MCT344/MCT342/CSE373/CSE471

Industrial Robotics

Lecture 3: Forward Kinematics for Robotics

Ain Shams University
Faculty of Engineering

Presented By: Dr. Diaa Emad

Introduction

- The problem of manipulator kinematics is to describe the motion of a manipulator without consideration of the forces and torques causing the motion.
- The forward kinematics problem for serial link manipulators is to determine the position and orientation of the end effector given the values for the joint variables of the robot.
- This problem is easily solved by attaching coordinate frames to each link of the robot and expressing the relationships among these frames as homogeneous transformations.
- We use a systematic procedure, known as the **Denavit-Hartenberg** convention, to attach these coordinate frames to the robot.
- The position and orientation of the robot end effector is then reduced to a matrix multiplication of homogeneous transformations.

Kinematic Chains

- A robot manipulator is composed of a set of links connected together by joints.
- The joints can either be very simple, such as a revolute joint or a prismatic joint, or they can be more complex, such as a ball and socket joint.
- The difference between the two situations is that in the first instance the joint has only a single degree-of-freedom of motion.
- In contrast, a ball and socket joint has two degrees of freedom.
- It is assumed throughout that all joints have only a single degree of freedom.
- Since joints such as a ball and socket joint (two degrees of freedom) or a spherical wrist (three degrees of freedom) can always be thought of as a succession of single degree-of-freedom joints with links of length zero in between.

Kinematic Chains

- A robot manipulator with n joints will have $n+1$ links, since each joint connects two links.
- We number the joints from 1 to n, and we number the links from 0 to n, starting from the base.
- By this convention, joint i connects link $i-1$ to link i.
- We will consider the location of joint i to be fixed with respect to link $i-1$.
- When joint i is actuated, link i moves.
- Link 0 (the first link or base) is fixed, and does not move when the joints are actuated.
- We attach a coordinate frame rigidly to each link. In particular, we attach $o_i x_i y_i z_i$ to link i.
- When joint i is actuated, link i and its attached frame, $o_i x_i y_i z_i$, experience a resulting motion.
- The frame $o_0 x_0 y_0 z_0$, which is attached to the robot base, is referred to as the base frame, inertial frame or world frame.

Kinematic Chains

- A_i is the homogeneous transformation matrix that gives the position and orientation of $o_i x_i y_i z_i$ with respect to $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$.
- A_i is a function of only a single joint variable, namely q_i

$$A_i = A_i(q_i)$$

Homogeneous Transformation Matrix:

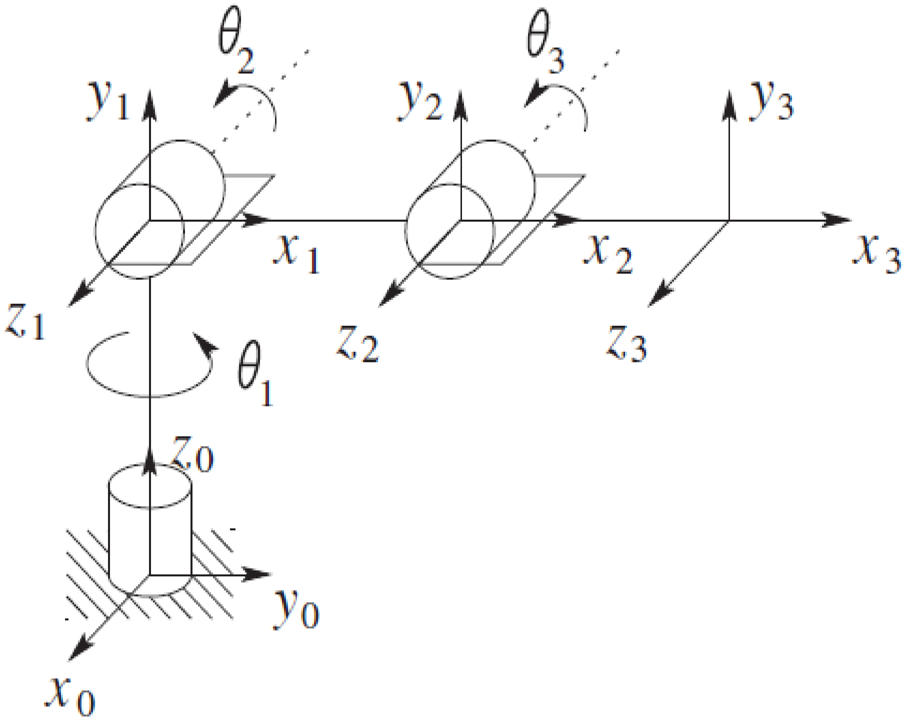
$$H = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix}$$

The position and orientation of the end effector in the inertial frame are given by the product:

$$H = T_n^0 = A_1(q_1) \cdots A_n(q_n)$$

Each homogeneous transformation A_i is of the form:

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix}$$



The Denavit-Hartenberg Convention

$$q_i = \begin{cases} \theta_i & \text{if joint } i \text{ is revolute} \\ d_i & \text{if joint } i \text{ is prismatic} \end{cases}$$

$$\begin{aligned}
A_i &= \text{Rot}_{z,\theta_i}\text{Trans}_{z,d_i}\text{Trans}_{x,a_i}\text{Rot}_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

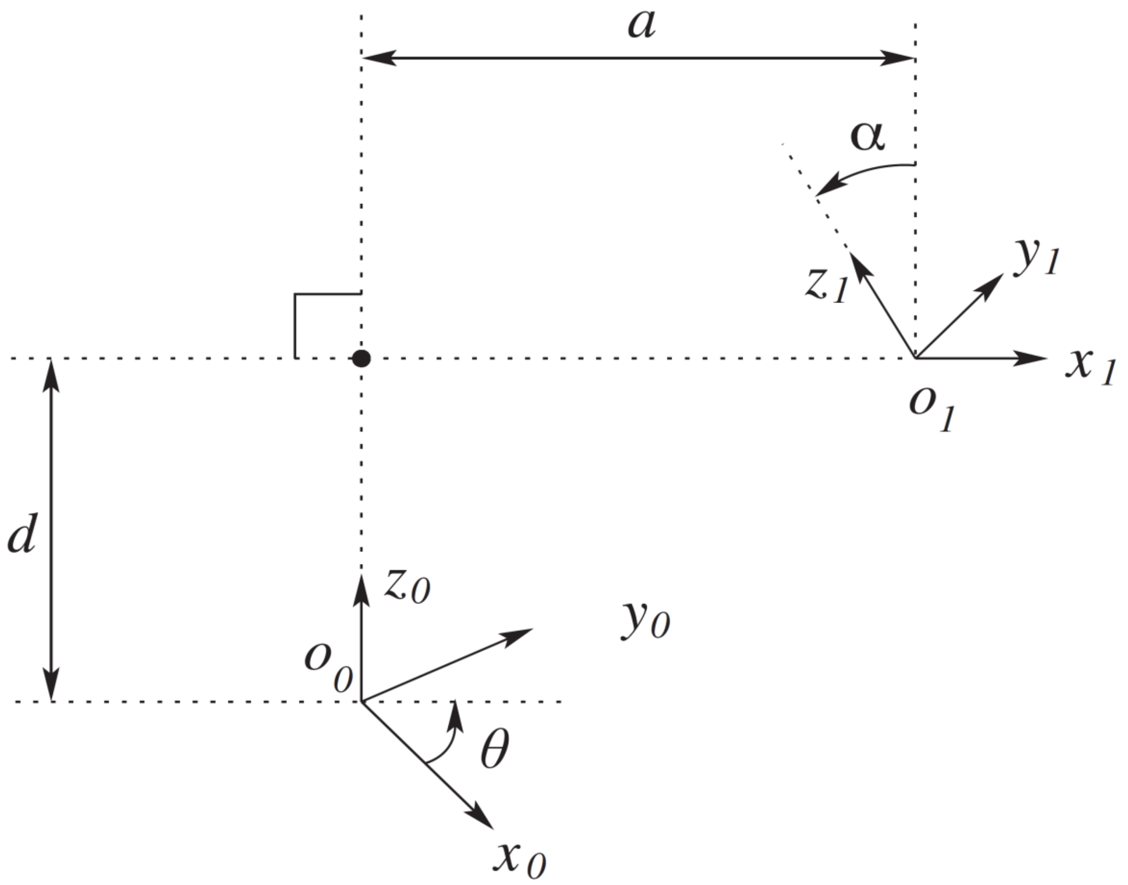
$\boldsymbol{a_i}$	Link Length
$\boldsymbol{\alpha_i}$	Link Twist
$\boldsymbol{d_i}$	Link Offset
$\boldsymbol{\theta_i}$	Joint Angle

Since the matrix A_i is a function of a single variable, three of the above four quantities are constant for a given link, while the fourth parameter, $\boldsymbol{\theta_i}$ for a revolute joint and $\boldsymbol{d_i}$ for a prismatic joint, is the joint variable.

The Denavit-Hartenberg Convention

Existence and Uniqueness

There exists a unique homogeneous transformation matrix A that takes the coordinates from frame *1* into those of frame *0*.



$\boldsymbol{a_i}$	Link Length
$\boldsymbol{\alpha_i}$	Link Twist
$\boldsymbol{d_i}$	Link Offset
$\boldsymbol{\theta_i}$	Joint Angle

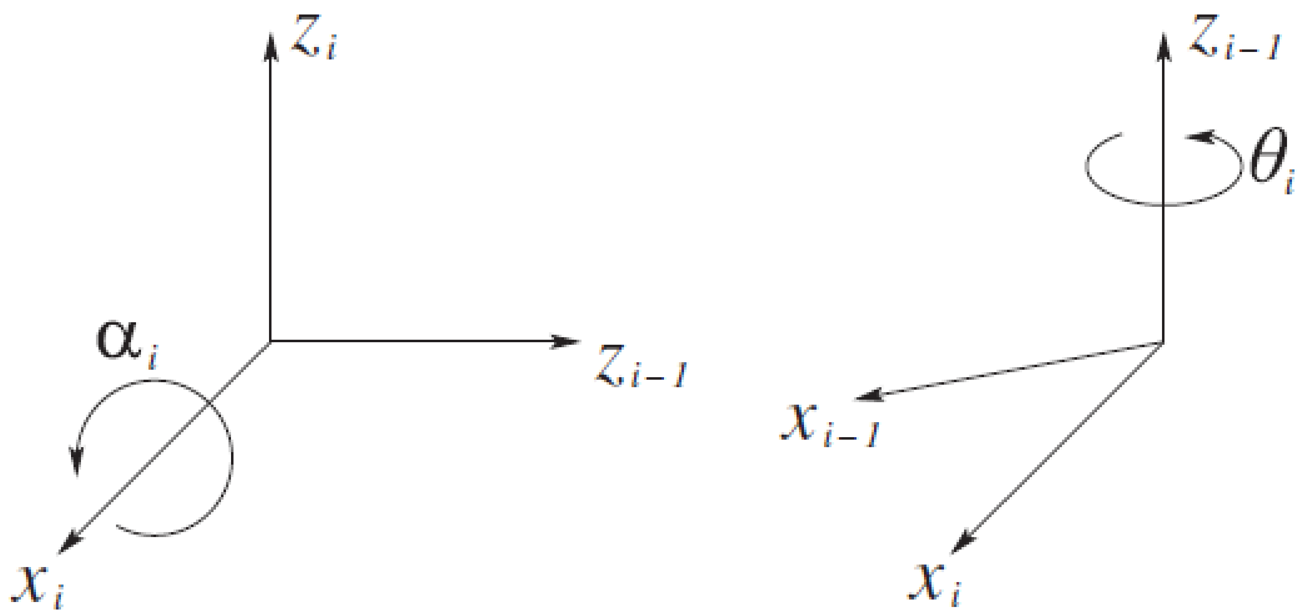
(DH1) The axis x_1 is perpendicular to the axis z_0

(DH2) The axis x_1 intersects the axis z_0.

The Denavit-Hartenberg Convention

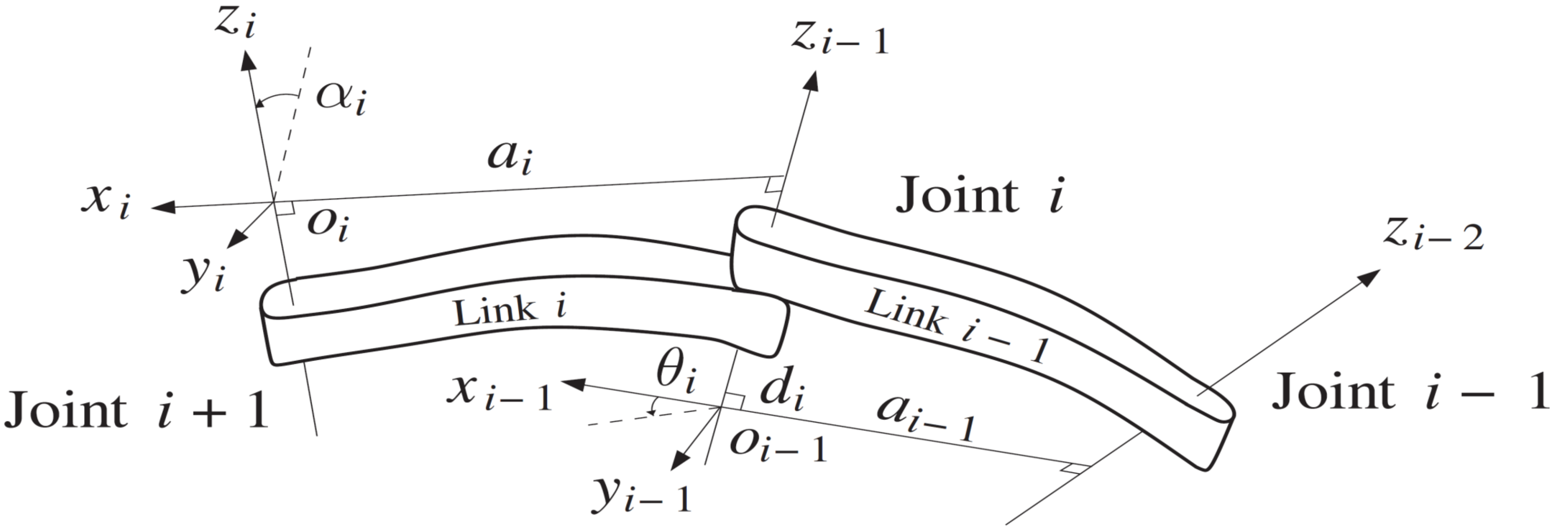
Positive sense for α_i and θ_i

- α is the angle from z_0 to z_1 measured in a plane normal to x_1 by the right-hand rule.
- θ is the angle from x_0 to x_1 measured in a plane normal to z_0.



The Denavit-Hartenberg Convention

Assigning the Coordinate Frames



- While frame ***i*** is required to be rigidly attached to link ***i***, we have considerable freedom in choosing the origin and the coordinate axes of the frame.
- For example, it is not necessary that the origin o_i of frame ***i*** be placed at the physical end of link ***i***.
- In fact, it is not even necessary that frame ***i*** be placed within the physical link.
- Frame ***i*** could lie in free space so long as frame ***i*** is rigidly attached to link ***i***.

The Denavit-Hartenberg Convention

DH Procedure

Step 1: Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.

For $i = 1, \ldots, n-1$ perform Steps 3 to 5.

Step 3: Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish y_i to complete a right-handed frame.

Step 6: Establish the end-effector frame $o_n x_n y_n z_n$. Assuming the n^{th} joint is revolute, set $z_n = a$ parallel to z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the center of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as $s \times a$. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

The Denavit-Hartenberg Convention

DH Procedure

Step 7: Create a table of DH parameters a_i, d_i, α_i, θ_i.

a_i = distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i.

d_i = distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. If joint i is prismatic, d_i is variable.

α_i = the angle from z_{i-1} to z_i measured about x_i.

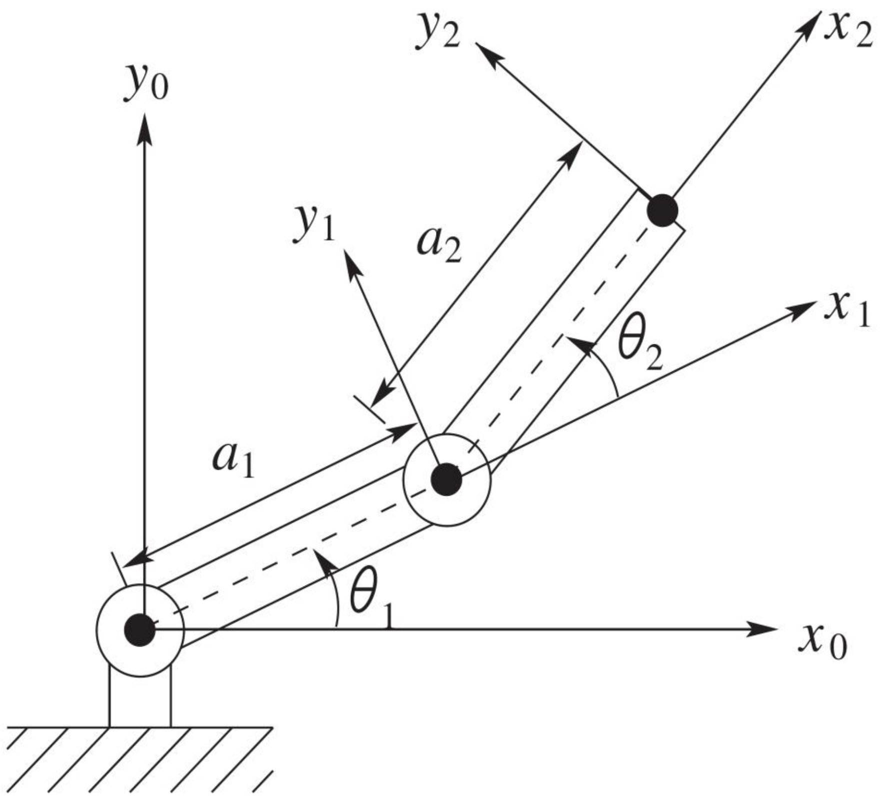
θ_i = the angle from x_{i-1} to x_i measured about z_{i-1}. If joint i is revolute, θ_i is variable.

Step 8: Form the homogeneous transformation matrices A_i by substituting the above parameters into Equation (3.10).

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

Examples

Planar Elbow Manipulator



Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

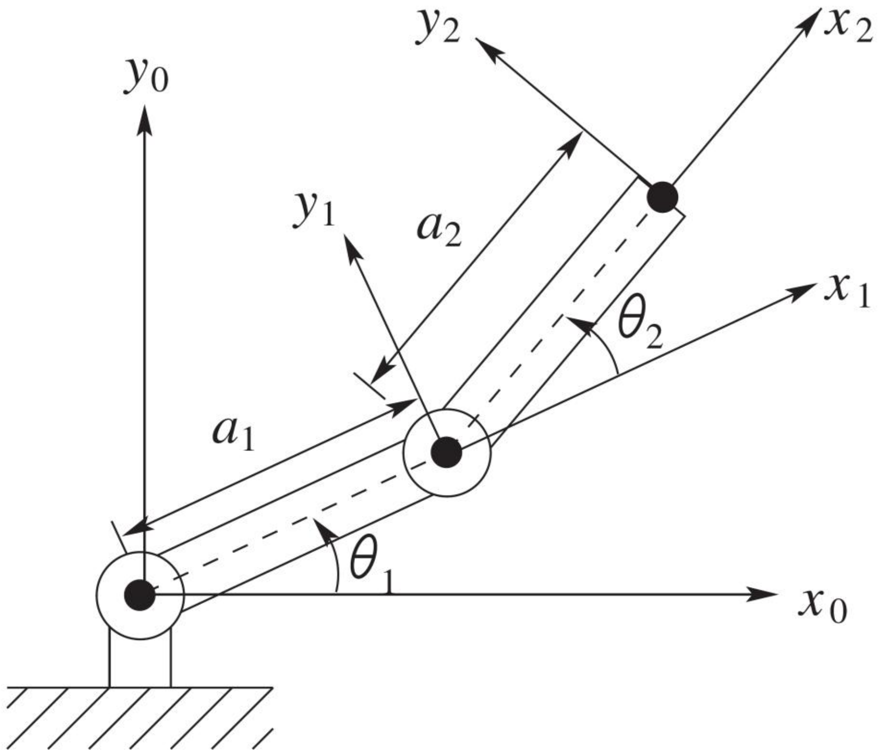
$$T_1^0 = A_1$$

$$T_2^0 = A_1A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The z-axes all point out of the page.
- Note that the direction of the x_0 is arbitrary.

Examples

Planar Elbow Manipulator



$$T_2^0 = A_1 A_2 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1 c_1 + a_2 c_{12} \\ s_{12} & c_{12} & 0 & a_1 s_1 + a_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

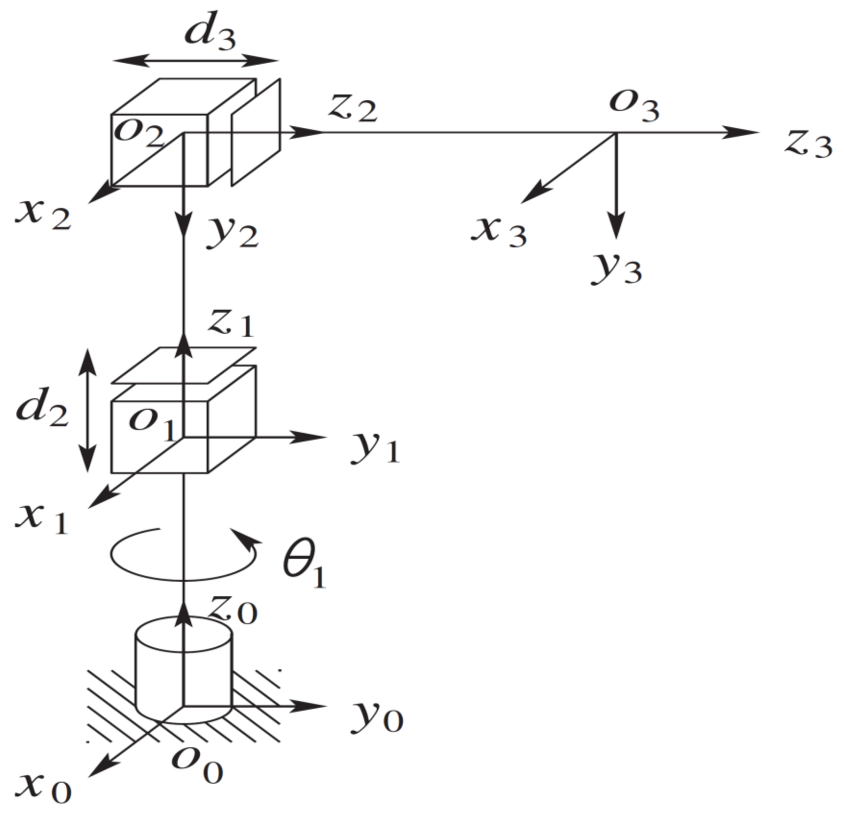
$$x = a_1 c_1 + a_2 c_{12}$$

$$y = a_1 s_1 + a_2 s_{12}$$

- Notice that the first two entries of the last column of T_2^0 are the $\boldsymbol{x}$ and $\boldsymbol{y}$ components of the origin $\boldsymbol{o_2}$ in the base frame; that is, are the coordinates of the end effector in the base frame.
- The rotational part of T_2^0 gives the orientation of the frame $\boldsymbol{o_2 x_2 y_2 z_2}$ relative to the base frame.

Examples

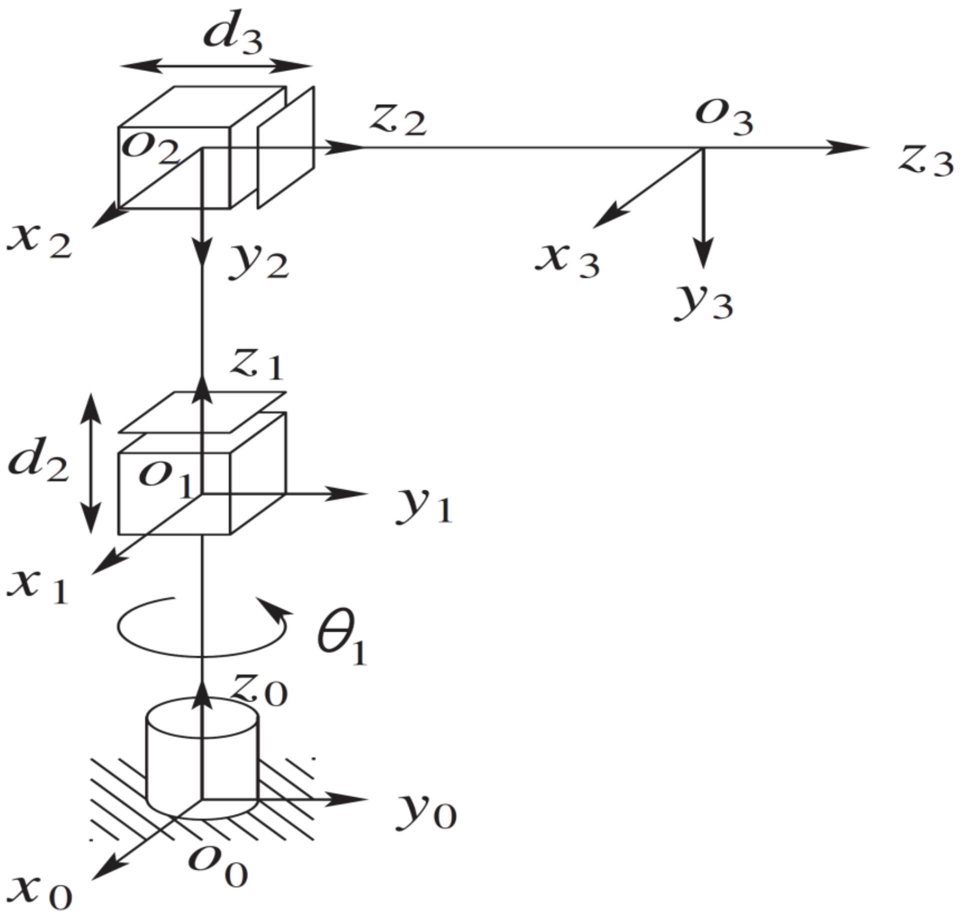
Three-Link Cylindrical Manipulator



- Note that the placement of the origin $\boldsymbol{o_0}$ along $\boldsymbol{z_0}$ and the direction of the $\boldsymbol{x_0}$ axis are arbitrary.
- $\boldsymbol{o_0}$ could just as well be placed at joint *2*.
- Since $\boldsymbol{z_0}$ and $\boldsymbol{z_1}$ coincide, the origin $\boldsymbol{o_1}$ is chosen at joint *2*.
- The $\boldsymbol{x_1}$ axis is parallel to $\boldsymbol{x_0}$ when $\boldsymbol{\theta_1 = 0}$ but, of course its direction will change since $\boldsymbol{\theta_1}$ is variable.
- Since $\boldsymbol{z_2}$ and $\boldsymbol{z_1}$ intersect, the origin $\boldsymbol{o_2}$ is placed at this intersection.
- The direction of $\boldsymbol{x_2}$ is chosen parallel to $\boldsymbol{x_1}$ so that $\boldsymbol{\theta_2}$ is zero.
- Finally, the third frame is chosen at the end of link *3*.

Examples

Three-Link Cylindrical Manipulator



Link	a_i	α_i	d_i	θ_i
1	0	0	d_1	θ_1
2	0	-90	d_2	0
3	0	0	d_3	0

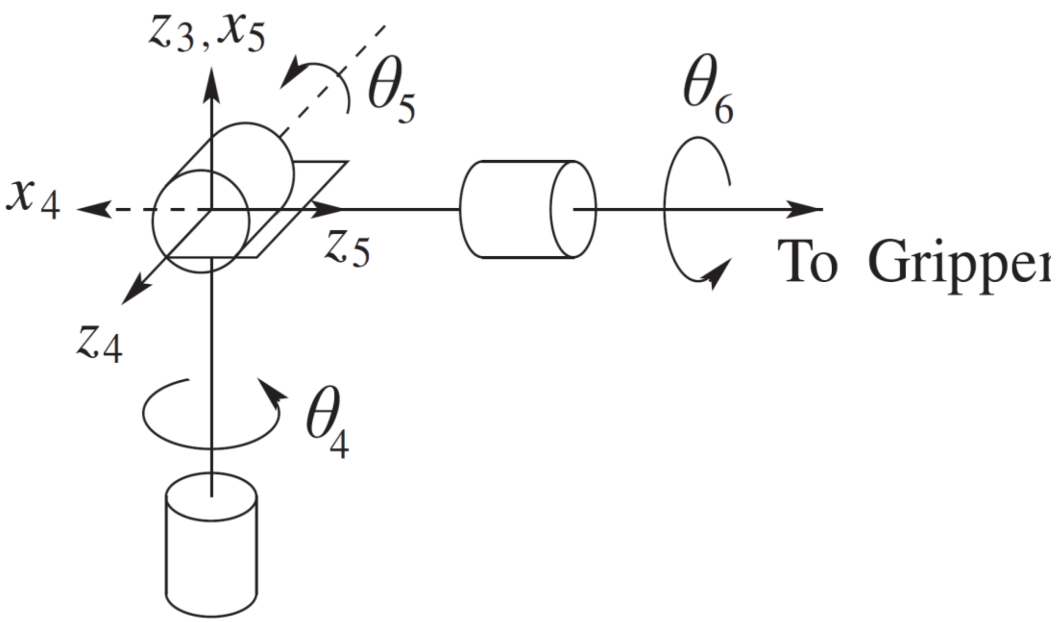
$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = A_1 A_2 A_3 = \begin{bmatrix} c_1 & 0 & -s_1 & -s_1 d_3 \\ s_1 & 0 & c_1 & c_1 d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Examples

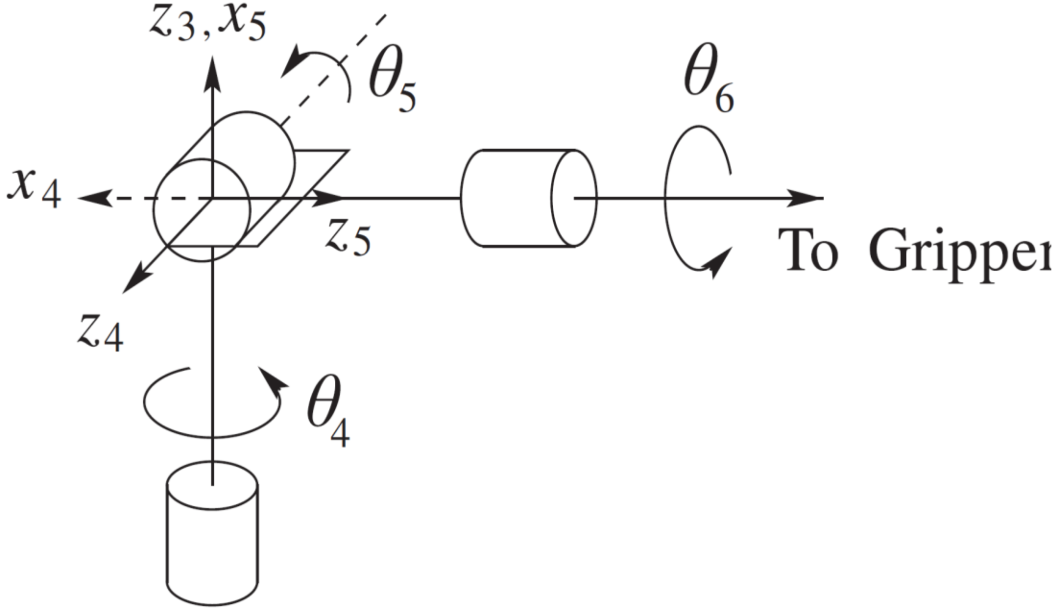
The Spherical Wrist



- A three-link wrist mechanism for which the joint axes z_3, z_4, z_5 intersect at o.
- The point o is called the **wrist center**.
- θ_4; θ_5; θ_6 can be identified as the Euler angles φ, θ and ψ with respect to the coordinate frame $o_3x_3y_3z_3$.

Examples

The Spherical Wrist



Link	a_i	α_i	d_i	θ_i
4	0	-90	0	θ_4
5	0	90	0	θ_5
6	0	0	d_6	θ_6

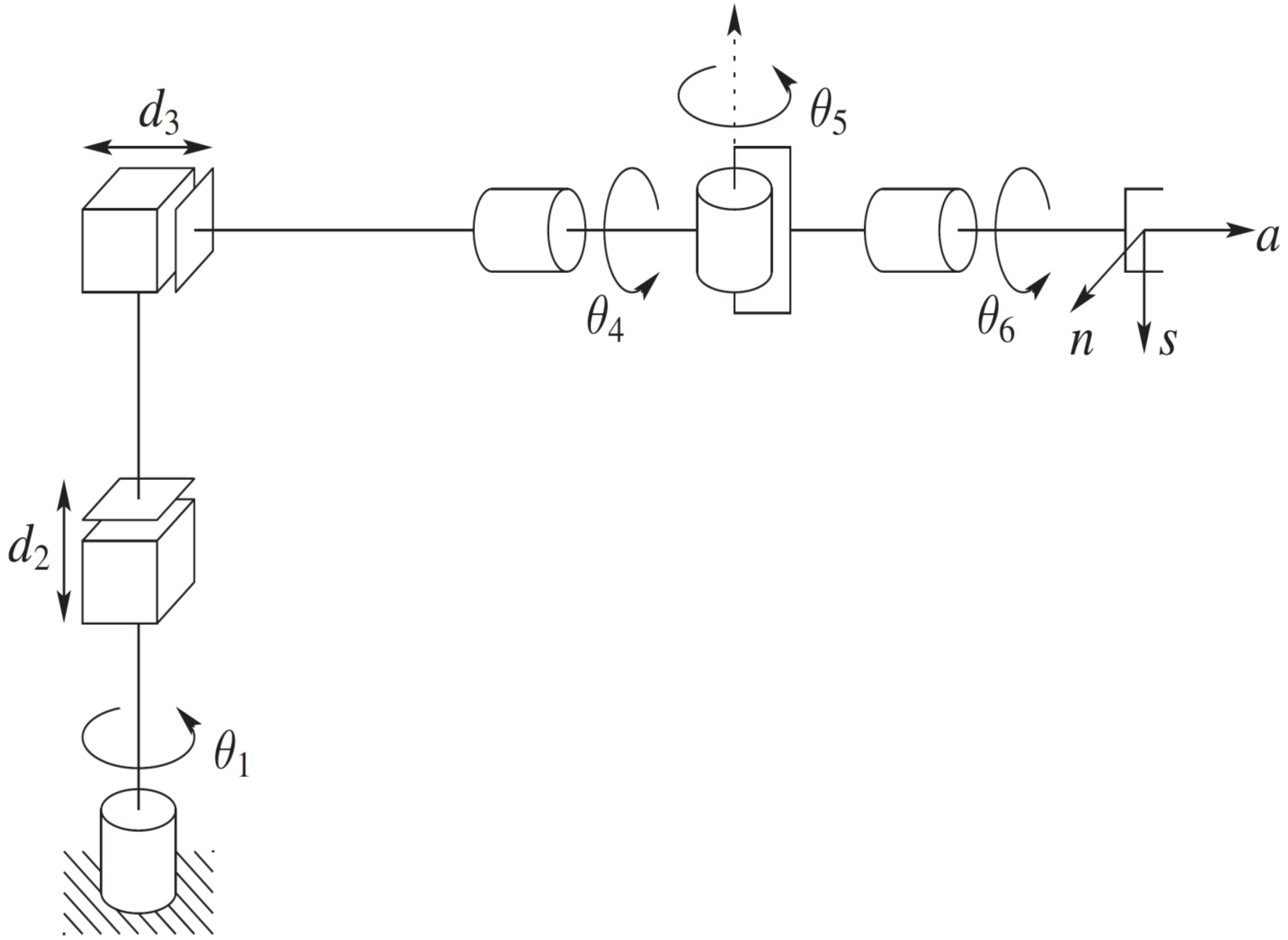
$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} T_6^3 &= A_4 A_5 A_6 \\ &= \begin{bmatrix} R_6^3 & o_6^3 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & c_4s_5d_6 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & s_4s_5d_6 \\ -s_5c_6 & s_5s_6 & c_5 & c_5d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Examples

Cylindrical Manipulator with Spherical Wrist



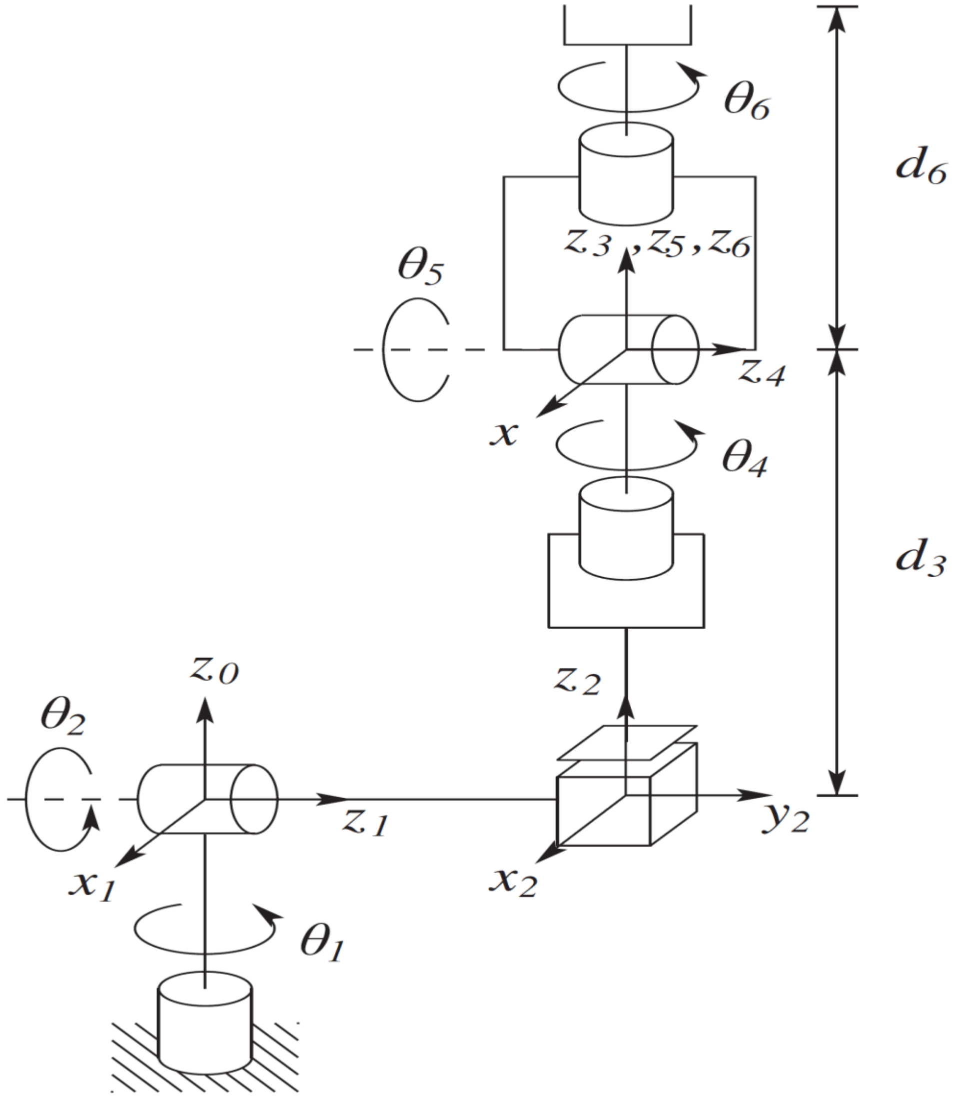
$$T_6^0 = T_3^0 T_6^3$$

From Example 2

From Example 3

Examples

Stanford Manipulator



Link	d_i	a_i	α_i	θ_i
1	0	0	−90	θ_1
2	d_2	0	+90	θ_2
3	d_3	0	0	0
4	0	0	−90	θ_4
5	0	0	+90	θ_5
6	d_6	0	0	θ_6

$$T_6^0 = A_1 \cdots A_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

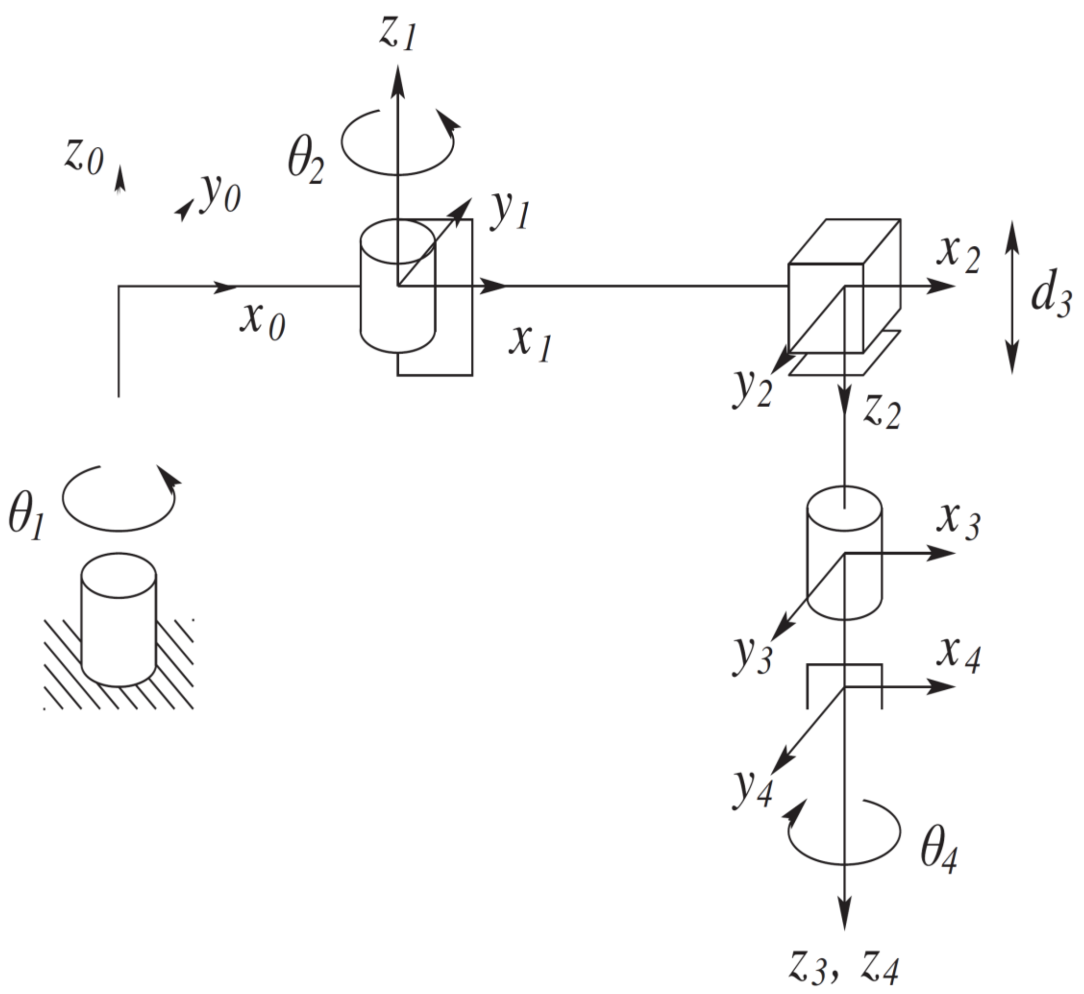
$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Examples

SCARA Manipulator



Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	180	0	θ_2
3	0	0	d_3	0
4	0	0	d_4	θ_4

$$T_4^0 = A_1 \cdots A_4$$

$$= \begin{bmatrix} c_{12}c_4 + s_{12}s_4 & -c_{12}s_4 + s_{12}c_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{12}c_4 - c_{12}s_4 & -s_{12}s_4 - c_{12}c_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} c_2 & s_2 & 0 & a_2c_2 \\ s_2 & -c_2 & 0 & a_2s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MATLAB: Robotics System Toolbox

Forward Kinematics

First Method

```
theta=deg2rad(60); d=0;
alpha=deg2rad(0); a=1;
Rot_z=axang2tform([0 0 1 theta]);
Trans_z=trvec2tform([0 0 d]);
Rot_x=axang2tform([1 0 0 alpha]);
Trans_x=trvec2tform([a 0 0]);
A=Rot_z*Trans_z*Trans_x*Rot_x;
```

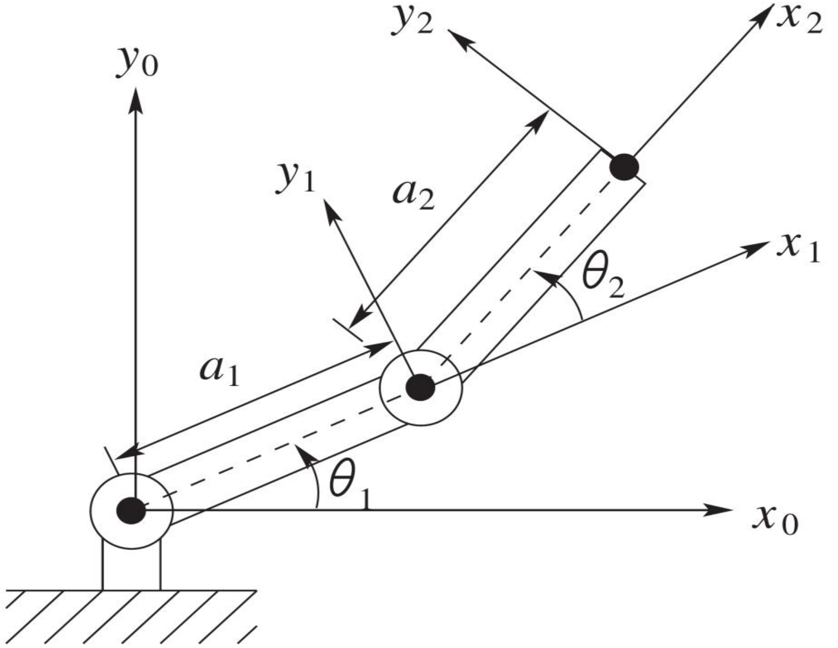
$$
\begin{aligned}
A_i &= \text{Rot}_{z,\theta_i}\text{Trans}_{z,d_i}\text{Trans}_{x,a_i}\text{Rot}_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

MATLAB: Robotics System Toolbox

Forward Kinematics

Example

```
theta1=deg2rad(60); d1=0;
alpha1=deg2rad(0); a1=1;
Rot_z1=axang2tform([0 0 1 theta1]);
Trans_z1=trvec2tform([0 0 d1]);
Rot_x1=axang2tform([1 0 0 alpha1]);
Trans_x1=trvec2tform([a1 0 0]);
A1=Rot_z1*Trans_z1*Trans_x1*Rot_x1;
```



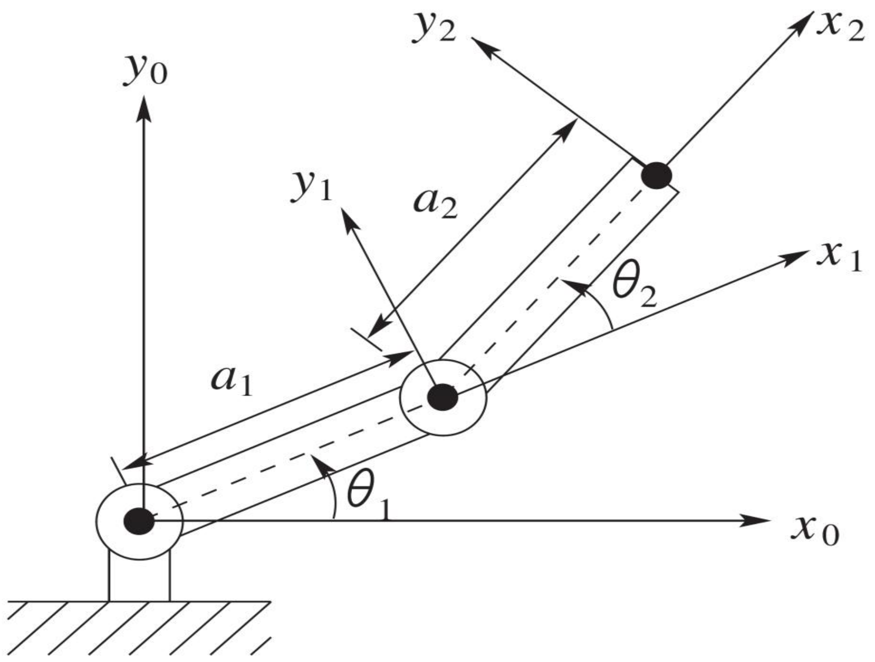
Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

MATLAB: Robotics System Toolbox

Forward Kinematics

Example

```
theta2=deg2rad(30); d2=0;

alpha2=deg2rad(0); a2=0.5;

Rot_z2=axang2tform([0 0 1 theta2]);

Trans_z2=trvec2tform([0 0 d2]);

Rot_x2=axang2tform([1 0 0 alpha2]);

Trans_x2=trvec2tform([a2 0 0]);

A2=Rot_z2*Trans_z2*Trans_x2*Rot_x2;

T=A1*A2;
```



Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

MATLAB: Robotics System Toolbox

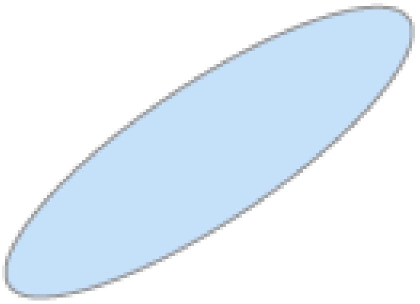
Rigid Body Tree Robot Model

The rigid body tree model is a representation of a robot structure. You can use it to represent robots such as manipulators or other kinematic trees. Use rigidBodyTree objects to create these models.

Rigid Body Tree Components

Base: Every rigid body tree has a base. The base defines the world coordinate frame and is the first attachment point for a rigid body.

Rigid Body: The rigid body is the basic building block of rigid body tree model and is created using rigidBody. A rigid body, sometimes called a link, represents a solid body that cannot deform. The distance between any two points on a single rigid body remains constant. Each rigid body has a coordinate frame associated with them and contains a rigidBodyJoint object.

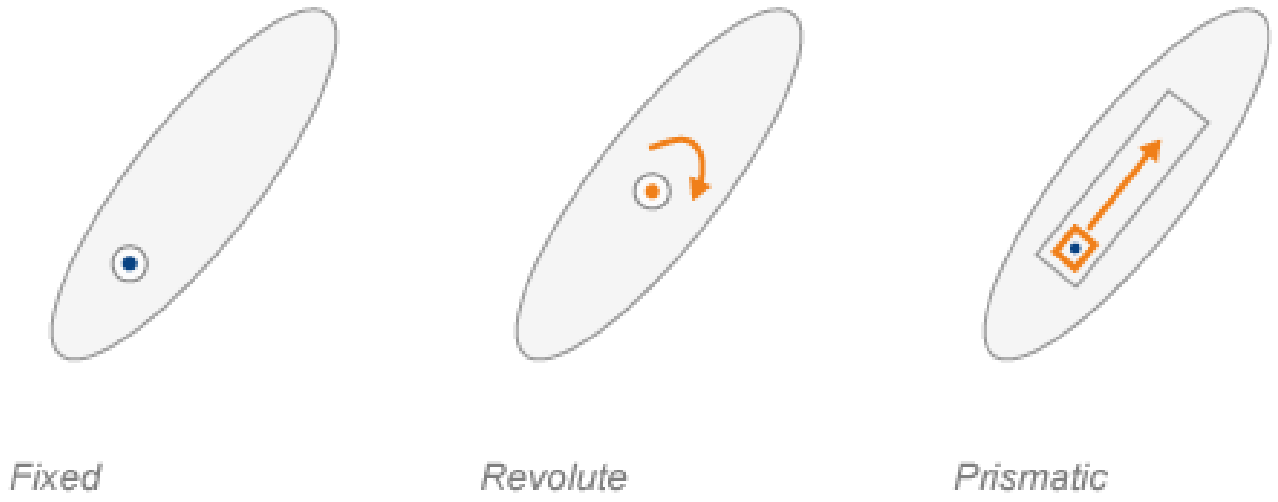


MATLAB: Robotics System Toolbox

Rigid Body Tree Robot Model

Joint: Each rigid body has one joint, which defines the motion of that rigid body relative to its parent. It is the attachment point that connects two rigid bodies in a robot model.

The rigidBodyJoint object supports fixed, revolute, and prismatic joints.



MATLAB: Robotics System Toolbox

Rigid Body Tree Robot Model

setFixedTransform

Set fixed transform properties of joint

Syntax

setFixedTransform(jointObj,tform)
setFixedTransform(jointObj,dhparams,"dh")

- DH parameters are given in the order [a alpha d theta].
- For revolute joints, the theta input is ignored when specifying the fixed transformation between joints because that angle is dependent on the joint configuration. For prismatic joints, the d input is ignored

MATLAB: Robotics System Toolbox

Rigid Body Tree Robot Model

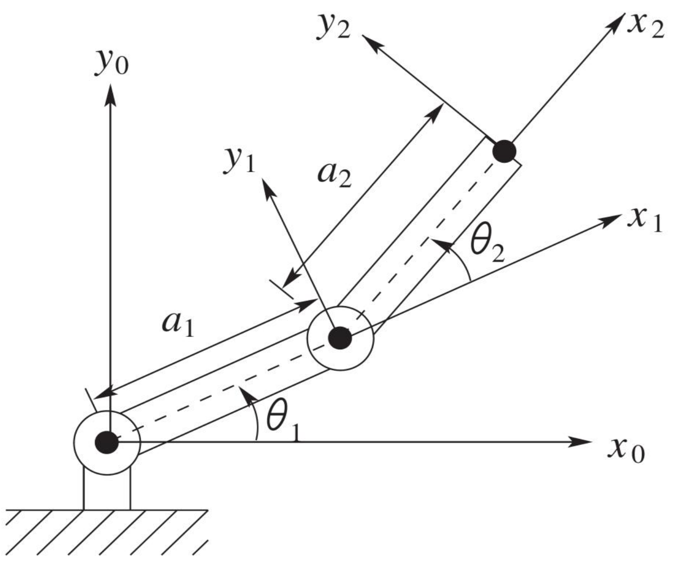
importrobot	Import rigid body tree model from URDF file, text, or Simscape Multibody model
loadrobot	Load rigid body tree robot model
rigidBodyTree	Create tree-structured robot
rigidBody	Create a rigid body
rigidBodyJoint	Create a joint
interactiveRigidBodyTree	Interact with rigid body tree robot models

MATLAB: Robotics System Toolbox

Forward Kinematics

Second Method

```
robot = rigidBodyTree;
body1 = rigidBody('body1');
body2 = rigidBody('body2');
bodyEndEffector = rigidBody('endeffector');
jnt1 = rigidBodyJoint('jnt1','revolute');
jnt2 = rigidBodyJoint('jnt2','revolute');
jnt1.HomePosition = deg2rad(0);
jnt2.HomePosition = deg2rad(0);
dh1=[1 0 0 deg2rad(60)];
dh2=[0.5 0 0 deg2rad(30)];
setFixedTransform(jnt1,dh1,'dh');
setFixedTransform(jnt2,dh2,'dh');
body1.Joint = jnt1;
body2.Joint = jnt2;
```



Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

MATLAB: Robotics System Toolbox

Forward Kinematics

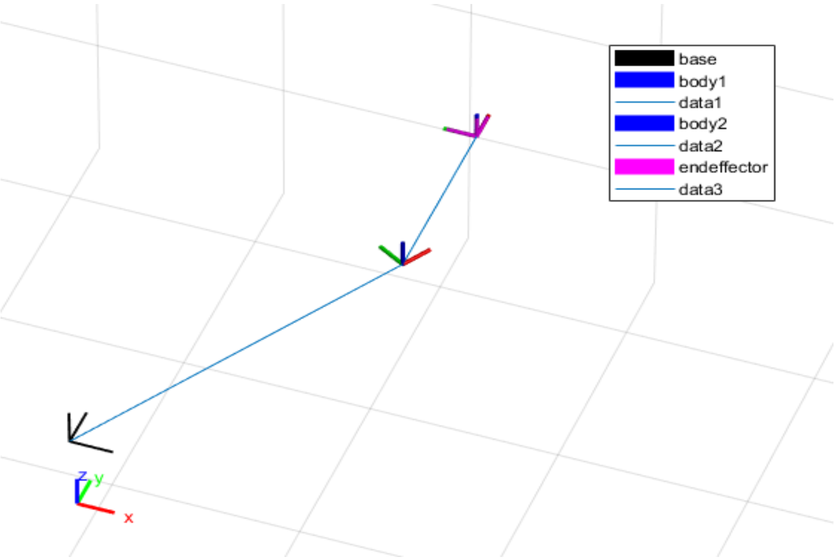
Second Method

```
addBody(robot,body1,'base');
addBody(robot,body2,'body1');
addBody(robot,bodyEndEffector,'body2');
config = homeConfiguration(robot)
config(1).JointPosition = deg2rad(60)
config(2).JointPosition = deg2rad(30)
T = getTransform(robot,config,'endeffector','base')
showdetails(robot)
show(robot,config)
```

Result

T =

0.0000	-1.0000	0	0.5000
1.0000	0.0000	0	1.3660
0	0	1.0000	0
0	0	0	1.0000



Thank You

Any Questions ??