Chapter 3
Forward Kinematics

Forward kinematics of a robot refers to the process of calculating the **position and orientation** of the robot's **end-effector** (e.g., a hand or tool) in space, **given the joint parameters** (such as angles for rotary joints or displacements for prismatic joints).

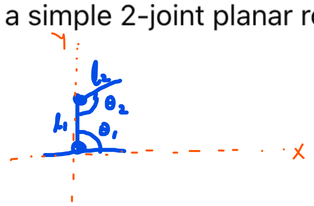
For a robotic arm with n joints, the overall transformation from the base to the end-effector is:

$$T = T_1 \cdot T_2 \cdot T_3 \cdot \cdots \cdot T_n$$

Where each Ti represents the transformation contributed by joint i.

Example:

For a simple 2-joint planar robotic arm (2 links of lengths l_1 and l_2, with joint angles θ_1 and θ_2):



$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

This gives the (x, y) position of the end-effector.

Difference between forward and inverse kinematics:

Aspect	Forward Kinematics (FK)	Inverse Kinematics (IK)
Definition	Calculates the **position and orientation** of the end-effector **from known joint parameters** (e.g., angles, displacements).	Calculates the **joint parameters** needed to achieve a **desired end-effector position and orientation**.
Input	Joint angles or joint displacements	Desired position and orientation of the end-effector
Output	Position and orientation of the end-effector (usually as a matrix or vector)	Joint angles or displacements that achieve the target position
Mathematical Complexity	**Simple and straightforward** (uses direct transformation matrices)	**Complex and non-linear** (requires solving systems of equations, often with multiple solutions)
Solvability	Always has a unique solution for given joint parameters	May have **no solution**, **one**, or **multiple solutions**, depending on reachability and constraints
Computation	Typically **analytical and fast**	Often requires **numerical methods** or **optimization algorithms**
Use Cases	Simulation, path visualization, animation	Robot control, pick-and-place tasks, trajectory planning
Tools/Techniques	Denavit-Hartenberg (DH) convention, matrix multiplication	Jacobian matrix, Newton-Raphson, Cyclic Coordinate Descent (CCD), optimization
Visualization	Predicts where the robot's arm will reach given the joint configuration	Determines how to move the joints to reach a particular target
Example	Given: $\theta_1 = 30°$, $\theta_2 = 45°$ → Find (x, y, z) of hand	Given: (x, y, z) = (5, 4, 2) → Find θ_1, θ_2, θ_3
Programming Use	Straightforward implementation (e.g., matrix multiplication)	May need iterative solving, libraries, or symbolic computation

Homogeneous Transformation Matrices:

A **homogeneous transformation matrix** is a mathematical tool used in robotics and 3D geometry to **represent both rotation and translation** of a coordinate frame **in a single matrix**.

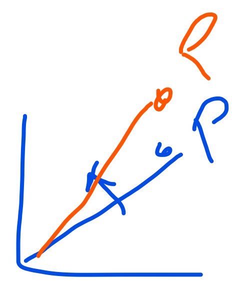
It allows you to **transform points** from one coordinate frame to another in a compact and consistent way using **4×4 matrices**.

In 3D space:

- A **rotation** is a 3×3 matrix.
- A **translation** is a 3×1 vector.

$$T = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- R: 3×3 rotation matrix
- d: 3×1 translation vector
- Bottom row: [0 0 0 1] makes it homogeneous

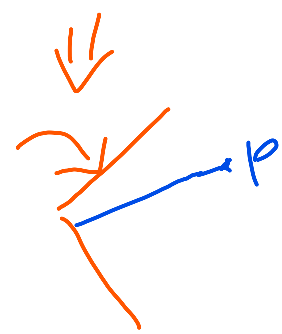


Example: 2D Transformation (for simplicity)

Let's rotate a point by **90° counterclockwise** and then **translate it by (2, 3)**.

Step 1: Rotation Matrix (90° CCW)

x' = xcosA - ysinA
y' = xsinA + ycosA



$$R = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ \\ \sin 90^\circ & \cos 90^\circ \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Step 2: Translation Vector

$$d = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

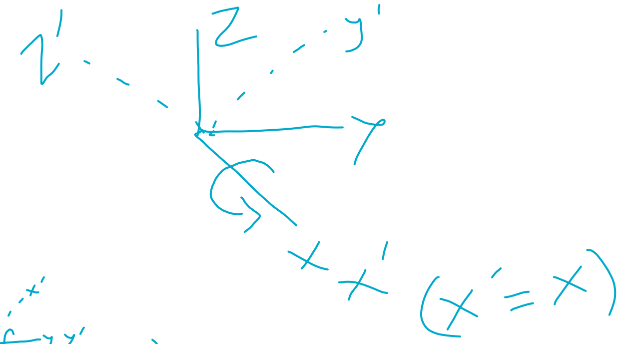
Step 3: Homogeneous Transformation Matrix (2D → 3×3)

$$T = \begin{bmatrix} 0 & -1 & 2 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

Examples of R for 3D Rotation:

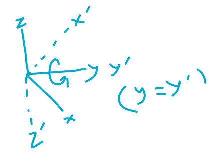
1. Rotation about X-axis by angle θ:

$$R_x(\theta) = \begin{array}{cc} & \begin{array}{ccc} x' & y' & z' \end{array} \\ \begin{array}{c} x \\ y \\ z \end{array} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \end{array}$$



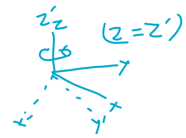
2. Rotation about Y-axis by angle θ:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$



3. Rotation about Z-axis by angle θ:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Let's compute the full **rotation matrix** R for 3D rotation using the angles:

- θ: rotation about X-axis
- λ: rotation about Y-axis
- γ: rotation about Z-axis

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\lambda) = \begin{bmatrix} \cos\lambda & 0 & \sin\lambda \\ 0 & 1 & 0 \\ -\sin\lambda & 0 & \cos\lambda \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_z \cdot R_y \cdot R_x$$

$$T = \begin{bmatrix} R_{3\times 3} & \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \\ 0\;0\;0 & 1 \end{bmatrix}$$

Mathematical Problem:

A point $P = (2, 1)$ is first **rotated 45° counterclockwise (CCW)** about the origin and then **translated by (3, 5)** units.

❓ **What are the final coordinates of the point after these transformations?**

Solution:

$$P_{\text{final}} = R(\theta) \cdot P + T$$

For this problem:

$$\theta = 45^\circ, \quad T = (3, 5), \quad P = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Rotation matrix:

$$R(45^\circ) = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

Step 1: Rotate the point

$$R \cdot P = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}(2) - \frac{\sqrt{2}}{2}(1) \\ \frac{\sqrt{2}}{2}(2) + \frac{\sqrt{2}}{2}(1) \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}(1) \\ \frac{\sqrt{2}}{2}(3) \end{bmatrix}$$

Using $\frac{\sqrt{2}}{2} \approx 0.7071$:

$$P_{\text{rotated}} = \begin{bmatrix} 0.7071 \\ 2.1213 \end{bmatrix}$$

Step 2: Translate the rotated point by $(3, 5)$

$$P_{\text{final}} = \begin{bmatrix} 0.7071 + 3 \\ 2.1213 + 5 \end{bmatrix} = \begin{bmatrix} 3.7071 \\ 7.1213 \end{bmatrix}$$

Denavit-Hartenberg (DH) Convention

The **Denavit-Hartenberg (DH) Convention** is a **systematic method** used in **robotics** to assign coordinate frames to the links and joints of a robot, and to describe their positions and orientations using a **standard 4-parameter representation**. It helps in calculating the **forward kinematics** of a robotic manipulator in a consistent and simplified way.

Why Use the DH Convention?

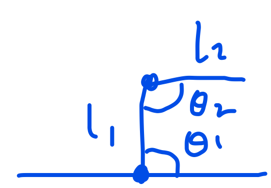
Robots often have multiple joints and links. Describing the position and orientation of each link in 3D space can get complex. The DH convention simplifies this by:

- Assigning frames systematically,
- Using a fixed order of transformations,
- Representing them with **a standard homogeneous transformation matrix**.

The 4 DH Parameters

Parameter	Symbol
Link Length	a_i
Link Twist	α_i
Link Offset	d_i
Joint Angle	θ_i

DH Transformation Matrix



Imagine a 2-joint arm (both revolute), with:

- $a_1 = L_1, a_2 = L_2$
- $\alpha_1 = \alpha_2 = 0$ No twist between the links — their joint axes are aligned in parallel (no angle between z-axes)
- $d_1 = d_2 = 0$ No offset along the z-axis — typical for a planar arm (2D).
- θ_1, θ_2 are the joint angles

Then the DH table:

Link i	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2

Each transformation matrix becomes simple due to zero twists and offsets.