T_1}(\mathcal{S}_2) \mid \cdots \mid \mathrm{Ad}_{T_6}(\mathcal{S}_7)] \tag{1}$$

Where T_i represents the transformation matrix accumulated up to joint i:

$$T_i = e^{[\mathcal{S}_1]\theta_1} \dots e^{[\mathcal{S}_i]\theta_i}$$

This iterative approach corresponds to the loop structure found in the `space_jacobian` Python function.

5.2 Methodology

The algorithm iteratively updates the joint angles vector θ to minimize the twist error $\mathcal{V}$ between the current end-effector pose $T_{sb}(\theta)$ and the desired grasp pose T_{sd}.

- **Target Pose:** Defined in Task 3 as the "Top Grasp" position ($x = 17.5, y = 2.5, z = 5.0$ cm) with the gripper pointing downwards ($R_x(180°)$).
- **Initial Guess:** $\theta_{init} = [0°, 30°, 0°, -45°, 0°, -45°, 0°]^T$.
- **Convergence Criteria:** $||\mathcal{V}|| < 10^{-4}$.

5.3 Results

The solver successfully converged after **8 iterations**.

Joint	Angle (°)
θ_1	-12.73
θ_2	68.82
θ_3	-60.13
θ_4	-81.36
θ_5	-44.14
θ_6	-41.83
θ_7	-55.14

Table 2: Calculated Joint Angles for Grasping

5.4 Verification and Implementation

The Forward Kinematics yielded a position error of 5.91×10^{-9} m.

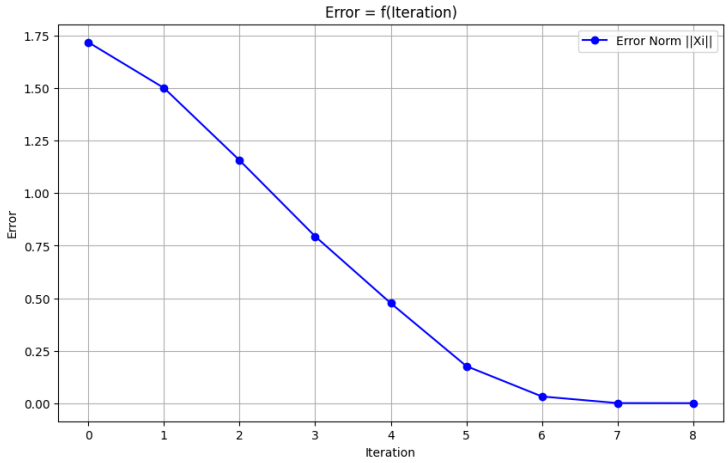

Figure 2: Algorithm convergence in 8 iterations

Code Block 1: "Newton-Raphson Inverse Kinematics Solver"

```
def inverse_kinematics_space_basic(T_target, q0, M,
            joint_limits,max_iters=50, tol=1e-4, alpha=1.0):
    q = np.array(q0, dtype=float).copy()
    error_history = []
    for k in range(max_iters):
        # 1. Compute current pose and error twist
        T_current = fk_poe(S_list, M, q)
        T_diff = T_target @ np.linalg.inv(T_current)
```

```
        Xi = se3_log(T_diff)
        err_norm = np.linalg.norm(Xi)
        error_history.append(err_norm)
        # 2. Check convergence
        if err_norm < tol:
            return q, {"converged": True, "iter": k, "err":
                error_history}
        # 3. Compute Jacobian
        J_s = space_jacobian(q, S_list, M)

        # 4. Compute change in joints (dq) using Pseudo-
            Inverse
        try:
            dq = np.linalg.pinv(J_s) @ Xi
        except np.linalg.LinAlgError:
            dq = np.linalg.lstsq(J_s, Xi, rcond=None)[0]

        # 5. Limit step size for stability
        if np.linalg.norm(dq) > 0.5:
            dq = dq * (0.5 / np.linalg.norm(dq))

        # 6. Project to Joint Limits
        dq = project_to_joint_limits(q, dq, joint_limits)

        # 7. Update Joints
        q = q + alpha * dq
    return q, {"converged": False, "iter": max_iters, "err":
        error_history}
```

6 Task 6: Placing Position (Approach Hole)

6.1 Placing Pose Strategy

To place the cube "right next to the hole," we defined a target pose T_{place} that aligns the end-effector with the diamond-shaped hole on the red box while maintaining a safe standoff distance.

Orientation (R_{place}) The orientation requires a horizontal approach (Pitch 90°) and a roll to match the diamond shape (Roll 45°):

$$R_{\text{place}} = R_y(90^\circ)R_x(45^\circ) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 45^\circ & -\sin 45^\circ \\ 0 & \sin 45^\circ & \cos 45^\circ \end{bmatrix} \approx \begin{bmatrix} 0 & 0.7071 & 0.7071 \\ 0 & 0.7071 & -0.7071 \\ -1 & 0 & 0 \end{bmatrix}$$

6.2 Calculated Target Matrix

Combining rotation and translation, the target transformation matrix is:

$$T_{\text{place}} = \begin{bmatrix} 0.0000 & 0.7071 & 0.7071 & 0.1050 \\ 0.0000 & 0.7071 & -0.7071 & 0.0750 \\ -1.0000 & 0.0000 & 0.0000 & 0.1000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

6.3 Inverse Kinematics Results (Final Thetas)

Using IK ,the algorithm converged successfully in **6 iterations**.

Joint	**Angle (°)**
θ_1	46.83
θ_2	64.76
θ_3	-70.94
θ_4	-103.01
θ_5	-83.66
θ_6	-108.91
θ_7	-100.89

Table 3: Final Joint Angles for Placing Position

Verification The Forward Kinematics (FK) calculation verifies that the solution aligns perfectly with the desired placing coordinates. As seen in the solver output:

The resulting position error is negligible:

$$\text{Error} = ||p_{\text{target}} - p_{\text{actual}}|| = 0.00000\,\text{m}$$

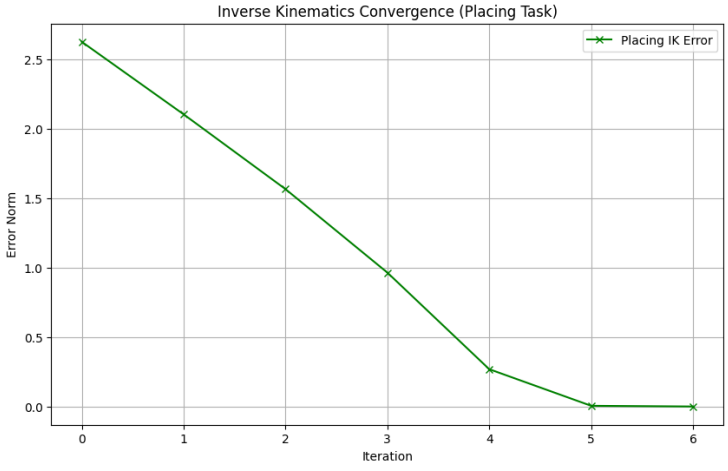

Figure 3: Error convergence vs Iterations for the Placing Task

7 Task 7: Visualization

In this section, we present the visualization of the calculated robot configurations using the Python environment.

7.1 Grasping Visualization

Figure 4 illustrates the manipulator successfully reaching the "Top Grasp" position calculated in Task 5. The end-effector is correctly aligned with the center of the blue cube ($z = 5$ cm) with the gripper pointing downwards.

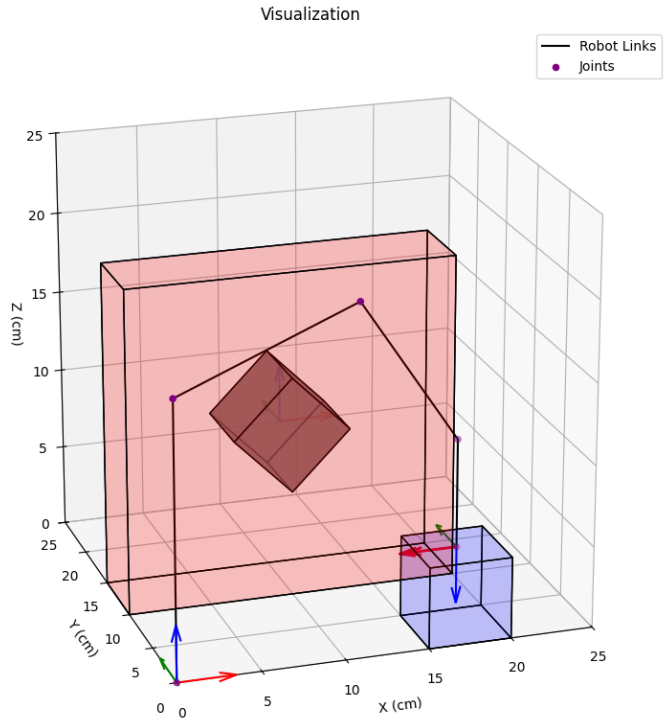

Figure 4: Visualization of the Elephant myArm manipulator grasping the blue cube.

7.2 PS on Placing Visualization

Regarding the "Placing" configuration (Task 6), please note that while the inverse kinematics solution was successfully computed and verified numerically (with zero position error), it was not possible to generate a visual rendering for the specific pose where the cube is positioned right next to the hole.