ME246 – INTRODUCTION TO ROBOTICS

TERM PROJECT

ANALYSIS & SIMULATION OF 6DOF AND 5DOF SERIAL MANIPULATORS FOR TRACING TRAJECTORIES GENERATED IN DIFFERENT WAYS USING MATLAB

Submitted by

SANKAR.B

(S.R. No. 18321)

RESEARCH SCHOLAR

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF SCIENCE, BANGALORE

भारतीय विज्ञान संस्थान

ABSTRACT

Mankind has always strived to give life-like qualities to its artifacts in an attempt to find the substitutes for himself to carry out his orders and also to work in a hostile environment. The popular concept of a robot is a machine that looks and works like human being. The industrial robots of today may not look like human being although all the research is directed to provide more and more anthropomorphic and human like features and superman capabilities in these.

The trajectory planning has a great importance in theAutomation Industries. D-H transformation, Iterative Newton-Euler dynamic formulation, Lagrangian formulation of manipulator dynamics, Trajectory Planning and Control Logics are thebuilding blocks of robot hand action reference to base. Among different Trajectory planning techniques,3 different ways are introduced and their simulation was done.

TABLE OF CONTENTS

1. INTRODUCTION

The tasks that a robot manipulator can perform will also vary greatly with the particular design. Although we have generally dealt with the robot manipulator as an abstract entity, its performance is ultimately limited by pragmatic factors such as load capacity, speed, size of workspace, and repeatability. For certain applications, the overall manipulator size, Weight, power consumption, and cost will be significant factors.

Robots are programmable physical machines that have sensors and actuators, and are given goals for what they should achieve in the world. Perception algorithms process the sensor inputs, a control program decides how the robot should behave given its goals and current circumstances, and commands are sent to the motors to make the robot act in the world. Some robots are mobile, but others are rooted to a fixed location.

“An industrial robot is an automatic, servo-controlled, freely programmable, multipurpose manipulator, with several areas, for the handling of the work pieces, tools or special devices. Variably programmed operations make the execution of a multiplicity of tasks possible.”

Robotics is an art, knowledge base, and the know how of designing, applying, and using robot in human endeavors. Robotics system consists of not just robots, but also other devices and systems that used together with the robot to perform the necessary task. It can be used in manufacturing environments, in underwater and space exploration, for aiding the disabled, or even for fun.

1.1 CLASSIFICATION OF ROBOTS

According to Japanese Industrial Robot Association (JIRA), Robots are following types:

Class 1: *Manual Handling Device:* a device with multiple degrees of freedom is actuated by an operator.

Class 2: *Fixed sequence robot:* A Device that performs the successive stages of task according to a predetermined, unchanging method and is hard to modify.

Class 3: *Variable sequence robot:* Same as Class 2 but easy to modify.

Class 4: *Playback Robot:* A human operator performs the task manually by leading the robot, which record the motions for later play back. The robot repeats the same motion according to recorded information.

Class 5: *Numerical Control Robot:* The operator supplies the robot with a movement of program rather than teaching it the task manually.

Class 6: *Intelligent Robot:* A robot with the means to understand its environment and the abilityto successfully complete a task despite change in the surrounding conditions performed.

1.2 PHYSICAL PARTS OF AN INDUSTRIAL ROBOT

Following are the main components of an Industrial Robot Manipulator:

(i) Mechanical part or manipulator (Body, Arm, Wrist)
(ii) End effector (Tool or Gripper)
(iii) Actuators
(iv) Controller (Sensors, Processor)
(v) Software

Mechanical part or manipulator: This is the main body of the robot and consists of the links, the joints, and other structural elements of the robot.

End effector**:* This is the part that is connected to the last joint of a manipulator which generally handles objects, makes connection to other machines, or performs the required task. ***Actuators: These are the “muscle” of the manipulators. Common types of actuators are servo motors, stepper motors, pneumatic cylinders and hydraulic cylinders.

***Controller (Sensors, Processor)**:* It is rather similar to cerebellum, and although it does not have the power of our brain, it still controls motions. The controller receives its data from the computer, controls the motion of the actuators, and coordinates the motion with sensory feedback information. *Sensors* are used to collect the information about the internal state of the robot or to communicate with the outside environment. Robot controller needs to know where each link of the robot is in order to know the robot configuration. *Processor* is the brain of robot. It calculates the motions of the robot joint, determines how much and how fast each joint move to achieve desired location and speeds, and oversees the coordinate actions of the controller and processor. Processor is generally a microcomputer dedicated to a single purpose.

Software: There are perhaps three groups of software used in a robot. One is a operating system, which operates the computers. The second is the Robotic Software which calculates the necessary motions of each joint based on the kinematic equation of the robot. Third group is collection of routines and application programs that developed in order to use the peripheral device of the robot, such as vision routines, or to perform specific tasks.

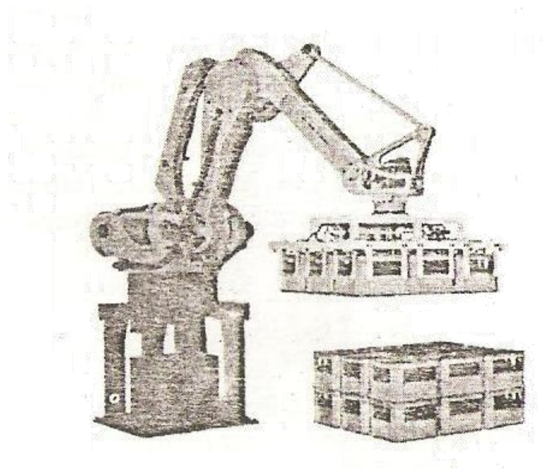


Figure 1.1 A Faunc M-410iWW palletizing robotic manipulator with it's end effector.

1.3 ROBOT ANATOMY

The robot anatomy is, therefore, the study of skeleton robot, that is, the physical construction of the manipulator structure. The mechanical structure of a manipulator that consist of *links* connected by means of *joints* is segmented into an *arm* that ensure the mobility and reach-ability, a wrist that confers the orientation, and the *end-effector* that perform the required task.

Link: The mechanical structure of a robotic manipulator is a mechanism, whose members are rigid *links*. A rigid link can connected with at most, with two other links is referred as binary link. Two links are connected by a joint.

Joint: Many kinds of joint can be made between two links. However, only two basic types are commonly used in industrial robots. These are:

- Revolute (***R***) joints and
- Prismatic (***P***) joints.

In case of revolute joints, two links are joined by a pivot about the axis of which the link can rotate w.r.t. each other. In case of prismatic joints, two links are so joined that these can slide(linearly move) w.r.t. each other. Other types of joint are *planner*, *cylindrical*, *twist*, *screw* and *spherical*.

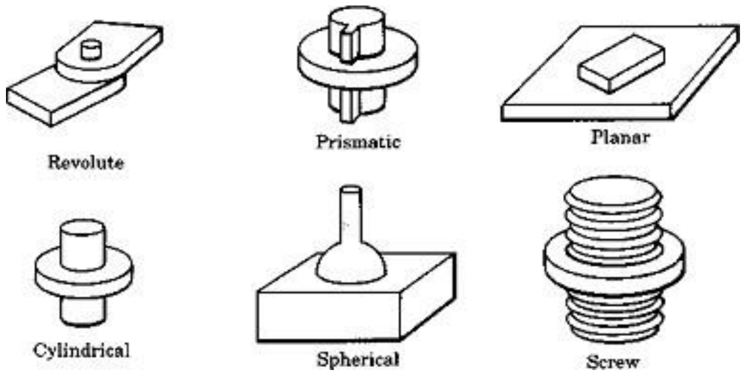


Figure 1.2 Different types of joints

At joint, links are connected such that they can be made to move relative to each other by an actuator. To form a manipulator, one(first) end of chain is connected to base and other(last) is to end-effector. The kinematic chain of manipulator is characterised by *degree of freedom* (DOF) it has, and the space its end-effector can sweep.

Degrees of freedom (DOF): The number of independent movements that an object can perform in a 3-D space is called number of *degree of freedom* (DOF). Thus, a rigid body free in space has three degree of freedom; three for *position* and three for *orientation*. DOF is always less tan equal to the number of joints. A manipulator with 6-DOF can point any position and orientation. But there are situations where five or less degree of freedom robots are enough for the required job. There are four basic coordinate configuration of work space as per the joints to point a position:

(a) Cartesian (rectangular) configuration- all three ***P*** joints.

(b) Cylindrical configuration- one ***R*** and two ***P*** joints.

(c) Polar (spherical) configuration- two ***R*** and one ***P*** joints.

(d) Articulate (Revolute or Jointed) configuration- all three ***R*** joints.

***End effector**:* This is the part that is connected to the last joint of a manipulator which generally handles objects, makes connection to other machines, or performs the required task. These can be grouped into two major categories: 1. Grippers and 2. Tools. Grippers are to grasp or hold the work piece during the work cycle. The proper shape and size of the grippers and method of holding are determined by the object to grasp and task to perform. Grippers employ mechanical grasping or other alternative way like magnetic, vacuum, bellows or other for holding. For many task to be performed by manipulator, the end-effector is *tool* rather than gripper. For examples, a cutting tool, a drill, a welding torch, a spray gun or a screw driver is end-effector for machine. *Tool changer* device can be attached to perform multiple tasks in work cycle.

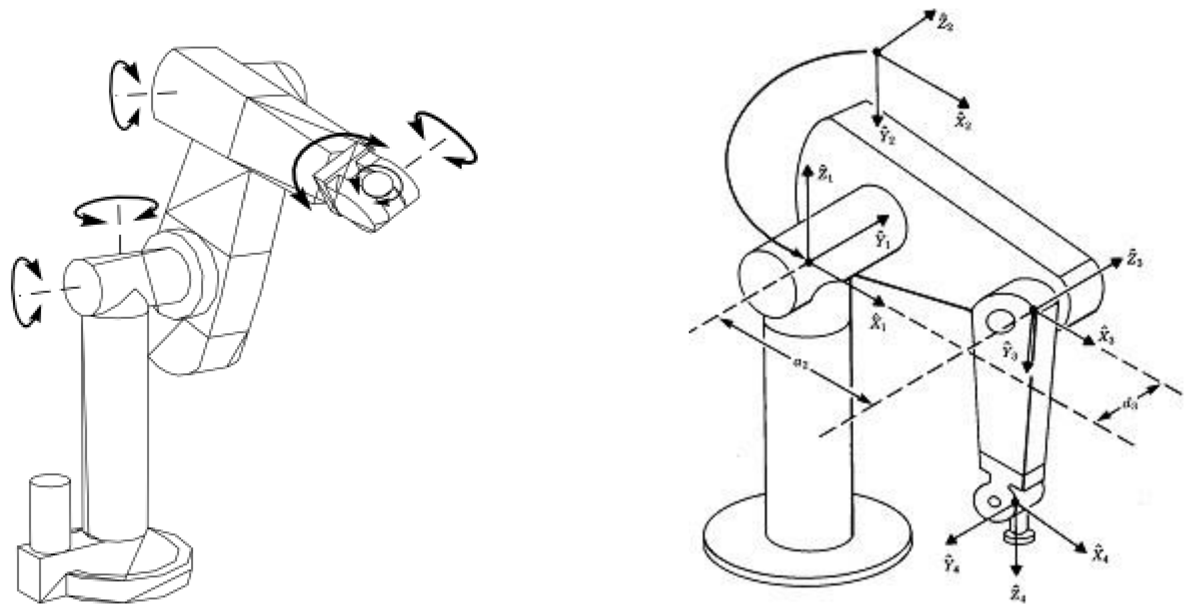


Figure 1.3 “PUMA” serial robot with axes of rotations(left) and Some kinematic parameters and frame assignment for PUMA-560 manipulator(right).

1.4 MANIPULATION AND CONTROL

The basic problem in the study of mechanical manipulation is of computing the position and orientation of the end-effector when the joint angles are known. This is referred as *forward kinematics* problem. The *inverse kinematic* problem is to determine the joint angle, given the position and orientation of the end-effector. In inverse kinematic, the solution for joint angles may mot be unique; there may be multiple solutions. Thus, inverse kinematics is to calculate all possible set of joint angles which can be used to attain a given position and orientation of the end-effector. Another problem is to find the end-effector velocity for given joints velocities and vice-versa. These two problems, direct and inverse need the manipulator *Jacobian* (matrix), which is obtained from the kinematic parameters. To perform an assign task or attain a desire position, a manipulator is required to accelerate from rest, move at specified velocity, traverse a specified path, and finally decelerate to stop. To accomplish this, the trajectory to be followed is computed. To traverse this trajectory, controlling torques are applied by actuators at the joints. These torques are computed from the equation of motion of manipulator, which describe the *dynamics* of the manipulator. The motion of end-effector and each joint must be smooth and controlled. Often this path is described by a number of intermediate locations, in addition to the desired destination. The goal of *trajectory planning* is to generate time law for the manipulator variables for a given description of joint or end-effector motion. Manipulator motion control and its force interaction with environment are monitored by control algorithm. The tasks to be performed by manipulator are: (*i*) to move the end-effector along a desired trajectory, and (*ii*) to exert a force on the environment to carry out the desired task. Controller has to control both the tasks, the former is called *position control* (or *trajectory control*) and latter is called *force control*. Here *Sensors* are to determine actual values of parameters of interest. Robotic vision provides the capability of viewing the workspace and interpreting what is seen. Vision equipped robots are used for *inspection, part recognition, identification*, *sorting, obstacle avoidance,* and other similar tasks.

2. CO-ORDINATE FRAMES, MAPPING AND TRANSFORMS

The robot manipulator consists of several rigid links, connected together by joints, to achieve the required motion in space and perform the desired task. The modeling of robot comprises of establishing a special relationship between the manipulator and manipulated object. The position of link in space and their motion are described by a spatial geometry.

A systematic and generalized approach for mathematical modeling of position and orientation of link in space w.r.t. a reference frame is carried out with the help of vector and matrix algebra. Because the motion of each link can be described w.r.t. a reference coordinate frame, it is convenient to have a coordinate frame attached to the body of each link.

2.1 COORDINATE FRAME

In a 3-D space, coordinate frame is set of three orthogonal right-handed axes *X, Y, Z,* called principal axes. Such frame is shown in Fig. 2.1 with the origin of principal axes at '*O*' along with three unit vectors $\hat{x}\hat{y}\hat{z}$, along these axes. This frame is labeled.

Any point *P* in 3-D space can be defined w.r.t. this coordinate frame by vector $\vec{OP}$. in vector notation

$$\vec{P} = \vec{OP} = p_x\,\hat{x} + p_y\,\hat{y} + p_z\,\hat{z}$$

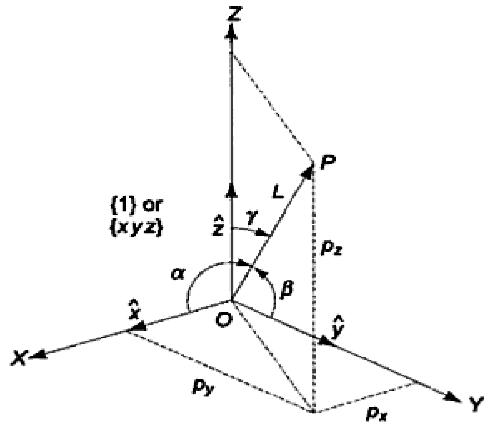


Figure 2.1 Position and Orientation of a point *P* in a coordinate frame

where p_x, p_y, p_z are the components of the vector $\vec{OP}$ along the three coordinate axes or projection of vector $\vec{OP}$ on the axes *X, Y, Z,* respectively. A *frame-space* notation is introduced as ${}^1P$ to refer to a point *P* w.r.t. frame {1} with its components in frame as ${}^1p_x$, ${}^1p_y$ and ${}^1p_z$

$$ {}^1\vec{P} = {}^1p_x\,\hat{x} + {}^1p_y\,\hat{y} + {}^1p_z\,\hat{z} $$

In vector matrix notation, this equation can be in terms of vector components only as:

$$ {}^1\boldsymbol{P} = \begin{bmatrix} {}^1p_x \\ {}^1p_y \\ {}^1p_z \end{bmatrix} = \begin{bmatrix} {}^1p_x & {}^1p_y & {}^1p_z \end{bmatrix}^T $$

In addition the direction of the position vector $\vec{OP}$ can be expressed by the direction cosines:

$$ \cos\alpha = \frac{{}^1p_x}{L},\ \cos\beta = \frac{{}^1p_y}{L},\ \cos\gamma = \frac{{}^1p_z}{L} $$

with
$$ L = |\vec{P}| = |\overrightarrow{OP}| = \sqrt{({}^1p_x)^2 + ({}^1p_y)^2 + ({}^1p_z)^2} $$

where α, β, γ are respectively the right handed angles measured from the coordinate axes to the vector $\vec{OP}$, which has length *L*.

2.2 MAPPING BETWEEN ROTATED FRAMES

Consider two frames, frame {1} with axes X,Y,Z and frame {2} with axes U, V, W with a common origin, as shown in Fig. 2.2. A point *P* in space can be described by two frames and expressed as vectors ${}^1P$ and ${}^2P$,

$$ {}^1\vec{P} = {}^1p_x\,\hat{x} + {}^1p_y\,\hat{y} + {}^1p_z\,\hat{z} $$

$$ {}^2\vec{P} = {}^2p_u\,\hat{u} + {}^2p_v\,\hat{v} + {}^2p_w\,\hat{w} $$

The description of point *P* in frame {2} is known and its description in frame {1} is found and vice-versa. This accomplished by projecting the vector ${}^2P$ on the coordinate of frame {1}. Projections of ${}^2P$ on frame {1} are obtained by taking the dot product of ${}^2P$ with the unit vectors of frame {1}. Thus we get the equations:

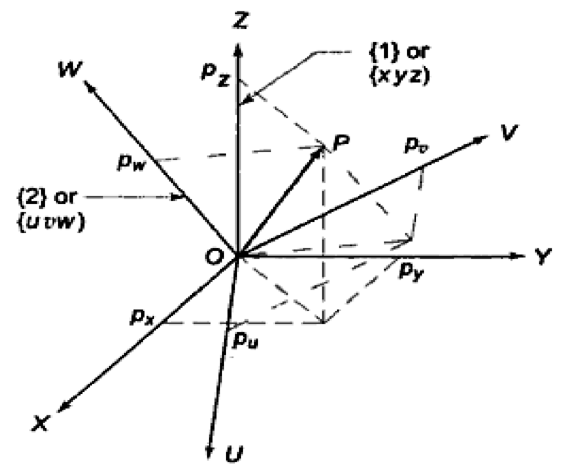


Figure 2.2 Representing a frame in two frames {1} and {2}.

$$^1p_x = \hat{x}\cdot {}^2\vec{P} = \hat{x}\cdot {}^2p_u\,\hat{u} + \hat{x}\,{}^2p_v\,\hat{v} + \hat{x}\cdot {}^2p_w\,\hat{w}$$

$$^1p_y = \hat{y}\cdot {}^2\vec{P} = \hat{y}\cdot {}^2p_u\,\hat{u} + \hat{y}\,{}^2p_v\,\hat{v} + \hat{y}\cdot {}^2p_w\,\hat{w}$$

$$^1p_z = \hat{z}\cdot {}^2\vec{P} = \hat{z}\cdot {}^2p_u\,\hat{u} + \hat{z}\cdot {}^2p_v\,\hat{v} + \hat{z}\cdot {}^2p_w\,\hat{w}$$

This can be written in matrix form as

$$\begin{bmatrix} ^1p_x \\ ^1p_y \\ ^1p_z \end{bmatrix} = \begin{bmatrix} \hat{x}\cdot\hat{u} & \hat{x}\cdot\hat{v} & \hat{x}\cdot\hat{w} \\ \hat{y}\cdot\hat{u} & \hat{y}\cdot\hat{v} & \hat{y}\cdot\hat{w} \\ \hat{z}\cdot\hat{u} & \hat{z}\cdot\hat{v} & \hat{z}\cdot\hat{w} \end{bmatrix} \begin{bmatrix} ^2p_x \\ ^2p_y \\ ^2p_z \end{bmatrix}$$

In compress vector matrix notation above can written as

$$^1P = {}^1R_2\,{}^2P$$

where

$$^1\boldsymbol{R}_2 = \begin{bmatrix} \hat{x}\cdot\hat{u} & \hat{x}\cdot\hat{v} & \hat{x}\cdot\hat{w} \\ \hat{y}\cdot\hat{u} & \hat{y}\cdot\hat{v} & \hat{y}\cdot\hat{w} \\ \hat{z}\cdot\hat{u} & \hat{z}\cdot\hat{v} & \hat{z}\cdot\hat{w} \end{bmatrix}$$

Here ***R*** matrix is called as rotation matrix as frame {1} and {2} have same origin, they are only rotated. Similarly we can find $^2P = {}^2R_1\ {}^1P$ where $^2R_1$ can be written as:

$$^2\boldsymbol{R}_1 = \begin{bmatrix} \hat{u}\cdot\hat{x} & \hat{u}\cdot\hat{y} & \hat{u}\cdot\hat{z} \\ \hat{v}\cdot\hat{x} & \hat{v}\cdot\hat{y} & \hat{v}\cdot\hat{z} \\ \hat{w}\cdot\hat{y} & \hat{w}\cdot\hat{y} & \hat{w}\cdot\hat{z} \end{bmatrix}$$

As vector product is commutative, we can find from above equations that

$$^{2}R_{1} = [\,^{1}R_{2}]^{T} = [\,^{1}R_{2}]^{-1}.$$

or, in general, for only any rotational transformation matrix ***R***

$$\boldsymbol{R}^{-1} = \boldsymbol{R}^{T} \quad \text{and} \quad \boldsymbol{R}\boldsymbol{R}^{T} = \boldsymbol{I}$$

where ***I*** is the 3 X 3 identy matrix.

2.3 MAPPING BETWEEN TRANSLATED FRAMES

Consider two frames, frame {1} with axes X,Y,Z and frame {2} with axes U, V, W with origins O_1 and O_2, as shown in Fig. 2.3. A point *P* in space can be described by two frames and expressed as vectors O_1P and O_2P, w.r.t. frames {1} and {2}, respectively.

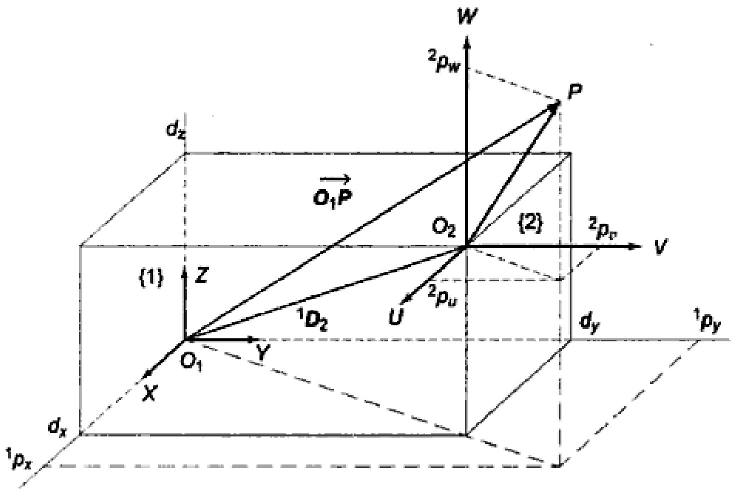


Figure 2.3 Translation of frames: *frame {2}* is translated w.r.t. *frame {1}* by a distance $^1D_2$

Two vectors are related as:

$$\overrightarrow{O_1P} = \overrightarrow{O_2P} + \overrightarrow{O_1O_2}$$

or in notation $\quad ^{1}P = {}^{2}P + {}^{1}D_2$

Substituting ^{2}P and $^{1}D_2$ in above equation we get

$$^{1}\boldsymbol{P} = (^{2}p_u + d_x)\hat{x} + (^{2}p_v + d_y)\hat{y} + (^{2}p_w + d_z)\hat{z}$$

$$^{1}\boldsymbol{P} = {}^{1}p_x\hat{x} + {}^{1}p_y\hat{y} + {}^{1}p_z\hat{z} \qquad \text{this gives}$$

$$^{1}p_x = {}^{2}p_u + d_x; \;\; ^{1}p_y = {}^{2}p_v + d_y; \;\; ^{1}p_z = {}^{2}p_z + d_z$$

We can represent it in matrix form as:

$$ {}^1\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^2p_u \\ {}^2p_v \\ {}^2p_w \\ 1 \end{bmatrix} $$

or $\quad {}^1P = {}^1T_2\ {}^2P$

Here ${}^1T_2$ is 4 X 4 homogeneous transformation matrix for translation of origin by ${}^1D_2 = O_1O_2 = [\ d_x\ d_y\ d_z\ ]^T$.

2.4 MAPPING BETWEEN RPTATED AND TRANSLATED FRAMES

Consider two frames, frame {1} with axes X,Y, Z and frame {2} with axes U, V, W with origins O_1 and O_2, as shown in Fig. 2.4. Frame {2} is rotated and translated w.r.t. frame {1}. The distance between too origins is O_1O_2 or ${}^1D_2$. Assume point *P* in space described w.r.t. frame {2} as ${}^2P$, it is required to refer it to frame {1}, that is, to find ${}^1P$.

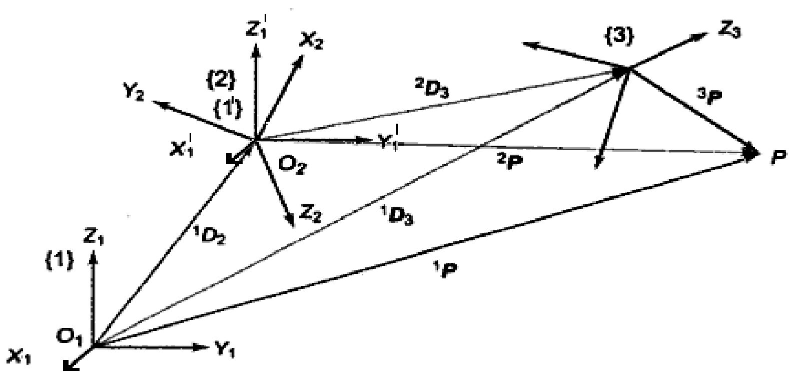


Figure 2.4 mapping between frames: *Translated* and *Rotated*

In terms of vectors in Figure 2.4,

$$ \overrightarrow{O_1P} = \overrightarrow{O_2P} + \overrightarrow{O_1O_2} $$

or in notation $\quad {}^1P = {}^2P + {}^1D_2$

The $\overrightarrow{O_2 P}$ is in frame {2}, it must be transformed to frame {1}. The intermediate frame is {1′} with its origin coincidence with O_2. The frame {1′} is rotated w.r.t. frame {2} such that its axes are parallel to axes of frame {1}. Thus frame {1′} is related to frame {2} by pure rotation and point *P* can be expressed in frame {1′} as:

$$^{1'}P = {}^{1'}R_2 \, {}^{2}P$$

Hence we get $\overrightarrow{O_1P} = {}^1R_2 \, \overrightarrow{O_2 P} + \overrightarrow{O_1 O_2}$

or in notation $^1P = {}^1R_2 \, {}^2P + {}^1D_2$

which can be written as $^1P = {}^1T_2 \, {}^2P$.

Here ^1P, ^2P are 4 X 1 matrices and $^1T_2$ is 4 X 4 matrix. It describe both position and orientation of frame {2} w.r.t. frame {1} or any frame w.r.t. any other frame. The ***T*** matrix is both containing ***R*** matrix and ***D*** vector with *scale factor* and three *perspective parameters* as given below. Perspective transformation matrix useful in vision system and is set to *zero vector*. Scale factor is having non-zero positive value and is called *global scaling* parameter and is 1 for robotic study.

$$^1T_2 = \left[\begin{array}{ccc|c} \hat{x}\cdot\hat{u} & \hat{x}\cdot\hat{v} & \hat{x}\cdot\hat{w} & d_x \\ \hat{y}\cdot\hat{u} & \hat{y}\cdot\hat{v} & \hat{y}\cdot\hat{w} & d_y \\ \hat{z}\cdot\hat{u} & \hat{z}\cdot\hat{v} & \hat{z}\cdot\hat{w} & d_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

2.5 DESCRIPTION OF OBJECT IN SPACE

The location of an object is completely specified in 3-D space by describing both position and its orientation. Consider a body *B* in space whose location is to specified w.r.t. frame {0}. Let a frame {1} with origin O_1 attached to body *B*, as shown in figure 2.5. The homogeneous transform $^0T_1$ completely describe the location (position and orientation) of body *B*. In robotic arm, the location of link is specified by assigning frames to each link, starting from the base to the tool or end-effector. The end-effector coordinate frame is shown in figure 2.6. The axes are defined as: (*i*) z-axis is approach vector $\hat{a}$, that is, the direction in which the end-effector approaches towards the target, (*ii*) y-axis is the direction of the sliding vector, that is, the direction of the opening and the closing of the end-effector as it manipulates the object. It is also called *orientation* vector $\hat{o}$ and

(iii) x-axis, the normal vector $\hat{n}$ which is orthogonal to the approach and sliding vectors in right handed manner, that is $\hat{o} \times \hat{a}$

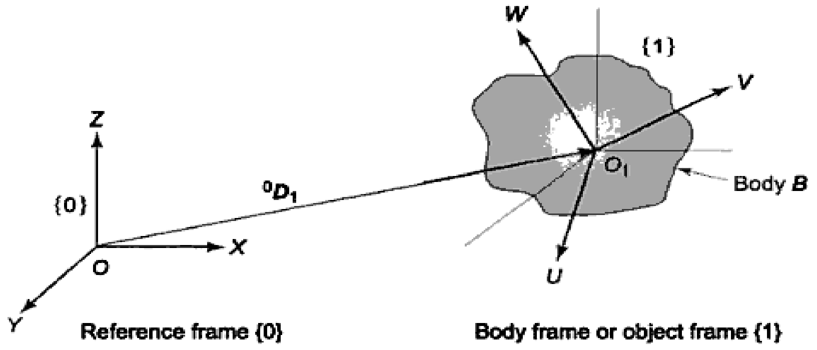


Figure 2.5 Description of a body *B* in space using homogeneous transformation

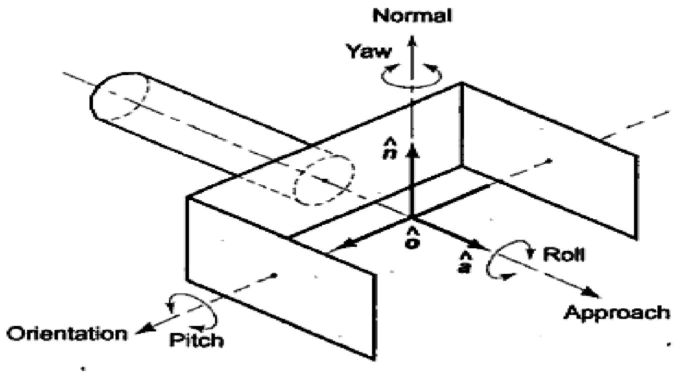


Figure 2.6 Assigning a frame to end-effector: approach, orientation and normal directions; and roll, pitch, yaw motions

Transformation matrix ***T*** for end-effector w.r.t. the coordinate frame { *n* *o* *a* } is written as:

$$\boldsymbol{T} = \begin{bmatrix} n_x & o_x & a_x & d_x \\ n_y & o_y & a_y & d_y \\ n_z & o_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} & \boldsymbol{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

There are some properties of ***n***, ***o*** and ***a*** vectors:

- These are mutually perpendicular.
- These are unity vectors.
- Determinant of rotation matrix is unity.

2.6 FUNDAMENTAL ROTATION MATRICES

Frame {2} is rotated about one or more of the principal axes, an arbitrary axis, or y some fixed angels relative to frame {1}. Here rotation of frame {2} w.r.t. frame {1} by an angel θ about z-axis of frame {1} shown in figure 2.7. The corresponding rotation matrix $^{1}\boldsymbol{R}_{2}$, known as *fundamental rotation matrix*, denoted by the symbol $\boldsymbol{R}_{z}(\theta)$ or $\boldsymbol{R}(z, \theta)$ or $\boldsymbol{R}_{z,\,\theta}$.

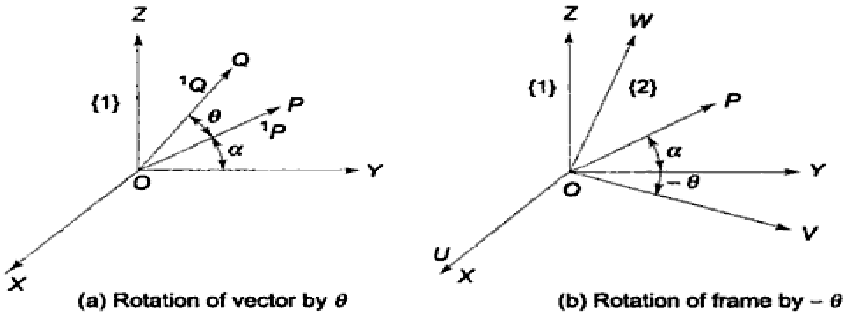


Figure 2.7 Fundamental rotation by an angel *θ* about z-axis

We can compute $\boldsymbol{R}(z, \theta)$ as

$$\boldsymbol{R}_z(\theta) = \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where Cθ = Cos θ and Sθ = Sin θ.

Also we can find $\boldsymbol{R}(x, \theta)$ and $\boldsymbol{R}(y, \theta)$ as

$$\boldsymbol{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix} \quad \text{and} \quad \boldsymbol{R}_z(\theta) = \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix}$$

It is important to note the sequence of multiplication of ***R*** matrices as matrix product is not commutative. If the transformation is done w.r.t. fixed frame at that time post-multiplication is done. If the transformation is done w.r.t. rotating frame at that time pre-multiplication is done.

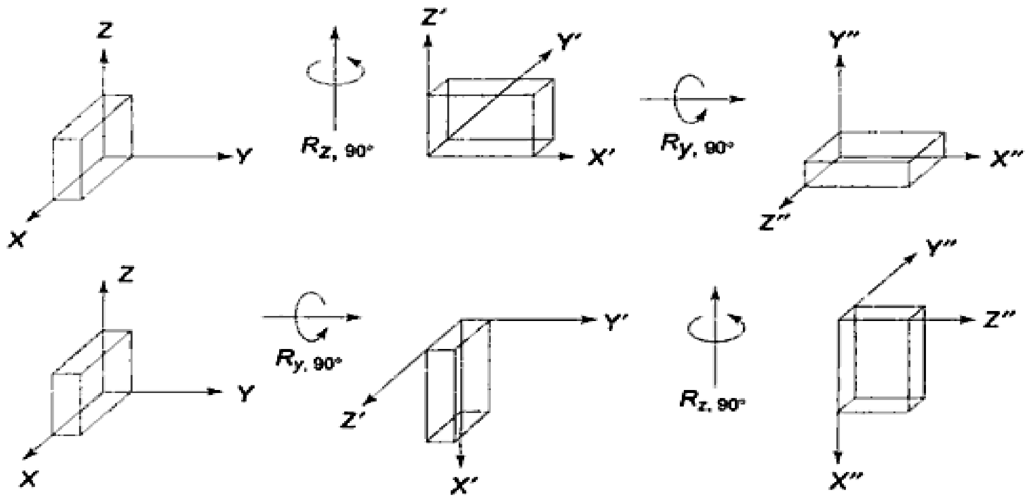


Figure 2.8 Effect of order of rotation about principal axes of fixed frame

2.7 EULER ANGELS

Euler angles are a means of representing the spatial orientation of any frame of reference (coordinate system) as a composition of rotations from a reference frame of reference (coordinate system). In the following the fixed system is denoted in lower case (x, y, z) and the rotated system is denoted in upper case letters (X, Y, Z).

- α (aka ψ) is the angle between the x-axis and the line of nodes.
- β (aka θ) is the angle between the *z*-axis and the *Z*-axis.
- γ (aka φ) is the angle between the line of nodes and the *X*-axis.

Normally, angles are defined in such a way that they are positive when they rotate counter-clock-wise. About the ranges:

- α and γ range are defined modulo 2π radians. A valid range could be (-π, π].
- β range covers π radians (but can't be said to be modulo π). For example could be [0, π] or [-π/2, π/2].

3. MANIPULATOR KINEMATICS

Kinematic model describes the spatial position of joints and links, and position and orientation of the end-effector. The derivatives of kinematics deal with the mechanics of the motion without considering the forces that cause it. The relationship between force and/or toques that cause them is dynamic problem. In designing a manipulator, kinematic and dynamic play a important role. The kinematic model gives relationship between the position and orientation of the end-effector and spatial position of the links. The *differential kinematics* of manipulators refers to differential motion i.e. velocity, acceleration, and all higher order derivatives of position variables.

3.1 MECHANICAL STRUCTURE AND NOTATIONS

A joint with *m* degrees of freedom can be modeled as *m* joints with one degree of freedom connected with ($m-1$) links of zero length. Most industrial robotic manipulators are open serial kinematic chain i.e. each link is connected with two other links, at the most, without the formation of closed loops. By convention, joint axis is z-axis. In case of ***R*** joint axis about the links rotate and incase of ***P*** joint axis of sliding or translational motion is joint axis, i.e. z-axis.

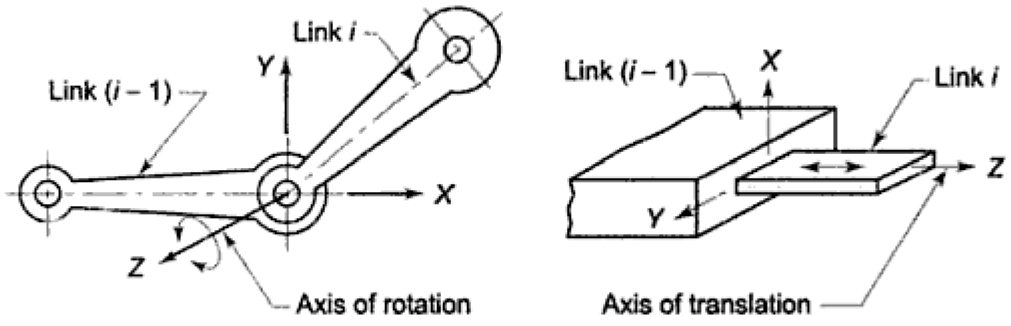


Figure 3.1 Two common types of joint and axis of motion.

Links of manipulator is numbered outwardly starting from immobile base as link-0, first moving body as link-1, towards the end-effector link-*n*. An *n*-DOF robot consists of $n+1$ links (including link-0) connected by *n* joints. To describe the position and orientation of the link in space, a coordinate frame is attached to each link, namely the frame $\{i\}$ is to link $\{i\}$. The position and orientation of the frame $\{i\}$, relative to previous frame $\{i\}$ can be described by a homogeneous transform matrix. For two links connected by either revolute or prismatic joint, the relative position of these links is measured by the displacement at the joint, which is either *joint distance* or *joint angle*, depending on the type of the joint. The joint distance d_i is the perpendicular distance between the two adjacent common normals a_{i-1} and a_i measured along axis $(i-1)$. The joint angle θ_i is angle between two adjacent common normals a_{i-1} and a_i, measured in right handed direction about the axis $(i-1)$.

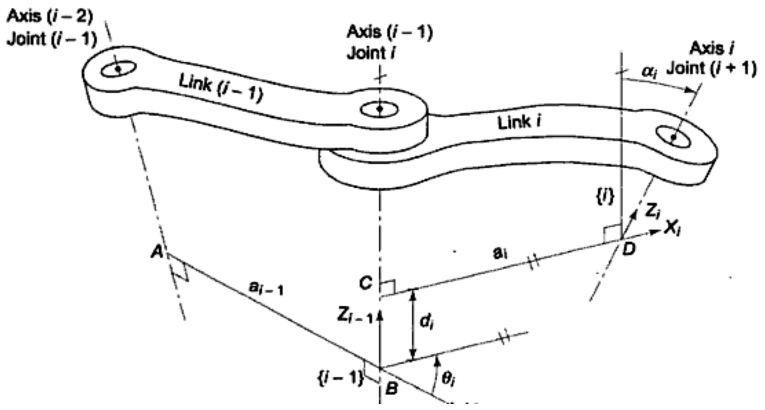


Figure 3.2 Description of joint-link parameters for joint *i* and link *i*.

For revolute joint, the joint distance d_i is zero or constant and the joint angle θ_i varies, while for a prismatic joint, the joint distance θ_i is zero or constant and the joint angle d_i varies, describing the relative position of the links. The varying parameter is known as *joint variable* and generalized parameter q is used to denote the joint displacement of either type.

$$q_i = \begin{cases} \theta_i & \text{for a revolute joint} \\ d_i & \text{for a prismatic joint} \end{cases}$$

3.2 D-H TRANSFORMATION

Figure 3.2 shows three joints. Each joint may rotate or translate. Let's assign joint number n to the first shown joint, *n+1* to the second shown joint, *n+2* to the third shown joint. There may be other joints before or after this joint. Each link is also assigned a link number. Link will be between joint n and *n+1*, link *n+1* is between joint *n+1* and *n+2*.

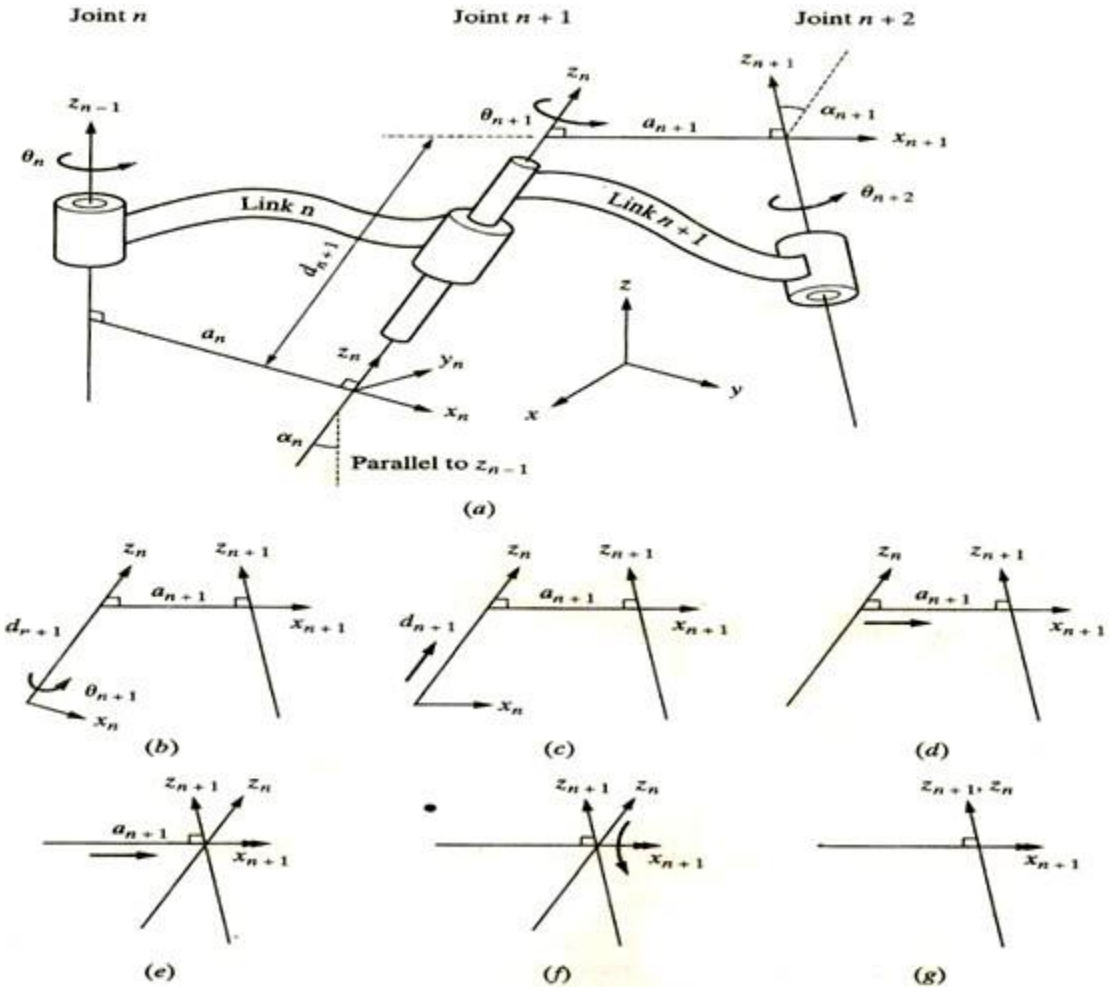


Figure 3.2 D-H representation of three links

To model the robot with D-H representation, the first thing we need to do is to assign a local reference frame for each and every point. Thus for each joint, we will have to assign a z-axis and an x-axis. We normally do not need to assign a y-axis since we always know that y-axes are mutually perpendicular to both x- and z-axes. In addition, the D-H representation does not use the y-axis at all. The procedure for assigning a local reference frame to each joint:

- All joints, without exception, are represented by z-axis. If the joint are revolute, the z-axis is in the direction of rotation as followed by the right hand rule for rotations. If the joint is prismatic, the z-axis for joint is along the direction of the linear movement. The index number for the z-axis of joint *n*(as well local reference frame foe the joint) in each case in n-1. The z-axis representing joint number $n+1$ is z_n. These simple rules allow us to quickly assign z-axes to all joins. For revolute joints, the rotation about the z-axis (θ) will be the joint variable. For prismatic joint, the length of the link along the z-axis represented by d will be the joint variable.

- As in the figure, the joints may not necessarily be parallel or intersecting. As a result, in general, the z-axes are skew lines. There is always one line mutually perpendicular to any twoskew lines, called the common normal, which has the shortest distance between the two skewlines. We always assign the x-axis of the local reference frames in the direction of the normal.Thus, if a_n represents the common normal between z_{n-1} and z_n, the direction of x_n will be along a_n. If the common normal between z_n and z_{n+1} is a_{n+1}, the direction of x_{n+1} will be along a_{n+1}. The common normal lines between successive joints are not necessarily interestingor collinear. As a result, the location of the origins of two successive frames also may not be at the same location. Based on the preceding information and with the exception of the following special cases, we can assign coordinate frames to all joints:

  - If two joints z-axes are parallel, there are infinite numbers of common normal present. We will pick the common normal that is collinear with the common normal of the previous joint.
  - If the z-axes of the two successive joints are interesting, there is no common normal between them. We will assign the x-axis along a line perpendicular to the plane formed by the two axes. This means that the common normal is a line perpendicular to the plane containing the two z-axes and is equivalent of picking the direction of the cross product of the two z-axes.

Here θ represents the rotation about the z-axis, d represents the distance on the z-axis between two successive common normal, a represents the length of each common normal, and α represents angle between two successive z-axes. Commonly, only d and θ are joint variables. To transform one reference frame to next reference frame i.e. X_n - Z_n to X_{n+1} - Z_{n+1}, we will follow four standard motions.

(I) Rotate about Z_n axis by an angle of θ_{n+1}. This will make. X_n and X_{n+1} parallel to each other.

(II) Translate along Z_n axis a distance of d_{n+1} to make X_n and X_{n+1} collinear.

(III) Translate along X_n axis a distance of a_{n+1} to bring origin of X_n and X_{n+1} together. At this point, origins of two reference frames are at same location.

(IV) Rotate Z_n-axis about X_n-axis an angle of α_{n+1} to align Z_n with Z_{n+1}. At this point, both the frames *n* and *n+1* are exactly same.

$$^{n}T_{n+1} = A_{n+1} = \mathrm{Rot}(Z, \theta_{n+1})\ \mathrm{Trans}(0,0, d_{n+1})\ \mathrm{Trans}(a_{n+1}, 0,0)\ \ \mathrm{Rot}(X, \alpha_{n+1})$$

$$= \left[\begin{array}{cccc:c} \cos\theta_{n+1} & -\sin\theta_{n+1}\cos\alpha_{n+1} & -\sin\theta_{n+1}\sin\alpha_{n+1} & & a_{n+1}\cos\theta_{n+1} \\ \sin\theta_{n+1} & -\cos\theta_{n+1}\cos\alpha_{n+1} & -\cos\theta_{n+1}\sin\alpha_{n+1} & & a_{n+1}\sin\theta_{n+1} \\ 0 & \sin\alpha_{n+1} & \cos\alpha_{n+1} & & d_{n+1} \\ \hdashline 0 & 0 & 0 & & 1 \end{array}\right]$$

In general a frame is represented in 4X4 format in which, 3X3 matrix represents the orientation and 3X1 represents position(x, y, z) of the origin of the frame. On post-multiplying above 4X4 matrix with 4X4 matrix of frame *n*, we will get 4X4 matrix of frame *n+1*. For six-degree-of-freedom robot, there will be six joints, hence six matrix multiplications as:

$$^{0}T_6 = {}^{0}T_1\ {}^{1}T_2\ {}^{2}T_3\ {}^{3}T_4\ {}^{4}T_5\ {}^{5}T_6 = A_1\ A_2\ A_3\ A_4\ A_5\ A_6$$

or

$$^{0}\mathbf{T}_i = {}^{0}\mathbf{T}_{i-1}\ {}^{i-1}\mathbf{A}_i$$

3.3 FORWARD AND INVERSE KINEMATICS

For an n-axis rigid-link manipulator, the forward kinematic solution gives the coordinate frame, or pose, of the last link. It is obtained by repeated application of $^{0}\mathbf{T}_i = {}^{0}\mathbf{T}_{i-1}\ {}^{i-1}\mathbf{A}_i$.

$$^{0}T_6 = {}^{0}T_1\ {}^{1}T_2\ {}^{2}T_3\ {}^{3}T_4\ {}^{4}T_5\ {}^{5}T_6 = A_1\ A_2\ A_3\ A_4\ A_5\ A_6 = K(q)$$

which is the product of the coordinate frame transform matrices for each link. The pose of the end-effector has 6 degrees of freedom in Cartesian space, 3 in translation and 3 in rotation, so robot manipulators commonly have 6 joints or degrees of freedom to allow arbitrary end-effector pose.

The overall manipulator transform ${}^{0}T_{n}$ is frequently written as T_n, or T_6 for a 6-axis robot. The forward kinematic solution may be computed for any manipulator, irrespective of the number of joints or kinematic structure. Of more use in manipulator path planning is the inverse kinematic solution

$$q = K^{-1}(T)$$

which gives the joint angles required to reach the specified end-effector position. In general this solution is non-unique, and for some classes of manipulator no closed-form solution exists. If the manipulator has more than 6 joints it is said to be redundant and the solution for joint angles is under-determined. If no solution can be determined for a particular manipulator pose that configuration is said to be singular. The singularity may be due to an alignment of axes reducing the effective degrees of freedom, or the point T being out of reach. The manipulator Jacobian matrix, Jθ, transforms velocities in joint space to velocities of the end-effector in Cartesian space. For an n-axis manipulator the end-effector Cartesian velocity is

$$\begin{aligned} {}^{0}\dot{x}_{n} &= {}^{0}\mathbf{J}_{\theta}\dot{q} \\ {}^{t_n}\dot{x}_{n} &= {}^{t_n}\mathbf{J}_{\theta}\dot{q} \end{aligned}$$

In base or end-effector coordinates respectively and where x is the Cartesian velocity represented by a 6-vector. For a 6-axis manipulator the Jacobian is square and provided it is not singular can be inverted to solve for joint rates in terms of end-effector Cartesian rates. The Jacobian will not be invertible at a kinematic singularity, and in practice will be poorly conditioned in the vicinity of the singularity, resulting in high joint rates. A control scheme based on Cartesian rate control

$$\dot{q} = {}^{0}\mathbf{J}_{\theta}^{-1}\,{}^{0}\dot{x}_{n}$$

was proposed by Whitney and is known as resolved rate motion control. For two frames A and B related by ${}^{A}T_{B} = [\,n\ o\ a\ p\,]$ the Cartesian velocity in frame A may be transformed to frame B by

$${}^{B}\dot{x} = {}^{B}\mathbf{J}_{A}\,{}^{A}\dot{x}$$

where the Jacobian is given by Paul as

$${}^{B}\mathbf{J}_{A} = f({}^{A}\mathbf{T}_{B}) = \begin{bmatrix} [n\ o\ a]^{T} & [p\times n\ p\times o\ p\times a]^{T} \\ 0 & [n\ o\ a]^{T} \end{bmatrix}$$

4. DYNAMIC MODELING

During the work cycle a manipulator must accelerate, move at constant speed and decelerate. This time-varying position and orientation of manipulator is termed as its dynamic behavior. Time-varying torques are applied at the joints to balance out the internal and external forces. The internal forces are cause by motion of link. Inertial, Coriolis, and frictional forces are some of the internal forces. The external forces are the forces exerted by the environment. These include load and gravitational forces. The mathematical equations, often referred as *manipulator dynamics*, are set of *equation of motion (EOM)* that describe the dynamic response of the manipulator to input actuator torques. The manipulator control maintains the dynamic response of the manipulator to obtain the desired performance, which directly depends on accuracy of the dynamic model and efficiency of the control algorithm. Simulation of manipulator motion permits the testing of control strategies, motion planning, and performance studies without a physical prototype of the manipulator. The serial link manipulator represents a complex dynamic system, which can be modeled by systematically using known physical laws of Lagrangian mechanics or Newtonian mechanics. Assuming rigid body motion, no backlash, no friction and neglecting effect of control component dynamic, the resulting EOM are set of second order, coupled, non-linear differential equation, consisting of inertia loading and coupling forces between joints. Anther method, the generalized d' Alembert principle, provide an "equivalent dynamic" model.

4.1 LAGRANGIAN FORMULATION OF MANIPULATOR DYNAMICS

We can say Newton-Euler formulation is "force balanced" approach to dynamics, Lagrangian Formulation is based on the "energy-based" approach to dynamics, and both will give same equation of motion. Our statement of Lagrangian dynamics will be brief and some what specialized to the case of a serial chain mechanical manipulator with rigid links.

We start by developing an expression for the kinetic energy of a manipulator. The kinetic energy of the ith link, k_i, can be expressed as

$$k_i = \frac{1}{2} m_i v_{C_i}^T v_{C_i} + \frac{1}{2}\, {}^i\omega_i^T \; {}^{C_i}I_i \; {}^i\omega_i$$

where k_i is kinetic energy due to linear velocity of link's centre of mass, and the second term is kinetic energy due to angular velocity of link. The total kinetic energy of manipulator is sum of kinetic energy of the individual links; that is ,

$$k = \sum_{i=1}^{n} k_i$$

Since ${}^{v}c_i$ and ${}^{i}w_i$ are function of Θ and ϴ , we see that the kinetic energy of the manipulator can be described by a scalar formula as a function of joint position and velocity, $k(\Theta, \dot{\Theta})$. In fact kinetic energy of the manipulator is given by

$$k(\Theta, \dot{\Theta}) = \frac{1}{2}\dot{\Theta}^T M(\Theta)\dot{\Theta}$$

where M(Θ) is $n \times n$ manipulator mass matrix.

The total potential energy of the ith link, u_i can be expressed as

$$u_i = -m_i \ {}^{0}g^{T} \ {}^{0}Pc_i + u_{ref\,i}$$

where ${}^{0}g$ is 3×1 gravity vector, ${}^{0}Pc_i$ is vector locating the centre of the mass of the ith link, and $u_{ref\,i}$ is a constant chosen so that the minimum value of u_i will be zero. The total potential energy stored in the manipulator is sum of potential energy of the individual links; that is,

$$u = \sum_{i=1}^{n} u_i$$

Since the ${}^{0}Pc_i$ is function of Θ, we see that the potential energy of a manipulator can be described by a scalar formula as a function of joint position, $u(\Theta)$.

The Lagrangian dynamic formulation provides a means of deriving the equations of motion from a scalar formula as a function called **Lagrangian**, which is defined as the difference between the kinetic energy and potential energy of a mechanical system. In our notation, the Lagrangian of a manipulator is

$$\mathcal{L}(\Theta, \dot{\Theta}) = k(\Theta, \dot{\Theta}) - u(\Theta).$$

The equation of motion for the manipulator are given by

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\Theta}} - \frac{\partial \mathcal{L}}{\partial \Theta} = \tau$$

where τ is *n*x*1* vector of actuator torques. In the case of a manipulator, the equation becomes

$$\frac{d}{dt}\frac{\partial k}{\partial \dot{\Theta}} - \frac{\partial k}{\partial \Theta} + \frac{\partial u}{\partial \Theta} = \tau$$

where the argument of *k(·)* and *u(·)* have been dropped for brevity.

4.2 ITERATIVE NEWTON-EULER DYNAMIC FORMULATION

***Euler's Equation**:* Figure shows a rigid body rotating with angular velocity, ω and with angular acceleration ώ. In such situation, the moment *N*, which must be acting on the body to cause this motion is given by Euler's equation

$$N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega.$$

where ${}^{C}I$ is the inertia tensor of the body written in a frame, *{C}*, whose origin is located at the centre of the mass.

We now consider the problem of computing the torques that correspond to a given trajectiory of a manipulator. We assume we know the position velocity, and acceleration of the joints, (θ, θ̇, θ̈). With this knowlledge and with knowlledge of the kinematics and mass distribution information of the robot, we can calculate the joint torques requred to cause this motion.

Outward Iterations to Compute the Velocity and the Acceleration: In order to compute inertial forces acting on the links, we have to calculate the rotational velocity and linear and rotational acceleration of the centre of each link of the manipulator at any given instant. These computation will done in an iterative nature starting with link1 and moving successfully, link by link, outward to link n.

The propagation of rotational velocity from link to link given by equation

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}$$

We can obtain the equation for transforming angular acceleration from one link to next

$$ {}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} $$

When the joint is prismatic it simplifies to

$$ {}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i $$

The linear acceleration of each link frame origin is obtained

$$ {}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left[{}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right] $$

Which, for prismatic joint *i+1*, become

$$ {}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right) + 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} $$

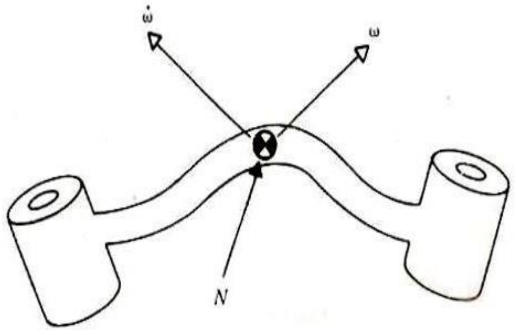


Figure 4.2 A moment N is acting on a body and body is rotating with a velocity ω and acceleration ώ

We also need the linear acceleration of the centre of mass of each link, which also can be found as

$$ {}^{i}\dot{v}_{C_i} = {}^{i}\dot{\omega}_i \times {}^{i}P_{C_i} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{C_i}\right) + {}^{i}\dot{v}_i $$

Where we imagine a frame, {C*i*}, attached to each link with its origin located at centre of mass of the link, and the same orientation as the link frame, {*i*}.

In general we take ${}^{0}\omega_0 = 0$, ${}^{0}\dot{\omega}_0 = 0$, ${}^{0}v_0 = 0$ and ${}^{0}\dot{v}_0 = [\, g_x, g_y, g_z \,]^T$.

The Force and Torque acting on a link: Having computed the linear and angular acceleration of the mass of the centre of each link, we can apply the Newton-Euler's equation to compute the inertial force and torque acting at the center of mass of each link. Thus, we have

$$F_i = m\,\dot{v}\ c_i$$

$$N_i = c_i\, I\, \dot{w}_i$$

Where {Ci} has it's origin at the center of the mass of the link, and has the same orientation as the link frame, {i}.

Inward Iterations to Compute the Velocity and the Acceleration: Having computed the forces and torques acting on each link, it now remains to calculate the joint torques which will result in these net forces and torques being applied to each link. We can do this by writing a force balance and moment balance equation based on a free body diagram of a typical link. Each link has forces and torques exerted on its neighbors, and in addition experiences an internal force and torque.

f_i = force exerted on link i by link i-1,

n_i = torque exerted on link i by link i-1.

By the forces acting on link i we arrive at a force balance relationship,

$$^{i}F_{i} = {}^{i}f_{i} - {}_{i+1}^{i}R\ \ {}^{i+1}f_{i+1}$$

$$\text{or} \qquad {}^{i}f_{i} = {}^{i}F_{i} + {}_{i+1}^{i}R\ \ {}^{i+1}f_{i+1}$$

By summing the torque about the centre of the mass and setting the m equal to zero we arrive at the torque balance equation:

$$^{i}N_{i} = {}^{i}n_{i} - {}_{i+1}^{i}R\ {}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_{i+1} + {}^{i}P_{i+1} \times {}_{i+1}^{i}R\ {}^{i+1}f_{i+1}$$

$$\text{or} \qquad {}^{i}n_{i} = {}^{i}N_{i} + {}_{i+1}^{i}R\ {}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_{i+1} + {}^{i}P_{i+1} \times {}_{i+1}^{i}R\ {}^{i+1}f_{i+1}$$

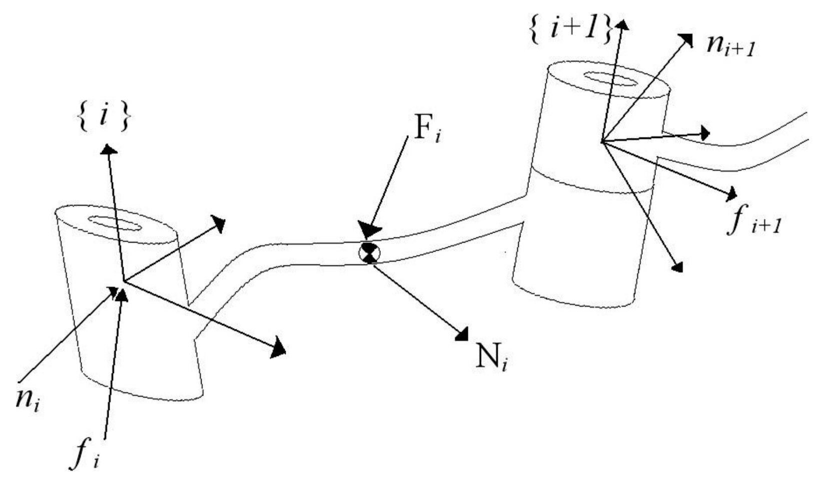


Figure 4.3 The force balance, including inertial forces, for a single manipulator link.

As in the static case, the required joint torques are found by taking the *Z* component of the torque applied by one link on its neighbor:

$$\tau_i = {}^{i}n_i^T \; {}^{i}\hat{Z}$$

For joint *i+1* prismatic, we use

$$\tau_i = {}^{i}f_i^{T} \; {}^{i}\hat{Z}$$

The Iterative Newton-Euler Algorithm: (For Revolute joints like in PUMA 560)

$${}^{0}\omega_0 = 0,\; {}^{0}\acute{\omega}_0 = 0,\; {}^{0}v_0 = 0 \text{ and } {}^{0}\acute{v}_0 = [\, g_x, g_y, g_z \,]^T .$$

Outward iteration: i : 0 to 5

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{\;\,i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}.$$

$${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{\;\,i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{\;\,i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}.$$

$${}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{\;\,i}R\left({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times \left({}^{i}\omega_i \times {}^{i}P_{i+1}\right) + {}^{i}\dot{v}_i\right).$$

$${}^{i+1}\dot{v}_{C_{i+1}} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} + {}^{i+1}\omega_{i+1} \times \left({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}}\right) + {}^{i+1}\dot{v}_{i+1}.$$

$${}^{i+1}F_{i+1} = m_{i+1}\,{}^{i+1}\dot{v}_{C_{i+1}}.$$

$${}^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1}.$$

Inward iteration: i : 6 to 1

$${}^{i}f_{i} = {}^{i}F_{i} + {}^{i}_{i+1}R\;{}^{i+1}f_{i+1}$$

$${}^{i}n_{i} = {}^{i}N_{i} + {}_{i+1}{}^{i}R\;{}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_{i+1} + {}^{i}P_{i+1} \times {}_{i+1}{}^{i}R\;{}^{i+1}f_{i+1}$$

$$\tau_{i} = {}^{i}n^{T}_{i}\;{}^{i}\hat{Z}$$

3.2.3 Description of 5-dof revolute Pioneer2 manipulator

The third configuration considered for the kinematic analysis is Pioneer2 manipulator with 5 joint rotations. If the manipulator is redundant or having high dof, than conventional solution for inverse kinematic problem becomes more complicated. Therefore, considering newly developed Pioneer2 manipulator with 5 joint rotations for the kinematic analysis as shown in Figure 3.3. This manipulator is compact, low cost and lightweight for the use in research as well as in academic purpose. The actuation of the joint is driven by open loop servo motors and gripper which is attached to the end of the last link of manipulator.

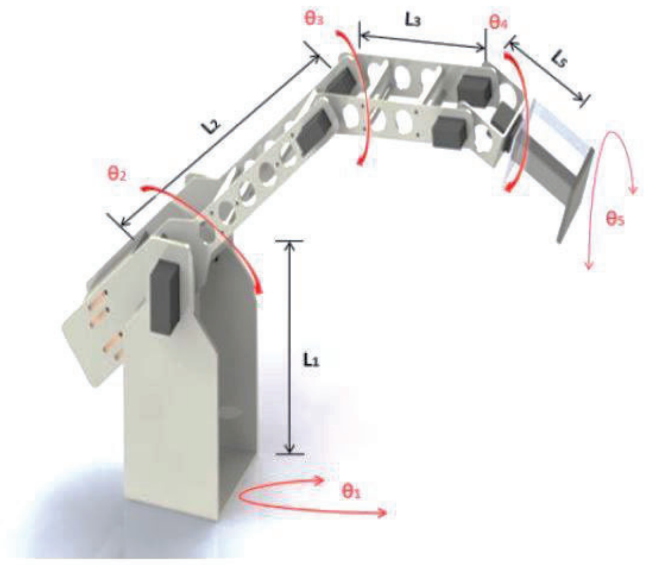


Figure 3.3 Structure of the Pinoneer arm2

The major application of this robot is for grasping and manipulation of objects like soda cans up to the weight limit of 150grams within the workspace. Joint of the Parm2 are;

Joints rotations:

- base rotation
- shoulder rotation

- elbow rotation
- wrist rotation
- gripper mount
- gripper fingers

All joints are driven by servo motors except gripper fingers. The joint limits and parameters taken for the research has presented in Table 3.4.

Table 3.4 Parm2 manipulator joint limits and kinematic parameters.

Joints	θ_i (degree)	d_i (mm)	a_i (mm)	α_i (degree)
0	$\theta_1 = \pm 180^0$	d_1= 150	a_1= 60	-90
1	$\theta_2 = \pm 180^0$	0	a_2= 145	0
2	$\theta_3 = \pm 180^0$	0	0	-90
3	$\theta_4 = \pm 180^0$	d_2= 125	0	90
4	$\theta_5 = \pm 180^0$	0	0	-90
5	0	d_3= 130	0	0

From Table 3.4 parameters and joint variables are listed and the ranges are presented. These parameters and joint variables are basis for the forward and inverse kinematic analysis of robot manipulator. Later using MATLAB programming the data sets for inverse kinematic solution will be used.

3.2.4 Description of 6-dof PUMA 560 manipulator

The fourth material for the kinematic analysis is PUMA 560 (Programmable Universal Machine for Assembly, or Programmable Universal Manipulation Arm) which is an industrial robot with six axis joints see Figure 3.4.

Table 3.5 Maximum limit of joint variables.

Joints	Limits (degree)
Waist	320
Shoulder	266
Elbow	284
Wrist pitch	200
Wrist roll	280
Wrist yaw	532

The end effector of the PUMA 560 robot is designed to operate a nominal load of 2.5kg with 0.1mm positional repeatability. The workspace of this manipulator is 0.92m from the centre axis to wrist centre and the maximum end effector velocity reaches 1m/s. All six joints are actuated through the brushed DC servo motors. The joints limits and parameters are given in Table 3.5 and Table 3.6.

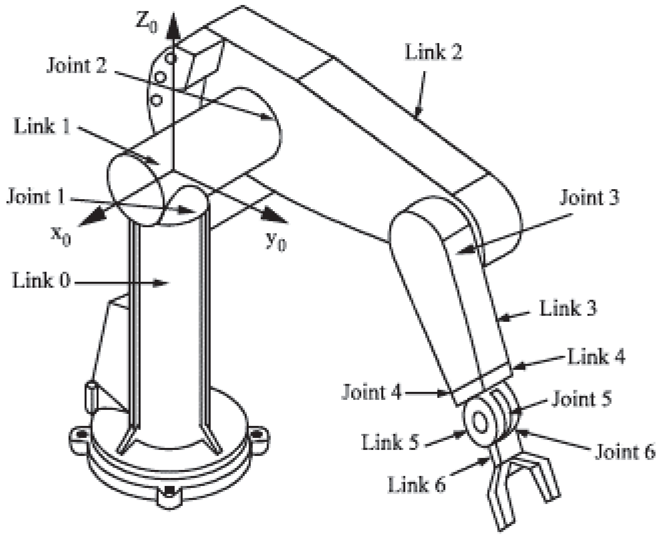


Figure 3.4 Structure of PUMA 560 robot manipulator [10]

Table 3.6 Joint variable and parameters of PUMA 560 robot

Joints	θ_i (degree)	d_i (m)	a_i (m)	α_i (degree)
0	$\theta_1 = \pm 160^0$	0	0	90
1	$\theta_2 = -225^0 \text{ to} + 45^0$	0	0	0
2	$\theta_3 = -45^0 \text{ to} + 225^0$	d_3=0.1244	a_2=0.4318	-90
3	$\theta_4 = \pm 110^0$	d_4=0.4318	a_3=0.0203	90
4	$\theta_5 = \pm 100^0$	0	0	-90
5	$\theta_6 = \pm 266^0$	0	0	0

PUMA 560 robots are most used in handling of small objects or parts in industrial due to its compact design, high speed ratio, repeatability and flexibility. Most complicated application or assembly of intricate parts can be done by PUMA 560 robot. For example PUMA 560 robot can be used for assembling of automotive panels, small electric motors, circuit board printings, appliances and so on. From Table 3.6 joint variables and parameters for DH-algorithms will be used to calculate the forward and inverse kinematic of PUMA 560 manipulator. Later the generated data sets will be input for the ANN models training and testing.

3.2.5 Description of 6-dof ABB IRb-1400 manipulator

The ABB IRB 1400 is 6-dof industrial robot which is specially desied for the manufacturing industries. The configuration of the manipulator is 6-dof revolute with rigid structure see Table 1.4. Due to its open structure it is easily adopted for the flexible automation use and also flexible communication with external systems. This type of robot manipulator is known as anthropomorphic with 6-dof mechanism. The shoulder joint with roll and pitch motions moves the upper arm $\pm 170^0$ and $\pm 70^0$; the elbow joint with pitch actions drives the forearm $+70^0$ to -65^0; and the wrist roll and pitch rotations together with the tool-plate roll move the hand (see Figure 3.5). The joint limits and associated parameters are listed in Table 3.7 for ABB IRb-1400 manipulator.

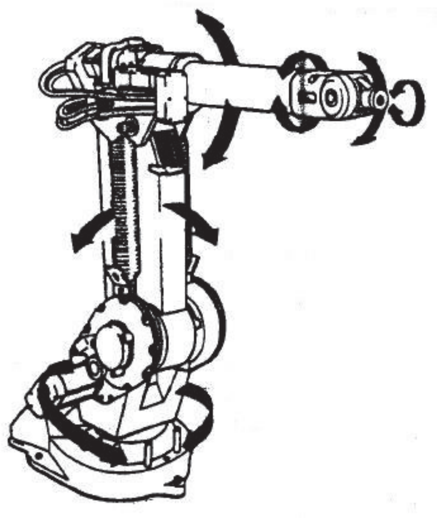


Figure 3.5 Configurations of ABB IRB 1400

Table 3.7 ABB IRB-1400 manipulator joint limits and kinematic parameters

Joints	θ_i (degree)	d_i (mm)	a_i (m)	α_i (degree)
1	$\theta_1 = \pm 170^0$	$d_1 = 0475$	0	0
2	$\theta_2 = \pm 70^0$	0	150	90
3	$\theta_3 = +70^0 \text{ to} -65^0$	0	600	0
4	$\theta_4 = \pm 150^0$	$d_4 = 720$	120	90
5	$\theta_5 = \pm 115^0$	0	0	-90
6	$\theta_6 = \pm 300^0$	$d_6 = 85$	0	90

This type of robot manipulator is mostly used for the arc welding process. The major advantages of this type of manipulator are its driveability, repeatability, accuracy with zero backlash and high resolution with nominal payloads. Apart from its technical advantages, it is commonly used in industries and research work. Therefore, this robot manipulator is selected for the forward and inverse kinematic analysis.

3.2.6 Description of 6-dof ASEA IRb-6 manipulator

The ASEA IRB 6 is 5-dof industrial robot manipulator which allows movement in 5-axis with maximum lfting capacity of 6 kg. This type of manipulator are commonly used in industries and research work. The configuration of the manipulator is 5-dof revolute with rigid structure see Table 1.4. The structure of the manipulator is rigid with maximum reach of 1114 mm. Due to its high dexterity it is accepted in industries for material handling, packaging, pick-n-place object, asemblely etc. The basic model of this manipulator is presented in Figure 3.6.

The shoulder joint with roll and pitch motions moves the upper arm 90^0 to 130^0 and 50^0 to 130^0; the elbow joint with pitch actions drives the forearm -130^0 to -50^0 to -25^0 to -220^0; and the wrist roll and pitch rotations together with the tool-plate roll move the hand. The joint limits and associated parameters are listed in Table 3.8 for ASEA IRb-6 manipulator.

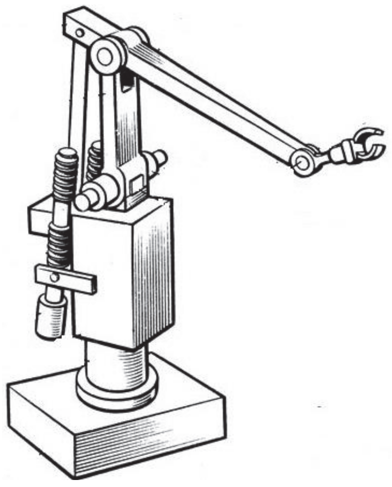


Figure 3.6 Configurations of ASEA IRb-6

Table 3.8 ASEA IRb-6 manipulator joint limits and kinematic parameters

Joints	θ_i (degree)	d_i (m)	a_i (m)	α_i (degree)
1	$\theta_1 = 90^0 - 130^0$	$d_1 = 070$	0	90
2	$\theta_2 = 50^0 - 130^0$	0	0.45	0
3	$\theta_3 = -130^0$ to -50^0	0	0.67	-0
4	$\theta_4 = -25^0$ to -220^0	0	0	90
5	$\theta_5 = 360^0$	$d_5 = 0.095$	0	0

This manipulator is basically designed to work on automated handling of grinding operation and later it became popular for many other applications such as material handling, packaging, assembling etc. The structure of this manipulator is compact and rigid. This type of manipulator is also accepted for research work. Therefore, in this research work, ASEA IRb-6 robot manipulator is selected for the kinematic analysis.

3.2.7 Description of 6-dof STAUBLI RX160L manipulator

The Stäubli RX160L industrial robot manipulator is designed to perform many industrial applications such as material handling, welding, spraying, and assembling and also for research work. The structure of this manipulator is rigid with 6-axis of rotations. The main feature of this robot is high dexterity and flexibility in industrials applications. The Stäubli RX160L is resembles the human hand dexterity with 6-dof revolute joints.

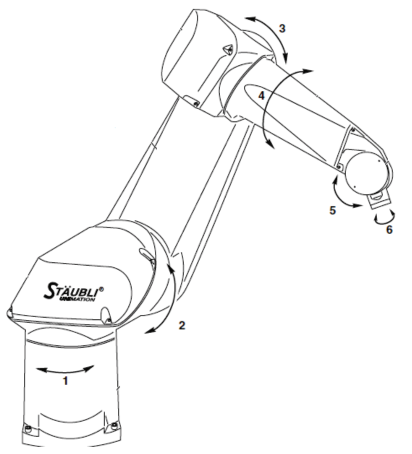


Figure 3.7 Configurations of STAUBLI RX160L

The major displacement joint angles are similar to PUMA, IRB-1400 but it allows more workspace as compared to other adopted manipulator. The maximum payload of this manipulator is 28 kg and nominal load is 14 kg. Maxium reach of this manipulator in between axis 1 to axis 6 is 2010mm which allows more work envelope. The basic model of this manipulator is prese4nted in Figure 3.7 and joint variable with kinematic parameters are presented in Table 3.9.

Table 3.9 STAUBLI RX160L manipulator joint limits and kinematic parameters.

Joints	θ_i (degree)	d_i (m)	a_i (m)	α_i (degree)
1	$\theta_1 = \pm 180^0$	$d_1 = 0.3170$	0	-90
2	$\theta_2 = \pm 101.05^0$	0	0	90
3	$\theta_3 = \pm 180^0$	$d_3 = 0.4500$	0	-90
4	$\theta_4 = \pm 153.73^0$	0	0	90
5	$\theta_5 = \pm 270^0$	$d_5 = 0.4800$	0	-90
6	$\theta_6 = \pm 180^0$	0	0	90

6. TRAJECTORY PLANNING

The end-effector of manipulator is to move in a particular fashion to accomplish the specific task. The execution of the specific task requires the manipulator to follow a pre-planned path, which is major problem of motion or trajectory planning and motion control. The goal of trajectory planning is to describe the request motion of a manipulator as a time sequence of joint/link/end-effector location and derivative of locations, which are generated by “interpolating” or “approximating” the desired path by a polynomial function. Here some techniques for trajectory planning for point-to-point motion and continuous-path motion are presented. Main objective of the trajectory planning is to get a smooth motion of manipulator and trajectories. The smooth motion has an important advantage of reducing the vibrations and wear of mechanical system.

6.1. STEPS IN TRAJECTORY PLANNING

(i) Task Description: The first step in motion planning is to identify the kind of motion required. Task can be grouped into 3 different categories.

In the case of application such as *pick and place* operation, task is specified as initial and final end-effector location. This is called point to *point-to-point* motion. No particular specification about the intermediate locations of end-effector.

If in addition to start and finish point, a specific path between them is required to be traced by the end-effector in Cartesian space, this is called *continuous path* motion and trajectory. Operations such as welding and plotting are example of continuous path motion.

There is third type of task description, where more than two points of path are specified. This is done to ensure a better monitoring of executed trajectory in case of application similar to point-to-point motion.

Apart from required task attributes, user can also include temporal attribute such as travel time in task specification.

(ii) ***Selecting and employing a Trajectory Planning Technique:*** The various trajectory-planning falls into one of two categories: *Joint space techniques* or *Cartesian space techniques.* In case of point-to-point motion, Joint space techniques are employed in which motion planning is done at joint level. The Joint space planning scheme generate time-dependent function of all joint variable and their first two derivatives to describe their motion of manipulator. For application requiring continuous path motion, *Cartesian space techniques* are used. The Cartesian space planning scheme provide time history of the location, velocity, acceleration of the end-effector w.r.t. the base. The corresponding joint variables and their derivatives are computed, using *inverse kinematics.*

(iii) ***Computing the Trajectory:*** Time sequence values are attained by the functions generated from the trajectory planning technique are computed at a particular path update or sampling rate. Path update rate in real time lies between 20Hz to 200Hz in typical industrial manipulator systems.

6.2. JOINT SPACE TECHNIQUE

The first step in these techniques is to obtain the corresponding set of joint variable valued for each specified path point, either by employing the inverse kinematics algorithm or variable can directly recorded if trajectory planning is performed by *technique-by-showing- technique*. The next step is to find smooth function *q(t)* for each n-joints of an n-DOF manipulator. For that, we need *travelling time, initial and final locations*. Two functions are examined here;

6.2.1. A *cubic polynomial* in which cubic connects the points of each joint in a smooth way.

6.2.2. A *linear function with parabolic blend* in which each segment between two successivepoints contains a linear function with parabolic blends near the path points.

6.2.1 Cubic Polynomial Trajectories:

Let q^s and q^g be the starting and goal point values, respectively, of the joint variable *q*. Generalized joint variable $q_i(\theta_i \text{ or } d_i)$ denotes for joint *i* of n-DOF manipulator. Motion begins at $t = 0$ and ends at $t = t_g$ where t_g is travel time decided by user or program. Both $q(t)$ and its derivatives should be smooth from $t = 0$ to $t = t_g$. Therefore, to describe joint motion, we assume that cubical polynomial is

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

which gives parabolic velocity and linear acceleration profile

$$\dot{q}(t) = a_1 + 2\, a_2 t + 3\, a_3 t^2$$

and $$\ddot{q}(t) = 2a_2 + 6\, a_3 t$$

From above we can get the values of a_0 , a_1 , a_2 and a_3.

Matlab Program:

```
qs=input('qs=');
qg=input('qg=');
t=input('t=');

  a=qs;
% b=0;
  c=3*(qg-qs)/(t*t);
  d=-2*(qg-qs)/(t*t*t);

% 1 unit = 50ms
% -----------------------------------------------------------------------------------------------------
tt= ceil(20*t);

for i=1:tt
        q(i)=(a+(c*i*i/400)+(d*i*i*i/8000));
```

```
end
q

for i=1:tt
        qd(i)=((2*c*i/20)+(3*d*i*i/400));
end
qd

for i=1:tt
        qdd(i)=((2*c)+(6*d*i/20));
end
qdd
```

6.2.2 Linear function with parabolic blends trajectories:

In these cases, there will be a constant acceleration up to certain time say t_b and then constant velocity say *v* i.e. zero acceleration and then for time period t_g constant retardation to make velocity from *v* to zero. There will be a trapezoidal velocity profile. Also we can make the triangular velocity profile for small value of joint parameter. Here we have to choose the travel time in parabolic blends and if we fix the total time of motion then we can calculate the value of acceleration else if we fix the value of acceleration then we can calculate the value of acceleration.

Matlab Program:

```
qs=input('qs= ');
qg=input('qg= ');
q=qg-qs;

% 1unit = 50ms; acceleration= retardation= pi/4; time of blends= 1sec each
% -------------------------------------------------------------------------------------------------
if(q>0)
  if (q>pi/4)
    ti= (4*q)/pi;
    ti20=floor(20*ti);
    ti21=floor(20*(ti+1));
        for i=1:20
        qd(i)=(pi/4)*i/20;
    end
    for i=21:ti20
        qd(i)=pi/4;
    end
    for i=(ti20)+1:(ti21)
    qd(i)=(pi/4)*(ti21-i)/20;
    end

  else (q<=pi/4)
    ti=abs(sqrt(4*q/pi));
    ti20=floor(20*ti);
      for i=1:ti20
        qd(i)=(pi/4)*(i/20);
      end
      for i=(ti20)+1:2*ti20
        qd(i)=(pi/4)*(2*ti20-i)/20;
      end
  end
else
   if (q>=-pi/4 )
      q=abs(q);
      ti20=floor(20*abs(sqrt(4*q/pi)));
      for i=1:ti20
        qd(i)=(-pi/4)*(i/20);
      end
      for i=(ti20)+1:2*ti20
        qd(i)=(-pi/4)*(2*ti20-i)/20;
      end

   else
    q=abs(q);
    ti= (4*q)/pi;
    ti20=floor(20*ti);
    ti21=floor(20*(ti+1));
    for i=1:20
        qd(i)=(-pi/4)*(i/20);
  end
   for i=21:ti20
       qd(i)=-pi/4;
   end
   for i=(ti20+1):(ti21)
       qd(i)=(-pi/4)*(ti21-i)/20;
   end
   end
end

qd
q=cumsum(qd/20)+qs
qdd=diff(qd*20)

i=size(qd)
k=size(qdd)
j=1:1:i(2);
```

```
stem(j/20,qd);
j=1:1:k(2);
stem(j/20,qdd);
```

The above program will generate trapezoidal velocity profile for a joint with acceleration/retardation of π/4 m/s^2 if joint angle difference q_g-q_s is more than π/4 radian else a triangular profile with same acceleration/retardation.

6.3. CARTESIAN SPACE TECHNIQUE

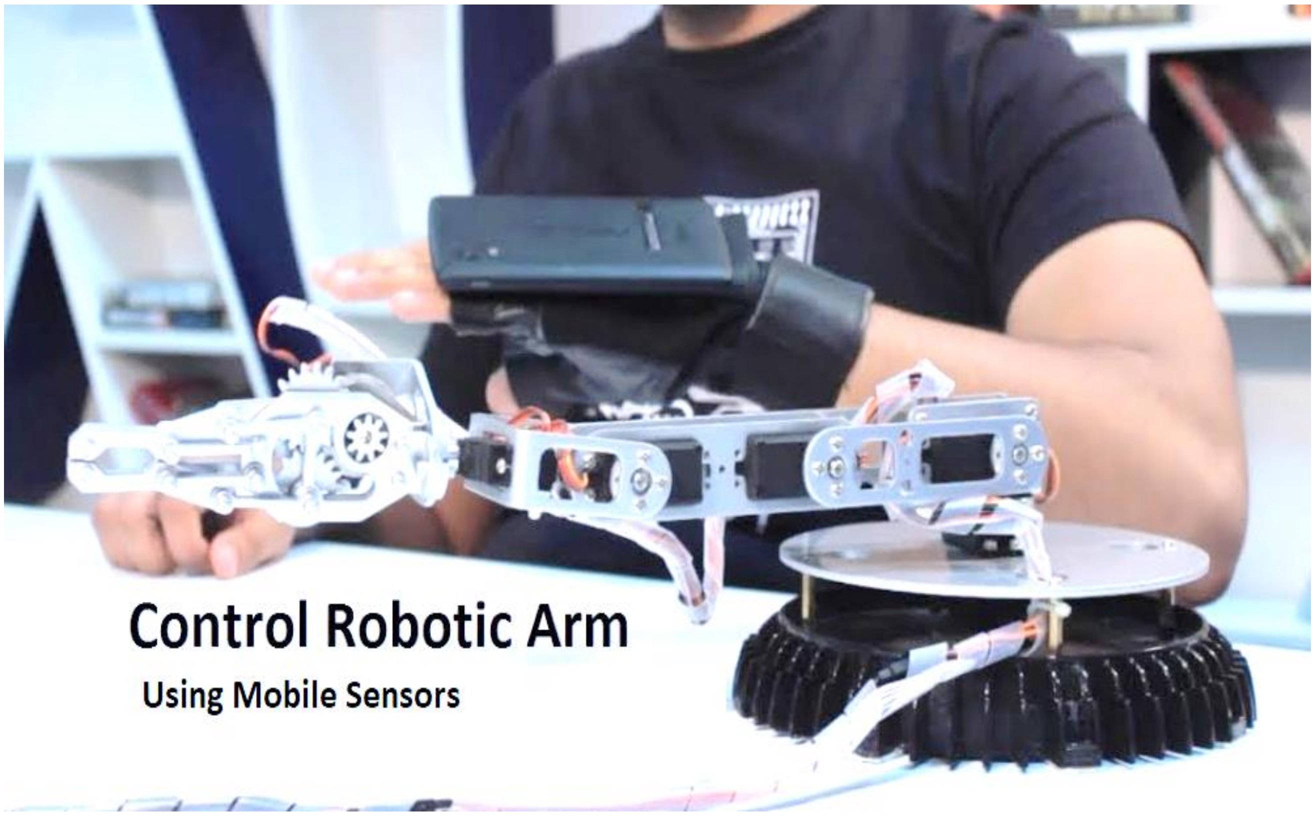
In Cartesian space, the position and orientation of rigid body can be clearly defined. The problem of planning trajectories that enable the manipulator's end effector to track a given path in Cartesian space is investigated in this section. The user specifies the desired end-effector path, the travelling time, and tool orientations along the path. It is helpful when there is an obstacle in the path but it is much more complex than joint space technique. Some common techniques for Cartesian space trajectory planning are:

6.3.1. Parametric Description of a Path
6.3.2. A Straight-Line Path
6.3.3. A Circular path
6.3.4. Position planning
6.3.5. Orientation Planning etc.

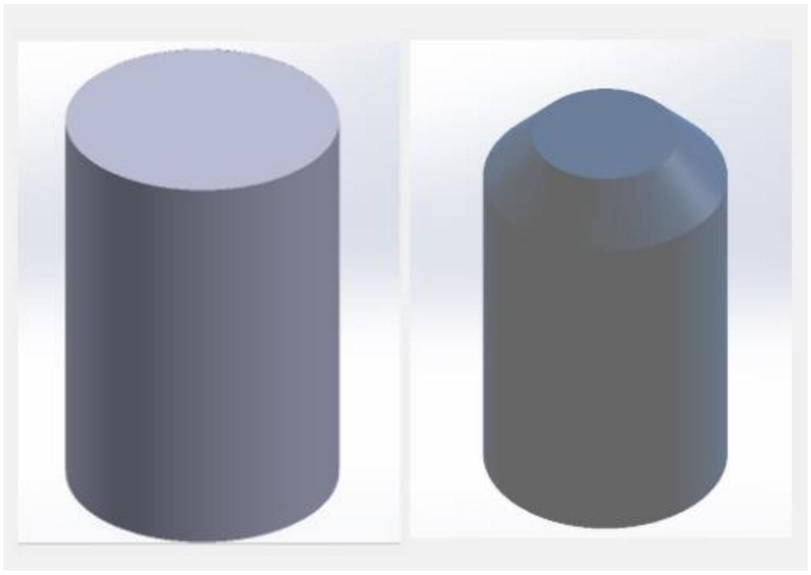
7. TRAJECTORY GENERATION

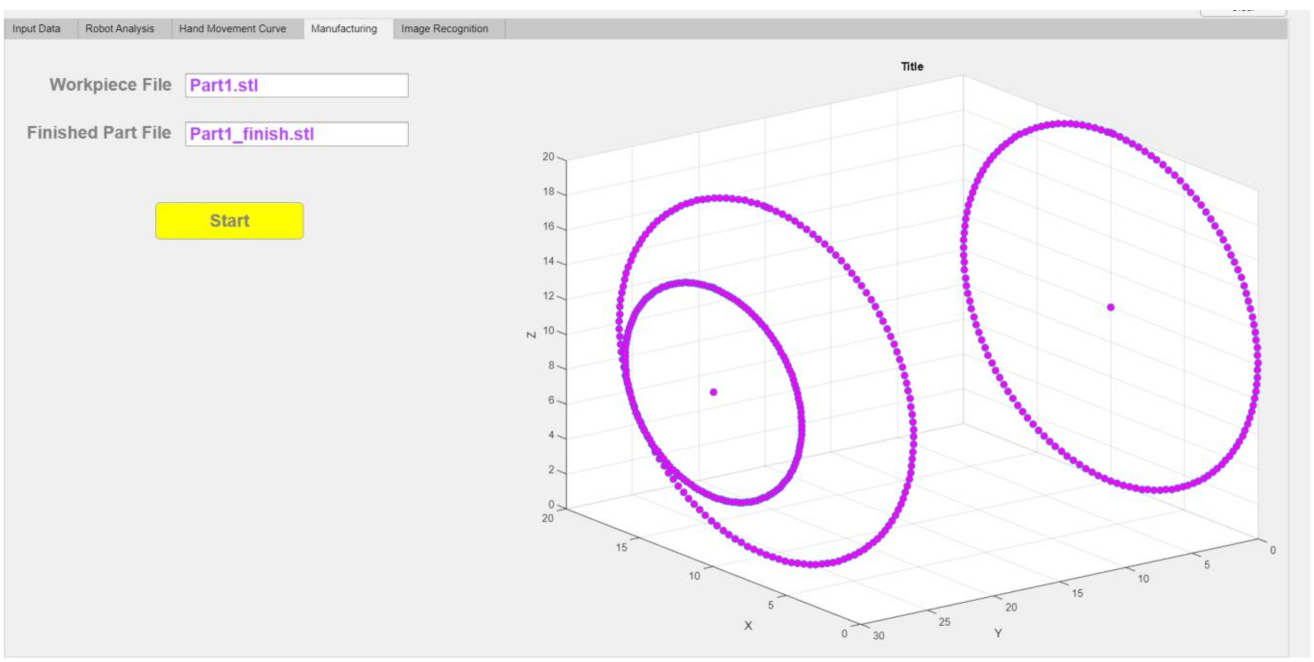
7.1. TRAJECTORY FROM REALTIME HAND MOVEMENT USING MOBILE SENSORS

1. Connect the Matlab with Mobile
2. Switch on the Sensors in the Mobile
3. Set the Sampling Rate
4. Obtain the Orientation Data from the mobile sensors
5. Store the orientation data for each millisecond
6. Simulate the robot using the data in Realtime by holding the mobile in hand and moving in space.



7.2. TRAJECTORY FROM 3D SOLID MODELS



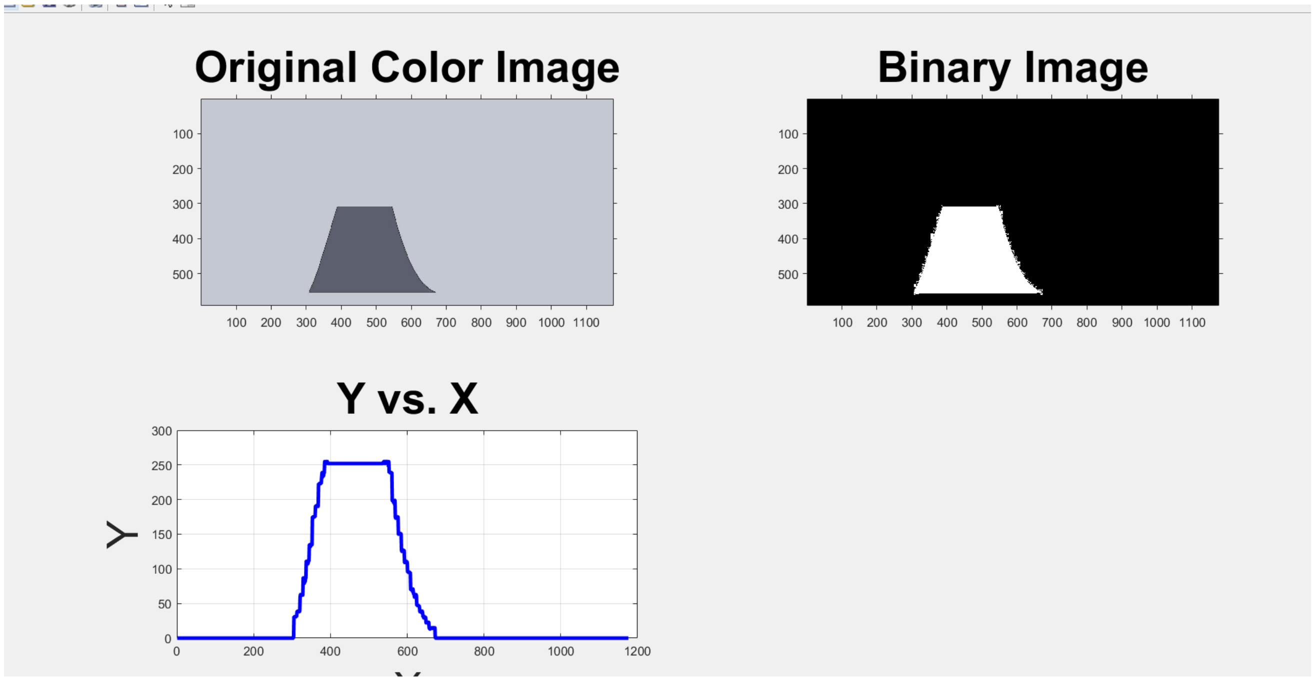


Workpiece File Part1.stl

Finished Part File Part1_finish.stl

7.3. TRAJECTORY FROM IMAGE RECOGNITION

1. Input the Image (jpg or png)
2. Convert the Color Image into its equivalent RGB
3. Extract the RGB channel
4. Convert into Binary Image
5. Use Canny Edge Detection Algorithm to detect the edges
6. Extract the coordinates of the edges and save it into matrix
7. Generate Trajectory from the extracted coordinates.



8. APPLICATIONS

1. Welding or Soldering
2. Additive Manufacturing
3. Geometric Shape Drawing
4. Other Manufacturing Processes
5. Reverse Engineering (Development of Models from 3D Scanning)
6. Artistic Sketching
7. Creation of Template profile for Laser sheet metal cutting

9. CONCLUSION

1. Simulation of Forward Kinematics, Inverse Kinematics, Statics, Dynamics equations studied in theory
2. Trajectory planning is very important task and proper smooth & continuous trajectory is needed for smooth operation of the robot to trace the trajectory.
3.Solving Inverse Kinematics equations requires high computation and is computationally expensive.
4. Trajectory generation in different ways was understood.

10. REFERENCE

1. Li, S., Jin, L., & Mirza, M. A. (2019). Zeroing Neural Networks for Robot Arm Motion Generation. *Kinematic Control of Redundant Robot Arms Using Neural Networks*, 147–165. https://doi.org/10.1002/9781119557005.ch9
2. Tirupati, I. I. T. (n.d.). *Wire Arc Additive Manufacturing of Pelton wheel bucket*.
3. Murakami, S., Takemoto, F., Fujimura, H., & Ide, E. (1989). Weld-line tracking control of arc welding robot using fuzzy logic controller. *Fuzzy Sets and Systems*, *32*(2), 221–237. https://doi.org/10.1016/0165-0114(89)90256-X
4. Deepak, B. B. V. L., Rao, C. A., & Raju, B. M. V. A. (2016). Weld seam tracking and simulation of 3-axis robotic arm for performing welding operation in cad environment. *Lecture Notes in Mechanical Engineering*, 405–415. https://doi.org/10.1007/978-81-322-2740-3_39
5. Ho, M. T., Rizal, Y., & Chu, L. M. (2013). Visual servoing tracking control of a ball and plate system: Design, implementation and experimental validation. *International Journal of Advanced Robotic Systems*, *10*. https://doi.org/10.5772/56525
6. Hub, D. (2021). *Using MATLAB and Arduino*. *Part 1*, 3–10.
7. Aggarwal, L., Aggarwal, K., & Urbanic, R. J. (2014). Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone(s). *Procedia CIRP*, *17*, 812–817. https://doi.org/10.1016/j.procir.2014.01.107
8. Jokić, D., Lubura, S., Rajs, V., Bodić, M., & Šiljak, H. (2020). Two open solutions for industrial robot control: The case of puma 560. *Electronics (Switzerland)*, *9*(6), 1–15. https://doi.org/10.3390/electronics9060972
9. Machine, P. U., & Scheinman, V. (1978). *-To identify the geometric relationship between input and output motions of a robot manipulator (viz . PUMA 560) -Know how a homegenous transformation matrix can be used to establish relationship between succesive links of a manipulator -Show how a kine*. *1*.
10. *Title Reinforcement learning for ball balancing using a robot manipulator Presenter Ari Biswas, Software Engineer, MathWorks*. (2020). *2017*, 7281.
11. Carter, T. J. (2009). *The Modeling of a Six Degree-of-freedom Industrial Robot for the Purpose of Efficient Path Planning*. *May*.
12. Wen Yu, & Ortiz, F. (2005). *Stability analysis of PD regulation for ball and beam system*. 517–522. https://doi.org/10.1109/cca.2005.1507178
13. Robot, U. P. (n.d.). *Software for Control and Dynamic Simulation of UNIMATE PUMA 560 ROBOT,*. *June 1993*.
14. Somasundar, A. V. S. S., Yedukondalu, G., & Shiva Kesavulu, K. (2019). Singularity analysis of kuka 6 dof robot for motion simulation. *International Journal of Mechanical and Production Engineering Research and Development*, *9*(2), 223–228. https://doi.org/10.24247/ijmperdapr201921

15. Raju, S. (2021). *Simulation of the forward kinematics of a 2R Robotic Arm* . 1–10.
16. Khan, K. A., & Ryu, J. C. (2017). ROS-based control of a manipulator arm for balancing a ball on a plate. *ASEE Annual Conference and Exposition, Conference Proceedings*, *2017-June*. https://doi.org/10.18260/1-2--28809
17. Corke, P. (2020). Robotics Toolbox. *Robotics Toolbox for Matlab*, *10*, 437. http://petercorke.com/Robotics_Toolbox.html
18. Findling, T. K. (2016). *Robotic Arm Tracing Curves Recognized by Camera by*.
19. Zhong, X. (1995). *Robot Calibration Using Artificial Neural Networks*. *November*.
20. Montague, P. R. (1999). Reinforcement Learning: An Introduction, by Sutton, R.S. and Barto, A.G. *Trends in Cognitive Sciences*, *3*(9), 360. https://doi.org/10.1016/s1364-6613(99)01331-5
21. Andrés, M. O. De. (n.d.). *Reinforcement Learning to improve 4-Finger-Gripper Manipulation*.
22. Piltan, F., Emamzadeh, S., Hivand, Z., Shahriyari, F., & Mirzaei, M. (2012). PUMA-560 Robot Manipulator Position Sliding Mode Control Methods Using MATLAB / SIMULINK and Their Integration into Graduate / Undergraduate Nonlinear Control , Robotics and MATLAB Courses. *International Journal of Robotic and Automation*, *6*(3), 106–150.
23. Kumari, P., Kumar, S., Agrwal, V. K., & Engineering, I. (2016). PUMA-560 Robot Manipulator Position Sliding Mode Control Methods Using MATLAB / SIMULINK. *International Journal of Scientific Development and Research*, *1*(5), 56–58.
24. Valluru, S. K., Singh, M., & Singh, S. (2017). Prototype design and analysis of controllers for one dimensional ball and beam system. *1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems, ICPEICES 2016*. https://doi.org/10.1109/ICPEICES.2016.7853133
25. Skoglund, A. (2009). Programming by Demonstration of Robot Manipulators. In *Technology*.
26. Alkamachi, A. (2019). Pole placement control of a ball and beam system a graphical user interface (GUI) approach. *ACM International Conference Proceeding Series*, *1*, 184–189. https://doi.org/10.1145/3321289.3321305
27. Carter, F. M., & Cherchas, D. B. (1999). Motion control of non-fixed base robotic manipulators. *Robotica*, *17*(2), 143–157. https://doi.org/10.1017/S0263574799001186
28. Bolívar-Vincenty, C. G., & Beauchamp-Báez. (2014). Modelling the Ball-and-Beam System From Newtonian Mechanics and from Lagrange Methods. *Twelfth LACCEI Latin American and Caribbean Conference for Engineering and Technology*, *1*, 1–9.
29. Dusek, F., Honc, D., & Sharma, K. R. (2017). Modelling of ball and plate system based on first principle model and optimal control. *Proceedings of the 2017 21st International Conference on Process Control, PC 2017*, 216–221. https://doi.org/10.1109/PC.2017.7976216
30. Colmenares, S. G., Moreno-Armendariz, M. A., Yu, W., & Ortiz Rodriguez, F. (2012). Modeling and nonlinear PD regulation for ball and plate system. *World Automation Congress Proceedings*, 1–6.
31. Salem, F. A. (2013). *Mechatronics Design of Ball and Beam System : Education and*. *3*(4), 1–27.
32. Prado, J. (2012). *Mechanic Development of a high-performance-6-DoF- biomechanical joint analysis system based on an industrial robot*. https://mediatum.ub.tum.de/doc/1095796/1095796.pdf

33. Kalyoncu, M. (2008). Mathematical modelling and dynamic response of a multi-straight-line path tracing flexible robot manipulator with rotating-prismatic joint. *Applied Mathematical Modelling*, *32*(6), 1087–1098. https://doi.org/10.1016/j.apm.2007.02.032
34. Online, M. (2021). *Manipulator Shape Tracing in MATLAB and Simulink*. *3*, 1–6.
35. Zhou, Y., Lin, J., Wang, S., & Zhang, C. (2021). Learning Ball-Balancing Robot through Deep Reinforcement Learning. *2021 International Conference on Computer, Control and Robotics, ICCCR 2021*, 1–8. https://doi.org/10.1109/ICCCR49711.2021.9349369
36. Findling, T. K., & Silaghi, M. C. (n.d.). *Laser Curve Tracing for Robotic Arms*.
37. Chati, S., & Patel, A. (2018). LabView implementation of ball and beam system using cascade PD control. *2018 2nd International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2018*, 1–5. https://doi.org/10.1109/IEMENTECH.2018.8465158
38. Pratap, D. (2013). Kineto-Elasto Dynamic Analysis of Robot Manipulator Puma-560. *IOSR Journal of Mechanical and Civil Engineering*, *8*(3), 33–40. https://doi.org/10.9790/1684-0833340
39. Mohammed, A. A., & Sunar, M. (2015). Kinematics modeling of a 4-DOF robotic arm. *Proceedings - 2015 International Conference on Control, Automation and Robotics, ICCAR 2015*, *May*, 87–91. https://doi.org/10.1109/ICCAR.2015.7166008
40. Moosavian, S. A. A., Pourreza, A., & Alipour, K. (2009). Kinematics and dynamics of a hybrid serial-parallel mobile robot. *Proceedings - IEEE International Conference on Robotics and Automation*, *June 2014*, 1358–1363. https://doi.org/10.1109/ROBOT.2009.5152746
41. *inverse-kinematics-and-control-of-a-7-dof-redundant-manipulator-based-on-the-closed-loop-algorithm.pdf*. (n.d.).
42. Singh, G., & Banga, V. K. (2020). Inverse kinematics Solution of PUMA by using ANFIS Technique. *International Journal of Innovative Technology and Exploring Engineering*, *9*(7), 847–850. https://doi.org/10.35940/ijitee.g5393.059720
43. Chen, N., & Parker, G. A. (1994). Inverse kinematic solution to a calibrated Puma 560 industrial robot. *Control Engineering Practice*, *2*(2), 239–245. https://doi.org/10.1016/0967-0661(94)90203-8
44. Jha, P. (2015). *Inverse Kinematic Analysis of Robot Manipulators A THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF Doctor of Philosophy IN INDUSTRIAL DESIGN*. *511*. http://ethesis.nitrkl.ac.in/6980/1/2015_Panchanand_Phd_511ID101.pdf
45. Shen, Y. (n.d.). *Introduction to Robotics Toolbox for MATLAB*.
46. Tiller, M. (2001). Introduction to Physical Modeling with Modelica. In *Introduction to Physical Modeling with Modelica*. https://doi.org/10.1007/978-1-4615-1561-6
47. Anjali, T., & Mathew, S. S. (2017). Implementation of optimal control for ball and beam system. *Proceedings of IEEE International Conference on Emerging Technological Trends in Computing, Communications and Electrical Engineering, ICETT 2016*, 3–7. https://doi.org/10.1109/ICETT.2016.7873763
48. Jubien, C. M., & Dimopoulos, N. J. (n.d.). *Identification of a PUMA-560 Two-Link Robot Using*. 568–572.

49. Vogg, G., Miskys, C. R., Garrido, J. A., Hermann, M., Eickhoff, M., & Stutzmann, M. (2004). High quality heteroepitaxial AIN films on diamond. In *Journal of Applied Physics* (Vol. 96, Issue 1). https://doi.org/10.1063/1.1759088
50. Kasula, A., Thakur, P., & Menon, M. K. (2018). GUI based Control Scheme for Ball-on-Plate System using Computer Vision. *2018 IEEE Western New York Image and Signal Processing Workshop, WNYISPW 2018*. https://doi.org/10.1109/WNYIPW.2018.8576461
51. Hwang, Y. K., Chen, P. C., Maciejewski, A. A., & Neidigk, D. D. (1994). Global motion planner for curve-tracing robots. *Proceedings - IEEE International Conference on Robotics and Automation*, *pt 1*, 662–667. https://doi.org/10.1109/robot.1994.351410
52. Kumar, A., & Kala, R. (2015). Geometric shape drawing using a 3 link planar manipulator. *2015 8th International Conference on Contemporary Computing, IC3 2015*, *August*, 404–409. https://doi.org/10.1109/IC3.2015.7346715
53. Trujillo, J. L. A., Pérez-Ruiz, A., & Serrezuela, R. R. (2017). Generation and Control of Basic Geometric Trajectories for a Robot Manipulator Using CompactRIO®. *Journal of Robotics*, *2017*. https://doi.org/10.1155/2017/7508787
54. Gamarra, D. F. T., & Cuadros, M. A. de S. L. (2015). Forward models for following a moving target with the puma 560 robot manipulator. *Inteligencia Artificial*, *18*(56), 31–42. https://doi.org/10.4114/ia.v18i56.1105
55. Singh, T. P., Suresh, P., & Chandan, S. (2017). Forward and Inverse Kinematic Analysis of Robotic Manipulators. *International Research Journal of Engineering and Technology(IRJET)*, *4*(2), 1459–1469. https://irjet.net/archives/V4/i2/IRJET-V4I2286.pdf
56. Sumega, M., Gorel, L., Varecha, P., Zossak, S., & Makys, P. (2018). Experimental study of ball on plate platform. *12th International Conference ELEKTRO 2018, 2018 ELEKTRO Conference Proceedings*, 1–5. https://doi.org/10.1109/ELEKTRO.2018.8398374
57. Dupuis, J. F., & Parizeau, M. (2006). Evolving a vision-based line-following robot controller. *Third Canadian Conference on Computer and Robot Vision, CRV 2006*, *2006*. https://doi.org/10.1109/CRV.2006.32
58. Saravanan, R., Ramabalan, S., & Balamurugan, C. (2008). Evolutionary optimal trajectory planning for industrial robot with payload constraints. *International Journal of Advanced Manufacturing Technology*, *38*(11–12), 1213–1226. https://doi.org/10.1007/s00170-007-1169-7
59. Ben-Ari, M., & Mondada, F. (2017). Elements of Robotics. In *Elements of Robotics*. https://doi.org/10.1007/978-3-319-62533-1
60. Galvan-Colmenares, S., Moreno-Armendáriz, M. A., Rubio, J. D. J., Ortíz-Rodriguez, F., Yu, W., & Aguilar-Ibáñez, C. F. (2014). Dual PD control regulation with nonlinear compensation for a ball and plate system. *Mathematical Problems in Engineering*, *2014*(April). https://doi.org/10.1155/2014/894209
61. Markovska, M. (n.d.). *Drawing robotic arm*.
62. Kaleli, A., Dumlu, A., Çorapsiz, M. F., & Erentürk, K. (2013). Detailed analysis of SCARA-type serial manipulator on a moving base with LabView. *International Journal of Advanced Robotic Systems*, *10*. https://doi.org/10.5772/56178
63. Talli, A., & V Meti, V. K. (2020). Design, simulation, and analysis of a 6-axis robot using robot visualization software. *IOP Conference Series: Materials Science and Engineering*, *872*(1). https://doi.org/10.1088/1757-899X/872/1/012040

64. Reddy, G. R., & Eranki, V. K. P. (2016). *Design and Structural Analysis of a Robotic Arm*. 101. http://www.diva-portal.org/smash/get/diva2:1068547/FULLTEXT02
65. Kumra, S., Saxena, R., & Mehta, S. (2012). Design and development of 6-DOF robotic arm controlled by Man Machine Interface. *2012 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2012, December 2012*. https://doi.org/10.1109/ICCIC.2012.6510243
66. Senapati, M. (2016). *Design and Control of an Articulated Robotic Arm using Visual Inspection for Replacement Activities Design and Control of an Articulated Robotic Arm using Visual Inspection for Replacement Activities National Institute of Technology Rourkela*.
67. Wronka, C. M., & Dunnigan, M. W. (2011). Derivation and analysis of a dynamic model of a robotic manipulator on a moving base. *Robotics and Autonomous Systems*, *59*(10), 758–769. https://doi.org/10.1016/j.robot.2011.05.010