Introduction to Robotics

A Brief Overview

Shameel Abdulla

TAMUQ

March 5, 2024

Outline

Control: Stepper motors

Stepper motors are essential in robotics for precise control of angular or linear position, velocity, and acceleration. They operate on the principle of electromagnetism to achieve fine motion control.

Communication: Serial

Serial communication, often used in robotics, is a process of sending data one bit at a time, sequentially, over a communication channel. It's fundamental for microcontroller to peripheral communication, like sensors and actuators.

Controller: Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's widely adopted in robotics for prototyping due to its simplicity and extensive community support.

Serial Robots

Serial robots, or serial manipulators, feature an open kinematic chain where each joint connects only two links. This configuration is common in industrial and research applications due to its flexibility and range of motion.

Parallel Robots

Parallel robots, or parallel manipulators, consist of closed-loop kinematic chains. Known for their rigidity and high-speed operation, they are commonly used in applications requiring high precision.

Forward Kinematics

Forward kinematics in robotics is the process of calculating the position and orientation of the robot's end effector based on the given joint parameters without considering forces or torques.

Forward Kinematics

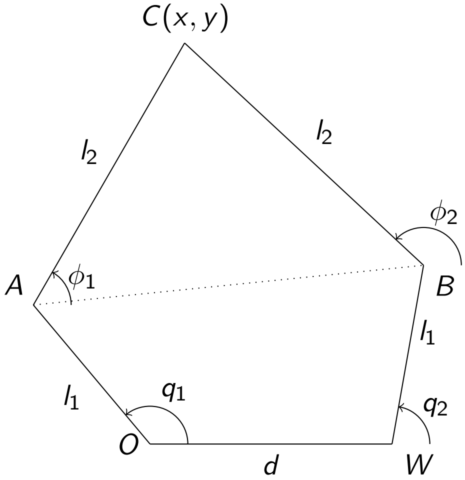

Figure: Forward Kinematics

Forward Kinematics

1. **Vector A Definition:**

$$\mathbf{A} = l_1 \begin{bmatrix} \cos(q_1) \\ \sin(q_1) \end{bmatrix}$$

2. **Vector B Definition:**

$$\mathbf{B} = \begin{bmatrix} d + l_1 \cos(q_2) \\ l_1 \sin(q_2) \end{bmatrix}$$

3. **Vector D (Difference between B and A)**:

$$\mathbf{D} = \mathbf{B} - \mathbf{A}$$

Since **D** is derived from **B** and **A**, substituting the values gives:

$$\mathbf{D} = \begin{bmatrix} d + l_1 \cos(q_2) - l_1 \cos(q_1) \\ l_1 \sin(q_2) - l_1 \sin(q_1) \end{bmatrix}$$

(Cano-Ferrer)

Forward Kinematics

4. **Angle ψ Calculation** (using the 2nd element over the 1st element of vector D for the arctan2 function):

$$\psi = \arctan 2(D_y, D_x)$$

With D_x and D_y being the first and second elements of **D**, respectively.

5. **Distance h (Euclidean norm of D):**

$$h = \|\mathbf{D}\| = \sqrt{D_x^2 + D_y^2}$$

6. **Angle δ_1 Calculation:**

$$\delta_1 = \arccos\left(\frac{h}{2l_2}\right)$$

Forward Kinematics

7. **Angle ϕ_1 Calculation:**

$$\phi_1 = \delta_1 + \psi$$

8. **Vector C Definition:**

$$\mathbf{C} = \mathbf{A} + l_2 \begin{bmatrix} \cos(\phi_1) \\ \sin(\phi_1) \end{bmatrix}$$

Expanding **A** from its definition, we get:

$$\mathbf{C} = l_1 \begin{bmatrix} \cos(q_1) \\ \sin(q_1) \end{bmatrix} + l_2 \begin{bmatrix} \cos(\phi_1) \\ \sin(\phi_1) \end{bmatrix}$$

Inverse Kinematics

Inverse kinematics is the process of determining the joint parameters needed to place the end effector at a desired position and orientation. This is more complex than forward kinematics and crucial for robotic manipulation.

Inverse Kinematics

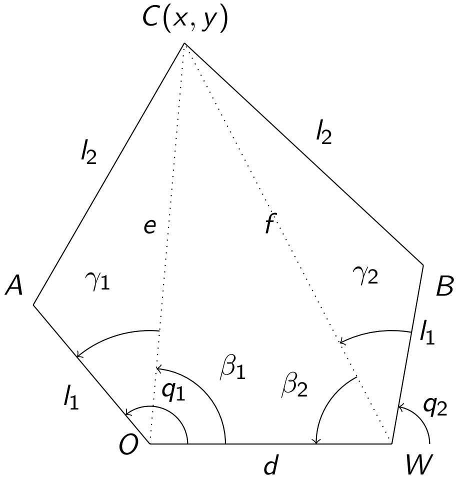

Figure: Inverse Kinematics

Inverse Kinematics

1. **Distance C Calculation:**

$$e = \sqrt{x^2 + y^2}$$

2. **Distance e Calculation:**

$$f = \sqrt{(d - x)^2 + y^2}$$

3. Angles

$$q_1 = \gamma_1 + \beta_1$$

$$q_2 = \pi - (\gamma_2 + \beta_2)$$

Inverse Kinematics

4. **Angle q_1 Calculation:**

$$q_1 = \arctan 2(y, x) + \arccos\left(\frac{l_1^2 + e^2 - l_2^2}{2 \cdot l_1 \cdot e}\right)$$

Here, the $\arctan 2(y, x)$ term calculates the angle to the point (x, y) from the positive x-axis, and the arccos term adjusts this angle based on the lengths l_1, l_2, and the calculated distance e.

5. **Angle q_2 Calculation:**

$$q_2 = \pi - \arctan 2(y, d - x) - \arccos\left(\frac{l_1^2 + f^2 - l_2^2}{2 \cdot l_1 \cdot f}\right)$$

In this equation, $\pi - \arctan 2(y, d - x)$ computes the angle considering the horizontal displacement from d to x, and the arccos term is used to find the required adjustment based on the lengths l_1, l_2, and the calculated distance f.

Adjusting coordinates

To describe the operations being performed on the ‘x‘ and ‘y‘ coordinates mathematically, let’s break down the process step by step:

1. **Creation of DataFrame:** This step creates a DataFrame named ‘result‘ with columns ‘x‘ and ‘y‘ containing the coordinates.
2. **Calculation of Middle Point in X:** This calculates the average of the minimum and maximum values of the ‘x‘ coordinates, effectively finding the middle point along the x-axis.

$$\text{middle} = \frac{\min(x) + \max(x)}{2}$$

3. **Shifting X Coordinates:** This shifts the ‘x‘ coordinates so that the middle point calculated above becomes the new origin (0,0).

$$x_{\text{new}} = x - \text{middle}$$

Adjusting coordinates

4. **Flipping the X Coordinates:** This step inverts the ‘x‘ coordinates to mirror the drawing across the y-axis.

$$x_{\text{flipped}} = -x_{\text{new}}$$

5. **Shifting the Flipped X Coordinates to the Right:** This shifts the flipped ‘x‘ coordinates further to the right by adding 40 units (d/2).

$$x_{\text{final}} = x_{\text{flipped}} + 40$$

6. **Shifting Y Coordinates Upwards:** This shifts the ‘y‘ coordinates upwards by adding 60 units (Safe distance away from the center of the motors).

$$y_{\text{final}} = y + 60$$

These equations describe the transformation applied to the ‘x‘ and ‘y‘ coordinates.

References

[Cano-Ferrer] Cano-Ferrer, X. Educational five-bar parallel robot. https://cdn.hackaday.io/files/1733257415536800/Educational%20Five-bar%20parallel%20robot_.pdf. Accessed: 2 February 2024.