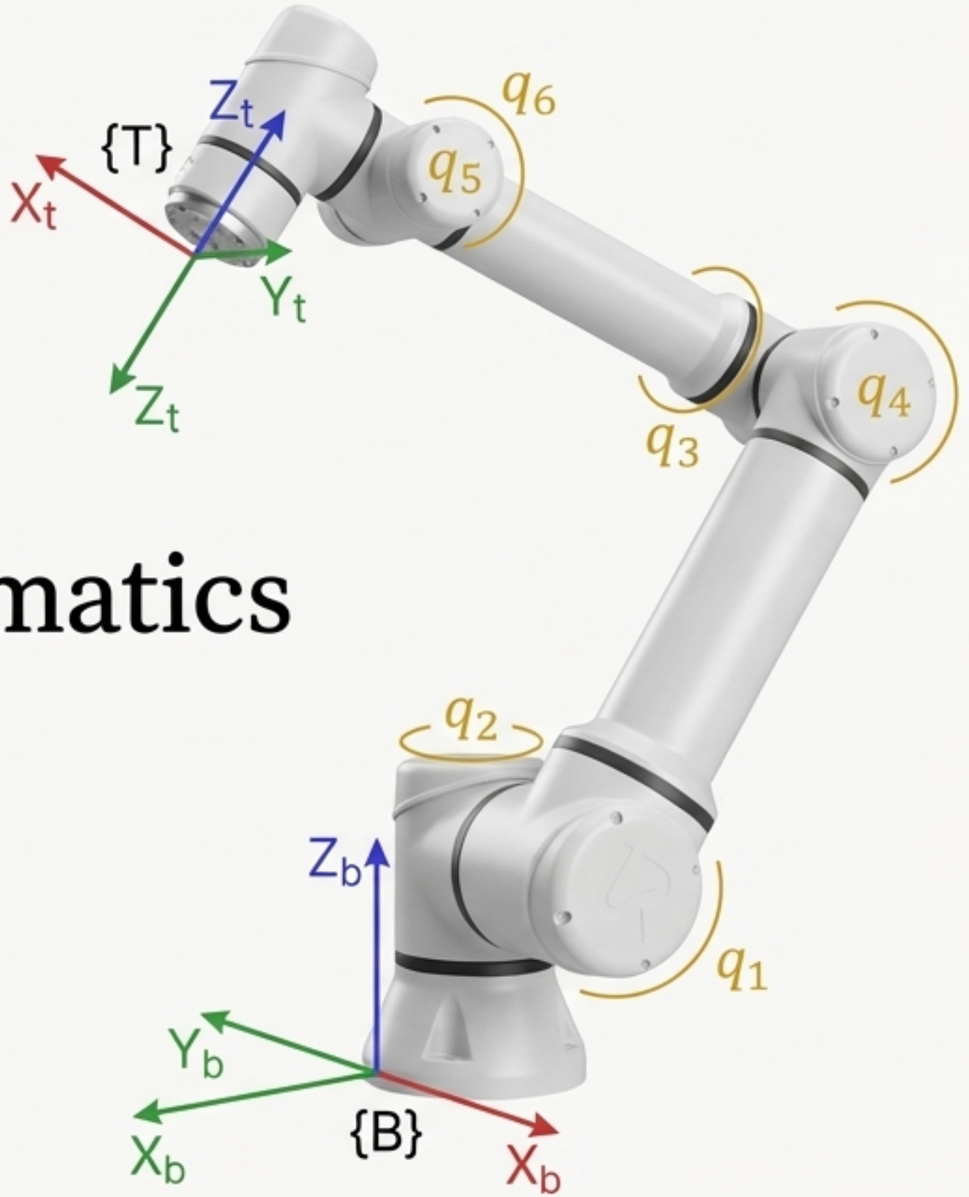
ME 5751: 3D Robot Kinematics

From Joints to Job-site: The Logic of Forward Kinematics



Lecture 2: Forward Kinematics of Serial Manipulators

Recap: The Language of a Single Pose

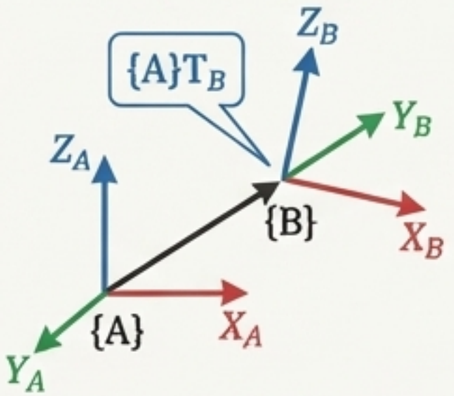
In our last lecture, we established that a complete pose (position and orientation) of a frame {B} relative to a frame {A} can be described by a single 4x4 mathematical object: the Homogeneous Transformation Matrix, ${}^{A}_{B}T$.

This matrix unifies a 3x3 rotation matrix (${}^{A}_{B}R$) and a 3x1 position vector (${}^{A}P_{BORG}$) into one powerful tool.

$$ {}^{A}_{B}T = \begin{bmatrix} {}^{A}_{B}R & {}^{A}P_{BORG} \\ \mathbf{0} & 1 \end{bmatrix} $$

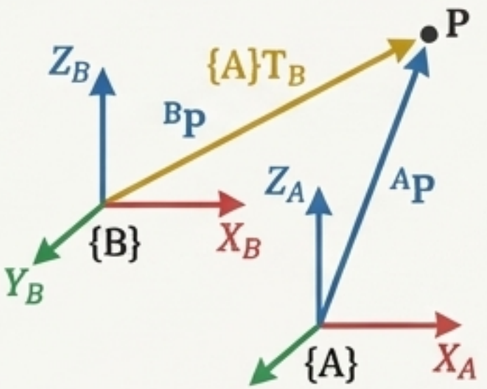
$$ \begin{bmatrix} \overbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}^{\text{Rotation } ({}^{A}_{B}R)} & \overbrace{\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}}^{\text{Position } ({}^{A}P_{BORG})} \\ \underbrace{\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}}_{1x3} & \underbrace{\begin{bmatrix} 1 \end{bmatrix}}_{1x1} \end{bmatrix} $$

1. A Description



It provides a full description of the pose (position and orientation) of frame {B} relative to frame {A}.

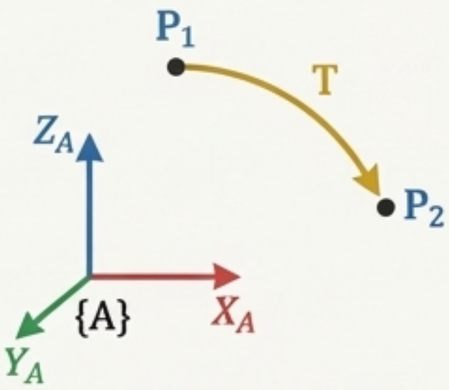
2. A Mapping



It is a function that maps the coordinates of a point from its description in frame {B} to its description in frame {A}.

$$ {}^{A}P = {}^{A}_{B}T \cdot {}^{B}P $$

3. An Operator



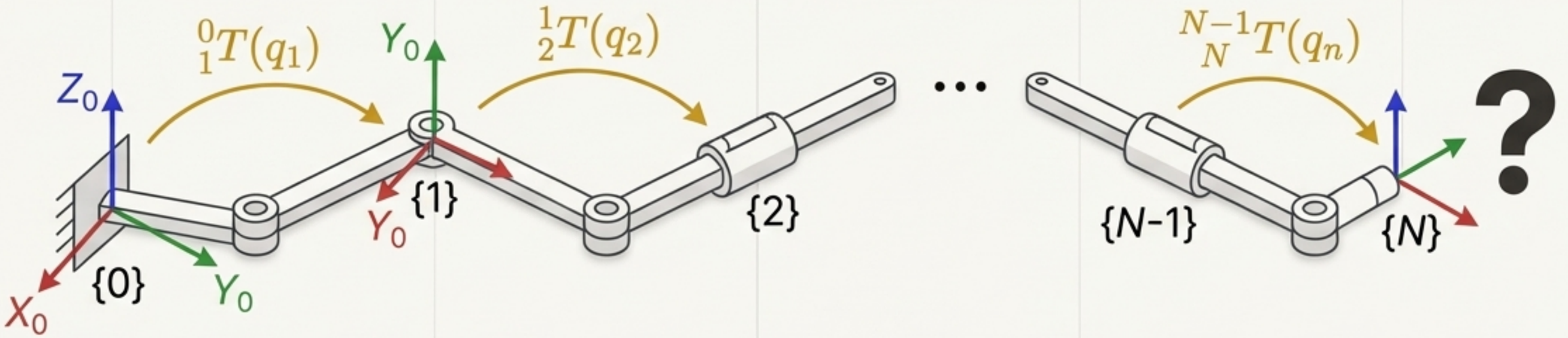
It is an operator that takes a point P1 and moves it to a new point P2 within the same frame {A}.

$$ {}^{A}P_2 = T \cdot {}^{A}P_1 $$

The Next Challenge: From a Single Link to a Kinematic Chain

A serial manipulator is a sequence of links connected by joints. The pose of each link can be described relative to the previous one using a transformation matrix.

The Forward Kinematics Problem: Given the set of joint variables ($q = [q_1, q_2, \dots, q_n]$), how do we systematically compute the final pose of the end-effector frame {*N*} relative to the base frame {0}?



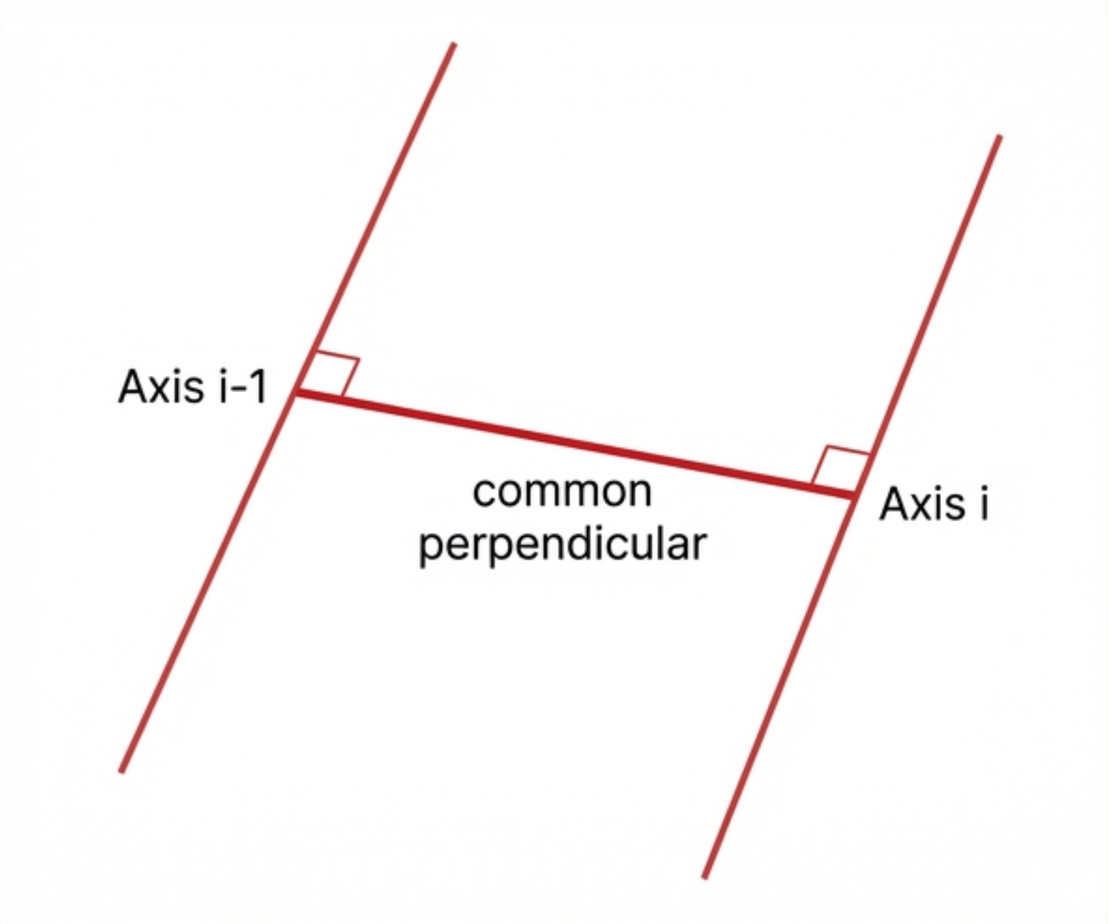
$$ {}^0_NT(q) = {}^0_1T(q_1)\,{}^1_2T(q_2)\cdots{}^{N-1}_NT(q_n) $$

The Method: The Denavit-Hartenberg Convention

To solve the forward kinematics problem for any serial robot, we need a systematic way to assign coordinate frames to each link.

The Denavit-Hartenberg (DH) convention is a powerful, standardized algorithm that achieves this. Its key insight is that the transformation between any two adjacent link frames can be described by just **four parameters**.

This reduces a complex spatial problem to a repeatable procedure and a concise set of parameters.



Anatomy of a Link: The Link Parameters

Two of the four DH parameters describe the fixed geometry of the link connecting two joint axes.

a_{i-1} : Link Length

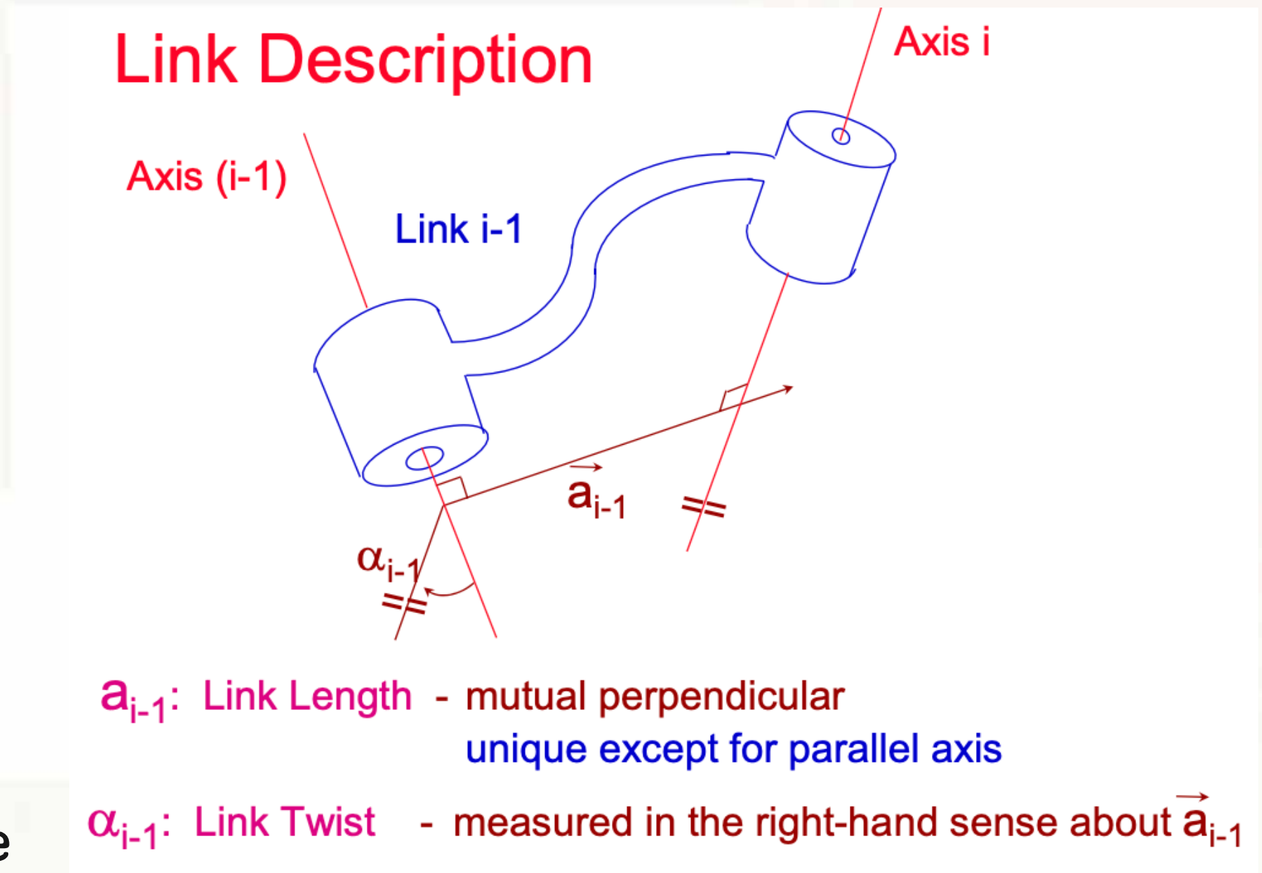
The distance from axis $\hat{Z}_{i-1}$ to $\hat{Z}_i$, measured along the common perpendicular, $\hat{X}_{i-1}$.

This is the shortest distance between the two joint axes.

α_{i-1} : Link Twist

The angle from axis $\hat{Z}_{i-1}$ to $\hat{Z}_i$, measured about the common perpendicular, $\hat{X}_{i-1}$.

This describes how 'twisted' the link is.



Anatomy of a Link: The Joint Parameters

The other two parameters describe the relative position of one link with respect to another at the joint connection.

d_i : Link Offset

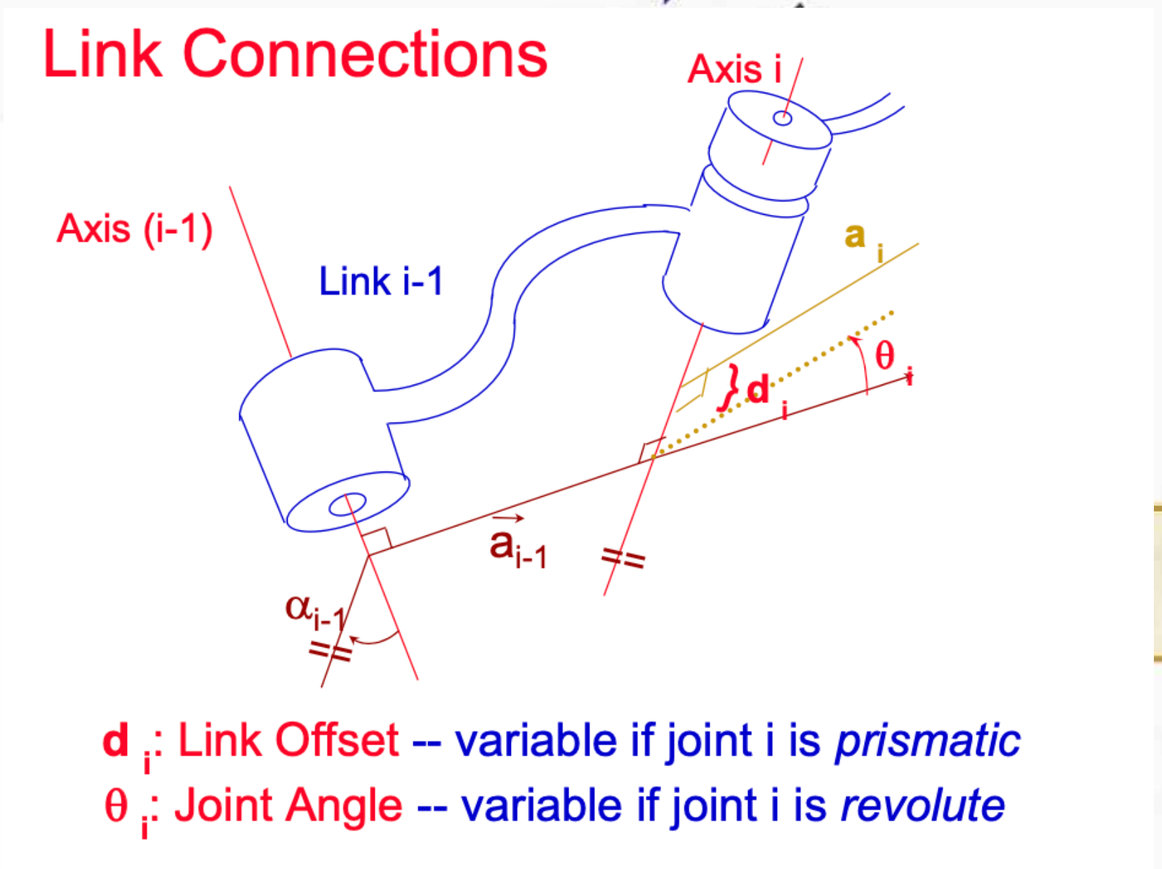
The distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along the $\hat{Z}_i$ axis.

This is the **joint variable** for a **prismatic** joint.

θ_i : Joint Angle

This is the **joint variable** for a **revolute** joint.

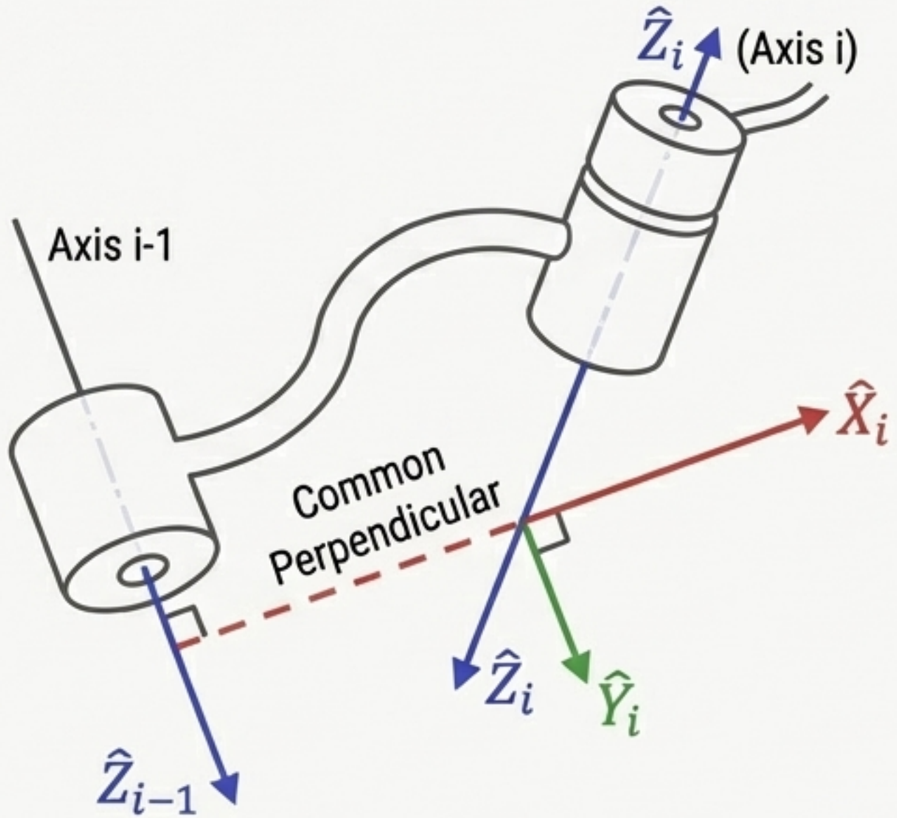
The angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about the Z_i axis.



The Rules of the Road: A Step-by-Step Algorithm for Frame Assignment

To derive the DH parameters, we follow a precise algorithm to attach a coordinate frame $\{i\}$ to each link i.

1. **Identify Joint Axes:** Locate and draw infinite lines for all joint axes, $\hat{Z}_0$ through $\hat{Z}_{n-1}$.
2. **Establish Base Frame {0}:** Place the origin of {0} anywhere on the $\hat{Z}_0$ axis. Set $\hat{X}_0$ and $\hat{Y}_0$ conveniently (often aligned with {1} at the home position).
3. **Assign Frames {i} for i = 1 to n-1:**
   - The $\hat{Z}_i$ axis lies along the axis of joint i+1.
   - The origin of {i} is at the intersection of $\hat{Z}_i$ and the common perpendicular between $\hat{Z}_{i-1}$ and $\hat{Z}_i$.
   - The $\hat{X}_i$ axis points along the common perpendicular from $\hat{Z}_{i-1}$ to $\hat{Z}_i$.
   - The $\hat{Y}_i$ axis is set to complete a right-handed coordinate frame.
4. **Assign End-Effector Frame {N}:** Choose frame {N} to be convenient for the task, often with $\hat{Z}_N$ aligned with the tool's approach direction.



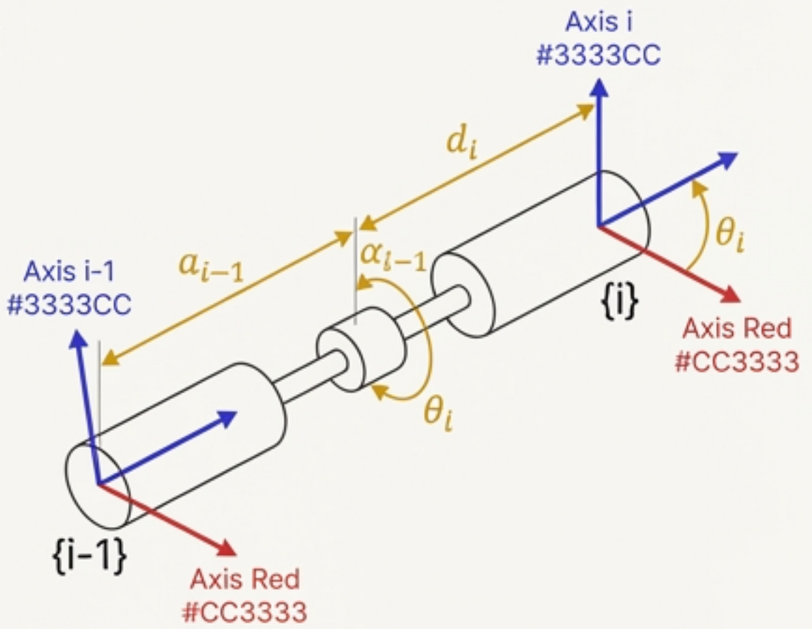
The Universal Building Block: The DH Transformation Matrix

Once the DH parameters $(a_{i-1}, \alpha_{i-1}, d_i, \theta_i)$ are determined, they populate a standard transformation matrix that represents the pose of frame {i} relative to frame {i-1}.

This transformation is a composition of four basic motions:

$$^{i-1}_{i}T = Rot_Z(\theta_i)\, Trans_Z(d_i)\, Trans_X(a_{i-1})\, Rot_X(\alpha_{i-1})$$

$$^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i\, c\alpha_{i-1} & s\theta_i\, s\alpha_{i-1} & a_{i-1}\, c\theta_i \\ s\theta_i & c\theta_i\, c\alpha_{i-1} & -c\theta_i\, s\alpha_{i-1} & a_{i-1}\, s\theta_i \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



The complete forward kinematics is the product of these matrices for each link:

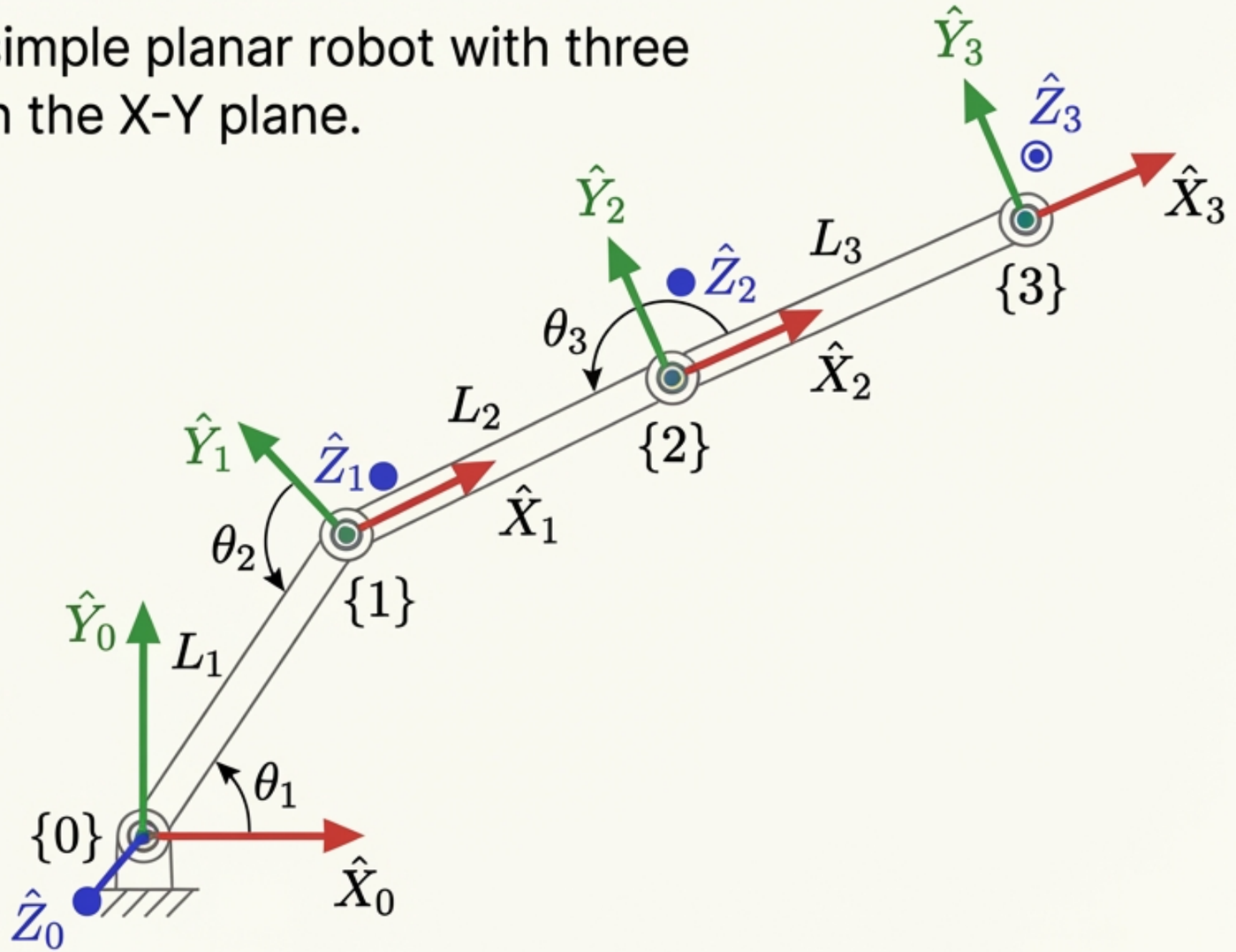
$$^{0}_{N}T = {}^{0}_{1}T\ {}^{1}_{2}T \cdots {}^{N-1}_{N}T.$$

Guided Practice: The Planar 3R Manipulator

Let's apply the DH algorithm to a simple planar robot with three revolute joints. All motion occurs in the X-Y plane.

Step 1: Assign Frames

We follow the DH rules to assign a coordinate frame to the base and each of the three links. Since all Z-axes are parallel and point out of the page, the link twists (α) are all zero. The origins are placed at the joint axes.



From Frames to Formulas: The 3R DH Table and Kinematics

Step 2: Derive DH Parameters

From the assigned frames, we measure the four DH parameters for each link and compile them into a table.

Link i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1^*
2	L_1	0	0	θ_2^*
3	L_2	0	0	θ_3^*

*Asterisk denotes the joint variable

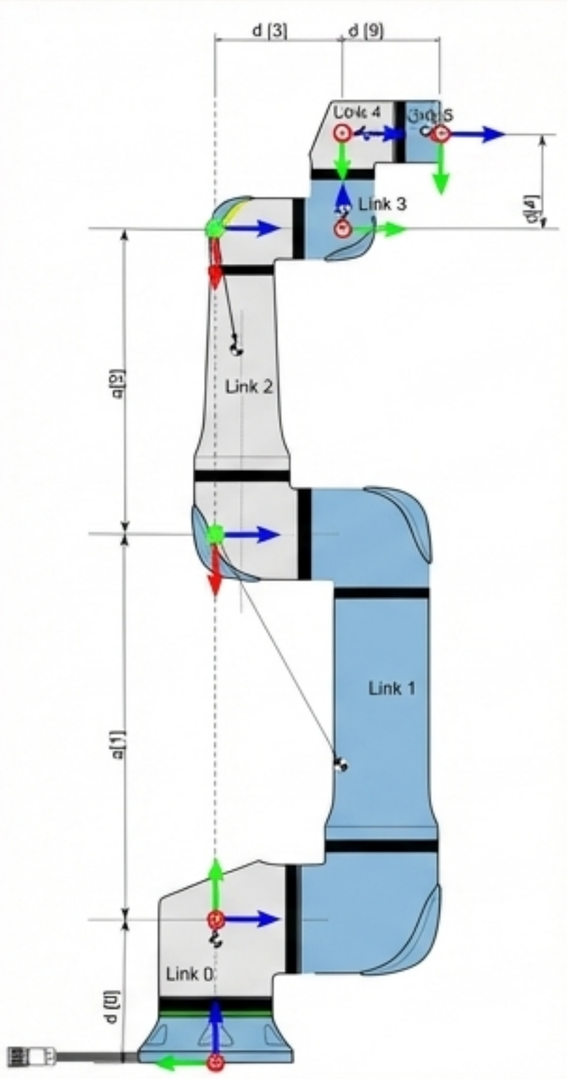
Step 3: Compute the Kinematics

Plugging these parameters into the DH transformation matrix template for each link and multiplying them together gives the final pose of the end-effector.

$$ {}^0_3T = {}^0_1T(\theta_1)\,{}^1_2T(\theta_2)\,{}^2_3T(\theta_3) $$

$$ {}^0_3T = \begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1 + L_2c_{12} \\ s_{123} & c_{123} & 0 & L_1s_1 + L_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Industrial Case Study: The UR5e Robot Arm



- The same principles apply to complex 6-DOF industrial robots like the Universal Robots UR5e.
- For established robots, manufacturers often provide the DH parameters, saving us the frame assignment step.
- **Note:** There are several variations of the DH convention (e.g., Standard/Craig vs. Modified). It's crucial to use the transformation matrix that matches the convention used for the table. RoboDK and some manufacturers use the Modified DH convention (DHM).

Table 1: Standard DH Parameters for UR5e

Link i	a_i	alpha_i	d_i	theta_i
Link 1	0	-πi/2	d1	theta1*
Link 2	a2	0	0	theta2*
Link 3	a3	0	0	theta3*
Link 4	0	-πi/2	d4	theta4*
Link 5	0	πi/2	d5	theta5*
Link 6	0	0	d6	theta6*

Table 2: Modified DH (DHM) Parameters for UR5e (RoboDK)

Link i	a{i-1	alpha_{i-1}	d_i	theta_i
Link 1	0	0	d1	theta1*
Link 2	0	-πi/2	0	theta2*
Link 3	a2	0	0	theta3*
Link 4	a3	0	d4	theta4*
Link 5	0	-πi/2	d5	theta5*
Link 6	0	pi/2	d6	theta6*

Putting Theory into Code: Calculating the UR5e Pose

Using the Modified DH parameters and a specific set of joint angles, we can write a simple program to compute the resulting end-effector pose.

Inputs:
Joint Angles (q): `[0, -90, -90, 0, 90, 0]` degrees.

DHM Parameters: Using the table from the previous slide.

```python
# Python Code for UR5e Forward Kinematics
import numpy as np

def dh_transform(a, alpha, d, theta):
    """Calculates the Modified DH transformation matrix."""
    c_theta = np.cos(theta)
    s_theta = np.sin(theta)
    c_alpha = np.cos(alpha)
    s_alpha = np.sin(alpha)

    return np.array([
        [c_theta, -s_theta, 0, a],
        [s_theta * c_alpha, c_theta * c_alpha, -s_alpha, -s_alpha * d],
        [s_theta * s_alpha, c_theta * s_alpha, c_alpha, c_alpha * d],
        [0, 0, 0, 1]
    ])

def forward_kinematics_ur5e(q_deg):
    """Computes the end-effector pose for UR5e using Modified DH."""
    # Modified DH Parameters for UR5e (from table)
    # a = [0, 0, a2, a3, 0, 0]
    # alpha = [0, -pi/2, 0, 0, -p/2, pi/2]
    # d = [d1, 0, 0, d4, d5, d6]
    # ... parameter definitions (assume predefined for this snippet)

    q = np.radians(q_deg) # Convert to radians
    T_total = np.eye(4) # Initialize total transformation

    # Loop through joints to compute total transformation
    for i in range(6):
        Ti = dh_transform(a[i], alpha[i], d[i], q[i])
        T_total = T_total @ Ti

    return T_total

# Example joint angles
q_input = np.array([0, -90, -90, 0, 90, 0])
T_0_6 = forward_kinematics_ur5e(q_input)

print(T_0_6)
```

The calculated end-effector pose matrix ${}^{0}_{6}T$ is:

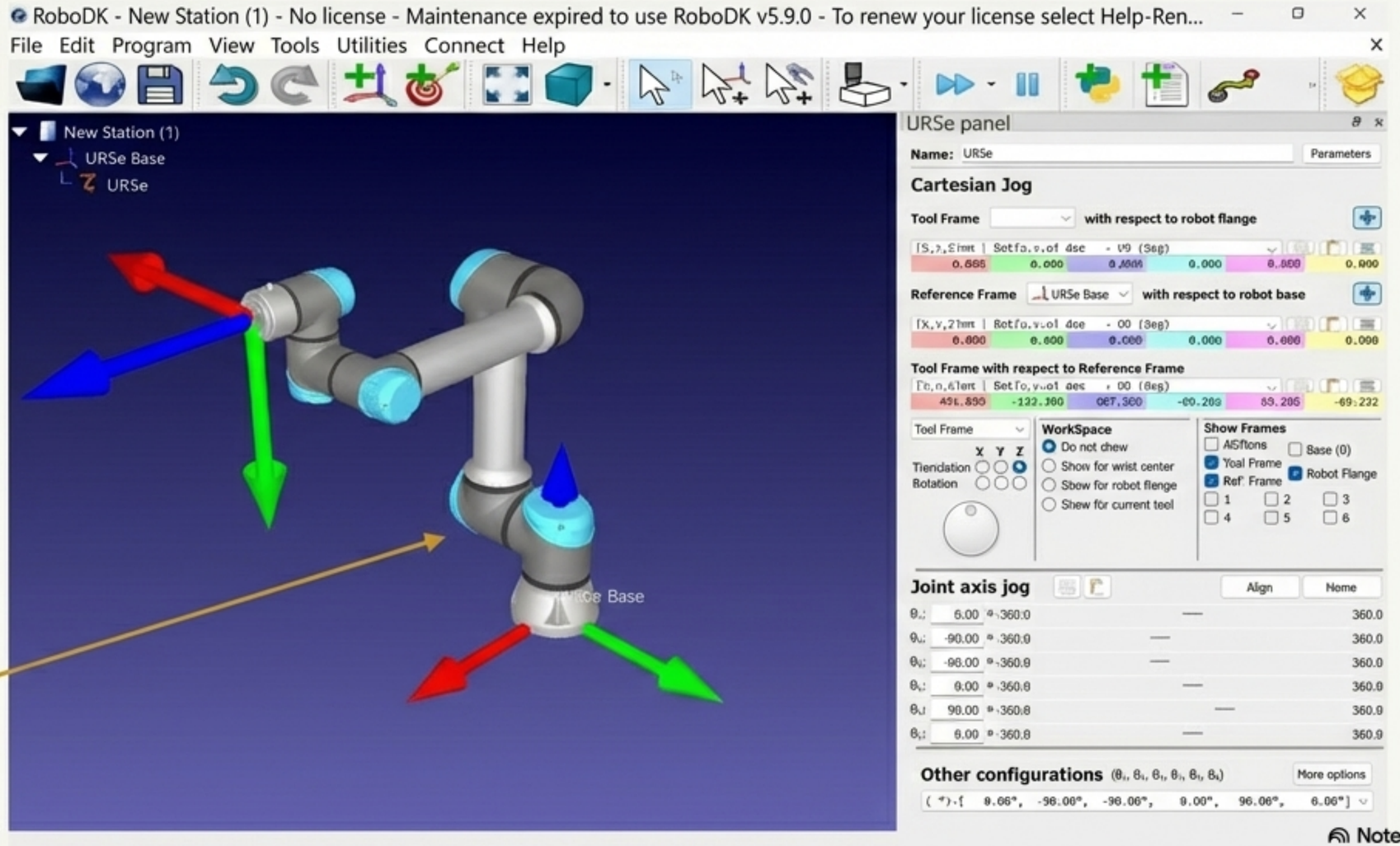
$$\begin{bmatrix} 0. & 0. & 1. & 491.85 \\ -1. & 0. & 0. & -133.30 \\ 0. & -1. & 0. & 687.20 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

Verification in Simulation: Introduction to RoboDK

How can we be confident in our calculation? We can verify it using professional robot simulation software like RoboDK.

RoboDK is a powerful environment for offline programming and simulation. It builds its internal model of the robot using a kinematic description, such as the Modified DH parameters we just used.

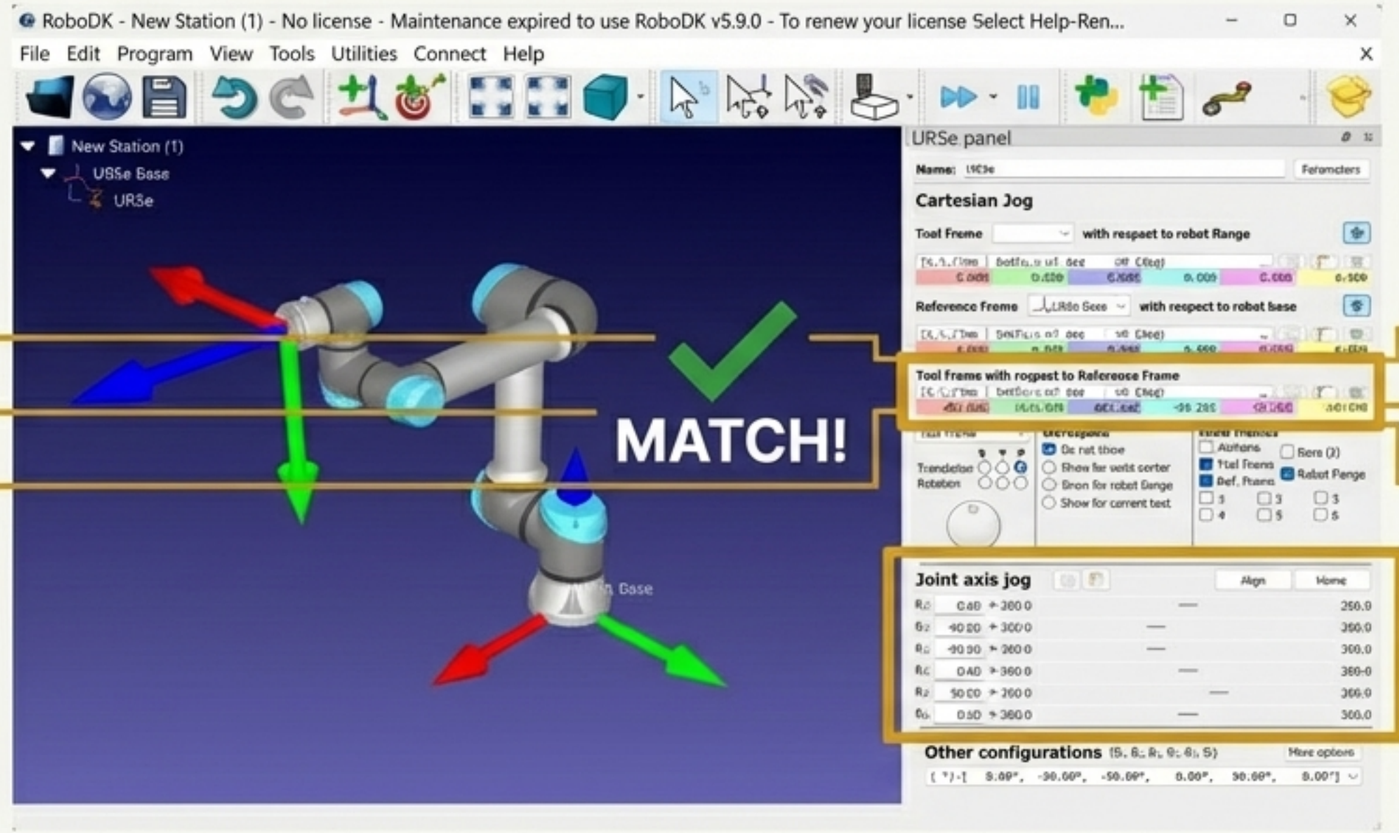
The virtual robot's motion is governed by its kinematic model (DHM parameters).



Closing the Loop: Theory Matches Reality

By inputting the exact same joint angles from our Python script into RoboDK, we can directly compare the resulting end-effector pose.

$$\begin{bmatrix} 491.85 \\ -133.30 \\ 687.20 \end{bmatrix}$$



X=491.850
Y=-133.300
Z=687.200

Input Angles

The calculated pose and the simulated pose are identical.
The Denavit-Hartenberg method works.

You Can Now Predict a Robot's Pose. What's Next?

Summary of Key Learnings

- **Forward Kinematics**: The problem of finding the end-effector pose from known joint angles.
- **DH Convention**: A systematic 4-parameter method for modeling any serial manipulator.
- **Verification**: The process of using code and simulation to validate theoretical models.

The Next Question

We have answered: “Given the joint angles, what is the pose?”. But what about the more common robotics problem: “I have a desired pose for my tool. What joint angles do I need to achieve it?”

Next Lecture: The Inverse Kinematics Problem