Robotics: Kinematics

Minor in AI - IIT ROPAR

29th May, 2025

1 Introduction to Kinematics in Robotics

Kinematics is a branch of classical mechanics that deals with the **geometry of motion** of objects without considering the forces that cause this motion. In the context of robotics, **robot kinematics** is the study of motion of robotic manipulators with respect to time, focusing solely on the positional and rotational aspects of its links and joints.

In robotics, kinematics helps us answer the fundamental question:

"Where is the robot's end-effector located in space, given a set of joint parameters?"

This question leads to the core kinematic problems: **Forward Kinematics (FK)** and **Inverse Kinematics (IK)**.

2 Key Elements of Robot Kinematics

Robot manipulators are typically made up of a combination of the following elements:

- **Links:** These are rigid bodies that connect two joints. They define the spatial structure of the manipulator.
- **Joints:** These are the points or axes at which relative motion between links is allowed. Common joint types include:
  - **Revolute Joints:** Allow rotational motion (like hinges).
  - **Prismatic Joints:** Allow translational motion (like sliders).
- **Degrees of Freedom (DoF):** The number of independent joint parameters that can be varied to configure the robot. For instance:
  - A revolute joint contributes one rotational DoF.
  - A prismatic joint contributes one translational DoF.

3 Types of Kinematics in Robotics

3.1 Forward Kinematics (FK)

- **Definition:** Forward Kinematics determines the position and orientation (pose) of the end-effector given the joint parameters (angles or displacements).
- **Input:**
  - Length and geometry of each link
  - Angle of each revolute joint or displacement of each prismatic joint
- **Output:**
  - The spatial pose of the end-effector, typically in Cartesian coordinates (e.g., x, y, z) and orientation (e.g., Euler angles or a rotation matrix)

- **Applications:**
  - Animation and simulation
  - Visualization and monitoring of motion
- **Characteristics:**
  - Deterministic: One unique solution exists for a given set of joint variables.
  - Computationally straightforward: Uses transformation matrices and trigonometry.

3.2 Inverse Kinematics (IK)

- **Definition:** Inverse Kinematics determines the joint parameters required to achieve a desired pose of the end-effector.
- **Input:**
  - Length and geometry of each link
  - Desired pose of the end-effector (position and orientation)
- **Output:**
  - Joint angles (for revolute joints) or joint displacements (for prismatic joints)
- **Applications:**
  - Motion planning
  - Robot control and automation
  - Grasping and manipulation tasks
- **Characteristics:**
  - Often complex: Involves solving non-linear equations.
  - May have:
    - No solution (when the target is unreachable)
    - A unique solution
    - Multiple solutions (due to redundancy or symmetry)
  - Requires numerical or analytical techniques.

4 Comparison: Forward vs Inverse Kinematics

Aspect	**Forward Kinematics (FK)**	**Inverse Kinematics (IK)**
Description	Joint parameters → End-effector pose	Desired pose → Joint parameters
Input	Joint angles or displacements	Desired position and orientation
Output	Position and orientation of end-effector	Joint values to achieve the pose
Complexity	Straightforward (uses geometry)	Often complex (non-linear equations)
Solvability	Always has a unique solution	May have multiple, unique, or no solutions
Application	Visualization, animation	Motion planning, control, grasping

5 Coordinate Frames and Spatial Representation

Why Coordinate Frames?

Robots function in a three-dimensional (3D) environment. In order to accurately describe and compute their movements, every component—whether a link, joint, or end-effector—must have its position and orientation described with respect to a defined reference. This is achieved using **coordinate frames**.

Coordinate Frame Definition

A **coordinate frame** is a reference system used to represent spatial positions and orientations. It consists of:

- An **origin** point in space.
- Three **mutually perpendicular axes**:
  - X-axis
  - Y-axis
  - Z-axis

Any position or orientation in 3D space can be expressed with respect to such a frame.

Common Coordinate Frames in Robotics

Robotic systems make use of several important coordinate frames to define positions and transformations:

- **Base Frame:**
  - Fixed to the robot's base or ground.
  - Serves as the primary reference for all other transformations.
- **Tool Frame:**
  - Attached to the robot's end-effector (e.g., a gripper or welding tool).
  - Moves with the end-effector.
- **Intermediate Frames:**
  - Defined at various joints or along links.
  - Useful for step-by-step transformations between the base and tool frame.

World Frame vs Local Frame

Understanding spatial representation also requires distinguishing between world and local frames:

- **World Frame (Global Frame):**
  - A fixed frame, often defined by the environment (e.g., lab floor or workspace table).
  - Used as a universal reference for comparing different robot positions and movements.
- **Local Frame (Body Frame):**
  - Moves along with the robot or its individual parts.
  - Describes motion or configuration from the perspective of a specific robot component.

Importance in Robotics

Coordinate frames allow precise mathematical modeling of:

- **Kinematic chains:** by chaining transformations from one frame to another.
- **Trajectory planning:** by expressing paths in a consistent coordinate system.
- **Sensor integration:** by converting sensor data into the appropriate reference frame.

Without a consistent frame representation, it would be impossible to define motion, apply control, or interpret measurements accurately.

6 End-Effector Orientation Using Rotation Matrices

Objective: Determining End-Effector Orientation Relative to an Object

In robotic manipulation and perception tasks, it is often necessary to determine how the robot should orient its end-effector with respect to an object in the environment. This requires calculating a rotation matrix that aligns the robot's tool frame (end-effector) with the object's frame.

Given: The following rotation matrices are provided:

- $\mathbf{R}_C^I$: Rotation from inertial (world) frame to camera frame.
- $\mathbf{R}_O^C$: Rotation from camera frame to object frame.
- $\mathbf{R}_B^I$: Rotation from inertial (world) frame to robot base frame.
- $\mathbf{R}_E^B$: Rotation from base frame to end-effector frame.

Objective: Compute the orientation of the end-effector relative to the object frame, i.e., find $\mathbf{R}_O^E$.

Transformation Chaining

To find $\mathbf{R}_O^E$, we utilize the chaining of transformations based on the composition of rotation matrices. The required transformation is obtained by:

$$\mathbf{R}_O^E = \mathbf{R}_O^C \cdot \mathbf{R}_C^I \cdot (\mathbf{R}_B^I)^\top \cdot (\mathbf{R}_E^B)^\top$$

Explanation of the Chain:

- $\mathbf{R}_O^C$: Rotates from camera frame to object frame.
- $\mathbf{R}_C^I$: Rotates from inertial frame to camera frame.
- $(\mathbf{R}_B^I)^\top$: Transposes the inertial-to-base rotation, giving base-to-inertial.
- $(\mathbf{R}_E^B)^\top$: Transposes the base-to-end-effector rotation, giving end-effector-to-base.

Each transpose represents an inverse rotation since rotation matrices are orthogonal: $\mathbf{R}^{-1} = \mathbf{R}^\top$. This formula effectively transforms the object's orientation into the end-effector's frame.

Rotation Matrix Convention

What does a rotation matrix do?

A rotation matrix $\mathbf{R}_A^B$ is used to express a vector originally defined in frame B in terms of frame A.

$$\mathbf{v}_A = \mathbf{R}_A^B \cdot \mathbf{v}_B$$

Interpretation:

- $\mathbf{v}_B$: Vector expressed in frame B.
- $\mathbf{v}_A$: Same vector expressed in frame A.
- $\mathbf{R}_A^B$: Rotation matrix that transforms from frame B to frame A.

Indexing Convention:

- The **subscript** indicates the frame in which the vector is currently expressed.
- The **superscript** indicates the frame into which the vector should be re-expressed.
- The rotation matrix $\mathbf{R}_A^B$ therefore transforms or reorients a vector from frame B into frame A.

Why This Matters

In robotic tasks involving visual servoing, grasping, or aligning tools with targets, precise control of orientation is critical. By chaining transformations through known frames (camera, inertial, base, etc.), robots can compute how to move their end-effector to match the orientation of an object detected in sensor data.

This mathematical approach ensures:

- Accurate alignment and grasping
- Coordination between sensors and actuators
- Proper execution of complex motion planning tasks

7 Moving Between Frames Using Rotation Matrices

Problem Statement

In robotics, it is often necessary to express a vector or an orientation that is originally defined in one frame (e.g., the end-effector frame E) in terms of another frame (e.g., the object frame O). However, a direct transformation $\mathbf{R}_O^E$ from the end-effector frame to the object frame may not be available. Instead, we may have access to a sequence of intermediate frames and their corresponding rotation matrices.

Goal: Given a chain of known rotations, compute the composite rotation matrix that transforms a vector from frame E (end-effector) into frame O (object).

Step-by-Step Transformation Path

To transform a vector from frame E to frame O, we follow a multi-step process through intermediate frames. The transformation chain is:

$$\mathbf{R}_O^E = \mathbf{R}_O^C \cdot \mathbf{R}_C^I \cdot (\mathbf{R}_B^I)^\top \cdot (\mathbf{R}_E^B)^\top$$

Let us now understand each step in detail.

Step 1: Convert from End-Effector Frame E to Base Frame B

$$\text{Apply } (\mathbf{R}_E^B)^\top$$

- The matrix $\mathbf{R}_E^B$ represents the rotation from base frame B to end-effector frame E.
- Since we are going in the reverse direction (from E to B), we use the transpose (or inverse) of this matrix.
- This step re-expresses a vector from end-effector coordinates into base coordinates.

Step 2: Convert from Base Frame B to Inertial Frame I

$$\text{Apply } (\mathbf{R}_B^I)^\top$$

- The matrix $\mathbf{R}_B^I$ gives the orientation of the base frame with respect to the inertial (world) frame.
- To move from base to inertial (i.e., reverse direction), we again apply the transpose.
- This step now expresses the vector in terms of the world coordinate system.

Step 3: Convert from Inertial Frame I to Camera Frame C

$$\text{Apply } \mathbf{R}_C^I$$

- This matrix rotates vectors from the inertial frame to the camera frame.
- Since the direction aligns with the matrix definition, no transpose is needed.
- The vector is now expressed in the camera frame.

Step 4: Convert from Camera Frame C to Object Frame O

$$\text{Apply } \mathbf{R}_O^C$$

- This matrix rotates from the camera frame to the object frame.
- Again, since the direction matches, we use the matrix directly.
- The vector is finally re-expressed in the object frame.

Final Orientation Transformation

After applying all the steps in sequence, we obtain:

$$\mathbf{R}_O^E = \mathbf{R}_O^C \cdot \mathbf{R}_C^I \cdot (\mathbf{R}_B^I)^\top \cdot (\mathbf{R}_E^B)^\top$$

This resulting matrix allows us to:

- Transform any orientation or direction vector from the end-effector's coordinate system into the object's coordinate system.
- Understand how the end-effector must be oriented to align with the object.

Visual Interpretation and Best Practices

- Always draw a diagram showing all relevant frames and the directions of known transformations.
- Confirm the direction of each transformation—whether you need a matrix or its transpose.
- Carefully follow the order of matrix multiplication, from the innermost transformation to the outermost.
- Remember that matrix multiplication is not commutative: $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$

Why This is Important

- Understanding how to re-express vectors across frames is foundational for robot perception, control, and manipulation.
- Tasks such as visual servoing, motion planning, and sensor fusion all depend on correct and consistent frame transformations.
- This procedure ensures the robot's motions are planned and executed with respect to the correct environmental context.

8 What is Pose in Robotics?

In robotics, the term **pose** refers to the complete description of the spatial configuration of a robot or its end-effector. Specifically, a pose combines two essential components:

- **Position**: Specifies the location of the point in space.
- **Orientation**: Specifies how the object is rotated in space relative to a reference frame.

Mathematically, we write:

$$\text{Pose} = \text{Position} + \text{Orientation}$$

Position

The position component defines the coordinates of a point in three-dimensional space (usually in meters). It is typically represented by a column vector:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- x, y, z specify the point's location along the three Cartesian axes.
- This vector describes "where" the robot or its end-effector is located in space.
- Position alone is not enough; it does not describe how the object is facing or oriented.

Orientation

The orientation defines how the object is rotated with respect to some reference frame. There are multiple mathematical representations for orientation:

- **Rotation Matrix (3×3):**
  - Orthogonal matrix with determinant +1.
  - Transforms vectors from one frame to another.
  - Provides a complete, non-redundant representation.
- **Euler Angles (Yaw, Pitch, Roll):**
  - Describes rotation as three sequential angles about coordinate axes.
  - Intuitive but susceptible to gimbal lock.
- **Quaternion:**
  - A four-dimensional representation (one scalar and three-vector components).
  - Compact, avoids singularities, and is numerically stable.
- **Axis-Angle:**
  - Represents orientation as a rotation about a fixed axis.
  - Useful for interpolation and concise expressions.

Pose as a Transformation Matrix

Ultimately, pose is expressed as a single mathematical object known as a **homogeneous transformation matrix**, which combines both position and orientation:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

Where:

- $\mathbf{R}$ is the 3×3 rotation matrix describing orientation.
- $\mathbf{p}$ is the 3×1 position vector.
- The last row [0 0 0 1] ensures proper matrix multiplication in homogeneous coordinates.

This transformation matrix allows for efficient computation of coordinate transformations and is widely used in forward/inverse kinematics, trajectory planning, and computer vision.

9 What is Translation?

Translation in robotics refers to the movement of a point or object from one location to another in space *without any rotation*. It purely involves shifting the position of the object while maintaining its original orientation.

Mathematical Representation

Translation is represented by a 3-dimensional vector that describes a point's location in space:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3$$

- The components x, y, z represent the coordinates of the point with respect to a given reference frame (such as the base of a robot).
- The translation vector $\mathbf{p}$ answers the question: *"Where is the robot or its end-effector located?"*
- Units are usually in meters (m), which is standard in robotic applications.

Geometric Interpretation

- Translation can be visualized as sliding an object along a straight path in 3D space.
- The object does not rotate or change orientation—it simply changes position.
- Translation vectors are fundamental in defining relative positions between parts of a robot or between the robot and objects in the environment.

Example

Suppose the robot's end-effector is located 0.3 meters along the X-axis, 0.2 meters along the Y-axis, and 0.5 meters along the Z-axis from the robot's base frame. This position is represented by:

$$\mathbf{p} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.5 \end{bmatrix} \text{ m}$$

This indicates:

- $x = 0.3\,\text{m}$: Rightward from the base.
- $y = 0.2\,\text{m}$: Forward from the base.
- $z = 0.5\,\text{m}$: Upward from the base.

Role in Robotics

- Translation is a key part of a robot's pose, which includes both **position (translation)** and **orientation (rotation)**.
- It is used in path planning, kinematics, and control to ensure the robot can reach or avoid specific locations.
- Translation vectors are also embedded into homogeneous transformation matrices to perform frame transformations.

10 What is Rotation?

In robotics, **rotation** refers to the orientation or angular displacement of an object in 3D space. While translation describes *where* an object is, rotation tells us *how* the object is aligned relative to a reference frame.

Rotation Matrix Representation

The most common way to represent rotation in 3D is using a **rotation matrix**:

$$\mathbf{R} \in \mathbb{R}^{3\times 3}$$

A valid rotation matrix satisfies two important properties:

- **Orthogonality:** $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.
- **Unit Determinant:** $\det(\mathbf{R}) = 1$

These properties ensure that the rotation matrix preserves distances and angles, and that it represents a proper rotation (not a reflection).

Elementary Rotations

In 3D space, rotation can be defined around the three principal Cartesian axes. These are known as **elementary rotations**. Each corresponds to rotation about one axis, keeping the other two fixed.

Rotation about the X-axis (Roll)

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

- This matrix rotates a vector by an angle θ about the X-axis.
- Commonly called *Roll*.
- It leaves the X-component unchanged and rotates the YZ-plane.

Rotation about the Y-axis (Pitch)

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

- This matrix rotates a vector by an angle θ about the Y-axis.
- Known as *Pitch*.
- It leaves the Y-component unchanged and rotates the XZ-plane.

Rotation about the Z-axis (Yaw)

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

- This matrix rotates a vector by an angle θ about the Z-axis.
- Often called *Yaw*.
- It leaves the Z-component unchanged and rotates the XY-plane.

Combined Rotations

Rotations can be composed by multiplying their corresponding matrices in sequence. For example, if a robot performs a roll, then pitch, then yaw, the combined rotation matrix is:

$$\mathbf{R} = \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi)$$

Where:

- ϕ is the roll angle (about X-axis)
- θ is the pitch angle (about Y-axis)
- ψ is the yaw angle (about Z-axis)

Applications in Robotics

- Describing the orientation of robotic arms and end-effectors.
- Changing coordinate representations between frames.
- Calculating relative angles between different parts of a robot or between robot and world.
- Essential in forward and inverse kinematics, trajectory generation, and control.

Rotation, along with translation, forms the full 6D **pose** of a robot or object in 3D space.

11 Homogeneous Transformation Matrix

Why Use a Homogeneous Transformation Matrix?

In robotics, a robot's position and orientation must often be transformed from one coordinate frame to another. While translations and rotations can be represented separately, combining them into a single operation greatly simplifies calculations and improves efficiency.

This is accomplished using a **homogeneous transformation matrix (HTM)**, which unifies rotation and translation into one 4×4 matrix:

- It allows both position and orientation to be transformed simultaneously using a single matrix multiplication.
- It enables consistent transformation of points, vectors, and coordinate frames between different spaces (e.g., world to robot, base to end-effector).

Matrix Structure

The homogeneous transformation matrix $\mathbf{T}$ is structured as follows:

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

- $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix.
- $\mathbf{d} = \begin{bmatrix} d_x & d_y & d_z \end{bmatrix}^\top \in \mathbb{R}^{3\times1}$ is the translation vector.
- The bottom row $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ is a structural element that enables affine transformations using matrix multiplication.

Transforming a Point Using HTM

To transform a 3D point using a homogeneous transformation matrix, we first convert the point into homogeneous coordinates.

Let a point $\mathbf{p}$ be expressed in homogeneous coordinates as:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Then, the transformed point $\mathbf{p}'$ in the new frame is computed by:

$$\mathbf{p}' = \mathbf{T} \cdot \mathbf{p}$$

Substituting the matrix structure:

$$\mathbf{p}' = \begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0}^\top & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{d} \\ 1 \end{bmatrix}$$

- The result is a new point in the transformed coordinate frame.
- The rotation matrix $\mathbf{R}$ rotates the point.
- The translation vector $\mathbf{d}$ shifts the rotated point to a new position.

Interpretation and Use

- Homogeneous transformation matrices are used to represent poses (position + orientation) of links, joints, and end-effectors in robotic systems.
- They are also used to chain multiple transformations together. For instance, to go from frame A to frame C through B:
$$\mathbf{T}_{AC} = \mathbf{T}_{AB} \cdot \mathbf{T}_{BC}$$
- HTMs are foundational to both **forward** and **inverse kinematics**, as well as motion planning and control.