Robot Kinematics

Haijun Su
The Ohio State University

22nd September 2025

Contents

Chapter 1: Preface

This is an unofficial summary of the robot kinematics lectures on *Robot Kinematics* at The Ohio State University.

We aim to cover all the content from the exercises and supplement it with explanations from the book *Introduction to Robotics - Mechanics and Control (3rd ed.)* by John J. Craig. Much of the lecture notes are by the professors Philipp Wulff and Jan Hansen-Palmus from the Technical University of Munich GitHub repository . And the original content of their lecture notes is based on the Stanford lecture *Introduction to Robotics* by professor Oussama Khatib, which you can watch on YouTube. Many of the figures in this summary are direct screenshots of his slides. Note, that the notations are not consistent throughout this summary.

Chapter 2: Kinematics Fundamentals

Joints usually have motion/position sensors, allowing to measure their relative position to neighboring links. The figure below shows some commonly used kinematic joints [1].

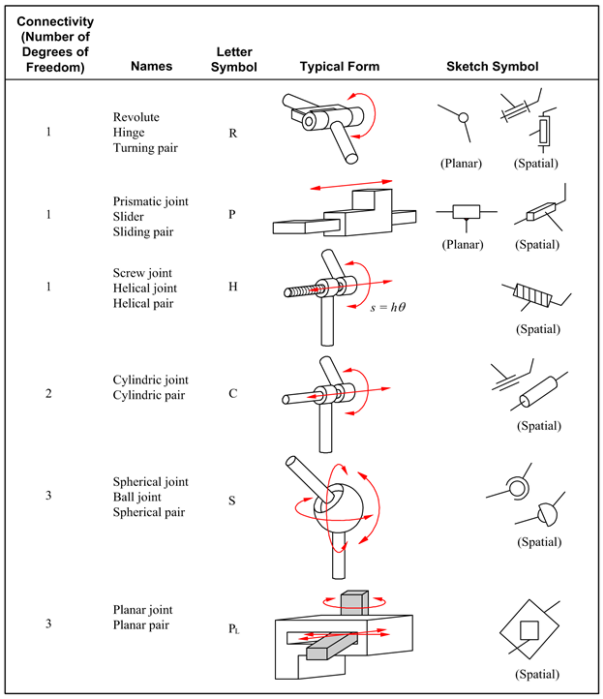
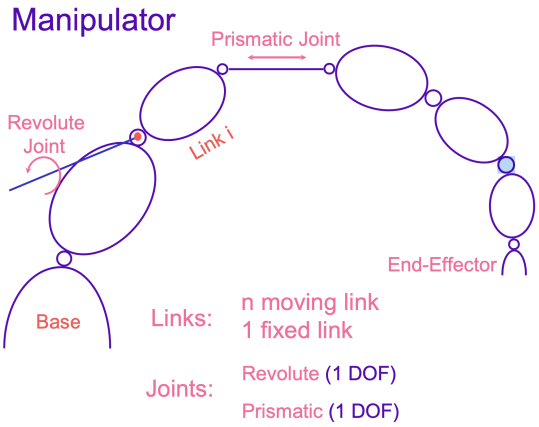


Figure 1: Commonly used kinematic joints [1].

The **end-effector** is located at the end of the chain of links that make up the manipulator. Different types of joints connect the links of a manipulator [1].



Chapter 3: Forward kinematics

3.1 Workspace

Kinematics is the science of motion that treats motion without regard to the forces which cause it. One studies position, velocity, acceleration, and all higher order derivatives of position variables.

The existence or nonexistence of a kinematic solution defines the **workspace** of a given manipulator [2]. If a solution doesn't exist, this means that the desired position/orientation lies outside of the manipulator's workspace. In other words, the workspace consists of all points that are reachable by the end-effector.

3.1.1 Configuration

The **configuration** of a moving object is a specification of the position of **every** point on the object [3].

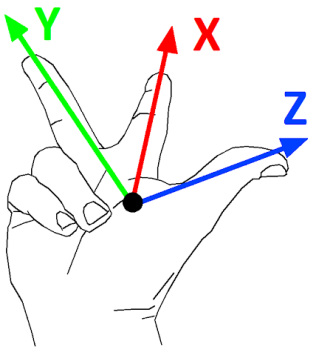
The **dimension of a config space** is the minimum number of parameters needed to specify the configuration of the object completely (also called the number of degrees of freedom of a moving object).

3.1.2 Degrees of freedom

The number of **degrees of freedom** that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. E.g., industrial robotic manipulators often have as many d.o.f. as their number of joints, since each joint has one d.o.f. (and has 5 constraints).

3.1.3 Right-hand-rule

The signs of angles are determined by the direction of the fingers, when the thumb is pointing in the axis direction. Fingers also show the order of axis.



3.2 Spatial descriptions

We attach a coordinate system to a body and give a description of this coordinate system relative to the reference system. In Figure 2.2, system B has been attached to the body, and its description relative to A is given through its positions and orientation relative to A.

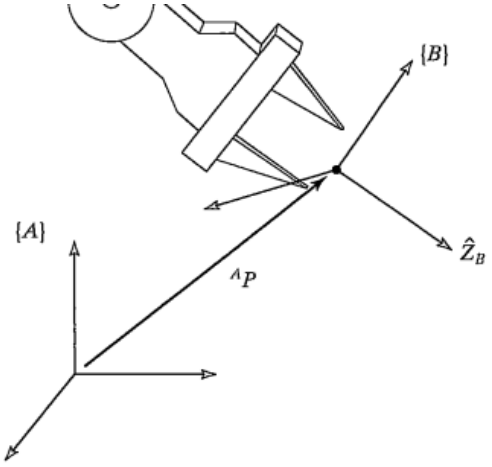


FIGURE 2.2: Locating an object in position and orientation.

3.2.1 Position

Description of a position: Since many coordinate systems will be used, one has to define to which system a vector refers, e.g.,

$$^A\vec{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix},$$

where $^A\vec{p}$ is a vector referring to coordinate system A.

3.2.2 Orientation

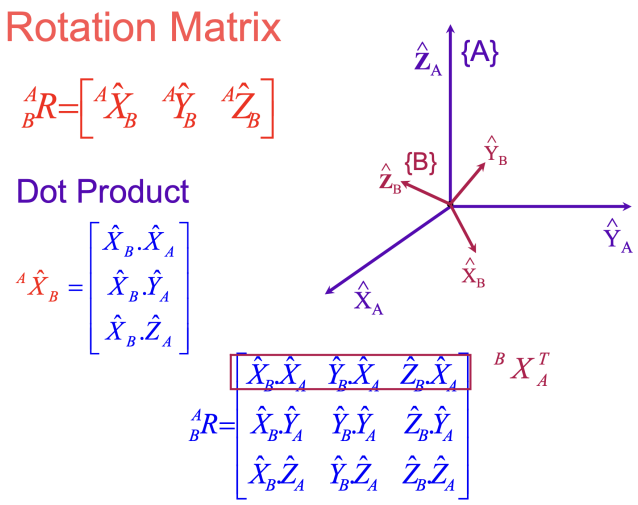
Description of an orientation: We stack three unit vectors (they specify the principal directions of the coord system) as columns, yielding the **rotation matrix:**

$$^A_B R = [^A\hat{X}_B \; ^A\hat{Y}_B \; ^A\hat{Z}_B] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = {}^B_A R^{-1} = {}^B_A R^T$$

The rotation matrix has three constraints:

- $|^A\hat{X}_B| = |^A\hat{Y}_B| = |^A\hat{Z}_B| = 1$
- $^A\hat{X}_B \cdot {}^A\hat{Y}_B = {}^A\hat{X}_B \cdot {}^A\hat{Z}_B = {}^A\hat{X}_B \cdot {}^A\hat{Z}_B = 0$
- $\det R = 1$

It describes the orientation of frame B relative to frame A, i.e. it is used as a *mapping* to **change the description of a vector from frame to frame**. Since we have unit magnitude and the vectors are orthogonal, the transposed matrix describes the orientation of system A written in B. These are orthonormal and length-preserving linear transformations. Also, rotation matrices preserve angles between vectors, i.e. $cos(\angle(\vec{p},\vec{q})) = cos(\angle(R\vec{p}, R\vec{q}))$. The projection of a vector $\hat{X}_B$ in coord system B into coord system A is derived from the dot product with the principal directions of the coord frame of A, hence the graphic.



Besides mappings, another use case of rotation matrices is in the form of (rotational) *operators* to move points within the same frame.

Python script example based on SciPy and NumPy:

```
import numpy as np
from scipy.spatial.transform import Rotation as R

# Create a random rotation
r = R.random()
R_matrix = r.as_matrix()
print("Rotation matrix:")
print(R_matrix)
print("Determinant:", np.linalg.det(R_matrix))
print("R^T R:", np.dot(R_matrix.T, R_matrix))
```

3.2.3 Special rotation matrices

The special rotation matrices for rotation about the axes are:

$$R_X(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}, R_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}, R_Z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
# Rotation about x by 45 degrees
r_X = R.from_rotvec([np.pi/4, 0, 0])
print("R_X(45 degrees):")
print(r_X.as_matrix())

# Similarly for y and z
r_Y = R.from_rotvec([0, np.pi/4, 0])
r_Z = R.from_rotvec([0, 0, np.pi/4])
```

3.2.4 Composition of rotations

A rigid body is subject to 2 or more consecutive rotations, defined by $[R_1], [R_2], \ldots$. The final rotation $[R]$ is defined by the product of these matrices. Depending on the reference coordinate system, there are two cases:

- Intrinsic: the 2nd rotation is about the axes of the body-fixed coordinate system, $[R] = [R_1][R_2]$
- Extrinsic: the 2nd rotation is about the global coordinate system, $[R] = [R_2][R_1]$

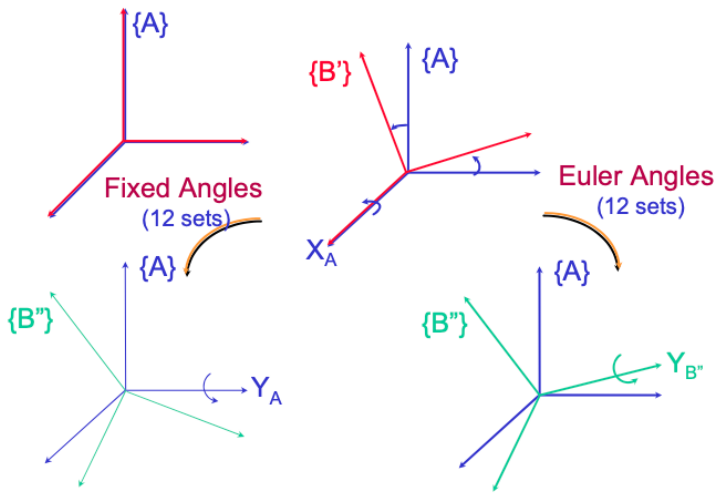
```
# Define two rotations
r1 = R.from_rotvec([np.pi/6, 0, 0]) # 30 degrees about x
r2 = R.from_rotvec([0, np.pi/4, 0]) # 45 degrees about y

# Intrinsic composition: [R] = [R1][R2]
r_intrinsic = r1 * r2
print("Intrinsic rotation matrix:")
print(r_intrinsic.as_matrix())

# Extrinsic composition: [R] = [R2][R1]
r_extrinsic = r2 * r1
print("Extrinsic rotation matrix:")
print(r_extrinsic.as_matrix())
```

3.2.5 Euler angles

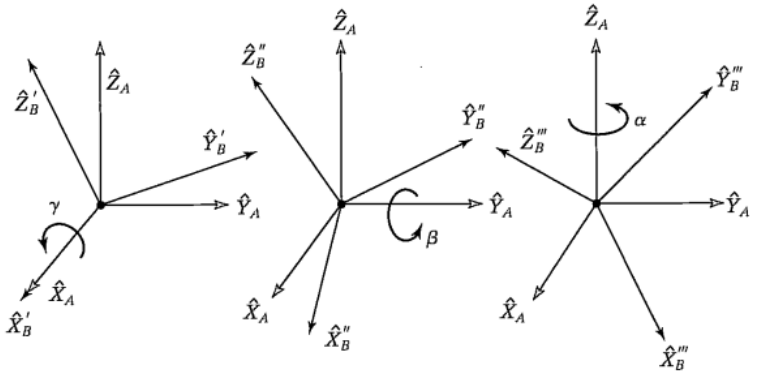
There is an issue when expressing orientations with a rotation matrix: When attempting to follow a trajectory in space by interpolating a current orientation from a start until a final orientation, the intermediary orientations are going to violate the constraints of the rotation matrix. Another possible description of a frame B uses a *three-angle-representation*. It works as follows: Frame B coincides with a known frame A. Rotate B first

- about $\hat{X}_A$ by an angle γ, then about
- $\hat{Y}_A$ by an angle β, and finally about
- $\hat{Z}_A$ by α.

If, each of the three rotations takes place about an axis in the fixed reference frame A, we call these angles *X-Y-Z fixed angles*. An alternative representation uses angles that are relative to the previously changed coordinate frame, so-called **Euler angles**.



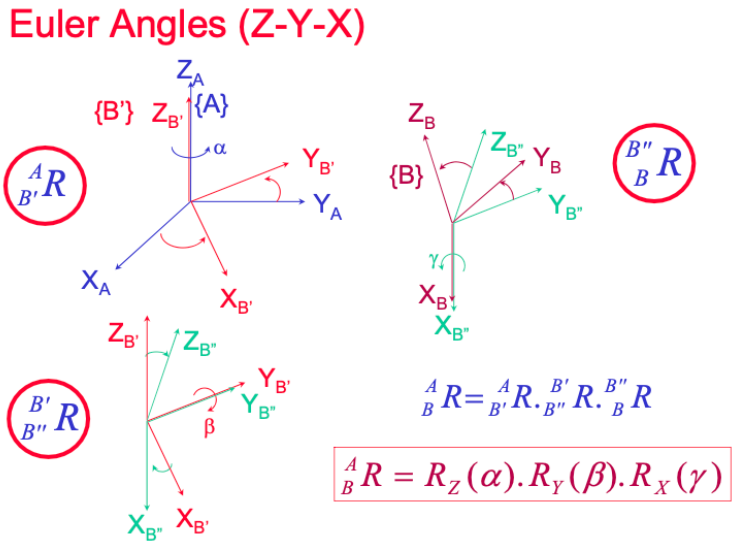
The next figure shows three subsequent rotations of a frame around the fixed axes $\hat{X}_A$, $\hat{Y}_A$ and $\hat{Z}_A$.



The combined rotation matrix is ${}^A_B R = {}^A_{B'} R \cdot {}^{B'}_{B''} R \cdot {}^{B''}_{B} R$, which is cumbersome to compute. Instead, we define:

$$
\begin{aligned}
{}^A_B R_{XYZ}(\gamma, \beta, \alpha) &= R_Z(\alpha) R_Y(\beta) R_X(\gamma) \\
&= \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \\
&= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}
\end{aligned}
$$

When rotating a frame with Euler angles in the order of Z-Y-X, they produce the same combined rotation as X-Y-Z fixed angles, i.e. $R_{Z'Y'X'}(\alpha, \beta, \gamma) = R_{XYZ}(\gamma, \beta, \alpha)$ for the same values of (α, β, γ). This next graphic shows the rotation using Euler angles:



Inverse problem: Given ${}^A_B R$ find the X-Y-Z fixed angles (α, β, γ). Let

$$ {}^A_B R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

If $\cos\beta \neq 0$,

$$ \begin{aligned} \beta_1 &= \arcsin(-r_{31}), \beta_2 = \pi - \arcsin(-r_{31}) \\ \alpha_i &= \operatorname{atan2}\left(r_{21}/\cos\beta_i, r_{11}/\cos\beta_i\right) \\ \gamma_i &= \operatorname{atan2}\left(r_{32}/\cos\beta_i, r_{33}/\cos\beta_i\right) \end{aligned} $$

See the definition of $\operatorname{atan2}(a, b)$ in the Appendix.

If $\cos\beta = 0$, this is a singularity (gimbal lock), we have infinite solutions of α and γ. There are two possible solution sets:

$$ \begin{aligned} \beta_1 &= \pi/2, \quad \gamma_1 - \alpha_1 = \operatorname{atan2}(r_{12}, r_{13}) \\ \beta_2 &= -\pi/2, \quad \gamma_2 + \alpha_2 = \operatorname{atan2}(-r_{12}, -r_{13}) \end{aligned} $$

There are 12 different Euler angle conventions, which arise from the different possible sequences of axis rotations (choosing which axes to rotate about and in what order). The examples given below are for intrinsic Euler angles, where each subsequent rotation is performed relative to the body's current orientation. There are 12 sets of intrinsic Euler angle conventions:

- **Proper Euler angles:** These are Euler angle sequences where the first and third rotations are performed about the same axis. The middle rotation is about a different axis.
  - z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y
- **Tait–Bryan angles:** These are Euler angle sequences where all three rotations are performed about different axes, providing a complete set of three distinct axes.
  - x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z

The corresponding extrinsic Euler angle sets are the reverse sequences. For instance, the extrinsic version of the intrinsic X-Y-Z is Z-Y-X. See 6.1 in the Appendices for all 12 sets of intrinsic Euler angles and their corresponding rotation matrices [4].

Example: x-y-z (intrinsic): Let α, β, γ be the Euler angles of three consecutive rotations in the sequence of x_0, y_1 and z_2, where the subscripts indicate the coorindate system the rotation is about.

$$ \begin{aligned} [R(\alpha, \beta, \gamma)]_{xyz} &= R_X(\alpha) R_Y(\beta) R_Z(\gamma) \\ &= \begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma + \sin\gamma\cos\alpha & -\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & -\sin\alpha\cos\beta \\ \sin\alpha\sin\gamma - \sin\beta\cos\alpha\cos\gamma & \sin\alpha\cos\gamma + \sin\beta\sin\gamma\cos\alpha & \cos\alpha\cos\beta \end{pmatrix} \end{aligned} $$

Inverse Kinematics: For the intrinsic x-y-z sequence, the angles can be recovered from $[R]$ as follows:

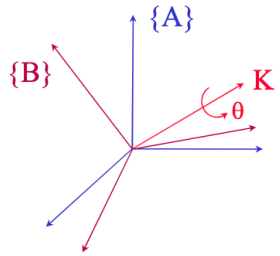
$$\beta_1 = \arcsin(r_{13}), \beta_2 = \pi - \arcsin(r_{13})$$
$$\alpha_i = \text{atan2}(\sin\alpha, \cos\alpha) = \text{atan2}\left(-\frac{r_{23}}{\cos\beta_i}, \frac{r_{33}}{\cos\beta_i}\right)$$
$$\gamma_i = \text{atan2}(\sin\gamma, \cos\gamma) = \text{atan2}\left(-\frac{r_{12}}{\cos\beta_i}, \frac{r_{11}}{\cos\beta_i}\right)$$

This gives two sets of solutions: $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$. When $\cos\beta = 0$ (gimbal lock), γ and α are not independent. Use the similar formulas as above to compute β and $\gamma - \alpha$ or $\gamma + \alpha$ as shown in the previous example.

```python
import numpy as np
from scipy.spatial.transform import Rotation as R

# Given Euler angles (alpha, beta, gamma) in radians
original_angles = np.array([0.1, 0.2, 0.3]) # example values

# Generate rotation matrix from Euler angles
r_original = R.from_euler('xyz', original_angles)
R_matrix = r_original.as_matrix()
print("Rotation matrix from original angles:")
print(R_matrix)

# Now, use inverse kinematics to extract Euler angles
extracted_angles = r_original.as_euler('xyz', degrees=False)
print("Extracted Euler angles (alpha, beta, gamma):")
print(extracted_angles)

# Check if they match (within numerical precision)
difference = np.abs(original_angles - extracted_angles)
print("Difference between original and extracted:")
print(difference)
print("Are they close?", np.allclose(original_angles, extracted_angles))
```

3.2.6 Unit quaternions (Euler parameters)

Another representation of orientation is by means of four values called the **Euler parameters**. It can be shown that every rotation can be expressed as one rotation of θ around a single axis $K = [k_x\ k_y\ k_z]^T$. This is also known as the *equivalent angle-axis representation*.



The Euler parameters are given by

$$(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4) = (k_x \sin\frac{\theta}{2}, k_y \sin\frac{\theta}{2}, k_z \sin\frac{\theta}{2}, \cos\frac{\theta}{2})$$

and

$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \epsilon_4^2 = 1$$

This 4×1 vector is known as a unit quaternion, the orientation it describes could be visualized as a point on a unit hypersphere in four-dimensional space. It turns out that the orientation representation through Euler parameters does not have a singularity.

3.3 Spatial transformations

Problem: We know the definition of a vector with respect to some frame B and we would like to express it with respect to another frame A, where the origins of the two frames are coincident. We can compute this, if we know a description of the orientation of B relative to A. Then,

$$^{A}P = {}^{A}_{B}R \; {}^{B}P$$

computes what a **vector that is expressed in coordinate system B "looks like" if observed from system A**.

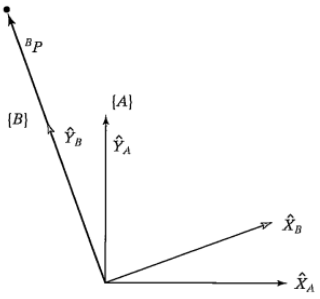


FIGURE 2.6: $\{B\}$ rotated 30 degrees about $\hat{Z}$.

Given

$$^{A}_{B}R = \begin{bmatrix} 0.866 & -0.500 & 0.000 \\ 0.500 & 0.866 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}.$$

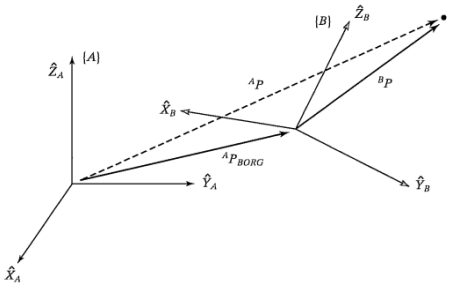
$$^{B}P = \begin{bmatrix} 0.0 \\ 2.0 \\ 0.0 \end{bmatrix},$$

we calculate ^{A}P as

$$^{A}P = {}^{A}_{B}R \; {}^{B}P = \begin{bmatrix} -1.000 \\ 1.732 \\ 0.000 \end{bmatrix}.$$

Now, the **general case**, where the systems don't have the same origin, but B is shifted by P_{BORG} from the origin (or $^{A}P_{BORG}$ when expressed in A):

$$^{A}P = {}^{A}_{B}R \; {}^{B}P + {}^{A}P_{BORG}$$



Computing a position of a point connected by multiple subsequent links of a manipulator with individual coordinate frames, involves the multiplication of sums given by the general transform equation. This is because *rotation* and *translation* are both needed to propagate from one frame to another. To simplify this calculation, a **homogeneous transform** combines rotation and translation into a matrix ${}^{A}_{B}T$, such that: $^{A}P = {}^{A}_{B}T \; {}^{B}P$. Therefore, the homogeneous transform is a *general operator*.

$$^{A}P = {}^{A}_{B}R\,{}^{B}P + {}^{A}P_{BORG}$$

$$\begin{bmatrix} {}^{A}P \\ \hline 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{Borg} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}P \\ \hline 1 \end{bmatrix}$$

$$\underset{(4\times1)}{{}^{A}P} = \underset{(4\times4)}{{}^{A}_{B}T}\ \underset{(4\times1)}{{}^{B}P}$$

Due to the generality of the homogeneous transform, it can be interpreted in different ways: 1.) as a description of a frame $\{B\} = \{{}^{A}_{B}R \;\; {}^{A}P_{BORG}\}$; 2.) as a mapping of a point in frame B to frame A; 3.) as an operator to rotate and translate a point in the same frame. This is an example for the second case:

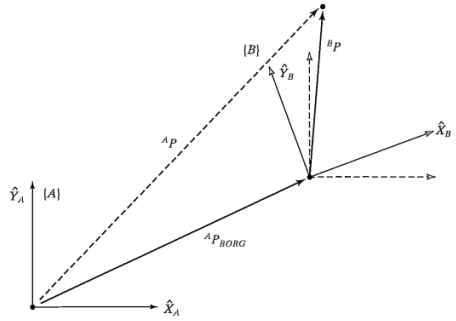


FIGURE 2.8: Frame $\{B\}$ rotated and translated.

Figure 2.8 shows a frame $\{B\}$, which is rotated relative to frame $\{A\}$ about $\hat{Z}$ by 30 degrees, translated 10 units in $\hat{X}_A$, and translated 5 units in $\hat{Y}_A$. Find ^AP, where $^BP = [3.07.00.0]^T$.

The definition of frame $\{B\}$ is

$$^{A}_{B}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 10.0 \\ 0.500 & 0.866 & 0.000 & 5.0 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.21}$$

Given

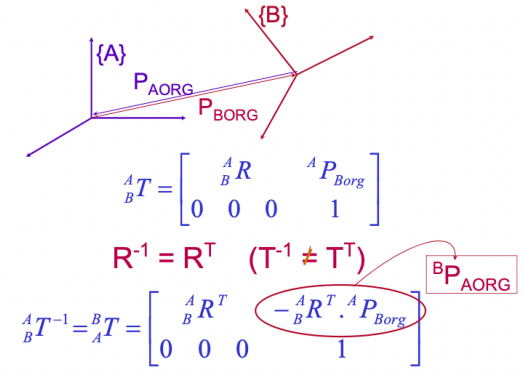
$$^{B}P = \begin{bmatrix} 3.0 \\ 7.0 \\ 0.0 \end{bmatrix}, \tag{2.22}$$

we use the definition of $\{B\}$ just given as a transformation:

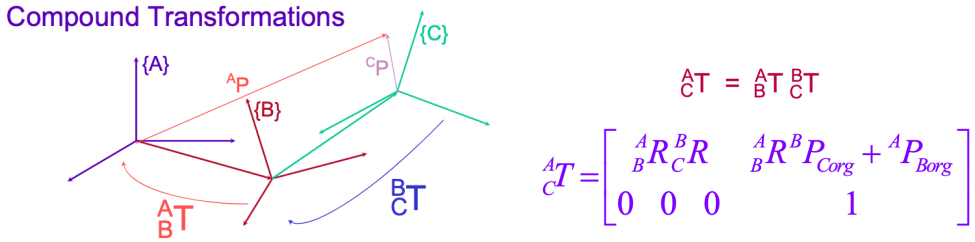
$$^{A}P = {}^{A}_{B}T\,{}^{B}P = \begin{bmatrix} 9.098 \\ 12.562 \\ 0.000 \end{bmatrix}. \tag{2.23}$$

```python
import numpy as np
from scipy.spatial.transform import Rotation as R

# Homogeneous transformation matrix from frame B to frame A
A_T_B = np.array([
    [0.866, -0.500, 0.000, 10.0],
    [0.500, 0.866, 0.000, 5.0],
    [0.000, 0.000, 1.000, 0.0],
    [0.000, 0.000, 0.000, 1.0]
])

# Point P in frame B (homogeneous coordinates)
B_P = np.array([3.0, 7.0, 0.0, 1.0])

# Transform point into frame A
A_P = A_T_B @ B_P

# Extract just the 3D coordinates
print("\nCoordinates of P in frame A:")
print(A_P[:3])
```

Inverse of the homogeneous transform:



Compound transformation:



Transform equation: ${}^A_BT\ {}^B_CT\ {}^C_DT\ {}^D_AT = I$.

```
import numpy as np
from scipy.spatial.transform import Rotation as R

# Define two homogeneous transformations

# Transformation T1: rotation by 30 degrees around z-axis, translation [10, 5, 0]
r1 = R.from_euler('z', np.pi/6)
T1 = np.eye(4)
T1[:3, :3] = r1.as_matrix()
T1[:3, 3] = [10, 5, 0]

# Transformation T2: rotation by 45 degrees around x-axis, translation [2, 3, 1]
r2 = R.from_euler('x', np.pi/4)
T2 = np.eye(4)
T2[:3, :3] = r2.as_matrix()
T2[:3, 3] = [2, 3, 1]

# Compound transformation: T_compound = T1 @ T2
T_compound = T1 @ T2
print("Compound transformation matrix:")
print(T_compound)

# Inverse of the compound transformation
T_inv = np.linalg.inv(T_compound)
print("Inverse transformation matrix:")
print(T_inv)

# Example: Apply compound transformation to a point
P = np.array([1.0, 2.0, 3.0, 1.0]) # Point in homogeneous coordinates
P_transformed = T_compound @ P
print("Point after compound transformation:", P_transformed[:3])
```

```
# Apply inverse to get back
P_back = T_inv @ P_transformed
print("Point after applying inverse:", P_back[:3])
```

3.4 Forward kinematics

Forward kinematics solves the static geometrical problem of computing the position and the orientation of the end-effector of the manipulator. Specifically, given a **set of joint angles**, the forward kinematic problem is to compute the position and orientation of the tool frame relative to the base frame.

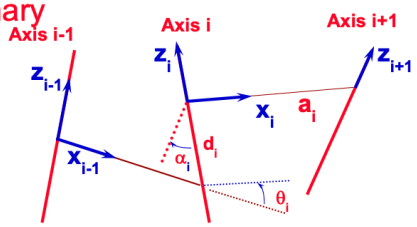
3.4.1 Denavit-Hartenberg Convention

The homogeneous transform between consecutive frames of links is represented via only 4 D-H-parameters (x-rot, x-trans, z-trans, z-rot). In general, 6 parameters are needed to represent an arbitrary rigid body transformation! Thus, a restriction of it is, that it **cannot represent a rotation around the y-axis and that the y- and z-position are coupled** [4].

Procedure for deriving the D-H-parameters:

1. Identify the joint axes and imagine (or draw) infinite lines along them. For steps 2 through 5 below, consider two of these neighboring lines (at axes i and $i+1$).
2. Identify the common perpendicular between them, or point of intersection. At the point of intersection, or at the point where the common perpendicular meets the ith axis, assign the link-frame origin.
3. Assign the $\hat{Z}_i$ axis pointing along the ith joint axis.
4. Assign the $\hat{X}_i$ axis pointing along the common perpendicular, or, if the axes intersect, assign $\hat{X}_i$ to be normal to the plane containing the two axes.
5. Assign the $\hat{Y}_i$ axis to complete a right-hand coordinate system.
6. Assign $\{0\}$ to match $\{1\}$ when the first joint variable is zero. For $\{N\}$, choose an origin location and $\hat{X}_N$ direction freely, but generally so as to cause as many linkage parameters as possible to become zero.

Summary



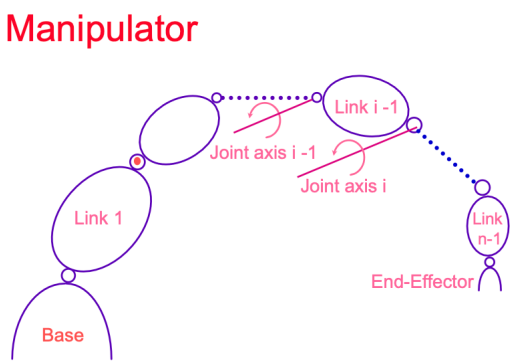
a_i = *the distance from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured along* $\hat{X}_i$;

α_i = *the angle from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured about* $\hat{X}_i$;

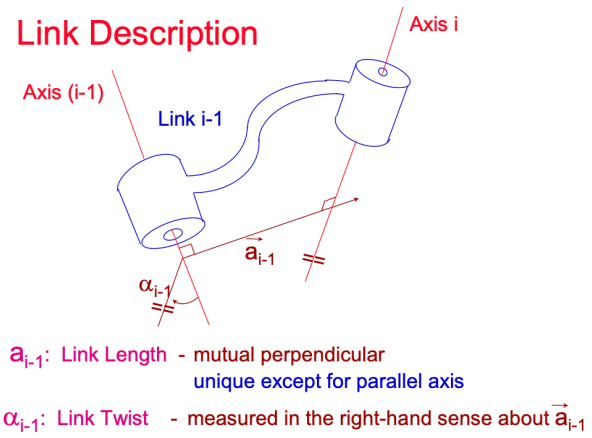
d_i = *the distance from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured along* $\hat{Z}_i$; and

θ_i = *the angle from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured about* $\hat{Z}_i$.

Among the four D-H-parameters are three fixed link parameters and one joint variable, which is θ_i for a revolute joint or d_i for a prismatic joint. The D-H-parameters are plugged into homogenous transformation matrices, which represent the single transformations between the frames of subsequent links. Then, we can propagate through all homogeneous transformations to calculate the position and orientation of the end-effector. Hereafter, the above described procedure is detailed further:

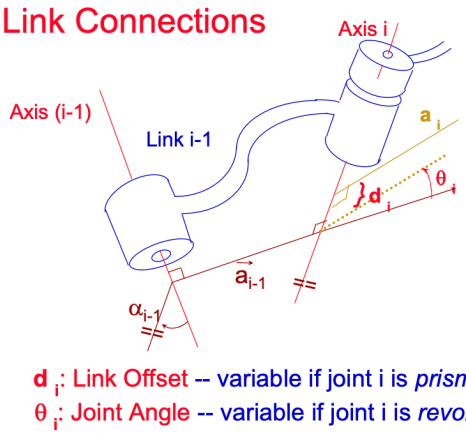


(a) Identify the joint axes; consider axes i and $i-1$. By convention, a joint axis points **in the direction of the rotation/ movement** for revolute/ prismatic joints.

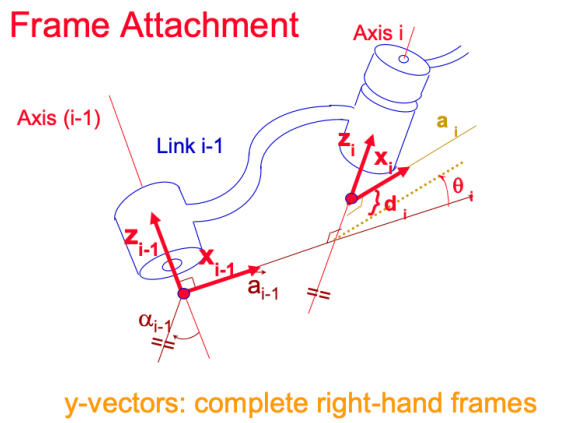


(b) Identify the common perpendicular. If the axes intersect, the common perpendicular is a normal through the plane they span and the direction of α_i is determined by the direction of this normal. a_i **and** α_i **describe the** i**th link**. ("We rotate axis $i-1$ around the normal about α so that it coincides with axis i. This aligns the Z-axes.")

Figure 2: D-H parameters assignment

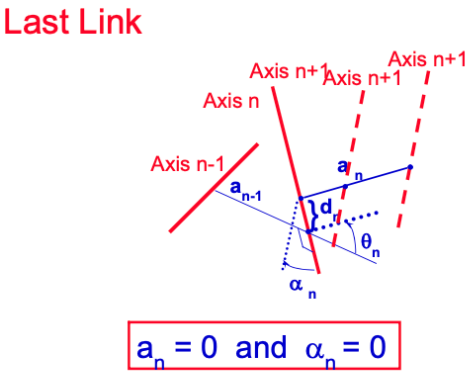


(c) d_i **and** θ_i **describe the** $i-1$**th link connection**. If the axes of link $i-1$ and link i are parallel, the origin of a coordinate frame $\{i-1\}$ should be attached, such that $d_i = 0$. ("θ_i aligns the X-axes by rotation around Z_i.")

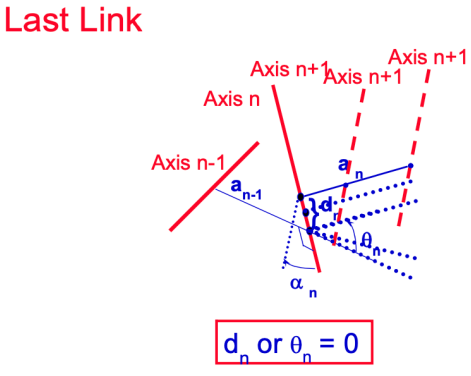


(d) Attach ith frame, such that $\hat{Z}_i$ is in the direction of the ith joint axis and $\hat{X}_i$ points along the common perpendicular (if $\hat{Z}_{i-1}$ and $\hat{Z}_i$ intersect, choose $\hat{X}_{i-1}$, such that $\alpha_{i-1} > 0$). $\hat{Y}_i$ completes the right-hand frame.

Figure 2: (Continued)

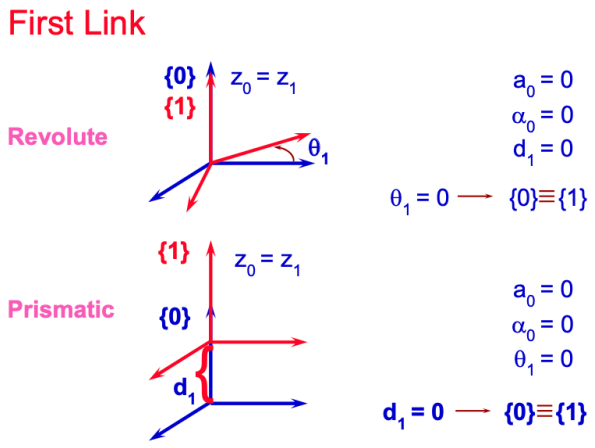


(e) a_i and α_i depend on joint axes i and $i+1$. Thus, select axes 0 and $n+1$, such that $a_0 = a_n = 0$ and $\alpha_0 = \alpha_n = 0$ (by making axis 0 coincident with axis 1 and axis $n+1$ coincident with axis n). This simplifies the forward kinematics.

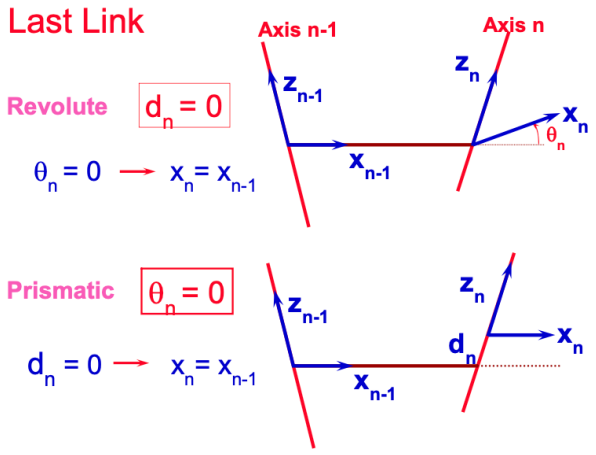


(f) θ_i and d_i depend on joint axes i and $i-1$. Again, select axes 0 and $n+1$, such that depending on the joint type a_0 or $\theta_0 = 0$ and a_n or $\theta_n = 0$ (by coinciding axes and moving the intersection point that becomes the origin of the frame so that $d = 0$ or orienting the axis so that $\theta = 0$).

Figure 2: (Continued)



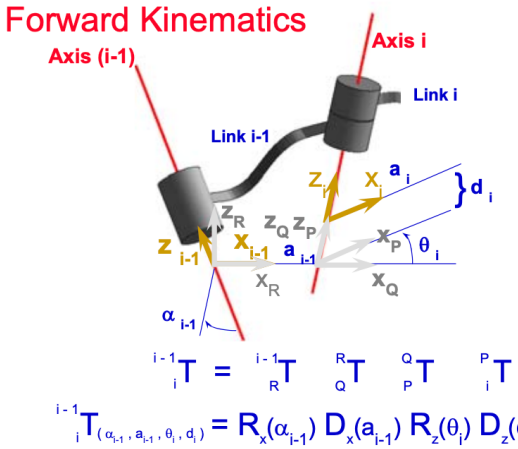
(g) $\{0\}$ is assigned, such that is equals $\{1\}$ when the first joint variable is 0.



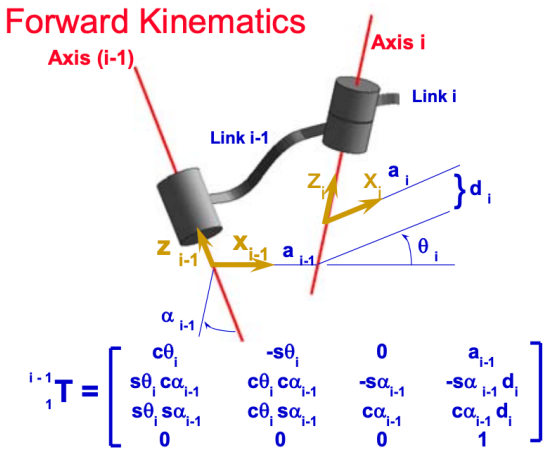
(h) Assign $\{N\}$, such that the most parameters are 0.

Figure 2: (Continued)

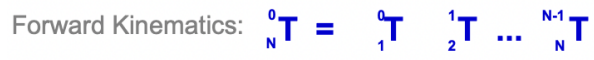
Then, the forward kinematics of a single link using the D-H-parameters derive as follows:



(a) To obtain ${}^{i-1}_{i}T$: Translate along Z_i about d_i ($\rightarrow \{P\}$), then rotate around Z_i about θ_i ($\rightarrow \{Q\}$), then translate along X_{i-1} about a_{i-1} ($\rightarrow \{R\}$) and finally rotate around X_i about α_i ($\rightarrow \{i-1\}$). This results in the homogeneous transform for a single link from $\{i\}$ to $\{i-1\}$ and also works in the other direction to compute the inverse homogeneous transform.



(b) The **homogeneous transformation from link i to link** $i-1$ (the formula in the slide should say i instead of 1).



(c) The homogeneous transformation which transforms points from the frame of link N to the frame of link 0.

Note, that the homogeneous transform ${}^{i-1}_{i}T$ cannot express arbitrary rigid body transformations, since no rotation about $\hat{Y}$ is possible. A good, practical explanation of how to place the coordinate frames at each link, derive the D-H-parameters and compute the homogeneous transformations and forward kinematics is given in this video (based on an example).

3.4.2 D-H Parameter Table for Robot Manipulators

The structural parameters of a serial chain robot manipulator can be described by a table of Denavit-Hartenberg (D-H) parameters. Each row in the table corresponds to a joint/link and specifies four parameters: a_{i-1} (link length), α_{i-1} (link twist), d_i (link offset), and θ_i (joint angle). These parameters are essential for systematically modeling the kinematics of robot arms.

Example: UR5e Robot Arm The official D-H parameters for the UR5e robot arm are as follows:[1].The table lists the link lengths (a_{i-1}), link twists (α_{i-1}), link offsets (d_i), and joint angles (θ_i) for each joint of the UR5e. The figure shows the kinematic diagram and the assignment of DH frames for the UR series robots.

The standard Denavit-Hartenberg (DH) parameters use a_{i-1}, α_{i-1}, d_i, and θ_i, while the modified DH (DHM) convention uses a_i, α_i, d_i, and θ_i, with positive link lengths preferred and potential offsets added to θ for negative lengths. RoboDK, a powerful software for robot simulation, offline programming, and

[1]Source: Universal Robots: Denavit-Hartenberg Parameters

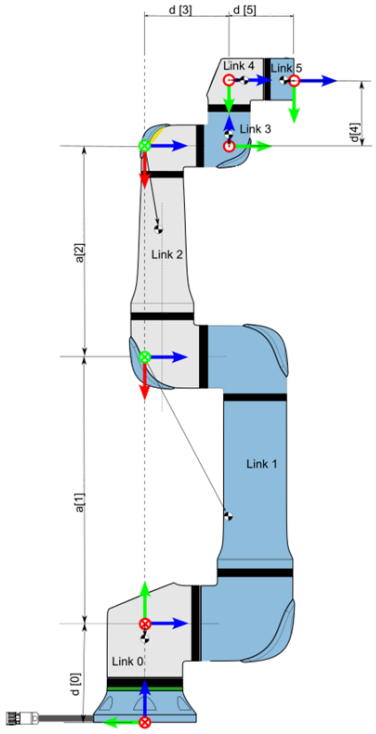


Table 1: Denavit-Hartenberg Parameters for UR5e

Joint i	a_{i-1} [m]	α_{i-1} [deg]	d_i [m]	θ_i [deg]
1	0	0	0.1625	θ_1
2	-0.425	0	0	θ_2
3	-0.3922	0	0	θ_3
4	0	90	0.1333	θ_4
5	0	-90	0.0997	θ_5
6	0	0	0.0996	θ_6

Table 2: Modified Denavit-Hartenberg Parameters (DHM) for UR5e (RoboDK)

Joint i	a_i [mm]	α_i [rad]	d_i [mm]	θ_i [rad]
1	0	0	162.5	0
2	0	$\frac{\pi}{2}$	0	π
3	425	0	0	0
4	392.25	0	133.3	0
5	0	$-\frac{\pi}{2}$	99.7	0
6	0	$\frac{\pi}{2}$	99.6	π

Figure 3: UR5e kinematic diagram (left) and DH parameter tables (right).

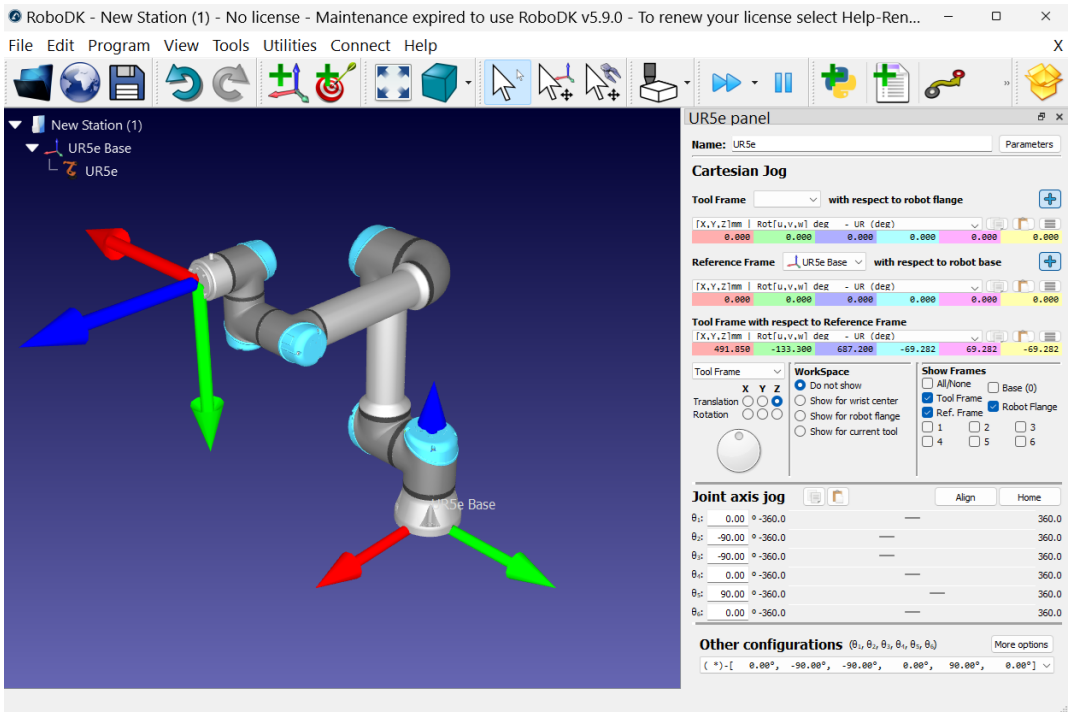
analysis, employs the DHM convention for its robot models. You can view and interact with the UR5e robot model in RoboDK's online library at RoboDK Robot Library.

```
import numpy as np
from scipy.spatial.transform import Rotation as R

# DHM parameters for UR5e (in mm and rad)
a = [0, 0, 425, 392.25, 0, 0]
alpha = [0, np.pi/2, 0, 0, -np.pi/2, np.pi/2]
d = [162.5, 0, 0, 133.3, 99.7, 99.6]
theta_offsets = [0, np.pi, 0, 0, 0, np.pi]

# Given joint angles in degrees
joint_angles_deg = [0, -90, -90, 0, 90, 0]
joint_angles_rad = np.deg2rad(joint_angles_deg)

# Add offsets to joint angles
theta = joint_angles_rad + theta_offsets

# Function to compute DH transformation matrix
def dh_transform(a, alpha, d, theta):
    T = np.eye(4)
    # Trans_x(a)
    T[0, 3] = a
    # Rot_x(alpha)
    T[:3, :3] = R.from_rotvec([alpha, 0, 0]).as_matrix()
    # Trans_z(d)
    T_z = np.eye(4)
    T_z[2, 3] = d
    T = T @ T_z
```

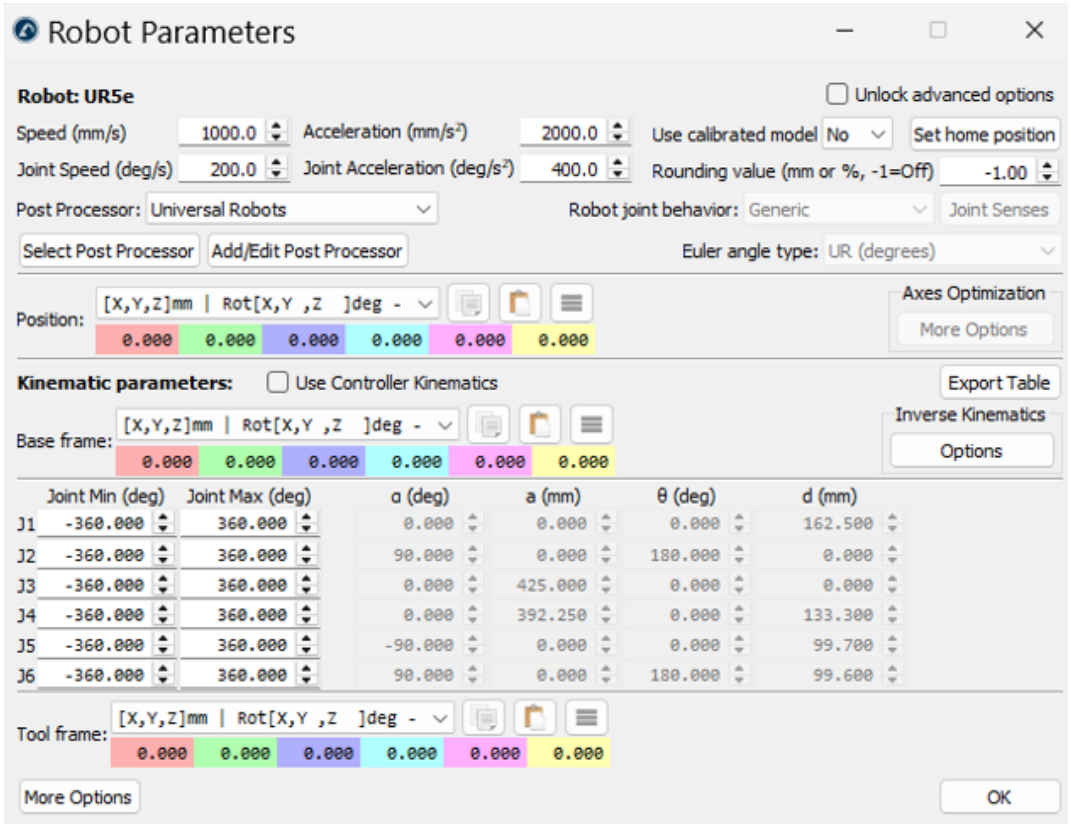
```
    # Rot_z(theta)
    T_theta = np.eye(4)
    T_theta[:3, :3] = R.from_rotvec([0, 0, theta]).as_matrix()
    T = T @ T_theta
    return T

# Compute total transformation
T_total = np.eye(4)
for i in range(6):
    T_i = dh_transform(a[i], alpha[i], d[i], theta[i])
    T_total = T_total @ T_i

print("End-effector transformation matrix:")
print(T_total)
```

The expected output is:

```
[[ 0.  0.  1. 491.85]
 [-1.  0.  0. -133.30]
 [ 0. -1.  0. 687.20]
 [ 0.  0.  0.  1.]]
```



(a) RoboDK software GUI.



(b) UR5e robot with DHM parameters in RoboDK.

Figure 4: RoboDK software interface (top) and UR5e robot model with DHM parameters (bottom).

Chapter 4: Inverse Kinematics

Recap forward kinematics: Given a joint configuration, find the pose of some part of the robot.
Inverse kinematics: Given a pose, figure out the joint configurations.

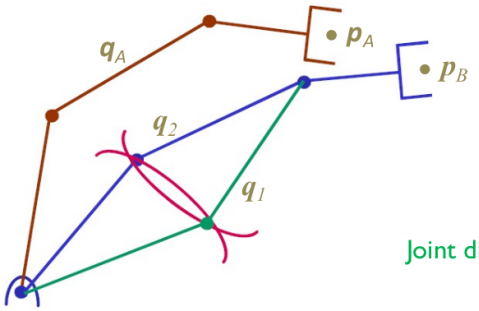
Input data for the problem is of the form:

$$T = \begin{bmatrix} R & t \\ 0\,0\,0 & 1 \end{bmatrix}$$

This is a nonlinear problem, thus it's not sure if there is a (unique/multiple/infinite/outside-of-workspace/none) solution. Inverse kinematics are not explicitly discussed the Stanford lecture and there is not particular exercise on it (albeit it is included in the lecture).

4.1 Multiplicity of Solutions

If there are multiple solutions (configurations), choose the closest one:



Initial configuration for p_A: q_A

New configurations for p_B: q_1, q_2

Is it better to choose q_1 or q_2? q_2

Joint distance between configurations:

$$d_1 \triangleq \|\mathbf{q}_1 - \mathbf{q}_A\|$$
$$d_2 \triangleq \|\mathbf{q}_2 - \mathbf{q}_A\|$$

Shortest distance is preferred

In general, if there are N possible configurations for $\mathbf{p}_B$, choose:

$$\mathbf{q}_b = \arg\min_{\mathbf{q}} \|\mathbf{q} - \mathbf{q}_A\| \quad \text{for} \quad \mathbf{q} \in \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N\}$$

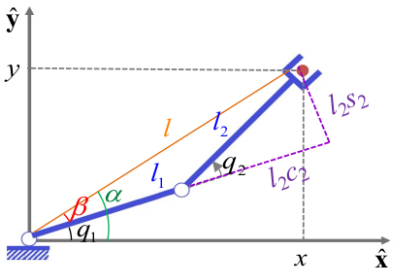
4.2 Analytic solutions for the inverse kinematics

4.2.1 Geometric solution

- only when robot has 3 or less dofs
- not a generic solution

Example:

Find the inverse kinematics for the position of the R-R robot using a geometric approach



Solution

For q_2:

- Using the law of cosines:

$$l^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(180 - q_2)$$

$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1 l_2 \cos(q_2)$$

$$c_2 = \frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1 l_2}$$

- Using a trigonometric identity:

$$s_2^2 + c_2^2 = 1$$

$$s_2 = \pm\sqrt{1 - c_2^2}$$

$$q_2 = \text{atan2}(s_2, c_2)$$

For q_1 (using the geometry of the figure)

$$q_1 = \alpha - \beta$$

$$\alpha = \text{atan2}(y, x)$$

$$\beta = \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$

Inverse kinematics:

$$q_1 = \text{atan2}(y, x) - \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$

$$q_2 = \text{atan2}(s_2, c_2)$$

4.2.2 Algebraic solution

Solution using algebraic (and polynomial) equations.
Given: Formula to find the kinematic equations of an arm easily, given link params. E.g.:

$$^{B}_{W}T = {}^{0}_{3}T = \begin{bmatrix} c_{123} & -s_{123} & 0.0 & l_1 c_1 + l_2 c_{12} \\ s_{123} & c_{123} & 0.0 & l_1 s_1 + l_2 s_{12} \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This describes the wrist (manipulator) frame relative to the base frame.
We also know what is necessary to describe such a position/orientation:

$$^{B}_{W}T = \begin{bmatrix} c_\phi & -s_\phi & 0.0 & x \\ s_\phi & c_\phi & 0.0 & y \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{\text{i}}$$

So just an x-y positon and an angle ϕ. By equating both, we get a set of nonlinear equations, which have to be solved for l_1, l_2, θ, using magical algebra:

$$c_\phi = c_{123},$$

$$s_\phi = s_{123},$$

$$x = l_1 c_1 + l_2 c_{12}$$

$$y = l_1 s_1 + l_2 s_{12}$$

4.2.3 Example

For a 3-*dof* robot, its end effector pose with respect to its base is given by

$$ {}^{0}\boldsymbol{T}_{3} = \begin{bmatrix} \cos(\theta_1+\theta_3) & 0 & \sin(\theta_1+\theta_3) & a_3\cos(\theta_1+\theta_3)+q_2\sin\theta_1 \\ \sin(\theta_1+\theta_3) & 0 & -\cos(\theta_1+\theta_3) & a_3\sin(\theta_1+\theta_3)-q_2\cos(\theta_1) \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where a_3 and d_1 are constants. The desired pose for the end effector is:

$$ T_{des} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Find the inverse kinematics () of this robot as a function of the elements of T_{des}.

- Finding :

$$ \begin{cases} p_x = a_3c_{13} + s_1q_2 \longrightarrow p_x - a_3c_{13} = s_1q_2 \\ p_y = a_3s_{13} - c_1q_2 \longrightarrow a_3s_{13} - p_y = c_1q_2 \end{cases} $$

$$ \tan(\theta_1) = \frac{p_x - a_3c_{13}}{a_3s_{13} - p_y} = \frac{p_x - a_3n_x}{a_3n_y - p_y} \Rightarrow \theta_1 = \text{atan2}(p_x - a_3n_x, a_3n_y - p_y) $$

- Finding θ_3:

$$ \frac{a_x}{n_x} = \tan(\theta_1+\theta_3) \longrightarrow \theta_1+\theta_3 = \text{atan2}(a_x, n_x) $$

$$ \theta_3 = \text{atan2}(a_x, n_x) - \theta_1 $$

- Finding θ_2:

$$ \begin{cases} s_1p_x = a_3s_1c_{13} + s_1^2q_2 \\ c_1p_y = a_3s_{13}c_1 - c_1^2q_2 \end{cases} $$

$$ s_1p_x - c_1p_y = a_3(s_1n_x - n_yc_1) + q_2 $$

$$ q_2 = s_1p_x - c_1p_y - a_3(s_1n_x - n_yc_1) $$

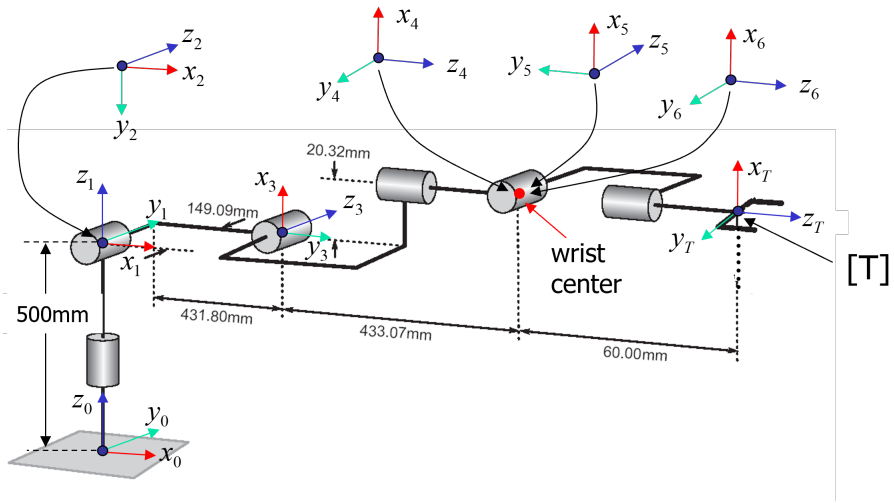
4.3 PUMA 560 Inverse Kinematics

4.3.1 DH Parameters

The PUMA 560 is a 6-DOF manipulator. Its Denavit-Hartenberg parameters are given in the following table:

Link	α_i (deg)	a_i (mm)	d_i (mm)	θ_i (deg)
1	0	0	0	θ_1
2	-90	0	0	θ_2
3	0	431.8	149.09	θ_3
4	-90	20.3	433.07	θ_4
5	90	0	0	θ_5
6	-90	0	0	θ_6

Table 3: DH Parameters for PUMA 560



4.3.2 Forward Kinematics

The forward kinematics for the PUMA 560 involves the following key steps:

1. Define the Denavit-Hartenberg (DH) parameters for each joint-link pair.
2. Compute the homogeneous transformation matrix for each link using the DH parameters. The transformation matrix for each link is:

$$[{}^{i-1}_{i}T] = [X(a_i, \alpha_i)][Z(d_i, \theta_i)] = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_i \\ \sin\theta_i \cos\alpha_i & \cos\theta_i \cos\alpha_i & -\sin\alpha_i & -d_i \sin\alpha_i \\ \sin\theta_i \sin\alpha_i & \cos\theta_i \sin\alpha_i & \cos\alpha_i & d_i \cos\alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Multiply the transformation matrices sequentially to obtain the end-effector pose:

$$[{}^{0}_{6}T] = [{}^{0}_{1}T] \cdot [{}^{1}_{2}T] \cdot [{}^{2}_{3}T] \cdot [{}^{3}_{4}T] \cdot [{}^{4}_{5}T] \cdot [{}^{5}_{6}T]$$

.

4. Multiply The Base Frame and The Tool Frame:

$$T = G \times [{}^{0}_{6}T] \times H$$

where G is base transform matrix and H is the transform from the wrist center to the end-effector (tool) frame.

4.3.3 Inverse Kinematics Solution

The inverse kinematics for the PUMA 560 is solved analytically using the following steps:

1. **Remove Base/Tool Offsets**: Compute the effective transformation by removing the base and tool transforms from the target pose:

$$[{}^{0}_{6}T] = G^{-1} \cdot T \cdot H^{-1}$$

2. **Extract Wrist Center**: The wrist center position is extracted from the 4th columns of the transformation matrix:
$$\mathbf{p}_{wc} = [{}^0_6T][1:3,4] = (p_x, p_y, p_z)$$
3. **Solve for (θ_1)**: Use a trigonometric equation to solve for possible values of θ_1 (θ_1^+, θ_1^-):
$$-\sin\theta_1 \cdot p_x + \cos\theta_1 \cdot p_y = d_3$$
which is in the form of $A\cos(x) + B\sin(x) = C$ and can be solved analytically using the formula given in the appendix.
4. **Solve for (θ_3)**: Use a distance equation and trigonometric solver to find possible θ_3 values (θ_3^+, θ_3^-):
$$2a_2(a_3\cos\theta_3 - d_4\sin\theta_3) = (p_x^2 + p_y^2 + p_z^2) - (a_2^2 + a_3^2 + d_3^2 + d_4^2)$$
rewritten as: $A\cos\theta_3 + B\sin\theta_3 = C$, with
$$A = 2a_2a_3, \quad B = -2a_2d_4, \quad C = (p_x^2 + p_y^2 + p_z^2) - (a_2^2 + a_3^2 + d_3^2 + d_4^2)$$
The solutions can be found using the trigonometric equation solver provided in the appendix.
5. **Solve for (θ_2)**: For each (θ_1, θ_3) pair, solve the following linear equations for c_2 and s_2 to obtain $\theta_2^{1,2,3,4}$:
$$c_2(c_1p_x + s_1p_y) - s_2p_z = a_2 + a_3\cos\theta_3 - d_4\sin\theta_3$$
$$-s_2(c_1p_x + s_1p_y) - c_2p_z = a_3\sin\theta_3 + d_4\cos\theta_3$$
And then solve for θ_2 by
$$\theta_2 = \text{atan2}(s_2, c_2)$$
where θ_2^1 for (θ_1^+, θ_3^+), θ_2^2 for (θ_1^+, θ_3^-), θ_2^3 for (θ_1^-, θ_3^+), θ_2^4 for (θ_1^-, θ_3^-).
6. **Compute Wrist Rotation**: For each of the 4 sets (θ_1, θ_2^k, θ_3) where $k = 1, 2, 3, 4$, calculate the wrist rotation matrix and solve for the last three joint angles (θ_4, θ_5, θ_6) using spherical wrist formulas (2 solutions for each set):
$$[{}^0_3T] = [{}^0_1T] \cdot [{}^1_2T] \cdot [{}^2_3T] \quad R_0^3 = [{}^0_3T][1:3,1:3]$$
$$R_0^6 = [{}^0_6T][1:3,1:3]$$
$$R_3^6 = Inv(R_0^3) \cdot R_0^6$$
For the spherical wrist:
   - $\theta_5^{\pm} = \pm\arccos(R_3^6[3,3])$
   - $\theta_4^{\pm} = \text{atan2}(\frac{R_3^6[2,3]}{-\sin\theta_5}, \frac{R_3^6[1,3]}{-\sin\theta_5})$
   - $\theta_6^{\pm} = \text{atan2}(\frac{R_3^6[3,2]}{-\sin\theta_5}, \frac{R_3^6[3,1]}{\sin\theta_5})$

In summary, the PUMA 560 inverse kinematics yields up to 8 solutions: θ_1^+, θ_1^-, θ_3^+, θ_3^-, leading to θ_2^1, θ_2^2, θ_2^3, θ_2^4, and for each of these 4 sets, two solutions for the wrist angles ($\theta_4^{\pm}$, $\theta_5^{\pm}$, $\theta_6^{\pm}$).

The Python implementation is available in the [GitHub repository:] https://github.com/haijunsu-osu/ik_puma560_notes. You can also explore and run the code interactively in Google Colab: [Open in Colab]https://colab.research.google.com/github/haijunsu-osu/ik_puma560_notes/blob/main/PUMA560KinemaFollowNotes_Answer.ipynb.

Chapter 5: Formula Cheat Sheet

Cross product: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$

Trigonometric Identities:

$$\sin(\alpha)^2 + \cos(\alpha)^2 = 1$$
$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$$
$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$$

Determinant:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
$$= aei + bfg + cdh - ceg - bdi - afh$$

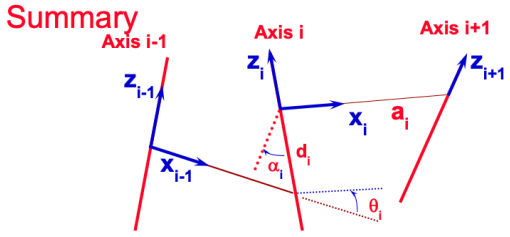
A manipulator may have special configurations, called "isotropic points", that are characterized by the Jacobi matrix having orthogonal columns of equal length, thus $J^TJ = \delta I$ for some $\delta \in \mathbb{R}$.

5.1 Denavit-Hartenberg Parameters

DH-table (link-index i):

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

1. Shift Z_{i-1} by a_{i-1} along X_{i-1}.
2. Rot. Z_{i-1} by α_{i-1} about X_{i-1} & shift by d_i along Z_i.
3. Rot. X_{i-1} by θ_i about Z_i & move it to Z_i.



a_i = *the distance from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured along* $\hat{X}_i$;

α_i = *the angle from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured about* $\hat{X}_i$;

d_i = *the distance from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured along* $\hat{Z}_i$; and

θ_i = *the angle from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured about* $\hat{Z}_i$.

Homogeneous transformation from link i to $i-1$:

$$^{i-1}_{i}T = \left[\begin{array}{ccc|c} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i\ c\alpha_{i-1} & c\theta_i\ c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}\ d_i \\ s\theta_i\ s\alpha_{i-1} & c\theta_i\ s\alpha_{i-1} & c\alpha_{i-1} & d_i\ c\alpha_{i-1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right].$$

Inverse of the homogeneous transform:

$$^{A}_{B}T^{-1} = {}^{B}_{A}T = \left[\begin{array}{ccc|c} & {}^{A}_{B}R^T & & -{}^{A}_{B}R^T\,{}^{A}P_{Borg} \\ \hline 0 & 0 & 0 & 1 \end{array}\right].$$

5.2 Jacobian

Singularity: the end-effector locally looses at least 1 DOF. This happens, when Z-axes are aligned $\Leftrightarrow$ the Jacobian does not have full rank $\Leftrightarrow$ $det(J) = 0$. Small end-effector motions require large joint motions near singularities.

5.2.1 Velocity Propagation

Linear and angular velocities at joint $i+1$ (with scalar $\dot{d}_{i+1}$ or $\dot{\Theta}_{i+1}$ for prismatic/ revolute joints):

$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}R \cdot {}^{i}\omega_i + \dot{\Theta}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1}$$
$$^{i+1}v_{i+1} = {}^{i+1}_{i}R({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) + \dot{d}_{i+1} \cdot {}^{i+1}\hat{Z}_{i+1}$$

Then read off the Jacobian from: $$\begin{bmatrix} \dot{x}_P \\ \dot{x}_R \end{bmatrix}_{6\times 1} = \begin{bmatrix} J_{x_P} \\ J_{x_R} \end{bmatrix}_{6\times n} \dot{\boldsymbol{\Theta}} = J_{6\times n} \begin{bmatrix} \dot{\Theta}_1 \\ \dots \\ \dot{\Theta}_n \end{bmatrix}_{n\times 1}.$$

5.2.2 Force/ Torque Propagation

Force f and torque n at joint i:
$$^i f_i = {}^i_{i+1}R \cdot {}^{i+1}f_{i+1}$$
$$^i n_i = {}^i_{i+1}R \cdot {}^{i+1}n_{i+1} + {}^i P_{i+1} \times {}^i f_i$$

The force/ torque exerted on the ith joint:
$$\tau_i = {}^i f_i^{\mathrm{T}\,i} Z_i = {}^i f_i^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad \tau_i = {}^i n_i^{\mathrm{T}\,i} Z_i = {}^i n_i^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Then read off the Jacobian from: $$\begin{pmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{pmatrix} = \tau = {}^A J^{TA}\mathcal{F} = {}^A J^T \begin{pmatrix} {}^A f \\ {}^A n \end{pmatrix},$$
where $\mathcal{F}$ is a 6×1 force-torque vector in frame $\{A\}$.

5.2.3 Explicit Form

Jacobian in frame $\{0\}$: $$^0 J = \begin{bmatrix} \frac{\partial}{\partial q_1}\left({}^0 x_P\right) & \frac{\partial}{\partial q_2}\left({}^0 x_P\right) & \cdots & \frac{\partial}{\partial q_n}\left({}^0 x_P\right) \\ \bar{\epsilon}_1 \cdot \left({}^0_1 R \cdot Z\right) & \bar{\epsilon}_2 \cdot \left({}^0_2 R \cdot Z\right) & \cdots & \bar{\epsilon}_n \cdot \left({}^0_n R \cdot Z\right] \end{bmatrix}$$

with ${}^0 Z_i = {}^0_i R^i Z_i$; $\quad {}^i Z_i = Z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. The indicator variable $\bar{\epsilon}_i$ is 1 if joint i is revolute, otherwise it is 0.

Then, rotate ${}^0 J$ into the required reference frame:
$$^A J(\Theta) = \begin{pmatrix} {}^A_B R & \mathbf{0} \\ \mathbf{0} & {}^A_B R \end{pmatrix} {}^B J(\Theta)$$

5.3 Solutions to Common Trigonometric Equations

To solve equations of the form $A\cos\theta + B\sin\theta = C$, use the identity:
$$\sqrt{A^2+B^2}\cos(\theta - \phi) = C$$

where $\tan\phi = B/A$.
Then, $\theta = \phi + \arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right)$ or $\theta = \phi - \arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right)$.
For the PUMA 560, this is used to solve for θ_3.

Chapter 6: Appendices

6.1 Euler Angles

Euler angles are used to describe the orientation of a rigid body with respect to a fixed coordinate system. There are 12 sets of Euler angles (extrinsic) and 12 sets of fixed (intrinsic) Euler angles, each corresponding to different rotation sequences.

$$R_{X'Y'Z'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\beta c\gamma & -c\beta s\gamma & s\beta \\ s\alpha s\beta c\gamma + c\alpha s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta \\ -c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma & c\alpha c\beta \end{bmatrix},$$

$$R_{X'Z'Y'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\beta c\gamma & -s\beta & c\beta s\gamma \\ c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma \\ s\alpha s\beta c\gamma - c\alpha s\gamma & s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma \end{bmatrix},$$

$$R_{Y'X'Z'}(\alpha,\beta,\gamma)=\begin{bmatrix} s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & s\alpha c\beta \\ c\beta s\gamma & c\beta c\gamma & -s\beta \\ c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha c\beta \end{bmatrix},$$

$$R_{Y'Z'X'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\alpha c\beta & -c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma \\ s\beta & c\beta c\gamma & -c\beta s\gamma \\ -s\alpha c\beta & s\alpha s\beta c\gamma + c\alpha s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma \end{bmatrix}$$

$$R_{Z'X'Y'}(\alpha,\beta,\gamma)=\begin{bmatrix} -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta & s\alpha s\beta c\gamma + c\alpha s\gamma \\ c\alpha s\beta s\gamma + s\alpha c\gamma & c\alpha c\beta & -c\alpha s\beta c\gamma + s\alpha s\gamma \\ -c\beta s\gamma & s\beta & c\beta c\gamma \end{bmatrix}$$

$$R_{Z'Y'X'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix},$$

(a) Extrinsic Euler angles set 1

$$R_{X'Y'X'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\beta & s\beta s\gamma & s\beta c\gamma \\ s\alpha s\beta & -s\alpha c\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta c\gamma - c\alpha s\gamma \\ -c\alpha s\beta & c\alpha c\beta s\gamma + s\alpha c\gamma & c\alpha c\beta c\gamma - s\alpha s\gamma \end{bmatrix}$$

$$R_{X'Z'X'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\beta & -s\beta c\gamma & s\beta s\gamma \\ c\alpha s\beta & c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma \\ s\alpha s\beta & s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma \end{bmatrix},$$

$$R_{Y'X'Y'}(\alpha,\beta,\gamma)=\begin{bmatrix} -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta & s\alpha c\beta c\gamma + c\alpha s\gamma \\ s\beta s\gamma & c\beta & -s\beta c\gamma \\ -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta & c\alpha c\beta c\gamma - s\alpha s\gamma \end{bmatrix},$$

$$R_{Y'Z'Y'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha s\beta & c\alpha c\beta s\gamma + s\alpha c\gamma \\ s\beta c\gamma & c\beta & s\beta s\gamma \\ -s\alpha c\beta c\gamma - c\alpha s\gamma & s\alpha s\beta & -s\alpha c\beta s\gamma + c\alpha c\gamma \end{bmatrix},$$

$$R_{Z'X'Z'}(\alpha,\beta,\gamma)=\begin{bmatrix} -s\alpha c\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta c\gamma - c\alpha s\gamma & s\alpha s\beta \\ c\alpha c\beta s\gamma + s\alpha c\gamma & c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha s\beta \\ s\beta s\gamma & s\beta c\gamma & c\beta \end{bmatrix},$$

$$R_{Z'Y'Z'}(\alpha,\beta,\gamma)=\begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix}.$$

(b) Extrinsic Euler angles set 2

Figure 5: 12 sets of Euler angles (extrinsic)

$$R_{XYZ}(\gamma,\beta,\alpha)=\begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix},$$

$$R_{XZY}(\gamma,\beta,\alpha)=\begin{bmatrix} c\alpha c\beta & -c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma \\ s\beta & c\beta c\gamma & -c\beta s\gamma \\ -s\alpha c\beta & s\alpha s\beta c\gamma + c\alpha s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma \end{bmatrix}$$

$$R_{YXZ}(\gamma,\beta,\alpha)=\begin{bmatrix} -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta & s\alpha s\beta c\gamma + c\alpha s\gamma \\ c\alpha s\beta s\gamma + s\alpha c\gamma & c\alpha c\beta & -c\alpha s\beta c\gamma + s\alpha s\gamma \\ -c\beta s\gamma & s\beta & c\beta c\gamma \end{bmatrix}$$

$$R_{YZX}(\gamma,\beta,\alpha)=\begin{bmatrix} c\beta c\gamma & -s\beta & c\beta s\gamma \\ c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma \\ s\alpha s\beta c\gamma - c\alpha s\gamma & s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma \end{bmatrix},$$

$$R_{ZXY}(\gamma,\beta,\alpha)=\begin{bmatrix} s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & s\alpha c\beta \\ c\beta s\gamma & c\beta c\gamma & -s\beta \\ c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha c\beta \end{bmatrix},$$

$$R_{ZYX}(\gamma,\beta,\alpha)=\begin{bmatrix} c\beta c\gamma & -c\beta s\gamma & s\beta \\ s\alpha s\beta c\gamma + c\alpha s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta \\ -c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma & c\alpha c\beta \end{bmatrix},$$

$$R_{XYX}(\gamma,\beta,\alpha)=\begin{bmatrix} c\beta & s\beta s\gamma & s\beta c\gamma \\ s\alpha s\beta & -s\alpha c\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta c\gamma - c\alpha s\gamma \\ -c\alpha s\beta & c\alpha c\beta s\gamma + s\alpha c\gamma & c\alpha c\beta c\gamma - s\alpha s\gamma \end{bmatrix},$$

$$R_{XZX}(\gamma,\beta,\alpha)=\begin{bmatrix} c\beta & -s\beta c\gamma & s\beta s\gamma \\ c\alpha s\beta & c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma \\ s\alpha s\beta & s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma \end{bmatrix},$$

$$R_{YXY}(\gamma,\beta,\alpha)=\begin{bmatrix} -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta & s\alpha c\beta c\gamma + c\alpha s\gamma \\ s\beta s\gamma & c\beta & -s\beta c\gamma \\ -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta & c\alpha c\beta c\gamma - s\alpha s\gamma \end{bmatrix},$$

$$R_{YZY}(\gamma,\beta,\alpha)=\begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha s\beta & c\alpha c\beta s\gamma + s\alpha c\gamma \\ s\beta c\gamma & c\beta & s\beta s\gamma \\ -s\alpha c\beta c\gamma - c\alpha s\gamma & s\alpha s\beta & -s\alpha c\beta s\gamma + c\alpha c\gamma \end{bmatrix},$$

$$R_{ZXZ}(\gamma,\beta,\alpha)=\begin{bmatrix} -s\alpha c\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta c\gamma - c\alpha s\gamma & s\alpha s\beta \\ c\alpha c\beta s\gamma + s\alpha c\gamma & c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha s\beta \\ s\beta s\gamma & s\beta c\gamma & c\beta \end{bmatrix},$$

$$R_{ZYZ}(\gamma,\beta,\alpha)=\begin{bmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{bmatrix}.$$

Figure 6: 12 sets of fixed (intrinsic) Euler angles

6.2 Algebraic Formula

Cross product: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$

Trigonometric Identities:

$$\sin(\alpha)^2 + \cos(\alpha)^2 = 1$$
$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$$
$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$$

Determinant:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
$$= aei + bfg + cdh - ceg - bdi - afh$$

To compute atan2(a, b), use the following definition:

$$\text{atan2}(a,b) = \begin{cases} \arctan\left(\frac{a}{b}\right) & \text{if } b > 0 \\ \frac{\pi}{2} & \text{if } b = 0, a > 0 \\ \text{undefined} & \text{if } b = 0, a = 0 \\ -\frac{\pi}{2} & \text{if } b = 0, a < 0 \\ \arctan\left(\frac{a}{b}\right) + \pi & \text{if } b < 0 \end{cases}$$

6.3 Solutions to Common Trigonometric Equations

The following formula presents solutions to some commonly used algebraic (trigonometric) equations in robotics, as compiled from Craig's book [4].

The single equation

$$\sin\theta = a \tag{C.1}$$

has two solutions, given by

$$\theta = \pm\text{Atan2}(\sqrt{1-a^2}, a). \tag{C.2}$$

Likewise, given

$$\cos\theta = b, \tag{C.3}$$

there are two solutions:

$$\theta = \text{Atan2}(b, \pm\sqrt{1-b^2}). \tag{C.4}$$

If both (C.1) and (C.3) are given, then there is a unique solution given by

$$\theta = \text{Atan2}(a, b). \tag{C.5}$$

The transcendental equation

$$a\cos\theta + b\sin\theta = 0 \tag{C.6}$$

has the two solutions

$$\theta = \text{Atan2}(a, -b) \tag{C.7}$$

and

$$\theta = \text{Atan2}(-a, b). \tag{C.8}$$

The equation

$$a\cos\theta + b\sin\theta = c, \tag{C.9}$$

which we solved in Section 4.5 with the tangent-of-the-half-angle substitutions, is also solved by

$$\theta = \text{Atan2}(b, a) \pm \text{Atan2}(\sqrt{a^2+b^2-c^2}, c). \tag{C.10}$$

The set of equations

$$\begin{aligned} a\cos\theta - b\sin\theta &= c, \\ a\sin\theta + b\cos\theta &= d, \end{aligned} \tag{C.11}$$

which was solved in Section 4.4, also is solved by

$$\theta = \text{Atan2}(ad - bc,\ ac + bd). \tag{C.12}$$

References

[1] K. J. Waldron, G. L. Kinzel, and S. K. Agrawal, *Kinematics, Dynamics, and Design of Machinery*. Hoboken, NJ, USA: Wiley, 3rd ed., 2016.

[2] J. M. McCarthy and G. S. Soh, *Geometric Design of Linkages*. New York, NY, USA: Springer, 2011.

[3] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge, UK: Cambridge University Press, 2017.

[4] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Upper Saddle River, NJ, USA: Pearson-/Prentice Hall, 3rd ed., 2005.