Robotics: DOF and Joints

Minor in AI - IIT ROPAR

24th May, 2025

What is a Degree of Freedom (DOF)?

In robotics, a **Degree of Freedom (DOF)** refers to an independent type of movement a robot or a mechanical system can perform. Think of DOF as the number of ways a robot can move. Each DOF is either:

- A **linear (translational)** movement (moving along an axis), or
- A **rotational (angular)** movement (rotating about an axis).

Simple Analogy: Imagine your own arm. You can:

- Move your hand left or right (1 way of moving)
- Move it up or down (2nd way)
- Move it forward or backward (3rd way)
- Rotate it at the shoulder, bend at the elbow, twist the wrist, etc.

Each of these is a different degree of freedom!

Types of Movements in 3D Space

A robot or object in 3D space can move in the following six ways. These are the six basic degrees of freedom:

Translational (Linear) Movements

These are straight-line motions along the X, Y, and Z axes.

- **Translation along X-axis:** Forward and backward movement.
  Example: A car driving straight on a road.
- **Translation along Y-axis:** Left and right movement.
  Example: A chess piece sliding sideways on a board.
- **Translation along Z-axis:** Up and down movement.
  Example: An elevator going up or down.

Rotational (Angular) Movements

These are rotations about the X, Y, and Z axes.

- **Rotation around X-axis (Roll):** Tilting side to side.
  Example: When an airplane tilts its wings while turning.
- **Rotation around Y-axis (Pitch):** Nodding up and down.
  Example: When you nod your head to say "yes".
- **Rotation around Z-axis (Yaw):** Turning left or right.
  Example: When you shake your head to say "no".

Diagram Tip: You can draw 3 arrows from a cube's center to represent X, Y, and Z directions, and show the corresponding motions with curved arrows.

Total DOF in 3D Space

For a rigid body in three-dimensional space:

- It can move in 3 ways (X, Y, Z).
- It can rotate in 3 ways (Roll, Pitch, Yaw).

Total = 3 translations + 3 rotations = 6 DOF

Why are DOF Important in Robotics?

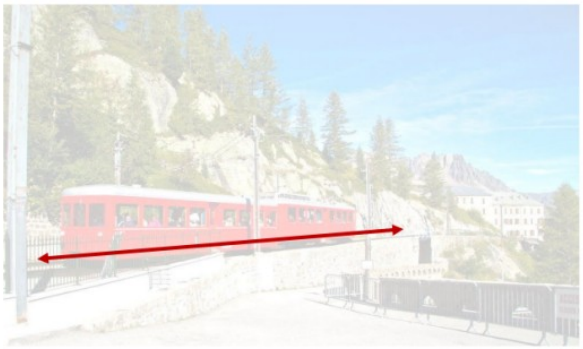
- A robot's DOF tells us how flexible it is.
- More DOF means the robot can perform more complex tasks.
- To reach any point and orientation in 3D space, a robot needs at least **6 DOF**.

Example: A robotic arm in a factory must:

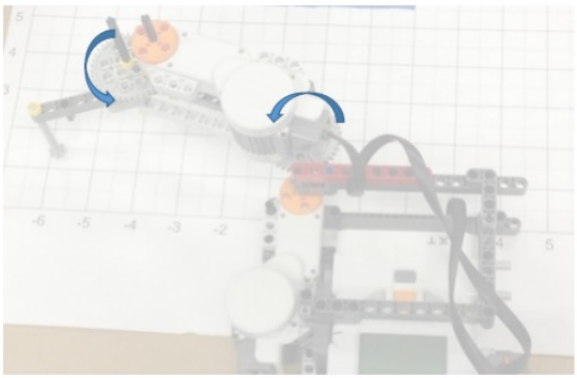
- Reach a point (X, Y, Z) — 3 DOF
- Align its gripper properly (Roll, Pitch, Yaw) — 3 DOF

Without 6 DOF, it cannot properly grab or place objects in space.

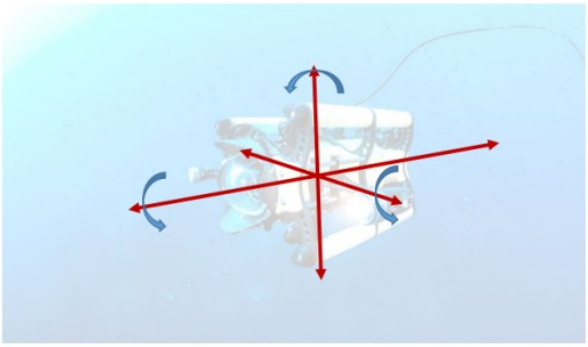
Real-Life Examples of DOF



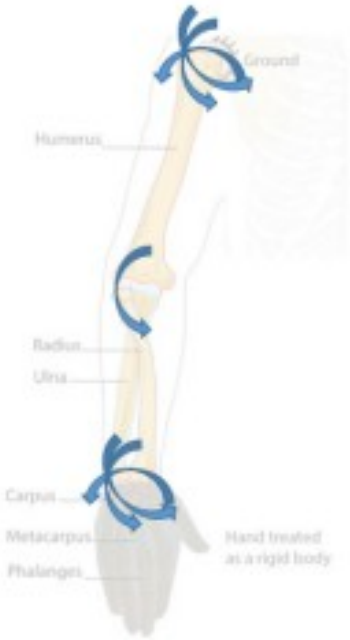
- **1 DOF:** A sliding drawer — it only moves in or out (along one axis).



- **2 DOF:** A desk lamp with a hinge and a swivel.
- **3 DOF:** A robotic leg that can lift, swing forward/backward, and rotate.



- **6 DOF:** A drone — it can move in all directions and rotate on all axes.



- **7 DOF:** A human arm:
  - Shoulder joint – 3 DOF (pitch, yaw, roll)
  - Elbow – 1 DOF (bending)
  - Wrist – 3 DOF (rotation and bending)

Introduction to Robotic Joints

In robotics, joints connect the rigid links of a robot and allow movement between them. These joints are essential for motion and control, as they define how a robot's parts can move. The most common joint types used in robotic manipulators are:

- **Revolute Joint (R)** – allows rotational motion
- **Prismatic Joint (P)** – allows linear or sliding motion

Each joint contributes to the robot's **Degrees of Freedom (DOF)**, which define how many independent movements a robot can perform.

Revolute Joint: Example – Human Elbow

One of the most intuitive examples of a revolute joint is the human elbow. The elbow allows your forearm to rotate relative to the upper arm. This motion is an angular rotation around a fixed axis.

Explanation: A revolute joint permits **rotational movement** between two connected links. The amount of rotation is measured by the angle θ, known as the joint variable.

- **Joint Type:** Revolute (symbol: `R`)
- **Type of Motion:** Pure rotation around a fixed axis
- **Joint Variable:** Angle θ
- **Degrees of Freedom:** 1 DOF (only rotation)
- **Real-world example:** Human elbow, door hinge, steering wheel shaft

Visual Analogy: Imagine a door. It swings open and closed around the hinge (the joint), but it doesn't slide up/down or forward/backward. This is the essence of a revolute joint — rotational movement without translation.

Prismatic Joint: Example – Hydraulic Piston

In contrast to a revolute joint, a prismatic joint allows one link to **slide linearly** along another. A good example of this is a hydraulic piston — commonly found in heavy machinery — where a rod slides in and out of a cylinder in a straight line.

Explanation: A prismatic joint enables **linear (translational) motion** along a single axis. It restricts all rotational movement and allows only straight-line displacement. The amount of movement is measured by a distance variable d.

- **Joint Type:** Prismatic (symbol: `P`)
- **Type of Motion:** Pure linear movement
- **Joint Variable:** Displacement d
- **Degrees of Freedom:** 1 DOF (only translation)
- **Real-world example:** Hydraulic piston, desk drawer, elevator actuator

Visual Analogy: Think of a desk drawer. It slides in and out along rails, maintaining the same orientation. It does not rotate at all. This is the fundamental behavior of a prismatic joint.

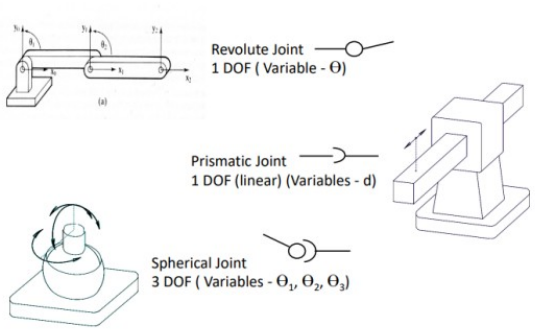


Figure: Types of Joints

Mathematical Representation of a Prismatic Joint

In robotics, movement is often described using transformation matrices. For a prismatic joint that moves along the X-axis by d units, the transformation matrix may look like this:

$$T = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, only the d value changes as the prismatic joint slides, which affects the position of the next link without altering its orientation.

Comparison Table: Revolute vs Prismatic Joint

Property	**Revolute Joint (R)**	**Prismatic Joint (P)**
Motion Type	Rotational (angular)	Translational (linear)
Joint Variable	θ (angle)	d (distance)
Degrees of Freedom	1 DOF (rotation)	1 DOF (translation)
Allowed Motion	Rotation around a fixed axis	Linear motion along a fixed axis
Restricts	Translation	Rotation
Symbol	`R`	`P`
Real-World Examples	Elbow, door hinge, robot arm joint	Hydraulic piston, drawer slide, 3D printer Z-axis

Gruebler-Kutzbach Criterion for DOF Calculation

To determine the number of independent movements (degrees of freedom) of a mechanism, the **Gruebler-Kutzbach criterion** is widely used. This formula is applicable to both **spatial mechanisms** (operating in 3D space) and **planar mechanisms** (constrained to 2D motion).

General Formula (Spatial Mechanisms)

For spatial mechanisms, which exist in 3D and can potentially have 6 DOF (3 translations + 3 rotations), the formula is:

$$\text{DOF} = 6(N - 1 - J) + \sum_{j=1}^{J} f_j$$

Where:

- N = Total number of links, including the base
- J = Number of joints in the mechanism
- f_j = Degrees of freedom provided by the j^{th} joint (usually 1 for revolute or prismatic joints)

Formula for Planar Mechanisms

For planar mechanisms, which operate only in a 2D plane (typically with 3 DOF: 2 translations + 1 rotation), the formula is modified by replacing 6 with 3:

$$\text{DOF} = 3(N - 1 - J) + \sum_{j=1}^{J} f_j$$

Example: 2-Link Planar Robot Arm

Let us consider a simple planar robot with 2 links connected by revolute joints.

- Total number of links $N = 3$ (2 links + 1 base)
- Total joints $J = 2$
- Each joint is revolute, so $f_j = 1$ for each

Using the planar DOF formula:

$$\text{DOF} = 3(3 - 1 - 2) + (1 + 1) = 3(0) + 2 = 2$$

So, the robot has 2 degrees of freedom, allowing it to position its end-effector anywhere in the plane (but not arbitrarily orient it).

Visualizing DOF with RoboAnalyzer

To help learners intuitively understand degrees of freedom and how they affect robot motion, the **RoboAnalyzer** software is an excellent educational tool.

Key Features:

- Create and simulate serial robot manipulators in 3D
- Add revolute or prismatic joints interactively
- Modify joint variables and observe the effect on the end-effector
- View transformation matrices and Denavit-Hartenberg parameters automatically

Website: `https://roboanalyzer.com`
Using RoboAnalyzer, students can visually confirm how changing joint configurations changes the robot's pose and how the number of DOF influences the robot's flexibility and reach.

Forward Kinematics and Inverse Kinematics

Kinematics in robotics refers to the study of motion without considering the forces that cause it. It deals with the geometry of motion — positions, velocities, and accelerations of robot parts. There are two primary types of kinematics:

- **Forward Kinematics (FK)** – computing the position of the end-effector from given joint parameters.
- **Inverse Kinematics (IK)** – computing the joint parameters required to achieve a desired end-effector position.

Forward Kinematics (FK)

Forward Kinematics is the process of calculating the position and orientation of a robot's end-effector when the joint variables (like joint angles or linear displacements) are known.

What is Given?

- The type and length of each link in the robot
- The current angle of each revolute joint or displacement of each prismatic joint

What is Computed?

- The precise position and orientation of the end-effector, typically in Cartesian coordinates (e.g., x, y, z)

Applications

- Robot simulation and animation
- Visualization of robot movement
- Predicting end-effector path given joint movements

Advantages

- Deterministic: there is only one output for a given input
- Easy to compute using simple matrix multiplication

Analogy

Think of a robotic arm like your own arm. If you bend your elbow and wrist at specific angles, your hand will end up in a specific position. Forward kinematics helps you compute exactly where that hand will be.

Inverse Kinematics (IK)

Inverse Kinematics is the reverse process — determining the joint angles or displacements required to place the end-effector at a specific position and orientation.

What is Given?

- The desired position (and possibly orientation) of the end-effector
- The type and length of each link

What is Computed?

- The joint parameters (angles or displacements) that will place the end-effector in the desired position

Applications

- Motion planning and trajectory design
- Grasping and manipulation in robotic arms
- Industrial tasks requiring precision movement

Challenges

- Non-linear equations must be solved
- Multiple (or no) solutions possible
- May require numerical methods or iterative solvers

Analogy

Imagine reaching out your hand to grab a cup on a table. You know the desired location of your hand (the cup's position), but your brain must calculate how much to rotate your shoulder, elbow, and wrist to reach that point — this is inverse kinematics.

Comparison Table

Aspect	Forward Kinematics (FK)	Inverse Kinematics (IK)
Input	Joint parameters (angles/displacements)	Desired end-effector position and orientation
Output	End-effector position and orientation	Required joint parameters
Nature	Deterministic, straightforward	Often non-linear, multiple solutions
Computation	Simple matrix operations (e.g., using DH parameters)	May require iterative numerical methods
Applications	Simulation, animation, visualization	Motion planning, real-time control, grasping
Difficulty Level	Easier	More complex

What is Forward Kinematics?

Forward Kinematics (commonly abbreviated as FK) is a fundamental concept in robotics. It refers to the mathematical process used to determine the position and orientation of a robot's end-effector (the tool or hand at the end of the robot arm) based on the known values of its joint parameters.

In simpler terms, if we already know how each joint in a robot is configured — that is, the angles of the rotating joints (called revolute joints) or the distances in sliding joints (called prismatic joints) — then forward kinematics tells us exactly where the end of the robot arm will be located in space, and what its orientation (direction it is pointing) will be.

Why Forward Kinematics Matters

Forward kinematics plays a vital role in many areas of robotics:

- **Simulation and Animation:** When simulating a robot in software, we often input joint angles to see how the robot moves. FK is what computes how the motion of each joint affects the position of the tool.
- **Control and Planning:** Before we can plan paths or avoid obstacles, we must understand where the robot is and how its joints influence that. FK provides that understanding.
- **Prerequisite for Inverse Kinematics:** Inverse kinematics (IK) involves figuring out which joint values will produce a desired position. But solving IK often requires solving FK many times as part of the process.
- **Kinematic Chains:** FK is the basis for modeling how multiple links and joints form a kinematic chain and how movement at one joint propagates to the entire arm.

Inputs to Forward Kinematics

To compute the position and orientation of the end-effector, forward kinematics requires the following inputs:

- **Joint Variables:** These define the current configuration of each joint. For revolute joints, the variable is the rotation angle (typically in radians or degrees). For prismatic joints, the variable is the linear displacement (usually in meters or millimeters).
- **Link Parameters:** These are geometric constants that describe the physical structure of the robot. Common parameters include:
  - Length of each link
  - Offset between joints
  - Twist angles between links

  These parameters are often defined using the Denavit–Hartenberg (DH) convention.

Outputs of Forward Kinematics

The output of FK is the pose of the end-effector relative to a fixed coordinate system, often called the **world frame** or **base frame**. This pose consists of:

- **Position:** A 3D point (x, y, z) describing the location of the end-effector.
- **Orientation:** The direction the end-effector is pointing, which can be represented using rotation matrices, Euler angles, or quaternions.

Together, position and orientation describe the full pose of the end-effector in space.

Deterministic Nature of FK

An important property of forward kinematics is that it is **deterministic**. That means, for a given set of joint values and fixed robot geometry, the resulting end-effector pose is unique. There is only one correct answer. This makes FK relatively straightforward to compute and extremely reliable for simulation and analysis.

Why Use Homogeneous Transformation Matrices?

In robotics, we frequently deal with both **rotations** and **translations**. Managing them separately can be cumbersome, especially when trying to combine multiple movements. Homogeneous transformation matrices solve this problem by allowing both rotation and translation to be represented within a single 4×4 matrix.

Purpose and Benefits

Homogeneous matrices are used to:

- **Represent combined transformations:** Each transformation matrix encodes both how a link is rotated and how it is translated (shifted) relative to the previous link.
- **Chain multiple movements:** In robots with multiple joints, each joint transforms the frame attached to it. Homogeneous matrices can be multiplied together to form a single matrix that represents the cumulative transformation from base to end-effector.
- **Work in 3D space:** These matrices simplify the math required to deal with 3D geometry, which includes both angular and linear motion.

Structure of a Homogeneous Transformation Matrix

A standard 4×4 homogeneous transformation matrix T is written as:

$$T = \begin{bmatrix} R_{3\times3} & d_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$$

Where:

- $R_{3\times3}$ is a rotation matrix that defines how the coordinate frame is rotated.
- $d_{3\times1}$ is a translation vector that defines how the coordinate frame is shifted.
- The bottom row [0 0 0 1] is used for matrix algebra and does not affect the geometry.

Why It Works

The beauty of this matrix is that when you multiply it with a point written in homogeneous coordinates (i.e., a 4×1 column vector like $[x\ y\ z\ 1]^T$), it applies both the rotation and the translation in a single operation. This makes transformation calculations efficient and easy to implement in code.

Usage in DH Convention

The Denavit–Hartenberg (DH) convention is a standardized method for modeling robotic arms. Each joint and link is described using four parameters, and these are converted into a homogeneous transformation matrix. These matrices are then multiplied sequentially from base to end-effector to compute the full FK.

Introduction to the Denavit–Hartenberg Method

In robotics, understanding how different parts of a robot arm are connected and move relative to each other is crucial. The Denavit–Hartenberg (DH) method is a standard technique that helps us describe the geometry of robot arms in a simple, systematic way. This method allows us to find the exact position and orientation of the robot's end-effector (such as a hand or tool) based on the angles and positions of its joints.

The main idea is to represent the robot's structure using coordinate frames attached to each link and then describe how to move from one frame to the next using a small set of parameters. This reduces the complexity of analyzing robot motion and makes the problem much easier to solve mathematically.

Why do we need the DH method?

- Robots can have many joints and links, making manual calculations complicated.
- We need a standardized, repeatable process for assigning coordinate systems to parts of the robot.
- The DH method simplifies calculations by using only four parameters per joint, regardless of the robot's complexity.
- It enables efficient computation of forward kinematics — predicting the position of the end-effector from given joint angles.

Step 1: Assigning Coordinate Frames

The first and perhaps most important step in the DH method is to attach a coordinate frame to each link of the robot. These frames are essential to describe how one link is positioned and oriented relative to the previous one.

The coordinate frames consist of three perpendicular axes — labeled x_i, y_i, z_i — that form a right-handed system for each link i.

Rules for Assigning Frames

Assigning these frames correctly is crucial. The DH method defines a clear set of rules:

1. **Assign the z_i axis:** For the i^{th} joint, the z_i axis is chosen to lie along the axis of motion of that joint.
   - For *revolute joints*, which rotate, z_i points along the axis about which the joint rotates.
   - For *prismatic joints*, which slide, z_i points along the direction in which the joint slides.

   This axis essentially describes the "direction" of the joint's movement.
2. **Assign the x_i axis:** The x_i axis is defined to be perpendicular to both the current joint axis z_i and the previous joint axis z_{i-1}.
   - More specifically, x_i points along the common normal between z_{i-1} and z_i. The common normal is the shortest line perpendicular to both axes.
   - If z_{i-1} and z_i intersect, x_i is chosen perpendicular to both, passing through their intersection.
3. **Origin of frame i:** The origin is placed at the intersection of the z_i axis and the x_i axis. This intersection is the point where the coordinate frame is attached to the link.

4. **Assign the y_i axis:** Finally, y_i is chosen to complete the right-handed coordinate system:

$$\vec{y_i} = \vec{z_i} \times \vec{x_i}$$

which means y_i is the vector product of z_i and x_i.

These frames provide a consistent way to describe the position and orientation of each link relative to its predecessor.

Step 2: Understanding DH Parameters

Once the coordinate frames are assigned, the next step is to describe how to get from one frame $i-1$ to the next frame i. The DH method does this using **four parameters**:

- θ_i — The angle of rotation about the z_{i-1} axis needed to align x_{i-1} with x_i.
- d_i — The offset distance along the z_{i-1} axis from the origin of frame $i-1$ to the point where the common normal intersects z_{i-1}.
- a_i — The length of the common normal, i.e., the distance along the x_i axis between the two joint axes z_{i-1} and z_i. It represents the length of the link.
- α_i — The angle between the axes z_{i-1} and z_i, measured about the x_i axis. It represents the twist of the link.

Intuition behind these parameters:

- θ_i tells us how much the link rotates about the joint axis.
- d_i tells us how far the link shifts along the joint axis.
- a_i is basically the physical length of the link.
- α_i captures how the link twists relative to the previous link.

Together, these parameters fully describe the relative position and orientation between two adjacent links.

Step 3: Constructing the DH Transformation Matrix

We want to represent the change in position and orientation from frame $i-1$ to frame i in a single 4×4 matrix called the **homogeneous transformation matrix** T_i^{i-1}.

This matrix combines rotations and translations and is crucial for computing forward kinematics.

The transformation from frame $i-1$ to i can be decomposed into four simpler steps:

1. **Rotate about z_{i-1} by θ_i:**

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

This rotation aligns the x_{i-1} axis with x_i.

2. **Translate along z_{i-1} by d_i:**

$$T_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This shifts the origin along the joint axis.

3. **Translate along x_i by a_i:**

$$T_x(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This moves along the link length.

4. **Rotate about x_i by α_i:**

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

This accounts for the twist between links.

The full transformation is the product of these four matrices in order:

$$T_i^{i-1} = R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i)$$

This matrix represents both rotation and translation and fully encodes the spatial relationship between two consecutive links.

Step 4: Final Combined Transformation Matrix

Multiplying out the above four matrices, the final Denavit–Hartenberg transformation matrix from frame $i-1$ to i is:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Explanation of the entries:

- The 3×3 upper-left submatrix represents the rotation from frame $i-1$ to frame i.
- The first three entries of the last column represent the translation vector (the position of frame i origin relative to frame $i-1$).
- The last row $[0 \quad 0 \quad 0 \quad 1]$ is necessary to use homogeneous coordinates for combined rotation and translation.

Step 5: Using DH Matrices for Forward Kinematics

To find the pose of the robot's end-effector relative to the base frame (frame 0), multiply the DH transformation matrices of all the links sequentially:

$$T_n^0 = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot \ldots \cdot T_n^{n-1}$$

Here:

- T_n^0 is the overall transformation matrix from the base frame to the end-effector frame.
- n is the total number of joints or links.

This matrix contains:

- The orientation of the end-effector relative to the base.
- The position of the end-effector relative to the base.

Summary and Intuition Recap

- The DH method standardizes the description of robot arms using coordinate frames attached to each joint.
- Only four parameters per joint $(\theta_i, d_i, a_i, \alpha_i)$ are needed to describe all spatial relationships.
- The transformation matrix T_i^{i-1} encodes how to move from one link's coordinate frame to the next.
- Multiplying all these matrices gives the position and orientation of the robot's tool or hand in space.

This systematic approach turns a complex mechanical problem into a series of simple matrix multiplications, allowing robots to be controlled precisely in three-dimensional space.

Key Takeaways

- Forward Kinematics calculates the end-effector position and orientation from known joint angles or displacements and is straightforward to compute.
- Inverse Kinematics finds the joint parameters needed to place the end-effector at a desired position, and is generally more complex with possible multiple solutions.
- Homogeneous transformation matrices combine rotation and translation into a single structure, allowing easy chaining of movements between robot links.
- The Denavit–Hartenberg method provides a standard way to assign coordinate frames to each robot link and describes spatial relationships using only four parameters per joint.
- DH frames are assigned based on joint axes, common normals, and right-handed coordinate systems to ensure consistent and systematic modeling.
- Each joint is described by four DH parameters that capture the relative rotation, translation, link length, and twist between links.
- Transformation matrices derived from DH parameters represent how to move from one link's frame to the next, combining rotations and translations.
- Multiplying these matrices from the base to the end-effector gives the complete pose of the robot's tool in space.
- Degrees of Freedom (DOF) represent the number of independent motions a robot has, including translations and rotations; a minimum of six DOF is required for full 3D control.
- Understanding and calculating DOF, along with using kinematics and DH method, are essential for designing, simulating, and controlling robot arms effectively.