UR5 Robot Control Methods: Inverse Kinematics, Resolved-Rate, and Jacobian-Transpose

Sungwoo Kim, Andrew Ying, Suhash Reddy

1 Start & Target Locations

As per the assignment instructions, the script `ur5_project.m` invokes three methods of control (Inverse Kinematics, Resolved-Rate, and Jacobian-Transpose) for a UR5 robot to perform a place-and-draw task. These methods are implemented in `ur5_IKcontrol.m`, `ur5_RRcontrol.m`, and `ur5_JTcontrol.m`. Really, Resolved-Rate, and Jacobian-Transpose control are implemented in `ur5_RRJTcontrol.m` since their logic is mostly shared and `ur5_RRcontrol.m` and `ur5_JTcontrol.m` call this function.

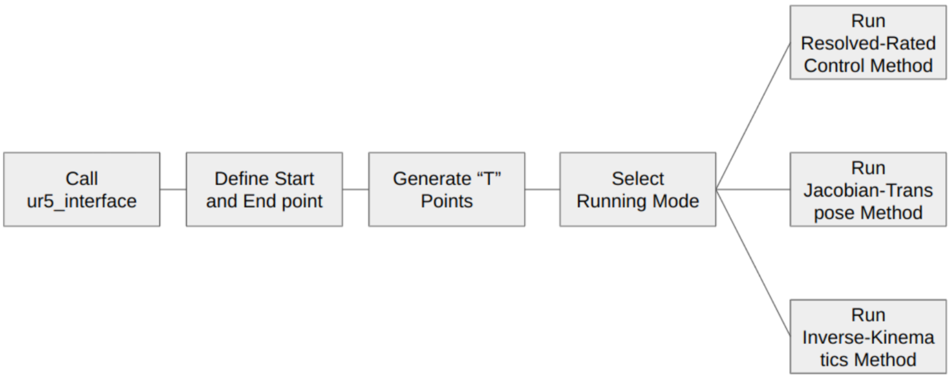

Figure 1: Flow Chart of the ur5_project.m function

At the beginning of `ur5_project.m`, the start and target locations are taught using pendant control. The user first switches to pendant control using the command `ur5.switch_to_pendant_control()`, followed by `waitforbuttonpress` to pause the program, allowing the user to manually position the robot at the desired location. Once the position is set, control is returned to ROS using `ur5.switch_to_ros_control()`, and the joint configuration is recorded using `ur5.get_current_joints()`. Forward kinematics (`ur5FwdKin`) is then used to compute the start pose for the tool frame. This process is repeated to define the target pose for the tool frame.

To generate the intermediate poses required to draw the letter "T," the direction vector from the x-y components of the start point (1) to the target point (4) is calculated as:

Compute the middle point where the second line will be drawn:

$$\vec{p}_{middle} = (\vec{p}_{start} + \vec{p}_{end})/2 \tag{1}$$

Travel to the point just above the midpoint:

$$\vec{p}_{middle\_up} = \vec{p}_{middle} + \begin{bmatrix} 0 \\ 0 \\ 0.02 \end{bmatrix} \tag{2}$$

Then set the pen down to the mid point.

Compute the projection and rejection to get the direction orthogonal to the line from the start position to the end, towards the base link origin. Give rejection the same length as the first line:

$$\vec{p}_{origin\_up} = \begin{bmatrix} 0 \\ 0 \\ p_{start,z} \end{bmatrix} \tag{3}$$

$$\vec{v}_{projection} = \frac{(\vec{p}_{end} - \vec{p}_{start}) \cdot (\vec{p}_{origin\_up} - \vec{p}_{start})}{\|\vec{p}_{end} - \vec{p}_{start}\|^2} (\vec{p}_{end} - \vec{p}_{start}) \tag{4}$$

$$\vec{p}_{projection} = \vec{v}_{projection} + \vec{p}_{start} \tag{5}$$

$$\vec{v}_{rejection} = \frac{\vec{p}_{origin\_up} - \vec{p}_{projection}}{\left\|\vec{p}_{origin\_up} - \vec{p}_{projection}\right\|} * \|\vec{p}_{end} - \vec{p}_{start}\| \tag{6}$$

Compute the end position for drawing the vertical line:

$$\vec{p}_{vertical\_end} = \vec{p}_{middle} + \vec{v}_{rejection} \tag{7}$$

Finally, compute the position lifted above the vertical end:

$$\vec{p}_{vertical\_end\_up} = \vec{p}_{vertical\_end} + \begin{bmatrix} 0 \\ 0 \\ 0.02 \end{bmatrix} \tag{8}$$

Each of these intermediate poses shares the orientation of the start pose:

$$\boldsymbol{R}_{middle} = \boldsymbol{R}_{middle\_up} = \boldsymbol{R}_{vertical\_end} = \boldsymbol{R}_{vertical\_end\_up} = \boldsymbol{R}_{start} \tag{9}$$

The order of the poses is $g_{start} \to g_{end} \to g_{middle\_up} \to g_{middle} \to g_{vertical\_end} \to g_{vertical\_end\_up}$

We included intermediate hover and lift points in the trajectory such that before transitioning between major points, the robot first lifted the pen in the z-direction to prevent unintentional drawing, then moved laterally to the next point, and finally lowered the pen to resume drawing. This procedure ensured clean and accurate trajectory execution, especially when transitioning between the horizontal and vertical segments of the letter "T."

Once all points are defined, the user selects the control method by inputting "IK," "RR," or "JT" into the script, initiating the respective trajectory execution. The rotational and positional errors are printed to the console as well as messages if singularities or out of safety limit errors occur.

2 Control Methods

2.1 Forward Kinematics using D-H Parameters

The inverse kinematics implementation relies on a correct formulation of the forward kinematics using the Denavit–Hartenberg (D-H) convention. The UR5 robot has six joints, and its kinematic chain can be described using the D-H parameters summarized in Table 1.

Table 1: DH Parameters for UR5

Joint	a	α	d	θ
1	0	$\frac{\pi}{2}$	0.089159	θ_1
2	-0.425	0	0	θ_2
3	-0.39225	0	0	θ_3
4	0	$\frac{\pi}{2}$	0.10915	θ_4
5	0	$-\frac{\pi}{2}$	0.09465	θ_5
6	0	0	0.0823	θ_6

The forward transformation from the base to the tool frame is computed as:

$$g_{st}(\theta) = A_1(\theta_1)A_2(\theta_2)A_3(\theta_3)A_4(\theta_4)A_5(\theta_5)A_6(\theta_6) \tag{10}$$

where each transformation matrix A_i is calculated using:

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

The correctness of the forward kinematics implementation was validated using the provided script `test_ur5InvKin.m`. This verification ensured that the computed poses aligned accurately with the solutions generated by the `ur5InvKin` function, which internally relies on the same D-H parameter convention.

2.2 Safety Limits

A safety limits check was implemented in `check_safety_limits.m`. First an axis-aligned bounding box was measured used a ruler and the following were determined as lower and upper bounds in table coordinates.

$$\vec{p}_{lower} = [-0.6, -0.2, 0.0]^T \vec{p}_{upper} = [0.14, 0.8, 1.0]^T \tag{12}$$

The units are the values are in meters. Table coordinates refers to frame computed by rotating base link coordinate system by $\pi/4$ about its z-axis, such that it is aligned with the edges of the table.

Joint radii were also measured to approximate the thickness of the cylindrical joints ($joint1 \leftarrow 0.06, 2 \leftarrow 0.06, 3 \leftarrow 0.06, 4 \leftarrow 0.04, 5 \leftarrow 0.04, 6 \leftarrow 0.04$). Thus if any part of this spherical boundary around the joint frame origin leaves the bounding box, then the safety limit is breached.

Finally, the forward kinematics were used to check that each joint is inside the bounding box after transforming the joint frame origin to table coordinates. We also checked that the pen, with a pose of $\boldsymbol{R}_{pen} = \boldsymbol{I}_3$, $\vec{p}_{pen} = [0, -0.049, 0.12228]^T$ from the tool0 frame is within the bounding box (no joint radii was used here).

2.3 Body Jacobian

The body Jacobian was reformulated to match the 0-angle joint configuration and match the forward kinematics. We also formulated it for the pen frame which is rigidly attached to the tool0 frame for use in the resolve-rate control and Jacobian transpose controls. See those sections and the Mathematica notebook for details.

3 Implementation

3.1 Inverse Kinematics Control

The inverse kinematics (IK) implementation uses the provided function `ur5InvKin.m` to compute all eight possible joint solutions that achieve the desired end-effector pose. These solutions are derived following the Denavit-Hartenberg convention and cover the full kinematic configuration space of the UR5.

To ensure safe and optimal execution, the implementation applies a filtering procedure. First, the determinant of the Jacobian matrix is computed for each solution to assess proximity to singularities. Solutions where the determinant magnitude falls below a threshold (1e-6) are discarded to avoid unstable or unsafe configurations. In parallel, the joint angles are checked against the mechanical safety limits of the UR5 using the `check_safety_limits` function, ensuring that the robot remains within its operational boundaries.

From the remaining valid solutions, the algorithm selects the configuration closest to the robot's current joint state. This is determined by calculating the Euclidean distance between each candidate solution and the current joint configuration and choosing the one with the minimum distance. This approach minimizes unnecessary joint motion and reduces the risk of sudden or large movements.

Once the target joint angles are selected, the command is sent to the robot using the `ur5.move_joints` function, with a calculated time interval that accounts for the maximum time required joint displacement and avoids going over the velocity limit. To ensure smooth execution, a short pause is introduced after each movement to allow the robot to settle before proceeding to the next point.

3.2 Resolve-Rate Control

The resolved-rate control (RR) method was implemented to iteratively drive the UR5 end-effector toward the desired target pose by computing the necessary joint velocity commands. This approach is based on differential kinematics, where the instantaneous error between the current and desired pose is transformed into joint space updates using the robot's Jacobian.

At each control step, the current joint angles are read and the pen pose is obtained using forward kinematics (get the tool0 pose from `ur5InvKin` and multiply by g_{pen} on the right). The error between the current and target poses is computed as a twist vector ξ_k, derived from the matrix logarithm of the transformation between the desired pose g_d and the current pose g_k:

$$\vec{\xi}_k = \log^{\vee}(g_d^{+} g_k) \tag{13}$$

This twist vector contains both translational (v_k) and rotational (ω_k) components, which represent the linear and angular errors. They are checked against thresholds (5/1000 and $3 * \pi/180$ respectively) for termination.

If the algorithm terminates, then the errors are reported as:

$$d_{SO(3)} = \sqrt{\operatorname{tr}\left((R - R_d)(R - R_d)^{\top}\right)}$$

$$d_{\mathbb{R}^3} = \|\mathbf{r} - \mathbf{r}_d\|.$$

The joint update is computed using the pseudo-inverse of the body Jacobian J_b as:

$$\Delta\theta_k = J_b^{+}(\theta_k) K \xi_k \tag{14}$$

$$\theta_{k+1} = \theta_k - \Delta\theta_k \tag{15}$$

where K is the control gain. The gain is chosen by the user and the time step is calculated which accounts for the maximum time required joint displacement and avoids going over the velocity limit. A gain of $K = 0.5$ was used primarily.

To ensure robust and safe execution, the determinant of the Jacobian matrix was monitored at each iteration. Configurations close to singularity were detected by thresholding the determinant magnitude, and if violated, the control was safely aborted. Furthermore, the updated joint values were wrapped to stay within the range $[-\pi, \pi]$ and checked against the robot's safety limits to avoid collisions and hardware limits.

3.3 Jacobian-Transpose Control

The Jacobian-Transpose (JT) control method was implemented as an alternative differential control strategy to move the UR5 end-effector toward the desired pose. Similar to the resolved-rate control (RR) method, the JT controller iteratively reduces the pose error by updating the joint configuration based on the error twist vector between the current and target poses.

The algorithmic componenets for the Jacobian-Tranpose control was the same as the Resolve-Rate control except the joint update was computed using the transpose of the body Jacobian J_b as:

$$\Delta\theta_k = J_b^T(\theta_k) K \xi_k \tag{16}$$

where K is the control gain. In this implementation, a static gain was used, as adaptive or distance-dependent gain tuning did not yield significant performance improvements. A typical gain value of $K = 0.4$ was used for physical experiments, while a slightly higher value was tested in simulation to improve convergence speed.

4 Simulation Results

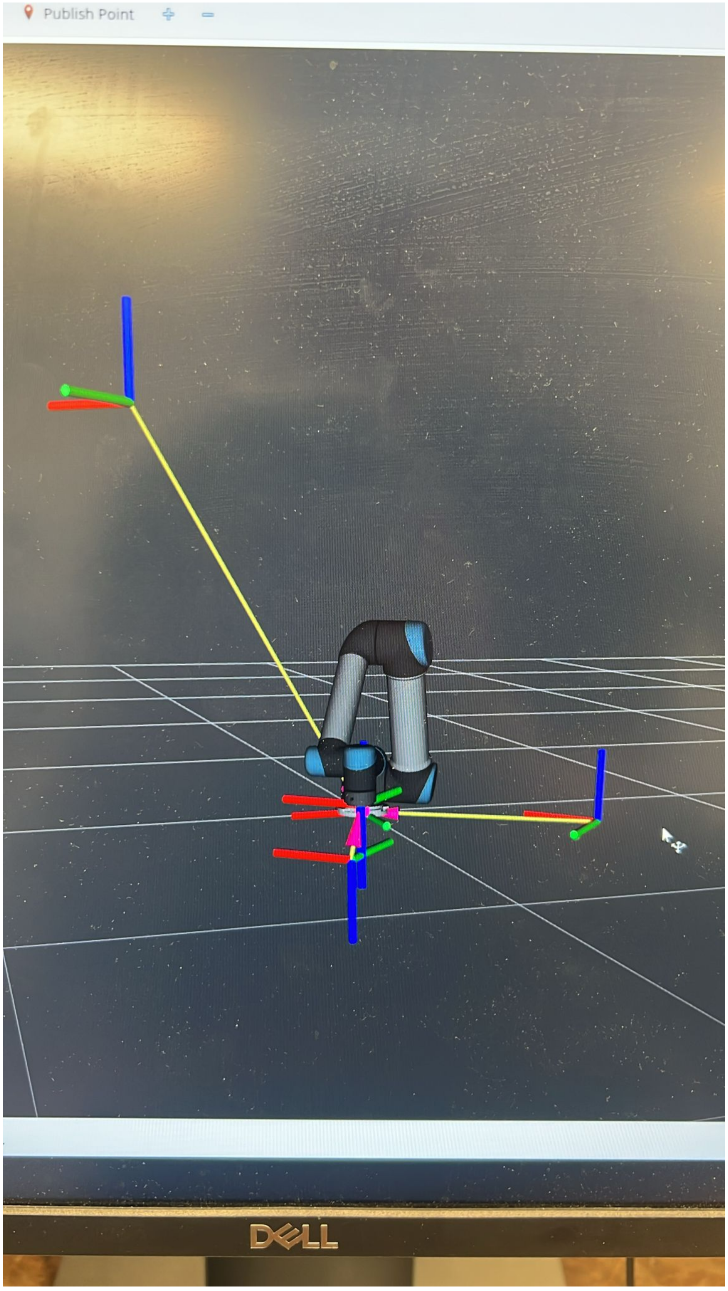

Figure 2: Simulation with frames at first goal point during operation

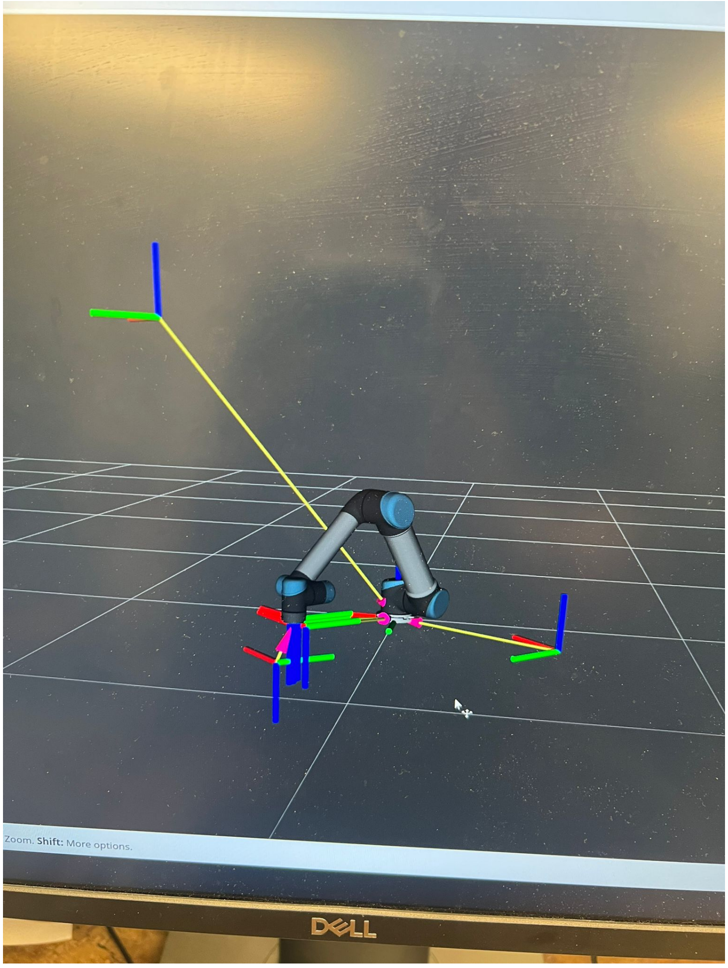

Figure 3: Simulation with frames at second goal point during operation

4.1 Inverse Kinematics Control

Note that the translation error is in meters. These are the errors in the pen pose when the control function is inputted with the following poses.

Table 2: Error Metrics for Inverse Kinematics Control

Pose	$d_{SO(3)}$	$d_{\mathbb{R}^3}$
g_{start}	0.5320×10^{-4}	0.4034×10^{-4}
g_{end}	0.1008×10^{-4}	0.0445×10^{-3}
g_{middle_up}	0.5542×10^{-3}	0.1749×10^{-3}
g_{middle}	0.2032×10^{-3}	0.0672×10^{-3}
$g_{vertical\_end}$	0.2744×10^{-3}	0.1171×10^{-3}
$g_{vertical\_end\_up}$	0.3009×10^{-4}	0.4303×10^{-4}

The inverse kinematics implementation was first evaluated in the RViz simulation environment before deployment on the physical UR5 robot. The start and target positions were taught manually, and intermediate points were computed as described in the implementation section.

The simulation results showed that the robot successfully reached all target positions, including intermediate and hover points, with minimal error.

These results indicate excellent alignment between the desired and achieved poses, confirming the accuracy and stability of the IK implementation. The robot's motion was smooth and collision-free, with no observed violations of joint limits or singularities during execution.

A representative trajectory is shown in RViz visualizations, where the robot moves sequentially through all key points, effectively tracing the intended “T” shape on the simulated board. The inclusion of intermediate and hover points was critical in maintaining the stability and quality of the path.

4.2 Resolve-Rate Control

Note that the translation error is in meters. These are the errors in the pen pose when the control function is inputted with the following poses.

Table 3: Error Metrics for Resolve-Rate Control

Pose	$d_{SO(3)}$	$d_{\mathbb{R}^3}$
g_{start}	0.0089	0.0039
g_{end}	0.0106	0.0045
g_{middle_up}	0.0205	0.0049
g_{middle}	0.0032	0.0026
$g_{vertical\_end}$	0.0007	0.0039
$g_{vertical\_end\_up}$	0.0007	0.0027

The resolved-rate control (RR) implementation was evaluated in RViz simulation before being tested on the physical UR5 robot. The control loop was executed using the computed intermediate points, including the horizontal and vertical segments of the letter “T,” and carefully planned hover positions to ensure smooth transitions.

The system gain was set to $K = 0.5$. This parameter were selected to achieve a balance between convergence speed and system stability. The RR controller effectively guided the robot through the trajectory, with the pen being lifted between strokes to avoid smearing or unintended markings.

These results demonstrate that the RR controller achieved accurate tracking at each stage of the trajectory. RViz visualizations showed a smooth and stable path, with the robot successfully executing the “T” shape without entering singular configurations or exceeding joint limits.

Overall, the resolved-rate control implementation proved to be reliable and effective, with only minor position errors and stable convergence across all key points.

4.3 Jacobian-Transpose Control

Note that the translation error is in meters. These are the errors in the pen pose when the control function is inputted with the following poses.

Table 4: Error Metrics for Jacobian-Transpose Control

Pose	$d_{SO(3)}$	$d_{\mathbb{R}^3}$
g_{start}	0.0012	0.0049
g_{end}	0.0026	0.0050
g_{middle_up}	0.0025	0.0049
g_{middle}	0.0029	0.0049
$g_{vertical\_end}$	0.0023	0.0049
$g_{vertical\_end\_up}$	0.0018	0.0050

The Jacobian-Transpose (JT) control implementation was tested in RViz simulation using the same start, intermediate, and target points as in the previous control methods. The JT controller was evaluated with a fixed control gain $K = 0.4$ for physical tests and a slightly higher gain during simulation to improve convergence speed.

The JT method successfully guided the UR5 end-effector along the planned trajectory, including the horizontal and vertical strokes of the letter "T." The method was generally slower to converge and less accurate than the resolved-rate control method, but this makes since because using the transpose instead of the pseudo-inverse sacrifices accuracy for a quicker computation speed.

5 Contributions

Suhash Reddy participated in the demo and worked on the resolve rate control and the Jacobian transpose control and their testing.

Sungwoo Kim participated in the demo and worked on the resolve rate control and Jacobian transpose control and their testing. Sungwoo also worked on Extra Credit 2.

Andrew Ying worked on the inverse kinematics control, resolve rate control, and Jacobian transpose control and their testing. Andrew also worked on Extra Credit 1.

Extra Credit Task 1

For the extra credit task, an additional script `bonus_bspline.m` was developed to enable the UR5 robot to follow a smooth, continuous trajectory through at least four user-defined control points. The user manually selects these control points by moving the robot to desired positions using the pendant control and pressing a key to record each point. The system automatically collects both the Cartesian positions and the orientation from the last recorded point to maintain consistent end-effector alignment along the trajectory.

The trajectory between the selected control points is generated using a B-spline interpolation implemented in the function `bspline_interp()`. This function calculates intermediate points along the curve by blending the control points with cubic B-spline basis functions, providing a smooth path that avoids sharp turns or abrupt changes in direction.

The user is prompted to select one of three control modes—Inverse Kinematics (IK), Resolved-Rate Control (RR), or Jacobian Transpose Control (JT)—to execute the motion. The script iterates over 100 interpolated points along the B-spline curve, constructing a target homogeneous transformation matrix (`g_desired`) for each point using the fixed orientation and interpolated position. Depending on the selected mode, the script calls `ur5IKcontrol`, `ur5RRcontrol`, or `ur5JTcontrol` to move the robot along the trajectory.

At each step, the system checks for singularities and safety violations, ensuring stable and reliable motion. Error messages or success confirmations are printed, and the final error metrics are reported for each interpolation point. This task demonstrates advanced trajectory generation and control integration and expands the robot's capability to follow complex, user-defined paths smoothly, earning extra credit in the project.

Extra Credit Task 2

For the second extra credit task, a Kakao character was drawn using computer vision and robotic control techniques. The process began with extracting the image contours using the `extract_drawing_path.m` script. The original color image was first converted to grayscale and then binarized using a defined threshold. A Canny edge detector was applied to extract the boundary lines of the image. Once the boundary lines were identified, the 2D path coordinates were extracted. To ensure a smooth drawing trajectory, the points were separated by individual paths. These coordinates were compiled and saved in a file named `draw_path.csv`. Using the

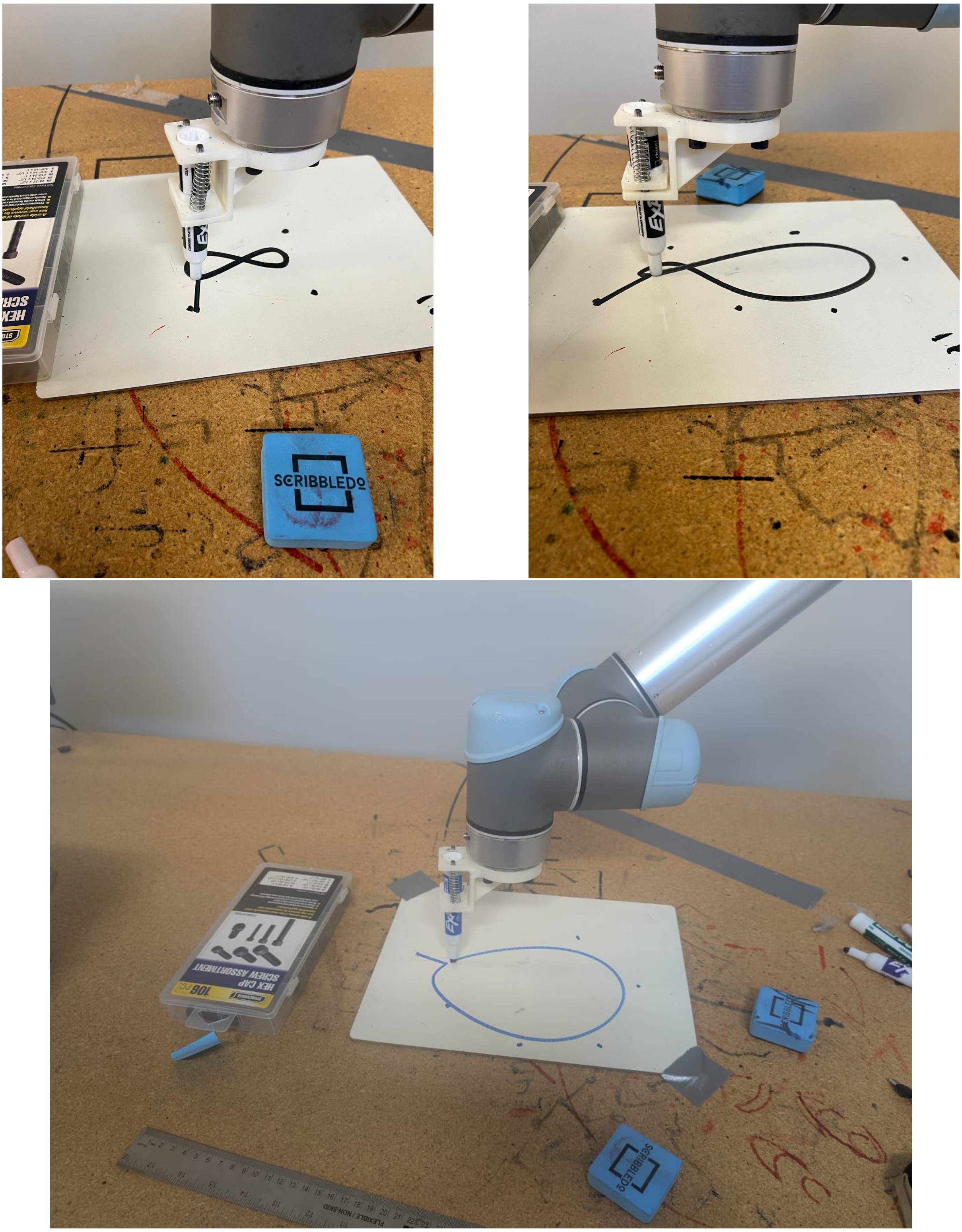

Figure 4: Demonstration of Extra Credit 1

extracted path points, the drawing was executed through inverse kinematics implemented in the `bonus_kakao.m` script. Due to the high number of points, every fifth point was selected for the final drawing. Additionally, after each line was completed, the marker was lifted by five centimeters and moved to the start of the next line to avoid overlapping. As a result, the paths were drawn smoothly, and the character was recognizable.

Figure 5: Preprocessing for Extra Credit 2: Original image (left), Contour image (center), Matlab 2D point visualization(right)

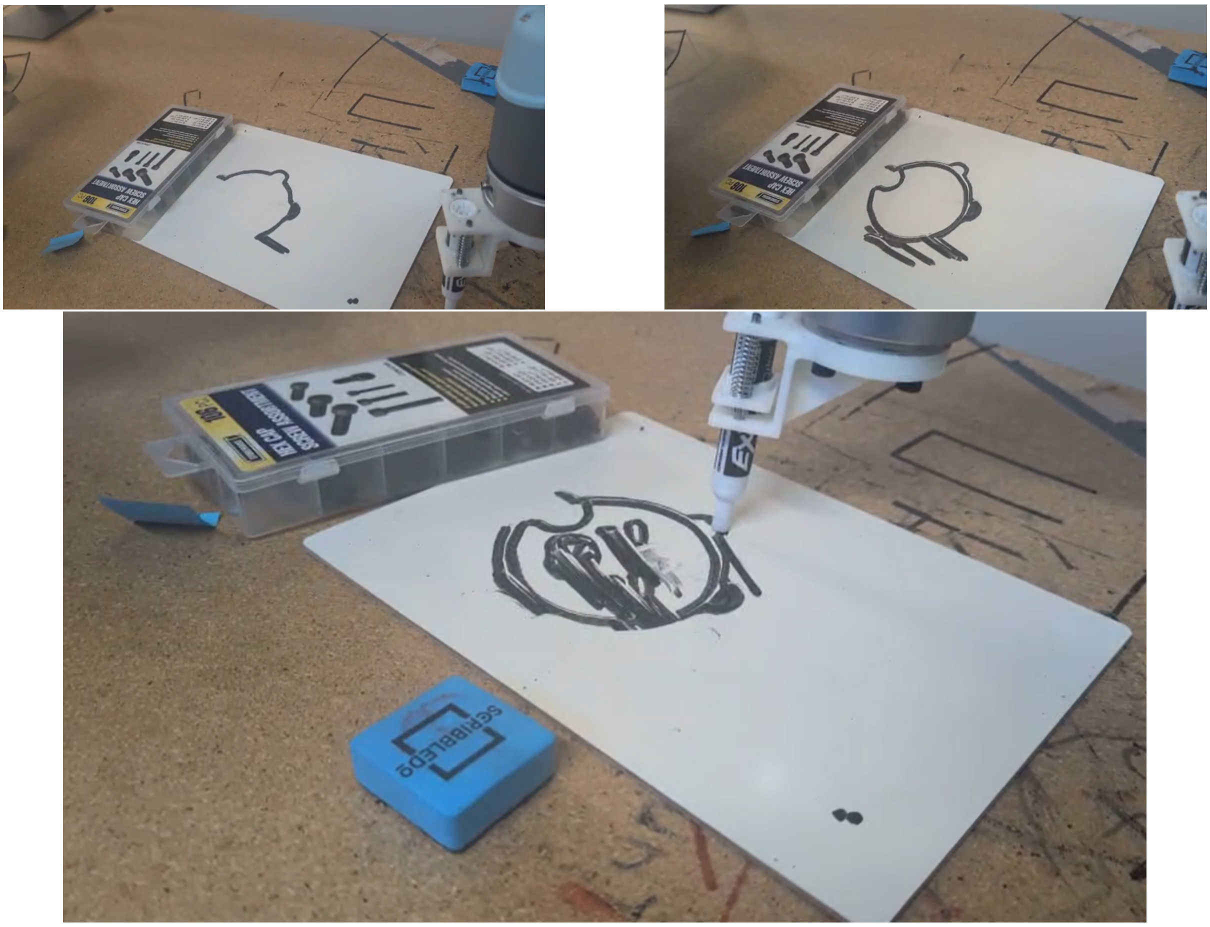

Figure 6: Demonstration of Extra Credit 2. Unfortunately the image got smeared during this run, but the outline is sharp.

Imports

[145]:=

```
Needs["Screws`"]
Needs["RobotLinks`"]
```

1.

UR5

a) Forward Kinematics

```
(* unit vectors *)
e1 = {1, 0, 0};
e2 = {0, 1, 0};
e3 = {0, 0, 1};

gst0 = RPToHomogeneous[RotationMatrix[π, e3] . RotationMatrix[π / 2, e1], {L1 + L2, L3 + L5, L0 - L4}];

ω1 = e3; q1 = {0, 0, 0};
ξ1 = RevoluteTwist[q1, ω1];

ω2 = e2; q2 = {0, 0, L0};
ξ2 = RevoluteTwist[q2, ω2];

ω3 = e2; q3 = {L1, 0, L0};
ξ3 = RevoluteTwist[q3, ω3];

ω4 = e2; q4 = {L1 + L2, 0, L0};
ξ4 = RevoluteTwist[q4, ω4];

ω5 = -e3; q5 = {L1 + L2, L3, 0};
ξ5 = RevoluteTwist[q5, ω5];

ω6 = e2; q6 = { L1 + L2, 0, L0 - L4};
ξ6 = RevoluteTwist[q6, ω6];

gst0 // MatrixForm
ξ1 // MatrixForm
ξ2 // MatrixForm
ξ3 // MatrixForm
ξ4 // MatrixForm
ξ5 // MatrixForm
ξ6 // MatrixForm
```

MatrixForm=

$$\begin{pmatrix} -1 & 0 & 0 & L1+L2 \\ 0 & 0 & 1 & L3+L5 \\ 0 & 1 & 0 & L0-L4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} -L0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} -L0 \\ 0 \\ L1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} -L0 \\ 0 \\ L1+L2 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} -L3 \\ L1+L2 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} -L0+L4 \\ 0 \\ L1+L2 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

```
f = FullSimplify[ForwardKinematics[{ξ1, θ1}, {ξ2, θ2}, {ξ3, θ3}, {ξ4, θ4}, {ξ5, θ5}, {ξ6, θ6}, gst0]];
(* analytic solution *)
f // MatrixForm

(* with link lengths in meters substituted *)
f /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825} // MatrixForm
```

MatrixForm=

$$\begin{pmatrix} -\text{Cos}[\theta6]\ (\text{Cos}[\theta1]\ \text{Cos}[\theta2+\theta3+\theta4]\ \text{Cos}[\theta5]+\text{Sin}[\theta1]\ \text{Sin}[\theta5])+\text{Cos}[\theta1]\ \text{Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta6] & \text{Sin}[\theta1]\ \text{Sin}[\theta5]\ \text{Sin}[\\ -\text{Cos}[\theta2+\theta3+\theta4]\ \text{Cos}[\theta5]\ \text{Cos}[\theta6]\ \text{Sin}[\theta1]+\text{Cos}[\theta1]\ \text{Cos}[\theta6]\ \text{Sin}[\theta5]+\text{Sin}[\theta1]\ \text{Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta6] & \text{Cos}[\theta6]\ \text{Sin}[\theta1]\ \text{Sin}[\\ \text{Cos}[\theta5]\ \text{Cos}[\theta6]\ \text{Sin}[\theta2+\theta3+\theta4]+\text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta6] & \text{Cos} \\ 0 & \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} -\text{Cos}[\theta6]\ (\text{Cos}[\theta1]\ \text{Cos}[\theta2+\theta3+\theta4]\ \text{Cos}[\theta5]+\text{Sin}[\theta1]\ \text{Sin}[\theta5])+\text{Cos}[\theta1]\ \text{Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta6] & \text{Sin}[\theta1]\ \text{Sin}[\theta5]\ \text{Sin}[\\ -\text{Cos}[\theta2+\theta3+\theta4]\ \text{Cos}[\theta5]\ \text{Cos}[\theta6]\ \text{Sin}[\theta1]+\text{Cos}[\theta1]\ \text{Cos}[\theta6]\ \text{Sin}[\theta5]+\text{Sin}[\theta1]\ \text{Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta6] & \text{Cos}[\theta6]\ \text{Sin}[\theta1]\ \text{Sin}[\\ \text{Cos}[\theta5]\ \text{Cos}[\theta6]\ \text{Sin}[\theta2+\theta3+\theta4]+\text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta6] & \text{Cos} \\ 0 & \end{pmatrix}$$

Forward Kinematics Testing

```
f /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825,
    θ1 → π / 12, θ2 → -π / 2, θ3 → π / 4, θ4 → -π / 3, θ5 → -π, θ6 → π / 6} // MatrixForm // N
```

MatrixForm=

$$\begin{pmatrix} -0.683013 & -0.683013 & 0.258819 & 0.349208 \\ -0.183013 & -0.183013 & -0.965926 & 0.121315 \\ 0.707107 & -0.707107 & 0. & 0.815909 \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

```
θ6] + Cos[θ1] (Cos[θ6] Sin[θ2 + θ3 + θ4] + Cos[θ2 + θ3 + θ4] Cos[θ5] Sin[θ6])   -Cos[θ5] Sin[θ1] + Cos[θ1] Cos[θ2 + θ3 + θ4] Sin[θ5
θ2 + θ3 + θ4] + (Cos[θ2 + θ3 + θ4] Cos[θ5] Sin[θ1] - Cos[θ1] Sin[θ5]) Sin[θ6]   Cos[θ1] Cos[θ5] + Cos[θ2 + θ3 + θ4] Sin[θ1] Sin[θ5]
s[θ2 + θ3 + θ4] Cos[θ6] - Cos[θ5] Sin[θ2 + θ3 + θ4] Sin[θ6]                    -Sin[θ2 + θ3 + θ4] Sin[θ5]
                          0                                                                        0
```


```
θ6] + Cos[θ1] (Cos[θ6] Sin[θ2 + θ3 + θ4] + Cos[θ2 + θ3 + θ4] Cos[θ5] Sin[θ6])   -Cos[θ5] Sin[θ1] + Cos[θ1] Cos[θ2 + θ3 + θ4] Sin[θ5
θ2 + θ3 + θ4] + (Cos[θ2 + θ3 + θ4] Cos[θ5] Sin[θ1] - Cos[θ1] Sin[θ5]) Sin[θ6]   Cos[θ1] Cos[θ5] + Cos[θ2 + θ3 + θ4] Sin[θ1] Sin[θ5]
s[θ2 + θ3 + θ4] Cos[θ6] - Cos[θ5] Sin[θ2 + θ3 + θ4] Sin[θ6]                    -Sin[θ2 + θ3 + θ4] Sin[θ5]
                          0                                                                        0
```

] -((L3 + L5 Cos[θ5]) Sin[θ1]) + Cos[θ1] (L1 Cos[θ2] + L2 Cos[θ2 + θ3] - L4 Sin[θ2 + θ3 + θ4] + L5 Cos[θ2 + θ3 + θ4] Sin[θ5]))
Cos[θ1] (L3 + L5 Cos[θ5]) + Sin[θ1] (L1 Cos[θ2] + L2 Cos[θ2 + θ3] - L4 Sin[θ2 + θ3 + θ4] + L5 Cos[θ2 + θ3 + θ4] Sin[θ5])
L0 - L4 Cos[θ2 + θ3 + θ4] - L1 Sin[θ2] - L2 Sin[θ2 + θ3] - L5 Sin[θ2 + θ3 + θ4] Sin[θ5]
1)

] -((0.1093 + 0.0825 Cos[θ5]) Sin[θ1]) + Cos[θ1] (0.425 Cos[θ2] + 0.392 Cos[θ2 + θ3] - 0.09475 Sin[θ2 + θ3 + θ4] + 0.0825 Cos[θ2 +
Cos[θ1] (0.1093 + 0.0825 Cos[θ5]) + Sin[θ1] (0.425 Cos[θ2] + 0.392 Cos[θ2 + θ3] - 0.09475 Sin[θ2 + θ3 + θ4] + 0.0825 Cos[θ2 + θ
0.0892 - 0.09475 Cos[θ2 + θ3 + θ4] - 0.425 Sin[θ2] - 0.392 Sin[θ2 + θ3] - 0.0825 Sin[θ2 + θ3 + θ4] Sin[θ5]
1

$$\left.\begin{matrix} + \theta 3 + \theta 4] \; \text{Sin}[\theta 5]) \\ \theta 3 + \theta 4] \; \text{Sin}[\theta 5]) \\ \\ \end{matrix}\right)$$

b) Jacobian

```
Js = FullSimplify[SpatialJacobian[{ξ1, θ1}, {ξ2, θ2}, {ξ3, θ3}, {ξ4, θ4}, {ξ5, θ5}, {ξ6, θ6}, gst0]];
Js // MatrixForm
Js /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825} // MatrixForm
```

MatrixForm=

$$\begin{pmatrix} 0 & -L0\,\mathrm{Cos}[\theta 1] & \mathrm{Cos}[\theta 1]\,(-L0 + L1\,\mathrm{Sin}[\theta 2]) & \mathrm{Cos}[\theta 1]\,(-L0 + L1\,\mathrm{Sin}[\theta 2] + L2\,\mathrm{Sin}[\theta 2 + \theta 3]) & \frac{1}{2}\,((L0 - L3)\,\mathrm{Cos}[\theta 1 - \theta 2 - \theta 3 - \theta 4] - (L0 \\ 0 & -L0\,\mathrm{Sin}[\theta 1] & \mathrm{Sin}[\theta 1]\,(-L0 + L1\,\mathrm{Sin}[\theta 2]) & \mathrm{Sin}[\theta 1]\,(-L0 + L1\,\mathrm{Sin}[\theta 2] + L2\,\mathrm{Sin}[\theta 2 + \theta 3]) & \frac{1}{2}\,(2\,\mathrm{Cos}[\theta 1]\,(L2\,\mathrm{Cos}[\theta 4] + L1\,\mathrm{Cos}[\theta 3 + \theta \\ 0 & 0 & L1\,\mathrm{Cos}[\theta 2] & L1\,\mathrm{Cos}[\theta 2] + L2\,\mathrm{Cos}[\theta 2 + \theta 3] & \\ 0 & -\mathrm{Sin}[\theta 1] & -\mathrm{Sin}[\theta 1] & -\mathrm{Sin}[\theta 1] & \\ 0 & \mathrm{Cos}[\theta 1] & \mathrm{Cos}[\theta 1] & \mathrm{Cos}[\theta 1] & \\ 1 & 0 & 0 & 0 & \end{pmatrix}$$

MatrixForm=

$$\begin{pmatrix} 0 & -0.0892\,\mathrm{Cos}[\theta 1] & \mathrm{Cos}[\theta 1]\,(-0.0892 + 0.425\,\mathrm{Sin}[\theta 2]) & \mathrm{Cos}[\theta 1]\,(-0.0892 + 0.425\,\mathrm{Sin}[\theta 2] + 0.392\,\mathrm{Sin}[\theta 2 + \theta 3]) & \frac{1}{2}\,(-0.0201\,\mathrm{Cos}[\\ 0 & -0.0892\,\mathrm{Sin}[\theta 1] & \mathrm{Sin}[\theta 1]\,(-0.0892 + 0.425\,\mathrm{Sin}[\theta 2]) & \mathrm{Sin}[\theta 1]\,(-0.0892 + 0.425\,\mathrm{Sin}[\theta 2] + 0.392\,\mathrm{Sin}[\theta 2 + \theta 3]) & \frac{1}{2}\,(2\,\mathrm{Cos}[\theta 1]\,(0 \\ 0 & 0 & 0.425\,\mathrm{Cos}[\theta 2] & 0.425\,\mathrm{Cos}[\theta 2] + 0.392\,\mathrm{Cos}[\theta 2 + \theta 3] & \\ 0 & -\mathrm{Sin}[\theta 1] & -\mathrm{Sin}[\theta 1] & -\mathrm{Sin}[\theta 1] & \\ 0 & \mathrm{Cos}[\theta 1] & \mathrm{Cos}[\theta 1] & \mathrm{Cos}[\theta 1] & \\ 1 & 0 & 0 & 0 & \end{pmatrix}$$

```
+ L3) Cos[θ1 + θ2 + θ3 + θ4] - 2 (L2 Cos[θ4] + L1 Cos[θ3 + θ4]) Sin[θ1])          -Sin[θ1] (-L4 + L0 Cos[θ2 + θ3 + θ4] + L2 Sin[θ4] + l
θ4]) + (L0 - L3) Sin[θ1 - θ2 - θ3 - θ4] - (L0 + L3) Sin[θ1 + θ2 + θ3 + θ4])  Cos[θ5] Sin[θ1] (-L0 + L4 Cos[θ2 + θ3 + θ4] + L1 Sin[θ2] + L
      L3 Sin[θ2 + θ3 + θ4]                                                                                   Cos[
   -Cos[θ1] Sin[θ2 + θ3 + θ4]
   -Sin[θ1] Sin[θ2 + θ3 + θ4]
      -Cos[θ2 + θ3 + θ4]

θ1 - θ2 - θ3 - θ4] - 0.1985 Cos[θ1 + θ2 + θ3 + θ4] - 2 (0.392 Cos[θ4] + 0.425 Cos[θ3 + θ4]) Sin[θ1])    -Sin[θ1] (-0.09475 + 0.0892 Co
.392 Cos[θ4] + 0.425 Cos[θ3 + θ4]) - 0.0201 Sin[θ1 - θ2 - θ3 - θ4] - 0.1985 Sin[θ1 + θ2 + θ3 + θ4])  Cos[θ5] Sin[θ1] (-0.0892 + 0.094
                              0.1093 Sin[θ2 + θ3 + θ4]
                             -Cos[θ1] Sin[θ2 + θ3 + θ4]
                             -Sin[θ1] Sin[θ2 + θ3 + θ4]
                                -Cos[θ2 + θ3 + θ4]
```

$$\text{L1 Sin}[\theta3+\theta4])\ \text{Sin}[\theta5]+\text{Cos}[\theta1]\ (\text{Cos}[\theta5]\ (-\text{L0}+\text{L4 Cos}[\theta2+\theta3+\theta4]+\text{L1 Sin}[\theta2]+\text{L2 Sin}[\theta2+\theta3])-\text{L3 Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta5]$$

$$\text{2 Sin}[\theta2+\theta3])+\frac{1}{2}\ ((\text{L0}-\text{L3})\ \text{Cos}[\theta1-\theta2-\theta3-\theta4]+(\text{L0}+\text{L3})\ \text{Cos}[\theta1+\theta2+\theta3+\theta4]+2\ \text{Cos}[\theta1]\ (-\text{L4}+\text{L2 Sin}[\theta4]+\text{L1 Sin}[\theta3+\theta4])$$

$$\theta5]\ (\text{L1 Cos}[\theta2]+\text{L2 Cos}[\theta2+\theta3]-\text{L4 Sin}[\theta2+\theta3+\theta4])-\text{L3 Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta5]$$

$$-\text{Cos}[\theta5]\ \text{Sin}[\theta1]+\text{Cos}[\theta1]\ \text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta5]$$

$$\text{Cos}[\theta1]\ \text{Cos}[\theta5]+\text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta1]\ \text{Sin}[\theta5]$$

$$-\text{Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta5]$$

$$\text{s}[\theta2+\theta3+\theta4]+0.392\ \text{Sin}[\theta4]+0.425\ \text{Sin}[\theta3+\theta4])\ \text{Sin}[\theta5]+\text{Cos}[\theta1]\ (\text{Cos}[\theta5]\ (-0.0892+0.09475\ \text{Cos}[\theta2+\theta3+\theta4]+0.425\ \text{Sin}[\theta$$

$$\text{475 Cos}[\theta2+\theta3+\theta4]+0.425\ \text{Sin}[\theta2]+0.392\ \text{Sin}[\theta2+\theta3])+\frac{1}{2}\ (-0.0201\ \text{Cos}[\theta1-\theta2-\theta3-\theta4]+0.1985\ \text{Cos}[\theta1+\theta2+\theta3+\theta4]+2\ \text{Cos}[$$

$$\text{Cos}[\theta5]\ (0.425\ \text{Cos}[\theta2]+0.392\ \text{Cos}[\theta2+\theta3]-0.09475\ \text{Sin}[\theta2+\theta3+\theta4])-0.1093\ \text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta$$

$$-\text{Cos}[\theta5]\ \text{Sin}[\theta1]+\text{Cos}[\theta1]\ \text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta5]$$

$$\text{Cos}[\theta1]\ \text{Cos}[\theta5]+\text{Cos}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta1]\ \text{Sin}[\theta5]$$

$$-\text{Sin}[\theta2+\theta3+\theta4]\ \text{Sin}[\theta5]$$

```
])
) Sin[θ5]

2] + 0.392 Sin[θ2 + θ3]) - 0.1093 Sin[θ2 + θ3 + θ4] Sin[θ5])
θ1] (-0.09475 + 0.392 Sin[θ4] + 0.425 Sin[θ3 + θ4])) Sin[θ5]
5]
```

```
Jb = FullSimplify[BodyJacobian[{ξ1, θ1}, {ξ2, θ2}, {ξ3, θ3}, {ξ4, θ4}, {ξ5, θ5}, {ξ6, θ6}, gst0]];
Jb // MatrixForm
Jb /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825} // MatrixForm

FullSimplify[Det[Jb]]
```

MatrixForm=

```
(        Sin[θ2] (-Cos[θ6] ((L5 + L3 Cos[θ5]) Sin[θ3 + θ4] + (L4 Cos[θ3 + θ4] + L2 Sin[θ3]) Sin[θ5]) - Cos[θ3 + θ4] (L3 + L5 Cos[
 -Cos[θ3] (L3 + L5 Cos[θ5]) Cos[θ6] Sin[θ2 + θ4] + ((L5 + L3 Cos[θ5]) Sin[θ3] Sin[θ2 + θ4] + (-Cos[θ2] (L1 + L2 Cos[θ3]) + L2 Sin
                                                                                       Cos[θ5] (L1 Cos[θ2] + L2
                                                                                                 Cos[θ5] Co
                                                                                                 Cos[θ2 + θ3
```

MatrixForm=

```
(        Sin[θ2] (-Cos[θ6] ((0.0825 + 0.1093 Cos[θ5]) Sin[θ3 + θ4] + (0.09475 Cos[θ3 + θ4] + 0.392 Sin[θ3]) Sin[θ5]) - Cos[θ3 +
 -Cos[θ3] (0.1093 + 0.0825 Cos[θ5]) Cos[θ6] Sin[θ2 + θ4] + ((0.0825 + 0.1093 Cos[θ5]) Sin[θ3] Sin[θ2 + θ4] + (-Cos[θ2] (0.425 +
                                                                                                     Cos[θ
```

$$-L1\,L2\,\mathrm{Sin}[\theta 2 \text{ wait}]$$

Body Jacobian Testing

```
Jb /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825,
    θ1 → π / 12, θ2 → -π / 2, θ3 → π / 4, θ4 → -π / 3, θ5 → -π, θ6 → π / 6} // MatrixForm // N
```

MatrixForm=

$$\begin{pmatrix} 0.0189505 & -0.774576 & -0.474056 & -0.0820559 & -0.0714471 & 0. \\ 0.0189505 & -0.253145 & 0.047375 & 0.047375 & 0.04125 & 0. \\ -0.368707 & 0. & 0. & 0. & 0. & 0. \\ 0.707107 & 0. & 0. & 0. & -0.5 & 0. \\ -0.707107 & 0. & 0. & 0. & -0.866025 & 0. \\ 0. & -1. & -1. & -1. & 0. & 1. \end{pmatrix}$$

θ5]) Sin[θ6]) + Cos[θ2] (Cos[θ3 + θ4] (L5 + L3 Cos[θ5]) Cos[θ6] + Cos[θ6] (L1 + L2 Cos[θ3] - L4 Sin[θ3 + θ4]) Sin[θ5] - (L3 + L5 Co
[θ2] Sin[θ3] + L4 Cos[θ3] Sin[θ2 + θ4]) Sin[θ5]) Sin[θ6] - Cos[θ2 + θ4] ((L3 + L5 Cos[θ5]) Cos[θ6] Sin[θ3] + (Cos[θ3] (L5 + L3 Cos
Cos[θ2 + θ3] - L4 Sin[θ2 + θ3 + θ4]) - L3 Cos[θ2 + θ3 + θ4] Sin[θ5]
s[θ6] Sin[θ2 + θ3 + θ4] + Cos[θ2 + θ3 + θ4] Sin[θ6]
3 + θ4] Cos[θ6] - Cos[θ5] Sin[θ2 + θ3 + θ4] Sin[θ6]
-Sin[θ2 + θ3 + θ4] Sin[θ5]

+ θ4] (0.1093 + 0.0825 Cos[θ5]) Sin[θ6]) + Cos[θ2] (Cos[θ3 + θ4] (0.0825 + 0.1093 Cos[θ5]) Cos[θ6] + Cos[θ6] (0.425 + 0.392 Cos[θ
- 0.392 Cos[θ3]) + 0.392 Sin[θ2] Sin[θ3] + 0.09475 Cos[θ3] Sin[θ2 + θ4]) Sin[θ5]) Sin[θ6] - Cos[θ2 + θ4] ((0.1093 + 0.0825 Cos[θ5
θ5] (0.425 Cos[θ2] + 0.392 Cos[θ2 + θ3] - 0.09475 Sin[θ2 + θ3 + θ4]) - 0.1093 Cos[θ2 + θ3 + θ4] Sin[θ5]
Cos[θ5] Cos[θ6] Sin[θ2 + θ3 + θ4] + Cos[θ2 + θ3 + θ4] Sin[θ6]
Cos[θ2 + θ3 + θ4] Cos[θ6] - Cos[θ5] Sin[θ2 + θ3 + θ4] Sin[θ6]
-Sin[θ2 + θ3 + θ4] Sin[θ5]

s[θ5]) Sin[θ3 + θ4] Sin[θ6])   Cos[θ5] Cos[θ6] (L4 − L2 Sin[θ4] − L1 Sin[θ3 + θ4]) − (L2 Cos[θ4] + L1 Cos[θ3 + θ4] + L5 Sin
[θ5]) − L4 Sin[θ3] Sin[θ5]) Sin[θ6])   −Cos[θ6] (L2 Cos[θ4] + L1 Cos[θ3 + θ4] + L5 Sin[θ5]) + Cos[θ5] (−L4 + L2 Sin[θ4] + L1 Sin[θ

(−L4 + L2 Sin[θ4] + L1 Sin[θ3 + θ4]) Sin[θ5]

Cos[θ6] Sin[θ5]

−Sin[θ5] Sin[θ6]

Cos[θ5]

3] − 0.09475 Sin[θ3 + θ4]) Sin[θ5] − (0.1093 + 0.0825 Cos[θ5]) Sin[θ3 + θ4] Sin[θ6])   Cos[θ5] Cos[θ6] (0.09475 − 0.392
]) Cos[θ6] Sin[θ3] + (Cos[θ3] (0.0825 + 0.1093 Cos[θ5]) − 0.09475 Sin[θ3] Sin[θ5]) Sin[θ6])   −Cos[θ6] (0.392 Cos[θ4] + 0.425 Co

n[θ5]) Sin[θ6]   Cos[θ5] Cos[θ6] (L4 - L2 Sin[θ4]) - (L2 Cos[θ4] + L5 Sin[θ5]) Sin[θ6]   L4 Cos[θ5] Cos[θ6] - L5 Sin[θ5] Sin[θ

3 + θ4]) Sin[θ6]   -Cos[θ6] (L2 Cos[θ4] + L5 Sin[θ5]) + Cos[θ5] (-L4 + L2 Sin[θ4]) Sin[θ6]   -L5 Cos[θ6] Sin[θ5] - L4 Cos[θ5] Sin[θ

(-L4 + L2 Sin[θ4]) Sin[θ5]   -L4 Sin[θ5]

Cos[θ6] Sin[θ5]   Cos[θ6] Sin[θ5]

-Sin[θ5] Sin[θ6]   -Sin[θ5] Sin[θ6]

Cos[θ5]   Cos[θ5]

Sin[θ4] - 0.425 Sin[θ3 + θ4]) - (0.392 Cos[θ4] + 0.425 Cos[θ3 + θ4] + 0.0825 Sin[θ5]) Sin[θ6]   Cos[θ5] Cos[θ6] (0.09475 - 0.39

os[θ3 + θ4] + 0.0825 Sin[θ5]) + Cos[θ5] (-0.09475 + 0.392 Sin[θ4] + 0.425 Sin[θ3 + θ4]) Sin[θ6]   -Cos[θ6] (0.392 Cos[θ4] + 0.0825

(-0.09475 + 0.392 Sin[θ4] + 0.425 Sin[θ3 + θ4]) Sin[θ5]   (-0.09

Cos[θ6] Sin[θ5]

-Sin[θ5] Sin[θ6]

Cos[θ5]

```
6]  -L5 Cos[θ6]  0
θ6]  L5 Sin[θ6]  0
        0        0
     -Sin[θ6]    0
     -Cos[θ6]    0
        0        1
```

```
2 Sin[θ4]) - (0.392 Cos[θ4] + 0.0825 Sin[θ5]) Sin[θ6]      0.09475 Cos[θ5] Cos[θ6] - 0.0825 Sin[θ5] Sin[θ6]   -0.0825 Cos[θ6]  0
Sin[θ5]) + Cos[θ5] (-0.09475 + 0.392 Sin[θ4]) Sin[θ6]   -0.0825 Cos[θ6] Sin[θ5] - 0.09475 Cos[θ5] Sin[θ6]    0.0825 Sin[θ6]  0
475 + 0.392 Sin[θ4]) Sin[θ5]                                            -0.09475 Sin[θ5]                            0          0
Cos[θ6] Sin[θ5]                                                          Cos[θ6] Sin[θ5]                         -Sin[θ6]      0
-Sin[θ5] Sin[θ6]                                                        -Sin[θ5] Sin[θ6]                         -Cos[θ6]      0
Cos[θ5]                                                                  Cos[θ5]                                    0          1
```

Body Jacobian for Pen Frame

```
gpen = RPToHomogeneous[IdentityMatrix[3], {0, L6, L7}];
Jb = FullSimplify[BodyJacobian[{ξ1, θ1}, {ξ2, θ2}, {ξ3, θ3}, {ξ4, θ4}, {ξ5, θ5}, {ξ6, θ6}, gst0 . gpen]];
Jb // MatrixForm
Jb /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825, L6 → -0.049, L7 → 0.12228} //
 MatrixForm

FullSimplify[Det[Jb]]
```

MatrixForm=

```
                                                  Cos[θ2 + θ3 + θ4] (L5 + L7 + L3 Cos[θ5]) Cos[θ6] + (L1 Cos[θ2] Co
-Cos[θ3] (L3 + (L5 + L7) Cos[θ5]) Cos[θ6] Sin[θ2 + θ4] + ((L5 + L7 + L3 Cos[θ5]) Sin[θ3] Sin[θ2 + θ4] + (-Cos[θ2] (L1 + L2 Cos[θ3
                                                                 Cos[θ5] (L1 Cos[θ2] + L2 Cos[θ2 + θ
```

MatrixForm=

```
                                                      Cos[θ2 + θ3 + θ4] (0.20478 + 0.1093 Cos[θ5]) Cos[θ6] + ((
-Cos[θ3] (0.1093 + 0.20478 Cos[θ5]) Cos[θ6] Sin[θ2 + θ4] + ((0.20478 + 0.1093 Cos[θ5]) Sin[θ3] Sin[θ2 + θ4] + (-Cos[θ2] (0.425
                                                                       Cos[θ5] (0.425 Cos[θ2] + 0
```

```
-L1 L2 Sin[θ3] (L1 Cos[θ2] + L2 Cos[θ2 + θ3] - L4 Sin[θ2 + θ3 + θ4]) Sin[θ5]
```

```
›s[θ6] + L2 Cos[θ2 + θ3] Cos[θ6] + (L6 - L4 Cos[θ6]) Sin[θ2 + θ3 + θ4]) Sin[θ5] - (L3 + (L5 + L7) Cos[θ5]) Sin[θ2 + θ3 + θ4] Sin[θ6]
]) + L2 Sin[θ2] Sin[θ3] + L4 Cos[θ3] Sin[θ2 + θ4]) Sin[θ5]) Sin[θ6] + Cos[θ2 + θ4] (-((L3 + (L5 + L7) Cos[θ5]) Cos[θ6] Sin[θ3]) +
›3] + (-L4 + L6 Cos[θ6]) Sin[θ2 + θ3 + θ4]) - L3 Cos[θ2 + θ3 + θ4] Sin[θ5] + L6 Cos[θ2 + θ3 + θ4] Sin[θ6]
 Cos[θ5] Cos[θ6] Sin[θ2 + θ3 + θ4] + Cos[θ2 + θ3 + θ4] Sin[θ6]
 Cos[θ2 + θ3 + θ4] Cos[θ6] - Cos[θ5] Sin[θ2 + θ3 + θ4] Sin[θ6]
 -Sin[θ2 + θ3 + θ4] Sin[θ5]

).425 Cos[θ2] Cos[θ6] + 0.392 Cos[θ2 + θ3] Cos[θ6] + (-0.049 - 0.09475 Cos[θ6]) Sin[θ2 + θ3 + θ4]) Sin[θ5] - (0.1093 + 0.20478 Cos
5 + 0.392 Cos[θ3]) + 0.392 Sin[θ2] Sin[θ3] + 0.09475 Cos[θ3] Sin[θ2 + θ4]) Sin[θ5]) Sin[θ6] + Cos[θ2 + θ4] (-((0.1093 + 0.20478 C
.392 Cos[θ2 + θ3] + (-0.09475 - 0.049 Cos[θ6]) Sin[θ2 + θ3 + θ4]) - 0.1093 Cos[θ2 + θ3 + θ4] Sin[θ5] - 0.049 Cos[θ2 + θ3 + θ4] Sin[θ(
 Cos[θ5] Cos[θ6] Sin[θ2 + θ3 + θ4] + Cos[θ2 + θ3 + θ4] Sin[θ6]
 Cos[θ2 + θ3 + θ4] Cos[θ6] - Cos[θ5] Sin[θ2 + θ3 + θ4] Sin[θ6]
 -Sin[θ2 + θ3 + θ4] Sin[θ5]
```

```
                                                          -Cos[θ5] (L6 + Cos[θ6] (-L4 + L2 Sin[θ4] + L1 Sin[θ3 + θ4])) -
(-Cos[θ3] (L5 + L7 + L3 Cos[θ5]) + L4 Sin[θ3] Sin[θ5]) Sin[θ6])  -((L2 Cos[θ4] + L1 Cos[θ3 + θ4]) Cos[θ6]) - (L5 + L7) Cos[θ6] Si
                                                                                         (-L4 + L6 Cos[θ6] + L2 Sin[θ4]
                                                                                                            Cos[θ6] S:
                                                                                                           -Sin[θ5] S
                                                                                                              Cos[θ

[θ5]) Sin[θ2 + θ3 + θ4] Sin[θ6]                                                                       -Cos[θ5] (-0.049 + Cos[ϵ
os[θ5]) Cos[θ6] Sin[θ3]) + (-Cos[θ3] (0.20478 + 0.1093 Cos[θ5]) + 0.09475 Sin[θ3] Sin[θ5]) Sin[θ6])  -((0.392 Cos[θ4] + 0.425
5]
```

(L2 Cos[θ4] + L1 Cos[θ3 + θ4] + (L5 + L7) Sin[θ5]) Sin[θ6]   -Cos[θ5] (L6 + Cos[θ6] (-L4 + L2 Sin[θ4])) - (L2 Cos[θ4] + (L5 + L7)
n[θ5] + Cos[θ5] (-L4 + L2 Sin[θ4] + L1 Sin[θ3 + θ4]) Sin[θ6]   -L2 Cos[θ4] Cos[θ6] - (L5 + L7) Cos[θ6] Sin[θ5] + Cos[θ5] (-L4 + L2
+ L1 Sin[θ3 + θ4]) Sin[θ5]   (-L4 + L6 Cos[θ6] + L2 Sin[θ4]) Sin[θ5]
in[θ5]   Cos[θ6] Sin[θ5]
in[θ6]   -Sin[θ5] Sin[θ6]
5]   Cos[θ5]

6] (-0.09475 + 0.392 Sin[θ4] + 0.425 Sin[θ3 + θ4])) - (0.392 Cos[θ4] + 0.425 Cos[θ3 + θ4] + 0.20478 Sin[θ5]) Sin[θ6]   -Cos[θ5] (-
Cos[θ3 + θ4]) Cos[θ6]) - 0.20478 Cos[θ6] Sin[θ5] + Cos[θ5] (-0.09475 + 0.392 Sin[θ4] + 0.425 Sin[θ3 + θ4]) Sin[θ6]   -0.392 Co
(-0.09475 - 0.049 Cos[θ6] + 0.392 Sin[θ4] + 0.425 Sin[θ3 + θ4]) Sin[θ5]
Cos[θ6] Sin[θ5]
-Sin[θ5] Sin[θ6]
Cos[θ5]

```
Sin[θ5]) Sin[θ6]  Cos[θ5] (-L6 + L4 Cos[θ6]) - (L5 + L7) Sin[θ5] Sin[θ6]  -((L5 + L7) Cos[θ6])  -L6
Sin[θ4]) Sin[θ6]     -((L5 + L7) Cos[θ6] Sin[θ5]) - L4 Cos[θ5] Sin[θ6]      (L5 + L7) Sin[θ6]     0
                          (-L4 + L6 Cos[θ6]) Sin[θ5]                          -L6 Sin[θ6]         0
                              Cos[θ6] Sin[θ5]                                  -Sin[θ6]           0
                             -Sin[θ5] Sin[θ6]                                  -Cos[θ6]           0
                                 Cos[θ5]                                          0               1

·0.049 + Cos[θ6] (-0.09475 + 0.392 Sin[θ4])) - (0.392 Cos[θ4] + 0.20478 Sin[θ5]) Sin[θ6]  Cos[θ5] (0.049 + 0.09475 Cos[θ6]) - 0.
s[θ4] Cos[θ6] - 0.20478 Cos[θ6] Sin[θ5] + Cos[θ5] (-0.09475 + 0.392 Sin[θ4]) Sin[θ6]        -0.20478 Cos[θ6] Sin[θ5] - 0.0947
            (-0.09475 - 0.049 Cos[θ6] + 0.392 Sin[θ4]) Sin[θ5]                                   (-0.09475 - 0.049 Cos[θ6]
                          Cos[θ6] Sin[θ5]                                                             Cos[θ6] Sin[θ5]
                         -Sin[θ5] Sin[θ6]                                                            -Sin[θ5] Sin[θ6]
                             Cos[θ5]                                                                     Cos[θ5]
```

20478 Sin[θ5] Sin[θ6]	-0.20478 Cos[θ6]	0.049
5 Cos[θ5] Sin[θ6]	0.20478 Sin[θ6]	0
) Sin[θ5]	0.049 Sin[θ6]	0
	-Sin[θ6]	0
	-Cos[θ6]	0
	0	1

Body Jacobian for Pen Frame Testing

```
Jb /. {L0 → 0.0892, L1 → 0.425, L2 → 0.392, L3 → 0.1093, L4 → 0.09475, L5 → 0.0825, L6 → -0.049, L7 → 0.12228,
    θ1 → π / 12, θ2 → -π / 2, θ3 → π / 4, θ4 → -π / 3, θ5 → -π, θ6 → π / 6} // MatrixForm // N
```

MatrixForm=

$$\begin{pmatrix} -0.0675146 & -0.823576 & -0.523056 & -0.131056 & -0.177345 & 0.049 \\ -0.0675146 & -0.253145 & 0.047375 & 0.047375 & 0.10239 & 0. \\ -0.403356 & 0. & 0. & 0. & 0.0245 & 0. \\ 0.707107 & 0. & 0. & 0. & -0.5 & 0. \\ -0.707107 & 0. & 0. & 0. & -0.866025 & 0. \\ 0. & -1. & -1. & -1. & 0. & 1. \end{pmatrix}$$