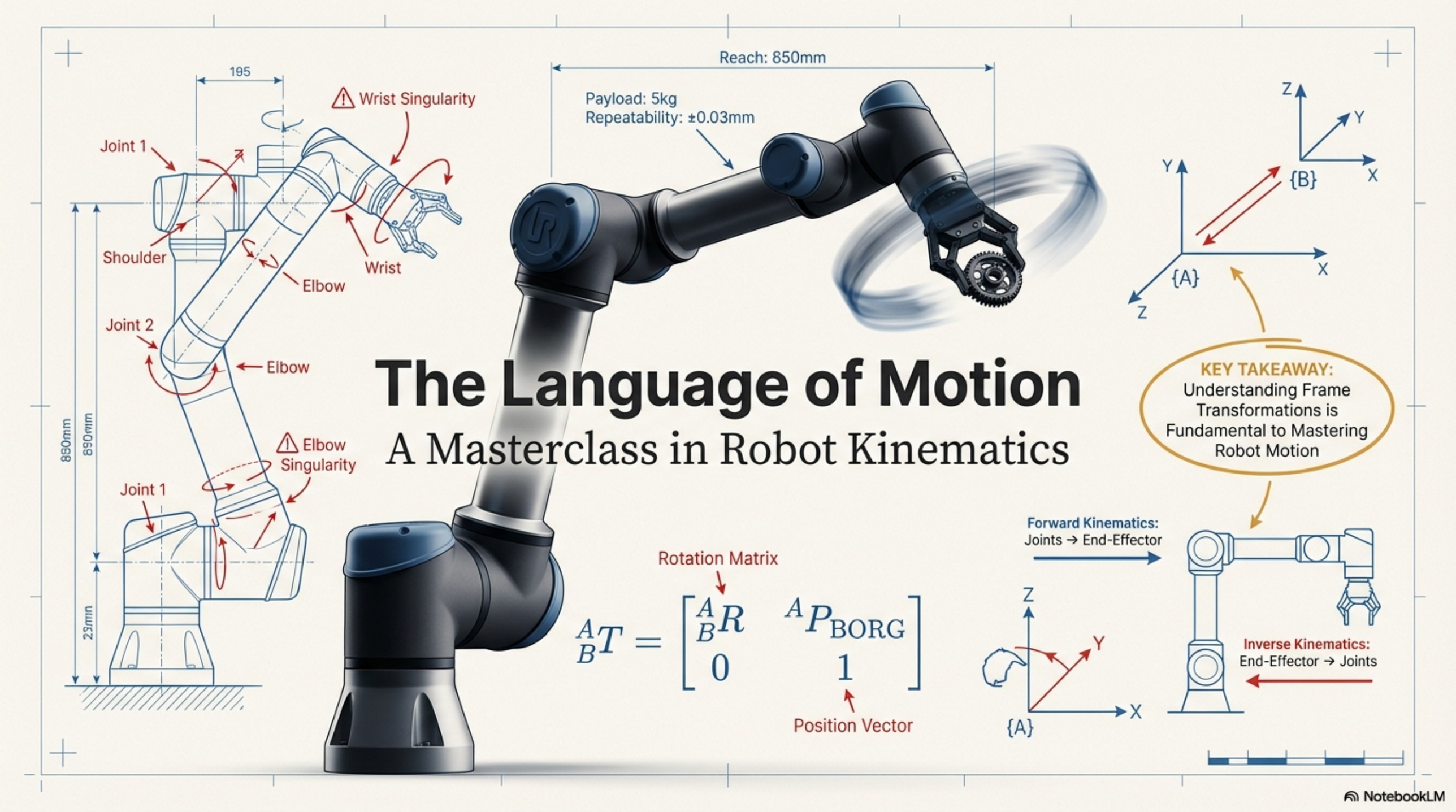
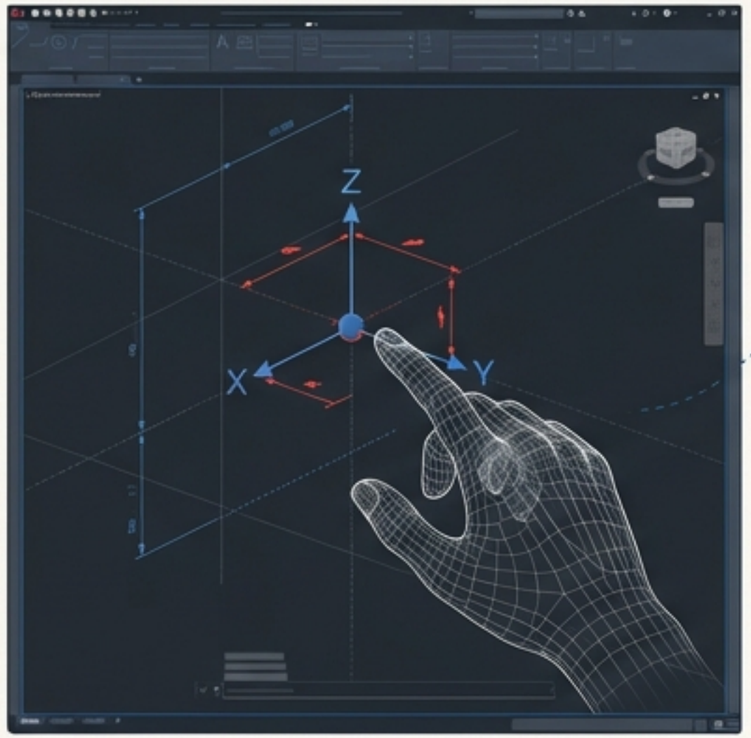
The Language of Motion

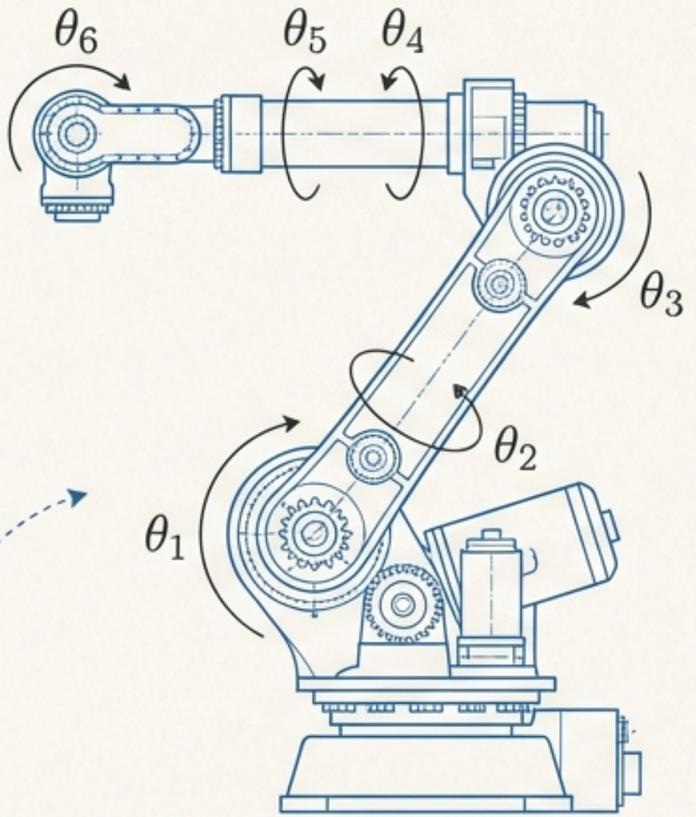
A Masterclass in Robot Kinematics





The Intent: Cartesian Space
(x, y, z, *roll*, *pitch*, *yaw*)

How do we translate intent into action?



The Reality: Joint Space
(θ_1, θ_2, ..., θ_6)

The Vocabulary of Space: Describing Orientation

The Concept

Before we can command a robot, we need a rigorous, mathematical way to describe an object's orientation in 3D space.

The Tool: The Rotation Matrix

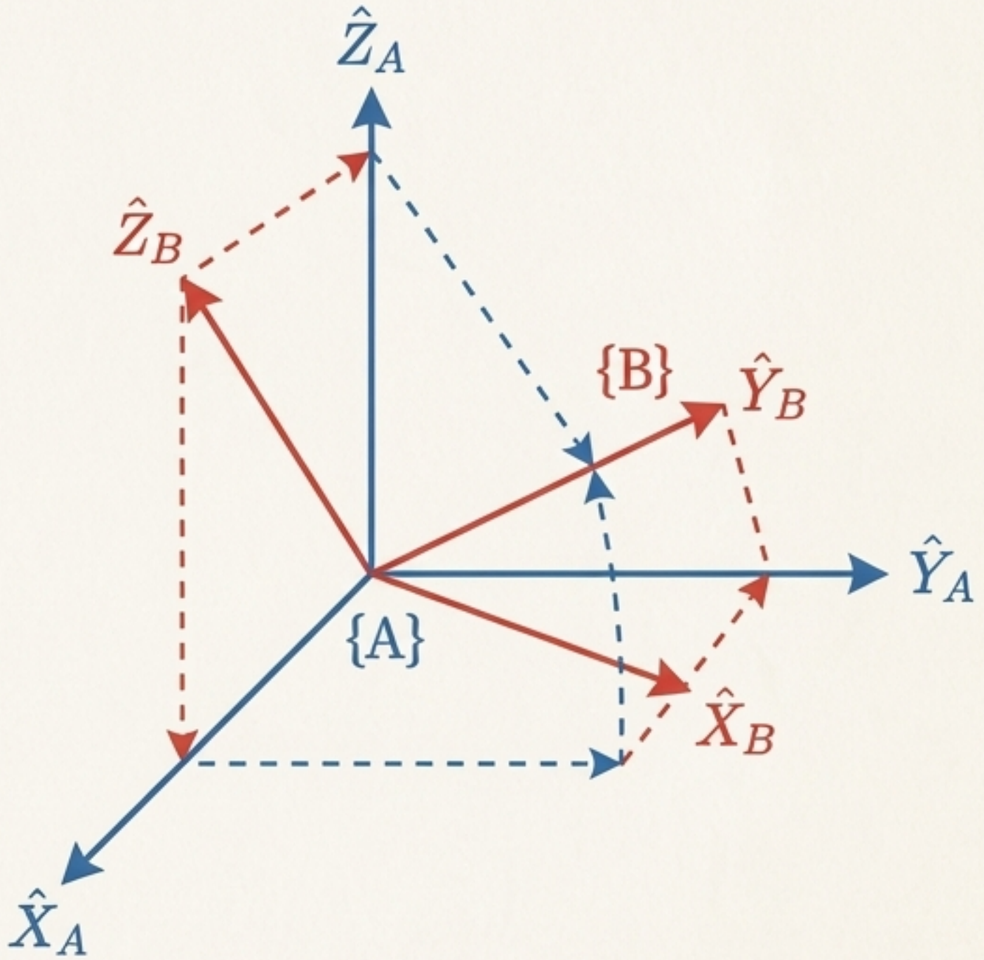
Coordinate frames serve as reference points. The Rotation Matrix describes a body-attached frame {B} relative to a reference frame {A}.

$$ {}^{A}_{B}R = \begin{bmatrix} {}^{A}\hat{X}_B & {}^{A}\hat{Y}_B & {}^{A}\hat{Z}_B \end{bmatrix} $$

The columns are the unit vectors of {B}'s axes expressed in {A}'s coordinates.

Key Property: A rotation matrix is orthonormal. Its inverse is simply its transpose: ${}^{B}_{A}R = ({}^{A}_{B}R)^{-1} = ({}^{A}_{B}R)^{T}$.

The Visual

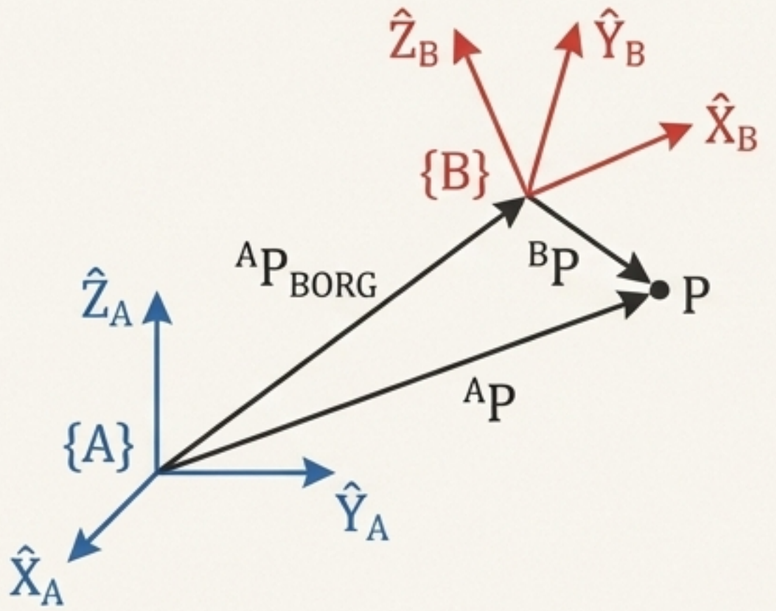


The Vocabulary of Space: Describing Pose

Describing orientation isn't enough. We need to specify an object's position and orientation simultaneously. This complete description is called a **pose**.

Rotation Matrix (3x3) Position Vector (3x1)

$$T = \left[\begin{array}{c|c} {}^{A}_{B}R & {}^{A}P_{BORG} \\ \hline \text{Perspective} & \text{Scale} \\ 0 \quad 0 \quad 0 & 1 \end{array}\right]$$

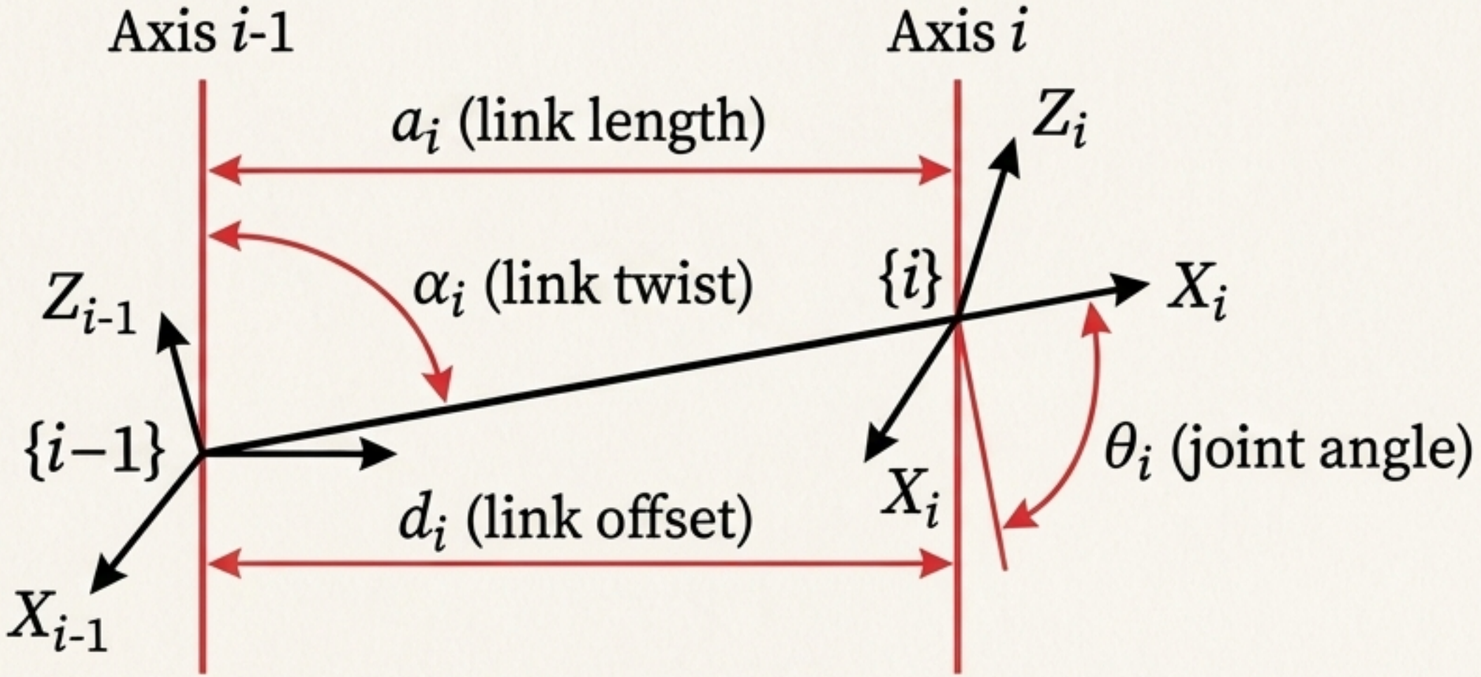


The Application: ${}^{A}P = {}^{A}_{B}T \cdot {}^{B}P$

- 1. A description of frame {B} relative to {A}.
- 2. A mapping of a point from coordinates {B} to {A}.
- 3. An operator that moves a point within a single frame.

A Universal Grammar for Robots: Denavit-Hartenberg

Problem: Every robot is built differently. Must we create a custom mathematical model from scratch for each one? No. We need a universal convention.

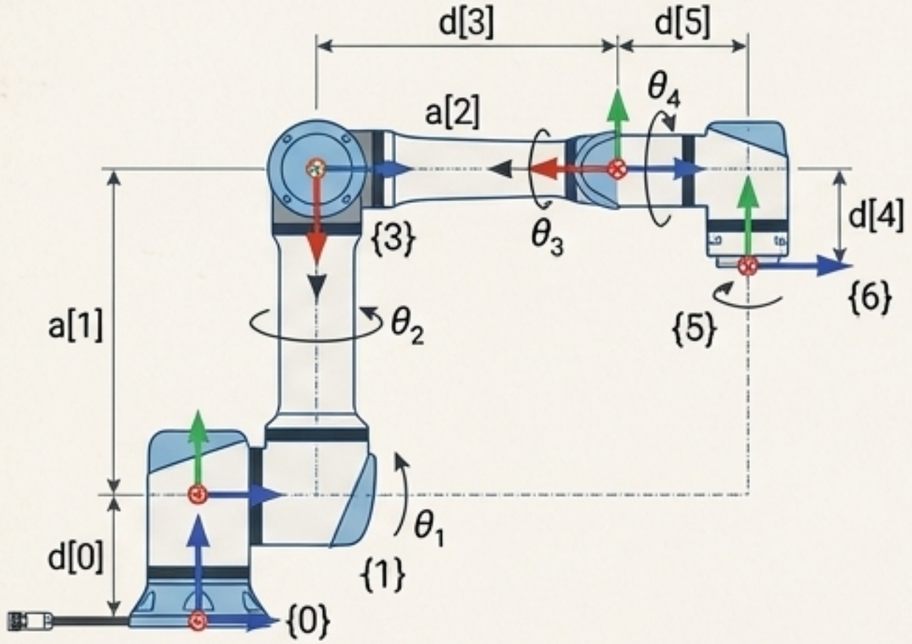


a_i: link length
α_i: link twist
d_i: link offset
θ_i: joint angle

Key Takeaway: This simple set of four numbers per joint is sufficient to describe the kinematics of *any* serial-chain robot.

The Forward Path: From Joints to Pose (Forward Kinematics)

If we know all the robot's joint angles, where exactly is its tool in space? This is Forward Kinematics (FK), essential for simulation, control, and safety.



UR5e Denavit-Hartenberg Parameters

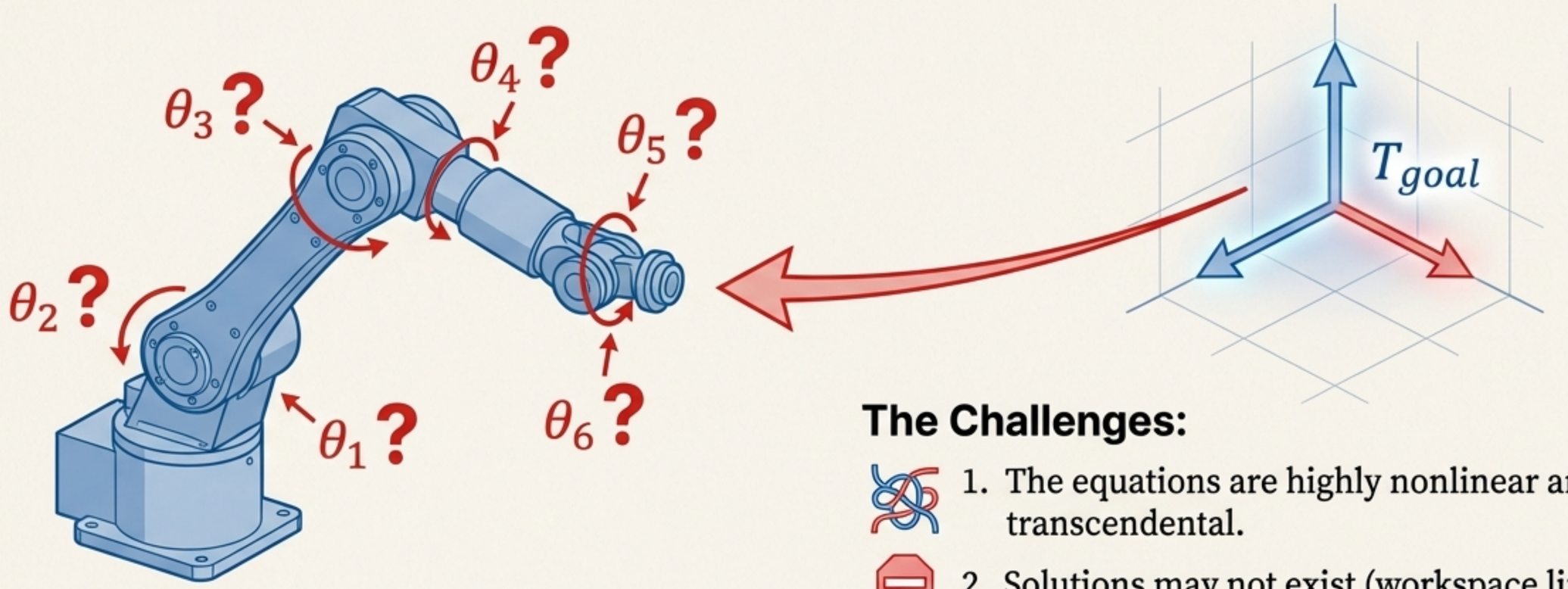
Joint i	a_{i-1} [m]	α_{i-1} [deg]	d_i [m]	θ_i [deg]
1	0	0	0.1625	θ_1
2	-0.425	0	0	θ_2
3	-0.3922	0	0	θ_3
4	0	90	0.1333	θ_4
5	0	-90	0.0997	θ_5
6	0	0	0.0996	θ_6

The FK Equation: $${}^{0}_{6}T = {}^{0}_{1}T(\theta_1) \cdot {}^{1}_{2}T(\theta_2) \cdot {}^{2}_{3}T(\theta_3) \cdot {}^{3}_{4}T(\theta_4) \cdot {}^{4}_{5}T(\theta_5) \cdot {}^{5}_{6}T(\theta_6)$$

Forward Kinematics translates the robot's internal language (joint angles) into the language of human intent (Cartesian pose).

The Reverse Path: From Pose to Joints (Inverse Kinematics)

Now for the most critical part of our translator. We have a desired tool pose in the world, T_{goal}. What set of joint angles (θ_1, ..., θ_6) will achieve this pose?



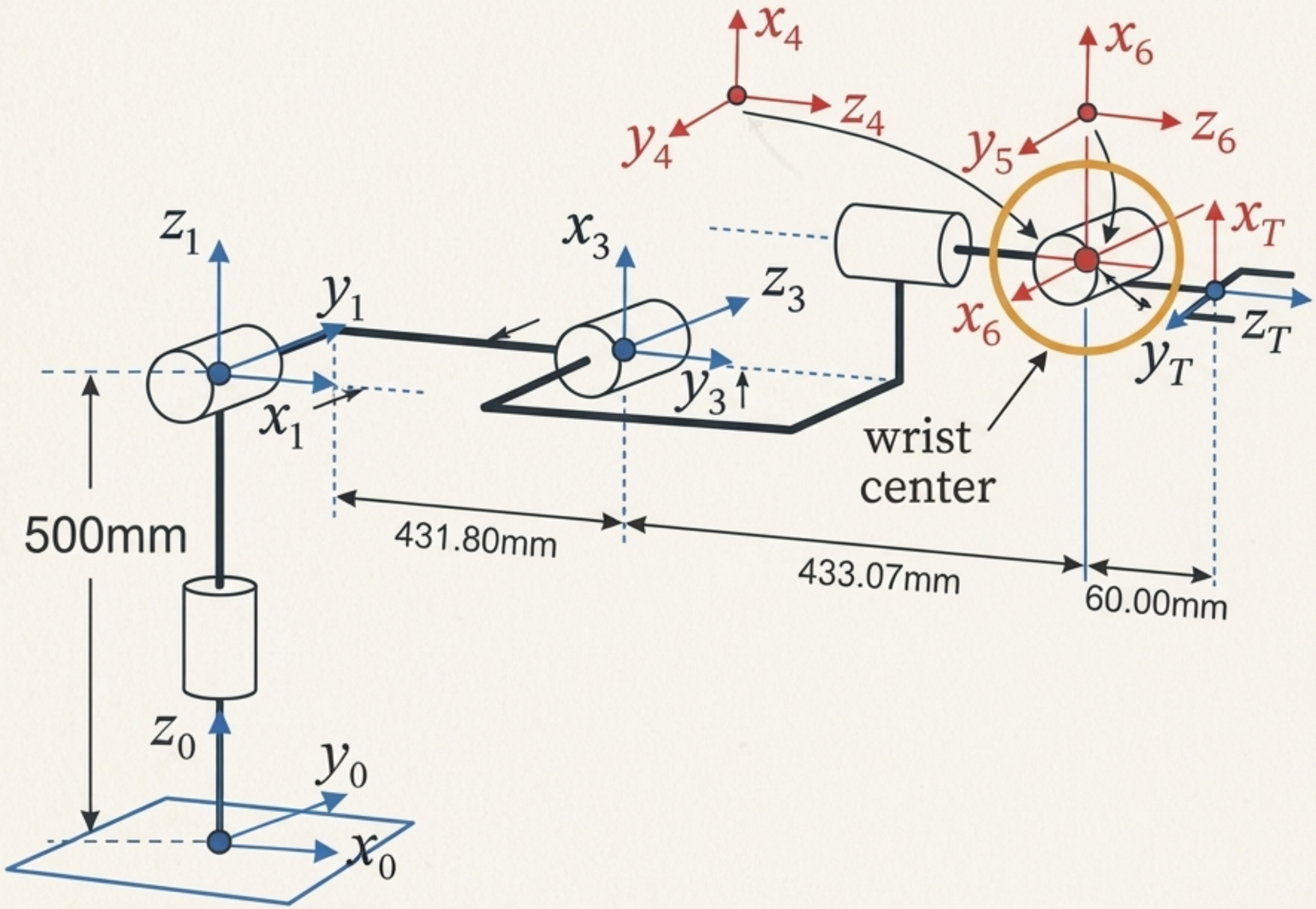
The Tool: Inverse Kinematics (IK)

IK is the process of solving for the joint vector $\boldsymbol{\theta}$ given a desired end-effector transformation matrix ${}^{0}_{N}T$.

The Challenges:

1. The equations are highly nonlinear and transcendental.
2. Solutions may not exist (workspace limits).
3. Multiple solutions often exist for a single pose.
4. Closed-form (analytic) solutions are rare and highly desirable for speed and predictability.

An Elegant Solution: The PUMA 560



Application:

PUMA 560, a classic 6-DOF industrial robot.

The Key Design Feature:

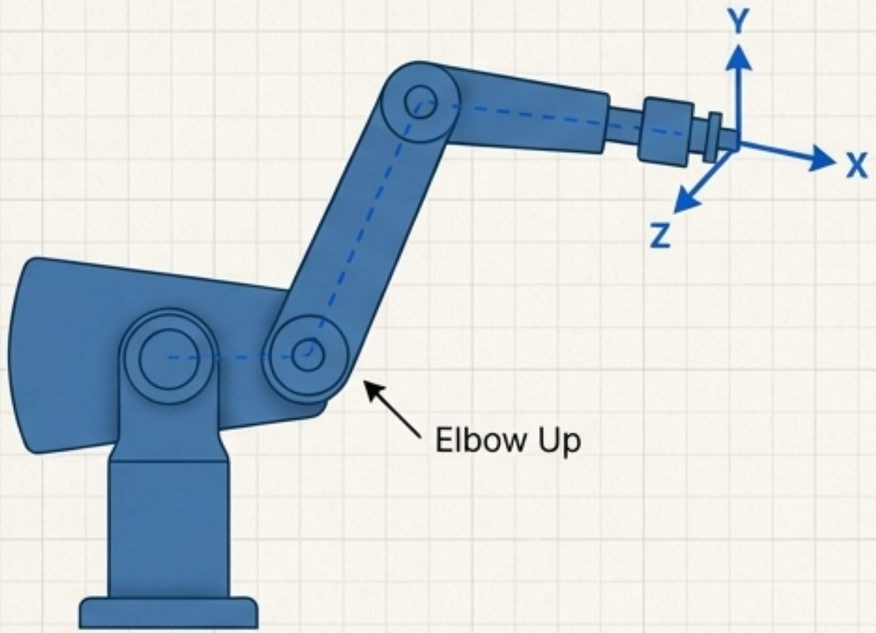
Its design is special because the last three joint axes (4, 5, and 6) intersect at a single point, forming a **spherical wrist**.

The Benefit:

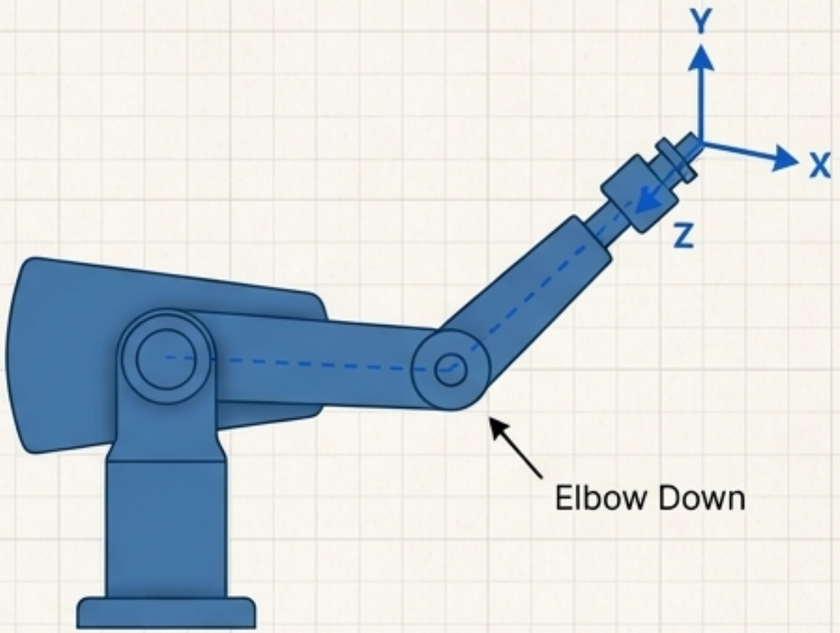
This specific geometric arrangement is a crucial simplification. It **decouples** the positioning problem (first 3 joints) from the orientation problem (last 3 joints). This allows for a rare and highly efficient **analytic (closed-form) solution** to the inverse kinematics problem, as first shown by Pieper. This means the joint angles can be calculated directly with formulas, without slow iterative numerical methods.

The Challenge of Multiple Realities

For a single target pose, there are often multiple valid joint configurations. For the PUMA 560, there can be up to eight. Which one should the robot choose?



Solution 1: Elbow Up



Solution 2: Elbow Down

Common Strategies for Choosing a Solution:

- **Closest Solution:** Choose the configuration that requires the minimum total joint movement from the current position.
- **Obstacle Avoidance:** Select a solution that does not result in a collision with objects in the workspace.
- **Singularity Avoidance:** Pick a configuration that is furthest from a singular pose (to be explained next).

The Forbidden Zones: Singularities

Problem: Are there certain poses where the robot gets 'stuck' or loses its ability to move in a particular direction? Yes. These are called **singularities**.

The Tool: The Jacobian Matrix

$$v = J(\theta) \cdot \dot{\theta}$$

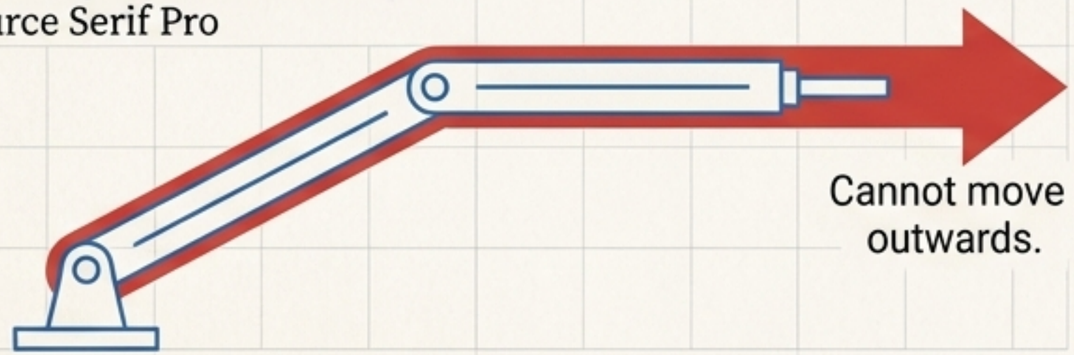
A singularity occurs when the Jacobian matrix becomes non-invertible (i.e., its determinant is zero).

Physical Meaning

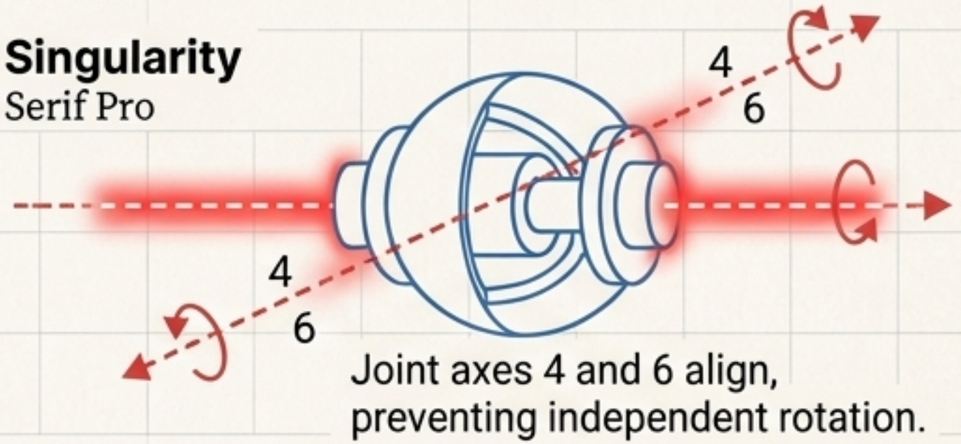
At a singularity, the robot loses one or more degrees of freedom. There are directions in Cartesian space where the end-effector cannot move, no matter how the joints move.

Visual Examples:

Workspace Boundary
Source Serif Pro

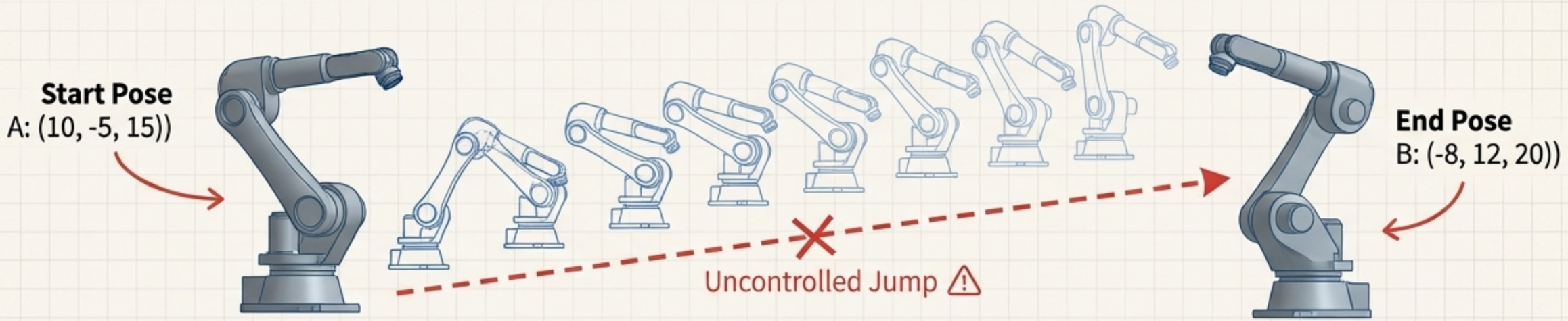


Wrist Singularity
Source Serif Pro



The Journey, Not Just the Destination

We've used IK to find the start and end joint angles for a move. But simply jumping between these configurations would be uncontrolled and dangerous. How do we move between them smoothly, predictably, and safely?



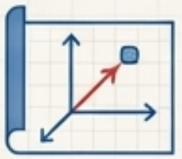
The Solution: Motion Planning & Trajectory Generation

Define **Trajectory Generation** as the process of computing a time-sequenced path of intermediate robot poses that connect the start and end points.

This process defines not just the path, but also the velocity and acceleration along that path for a smooth motion.

Two Fundamental Approaches:

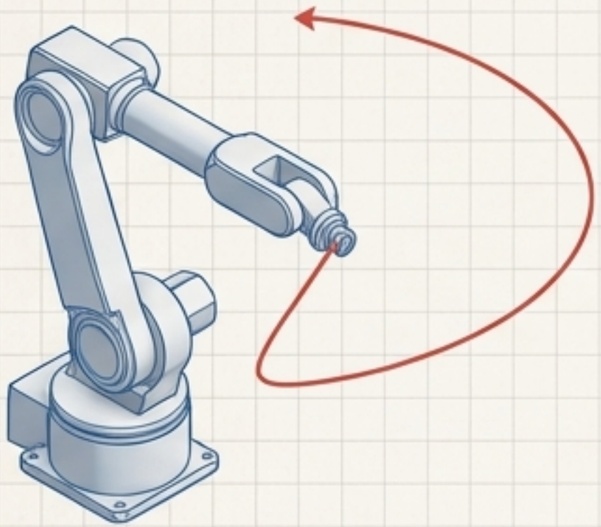
Joint Space Interpolation:
The controller calculates a smooth path for each joint angle independently.



Cartesian Space Interpolation:
The controller calculates a smooth path for the end-effector (e.g., a straight line) in Cartesian space.

Crafting the Motion: Two Paths

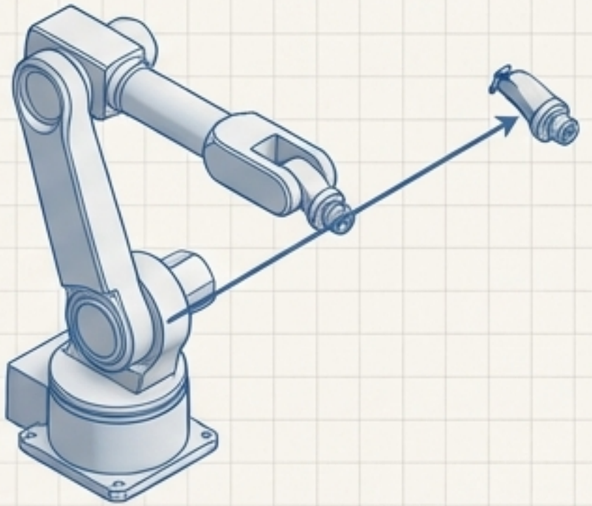
Joint Space Interpolation



Pros: Computationally simple, always achievable if start/end points are in workspace.

Cons: Tool path is not a straight line, which can be problematic for tasks like welding or sealing.

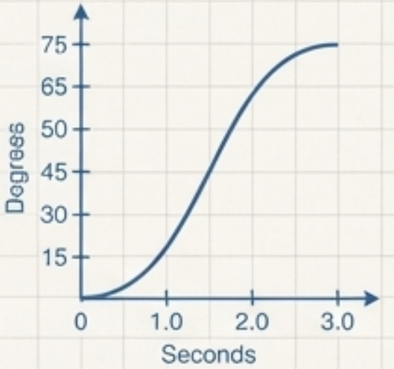
Cartesian Space Interpolation



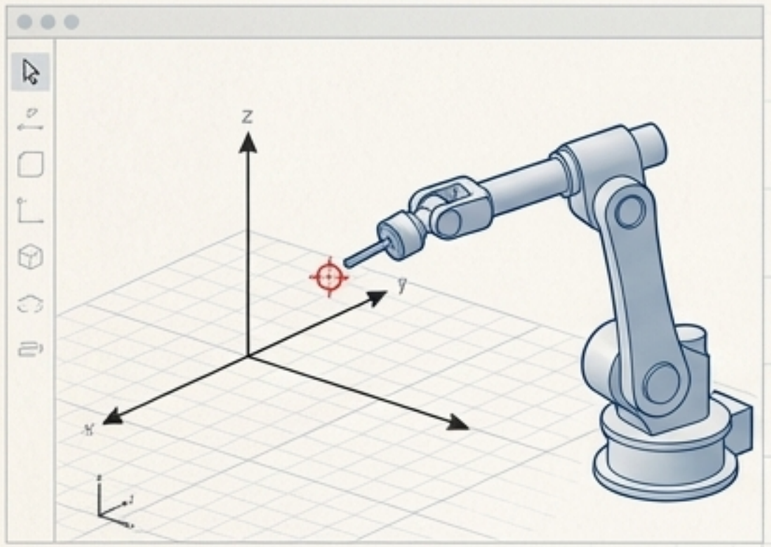
Pros: Provides a predictable, straight-line tool path.

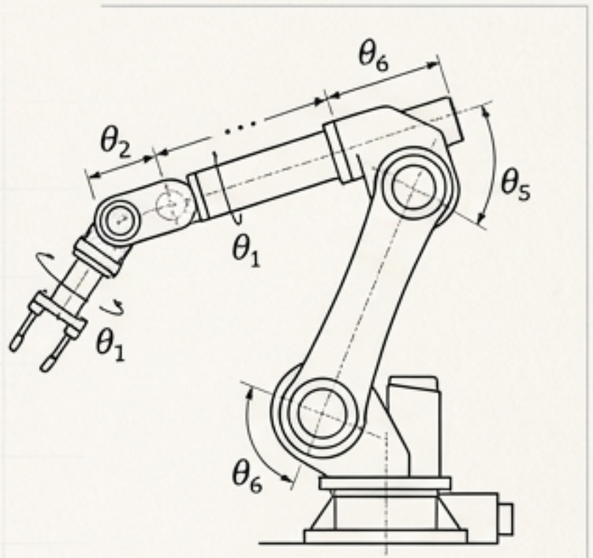
Cons: Computationally intensive (requires many IK calculations), may fail if the path passes through a singularity or out of the workspace.

Functions like **Cubic Polynomials** or **LSPB (Linear Segments with Parabolic Blend)** are used to generate the trajectory profiles, ensuring continuous velocity and acceleration to prevent jerky motion.



The Complete Translator





Desired Cartesian Pose (T_{goal})

[Inverse Kinematics]

Target Joint Angles (θ_{goal})

[Motion Planning]

Joint Trajectory ($\theta(t), \dot{\theta}(t), \ddot{\theta}(t)$)

[Robot Control]

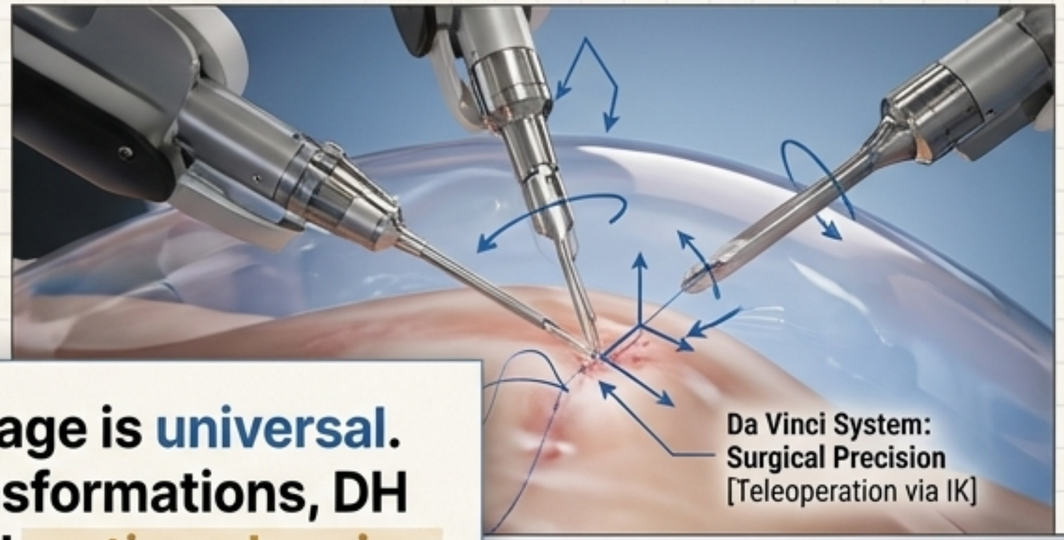
Robot Action

The Intent: Cartesian Space

The Reality: Joint Space

The Rosetta Stone is complete. We have successfully bridged the gap between the language of human intent and the language of the machine.

From Theory to Reality: A Universal Language



This kinematic language is universal. The principles of transformations, DH parameters, IK/FK, and motion planning are the foundation that powers nearly every robot in operation today.

Kinematics is more than mathematics. It is the fundamental grammar that allows machines to purposefully and precisely interact with our world.

For further resources, scan here.