Fundamentals of Robotics

A review of fundamental concepts in robotics, from coordinate frames to manipulator dynamics and control.

Joe(y) Carpinelli

2023-08-13

Table of contents

Preface

This is still being written! The following topics will eventually be explored in this note-set.

Review of Coordinate Transformations

- Rotations Matrices
- Passive, active rotations
- 4×4 coordinate transformation matrix

Kinematic Descriptions

- Going from robot diagram to modified DH parameters
- Assigning coordinate frames

Forward Kinematics

- Use your kinematic description to map joint angles to Cartesian positions & orientations

Translational & Rotational Jacobians

- Differentiate your forward kinematic solutions with respect to joint angles to find how **joint velocities** map to **Cartesian velocities** and **anglular velocities**

Inverse Kinematics

- Find joint angles to match some Cartesian *pose*
- Generally relies on the Jacobian having full rank

Singularities

- Find when the Jacobian does not have full rank

Dynamics

- Apply forces through joint angles
- Incorporate mass properties, and how they interact with Cartesian velocities & angular rates
- Incorporating moments of inertia, other dynamical elements

Control

- PID Control

Applications

- ROS: *Robot Operating System*
- MoveIt: motion planning framework within ROS
- Orocos KDL: Open Source Kinematics & Dynamics Solvers in C++

1 Reference Frames

Reference frames exist at some point in space, at some orientation in space, relative to another reference frame. There are two ways to describe multi-axis rotations: fixed axes, and euler (rotating) axes.

1.1 Why Rotations

We use rotations to answer two different questions...

1. Given P, what are the coordinates of **that same point** relative to F_j?
   - Answer looks like: ${}^{j}(P_0) = R_j P$
2. Given P, what are the coordinates of a point **in** F_2 that are rotated about some axis?
   - Answer looks like: ${}^{2}(P) = R_i P$

How can we reconcile this?

- $R_j \equiv R_i^T$

1.2 Rotation Matrices

Relating reference frames requires many (somewhat arbitrary) definitions. We may choose to relate frames by rotating axes, or fixed axes. We may choose to change the basis of a three-valued vector through rotation, or we may choose to rotate a three-valued vector in its original reference frame. For these reasons, you may see two different definitions of rotation matrices about principle axes. Both definitions are the transpose — and due to orthonormality, the inverse — of one another.

The definitions below are the principle rotation matrices we will use moving forward. The following matrices rotate a three-valued vector by an angle of θ about each of the three basis vectors; see Equations 2.77, 2.78, and 2.79 in Craig's Introduction to Robotics [1].

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{1.1}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{1.2}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

1.3 Euler Axis Rotations

There are many ways to describe rotations about multiple axes. One way is to apply a sequence of rotations about principle axes, applying each rotation to the preceding *intermediate* frame. This technique is known as *euler* sequence rotations.

You might say: to get frame B, rotate frame A by θ_1 degrees about frame A's X axis, then rotate **that intermediate frame** θ_2 degrees about the intermediate frame's Y axis, etc.

Euler ZYX: $R_z(\theta_3)R_y(\theta_2)R_x(\theta_1)$

1.4 Fixed Axis Rotations

Fixed XYZ: $R_x(\theta_1)R_y(\theta_2)R_z(\theta_3)$

1.5 Translations & Rotations (Transformations)

You can combine translations and rotations to create *transformations*.

2 Kinematic Descriptions

Kinematics is the study of an object's motion independent of the forces that cause the object's motion. For portability, we want the kinematic expressions for robotic manipulators to be valid for all manipulators. A kinematic description, in this context, contains all of the geometric information about a manipulator necessary to form the standard kinematic equations of motion. The robotics industry uses several standard kinematic descriptions, including URDF/XACRO files, and (modified) DH parameters.

2.1 Modified DH Parameters

Given a specific serial manipulator, you may apply the following steps to construct the Modified DH parameters. Note that Modified DH Parameters are **not** unique. You may select multiple sets of parameters which yield the same kinematics.

1. Copy the contents of Table 2.1, where N is the manipulator's number of joints, and i is the index of the *current* joint you're inspecting. The frame with index 0 should be the *base frame* or *world frame*: some frame from which all joint positions can be referenced.

Table 2.1: Template for Modified DH Parameters Table

i	a_{i-1}	α_{i-1}	d_i	θ_i
1				
2				
...				
N				

2. For each index (row), assign the corresponding link frame axes and fill in the values for the parameters a, α, d, and θ using the rules presented in Figure 2.1. The index offset $i-1$ in the column headers can be confusing. To be clear, the first values you will fill in are a_0, α_0, d_1, and θ_1. The next values are a_1, α_1, d_2, and θ_2.

1. Assign axis $\hat{z}_i$ to the direction of joint rotation if revolute, or joint translation if prismatic. Assign axis $\hat{x}_i$ in any direction that is **not** the direction of $\hat{z}_{i+1}$; if you can while satisfying this condition, assigning $\hat{x}_i$ in a direction pointing towards the next joint is helpful. Place the origin of frame i where $\hat{z}_i$ and $\hat{x}_i$ intersect.
2. The link length a_i is the distance from $\hat{z}_i$ to $\hat{z}_{i+1}$ along $\hat{x}_i$. Usually this value is positive or zero; negative a_i values are discouraged. If your selected a_i is negative, consider changing your definition of $\hat{x}_i$ in Rule 1.
3. The link twist α_i is the angle from $\hat{z}_i$ to $\hat{z}_{i+1}$ about $\hat{x}_i$.
4. The joint offset d_i is the distance from $\hat{x}_{i-1}$ to $\hat{x}_i$ along $\hat{z}_i$.
5. The joint angle θ_i is the angle from $\hat{x}_{i-1}$ to $\hat{x}_i$ about $\hat{z}_i$.

Figure 2.1: Rules for Selecting Modified DH Parameter Values

2.2 Forward Kinematics

$$P_{i-1}^{i} = \begin{bmatrix} a_{i-1} \\ -d_i \sin(\alpha_{i-1}) \\ d_i \cos(\alpha_{i-1}) \end{bmatrix} \tag{2.1}$$

$$R_{i-1}^{i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \cos(\alpha_{i-1})\sin(\theta_i) & \cos(\alpha_{i-1})\cos(\theta_i) & -\sin(\alpha_{i-1}) \\ \sin(\alpha_{i-1})\sin(\theta_i) & \sin(\alpha_{i-1})\cos(\theta_i) & \cos(\alpha_{i-1}) \end{bmatrix} \tag{2.2}$$

$$T_{i-1}^{i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \cos(\alpha_{i-1})\sin(\theta_i) & \cos(\alpha_{i-1})\cos(\theta_i) & -\sin(\alpha_{i-1}) & -d_i\sin(\alpha_{i-1}) \\ \sin(\alpha_{i-1})\sin(\theta_i) & \sin(\alpha_{i-1})\cos(\theta_i) & \cos(\alpha_{i-1}) & d_i\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$${}^{A}P = T_B^A P_B \tag{2.4}$$

2.3 Examples

Transformations matrices are **associative**, but are **not** commutative.

$$T_0^4 = T_0^1 T_1^2 T_2^3 T_3^4 \tag{2.5}$$

$$\text{inv}(T_i^j) \triangleq T_j^i \tag{2.6}$$

We might want to know the end-effector position ${}^{B}P_{e}$ in the base frame B.

$$ {}^{B}P_{e} = \text{position}(T_{B}^{0} T_{0}^{1} ... T_{N}^{e}) $$

3 Jacobians

Jacobians map joint velocities to translational and angular (Cartesian) velocities. The *angular Cartesian velocities* are commonly referred to as *body rates*: p is the angular rate about the X axis (roll), q is the angular rate about the Y axis (pitch), and r is the angular rate about the Z axis (yaw).

3.1 Derivations

The translation and rotation Jacobians can be derived using two methods: direct differentiation, and the cross-product method.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = J_T \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ ... \\ \dot{q_n} \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = J_\Omega \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ ... \\ \dot{q_n} \end{bmatrix} \tag{3.2}$$

3.2 Direct Differentiation

3.2.1 Translation Jacobian

$$^0J_T = \frac{\partial}{\partial \vec{q}} {}^0P_T$$

3.3 Cross Product Method

4 Inverse Kinematics

4.1 Steps for Command a Robot Arm

1. We are given the joint positions, $\vec{q}$. We want the arm to go to Cartesian state $\vec{p}$.
2. Using forward kinematics, we **know** the current Cartesian state.

References

[1] Craig, J. J. *Introduction to Robotics*. Pearson Educacion, 2006.