الجمهورية الجزائرية الديمقراطية الشعبية
Algerian Democratic and Popular Republic

وزارةالتعليم العالي والبحث العلمي
Higher Education And Scientific Research Minister
جامعة سعد دحلب البليدة
SAAD DAHLEB University of Blida

كلية التكنولوجيا
Faculty Of Technology

قسم الإلكترونيك
Department of Electronics

Master Thesis In Electronics

Vision System and Robotics

By

Omar MORCELI

Khadidja BOUMDEL

Modeling ,Control and Identification of a Robot Manipulator

Vision Based Position Control Application

Supervisor : Pr.Boualem KAZED

2016-2017

بسم الله الرحمن الرحيم

ACKNOWLEDGEMENTS

This master's thesis is the result of a research work of several months. In preamble, we would like to thank all those who have supported us and who have contributed to the preparation of this brief

First of all the great thanks to our mighty god

A big thank you to our developer Mr. B. kazed, for his precious help and for the time he kindly dedicated us and for guiding and supporting us over the year. You have set an example of excellence as a researcher, mentor, instructor, and role model.

We would especially like to thank our amazing family for the love, support, and constant encouragement we have gotten over the years.

ملخص: ناقشنا في هذه المذكرة حل شامل لمشكل التحكم في ذراع الروبوت، أولا قمنا بالنمذجة الكينماتيكة و الديناميكية للذراع، ثم برمجنا طرق معتمدة في التحكم وكشف الأنظمة الخاصة بالأذرع الروبتية ذات خصائص ديناميكية مجهولة، والذي مكننا من التحكم في الموضع وكذا تتبع المسار المنشود، ثم أضفنا تحكم موضعي مبني على الرؤية الاصطناعية .

كلمات المفاتيح: ذراع الروبوت، ديناميكية، تحكم الاخطي،كشف الأنظمة.

Résumé: dans cette thèse on a discuté une solution complète de command des bras manipulateur, premièrement on a fait la modilisation de la cinematique et la dynamique du bras, apres on a implementé des methodes efficaces pour la commande et l'identification du bras manipulateur avec des parametres inconnus de la dynamique, ce qui nous a permis de commandé la position ainsi le suivi de la trajectoire, puis on a finalsé avec une commande de la position basé sur la vision.

Mots clés: Robot manipulateur, Dynamique, la command PID, Commande nonlinear, identification des systems.

Abstract : In this thesis we discussed a complete robot manipulator control solving problem, first we modeled the manipulator's kinematics and dynamics, then we implemented reliable methods for control and identification of a the robot manipulator with unknown dynamical parameters .which allows us to control position as well as trajectory tracking, and then we conducted a vision based position control.

Keywords: Robot manipulator, Dynamics, PID controller, nonlinear control, System identification.

Notations and Acronyms

Symbols and Operators

ξ : The relative pose of a frame with respect to a reference coordinate frame.

{A}: the coordinate frame A.

x_i: The x axis of the i^{th} frame.

y_i: The y axis of the i^{th} frame.

z_i: The z axis of the i^{th}frame.

o_i : The origine point of coordinate frame i.

$\hat{x}, \hat{y}, \hat{z}$: Unite vectors of the axes x, y, z

${}^{A}R_{B}$: 3x3 Orthonormal Rotation Matrix of frame B with respect to frame A.

${}^{A}T_{B}$: 4x4 homogeneous transformation matrix of frame B with respect to frame A.

q_i: The i^{th} angle of rotation in the case of a revolute joint.

Ai: The homogeneous transformation matrix that expresses the position and the orientation of $o_i\ x_i\ y_i z_i$ with respect to $o_{i-1}x_{i-1}\ y_{i-1}z_{i-1}$.

t_n^0 : The position vector of the end effector with respect to the inertial or base frame .

${}^{i}\tilde{P}$: 4X1 homogeneous position vector with respect to frame i.

θ_i: The angle of rotation around the x axis.

d_i: The sliding distance along the z axis.

a: The length of the common normal.

α: The angle about common normal, from old z axis to new z axis.

$\text{trans}_{z_i}(d_i)$: The translation matrix in the z axis with distance d_i with respect to the i^{th} coordinate frame.

$\text{Rot}_{z_i}(\theta i)$: The rotation matrix about the z axis with angle θi with respect to the i^{th} coordinate frame.

ω : The angular velocity

V : The linear velocity

J_ω: The derivative of the angular velocity or angular Jacobian matrix

J_v : The derivative of the Linear Velocity or linear Jacobian matrix

J: The robot manipulator Jacobian matrix.

M: Robot manipulator mass matrix.

C: Centrifugal and Coriolis forces matrix.

G : Gravity vector of a Robot manipulator.

F: Friction matrix of a Robot manipulator.

K: Kinetic energy.

U: Potential energy.

I : The inertia Tensor.

τ: The vector of motor torques.

b_{ijk}: Christoffel Symbol.

ρ : The mass density.

$\hat{\mathrm{p}}$: 3x3 skew matrix of 3x1 p position vector.

I_3 : 3x3 identity matrix.

g: The gravitational force.

f_c: Coulomb friction coefficient.

f_v: Viscous friction coefficient.

f_s: Stiction friction coefficient.

qs: Stribeck velocity.

Tf: Actuator friction term.

J: Actuator inertia.

r(t): The reference or the set point signal in the control loop.

e(t) : The error signal between the set point and the feedback in the control loop.

$\dot{\mathrm{e}}, \ddot{\mathrm{e}}$: The first and the second derivative of the error.

u(t): The control input signal to the system in the control loop.

y(t) : Output of the system in control loop.

$\mathrm{q_i}$: The i^{th} joint angle of the robot manipulator.

q: The vector of the joints angles of the robot manipulator.

$\mathrm{q_d}$: The vector of the desired joints angles of the robot manipulator.

$\dot{\mathrm{q}}, \ddot{\mathrm{q}}$: First and second derivative of joints angles vector of the robot manipulator.

Kp,Kd,Ki: PID controller gains.

Y: System output vector.

$\hat{Y}$: The estimated output vector.

Θ: The Vector of (unknown) parameters.

$\hat{\theta}$: The Vector of the estimated parameters.

ϕ_i: The Regression variables depend on the i^{th} parameter.

Φ: The regresseur matrix.

$V(\theta, t)$: The least squares cost function.

U_i: The PWM input to the i^{th} motor.

$f(x, y)$: 2d function represents a digital image.

α: Distance between the camera lens and the object.

β : Distance between the camera lens and the image plane.

f: The focal length.

P: Camera matrix.

E: Extrinsic camera matrix.

K: Intrinsic camera matrix.

Abbreviations and Acronyms

DOF: Degrees of freedom.

PID: Proportional Integral Derivative controller.

LS: least square.

RLS: recursive least squares.

RGB: Digital image of three color planes Red, Green and Blue.

HSV: Digital image transform to Hue, Saturation and Value space.

CHT: The circle Hough Transform.

PWM: Pulse-width modulation can generate an analogical voltage from digital output by switching between 0 and 1 with high frequency and with specific duty cycle.

DC: Direct courant.

USB: UNIVERSAL SERIAL BUS.

IO: INPUT/OUTPUT.

Te: Sampling time.

Fe: Sampling frequency.

Fb: lowpass Filter *cut*-off *frequency.*

RMS: Root Mean Square is the square root of the arithmetic mean of the squares of the values.

SVD: Singular Value Decomposition

Table of Contents

Page number

Introduction……..1

1. History and Motivation …… 2

2. The Eurobtec IR50p (or ROB3i) …… 3

3. Objectives …….. 4

4. Software ……... 5

4.1 MATLAB R2014a with SIMULINK ……..5

4.2 Arduino IO (MATLAB support package for Arduino)……..5

4.3 MATLAB Camera Calibration Toolbox……5

5 . Hardware ……..5

5.1 Arduino Mega2560board ……6

5.2 Dc Motors Driver Board……6

5.3 Webcam……..…7

6 . Outline ……..8

Chapter 1: Modeling ……9

1.1 Representing Position and Orientation……9

1.1.1 Representing Pose in 3-Dimensions……11

1.1.2 Representing Orientation in 3-Dimensions……12

a. Orthonormal Rotation Matrix……12

b. Three-Angle Representation……13

1.1.3 Combining Translation and Orientation……13

1.2 Forword Kinematics……13

1.2.1 Kinematic Chains……14

1.2.2 Denavit and Hartenberg Representation……17

a. Four parameters……18

b. Denavit-Hartenberg matrix……19

1.2.3 Application on the chosen robot……20

1.3 Differential Kinematics……22

1.3.1 Derivation of the Jacobian……22

a. Angular Velocity……22

b. Linear Velocity……23

c. Combining the Angular and Linear Jacobians……24

1.3.2 Application on the chosen robot ……24

1.4 Inverse kinematics ……26

1.4.1 Iterative Method……26

1.4.2 The geometric Approach……27

1.4.3 Application on the chosen robot ……27

1.5 Dynamics ……28

1.5.1 Lagrange-Euler formulations ……29

1.5.1.1 Inertial force……29

a. Mass matrix ……29

b. Centrifugal& Coriolis Forces ……31

c. Inertia matrix ……32

1.5.1.2 Gravity term (Potential Energy) ……33

1.5.1.3 Friction modeling……34

1.5.2 Actuator Dynamics ……36

1.5.3 Application on the chosen robot……36

Chapter 2: Control and System Identification……45

2.1 Manipulator Control ……45

2.1.1 PID controller……47

a. Implementation and simulation……49

2.1.2 Computed Torque Control (trajectory tracking)…… ……50

a. Feedback Linearization……51

b. Implementation and simulation ……53

2.2 identification and parameters estimation ……56

2.2.1 Introduction ……56

2.2.2 Nonparametric Models……57

2.2.3 Parametric Models……58

a. the least squares method LS……58

b. the recursive least squares method RLS……60

2.2.4 Applying the LS on the chosen robot ……62

Chapter 3: Vision ……65

3.1 Image Processing ……65

3.1.1 The image ……65

3.1.2 RGB color model……66

3.1.3 HSV color model……67

3.1.4 Thresholding……68

3.1.5 Edge detection……69

a. Canny Edge detector……69

3.1.6 Hough transform……70

a. Hough transform for circle detecti……70

3.2Vision system……72

3.2.1 Camera System……72

3.2.2 Camera Modeling and Calibration……73

3.2.2.1 Camera Model……73

a. Pinhole Camera Model……73

b. Distortion in Camera ……75

b.1 Radial distortion……75

b.2 Tangential Distortion……75

3.2.2.2 Camera Calibration Parameters……76

a. Extrinsic Parameters……76

b. Intrinsic Parameters……..76

Chapter 4 : Implementation and practical results……77

4.1 Manipulator Setup……77

4.1.1 Hardware ……77

4.1.2 Software……78

4.1.3 Filtering ……79

4.2 independent Joint Control (PID Position Control)……80

4.2.1 Design and concept ……80

4.2.2 Tuning and results……82

4.3 Parameter Estimation ……84

4.3.1 The Exciting Trajectory……85

4.3.2 Applying the LS method……86

4.3.3 Parameters validation……86

4.4 Computed toque control (Trajectory tracking) ……88

4.4.1 Design and concept ……88

4.4.2 Tuning and results……89

4.5 Vision Control application……91

4.5.1 Object detection……93

4.5.2 Camera calibration and pose estimation……94

4.5.3 Vision position control……95

6. Conclusion ……98

Bibliography ……99

Figures List

- Figure 1: CAD model of IR50P robot manipulator.
- Figure 2: Arduino Mega 2560
- Figure 3: Power Board
- Figure 4: Webcam
- Figure (1.1a): the displacement of the point P with respect to a coordinate frame .
- Figure (1.1b): the position and orientation of the object by the position and orientation of its coordinate frame.
- Figure (1.2): two frames {*A*} and {*B*} and the relative pose A_{ξ_B} which describes {*B*} with respect to {*A*}.
- Figure (1.3):Two 3D coordinate frames {*A*} and {*B*}. {*B*} is rotated and translated with respect to {*A*}.
- Figure (1.4):Symbolic representation of robot joint.
- Figure (1.5):Links, joints , coordinate frames and transformation vector on an 3 DOF elbow manipulator.
- Figure (1.6):DH Parameters, Joints axis and common normal representation.
- Figure (1.7):Links, joints , coordinate frames and transformation vector on an 3 DOF elbow manipulator.
- Figure (1.8): angels and distances of a 3 DOF elbow manipulator.
- Figure (1.9):Friction model with coulomb ,viscous and stiction term.
- Figure (1.10):angels and distances of center of masses of a 3 DOF elbow manipulator.
- Figure(2.1):A block diagram of a PID controller in a feedback loop.
- Figure (2.2):A block diagram of independent joint control for n joints manipulator.
- Figure (2.3): Simulink model of 3 DOF manipulator dynamics.

- Figure (2.4): Simulink model of independent joint control for a 3 DOF manipulator dynamics.
- Figure(2.5): simulation result of an independent joint control for a 3 DOF manipulator.
- Figure (2.6):A block diagram of Computed-torque control.
- Figure(2.7) : Simulink model of computed torque control for a 3 DOF manipulator dynamics.
- Figure (2.8) :Simulink model of computed torque control for a 3 DOF manipulator dynamics .
- Figure(2.9): An open-loop system.
- Figure (2.10): A closed-loop system.
- Figure (2.11): Step response of a first order process with delay.
- Figure (3.1): a grayscale image and its matrix of intensity level values.
- Figure (3.2): RGB image and its 3 colors plane, red, green and red plane.
- Figure (3.3): The RGB color model mapped to a cub.
- Figure (3.4): The HSV color model cone describe the Hue, Saturation and Value ranges.
- Figure (3.5): describe transformation from RGB to HSV then getting a Binary image by thresholding the HSV image.
- Figure (3.6): The original image on the left and the detected edges on the right using Canny edge detection algorithm.
- Figure (3.7): an edge image of a circle in the left and its Hough transform space for 4 point in the right.
- Figure (3.8) : Illustration of the lens and object distances and focal lengths.
- Figure (3.9): Illustration of the camera model from the object to the 2d image.

- Figure (3.10): Illustration of the camera model showing the transformation by the Extrinsic parameters and the mapping into the image plane using the intrinsic parameters.
- Figure (3.11): radial distortion of 3 type of lens.
- Figure (3.12): illustration of Tangential Distortion of a lens.
- Figure (3.13): Diagram describe the transformation from the World coordinates into Pixel coordinates passing by Extrinsic and intrinsic camera parameters matrices .
- Figure (4.1a): Hardware Setup.
- Figure (4.1b) The previously discussed manipulator workbench.
- Figure(4.2): Simulink model of driving a Dc Motor and measuring position from its sensor.
- Figure (4.3): Lowpass Filter frequency response with 300Hz cut-off frequency.
- Figure (4.4): joint measured signal and the filtered one.
- Figure (4.5): Simulink model of independent joints positions control following a Cartesian space trajectory.
- Figure (4.6): *Simulink model of PID controller for a single joint with supervision of all the control loop signal values*
- Figure(4.7): Simulink Subsystem model of a Dc motor command with stopping criteria and joint sensor measurement.
- Figure(4.8): Angles responses for 3 joints using PID controllers.
- Figure(4.9): end effector position signals (X,Y,Z) following a desired trajectory .
- Figure(4.10): three joint angles signals and their inputs PWM signals
- Figure (4.11): Validation Signals for the three joints manipulator represented by the measured and the estimated output signals.
- Figure (4.12): *Simulink model of Computed torque control for three joints manipulator using Dynamics inversion and 3 PD controller and supervision of all the loop signals values.*

- Figure (4.13): joints Angles Responses after using a computed torque control for tracking desired trajectories
- Figure (4.14): Vision Control for diagram a Robot Manipulator
- Figure (4.15): illustration of robot manipulator ,camera and object coordinate frames.
- Figure (4.16): circle detection after thresholding applying circle Hough transform.
- Figure (4.17) examples of several checkerboard photos that are used in camera calibration.
- Figure (4.18): MATLAB camera calibrator app showing the world coordinate frame after calibrating the camera and showing the mean error of the estimation.
- Figure (4.19): examples of real-time position estimation using MATLAB and calibrated camera.
- Figure 4.20 end effector position following a desired position feeding from camera vision.

Tables List

- Table 1.1: DH parameters for 3-link Elbow manipulator.
- Table 2.1 : constant Setpoint and Parameters for PID control of the first joint
- Table 2.2 : constant Setpoint and Parameters for PID control of the second joint
- Table 2.3 : constant Setpoint and Parameters for PID control of the third joint
- Table 2.4 : the PID Parameters for control of the first joint
- Table 2.5 : the PID Parameters for control of the second joint
- Table 2.6 : the PID Parameters for control of the third joint.
- Table 4.1 : the PID gains of the first joint controller.
- Table 4.2 : the PID gains of the second joint controller.
- Table 4.3 : the PID gains of the third joint controller.
- Table 4.4 :RMS error and the fit between the measured and the estimated outputs for the three joint.
- Table 4.5 : the PID gains of the first joint controller.
- Table 4.6 : the PID gains of the second joint controller.
- Table 4.7: the PID gains of the third joint controller.
- Table 4.8 : real positions of the ball compared with the estimated positions.

Introduction

The manipulator robot has become a necessity because the industry handles heavy objects repetitively and in hazardous environments. All the research has led to arms of all sizes and weights, of all speeds and precisions, and adapted to the tasks entrusted.

The appearance of the manipulator robots is pulsated by the fact of an era where the manufacturing is back in chain, which requires a repetitive and painful working time. In recent decades, more complex tasks requiring displacements in environments not allowed to human beings (nuclear, mine, military, space, etc.) have favored the Mobile robot to settle.

The rapid development of the industry has invoked the improvement of robots manipulators. Then the robot must handle with increasing speeds and precisions. This requires more appropriate mechanical structures but also better new control techniques.

The researchers did not cease these last two decades to investigate the different axes that deal with robotics. This research is distinguished by the different angles with which are addressed the preoccupations with robotics that can be classified as follows:

a. Modeling:

Most of the time the robot model (Manipulator or mobile), Is necessary to perform a command for example , But sometimes the model can be non-linear and couplet, even with variable parameters, Which requires orienting towards nonlinear

modeling and identification methods instead of being satisfied with the Lagrange and Euler formalisms for the dynamic model.

b. Scheduling tasks:

These are strategies that manage a set of operations that make up a task. We are talking about the coordination between these operations and their realization. The Planning is global if you have all the information about the environment and often static. The planning is local if the movement cannot be foreseen in advance, which requires reactive reflexes.

c. Trajectory generation:

The trajectory generation that must be traveled by a robot and carried out off line or online.

d. The command :

This is the step that generates the control signal to send to the robot's shareholders to ensure the trajectory to be followed. Among these commands, the PID command, the adaptive control, linear feedback control, Robust control.

e. The identification :

Conventional methods such as least squares are used if the system is assumed to be linear in the parameters, if it’s not the case, thus using non-linear approximation methods. [1]

1. History and Motivation :

The English term robot was derived from the Czech word robota that means executive labor, and was first introduced by the Czech playwright Karle Capek in his 1921 play Rossum's Universal Robots. Since then the term has been applied to virtually anything that operates with some degree of autonomy, usually under computer control. An official definition of the term, dated to 1980, comes from the Robot Institute of America (RIA) and reflects to days status of robotics technology:

"A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks." [2]

In the early 1980's, robot manipulators were touted as the ultimate solution to automated manufacturing. Predictions were that entire factories of the future would require few, if any, human operators. It turned out that these predictions were a little exaggerated, as the savings in labor costs often did not outweigh the development costs of creating robot systems. Quite simply, people are good at what they do, and installing a robot involves complex systems integration problems. As a result, robotics fell out of favor in the late 1980's.

A resurgence of interest in robotics can be witnessed in the recent years. Deeper understanding of the subject and new technology have made it possible for robots to explore the surface on Mars, locate sunken ships, searching out land mines, and finding victims in collapsed buildings. In an industrial environment the advantages of robots are reduction of manufacturing costs, increase of productivity, improvement of quality standards, and the possibility of eliminating harmful tasks for human operator.[3]

2. TheEurobtec IR50p(or ROB3i) :

The IR50p (or ROB3i) is a robot manipulator manufactured by Eurobtec, it's a 5 DOF manipulator, each joint has a DC motor that operates in a nominal voltage of 24V, and a potionmeter to measure the joint's angle.

Originally,IR50p has an integrated controller in the base of the robot, it's easy and fast to install and implement but it's an old technology, doesn't include such a trajectory tracking algorithms, as well intelligence represented by vision, as well data supervision, and speed precision ratio perspective.

Our mission consists of removing all the hardware and electronics except the arm (motors and Potionmeters), and implement improved nonlinear controller

design solution as well, estimate accurate IR50pmanipulatorparameters, then we'll add an intelligence represented by vision control.

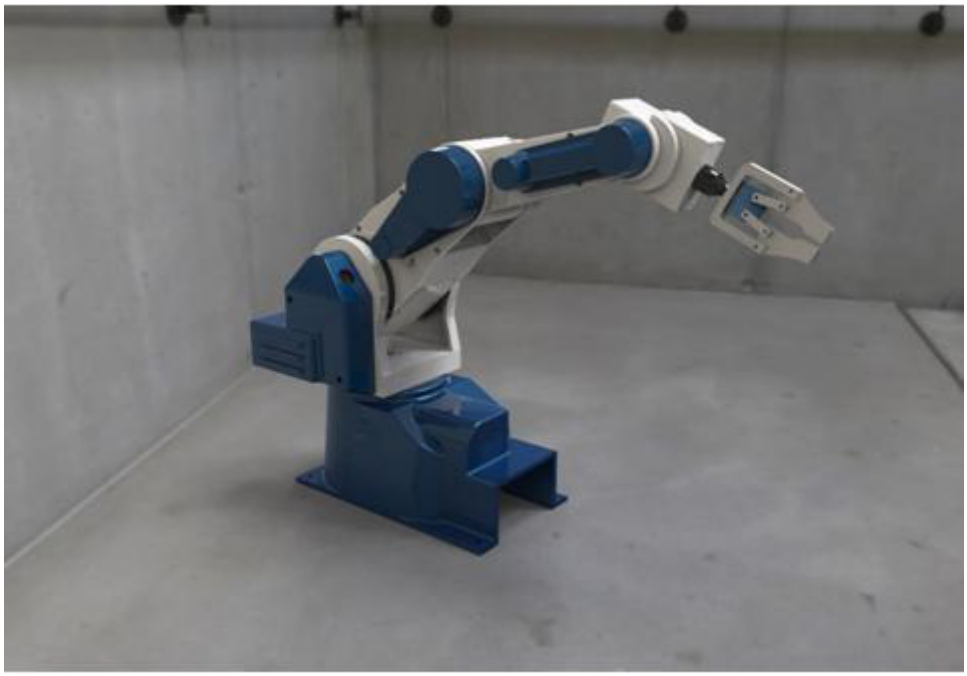


Figure 1: CAD model of IR50P robot manipulator

3. Objectives :

Our objective is improving the control of the first 3 DOF (Degrees of freedom) of the robot using an Arduino board and Simulink, this makes the computing more responsive, stable and efficient. This allows us to supervise all feedback, signals, positions and joint angles in real-time. Such a complex control problem would be better solved by dividing it into mini objectives as follows:

1 . Kinematics and dynamics modeling of the 3 DOF robot manipulator.
2. Simulate the model, design different control methods.
3. Design a PID controller for the 3 DOF manipulator for position control.
4. Complete our robot dynamical model by estimating reliable parameters of the robot.

5. Appling computed torque control using the estimated parameters for trajectory tracking.
6. Estimate the position of an object using a camera, the arm has to reach the object which required: camera modeling and calibration, image processing for color and shape detection of an object.

4. Software:

Four computer programs have been used to solve the thesis assignment. Following is a short description of this software and the using area.

4.1 MATLAB R2014a with SIMULINK:

MATLAB [4] is developed by MathWorks, and is a high-level language and numerical computing environment for performing computationally intensive tasks faster than traditional programming languages. It solves tight integration and mathematical problems with other MathWorks products, among them SIMULINK [5] which is an environment for multi domain simulation and Model-Based Design for dynamic and embedded systems. MATLAB and SIMULINK have been used to simulate the dynamic model for The IR50p, and to present the results graphically.

4.2 Arduino IO (MATLAB support package for Arduino) :

MATLAB support package for Arduino *i*s a MATLAB class and Simulink blocks for communicating with an Arduino microcontroller board.

It also has a specific code for Arduino Hardware that enables the serial communication with SIMULINK. We can read data from sensors, write and generate signals through the Arduino board, and immediately see the results in SIMULINK without having to compile. [6]

4.3 MATLAB Camera Calibration Toolbox :

We have used this toolbox to estimate the parameters of the camera. We can use these parameters to correct for lens distortion, measure the size of an object in real world units, as well determine the location of the camera in the scene.

5 Hardware:

For this part we have used three principal electronics' tools:

5.1 Arduino Mega2560board:

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

We have chosen the Arduino Mega 2560 board as it is based on the ATmega2560 microcontroller shown in figure (2), it operates at 16 MHz to control all of the onboard functions as well sending the data over serial communication to SIMULINK in order to be plotted and interpreted.

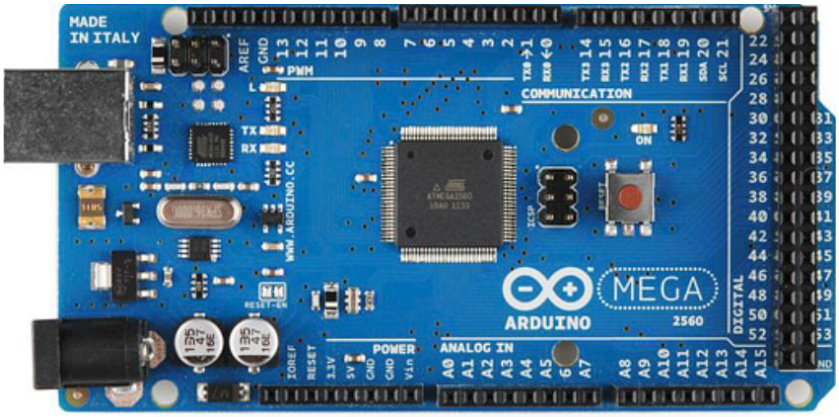


Figure 2: Arduino Mega 2560

5.2 Dc Motors Driver Board:

A power board is intended to distribute power at the desired dose to electrical components including sensors and actuators, in our case we have the Dc Motors Driver board used to control the three dc motors according to the command signals delivered by the Arduino mega board.

The power board has 3 H-bridges related to the motors, and it's powered by a 24V power source, and the Arduino mega controls this driver board in order to decide:

- The power distributed to the motor, which will make it turn more or less quickly according to the PWM signal from the Arduino.
- The direction of the voltage to be applied to the motor, which will make it turn in one direction or another, this can be done using the H-bridges.

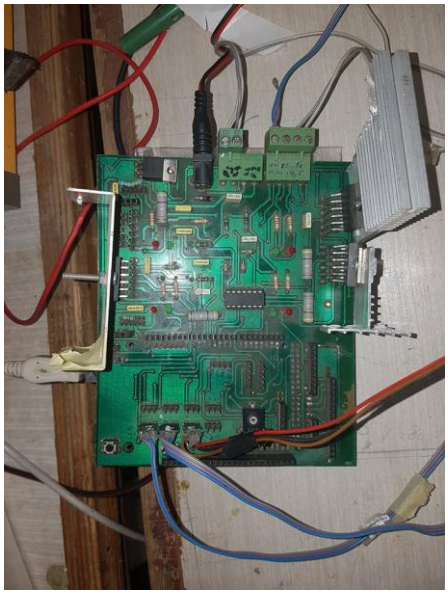


Figure 3 Power Board

5.3 Webcam:

In vision application, we used an arbitrary webcam with a quality of 640X480.

Figure 4 Webcam

6. Outline

- Chapter 1 discusses the mathematical model of the IR50p robot manipulator, which is concluded using the kinematics and dynamical modeling.

- Chapter 2 discusses different control theories of the manipulator such PID, independent joint control and computed torque control and simulate each of them, and also discusses System identification methods and the way to apply it on robot manipulators.
- Chapter 3 discusses computer vision theory and image processing tools that are used to detect and position an object, and then discusses the camera model which can transfer the position of the object from the digital image position into the real world coordinate frame.
- Chapter 4 discusses the obtained practical results after applying the PID controller as well, a comparison between the estimated manipulator model and the real one, results obtained after applying computed torque controller using the estimated parameters, and finally shows the results obtained from vision control application.

Chapter 1: Modeling

1.1 Representing Position and Orientation:

A fundamental requirement in robotics and computer vision is to represent the position and orientation of objects in an environment. Such objects include robots, cameras, work pieces, obstacles and paths.

A point in space is a familiar concept from mathematics and can be described by a coordinate vector, also known as a bound vector, as shown in Figure 1.1.a the vector represents the displacement of the point with respect to some reference coordinate frame. A coordinate frame, or Cartesian coordinate system, is a set of orthogonal axes which intersect at a point known as the origin.

More frequently we need to consider a set of points that comprise some object. We assume that the object is rigid and that its constituent points maintain a constant relative position with respect to the object's coordinate frame as shown in Figure 1.1.b Instead of describing the individual points we describe the position and orientation of the object by the position and orientation of its coordinate frame. A coordinate frame is labeled, {B} in this case, and its axis labels $\mathrm{x_B}$ and $\mathrm{y_B}$ adopt the frame's label as their subscript.

The position and orientation of a coordinate frame is known as its pose and is shown graphically as a set of coordinate axes. The relative pose of a frame with respect to reference coordinate frame, is denoted by the symbol ξ pronounced ksi.

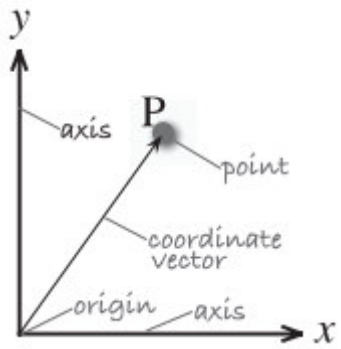


Figure (1.1a): the displacement of the point P with respect to a coordinate frame

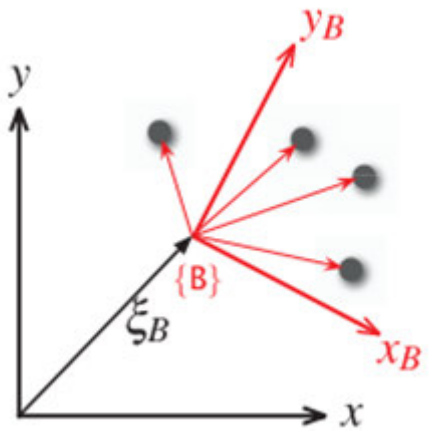
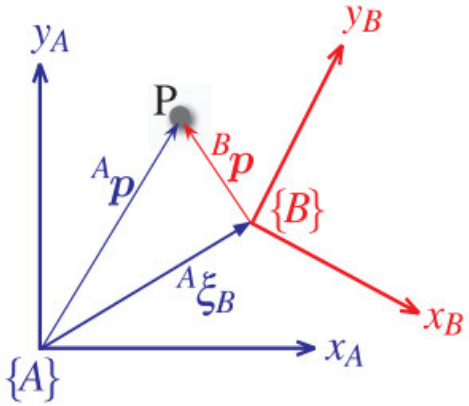


Figure (1.1.b): the position and orientation of the object by the position and orientation of its coordinate frame.



Figure(1.2) : two frames {*A*} and {*B*} and the relative pose ${}^{A}\xi_{B}$ which describes {*B*} with respect to {*A*}

Figure 1.2 shows two frames {A} and {B} and the relative pose ,which describes {B} with respect to {A}.The point P can be described with respect to either coordinate frame. Formally we express this as:

$$ {}^{A}P = {}^{A}\xi_{B} \cdot {}^{B}P \qquad (1.1) $$

1.1.1 Representing Pose in 3-Dimensions:

The 3-dimensional case is an extension of the 2-dimensional case and we add an extra coordinate axis, typically denoted by z, which is orthogonal to both the x- and y-axes.

The point P is represented by its x-, y- and z-coordinates (x, y, z) or as a bound vector

$$ P = x\,\hat{x} + y\,\hat{y} + z\,\hat{z} \qquad (1.2) $$

With $\hat{x} = [1,0,0]^{t}$

Figure 1.1.a shows a coordinate frame {B} that we wish to describe with respect to the reference frame {A}. We can see clearly that the origin of {B} has been displaced by the vector t = (x, y, z) and then rotated in some complex fashion. The way we represent orientation is very important. Our approach is to consider an arbitrary point P with respect to each of the coordinate frames and to determine the relationship between, and we will consider the problem in two parts: rotation and then translation. Rotation is surprisingly complex for the 3-dimensional case and we devote all of the next section to it.

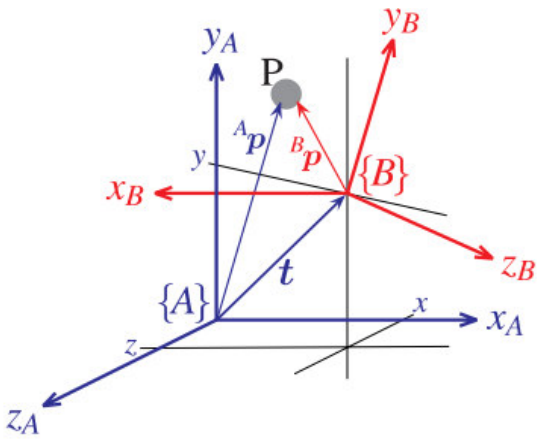


Figure (1.3): Two 3D coordinate frames {*A*} and {*B*}. {*B*} is rotated and translated with respect to {*A*}

1.1.2 Representing Orientation in 3-Dimensions:

a. Orthonormal Rotation Matrix:

We can represent the orientation of a coordinate frame by its unit vectors expressed in terms of the reference coordinate frame. Each unit vector has three elements and they form the columns of a 3×3 orthonormal matrix ${}^{A}R_{B}$

$$\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = {}^{A}R_{B} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \qquad (1.3)$$

The orthonormal rotation matrices for rotation of θ about the x, y and z axes are

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad (1.4)$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \qquad (1.5)$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (1.6)$$

b. Three-Angle Representations:

Euler's rotation theorem requires successive rotation about three axes such that no two successive rotations are about the same axis. There are two classes of rotation sequence: Eulerian and Cardanian, named after Euler and Cardano respectively.

The Eulerian type involves repetition, but not successive, of rotations about one particular axis: XYX, XZX, YXY, YZY, ZXZ, or ZYZ. The Cardanian type is characterized by rotations about all three axes: XYZ, XZY, YZX, YXZ, ZXY, or ZYX. In common usage all these sequences are called Euler angles and there are a total of twelve to choose from.

The XYZ sequence is commonly used in aeronautics and mechanical dynamics and Robotics. which represent the rotations about φ, θ and ψ ,which known as roll, pitch and yaw angles.

$$R = R_x(\varphi)R_y(\theta)R_z(\psi) \qquad (1.7)$$

1.1.3 Combining Translation and Orientation:

Alternatively we can use a homogeneous transformation matrix to describe rotation and translation

$$\begin{bmatrix} {}^A\boldsymbol{x} \\ {}^A\boldsymbol{y} \\ {}^A\boldsymbol{z} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} {}^A\boldsymbol{R}_B & \boldsymbol{t} \\ \mathbf{0}_{1*3} & \mathbf{1} \end{bmatrix} \begin{bmatrix} {}^B\boldsymbol{x} \\ {}^B\boldsymbol{y} \\ {}^B\boldsymbol{z} \\ \mathbf{1} \end{bmatrix} \qquad (1.8)$$

The Cartesian translation vector between the origins of the coordinates frames {A} and {B} is $\boldsymbol{t}$, and the change in orientation is represented by a 3 × 3 orthonormal submatrix R. The vectors are expressed in homogenous form and we write

$$^{A}\widetilde{\boldsymbol{P}} = \begin{bmatrix} ^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \boldsymbol{0}_{1*3} & \boldsymbol{1} \end{bmatrix} {}^{B}\widetilde{\boldsymbol{P}} = {}^{A}\boldsymbol{T}_{B}\ {}^{B}\widetilde{\boldsymbol{P}} \qquad (1.9)$$

with $^{A}\boldsymbol{T}_{B}$ is a 4 × 4 homogeneous transformation. The matrix has a very specific structure and belongs to the special Euclidean group of dimension 3. [7]

1.2 Forward Kinematics:

In this section we develop the forward or configuration kinematic equations for rigid robots. The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector.

Stated more formally, the forward kinematics problem is to determine the position and orientation of the end-effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints. The forward kinematics problem is to be contrasted with the inverse kinematics problem, which will be studied in the next chapter, and which is concerned with determining values for the joint variables that achieve a desired position and orientation for the end-effector of the robot.

1.2.1 Kinematic Chains:

A robot manipulator is composed of a set of links connected together by various joints. The joints can either be very simple, such as a revolute joint or a prismatic joint, or else they can be more complex.

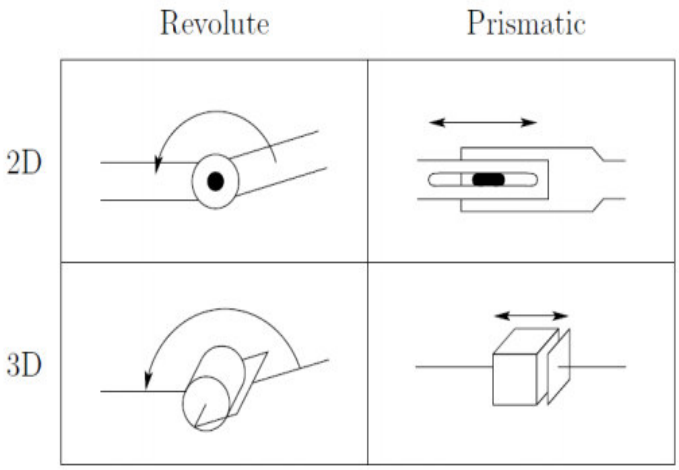


Figure (1.4): Symbolic representation of robot joint

With the assumption that each joint has a single degree-of-freedom, the action of each joint can be described by a single real number: the angle of rotation in the case of a revolute joint or the displacement in the case of a prismatic joint. The objective of forward kinematic analysis is to determine the cumulative effect of the entire set of joint variables

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. Therefore, link 0 (the first link) is fixed, and does not move when the joints are actuated.

With the i^{th} joint, we associate a joint variable, denoted by q_i. In the case of a revolute joint, $q_i$is the angle of rotation and in the case of a prismatic joint, $q_i$is the joint displacement:

$$q_i = \begin{Bmatrix} \theta_i : \text{ joint i revolute} \\ d_i : \text{joint i prismatic} \end{Bmatrix}$$

To perform the kinematic analysis, we rigidly attach a coordinate frame to each link .In particular, we attach $o_i x_i y_i z_i$ to link i. This means that, whatever motion the robot executes, the coordinates of each point on link i are constant when expressed in the i^{th} coordinate frame. Furthermore, 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 inertial frame.

Now suppose Ai is the homogeneous transformation matrix that expresses the position and orientation of $o_i x_i y_i z_i$ with respect to $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$.

The matrix Ai is not constant, but varies as the configuration of the robot is changed. However, the assumption that all points are either revolute or prismatic means that Ai is a function of only a single joint variable, namely qi. In other words,

$$A_i = A_i(q_i) \qquad (1.10)$$

Now the homogeneous transformation matrix that expresses the position and orientation of $o_j x_j y_j z_j$ with respect to $o_i x_i y_i z_i$ is called, by convention, a transformation matrix, and it is denoted by T_j^i

$$T_j^i = A_{i+1} A_{i+2} \ldots A_{j-1} \quad \text{if } i<j \qquad (1.11)$$

$$T_j^i = I \quad \text{if } i=j \qquad (1.12)$$

$$T_j^i = \left(T_j^i\right)^{-1} \text{if } i>j \qquad (1.13)$$

By the manner in which we have rigidly attached the various frames to the corresponding links, it follows that the position of any point on the end-effector, when expressed in frame n, is a constant independent of the configuration of the robot. Denote the position and orientation of the end-effector with respect to the inertial or base frame by a three-vector t_n^0 (which gives the coordinates of the origin of the end-effector frame with respect to the base frame) and the 3 × 3 rotation matrix R_n^0, and define the homogeneous transformation matrix

$$T_n^0 = \begin{bmatrix} R_n^0 & t_n^0 \\ 0 & 1 \end{bmatrix} \qquad (1.14)$$

Then the position and orientation of the end-effector in the inertial frame are given by

$$T_n^0 = A_1(q_1) \dots A_n(q_n) \qquad (1.15)$$

Each homogeneous transformation Ai is of the form

$$A_i = \begin{bmatrix} R_i^{i-1} & t_i^{i-1} \\ 0 & 1 \end{bmatrix} \qquad (1.16)$$

Hence

$$T_j^i = A_{i+1} \dots A_j = \begin{bmatrix} R_j^i & t_j^i \\ 0 & 1 \end{bmatrix} \qquad (1.17)$$

The matrix R_j^i expresses the orientation of $o_j x_j y_j z_j$ relative to $o_i x_i y_i z_i$ and is given by the rotational parts of the A-matrices as

$$R_j^i = R_{i+1}^i \dots R_j^{j-1} \qquad (1.18)$$

The coordinate vectors o_j^i are given recursively by the formula

$$t_j^i = t_{j-1}^i + R_{j-1}^i t_j^{j-1} \qquad (1.19)$$

These expressions will be useful when we study Jacobian matrices . [8]

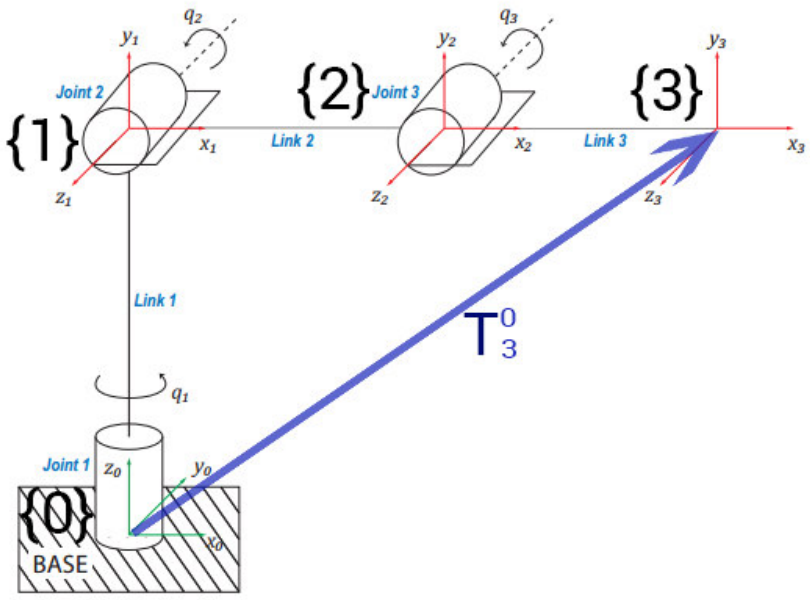


Figure (1.5): Links, joints , coordinate frames and transformation vector on an 3 DOF elbow manipulator

1.2.2 Denavit and Hartenberg Representation:

A commonly used convention for selecting frames of reference in robotics applications is the Denavit and Hartenberg (D–H) convention which was introduced by Jacques Denavit and Richard S. Hartenberg. In this convention, coordinate frames are attached to the joints between two links such that one transformation is associated with the joint [Z], and the second is associated with the link [X]. The coordinate transformations along a serial robot consisting of n links form the kinematics equations of the robot,

$$T_n^0=[Z_1][X_1][Z_2][X_2]\ldots.[X_{n-1}][Z_n] \qquad (1.20)$$

In order to determine the coordinate transformations [Z] and [X], the joints connecting the links are modeled as either hinged or sliding joints, each of which have a unique line S in space that forms the joint axis and define the relative movement of the two links. A typical serial robot is characterized by a sequence of six lines S_i, i=1,...,6, one for each joint in the robot. For each sequence of lines S_i and S_{i+1}, there is a common normal line $A_{i,i+1}$. The system of six joint axes S_i and five common normal lines $A_{i,i+1}$ form the kinematic skeleton of the typical six degree of freedom serial robot. Denavit and Hartenberg introduced the convention that Z coordinate axes are assigned to the joint axes S_i and X coordinate axes are assigned to the common normal's$A_{i,i+1}$.

This convention allows the definition of the movement of links around a common joint axis S_i by the screw displacement,

$$[Z_i]=\begin{bmatrix}\cos\theta i & -\sin\theta i & 0 & 0\\ \sin\theta i & \cos\theta i & 0 & 0\\ 0 & 0 & 1 & di\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (1.21)$$

where θ_i is the rotation around the X and d_i is the slide along the Z axis either of the parameters can be constants depending on the structure of the robot. Under this convention the dimensions of each link in the serial chain are defined by the screw displacement around the common normal $A_{i,i+1}$ from the joint S_i to S_{i+1}, which is given by

$$[X_i]=\begin{bmatrix}1 & 0 & 0 & a_i\\ 0 & \cos\alpha_{i,i+1} & -\sin\alpha_{i,i+1} & 0\\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (1.22)$$

Here $\alpha_{i,i+1}$ and $a_{i,i+1}$ define the physical dimensions of the link in terms of the angle measured around and distance measured along the X axis. [9]

a. Four parameters

The following four transformation parameters are known as D–H parameters[10]

d: offset along previous z to the common normal.

θ: angle about previous z, from old x to new x

a: length of the common normal .

α: angle about common normal, from old z axis to new z axis

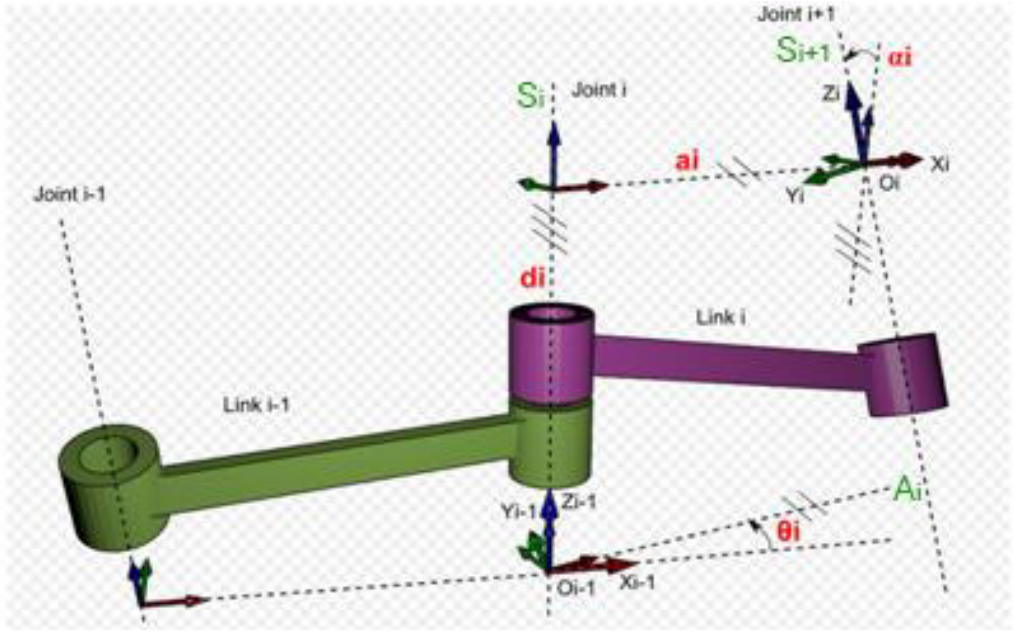


Figure (1.6):DH Parameters, Joints axis and common normal representation

b. Denavit-Hartenberg matrix

It is common to separate a screw displacement into the product of a pure translation along a line and a pure rotation about the line, so that

$$[Z_i] = \text{trans}_{z_i}(d_i)\text{Rot}_{z_i}(\theta i) \qquad (1.23)$$

And

$$[X_i] = \text{trans}_{x_i}(a_{i,i+1})\text{Rot}_{x_i}(\alpha_{i,i+1}) \qquad (1.24)$$

Note that this is the product of two screw displacements, The matrices associated with these operations are

$$\text{trans}_{z_i}(d_i)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_i\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (1.25)$$

$$\text{Rot}_{z_{i-1}}(\theta_i)=\begin{bmatrix}c\theta_i & -s\theta_i & 0 & 0\\ s\theta_i & c\theta_i & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}(1.26)$$

$$\text{trans}_{x_i}(a_i)=\begin{bmatrix}1 & 0 & 0 & a_i\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}(1.27)$$

$$\text{Rot}_{x_i}(\alpha_i)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & c\alpha_i & -s\alpha_i & 0\\ 0 & s\alpha_i & c\alpha_i & 0\\ 0 & 0 & 0 & 1\end{bmatrix}(1.28)$$

This gives :

$$^{i-1}T_i=\begin{bmatrix}c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i\\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i\\ 0 & s\alpha_i & c\alpha_i & d_i\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix} & R & & t\\ 0 & 0 & 0 & 1\end{bmatrix}(1.29)$$

where R is the 3×3 submatrix describing rotation and t is the 3×1 submatrix describing translation[3]

1.2.3 Application on the chosen robot:

Consider now the 3 DOF Elbow manipulator represented symbolically by Figure 1.70.

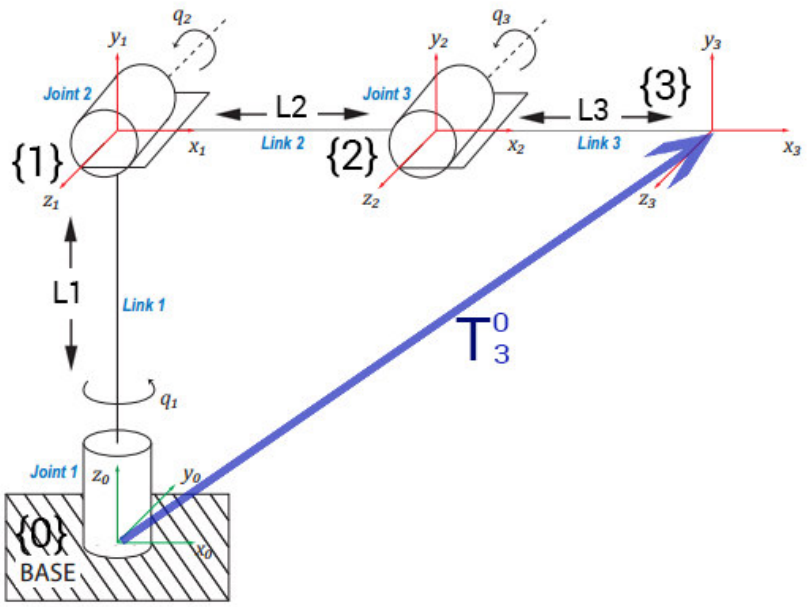


Figure (1.7) : Links, joints , coordinate frames and transformation vector on an 3 DOF elbow manipulator

Table 1.1 DH parameters for 3-link Elbow manipulator.

i	θ_{i-1}	d_{i-1}	ϕ_i	α_i
1	q_1	l_1	0	$\pi/2$
2	q_2	0	l_2	0
3	q_3	0	l_3	$-\pi/2$

We have

$$T_3^0=[Z_1][X_1][Z_2][X_2][X_3][Z_3]$$

And we know from (1.23) and (1.24) that

$$[Z_i]= \text{trans}_{z_i}(d_i)\text{Rot}_{z_i}(\theta i)\text{And} \quad [X_i]= \text{trans}_{x_i}(a_{i,i+1})\text{Rot}_{x_i}(\alpha_{i,i+1})$$

And by replacing the parameters we get

$$T_3^0 = \mathrm{Rot_Z}(q_1)\mathrm{trans}_{\ Z}(l_1)\mathrm{Rot_X}(\alpha_2)\mathrm{Rot_Z}(q_2)\mathrm{trans}_{\ X}(l_2)\mathrm{Rot_Z}(q_3)\ \mathrm{trans}_{\ X}(l_3)\mathrm{Rot_X}(\pi/2) \quad (1.30)$$

And the Transformation matrix is given by:

$$\boldsymbol{T_3^0} = \begin{bmatrix} \boldsymbol{c_{23}c_1} & \boldsymbol{-s_1} & \boldsymbol{-s_{23}c_1} & \boldsymbol{c_1(l_3c_{23}+l_2c_2)} \\ \boldsymbol{c_{23}s_1} & \boldsymbol{c_1} & \boldsymbol{-s_{23}c_1} & \boldsymbol{c_1(l_3c_{23}+l_2c_2)} \\ \boldsymbol{s_{23}} & \boldsymbol{0} & \boldsymbol{c_{23}} & \boldsymbol{l_1+l_3s_{23}+l_2s_2} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \quad (1.31)$$

And we can derive

$$R_3^0 = \begin{bmatrix} c_{23}c_1 & -s_1 & -s_{23}c_1 \\ c_{23}s_1 & c_1 & -s_{23}c_1 \\ s_{23} & 0 & c_{23} \end{bmatrix} \quad \text{and} \quad t_3^0 = \begin{bmatrix} c_1(l_3c_{23}+l_2c_2) \\ c_1(l_3c_{23}+l_2c_2) \\ l_1+l_3s_{23}+l_2s_2 \end{bmatrix}$$

Denote that we have :

$$c_{23} = \cos(q_2+q_3) \quad , \quad c_1 = \cos(q_1) \ , \quad c_2 = \cos(q_2)$$
$$s_1 = \sin(q_1) \quad , \quad s_{23} = \sin(q_2+q_3)$$

1.3 Differential Kinematics:

1.3.1 Derivation of the Jacobian:

In vector analysis, the Jacobian matrix is a matrix associated with a vector function at a given point. Its name comes from the mathematician Charles Jacobi. The determinant of this matrix, called Jacobian, plays an important role in solving nonlinear problems.

The time derivative of the kinematics equations yields the Jacobian of the robot, which relates the joint rates to the linear and angular velocity of the end-effector, The robot Jacobian results in a set of linear equations that relate the joint

rates to the six-vector formed from the angular and linear velocity of the end-effector.

a. Angular Velocity:

Consider that the angular velocity is noted as ω,Now when a rigid body moves in a pure rotation about a fixed axis, every point of the body moves in a circle. The centers of these circles lie on the axis of rotation.

As the body rotates, a perpendicular from any point of the body to the axis sweeps out an angle θ, and this angle is the same for every point of the body. [11]

The angular velocity ω with respect to a joint frame is expressed by its angular velocity and its axe of rotation vector and can be written as

$\omega = \hat{Z}\dot{q}$ with $\hat{Z}$ is the unite vector which describes the axe of rotation of the joint , denote that

$$\hat{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad (1.32)$$

For a robot manipulator, we consider ω the angular velocity of the end-effector, this angular velocity is the vectorial sum of the provided angular velocity of each joint that they are expressed relative to a common coordinate frame, in our case the base coordinate frame, and all the axis of rotation must be represented in the base frame, ωcan be expressed by:

$\omega = \sum_{i=1}^{n} Z_i \dot{q}_i$(1.33)

We know that $Z_i$is the axe of rotation of the ith joint expressed in the base frame (frame {0}), thus we can write it

$Z_i = {}^{0}_{i}R\hat{Z}$(1.34)

By replacing Z_i in (1.33) , becomes

$$\omega = \left[{}^{0}_{1}R\hat{Z}\ {}^{0}_{2}R\hat{Z}\cdots\ {}^{0}_{n}R\hat{Z}\right] * \begin{bmatrix}\dot{q_1}\\ \dot{q_2}\\ \vdots\\ \dot{q_n}\end{bmatrix} \quad (1.35)$$

And we can write

$$\omega = J_{\omega}\dot{q} \quad (1.36)$$

which gives us

$$J_{\omega} = \left[{}^{0}_{1}R\hat{Z}\ {}^{0}_{2}R\hat{Z}\cdots\ {}^{0}_{n}R\hat{Z}\right] (1.37)$$

and J_{ω} is the derivative of the angular velocity

b. Linear Velocity:

We now consider the linear velocity of a point that is rigidly attached to a moving frame.

Suppose the point t^0 is rigidly attached to the frame o1x1y1z1, and that o1x1y1z1 is rotating relative to the frame o0x0y0z0, so we can express the linear velocity of frame {1} with respect to {0} frame, and it can be written as[12] :

$$V = \frac{d\,t^0}{dt} = \frac{d\,t^0}{dq_1}\frac{d\,q_1}{dq_1} = \frac{d\,t^0}{dq_1}\dot{q_1}$$

In case of robot manipulator, t^0_n is the end-effector position vector, the linear velocities can be added vectorially and becomes

$$V = \left[\frac{d\,t^0_n}{dq_1}\dot{q_1} + \frac{d\,t^0_n}{dq_2}\dot{q_2} + \cdots + \frac{d\,t^0_n}{dq_n}\dot{q_n}\right] \quad (1.38)$$

And it becomes

$$V = \left[\frac{d\,t^0_n}{dq_1}\frac{d\,t^0_n}{dq_2}\ \ \ldots\ \ \frac{d\,t^0_n}{dq_n}\right] * \begin{bmatrix}\dot{q_1}\\ \dot{q_2}\\ \vdots\\ \dot{q_n}\end{bmatrix} \quad (1.39)$$

And we can write it as

$$V = J_V \dot{q} \quad (1.40)$$

hence

$$J_v = \left[\frac{d\, t_n^0}{dq_1} \frac{d\, t_n^0}{dq_2} \dots \frac{d\, t_n^0}{dq_n}\right] \quad (1.41)$$

And J_v is the derivative of the Linear Velocity

C. Combining the Angular and Linear Jacobians:

The jacobian matrix expressed by combining each of jacobian, the linear and the angular [13]

We have

$$\begin{pmatrix} V \\ \omega \end{pmatrix} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \dot{q} \quad (1.42)$$

And we can write

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \quad (1.43)$$

And J the jacobian matrix of the manipulator

1.3.2 application on the chosen robot :

from what we have studied previously we can get :

$$V = J_V \dot{q} \quad ; \quad \omega = J_\omega \dot{q} \rightarrow \begin{pmatrix} V \\ \omega \end{pmatrix} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \dot{q} \rightarrow J = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix}$$

And we know from (1.41)that :

$$J_v = \left[\frac{d\, t_3^0}{dq_1} \frac{d\, t_3^0}{dq_2} \frac{d\, t_3^0}{dq_3}\right] \quad (1.44)$$

First we have the position vector of our robot dented by (1.32)

$$t_3^0 = \begin{bmatrix} c_1(l_3c_{23} + l_2c_2) \\ c_1(l_3c_{23} + l_2c_2) \\ l_1 + l_3s_{23} + l_2s_2 \end{bmatrix} \quad (1.45)$$

$$\frac{d\, t_3^0}{dq_1} = \begin{bmatrix} -s_1(l_3c_{2.3} + l_2c_2) \\ c_1(l_3c_{2.3} + l_2c_2) \\ 0 \end{bmatrix} \quad (1.46)$$

$$\frac{dt_3^0}{dq_2} = \begin{bmatrix} -c_1(l_3s_{2.3} + l_2s_2) \\ -\ s_1(l_3s_{2.3} + l_2s_2) \\ l_3c_{23} + l_2c_2 \end{bmatrix} \quad (1.47)$$

$$\frac{dt_3^0}{dq_3} = \begin{bmatrix} -l_3s_{2.3}c_1 \\ -l_3s_{2.3}s_1 \\ l_3c_{2.3} \end{bmatrix} \quad (1.48)$$

Thence:

$$J_v = \begin{bmatrix} -s_1(l_3c_{2.3} + l_2c_2) & -c_1(l_3s_{2.3} + l_2s_2) & -l_3s_{2.3}c_1 \\ c_1(l_3c_{2.3} + l_2c_2) & -\ s_1(l_3s_{2.3} + l_2s_2) & -l_3s_{2.3}s_1 \\ 0 & l_3c_{23} + l_2c_2 & l_3c_{2.3} \end{bmatrix} \quad (1.49)$$

And we know from (1.37) that the angular jacobian is

$$J_\omega = \begin{bmatrix} {}_1^0R\hat{Z}\ {}_2^0R\hat{Z}\ {}_3^0R\hat{Z} \end{bmatrix} \quad (1.50) \quad \text{with } \hat{Z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$Z_1 = {}_1^0R\hat{Z} = \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (1.51)$$

$$Z_2 = {}_2^0R\hat{Z} = \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\frac{\pi}{2}) & -s(\frac{\pi}{2}) \\ 0 & c(\frac{\pi}{2}) & c(\frac{\pi}{2}) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} c_1 & 0 & s_1 \\ s_1 & 0 & -c_1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \quad (1.52)$$

$$Z_3 = {}_3^0R\hat{Z} = \begin{bmatrix} c_1 & 0 & s_1 \\ s_1 & 0 & c_1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \quad (1.53)$$

thence

$$J_\omega = [\, Z_1 \;\; Z_2 \;\; Z_3 \,] = \begin{bmatrix} 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix} \qquad (1.54)$$

By combining the two jacobian we get:

$$J = \begin{bmatrix} -s_1(l_3 c_{2.3} + l_2 c_2) & -c_1(l_3 s_{2.3} + l_2 s_2) & -l_3 s_{2.3} c_1 \\ c_1(l_3 c_{2.3} + l_2 c_2) & -s_1(l_3 s_{2.3} + l_2 s_2) & -l_3 s_{2.3} s_1 \\ 0 & l_3 c_{23} + l_2 c_2 & l_3 c_{2.3} \\ 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix} \qquad (1.55)$$

1.4 Inverse kinematics:

The inverse kinematics problem is, given the position and orientation of the tool frame, to compute the corresponding joint angles. The inverse kinematics problem is considerably harder than the forward kinematics problem, where a unique closed form solution always exists, and there are several methods can be used to get the inverse kinematics .

1.4.1 Iterative Method:

In this method we consider a few change of the variable q and a few change of the variable x and we can express them as Δx and Δq respectively

with J is the Jacobian Matrix,we can write

$$\Delta x = J\, \Delta q \qquad (1.56)$$

$$\Delta x = xd - x \qquad (1.57)$$

with $\begin{cases} x : \text{actual position} \\ xd: \text{desired position} \end{cases}$

The Algorithm that's used in this method

1. Compute actual position :x
2. Compute :$\Delta x = xd - x$
3. Using pseudo-inverse : $\Delta q = (J^T J)^{-1} J^T \Delta x$ (1.58)
4. Calculate the new angle q_{new} : $q_{new} = q_{old} + \Delta q$ (1.59)

1.4.2 The geometric Approach:

This method is used to solve the unknown joint angles required for the autonomous positioning of a robotic arm. A plethora of complex mathematical processes is reduced using basic trigonometric in the modeling of the robotic arm.

1.4.3 Application on the chosen robot :

As we have mentioned we have to found the unknown angles for each joint; and in our case we have 3 link arm robot so we have to find (q_1, q_2, q_3) , for the this we have used basic trigonometric equations :

For the first joint angel q_1 we have

$$q_1 = \arctan\left(\frac{y}{x}\right) \quad (1.60)$$

And for the third joint angel q_3:

$$\text{Sin } q_3 = \sqrt{1 + \cos q_3} \quad (1.61)$$

And we have $\Delta = \sqrt{x^2 + y^2}$ (1.62)

and $r^2 = \Delta^2 + z_1^2$ (1.63)

and $r^2 = l_2^2 + l_3^2 - 2 l_2 l_3 \cos\alpha$ (1.64)

Thence $\text{Cos } \alpha = \frac{(l_2^2 + l_3^2 - \Delta^2 - z_1^2)}{2 l_2 l_3}$ (1.65)

And $q_3 = \pi - \alpha$ (1.66)

and $\cos q_3 = -\text{Cos}\alpha$ (1.67)

so $\cos q_3 = -\frac{(l_2^2 + l_3^2 - \Delta^2 - z_1^2)}{2 l_2 l_3}$ (1.68)

thence $q_3 = \text{acos}\left(-\frac{(l_2^2 + l_3^2 - \Delta^2 - z_1^2)}{2 l_2 l_3}\right)$ (1.69)

and for the second joint angel q_2 we have

$$\tan(B) = \frac{(l_3 \sin q_3)}{l_2 + l_3 \cos q_3}$$

$$B = \tan^{-1}\left(\frac{(l_3 \sin q_3)}{l_2 + l_3 \cos q_3}\right) \quad (1.70)$$

In the other hand we have : $\Upsilon = B + q_2 \Leftrightarrow q_2 = \Upsilon - B$ (1.71)

and $\tan(\Upsilon) = \frac{z_1}{\Delta} \Leftrightarrow \Upsilon = \tan^{-1}(z_1/\Delta)$ (1.72)

Hence

$$q_2 = \tan^{-1}(z_1/\Delta) - \tan^{-1}\left(\frac{(l_3 \sin q_3)}{l_2 + l_3 \cos q_3}\right) \quad (1.73)$$

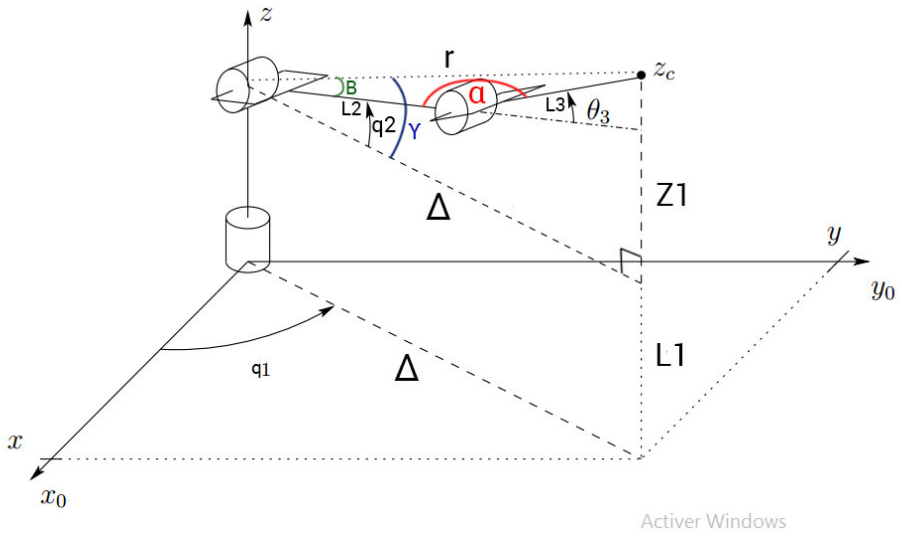


Figure (1.8) :angels and distances of a 3 DOF elbow manipulator

1.5 Dynamics:

For control design purposes, it is necessary to have a mathematical model that reveals the dynamical behavior of a system. Therefore, in this section we derive the dynamical equations of motion for a robot manipulator. Our approach is to derive the kinetic and potential energy of the manipulator and then use Lagrange's equations of motion to describe the dynamic properties of the robot arm. This relation may be on the form [14]

$$\underbrace{M(q)\ddot{q}}_{\text{Mass Matrix}} + \underbrace{C(q,\dot{q})\dot{q}}_{\text{Centrifugal and Coriolis Forces}} + \underbrace{G(q)}_{\text{Gravity}} + \underbrace{F(\dot{q})}_{\text{Friction}} = \underbrace{\tau}_{\text{Input Torque}} \quad (1.74)$$

1.5.1 Lagrange-Euler formulations

The Lagrangian is formulated as [15]

$$\frac{d}{dt}\left(\frac{\partial l}{\partial \dot{q}}\right) - \frac{\partial l}{\partial q} = \tau \qquad (1.75)$$

The function L, which is the difference of the kinetic and potential energy, is called the Lagrangian of the system, and Equation (1.96)is called the Euler-Lagrange Equation.

$$L = k - u \qquad (1.76)$$

Thence

$$\frac{\partial}{\partial t}\left(\frac{\partial k}{\partial \dot{q}}\right) - \frac{\partial k}{\partial q} + \frac{\partial u}{\partial q} = \tau \qquad (1.77)$$

Where T is the total kinetic energy and U is the total potential energy of a system that consists of n rigid links.

1.5.1.1 Inertial force:

And here we must talk about the mass matrix

a. *Mass matrix* :

the mass matrix M(q) is a symmetric matrix M that expresses the connection between the time derivatives $\dot{q}$of the generalized coordinate vector q of a system

The kinetics energy for a manipulator is define as

We have : $$K = \frac{1}{2}\dot{q}^t M(q)\,\dot{q} \qquad (1.78)$$

So : $$\frac{\partial k}{\partial \dot{q}} = \frac{\partial}{\partial \dot{q}}\left(\frac{1}{2}\dot{q}^t M(q)\,\dot{q}\right) = M(q)\,\dot{q} \qquad (1.79)$$

hence : $$\frac{\partial}{\partial t}\left(\frac{\partial k}{\partial \dot{q}}\right) = \frac{\partial}{\partial t}(M(q)\,\dot{q}) = M(q)\ddot{q} + \dot{M}(q)\,\dot{q} \qquad (1.80)$$

And we get: $$M(q)\ddot{q} + \dot{M}(q)\,\dot{q} - \frac{1}{2}\begin{bmatrix} \dot{q}^t \frac{\partial M}{\partial \dot{q}_1} \\ \vdots \\ \dot{q}^t \frac{\partial M}{\partial \dot{q}_n} \end{bmatrix} \qquad (1.81)$$

The equation (1.81) describes the Inertial forces Represented in mass matrix and Centrifugal &Coriolis Forces.

For the kinetic energy we can defined it as :

$$K= \sum_{i=1}^{n} k_i = \frac{1}{2}\dot{q}^t\, M(q)\, \dot{q} \qquad (1.82)$$

$$\left\{ \begin{array}{c} \text{for lineair movement : } k = \frac{1}{2}mv^2 \\ \text{for rotational movement : } k = \frac{1}{2}\omega^t I_C \omega \end{array} \right\}$$

In addition to this we have

$$k_i = \frac{1}{2}\left(m_i {v^t}_c v_c + {\omega_i}^t I_{C_i} \omega_i\right) \qquad (1.83)$$

From (1.83) and (1.82)

$$K= \frac{1}{2}\dot{q}^t\, M(q)\, \dot{q} = \frac{1}{2}\sum_{i=1}^{n}\left(m_i {v^t}_c v_c + {\omega_i}^t I_{C_i} \omega_i\right) \qquad (1.84)$$

And we have $v_{c_i} = J_{v_i}\dot{q}$ (1.85)

With J_{v_i} the linear jacobian for the ith joint define as

$$J_{v_i} = \left[\frac{\partial p_{c_i}}{\partial q1} \quad \cdots \quad \frac{\partial p_{c_i}}{\partial qn}\right] \qquad (1.86)$$

Denoted that Pci : center of mass position vector in frame {0}

And we have also $\omega_i = J_{\omega_i}\dot{q}$ (1.87)

With J_{ω_i} the angular jacobian for the ith joint define as

$$J_{\omega_i} = [Z_1 \quad \cdots \quad Z_n] \qquad (1.88)$$

Denoted that Zi : axe of Rotation in frame {0}

Thence $\frac{1}{2}\dot{q}^t\, M(q)\, \dot{q} = \frac{1}{2}\dot{q}^t \sum_{i=1}^{n}\left(m_i {J^t}_v J_v + {\omega_i}^t I_{C_i} \omega_i\right)\dot{q}$ (1.89)

And we get the mass matrix expression

$$M(q) = \sum_{i=1}^{n} \left(m_i J^t{}_v J_v + \omega_i{}^t I_{C_i} \omega_i \right) \qquad (1.90)$$

The mass matrix can be written as this form

$$M(q) = \begin{bmatrix} M_{11} M_{12} \dots M_{1n} \\ \ddots \\ M_{n1} M_{n2} \dots M_{nn} \end{bmatrix} \qquad (1.91)$$

b. Centrifugal and Coriolis Forces :

The Coriolis force is an inertial force (also called a fictitious force) that acts on objects that are in motion relative to a rotating reference frame. In a reference frame with clockwise rotation, the force acts to the left of the motion of the object. In one with anticlockwise rotation, the force acts to the right [16]

Whereas the centripetal (Centrifugal) force is seen as a force which must be applied by an external agent to force an object to move in a curved path, the Centrifugal and Coriolis forces are "effective forces" which are invoked to explain the behavior of objects from a frame of reference which is rotating. [17]

Centrifugal &Coriolis terms can be extracted from (1.81) and we can write

$$C(q, \dot{q})\dot{q} = \dot{M}(q)\dot{q} - \frac{1}{2}\begin{bmatrix} \dot{q}^t \frac{\partial M}{\partial \dot{q}_1} \\ \vdots \\ \dot{q}^t \frac{\partial M}{\partial \dot{q}_n} \end{bmatrix} = \begin{bmatrix} \dot{M}_{11} \dot{M}_{12} \dots \dot{M}_{1n} \\ \ddots \\ \dot{M}_{n1} \dot{M}_{n2} \dots \dot{M}_{nn} \end{bmatrix} \dot{q} - \frac{1}{2}\begin{bmatrix} \dot{q}^t \begin{bmatrix} \dot{M}_{11} \dot{M}_{12} \dots \dot{M}_{1n} \\ \ddots \\ \dot{M}_{n1} \dot{M}_{n2} \dots \dot{M}_{nn} \end{bmatrix} \dot{q} \\ \vdots \\ \dot{q}^t \begin{bmatrix} \dot{M}_{11} \dot{M}_{12} \dots \dot{M}_{1n} \\ \ddots \\ \dot{M}_{n1} \dot{M}_{n2} \dots \dot{M}_{nn} \end{bmatrix} \dot{q} \end{bmatrix} \qquad (1.92)$$

We define $$M_{ijk} = \frac{dM_{ij}}{dq_k} \qquad (1.93)$$

And $$\dot{M}_{ij} = M_{ij1}\dot{q}1 + M_{ij2}\dot{q}2 + \cdots + M_{ijn}\dot{q}n \qquad (1.94)$$

And we can write

$$C(q,\dot{q})\dot{q} = \begin{bmatrix} \frac{1}{2}\left(\dot{M}_{111}\dot{M}_{121}\dots\dot{M}_{ij1}\right)+\frac{1}{2}\left(\dot{M}_{112}\dot{M}_{122}\dots\dot{M}_{ij2}\right)+\cdots\frac{1}{2}\left(\dot{M}_{11n}\dot{M}_{12n}\dots\dot{M}_{ijn}\right) \\ \ddots \\ \frac{1}{2}\left(\dot{M}_{11n}\dot{M}_{12n}\dots\dot{M}_{ijn}\right)+\frac{1}{2}\left(\dot{M}_{11n}\dot{M}_{12n}\dots\dot{M}_{ijn}\right)+\cdots\frac{1}{2}\left(\dot{M}_{11n}\dot{M}_{12n}\dots\dot{M}_{ijn}\right) \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} M_{111}M_{121}\dots M_{ij1} \\ \ddots \\ M_{n1n}M_{nn1}\dots M_{1nn} \end{bmatrix} \begin{bmatrix} q_1 q_2 \\ \vdots \\ q_{n-1}q_n \end{bmatrix} \end{bmatrix} \quad (1.95)$$

Using Christoffel Symbols :

$$b_{ijk} = \frac{1}{2}\left(M_{ijk} + M_{ikj} - M_{jki}\right) \quad (1.96)$$

We can write a general form :

$$C(q,\dot{q})\dot{q} = c_1(q)\dot{q}^2 + c_2(q)\dot{q}\dot{q} \quad (1.97)$$

Centrifugal term : $$c1(q)\dot{q}^2 = \begin{bmatrix} \begin{bmatrix} b_{111}b_{122}\dots b_{1nm} \\ \ddots \\ b_{n11}b_{n22}\dots b_{nnm} \end{bmatrix} \begin{bmatrix} \dot{q}^2_1 \\ \vdots \\ \dot{q}^2_n \end{bmatrix} \end{bmatrix} \quad (1.98)$$

Coriolis term: $$c2(q)[\dot{q}\dot{q}] = \begin{bmatrix} 2b_{111}2b_{122}\dots 2b_{n(n-1)n} \\ \ddots \\ 2b_{n11}2b_{n22}\dots 2b_{n(n-1)n} \end{bmatrix} \begin{bmatrix} \dot{q}^2_1\dot{q}^2_2 \\ \vdots \\ \dot{q}^2_{n-1}\dot{q}^2_n \end{bmatrix} \quad (1.99)$$

c. Inertia matrix :

the moment of inertia is a scalar value expressing the resistance to changes to the rotation of an object. If the axis o rotation is not given, it is possible to generalize the scalar moment of inertia as 3X3 matrix expressing the moment of inertia about arbitrary axes. This matrix called also inertia tensor.

Let the mass density of an object ρ, and the inertia tensor in frame attached to center of mass of the object and p the position vector defin as

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (1.100)$$

We have $I = \int -\hat{p}\,\hat{p}\rho\, dv$ and $-\hat{p}\hat{p} = (p^t p)I_3 - pp^t$ (1.101)

Thence $$I = \int [-(p^t p)I_3 - pp^t]\,\rho dv \quad (1.102)$$

We have $-\hat{p}\hat{p} = [-(p^t p)I_3 - pp^t] = \begin{bmatrix} y^2+z^2 & -xy & xz \\ -xy & z^2+x^2 & -yz \\ x^2 & -y^2 & x^2+y^2 \end{bmatrix}$ (1.103)

the inertia tensor expressed as $I = \begin{bmatrix} I_{XX} & -I_{XY} & I_{XZ} \\ -I_{XY} & I_{YY} & -I_{YZ} \\ -I_{XX} & -I_{YZ} & I_{ZZ} \end{bmatrix}$ (1.104)

and we define the Moments of Inertia

$$I_{XX} = \iiint (y^2+z^2)\rho dxdydz \quad (1.105)$$

$$I_{YY} = \iiint (x+z^2)\rho dxdydz \quad (1.106)$$

$$I_{ZZ} = \iiint (y^2+x^2)\rho dxdydz \quad (1.107)$$

and we have also the Products of Inertia

$$I_{XY} = \iiint xy\rho dxdydz \quad (1.108)$$

$$I_{XZ} = \iiint xz\rho dxdydz \quad (1.109)$$

$$I_{YZ} = \iiint yz\rho dxdydz \quad (1.110)$$

1.5.1.2 Gravity term (Potential Energy):

Now consider the potential energy term. In the case of rigid dynamics, the only source of potential energy is gravity. The potential energy of the i-th link can be

computed by assuming that the mass of the entire object is concentrated at its center of mass and is given by

$$\left.\begin{matrix} u_i = m_i g\, h_i \\ u_i = m_i(-g_0{}^t p_{ci}) \end{matrix}\right\} \text{ and } U = \sum_{i=1}^{n} u_i \quad (1.111)$$

We have the Gravity Vector : $g_0 = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix}$ (1.112)

And we have the linear Jacobin $J_{v_i} =$

$$\left[\frac{\partial p_{c_i}}{\partial q1} \quad \dots \quad \frac{\partial p_{c_i}}{\partial qn}\right] \quad (1.113)$$

We define $G = \frac{\partial U}{\partial q} = -\sum_{i=1}^{n}(m_i g_0{}^t \frac{\partial p_{ci}}{\partial q})$

And from the previous equation we get the gravity term define as

$$G(\dot{q}) = -\left(J_{v_1}{}^t J_{v_2}{}^t \dots J_{v_n}{}^t\right)\begin{pmatrix} m_1 g_0 \\ \vdots \\ m_n g_0 \end{pmatrix} = -(J_{v_1}{}^t(m_1 g_0) + J_{v_2}{}^t(m_2 g_0) + \dots + J_{v_n}{}^t(m_n g_0)) \quad (1.114)$$

1.5.1.3 Friction modeling:

Although joint frictions are complicated in reality, a simple model which is the combination of viscous and Coulomb and stiction(stribeck effect) , is normally used to describe the friction phenomenon for all joints:

$$F(\dot{q}) = \underbrace{F_v \dot{q}}_{\substack{\text{viscous} \\ \text{friction}}} + \underbrace{F_c \operatorname{sign}(\dot{q})}_{\substack{\text{Coulomb} \\ \text{friction}}} + \underbrace{F_s(\dot{q})}_{\substack{\text{stribeck} \\ \text{effect}}} \quad (1.115)$$

I. <u>viscous friction :</u>

The viscous friction element models the friction force as a force proportional to the sliding velocity.

II. <u>Coulomb friction :</u>

The Coulomb approximation mathematically follows from the assumptions that surfaces are in atomically close contact only over a small fraction of their overall area, that this contact area is proportional to the normal force.

and that the frictional force is proportional to the applied normal force, independently of the contact area.

the Coulomb approximation is an adequate representation of friction for the analysis of many physical systems.[18]

III. stiction(stribeck effect):

The Stribeck curve is a more advanced model of friction as a function of velocity. Although it is still valid only in steady state, it includes the model of Coulomb and viscous friction as built-in elements.[19]

And We write $F_s(\dot{q}) = (f_s - f_c)\,\mathrm{sign}(\dot{q})e^{-\left|\frac{\dot{q}}{qs}\right|^2}$ (1.116)

With q_s is Stribeck velocity

And we can represent all the friction component by the following figure

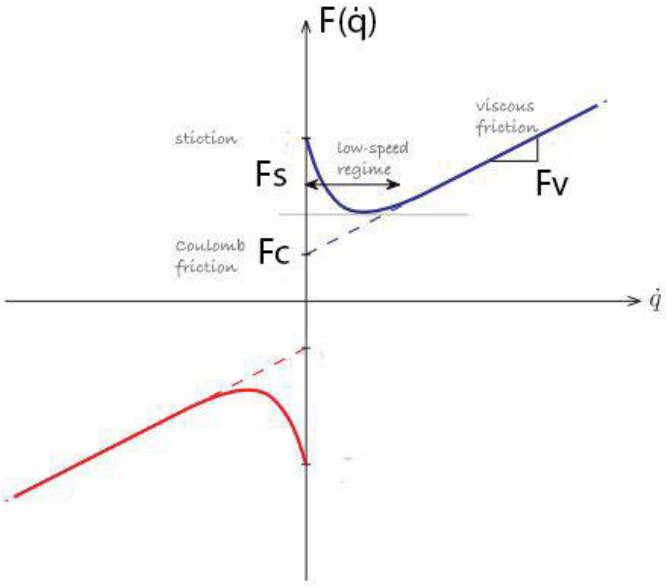


Figure (1.9): Friction model with coulomb ,viscous and stiction trem

For a serial robot manipulator we can write the friction term as [20]

$$F(\dot{q}) = \begin{bmatrix} f_{v_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & f_{v_n} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} +$$

$$\begin{bmatrix} f_{c_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & f_{c_n} \end{bmatrix} \begin{bmatrix} \text{sign}(\dot{q}_1) \\ \vdots \\ \text{sign}(\dot{q}_n) \end{bmatrix} + \begin{bmatrix} (f_{s_1} - f_{c1}) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & (f_{s_n} - f_{cn}) \end{bmatrix} \begin{bmatrix} \text{sign}(\dot{q}_1) e^{-\left|\frac{\dot{q}_1}{q_s}\right|^2} \\ \vdots \\ \text{sign}(\dot{q}_n) e^{-\left|\frac{\dot{q}_n}{q_s}\right|^2} \end{bmatrix} \quad (1.117)$$

Denote that

Fci : express Coulomb friction of the ith joint .

Fvi : express viscous friction of the ith joint .

Fsi : express stiction term of the ith joint .

qs : expresses the Stribeck velocity .

1.5.2 Actuator Dynamics :

For the actuators which are dc motors, we choose a very simple model, because of the huge nonlinearity present in the manipulator joints, the motor dynamics could be Neglected or represented by a simple linear model as follow [20]

$$U = J\ddot{\theta} + \tau + Tf \qquad (1.118)$$

With $\boldsymbol{\tau}$ joints torques, Tf represent the actuator friction , U represent the inputs voltage signals and J represent the actuator inertia. by replacing Eq(1.74) we get

$$U = J\ddot{\theta} + M(q)\ddot{q} + C(q, \dot{q})q + G(\dot{q}) + F(\dot{q}) + Tf \qquad (1.119)$$

And $\ddot{\theta} = G\ddot{q}$, G represent the gears ratios, denote that

$$U = \begin{bmatrix} U_1 \\ \vdots \\ U_n \end{bmatrix}, \acute{J} = \begin{bmatrix} j_1 G_1 \\ \vdots \\ j_n G_n \end{bmatrix}, Tf = \begin{bmatrix} Tf_1 \\ \vdots \\ Tf_n \end{bmatrix}$$

Yield

$$U = \acute{J}\ddot{q} + M(q)\ddot{q} + C(q, \dot{q})q + G(\dot{q}) + F(\dot{q}) + Tf \qquad (1.120)$$

1.5.3 Application on the chosen robot :

we have the Figure 1.8 describe our 3dof elbow manipulator, all center of masses represented with respect to the base frame .

the choice of centre of masses depends on the real mass distribution of our robot .

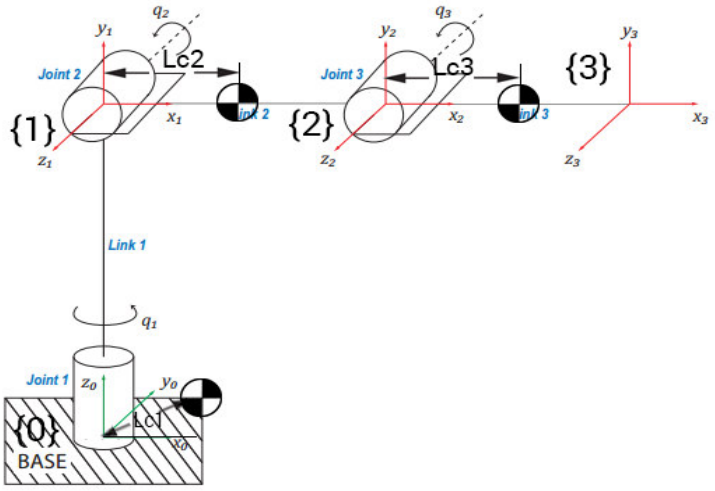


Figure (1.10): angels and distances of center of masses of a 3 DOF elbow manipulator

first we have the position vectors of center of masses denote as

$$p_{c1}=\begin{bmatrix} l_{c1}c_1 \\ l_{c1}s_1 \\ 0 \end{bmatrix} \quad p_{c1}=\begin{bmatrix} c_1(l_{c2}c_2) \\ s_1(l_{c2}c_2) \\ l_1+l_{c2}s_2 \end{bmatrix} p_{c3}=\begin{bmatrix} c_1(l_{c3}c_{23}+l_2c_2) \\ s_1(l_{c3}c_{23}+l_2c_2) \\ l_1+l_{c3}s_{23}+l_2s_2 \end{bmatrix}$$

a. Mass matrix

and we have the mass matrix of our 3dof robot as we define in (1.90)

$$M=\left(M_1{J^t}_1J_1+J_{\omega_1}{}^tI_{n_1}J_{\omega_1}\right)+\left(M_2{J^t}_2J_2+J_{\omega_2}{}^tI_{n_2}J_{\omega_2}\right)+\left(M_3{J^t}_3J_3+J_{\omega_3}{}^tI_{n_3}J_{\omega_3}\right)$$

we derive of the linear jacobian of the first joint

$$J_{v1}=\left[\frac{dp_{c1}}{dq_1}\frac{dp_{c1}}{dq_2}\frac{dp_{c1}}{dq_3}\right] = \begin{bmatrix} -l_{c1}s_1 & 0 & 0 \\ l_{c1}c_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

And also the angular jacobian of the first joint

$$J_{\omega_1}=[z_1 \quad 0 \quad 0]=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

and we have the Inertia matrix of the first joint express as

$$I_{n_1} = \begin{bmatrix} I_{XX1} & -I_{YX1} & -I_{ZX1} \\ -I_{YX1} & I_{YY1} & -I_{YZ1} \\ -I_{ZX1} & -I_{YZ1} & I_{ZZ1} \end{bmatrix}$$

so we can write the first part of the mass matrix that depend on the first joint

$$\left(M_1 J^t{}_1 J_1 + J_{\omega_1}{}^t I_{n_1} J_{\omega_1}\right) =$$

$$M_1 \begin{bmatrix} -l_{c1}s_1 & l_{c1}c_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -l_{c1}s_1 & 0 & 0 \\ l_{c1}c_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} I_{XX1} & -I_{YX1} & -I_{ZX1} \\ -I_{YX1} & I_{yy1} & -I_{YZ1} \\ -I_{ZX1} & -I_{YZ1} & I_{zz1} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} M_1 l_{c1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} I_{zz1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} M_1]^2{}_{c1} + I_{zz1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Same thing for the second joint we have linear jacobian and angular jacobian express as

$$J_{v2} = \begin{bmatrix} \frac{dp_{c2}}{dq_1} & \frac{dp_{c2}}{dq_2} & \frac{dp_{c2}}{dq_3} \end{bmatrix} = \begin{bmatrix} -s_1 l_{c2} c_2 & -c_1 l_{c2} s_2 & 0 \\ c_1 l_{c2} c_2 & -s_1 l_{c2} s_2 & 0 \\ 0 & l_{c2} c_2 & 0 \end{bmatrix}$$

$$J_{\omega_2} = \begin{bmatrix} z_1 & z_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Also we have the Inertia matrix of the second joint :

$$I_{n_2} = \begin{bmatrix} I_{XX2} & -I_{YX2} & -I_{ZX2} \\ -I_{YX2} & I_{YY2} & -I_{YZ2} \\ -I_{ZX2} & -I_{YZ2} & I_{ZZ2} \end{bmatrix}$$

And we get the second part of the mass matrix

$$\left(M_2 J^t{}_2 J_2 + J_{\omega_2}{}^t I_{n_2} J_{\omega_2}\right) =$$

$$M_2\begin{bmatrix}-s_1l_{c2}c_2 & c_1l_{c2}c_2 & 0\\ -c_1l_{c2}s_2 & -s_1l_{c2}s_2 & l_{c2}c_2\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}-s_1l_{c2}c_2 & -c_1l_{c2}s_2 & 0\\ c_1l_{c2}c_2 & -s_1l_{c2}s_2 & 0\\ 0 & l_{c2}c_2 & 0\end{bmatrix}+$$

$$\begin{bmatrix}0 & 0 & 1\\ 0 & -1 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}I_{XX2} & -I_{YX2} & -I_{ZX2}\\ -I_{YX2} & I_{yy2} & -I_{YZ2}\\ -I_{ZX2} & -I_{YZ2} & I_{zz2}\end{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & -1 & 0\\ 1 & 0 & 0\end{bmatrix}=$$

$$\begin{bmatrix}M_2{l_{c2}}^2{c_2}^2 & 0 & 0\\ 0 & M_2{l_{c2}}^2 & 0\\ 0 & 0 & 0\end{bmatrix}+\begin{bmatrix}I_{zz2} & I_{YZ2} & 0\\ I_{YZ2} & I_{YY2} & 0\\ 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}M_2{l_{c2}}^2{c_2}^2+I_{zz2} & I_{YZ2} & 0\\ I_{YZ2} & M_2{l_{c2}}^2+I_{YY2} & 0\\ 0 & 0 & 0\end{bmatrix}$$

And also for the third joint, we have linear and angular jacobian express as

$$J_{v3}=\left[\frac{dp_{c3}}{dq_1}\frac{dp_{c3}}{dq_2}\frac{dp_{c3}}{dq_3}\right] = \begin{bmatrix}-s_1(l_{c3}c_{23}+l_2c_2) & -c_1(l_{c3}s_{23}+l_2s_2) & -c_1l_{c3}s_{23}\\ c_1(l_{c3}c_{23}+l_2c_2) & -s_1(l_{c3}s_{23}+l_2s_2) & -s_1l_{c3}s_{23}\\ 0 & l_{c3}c_{23}+l_2c_2 & l_{c3}c_{23}\end{bmatrix}$$

$$J_{\omega_3}=[z_1 \quad z_2z_3]=\begin{bmatrix}0 & 0 & 0\\ 0 & -1 & -1\\ 1 & 0 & 0\end{bmatrix}$$

the Inertia matrix of the third joint :

$$I_{n_3}=\begin{bmatrix}I_{zz3} & -I_{YX3} & -I_{ZX3}\\ -I_{YX3} & I_{yy3} & -I_{YZ3}\\ -I_{ZX3} & -I_{YZ3} & I_{yy3}\end{bmatrix}$$

$$=M_3\begin{bmatrix}-s_1(l_{c3}c_{23}+l_2c_2) & c_1(l_{c3}c_{23}+l_2c_2) & 0\\ -c_1(l_{c3}s_{23}+l_2s_2) & -s_1(l_{c3}s_{23}+l_2s_2) & l_{c3}c_{23}+l_2c_2\\ -c_1l_{c3}s_{23} & -s_1l_{c3}s_{23} & l_{c3}c_{23}\end{bmatrix}$$

$$\begin{bmatrix}-s_1(l_{c3}c_{23}+l_2c_2) & -c_1(l_{c3}s_{23}+l_2s_2) & -c_1l_{c3}s_{23}\\ c_1(l_{c3}c_{23}+l_2c_2) & -s_1(l_{c3}s_{23}+l_2s_2) & -s_1l_{c3}s_{23}\\ 0 & l_{c3}s_{23}+l_2s_2 & l_{c3}c_{23}\end{bmatrix}+\begin{bmatrix}0 & 0 & 1\\ 0 & -1 & 0\\ 0 & -1 & 0\end{bmatrix}$$

$$=\begin{bmatrix}I_{XX3} & -I_{YX3} & -I_{ZX3}\\ -I_{YX3} & I_{yy3} & -I_{YZ3}\\ -I_{ZX3} & -I_{YZ3} & I_{zz3}\end{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & -1 & -1\\ 1 & 0 & 0\end{bmatrix}$$

Yield

$$\left(M_3 J^t{}_3 J_3 + J_{\omega_3}{}^t I_{n_3} J_{\omega_3}\right)=$$

$$M_3\begin{bmatrix} l^2{}_{c3}c^2{}_{23} + l^2{}_2c^2{}_2 + 2l_{c3}c_{23}l_2c_2 & 0 & 0 \\ 0 & l^2{}_{c3} + l_2{}^2 + 2l_{c3}l_2c_3 & l^2{}_{c3} + l_{c3}l_2c_3 \\ 0 & l^2{}_{c3} + l_{c3}l_2c_3 & l^2{}_{c3} \end{bmatrix}+$$

$$\begin{bmatrix} I_{zz3} & I_{YZ3} & I_{YZ3} \\ I_{YZ3} & I_{yy3} & I_{YY3} \\ I_{YZ3} & I_{YY3} & I_{yy3} \end{bmatrix}=$$

$$\begin{bmatrix} M_3(l^2{}_{c3}c^2{}_{23} + l^2{}_2c^2{}_2 + 2l_{c3}c_{23}l_2c_2) + Izz3 & I_{YZ3} & I_{YZ3} \\ I_{YZ3} & M_3(l^2{}_{c3} + l_2{}^2 + 2l_{c3}l_2c3) + I_{yy3} & M3(l^2{}_{c3} + l_{c3}l_2c_3) + I_{YY3} \\ I_{YZ3} & M_3(l^2{}_{c3} + l_{c3}l_2c_3) + I_{YY3} & M_3l^2{}_{c3} + I_{yy3} \end{bmatrix}$$

And The general mass matrix becomes

$$M=\begin{bmatrix} M_1l^2{}_{c1} + I_{zz1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} M_2l_{c2}{}^2c_2 + I_{zz2} & I_{YZ2} & 0 \\ 0 & M_2l^2{}_{c2} + I_{yy2} & 0 \\ 0 & 0 & 0 \end{bmatrix} +$$

$$\begin{bmatrix} I_{zz3} + M_3(l^2{}_{c3}c^2{}_{23} + l^2{}_2c^2{}_2 + 2l_{c3}c_{23}l_2c_2) & -I_{Yz3} & -I_{Zy3} \\ -I_{YX3} & I_{yy3} + M_3(l^2{}_{c3} + l_2{}^2 + 2l_{c3}l_2) & -I_{YZ3} \\ -I_{ZX3} & M_3(l^2{}_{c3} + l_{c3}l_2c_3) - I_{Yy3} & I_{yy3} + M_3l^2{}_{c3} \end{bmatrix}$$

And the other hand we have :

$$M=\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

$$M_{21} = M_{21}$$

$$M_{23} = M_{32}$$

$$M_{13} = M_{31}$$

$$M_{11} = M_1l^2{}_{c1} + I_{zz1}+M_2l_{c2}{}^2c_2 + I_{zz2}+I_{zz3} + M_3(l^2{}_{c3}c^2{}_{23} + l^2{}_2c^2{}_2 + 2l_{c3}c_{23}l_2c_2)$$

$$M_{21} = M_{12} = I_{YZ2} + I_{Yz3}$$

$$M_{13} = M_{31} = I_{yz3}$$

$$M_{23} = M_{32}=M_3(l^2{}_{c3} + l_{c3}l_2c_3) + I_{Yy3}$$

$$M_{22} = M_2 l^2_{c2} + I_{yy2} + I_{yy3} + M_3(l^2_{c3} + l_2{}^2 + 2l_{c3}l_2c_3)$$

$$M_{33} = I_{yy3} + M_3 l^2_{c3}$$

b. Centrifugal &Coriolis Forces

We begin with the Centrifugal Force as we define in (1.119)

$$c_1(q)\dot{q}^2 = \begin{bmatrix} b_{111} & b_{122} & b_{133} \\ b_{211} & b_{222} & b_{233} \\ b_{311} & b_{322} & b_{333} \end{bmatrix} \begin{bmatrix} \dot{q}^2_1 \\ \dot{q}^2_2 \\ \dot{q}^2_3 \end{bmatrix}$$

We know from (1.93) and (1.96) that

$$M_{ijk} = \frac{dM_{ij}}{dq_k} \quad \text{and} \quad b_{ijk} = \frac{1}{2}\left(M_{ijk} + M_{ikj} - M_{jki}\right)$$

So we get

$$M_{111} = 0$$

$$M_{122} = M_{212} = 0$$

$$M_{112} = -2M_2(l_{c2}{}^2 c_2 s_2 + l^2_{c3} c_{23} s_{23} + l^2{}_2 c_2 s_2 + l_{c3} s_{23} l_2 c_2 + l_{c3} l_2 c_{23} s_2)$$

$$M_{233} = -M_3 l_{c3} l_2 s_3$$

$$M_{332} = 0$$

$$M_{311} = M_{131} = 0$$

$$M_{333} = 0$$

$$M_{222} = 0$$

$$M_{322} = 0$$

$$M_{113} = -2M_2(l^2_{c3} c_{23} s_{23} + l_{c3} l_2 s_{23} c_2)$$

$M_{223} = 2M_2 l_{c3} l_2 s_3$

$M_{213} = 0$

$M_{231} = 0$

$M_{132} = 0$

$b_{111} = \frac{1}{2}(M_{111} + M_{111} - M_{111}) = \frac{M_{111}}{2} = 0$

$b_{122} = \frac{1}{2}(M_{122} + M_{122} - M_{221}) = M_{122} - \frac{M_{221}}{2} = 0$

$b_{133} = \frac{1}{2}(M_{133} + M_{133} - M_{331}) = M_{133} - \frac{M_{331}}{2} = 0$

$b_{211} = \frac{1}{2}(M_{211} + M_{211} - M_{112}) = M_{211} - \frac{M_{112}}{2} = M_2(l_{c2}{}^2 c_2 s_2 + l^2{}_{c3} c_{23} s_{23} + l^2{}_2 c_2 s_2 + l_{c3} s_{23} l_2 c_2 + l_{c3} l_2 c_{23} s_2)$

$b_{222} = \frac{1}{2}(M_{222} + M_{222} - M_{222}) = \frac{M_{222}}{2} = 0$

$b_{233} = \frac{1}{2}(M_{233} + M_{233} - M_{332}) = M_{233} - \frac{M_{332}}{2} = 0$

$b_{311} = \frac{1}{2}(M_{311} + M_{311} - M_{311}) = M_{311} - \frac{M_{113}}{2} = M_2(l^2{}_{c3} c_{23} s_{23} + l_{c3} l_2 s_{23} c_2)$

$b_{333} = \frac{1}{2}(M_{333} + M_{333} - M_{333}) = \frac{M_{333}}{2} = 0$

$b_{322} = \frac{1}{2}(M_{322} + M_{322} - M_{223}) = M_{322} - \frac{M_{223}}{2} = M_2 l_{c3} l_2 s_3$

And the final matrix becomes

$c_1(q)\dot{q}^2 =$

$$\begin{bmatrix} 0 & 0 & 0 \\ M_2 l_{c2}{}^2 c_2 s_2 + M_3(l^2{}_{c3} c^2{}_{23} s^2{}_{23} + l^2{}_2 c_2 s_2 + l_{c3} s_{23} l_2 c_2) - M_3 l_{c3} l_2 c_{23} s_2 & 0 & 0 \\ M_3(l^2{}_{c3} c_{23} s_{23} + l_{c3} l_2 s_{23} c_2) & M_3 l_{c3} l_2 s_3 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}^2{}_1 \\ \dot{q}^2{}_2 \\ \dot{q}^2{}_3 \end{bmatrix}$$

And for the Coriolis term we have from (1.99)

$$c_2(q)[\dot{q} \quad \dot{q}]=\begin{bmatrix}2b_{112} & 2b_{113} & 2b_{123}\\ 2b_{212} & 2b_{213} & 2b_{223}\\ 2b_{312} & 2b_{313} & 2b_{323}\end{bmatrix}\begin{bmatrix}\dot{q_1}\dot{q_2}\\ \dot{q_1}\dot{q_3}\\ \dot{q_2}\dot{q_3}\end{bmatrix}$$

And we get

$$2b_{112}=\frac{1}{2}(M_{112}+M_{112}-M_{112})=\frac{M_{112}}{2}=M_{112}$$

$$2b_{113}=\frac{1}{2}(M_{113}+M_{131}-M_{131})=M_{113}$$

$$2b_{212}=\frac{1}{2}(M_{112}+M_{221}-M_{122})=0$$

$$2b_{312}=\frac{1}{2}(M_{213}+M_{231}-M_{132})=0$$

$$2b_{312}=\frac{1}{2}(M_{213}+M_{231}-M_{132})=0$$

$$2b_{123}=\frac{1}{2}(M_{123}+M_{132}-M_{231})=0$$

$$2b_{223}=\frac{1}{2}(M_{223}+M_{232}-M_{232})=M_{223}$$

$$2b_{323}=\frac{1}{2}(M_{323}+M_{332}-M_{233})=0$$

$$2b_{312}=\frac{1}{2}(M_{312}+M_{321}-M_{123})=0$$

$$2b_{313}=\frac{1}{2}(M_{313}+M_{331}-M_{133})=0$$

And the Coriolis matrix expressed as

$$\mathbf{c_2(q)[\dot{q} \quad \dot{q}]=}$$

$$\begin{bmatrix}\mathbf{2M_2l_{c2}^{2c_2s_2}+2M_3(l^2_{c3}c^2_{23}s^2_{23}+l^2_2c_2s_2+2l_{c3}s_{23}l_2c_2)-2M_3l_{c3}l_2c_{23}s_2} & \mathbf{-2M_3(l^2_{c3}c_{23}s_{23}+l_{c3}s_{23}c_2)} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{-2M_3l_{c3}l_2s_3}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$$

$$\begin{bmatrix}\dot{q_1}\dot{q_2}\\ \dot{q_1}\dot{q_3}\\ \dot{q_2}\dot{q_3}\end{bmatrix}$$

C. Gravity term

For Gravity term we have from(1.114)

$$G(q) = -\left({J_{v_1}}^t {J_{v_2}}^t {J_{v_3}}^t \right) \begin{pmatrix} m_1 g_0 \\ m_2 g_0 \\ m_3 g_0 \end{pmatrix} = -\left({J_{v_1}}^t m_1 g_0 + {J_{v_2}}^t m_2 g_0 + {J_{v_3}}^t m_3 g_0 \right) =$$

$$\begin{bmatrix} l_{c1} s_1 & -l_{c1} c_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ m_1 g_0 \end{bmatrix} +$$

$$\begin{bmatrix} -s_1(l_{c2} c_2) & c_1(l_{c2} c_2) & 0 \\ -c_1(l_{c2} c_2) & -s_1(l_{c2} c_2) & l_{c2} c_2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ m_2 g_0 \end{bmatrix} +$$

$$\begin{bmatrix} -s_1(l_{c2} c_2 + l_{c3} c_{23}) & c_1(l_{c2} c_2 + l_{c3} c_{23}) & 0 \\ -c_1(l_{c2} c_2 + l_{c3} c_{23}) & -s_1(l_{c2} c_2 + l_{c3} c_{23}) & l_{c2} c_{2+l_{c3} c_{23}} \\ -c_1 l_{c3} c_{23} & -s_1 l_{c3} c_{23} & l_{c3} c_{23} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ m_3 g_0 \end{bmatrix} = \begin{bmatrix} 0 \\ l_{c2} c_2 m_2 g_0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ (l_{c2} c_2 + l_{c3} c_{23}) m_3 g_0 \\ l_{c3} c_{23} m_3 g_0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ (l_{c2} c_2 m_2 (l_{c2} c_2 + l_{c3} c_{23}) m_3) g_0 \\ l_{c3} c_{23} m_3 g_0 \end{bmatrix}$$

d. Friction modeling

As we seen in (1.117) we can write the friction terms as

$$F(\dot{q}) = \begin{bmatrix} f_{v_1} & 0 & 0 \\ 0 & f_{v_2} & 0 \\ 0 & 0 & f_{v_3} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} +$$

$$\begin{bmatrix} f_{c_1} & 0 & 0 \\ 0 & f_{c_2} & 0 \\ 0 & 0 & f_{c_3} \end{bmatrix} \begin{bmatrix} \text{sign}(\dot{q}_1) \\ \text{sign}(\dot{q}_2) \\ \text{sign}(\dot{q}_3) \end{bmatrix} + \begin{bmatrix} (f_{s_1} - f_{c1}) & 0 & 0 \\ 0 & (f_{s_2} - f_{c2}) & 0 \\ 0 & 0 & (f_{s_3} - f_{c3}) \end{bmatrix} \begin{bmatrix} \text{sign}(\dot{q}_1) e^{-\left|\frac{\dot{q}_1}{qs}\right|^2} \\ \text{sign}(\dot{q}_2) e^{-\left|\frac{\dot{q}_2}{qs}\right|^2} \\ \text{sign}(\dot{q}_3) e^{-\left|\frac{\dot{q}_3}{qs}\right|^2} \end{bmatrix}$$

e. Equation of motion for a 3 DOF elbow manipulator

And the general dynamic's equation becomes

$$\begin{bmatrix} \mathbf{M_1 l^2_{c1} + I_{zz1} + M_2 l^{2c_2}_{c2} + I_{zz2} + I_{zz3} + M_3(l^2{}_{c3} c^2{}_{23} + l^2{}_2 c^2{}_2 + 2 l_{c3} c_{23} l_2 c_2)} & \mathbf{I_{YZ2} - I_{Yz3}} & \mathbf{-I_{Zy3}} \\ \mathbf{-I_{Yx3}} & \mathbf{M_2 l^2_{c2} + I_{yy2} + I_{yy3} + M_3(l^{2\,2}{}_{c3} + l_2{}^2 + 2 l_{c3} l_2)} & \mathbf{-I_{YZ3}} \\ \mathbf{-I_{ZX3}} & \mathbf{M_3(l^2_{c3} + l_{c3} l_2 c_3) - I_{Yy3}} & \mathbf{I_{yy3} + M_3 l c^2_3} \end{bmatrix}$$

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & 0 \\ M_2 l_{c2}{}^2 c_2 s_2 + M_3(l^2{}_{c3} c^2{}_{23} s^2{}_{23} + l^2{}_2 c_2 s_2 + l_{c3} s_{23} l_2 c_2) - M_3 l_{c3} l_2 c_{23} s_2 & 0 & 0 \\ M_3(l^2{}_{c3} c_{23} s_{23} + l_{c3} l_2 s_{23} c_2) & M_3 l_{c3} l_2 s_3 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}^2{}_1 \\ \dot{q}^2{}_2 \\ \dot{q}^2{}_3 \end{bmatrix} +$$

$$\begin{bmatrix} 2M_2 l_{c2}^{2 c_2 s_2} + 2M_3(l^2{}_{c3} c^2{}_{23} s^2{}_{23} + l^2{}_2 c_2 s_2 + 2 l_{c3} s_{23} l_2 c_2) - 2M_3 l_{c3} l_2 c_{23} s_2 & -2M_3(l^2{}_{c3} c_{23} s_{23} + l_{c3} s_{23} c_2) & 0 \\ 0 & 0 & -2M_3 l_{c3} l_2 s \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_1 \dot{q}_2 \\ \dot{q}_1 \dot{q}_3 \\ \dot{q}_2 \dot{q}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ (l_{c2} c_2 m_2 (l_{c2} c_2 + l_{c3} c_{23}) m_3) g_0 \\ l_{c3} c_{23} m_3 g_0 \end{bmatrix} + \begin{bmatrix} f_{v_1} & 0 & 0 \\ 0 & f_{v_2} & 0 \\ 0 & 0 & f_{v_3} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} + \begin{bmatrix} f_{c_1} & 0 & 0 \\ 0 & f_{c_2} & 0 \\ 0 & 0 & f_{c_3} \end{bmatrix} \begin{bmatrix} sign(\dot{q}_1) \\ sign(\dot{q}_2) \\ sign(\dot{q}_3) \end{bmatrix} +$$

$$\begin{bmatrix} (f_{s_1} - f_{c1}) & 0 & 0 \\ 0 & (f_{s_2} - f_{c2}) & 0 \\ 0 & 0 & (f_{s_3} - f_{c3}) \end{bmatrix} \begin{bmatrix} sign(\dot{q}_1) e^{-\left|\frac{\dot{q}_1}{qs}\right|^2} \\ sign(\dot{q}_2) e^{-\left|\frac{\dot{q}_2}{qs}\right|^2} \\ sign(\dot{q}_3) e^{-\left|\frac{\dot{q}_3}{qs}\right|^2} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

If we add the actuators dynamics we can write

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} j_1 G_1 & 0 & 0 \\ 0 & j_2 G_2 & 0 \\ 0 & 0 & j_3 G_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} + \begin{bmatrix} Tf_1 \\ Tf_2 \\ Tf_3 \end{bmatrix}$$

Chapter 2: Control and System Identification

2.1 Manipulator Control:

The control problem for robot manipulators is the problem of determining the time history of joint inputs required to cause the end-effector to execute a commanded motion.

There are many control techniques and methodologies that can be applied to the control of manipulators.

2.1.1 PID controller *:*

In this section we consider the simplest type of control strategy, namely, independent joint control. In this type of control each axis of the manipulator is controlled as a single input/single output (SISO) system.

A proportional–integral–derivative controller (PID controller) is a feedback control loop mechanism ,commonly used in industrial control systems. A PID controller continuously calculates an *error value* e (t) as the difference between a desired setpoint r(t) and a measured process variable y(t), and applies a correction based on proportional, integral, and derivative terms (sometimes denoted *P*, *I*, and *D* respectively) which give their name to the controller type.

$$u(t) = k_p\, e(t) + k_d \frac{de(t)}{dt} + k_i \int e(t)dt \quad (2.1)$$

With e(t)=r(t)-y(t)

where $\mathrm{k_p}$,k_i and $\mathrm{k_d}$, all non-negative, denote the coefficients for the proportional, integral, and derivative terms, In this model:

P accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.

- P accounts for past values of the error. For example, if the current output is not sufficiently strong, the integral of the error will accumulate over time, and the controller will respond by applying a stronger action
- D accounts for possible future trends of the error, based on its current rate of change

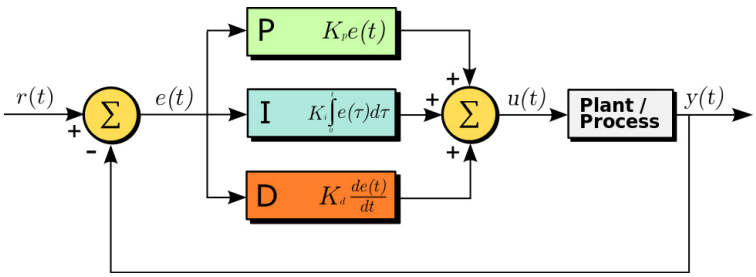


Figure (2.1): A block diagram of a PID controller in a feedback loop.

For a serial robot manipulator we applied a PID control law for each joints considering that the desired joints positions are constant, which is commonly known as independent joint control.

Now as we have mentioned on the previous chapter by using the equation (1.74) and the equation (2.1) we get :

$$\mathrm{M}(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{C}(\mathrm{q}, \dot{\mathrm{q}})\mathrm{q} + \mathrm{G}(\mathrm{q})^{\cdot} + \mathrm{F}(\dot{\mathrm{q}}) = \tau \qquad (2.2)$$

So The control law is given by

$$\tau = \mathrm{Kp}\, e(t) + \mathrm{Kd}\frac{de(t)}{dt} + \mathrm{Ki}\int e(t)\, dt \qquad (2.3)$$

with $$e(t) = qd(t) - q(t) \qquad (2.4)$$

And $$\frac{de(t)}{dt} = -\dot{q}(t) \qquad (2.5)$$

With qd(t) represent the desired angles and q(t) is the actual joints angles and τ is the torques inputs .

Denote that Kp,Kd and Ki are n*n positive diagonal matrix.

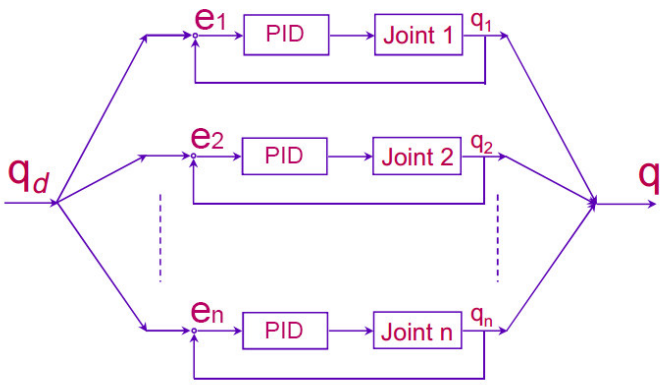


Figure (2.2) : A block diagram of independent joint control for n joints manipulator

a. Implementation and simulation

We've implemented the dynamics equations that we got in section 1.5

With choosing the manipulator parameters as follow :

m1=3; m2=1;m3=1; %masses

L1=1; L2=1;L3=1; %links lengths

Lc1=0.5; Lc2=0.5;Lc3=0.5; %center of masses lengths

izz1=0.05; izz2=0.05;izz3=0.05;iyy2=0.05;iyy3=0.05;%moment of inertia

iyz2=0.05;iyz3=0;%products of inertia

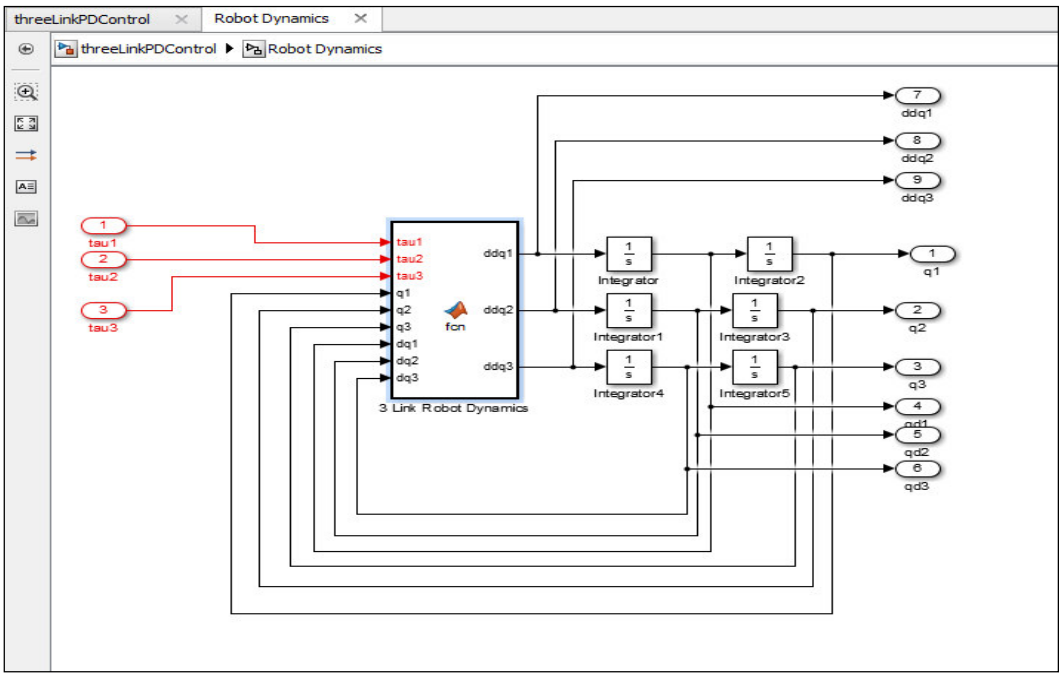


Figure (2.3): Simulink model of 3dof manipulator dynamics

Now we design a Simulink model for an independent joint control, which composed of 3 PID controller.

For the dc motors we consider them as a linear saturated gain.

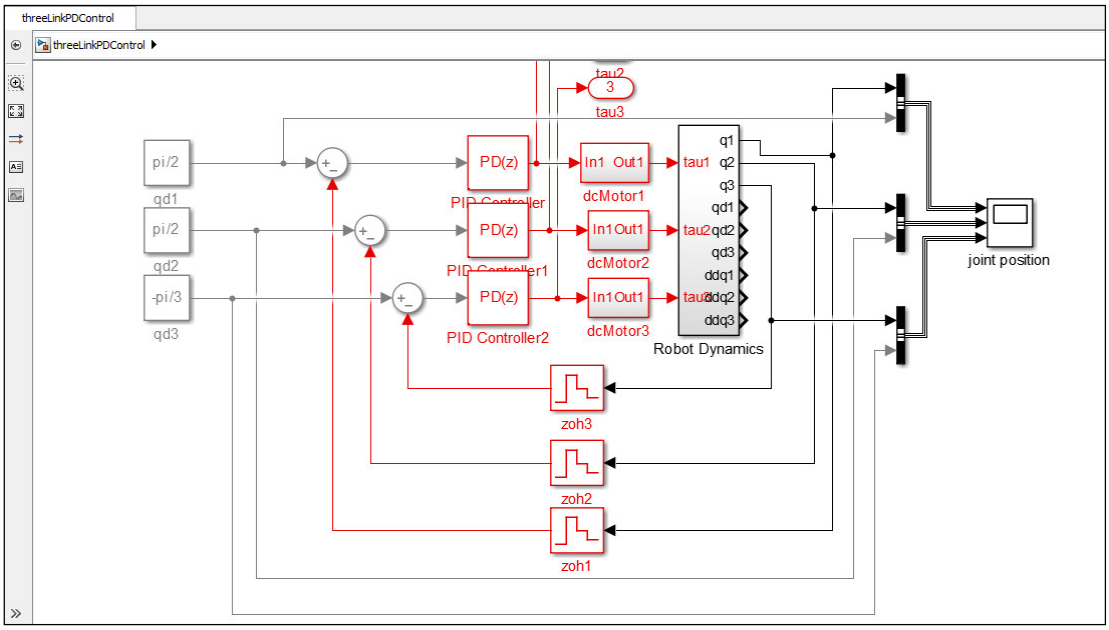


Figure (2.4): Simulink model of independent joint control for a 3 DOF manipulator dynamics

The PID gains are chosen experimentally as follow

Table 2.1 : constant Setpoint and Parameters for PID control of the first joint

Setpoint	Kp	Kd	Ki
$\pi/2$	25	10	0

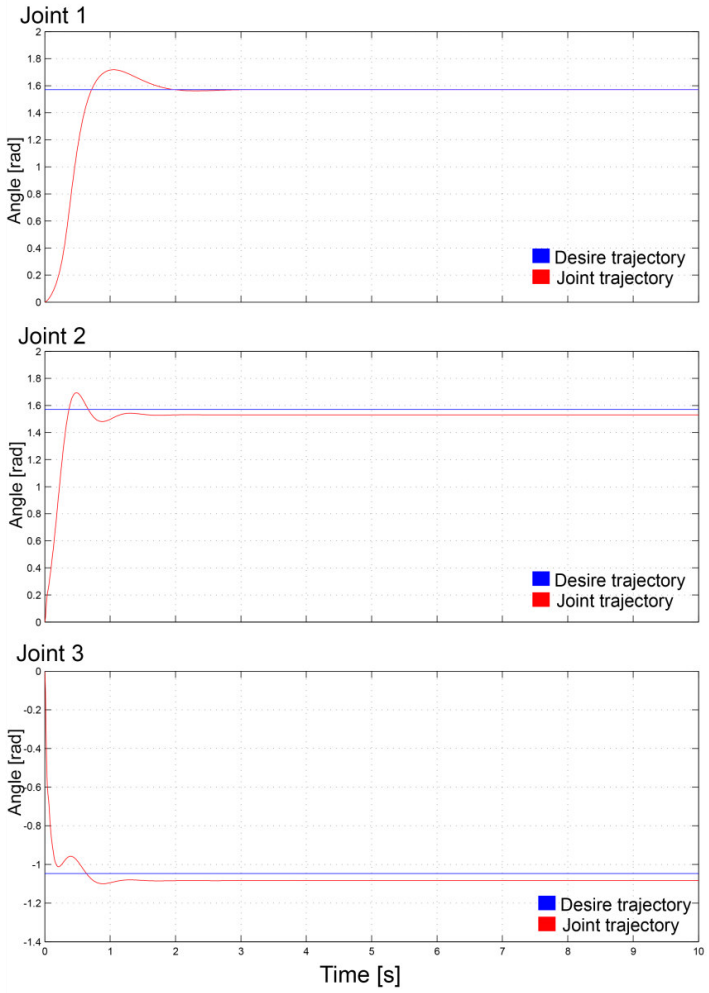
Table 2.2 : constant Setpoint and Parameters for PID control of the second joint

Setpoint	Kp	Kd	Ki
$\pi/2$	200	40	0

Table 2.3 : constant Setpoint and Parameters for PID control of the third joint

Setpoint	Kp	Kd	Ki
$-\pi/3$	200	40	0

After 10 second of simulation we get the following figures with sample time of 0.01S



Figure(2.5):simulation result of an independent joint control for a 3 DOF manipulator

2.1.2 Computed Torque Control (trajectory tracking):

A basic problem in controlling robots is to make the manipulator follow a preplanned desired trajectory. Before the robot can do any useful work, we must position it in the right place at the right instances. In this section we discuss computed-torque control

The PID controller is not an efficient controller to control a manipulator because the torques output signal that is generated by the PID controller is not dependant on the other joints. The motion of the other links may apply considerable torque and force to the joint. This unpredicted torque may not be compensated with the PID controller therefore the performance of the controller drops when the robot performs in high speeds, a much better method to control the robot is to calculate the inverse dynamics of the robot and consider the computed torques to generate control signal.

By this method the robot performs well even in high speeds. The problem with this controller is that in order to calculate the inverse dynamics of the robot, its parameters must be determined and very nonlinear equations should be solved. Feedback linearization method is one of the most common methods for controlling a robot and is widely used in the industry.

We consider the dynamic equation of the manipulator

$$\mathrm{M}(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{C}(\mathrm{q}, \dot{\mathrm{q}})\mathrm{q} + \mathrm{G}(\dot{\mathrm{q}}) + \mathrm{F}(\dot{\mathrm{q}}) = \tau \quad (2.6)$$

Suppose that a desired trajectory qd(t) has been selected for the arm motion ,and as we have defined the output or the *tracking error* as :

$$\mathrm{e}(\mathrm{t}) = \mathrm{qd}(\mathrm{t}) + \mathrm{q}(\mathrm{t}) \quad (2.7)$$

To demonstrate the influence of the input $\tau(t)$ on the tracking error, we have to use the feedback Linearization.

a. Feedback Linearization

The idea of feedback linearization based on canceling the nonlinearities and Imposing a desired linear dynamics. can be simply applied to a class of nonlinear systems can be represented as follow [21]

$$\mathrm{x}^{(\mathrm{n})} = \mathrm{f}(\mathrm{x}) + \mathrm{b}(\mathrm{x})\mathrm{u} \quad (2.8)$$

Denote that $f(x)$and$b(x)$are nonlinear function and u is the control input

For systems which can be expressed as we mentioned above in *(2.8)*, we can use the control input

$$u = \frac{1}{b}(V - f) \qquad (2.9)$$

We can cancel the nonlinearities and obtain the simple input-output relation

$$x^{(n)} = V \qquad (2.10)$$

Thus, the control law

$$V = -K_0 x - K_1 \dot{x} \dots - K_{n-1} x^{(n-1)} \qquad (2.11)$$

With$K_i$positive gains, and from(2.10) and (2.11)we get

$$x^{(n)} + K_0 x + K_1 \dot{x} \dots + K_{n-1} x^{(n-1)} = 0 \qquad (2.12)$$

Which implies that $x(t) — > 0$.For tasks involving the tracking of a desired output $x_d(t)$, the control law becomes

$$V = {x_d}^{(n)} - K_0 e - K_1 \dot{e} \dots - K_{n-1} e^{(n-1)}$$

where$e = x(t) - x_d(t)$is the tracking error) leads to exponentially convergent tracking. Note that similar results would be obtained if the scalar x was replaced by a vector and the scalar b by an invertible square matrix.

For Applying a feedback linearization on robot manipulator dynamics we get from(2.6)

$$\ddot{q}(t) = M\big(q(t)\big)^{-1}\Big(\tau(t) - C\big(\dot{q}(t), q(t)\big)\dot{q}(t) - F\big(\dot{q}(t)\big) - G\big(q(t)\big)\Big) \quad (2.13)$$

We have $V = \ddot{q}(t)$ (2.14)

And this result

$$\begin{cases} b = M(q)^{-1} & (2.15) \\ f = M(q)^{-1}\left(-C(\dot{q},q)\dot{q} - F(\dot{q}) - G(q)\right) & (2.16) \end{cases}$$

We choose $$V = \ddot{q}d - K0\dot{e}(t) - K1e(t) \quad (2.17)$$

So the corresponding feedback linearization control law is given by (2.18)

$$\tau = \frac{1}{b}(V - f)$$

K1 and K0 represent the proportional the derivative gains we can name it Kp and Kd respectively, and that's brings us to following input control law [22]

$$\tau = \boldsymbol{M(q)}\left(\boldsymbol{\ddot{q}d - K\dot{e} - Kpe}\right) + \boldsymbol{C(\dot{q},q)\dot{q}} + \boldsymbol{F(\dot{q})} + \boldsymbol{G(q)} \quad (2.19)$$

With $$\boldsymbol{e = x(t) - x_d(t)}$$

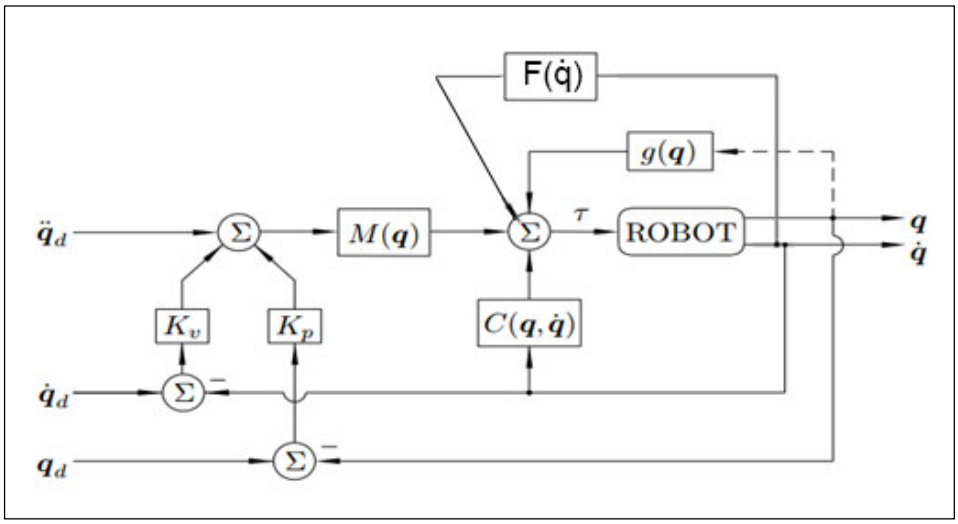


Figure (2.6): A block diagram of Computed-torque control.

For Stability analysis we place the control law we got in (2.19)in the dynamic equation of the manipulator, and we get [23]

$$\ddot{e} + K\dot{e} + Kpe = 0 \quad (2.20)$$

Which verified the exponentially convergence of the tracking error ($e => 0$).

b. Implementation and simulation

We've implemented the dynamics equations that we got in section 1.5

With choosing the manipulator parameters as follow :

m1=3; m2=1;m3=1; %mass with kg

L1=0.2; L2=0.2;L3=0.2; %link length with meters

Lc1=0.5; Lc2=0.5;Lc3=0.5; %center of mass length with meters

izz1=0.05; izz2=0.05;izz3=0.05;iyy2=0.05;iyy3=0.05;%moment of inertia

iyz2=0.05;iyz3=0; % products of inertia

and we apply a computed torque control law on it , which is represented by dynamic inversion block, and 3 PID blocks as we mention in (2.19).

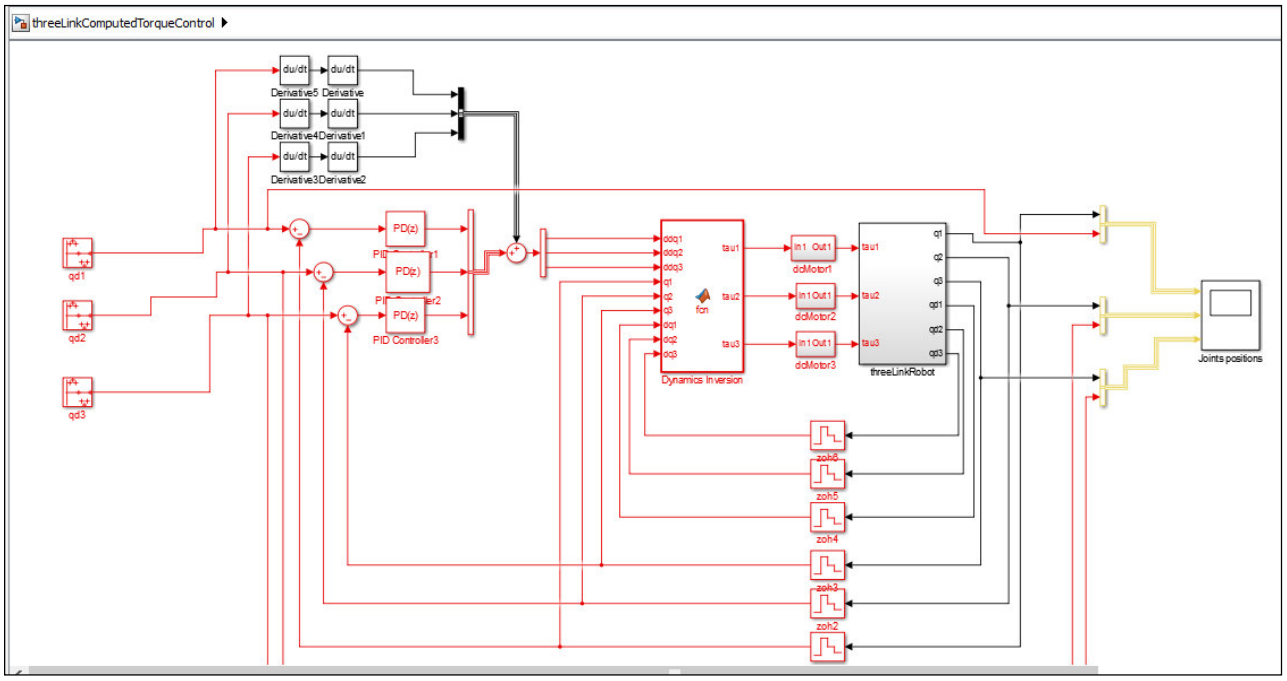


Figure (2.7): Simulink model of computed torque control for a 3 DOF manipulator dynamics

The PID gains are chosen experimentally as follow

Table 2.4 : the PID Parameters for control of the first joint

Kp	Kd	Ki
20^2	40	0

Table 2.5 : the PID Parameters for control of the second joint

Kp	Kd	Ki
50^2	100	0

Table 2.6 : the PID Parameters for control of the third joint

Kp	Kd	Ki
50^2	100	0

After 15 second of simulation we get the following figures with sample time of 0.01S

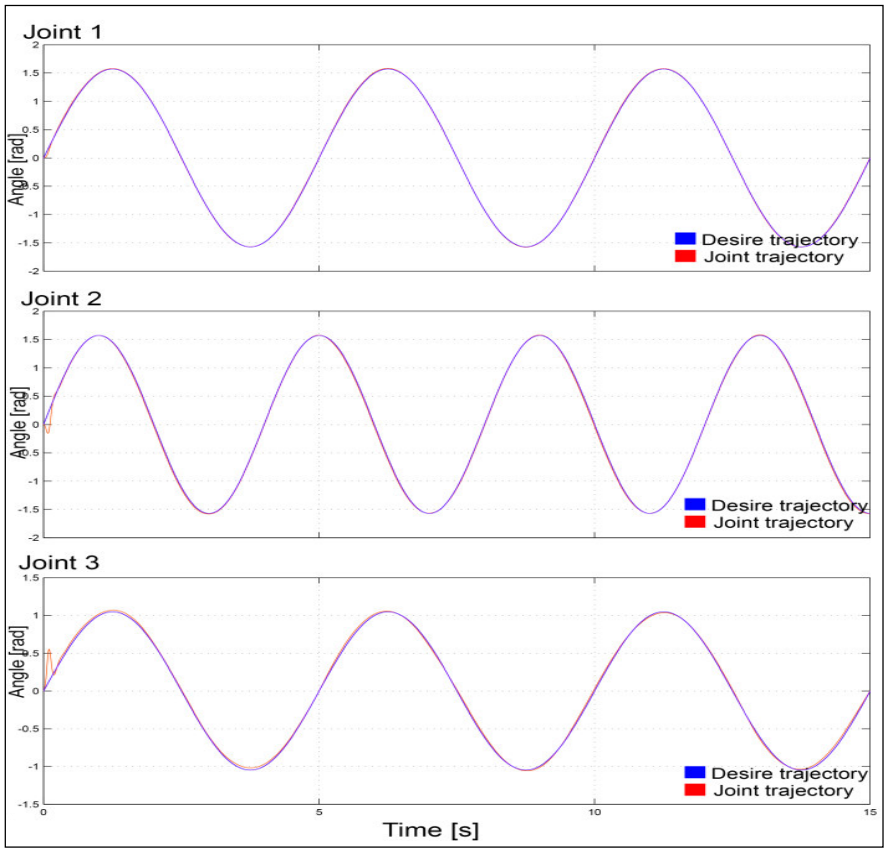


Figure (2.8): Simulink model of computed torque control for a 3 DOF manipulator dynamics

As we seen in Figure 2.8 the joints trajectories follow the desired trajectories and the tracking condition is verified as we demonstrate in (2.19).

2.2 identification and parameters estimation:

2.2.1 Introduction :

This section gives a short introduction to system identification in general, and to the identification of robot manipulators in particular.

system identification is the mathematical mechanism which allow us to build mathematical models of dynamical systems from measured data, by modeling and estimate the real parameters of the system and we name it parametric, or by building a non- parametric model which have the same behavior as the real system and then estimate its parameters.

the identification experiment can be performed in *open loop* or *closed loop*.

Identification of a system not subject to feedback control, is known as open-loop system which is illustrated in Figure 2.9. This system has input *u*, output *y*, and is affected by a disturbance *v*. The disturbance can include measurement noise as well as external system inputs, not included in *u*.

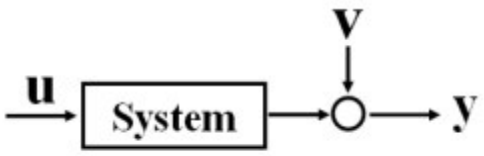
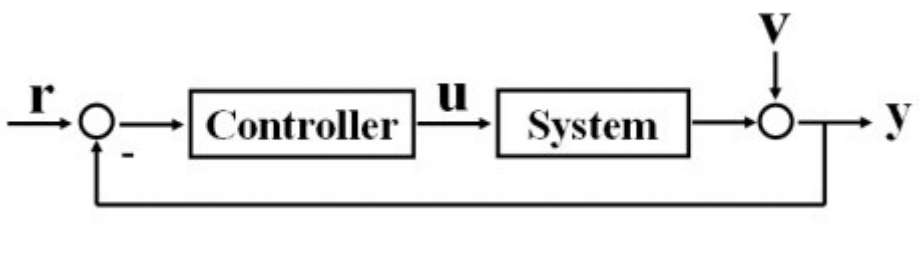


Figure (2.9): An open-loop system.

An identification experiment on a system subject to feedback control, is known as closed-loop system which is shown in Figure 2.10 where *r* is the reference signal for the system. A reason for performing a closed-loop experiment could be that the system is unstable, and must be controlled in order to remain stable. This is typically the case for a robot manipulator.



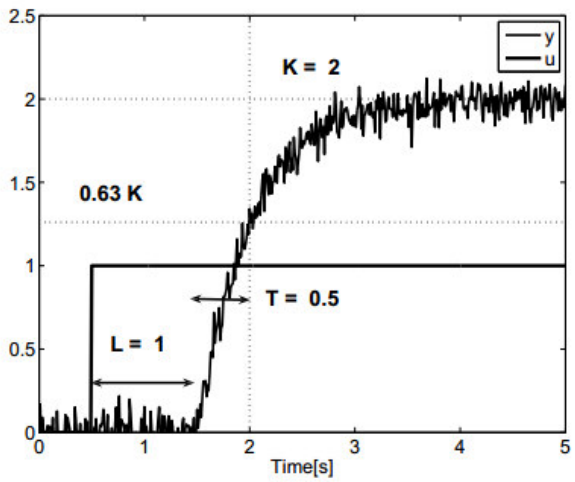
Figure(2.10): A closed-loop system.

Moreover, models can be described as *continuous-time models* or *discrete-time models* although the measurements, $u(t)$ and $y(t)$, are normally represented as sampled, discrete-time, data. It is assumed that the reader has a basic knowledge of linear system theory for continuous-time and discrete-time systems.[24]

2.2.2 Nonparametric Models:

Examples of nonparametric models in the *time-domain* are impulse responses or step responses. Such models consist of vectors of system outputs and the corresponding time stamps. An example of a step response of a first-order system with a time-delay is shown in Figure 2.3. The measured output is affected by measurement noise. The nonparametric step response model can in this case be described by a parametric transfer function model

$$G(s) = \frac{K}{sT+1} e^{-Ls} \quad (2.21)$$



Figure(2.11): Step response of a first order process with delay.

This three-parameter model is often used to describe systems in the process industry. The parametric model (2.1) can be identified by inspection of the step response according to Figure 2.11 This model can then be used for tuning of a PI- or a PID controller .[25]

2.2.3 Parametric Models:

A parametric model is a model described as, e.g., differential or difference equations. System identification is one route for obtaining a parametric model of a system. Another route is physical modeling, i.e., deriving a mathematical model from the basic laws of physics.

If the parameters of a physical model are known with sufficient accuracy, we get a white-box model, where both the model structure and the model parameters are known.

A gray-box model is a physical model where the model structure is known but the physical parameters are unknown or only partly known. Identification of parameters or parameters estimation in this case is called gray-box identification, which our case in robot manipulator identification.[26]

a the least squares method LS:

Least-squares estimation methods have been used in many types of parameter identification schemes [Astrom and Wittenmark 1989] [27], Least-squares method can be applied to large variety of problems. It's particularly simple for mathematical model that can be write in the form [37]

$$y(t) = \varphi_1(t)\Theta_1^0 + \varphi_2(t)\Theta_2^0 + \cdots + \varphi_n(t)\Theta_n^0 = \varphi^T(i)\Theta^0 \quad (2.22)$$

We can write it as

$$Y = \Phi\Theta \quad (2.23)$$

Where y is the observed variable, $\Theta_1^0, \Theta_2^0 \dots \Theta_n^0$are parameters of the model to be determined, and $\varphi_1, \varphi_2 \dots \varphi_n$are known functions that may depend on other known variables like output variable and its derivative. and we have the vectors

$$\varphi^T(i) = \{\varphi_1(i), \varphi_2(i) \dots \varphi_n(i)\} \quad (2.24)$$

$$\Theta^0 = \{\Theta_1^0, \Theta_2^0 \dots \Theta_n^0\} \quad (2.25)$$

The model is indexed by the variable i, which often denotes time .it will be assumed initially that the index set is a discrete set .

the variables φ_i are called the regression variables and φ^Tvector called regressor, pairs of observations and regressors$\{y(i), \varphi_1(i), i = 1,2, \dots t\}$are obtained from experiment .

to determine the parameters vector Θ^0we define a quadratic cost function, and the parameters should be chosen to minimize this function

$$V(\Theta, t) = \frac{1}{2}\sum_{i=1}^{t}(y(i) - \varphi^T(i)\Theta)^2 \quad (2.26)$$

Since the measured variable $y$is linear in parameters $\theta^0$and least squares criterion is quadratic, the problem admits an analytical solution. We define the notations

$$Y(t) = \{y(1)y(2) \dots y(t)\}^T \quad (2.27)$$

$$E(t) = \{e(1)e(2) \dots e(t)\}^T \quad (2.28)$$

$$\Phi(t) = \begin{Bmatrix} \varphi^T(1) \\ \vdots \\ \varphi^T(t) \end{Bmatrix} \quad (2.29)$$

$$\mathrm{P}(t) = \left(\Phi(t)^{\mathrm{T}}\Phi(t)\right)^{-1} = \left(\sum_{i=1}^{t} \varphi(i)\varphi^T(i)\right)^{-1} \quad (2.30)$$

Where the error $\mathrm{e(i)}$are define by

$$e(i) = y(i) - \varphi^T(i)\theta \quad (2.31)$$

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t} e(i)^2 = \frac{1}{2}E^T E = \frac{1}{2}\|E\|^2 \quad (2.32)$$

Where Ecan be written as

$$E = Y - \hat{Y} = Y - \Phi\theta \quad (2.33)$$

The solution to the least-squares problem is given by the following demonstration

$$\Phi^{\mathrm{T}}\Phi\hat{\theta} = \Phi^{\mathrm{T}}\mathrm{Y} \quad (2.34)$$

if the matrix $\Phi^{\mathrm{T}}\Phi$is nonsingular, the minimum is unique and given by

$$\hat{\theta} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{Y} \quad (2.35)$$

b the recursive least squares method RLS:

The Recursive least squares (RLS) is an adaptive filter which recursively finds the coefficients that minimize a squares cost, these coefficients define as the parameters of the system that we are estimate, the algorithm based on real-time parameters estimation, which is commonly used in adaptive control strategy .

In adaptive controller the observations are obtained sequentially in real time, computation of the least squares estimate can be arranged in such a way that the result obtained at time t-1 can be used to get the estimates at time t.

Let $\widehat{\Theta}(\mathrm{t}-1)$denote the least-squares estimate based on t-1 measurements. Assume that $\Phi(t)$has full rank, that means that the matrix $\Phi^{\mathrm{T}}\Phi$ is nonsingular for all t>t0, the least-squares estimate $\widehat{\Theta}(\mathrm{t})$then satisfies the recursive equations, it follows from the definition of $\mathrm{P}(t)$in Eq (2.30)

$$\mathrm{P}(t)^{-1} = \Phi(t)^{\mathrm{T}}\Phi(t) = \sum_{i=1}^{t} \varphi(i)\varphi^{T}(i) = \sum_{i=1}^{t-1} \varphi(i)\varphi^{T}(i) + \varphi(t)\varphi^{T}(t)$$

$$= \mathrm{P}(t-1)^{-1} + \varphi(t)\varphi^{T}(t) \quad (2.36)$$

The least squares estimate $\widehat{\Theta}(\mathrm{t})$is given by (2.36),so we get

$$\widehat{\Theta}(\mathrm{t}) = \left(\sum_{\mathrm{i}=1}^{\mathrm{t}-1} \varphi(\mathrm{i})\varphi^{\mathrm{T}}(\mathrm{i})\right)^{-1} \left(\sum_{\mathrm{i}=1}^{\mathrm{t}} \varphi(\mathrm{i})\mathrm{y}(\mathrm{i})\right)$$

$$= P(t)\left(\sum_{i=1}^{t} \varphi(i)y(i)\right) \quad (2.37)$$

Yield

$$\widehat{\Theta}(\mathrm{t}) = \mathrm{P}(\mathrm{t})\left(\sum_{\mathrm{i}=1}^{\mathrm{t}-1} \varphi(\mathrm{i})\mathrm{y}(\mathrm{i}) + \varphi(\mathrm{t})\mathrm{y}(\mathrm{t})\right) (2.38)$$

it follows from (2.37)and (2.36)that

$$\sum_{i=1}^{t} \varphi(i)y(i) = \mathrm{P}(t-1)^{-1}\hat{\theta}(t-1) = \mathrm{P}(t)^{-1}\hat{\theta}(t-1) - \varphi(t)\varphi^{T}(t)\hat{\theta}(t-1)$$
(2.39)

The estimate at time t can now be written as

$$\widehat{\Theta}(\mathrm{t}) = \widehat{\Theta}(\mathrm{t}-1) - \mathrm{P}(\mathrm{t})\varphi(\mathrm{t})\varphi^{\mathrm{T}}(\mathrm{t})\widehat{\Theta}(\mathrm{t}-1) - \mathrm{P}(\mathrm{t})\varphi(\mathrm{t})\mathrm{y}(\mathrm{t})$$

$$= \widehat{\Theta}(\mathrm{t}-1) + \mathrm{P}(\mathrm{t})\varphi(\mathrm{t})\left(\mathrm{y}(\mathrm{t}) - \varphi^{\mathrm{T}}(\mathrm{t})\widehat{\Theta}(\mathrm{t}-1)\right) \quad (2.40)$$

Where

$$K(t) = P(t)\varphi(t) \quad (2.41)$$

$$e(t) = y(t) - \varphi^T(t)\hat{\theta}(t-1) \quad (2.42)$$

The residual $\mathrm{e(t)}$can be interpreted as the error in predicting the signal $\mathrm{y(t)}$one step ahead based on estimate $\hat{\theta}(t-1)$.

To proceed, it is necessary to derive a recursive equation for $\mathrm{P(t)}$rather than for $\mathrm{P}(t)^{-1}$as in Eq (2.36) **.** we apply the matrix inversion lemma and we get

$$\mathrm{P(t)} = \left(\Phi(\mathrm{t})^{\mathrm{T}}\Phi(\mathrm{t})\right)^{-1}\left(\Phi(\mathrm{t}-1)^{\mathrm{T}}\Phi(\mathrm{t}-1) + \varphi(\mathrm{t})\varphi^{\mathrm{T}}(\mathrm{t})\right)^{-1}$$

$$= \mathrm{P(t-1)}^{-1} + \left(\varphi(\mathrm{t})\varphi^{\mathrm{T}}(\mathrm{t})\right)^{-1}$$

$$= \mathrm{P(t-1)} - \mathrm{P(t-1)}\varphi(\mathrm{t})\left(\mathrm{I} + \varphi^{\mathrm{T}}(\mathrm{t})\mathrm{P(t-1)}^{-1}\varphi(\mathrm{t})\right)^{-1}\varphi^{\mathrm{T}}(\mathrm{t})\mathrm{P(t-1)} \quad (2.43)$$

This implies that

$$K(t) = P(t)\varphi(t) = P(t-1)\varphi(t)\left(I + \varphi^T(t)\mathrm{P}(t-1)^{-1}\varphi(t)\right)^{-1} \quad (2.44)$$

And that arrive us

$$\mathrm{P(t)} = \left(\mathrm{I} - \mathrm{K(t)}\varphi^{\mathrm{T}}(\mathrm{t})\right)\mathrm{P(t-1)} \quad (2.45)$$

So the Recursive least squares estimation represented by these equation[37]

$$\begin{cases} \hat{\theta}(t) = \hat{\theta}(t-1) + P(t)\varphi(t)\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \\ K(t) = P(t)\varphi(t) = P(t-1)\varphi(t)\left(I + \varphi^T(t)\mathrm{P}(t-1)^{-1}\varphi(t)\right)^{-1} \\ P(t) = \left(I - K(t)\varphi^T(t)\right)P(t-1) \end{cases} \quad (2.46)$$

2.2.4 Applying the LS on the chosen robot:

The first thing we have to setup the model we get in section **1.5**, we derive this equation from the dynamical model matrices, each equation describe a joint model,

which is given by an input voltage U_i , the velocities and accelerations of the joints and several parameters which represent the masses, inertias, lengths, friction coefficients,motors parameters .

$$U = Jq + M(q)\ddot{q} + C(q, \dot{q})q + G(q) + F(\dot{q}) + Tf \qquad (2.47)$$

Denote that we neglect the term of stribeck effect, because its nonlinearity in the parameters .

$$\begin{aligned}
U_1 = {} & (j_1 G_1 + m_1 lc_1^2 + I_{zz1} + m_2 lc_2^2 c_2^2 + I_{zz2} + m_3 lc_3^2 c_{23}^2 + m_3 lc_2^2 c_2^2 + 2m_3 lc_3^2 l_2\, c_{23} c_2 + Izz_3)\ddot{q}_1 + (Iyz_2 + Iyz_3)\ddot{q}_2 \\
& + Iyz_3 \ddot{q}_3 - 2(m_2 lc_2^2 s_2 c_2 + m_3 lc_3^2 s_{23} c_{23} + m_3 l_2^2 s_2 c_2 + m_3 lc_3\, l_2 s_{23} c_2 + m_3 lc_3\, l_2 c_{23} s_2)\dot{q}_1 \dot{q}_2 \\
& - 2(m_3 lc_3^2 s_{23} c_{23} + m_3 lc_3 s_{23} c_2)\dot{q}_1 \dot{q}_3 + f_{v_1} \dot{q}_1 + f_{c_1} sign(\dot{q}_1) + Tf_1 \\
U_2 = {} & (Iyz_2 + Iyz_3)\ddot{q}_1 + (j_2 G_2 + m_2 lc_2^2 + Iyy_2 + m_3 lc_3^2 + m_3 l_2^2 + 2m_3 lc_3 l_2 c_2 + Iyy_3)\ddot{q}_2 + m_3 lc_3^2 + m_3 lc_3 l_2 c_3 \\
& + Iyy_3)\ddot{q}_3 + (m_2 lc_2^2 c_2 s_2 + m_3 lc_3^2 c_{23} s_{23} + m_3 l_2^2 s_2 c_2 + m_3 lc_3 l_2 c_{23} s_{23} + m_3 lc_3\, l_2\, c_{23} s_2)\dot{q}_1 \\
& - (2m_3 lc_3 l_2 s_3)\dot{q}_1 \dot{q}_3 + (lc_2 c_2 m_2 + (lc_3 c_{23} + lc_2 c_2)m_3)g + f_{v_2} \dot{q}_2 + f_{c2}\, sign(\dot{q}_2) + Tf_2 \\
U_3 = {} & (Iyz_3)\ddot{q}_1 + (m_3 lc_3^2 + m_3 lc_3 l_2 c_3 + Iyy_3)\ddot{q}_2 + (j_3 G_3 + m_3 lc_3^2 + Iyy_3)\ddot{q}_3 + (m_3 lc_3^2 c_{23} s_{23} + m_3 lc_3 l_2 c_2 s_{23})\dot{q}_1 \\
& + (m_3 lc_3 l_2 s_2)\dot{q}_2 + lc_3 c_{23} m_3 g + f_{v3}\, \dot{q}_2 + f_{c3}\, sign(\dot{q}_3) + Tf_3
\end{aligned} \qquad (2.48)$$

Second thing we have to arrange them such a way they are linear in the parameters as we seen in Eq (2.3), to allow us to apply the least squares parameters estimation, so we write as follow $\boldsymbol{Y = \Phi\theta}$

So we have the output vector$\boldsymbol{Y}$,which represent the PWM inputs to the motors

$$\boldsymbol{Y} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \qquad (2.49)$$

And we extract the parameters we want to estimate, $\boldsymbol{\theta}$represent tha parameters vector

$$\theta = \begin{bmatrix} \mathbf{j_1 G_1 + m_1 lc_1^2 + I_{zz1} + I_{zz2} + I_{zz3}} \\ \mathbf{m_2 lc_2^2 + m_3 l_2^2} \\ \mathbf{m_2 lc_2^2 + m_3 l_2} \\ \mathbf{m_3 lc_3} \\ \mathbf{m_3 lc_3^2} \\ \mathbf{m_3 lc_3^2 l_2} \\ \mathbf{Iyz_2} \\ \mathbf{Iyz_3} \\ \mathbf{Iyy_2 + j_2 G_2} \\ \mathbf{Iyy_3} \\ \mathbf{f_{v1}} \\ \mathbf{f_{c1}} \\ \boldsymbol{f_{v2}} \\ \boldsymbol{f_{c2}} \\ \boldsymbol{f_{v3}} \\ \boldsymbol{f_{c3}} \\ \mathbf{Tf_1} \\ \mathbf{Tf_2} \\ \mathbf{Tf_3} \\ \mathbf{j_3 G_3} \end{bmatrix} \quad (2.50)$$

then the regresseur matrix $\boldsymbol{\Phi}$given by

$$\Phi = [\varphi_1 \quad \varphi_2 \quad \varphi_3\varphi_4 \quad \varphi_5 \quad \varphi_6\varphi_7 \quad \varphi_8 \quad \varphi_9\varphi_{10} \quad \varphi_{11} \quad \varphi_{12}\varphi_{13} \quad \varphi_{14}\varphi_{15} \quad \varphi_{16}\varphi_{17} \quad \varphi_{18}\varphi_{19} \quad \varphi_{20}] \quad (2.51)$$

With the regresseurs

$$\varphi_1 = \begin{bmatrix} \ddot{q}_1 \\ 0 \\ 0 \end{bmatrix}, \varphi_2 = \begin{bmatrix} c_2^2\ddot{q}_1 + 2s_2c_2\dot{q}_1\dot{q}_2 \\ \ddot{q}_2 + s_2c_2\dot{q}_1 \\ 0 \end{bmatrix}, \varphi_3 = \begin{bmatrix} 0 \\ c_2 g \\ 0 \end{bmatrix}, \varphi_4 = \begin{bmatrix} 0 \\ c_{23} g \\ c_{23} g \end{bmatrix}, \varphi_5 = \begin{bmatrix} c_{23}^2\dot{q}_1 - 2s_{23}c_{23}\dot{q}_1\dot{q}_2 - 2s_{23}c_{23}\dot{q}_1\dot{q}_3 \\ s_{23}c_{23}\dot{q}_1 + \ddot{q}_2 + \ddot{q}_3 \\ \ddot{q}_3 + \ddot{q}_2 + s_{23}c_{23}\dot{q}_1 \end{bmatrix}$$

$$\varphi_6 = \begin{bmatrix} 2c_{23}c_2\ddot{q}_1 - 2c_{23}c_2\dot{q}_1\ddot{q}_2 - 2c_{23}s_2\ddot{q}_1\ddot{q}_2 - 2s_{23}c_2\ddot{q}_1\ddot{q}_2 \\ 2c_2\ddot{q}_2 + c_3\ddot{q}_3 + (s_{23}c_2 + c_{23}s_2)\dot{q}_1 - 2s_3\dot{q}_1\dot{q}_3 \\ c_3\ddot{q}_2 + s_{23}c_2\dot{q}_1 + s_3\ddot{q}_2 \end{bmatrix}, \varphi_7 = \begin{bmatrix} \ddot{q}_2 \\ \ddot{q}_1 \\ 0 \end{bmatrix}, \varphi_8 = \begin{bmatrix} \ddot{q}_2 + \ddot{q}_3 \\ \ddot{q}_1 \\ \ddot{q}_1 \end{bmatrix}, \varphi_9 = \begin{bmatrix} 0 \\ \ddot{q}_2 \\ 0 \end{bmatrix},$$

$$\varphi_{10} = \begin{bmatrix} 0 \\ \ddot{q}_3 + \ddot{q}_2 \\ \ddot{q}_3 + \ddot{q}_2 \end{bmatrix} \varphi_{11} = \begin{bmatrix} \dot{q}_1 \\ 0 \\ 0 \end{bmatrix}, \varphi_{12} = \begin{bmatrix} \text{sign}(\dot{q}_1) \\ 0 \\ 0 \end{bmatrix}, \varphi_{13} = \begin{bmatrix} 0 \\ \dot{q}_2 \\ 0 \end{bmatrix}, \varphi_{14} = \begin{bmatrix} 0 \\ \text{sign}(\dot{q}_2) \\ 0 \end{bmatrix}, \varphi_{15} = \begin{bmatrix} 0 \\ 0 \\ \dot{q}_3 \end{bmatrix}, \varphi_{16} = \begin{bmatrix} 0 \\ 0 \\ \text{sign}(\dot{q}_3) \end{bmatrix}$$

$$\varphi_{17=}\begin{bmatrix}1\\0\\0\end{bmatrix}, \varphi_{18=}\begin{bmatrix}0\\1\\0\end{bmatrix}, \varphi_{19=}\begin{bmatrix}0\\0\\1\end{bmatrix}, \varphi_{20=}\begin{bmatrix}0\\0\\\ddot{q}_3\end{bmatrix}$$

Now we apply the LS estimation to get the estimated parameters vector $\hat{\theta}$ as we define in (2.35)

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{Y}$$

Chapter 3: Vision

This chapter will describe theories necessary for understanding the rest of the thesis. The chapter is divided in image processing and vision system theory.

3.1 Image Processing

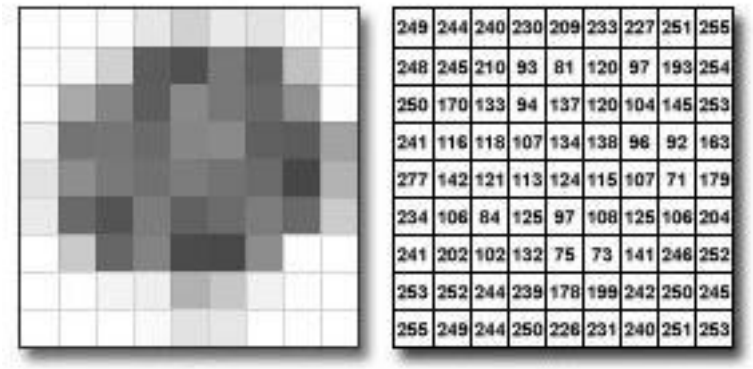
Image processing is a set of computational techniques for analyzing, enhancing, compressing, and reconstructing images. The main components are importing, in which an image is captured through scanning or digital photography, analysis and manipulation of the image, accomplished using various specialized software applications, and output. Image processing has extensive applications in many areas, including astronomy, medicine, industrial robotics, and remote sensing by satellites). Image processing for robot vision will improve products quality, save time and reduce labor cost.

In this section we will present some image processing tools that we used in our work.

3.1.1 The image

An image is defined as a two-dimensional function $f(x, y)$.where x and y are spatial (plane) coordinates. The intensity or gray level of the image at the point of coordinates (x, y) is the amplitude of f at that point .

This image known as grayscale image or intensity level image, The typical range of intensity values for each pixel is , 0 to 255, is based on taking a binary number 8 bits that can hold a value from 0 to 255.



249	244	240	230	209	233	227	251	255
248	245	210	93	81	120	97	193	254
250	170	133	94	137	120	104	145	253
241	116	118	107	134	138	96	92	163
277	142	121	113	124	115	107	71	179
234	106	84	125	97	108	125	106	204
241	202	102	132	75	73	141	246	252
253	252	244	239	178	199	242	250	245
255	249	244	250	226	231	240	251	253

Figure 3.1 a grayscale image and its matrix of intensity level values

3.1.2 RGB color model

The RGB Image has 3 planes of intensity levels, red, green and blue plane, so each pixel in the image has 3 values, and these values describe the color of the pixel.

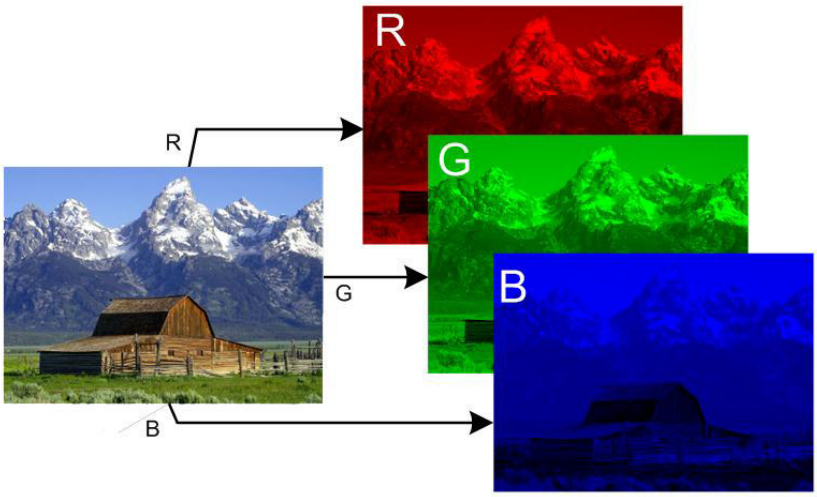


Figure (3.2): RGB image and its 3 colors plane, red, green and red plane

RGB colors is usually represented as axes of a 3D cube as shown in Fig. 3.3 The cubic represents all possible colors. A specific color is represented by three values to be summed: (R, G, B). Black is (0,0,0) or 0+0+0, or no measurements on any of the

three color planes. White is (255, 255, 255). The pure colors of red, green, and blue are represented by (255,0,0), (0,255,0), and (0,0,255) respectively. This is the same as in colorgraphics.

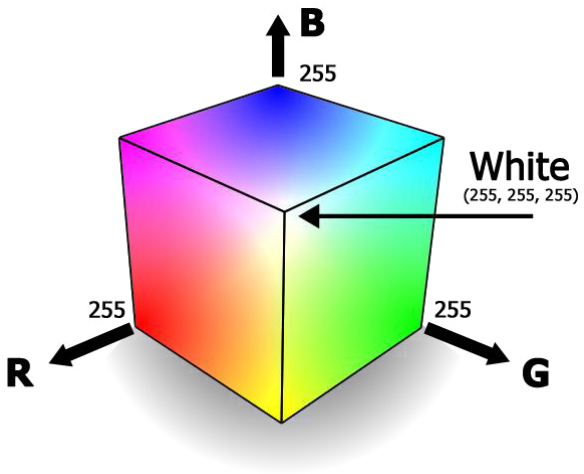


Figure (3.3): The RGB color model mapped to a cube

3.1.3 HSV color model

HSV is a three-dimensional space in that it has three variables, but it is definitely not a cube representation, more of a cone as seen in Fig. 3.3. The hue, or color, is measured in degrees from 0 to 360. Saturation and intensity are real numbers between 0 and 1. These are generally scaled to 8-bit numbers. Accordingly, red is both 0 and 255, orange is 17, green is at 85, blue is 170, with magenta at 200.

HSV space is very used in robotics and object detection, if we have a object with a specific color , in HSV space we can detect the color even if the light is not uniform, by selecting a rang in the hue plane ,which is very hard to do it in RGB space.

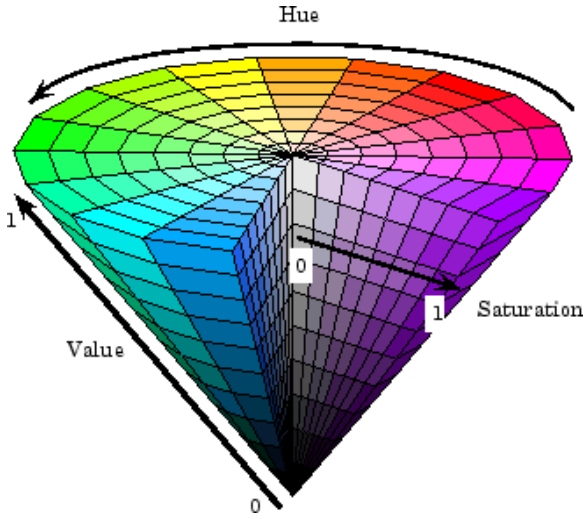


Figure (3.4): The HSV color model cone describe the Hue, Saturation and Value ranges

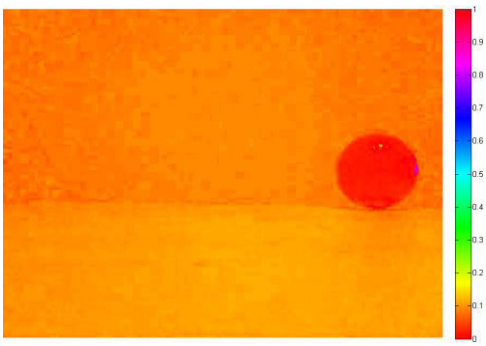
3.1.4 Thresholding:

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity is less than some fixed constant T, or a white pixel if the image intensity is greater than that constant and the result image called binary image

$$b(x, y) = \begin{cases} 1 \text{ if } f(x,y) \geq T \\ 0 \text{ if } f(x,y) < T \end{cases} \qquad (3.1)$$

This application very useful to extract an object by thresholding using a hue rang for example.

a.RGB image



b.HSV image

c. Binary image

Figure(3.5) : describe transformation from RGB to HSV then getting a Binary image by thresholding the HSV image

3.1.5 Edge detection

Edge detection includes a variety of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed *edges*. The same problem of finding discontinuities in one-dimensional signals is known as step detection and the problem of finding signal discontinuities over time is known as change detection.

Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction [30]

a. Canny Edge detector

John Canny considered the mathematical problem of deriving an optimal smoothing filter given the criteria of detection, localization and minimizing multiple responses to a single edge.[31] He showed that the optimal filter given these assumptions is a sum of four exponential terms. He also showed that this filter can be well approximated by first-order derivatives of Gaussians.

Here is an example of a 5×5 Gaussian filter, used to create the adjacent image, with standard deviation of σ=1.4. (The * denotes a convolution operation.)

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}. \qquad \textbf{(3.2)}$$

It is important to understand that the selection of the size of the Gaussian kernel will affect the performance of the detector. The larger the size is, the lower the detector's sensitivity to noise. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is a good size for most cases, but this will also vary depending on specific situations.

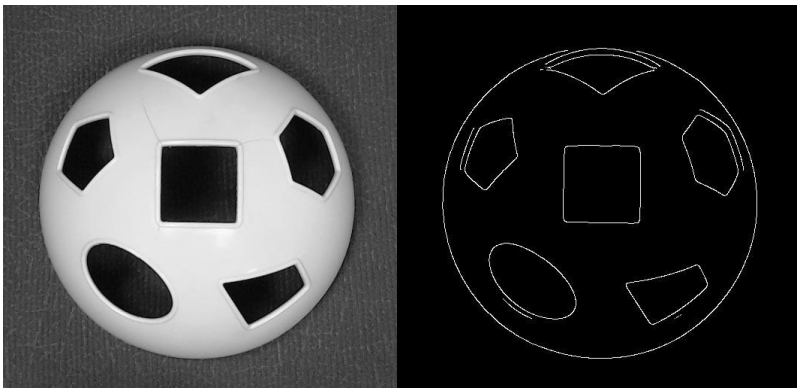


Figure (3.6):The original image on the left and the detected edges on the right using Canny edge detection algorithm

3.1.6 Hough transform:

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing.[29]

The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a "generalized Hough transform"[32]

a. Hough transform for circle detections

The circle Hough Transform (CHT) is a feature extraction technique for detecting circles. It is a specialization of Hough Transform. The purpose of the technique is to find circles in imperfect image inputs. The circle candidates are produced by "voting" in the Hough parameter space and then select the local maxima in the accumulator matrix.

In a two-dimensional space, a circle can be described by:

$$(\mathbf{x}-\mathbf{a})^2+(\mathbf{y}-\mathbf{b})^2=\mathbf{r}^2 \qquad (3.3)$$

Where (a,b) is the center of the circle, and r is the radius. If a 2D point (x,y) is fixed, then the parameters can be found according to (3.3). The parameter space would be three dimensional, (a, b, r). And all the parameters that satisfy (x, y) would lie on the surface of an inverted right-angled cone whose apex is at (x, y, 0). In the 3D space, the circle parameters can be identified by the intersection of many conic surfaces that are defined by points on the 2D circle. This process can be divided into two stages. The first stage is fixing radius then find the optimal center of circles in a 2D parameter space. The second stage is to find the optimal radius in a one dimensional parameter space.

So For each point (x, y) on the original circle, it can define a circle centered at (x, y) with radius r ,The intersection point of all such circles in the parameter space would be corresponding to the center point of the original circle as we see in fig 3.7

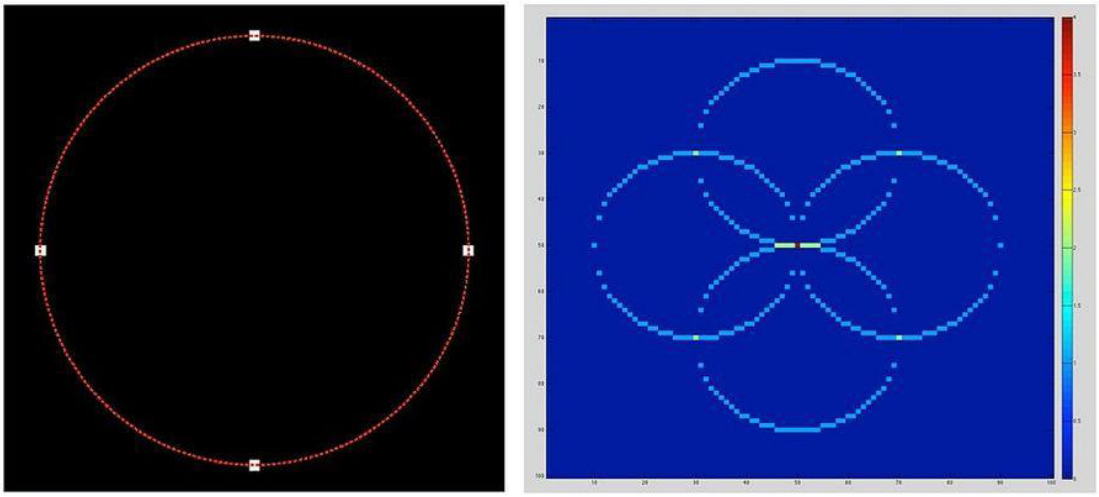


Figure (3.7): an edge image of a circle in the left and its Hough transform space for 4 point in the right

Consider 4 points on a circle in the original image (left). The circle Hough transforms is shown in the right. Note that the radius is assumed to be known. For each (x,y) of the four points (white points) in the original image, it can define a circle in the Hough parameter space centered at (x, y) with radius r. An accumulator matrix is used for tracking the intersection point.

In the parameter space, the voting number of points through which the circle passing would be increased by one. Then the local maxima point (the red point in the center in the right figure) can be found. The position (a, b) of the maxima would be the center of the original circle. [33]

3.2 Vision system

3.2.1 Camera System

The lens inside the camera refracts all rays of light from a certain object point to one single point in the image plane. If the lens is thin implying the distortion can be neglected, the lens law is valid.

$$\frac{1}{\alpha}+\frac{1}{\beta}=\frac{1}{f} \qquad \textbf{(3.4)}$$

Where α is the distance between the lens and the object, β is the distance between the lens and the image plane and f is the focal length. Figure 2.1 illustrates the lens law. [33]

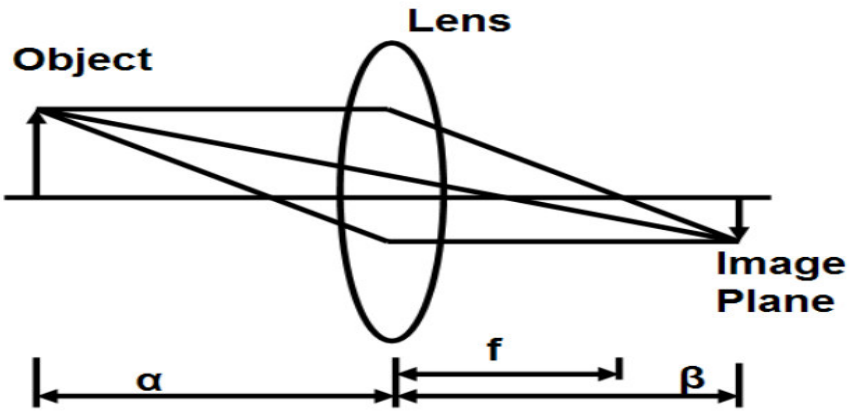


Figure (3.8): Illustration of the lens and object distances and focal lengths

By the lens law it is obvious that an object at the distance α from the lens will be reproduced with complete sharpness on the image plane. If the distance between the object and the lens differs from α, the reproduction on the image plane will be more or less blurred.

3.2.2 Camera Modeling and Calibration:

The Camera Calibrator It allows us to estimate camera intrinsic, extrinsic, and lens distortion parameters. You can use these camera parameters for various computer vision applications. These applications include removing the effects of lens distortion from an image, measuring planar objects, or reconstructing 3-D scenes from multiple cameras.

In robotics application, camera calibration used to estimate the camera parameters to be able to get a position or a distance of an object, and in stereo vision we can do the mapping and pose estimation by two calibrated camera.

3.2.2.1 Camera Model:

The Computer Vision System Toolbox™ calibration algorithm uses the camera model proposed by Jean-Yves Bouguet [34] . The pinhole camera model does not account for lens distortion because an ideal pinhole camera does not have a lens. To

accurately represent a real camera, the full camera model used by the algorithm includes the radial and tangential lens distortion.

a. Pinhole Camera Model

A pinhole camera is a simple camera without a lens and with a single small aperture. Light rays pass through the aperture and project an inverted image on the opposite side of the camera. Think of the virtual image plane as being in front of the camera and containing the upright image of the scene.

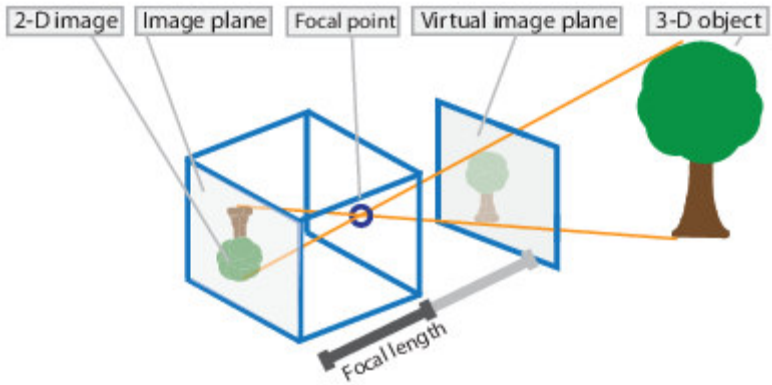


Figure (3.9):Illustration of the camera model from the object to the 2d image

The pinhole camera parameters are represented in a 4-by-3 matrix called the *camera matrix*. This matrix maps the 3-D world scene into the image plane. The calibration algorithm calculates the camera matrix using the extrinsic and intrinsic parameters. The extrinsic parameters represent the location of the camera in the 3-D scene. The intrinsic parameters represent the optical center and focal length of the camera.

$$w\,[x\ y\ 1] = [X\ Y\ Z\ 1]\,P \qquad \textbf{(3.5)}$$

Scale factor — Image points — World points

$$P = \begin{bmatrix} R \\ t \end{bmatrix} K \qquad \textbf{(3.6)}$$

Camera matrix — Extrinsics: Rotation and translation — Intrinsic matrix

The world points are transformed to camera coordinates using the extrinsic parameters. The camera coordinates are mapped into the image plane using the intrinsic parameters. [35]

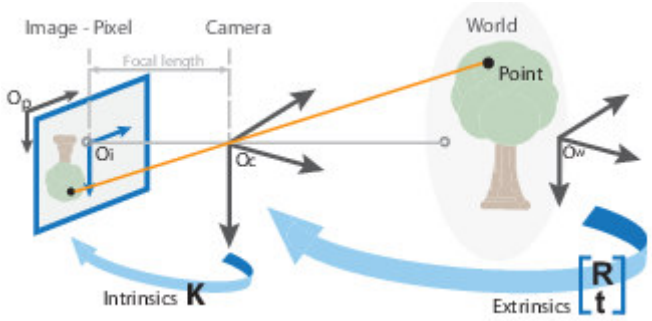


Figure (3.10): Illustration of the camera model showing the transformation by the Extrinsic parameters and the mapping into the image plane using the intrinsic parameters

b. Distortion in Camera

The camera matrix does not account for lens distortion because an ideal pinhole camera does not have a lens. To accurately represent a real camera, the camera model includes the radial and tangential lens distortion. [36]

b.1 Radial distortion

Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center. The smaller the lens, the greater the distortion and the radial distortion coefficients model this type of distortion.

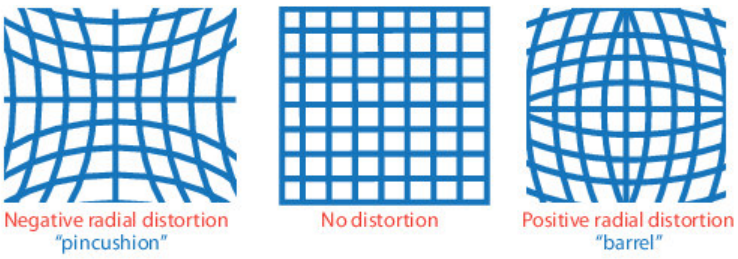


Figure (3.11): radial distortion of 3 type of lens

b.2 Tangential Distortion

Tangential distortion occurs when the lens and the image plane are not parallel. The tangential distortion coefficients model this type of distortion

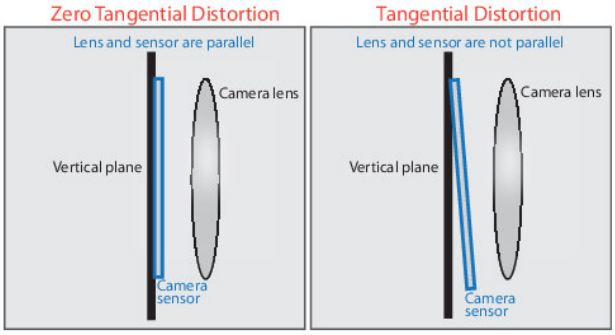
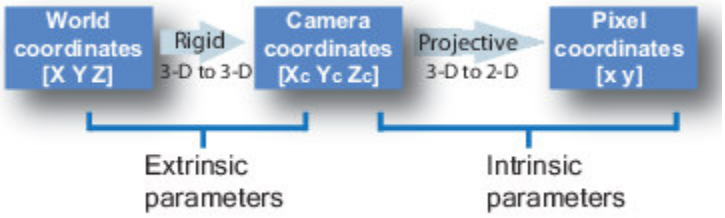


Figure (3.12): illustration of Tangential Distortion of a lens

3.2.2.2 Camera Calibration Parameters

The calibration algorithm calculates the camera matrix using the extrinsic and intrinsic parameters. The extrinsic parameters represent a rigid transformation from 3-D world coordinate system to the 3-D camera's coordinate system. The intrinsic parameters represent a projective transformation from the 3-D camera's coordinates into the 2-D image coordinates.



Figure(3.13): Diagram describe the transformation from the World coordinates into Pixel coordinates passing by Extrinsic and intrinsic camera parameters matrices

a. Extrinsic Parameters

The extrinsic parameters consist of a rotation, *R*, and a translation, *t*. The origin of the camera's coordinate system is at its optical center and its *x*- and *y*-axis define the image plane. Define the camera Extrinsic matrix E by

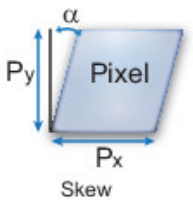
$$E = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \quad \textbf{(3.7)}$$

b Intrinsic Parameters

The intrinsic parameters include the focal length, the optical center, also known as the *principal point*, and the skew coefficient. The camera intrinsic matrix, *K*, is defined as:

$$K = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \quad \textbf{(3.8)}$$

The next figure describe the pixel skew



Skew

Chapter 4: Implementation and practical results

4.1 Manipulator Setup

4.1.1 Hardware:

In our work, we use an Arduino mega microcontroller and a motor driver to control our manipulator, but the computations are done in the computer instead of the microcontroller (Arduino), who just does the work of the Analog/Digital conversion for the analogical sensors and the PWM generation which represents the voltage signals for the DC Motors.

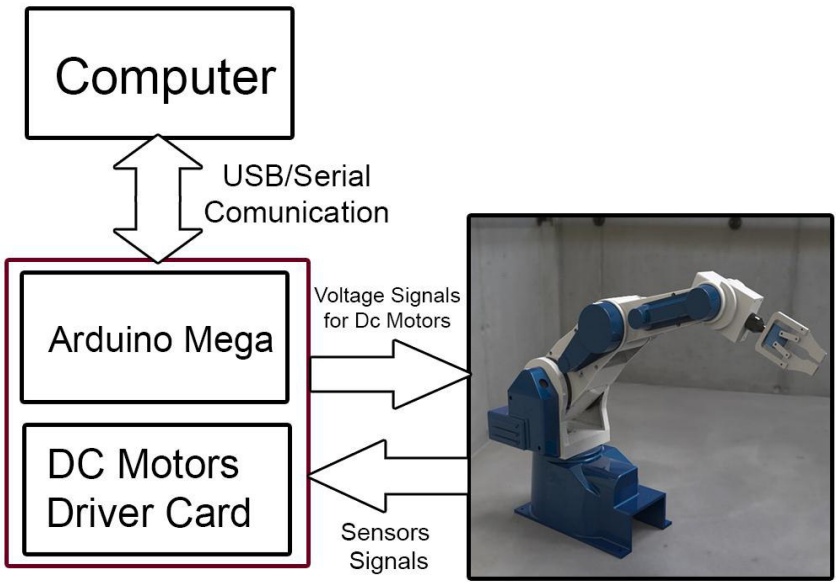


Figure (4.1a): Hardware Setup

Figure (4.1b) The previously discussed manipulator workbench

4.1.2 Software:

The control implementation is designed in MATLAB Simulink as diagram blocks, where we design the control law, generation trajectories and filtering the noisy data measured from each manipulator's sensor via the Arduino analog inputs, and finally the resulting control law will be transferred to the microcontroller (Arduino) which generates a PWM signals. The DC motor Driver Card contains three L6203 H-Bridges, which eventually amplifies the signals from 5V (Arduino command signals) to the driving signals 12V .

As we mentioned in the introduction, we used the Arduino IO (MATLAB AND SIMULINK SUPPORT PACKAGE FOR ARDUINO). We upload the Arduino IO code into the Arduino board, this code does the Serial communication by reading and writing from and to Simulink blocks with a specific communication protocol.

The reason of using the MATLAB AND SIMULINK SUPPORT PACKAGE FOR ARDUINO, is to minimize the time of computation, because in our case all the computation run in computer speed instead of the Arduino microcontroller speed which is very slow compared to the computer, so the role of the Arduino is just to

receive the command signals and send the measurements of the potionmeter. Other reason, is the filters and the complicated iterative functions that are already implemented in Simulink and we can use them, and also all the measured and the filtered data are saved automatically in MATLAB workspace, and there's also an important reason, is the supervision which we can visualize real-time values of the closed-loop, and we can tune the PID in real-time which makes it a very helpful tool, and makes our work very clear, efficient and time saving.

The figure 4.2, represents a Simulink model example of driving a dc motor using **analogWrite** function to Generate PWM signal to drive the DC motor, also we used a **digitalWrite** function to enable and disable the motor.

For the sensor measuring we used **analogRead** function to read the voltage from the potienmeters, and then we can get the angle value using an experimental calibration which relates the measured voltage with angle value in radian.

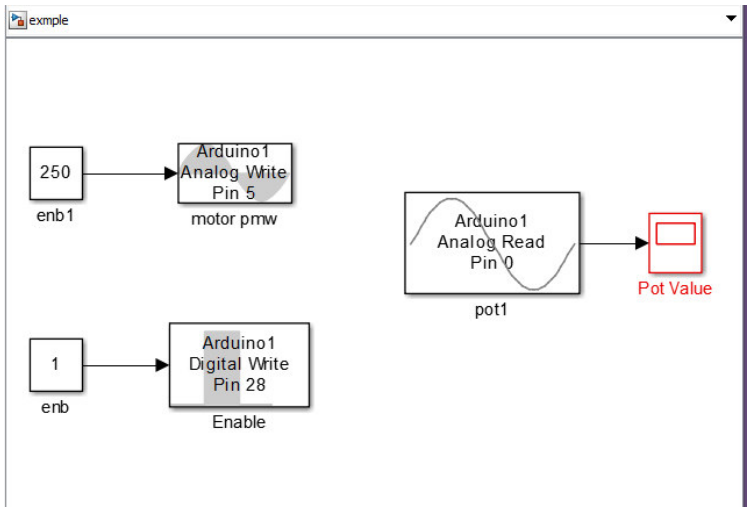


Figure (4.2): Simulink model of driving a Dc Motor and measuring position from its sensor

4.1.3 Filtering:

When we measured the joints angles with the potienmeters, the data we got was very noisy, so we had to use a Lowpass Filter.

Denote that the sampling time Te=0.01S, so we have sampling frequency Fe=100Hz.

Appling Shannon theorem we have:

$$\mathrm{Fb} \geq 2\mathrm{Fe}$$

With Fb is the lowpass Filter *cut*-off *frequency. In our case we choose* $\mathrm{Fb} = 300\mathrm{Hz}$

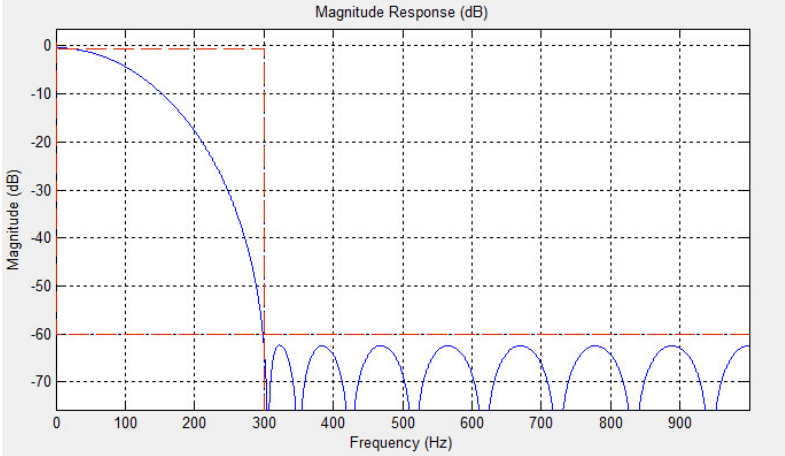


Figure (4.3): Lowpass Filter Response with 300Hz *cut*-off *frequency*

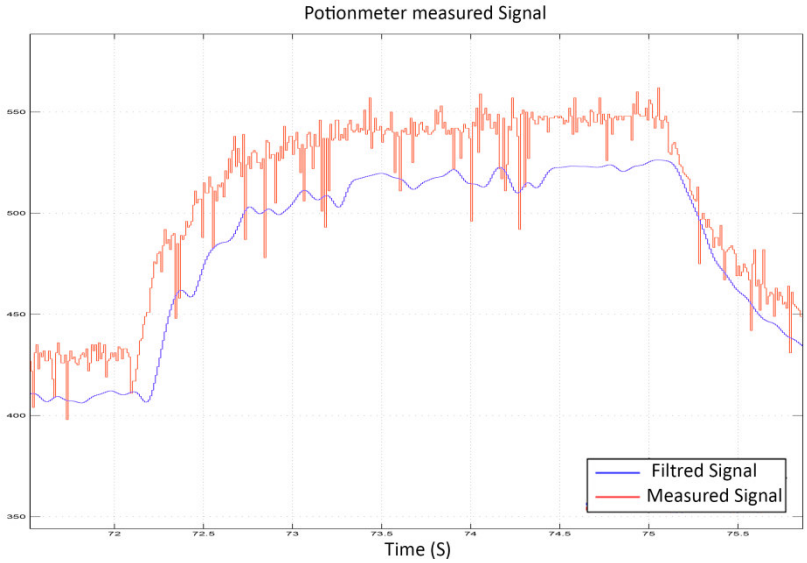


Figure (4.4): joint measured signal and the filtered one

4.2 independent Joint Control (PID Position Control)

4.2.1 Design and concept

As we mentioned in section ***2.1.1*** we apply a PID controller for each joint independently, and we add each of forward and inverse kinematics functions, to translate from Cartesian space to the joints space and vice versa. To achieve the desired positions we generate a trajectory in Cartesian space, this trajectory composed of three trajectories Xtraj, Ytraj and Ztraj, then they will transform by the inverse kinematics function to joints space trajectories.

So the PID controllers try to achieve these desired trajectories. And to compare the real position of the manipulator with the desired one, we used the forward kinematics function.

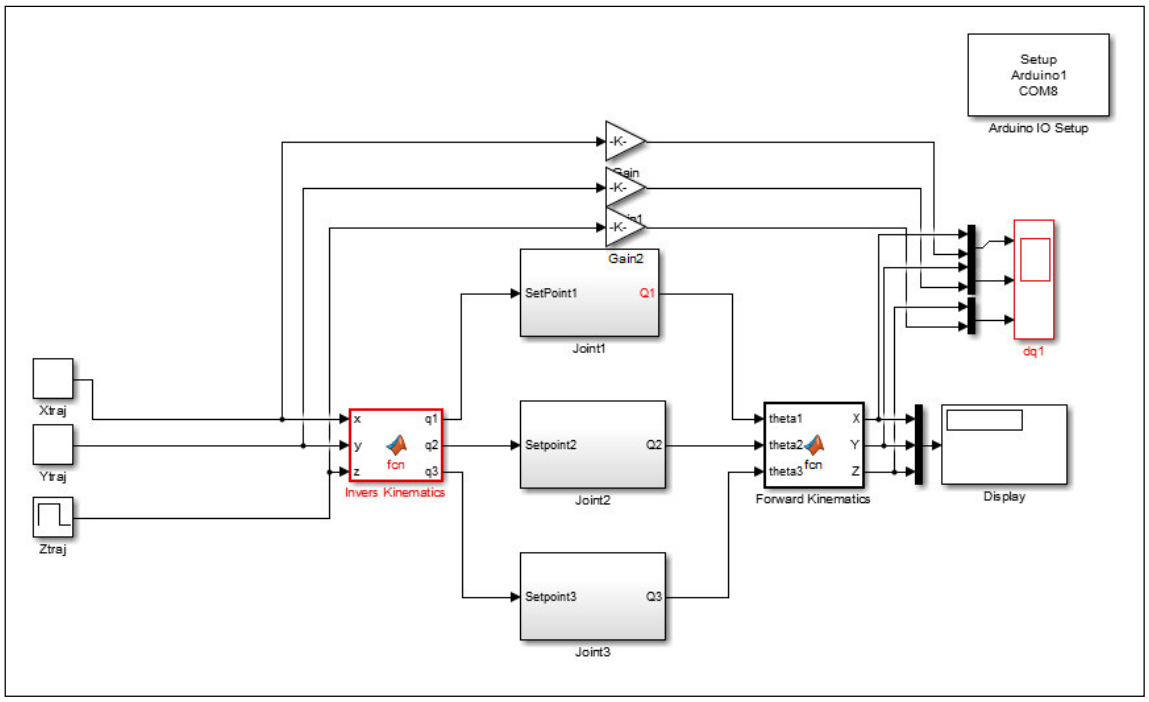


Figure (4.5): Simulink model of independent joints positions control following a Cartesian space trajectory

The figure below represent a single joint subsystem, which has an input that is the Set Point (desired angle in Radians), the closed-loop with discrete time PID

controller, the PID output is limited (from -255 to 255), so we use a map function to map the output value into the work range of the analogeWrite function (from 0 to 255), then the measured data will be filtered and map from Voltage value into radian, all the values in the closed-loop are displayed and also plotted on a scopes, finally the output of the subsystem represent the first joint angle named q1.

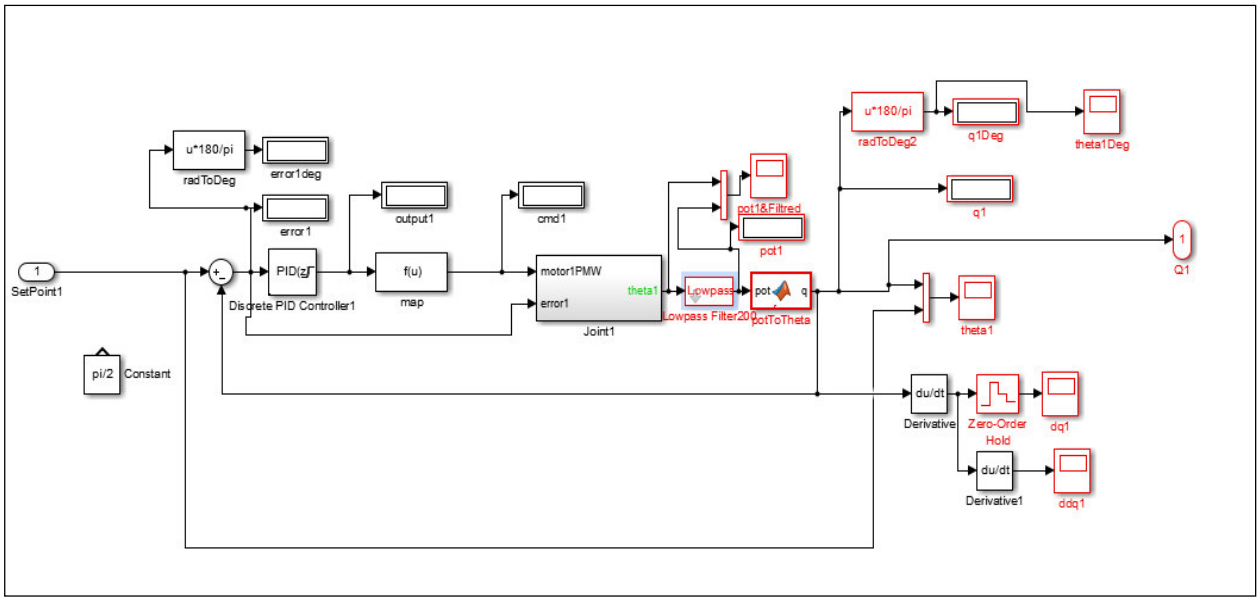
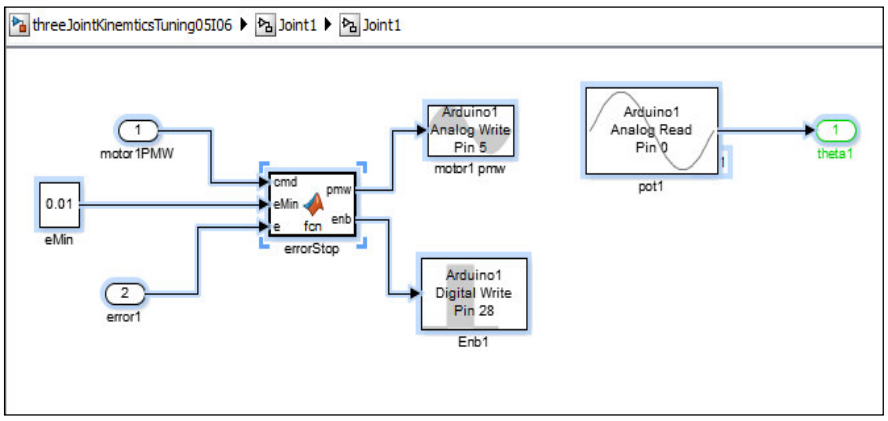


Figure (4.6): *Simulink model of PID controller for a single joint with supervision of all the control loop signal values*

We used a Stopping criteria that disables the motor if the error is reaching zero (<0, 01 Radians).



Figure(4.7): Simulink Subsystem model of a Dc motor command with a stopping criteria and joint sensor measurement

4.2.2 Tuning and results

By several experiments we get the PID gains Kp,Kd and Ki .

Table 4.1 : the PID gains of the first joint controller

Set point (rad)	Kp	Kd	Ki
π/2	289	40	50

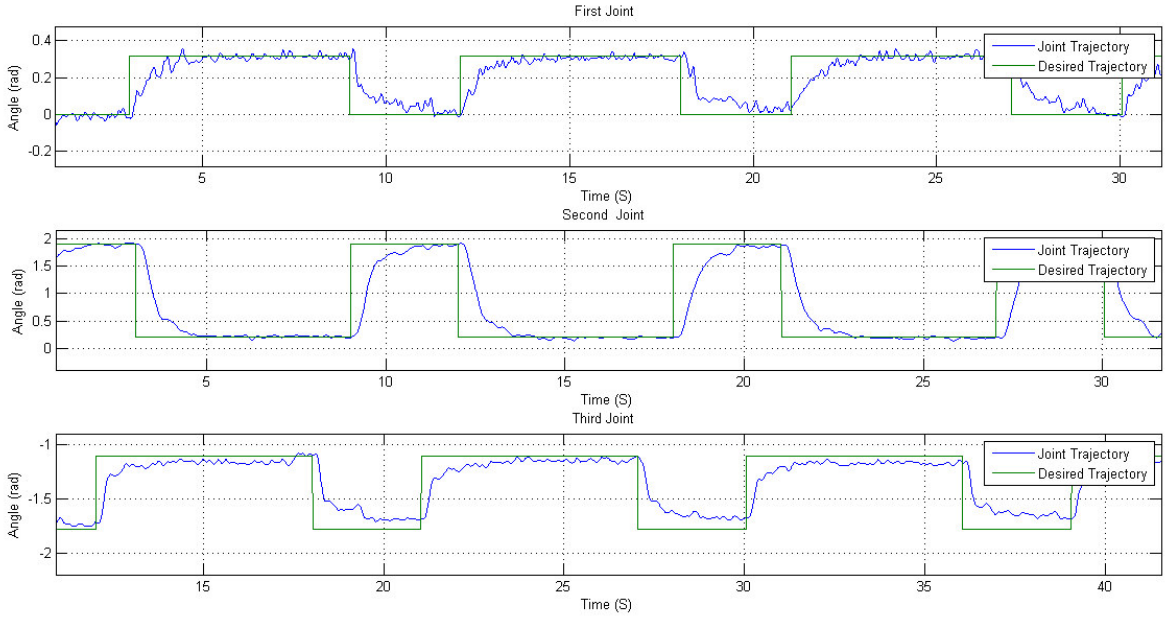
Table 4.2: the PID gains of the second joint controller

Set point (rad)	Kp	Kd	Ki
π/2	225	30	10

Table 4.3 : the PID gains of the third joint controller

Set point (rad)	Kp	Kd	Ki
-π/3	270	30	50

After 30 seconds of data recording we get the following figures with sample time of 0.01S.



Figure(4.8): Angles responses for 3 joints using PID controllers

As we see, the responses are acceptable, for the first joint there is no steady stat error but there are oscillations caused by the high gain Kp, we try to reduce it but that lead us to a big steady stat error specially in small Set Points, and this due to the friction in the joint.

For the second joint and third joint we have good responses only a small steady stat error, and the oscillations due to noise in the joints sensors.

Same experiment but the figures below shows the responses of the end effector position in the Cartesian space.

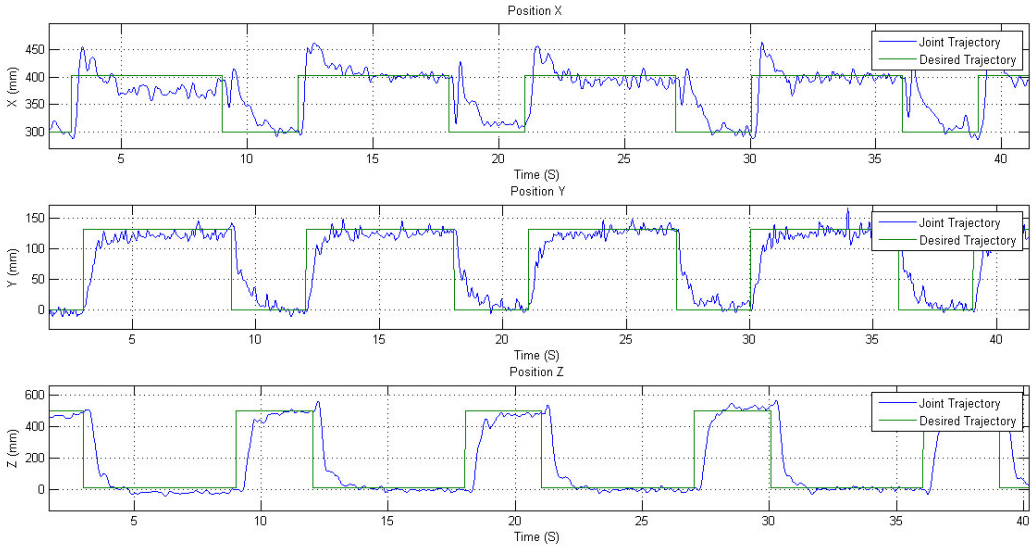


Figure (4.9): end effector position signals (X,Y,Z) following a desired trajectory

4.3 Parameter Estimation

as we mentioned in section **2.1.2**, for a robot manipulator to track a specific trajectory we, have to apply a computed torque control, to do that we need to know the inverse dynamics model, which means, knowing or estimating the manipulator parameters, in our case the robot is too ancient and not so commonly used, and the datasheet is not available, therefore, the parameters values are unknowns, so the solution is to estimate these parameters with the simplest system identification tool, known by least-squares method.

Denote that the Least-Squares model of the manipulator is an inverse dynamical model, which means the inputs are the joints angles and their derivatives, and the outputs are the PWM signals, and that's what we need in computed torque control method.

In section **2.2.3.a** we've seen the demonstration of LS method, and in **2.2.3** we've written the manipulator's inverse dynamics as follow:

$$Y = \Phi\hat{\theta} \quad (4.1)$$

Where Y *is the output vector,* Φ the regresseur matrix and $\widehat{\Theta}$ the estimated parameters vector.

In our work, to apply LS method we implemented the regresseur matrix $\boldsymbol{\Phi}$ Eq (2.51) in MATLAB, then we generate a spesific signals comunoly used in system identification, and we use them to excite the robot dynamics, but this signals are exciting the robot in closed-loop using PID contollers, because the robot is unstable in open-loop as we mentioned in 2.2.1.

4.3.1 The Exciting Trajectory

We used very efficient signals to excite the robot joints, these signals are of type Fourier series, which can be parameterized as a sum of finite Fourier series as follow

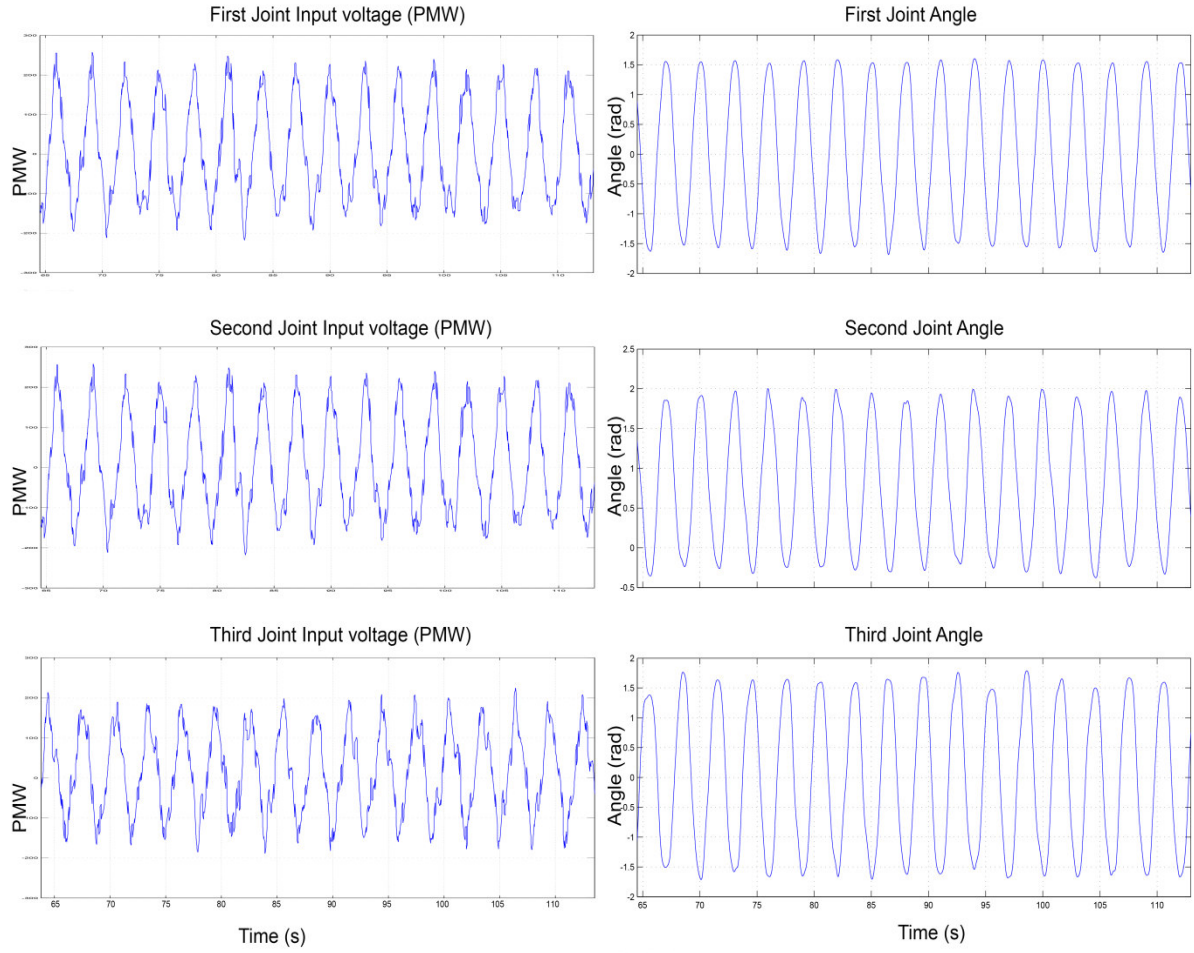
$$\mathrm{q_i}(t) = q_{i0} + \sum_{k=0}^{n} \mathrm{a}\sin(\mathrm{w_f}\,kt) - bcos(\mathrm{w_f}kt) \quad (4.2)$$

Where $\mathrm{w_f}$ is fundamental frequency of the excitation trajectory.

We recorded about 110 seconds of data, in the left side of the figure 4.8 below represent the three excitation signals (PWM signals) which are going to transfer to voltage by the motors driver board, and this represents the output vector $\boldsymbol{Y}$, in the right side of the figure 4.8 we have the three joints angles values represented in (rad).

With sampling time Te=0.01 s, for each signal we have 11000 sample so that

$Y = \begin{bmatrix} \mathrm{U_1} \\ \mathrm{U_2} \\ \mathrm{U_3} \end{bmatrix}$ (4.3) with $\mathrm{U_1}, \mathrm{U_2}$ and $\mathrm{U_3}$ are 11000X1 vector, and Y is 33000X1 vector.



Figure(4.10): three joint angles signals and their inputs PWM signals

4.3.2 Applying the LS method

After we get the joint angles, we compute the regresseur matrix **Φ** using the joints angles values and its derivatives, and we arrange them just as we did in Eq (2.51)**,** then we apply the LS method to estimate the 20 parameters Eq (2.50) As follow

$$\hat{\theta} = \left(\Phi^{\mathrm{T}}\Phi\right)^{-1}\Phi^{\mathrm{T}}\mathrm{Y} \quad (4.4)$$

With $\Phi$is 33000X20 matrix and$\widehat{\Theta}$is 20X1 vector

4.3.3 Parameters validation

To validate our parametric model using the parameters that we estimated, we have to excite the system with other trajectories, then use the given outputs which are the joints angles signals and its derivatives to make the estimated model represented by the regresseur matrix Φ and the estimated parameters $\widehat{\Theta}$, and we compute the outputs vector Υ which represents the Estimated PWM signals , we computed as follow :

$$Y = \Phi\hat{\theta} \quad (4.5)$$

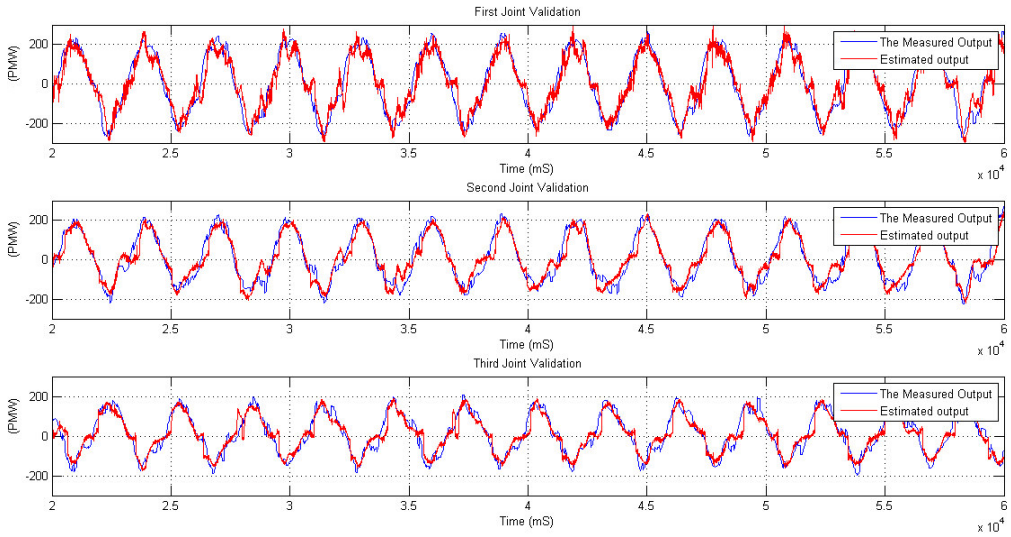


Figure (4.11): Validation Signals for the three joints manipulator represented by the measured and the estimated output signals

We compute the RMS error between the real and the estimated output signals for the three joints and we got:

Table 4.4: RMS error and the fit between the measured and the estimated outputs for the three joint

	Joint 1	Joint 2	Joint 3
RMS error	47	37	32
fit	63%	70%	74%

So the model we estimate fits the real one about 70%, which are acceptable results.

The errors we got are due to the nonlinearity in the parameters that we cannot estimate with LS method as stribeck effect and some mechanical problems like hysteresis and the dead zones in the Dc motors.

And also the big problem is data filtering and recording, as we seen in section 4.1.3 there is a lot of noise in the sensors, after filtering we've seen the lag between the real and the filtered signals ,and also the first and the second derivatives are even further from the real ones.

Also there's problem of unites in the parameters we got , in our work we take the output data as voltage but in PWM , and we consider the relation between the PWM value and the real voltage in the motors are linear, which is not guaranteed ,and if it was linear we don't know this scaling factor which could make our parameters in wrong unites.

4.4 Computed toque control (Trajectory tracking)

As we mentioned in section 2.1.2,to track a specific trajectory with manipulator we need to apply a nonlinear control law using feedback linearization and known in robotics as Computed torque control.

4.4.1 Design and concept

As we've seen in Eq 2.19, the control law is

$$\tau = M(q)\left(\ddot{q}d - Kd\dot{e} - Kpe\right) + C(\dot{q}, q)\dot{q} + F(\dot{q}) + G(q)$$

With $e = x(t) - x_d(t)$

By adding the motors dynamics Eq (2.47) the control law become

$$\mathrm{U} = (\acute{\mathrm{J}} + M(q))\left(\ddot{q}d - Kd\dot{e} - Kpe\right) + C\ (q\ , \dot{q})q + F(\dot{\dot{q}}) + G(q) + \mathrm{Tf} \quad (4.6)$$

And we have from section 2.2.3**,** the inverse dynamic model we got from LS estimation method

$$Y = \Phi\hat{\theta}$$

Where $Y = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}$ so that$U_1, U_2 \text{and} U_3$ are the motors inputs PWM signals, and we have *the* regresseur matrix Φ representing all the dynamics equations known as regressors function, and $\widehat{\Theta}$ contain the estimated parameters

So we can write

$$Y = \mathrm{U} = \int \ddot{q} + M(q)\ddot{q} + C(q\,,\dot{q})q + G(\dot{q}) + F(\dot{q}) + \mathrm{Tf} = \Phi(q, \dot{q}, \ddot{q})\hat{\theta} \quad (4.5)$$

And if we use the new control law (4.4) , and by replacing $\ddot{q}$ by $\left(\ddot{q}d - Kd\dot{e} - Kpe\right)$ which represents the PD Controller plus the feedforward, we get this control law

$$\mathrm{U} = \Phi\left(q, \dot{q}, \left(\ddot{q}d - Kd\dot{e} - Kpe\right)\right)\hat{\theta} \qquad (4.6)$$

And there is the implementation of the computed torque control law using the inverse dynamics model that we get in the identification part

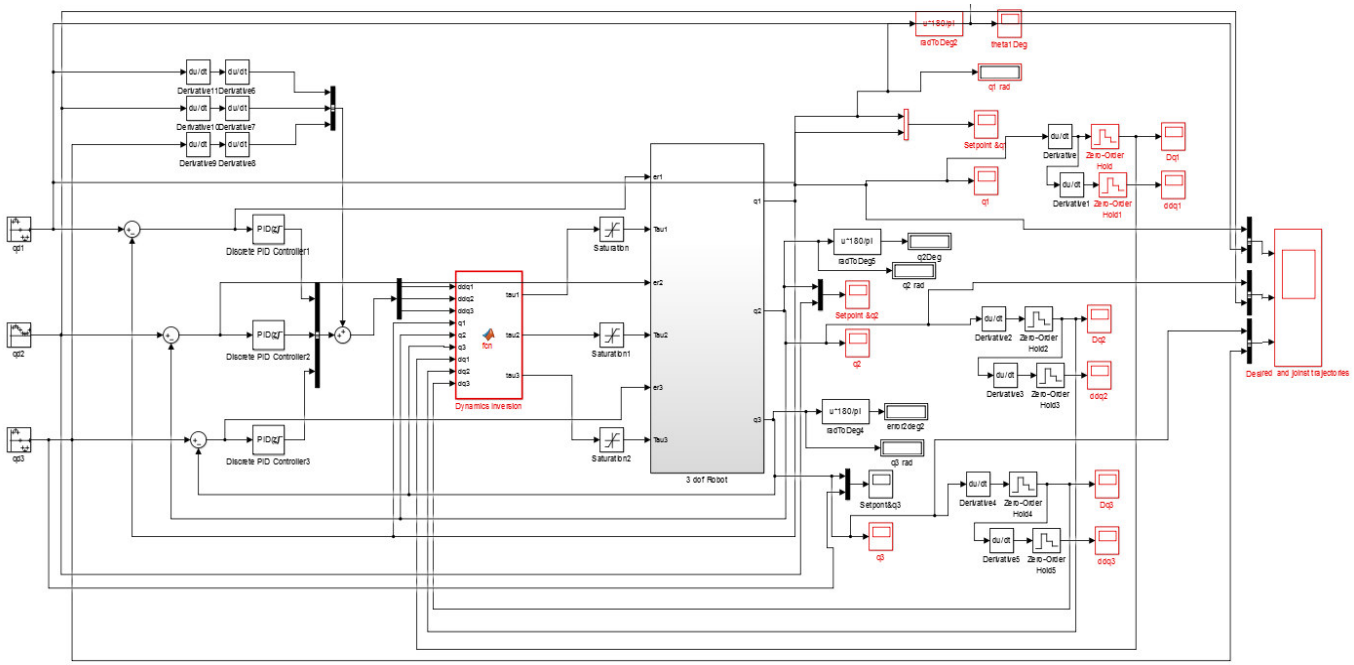


Figure (4.12): *Simulink model of Computed torque control for three joints manipulator using Dynamics inversion and 3 PD controller and supervision of all the loop signals values*

4.4.2 tuning and results

By several experiments we get the PD gains Kp and Kd .

Table 4.5 : the PID gains of the first joint controller

Set point (rad)	Kp	Kd	Ki
π/2	25	10	0

Table 4.6 : the PID gains of the second joint controller

Set point (rad)	Kp	Kd	Ki
π/2	9	6	0

Table 4.7: the PID gains of the third joint controller

Set point (rad)	Kp	Kd	Ki
-π/3	49	14	0

After 15 seconds of data recording we get the following figures with sample time of 0.01S

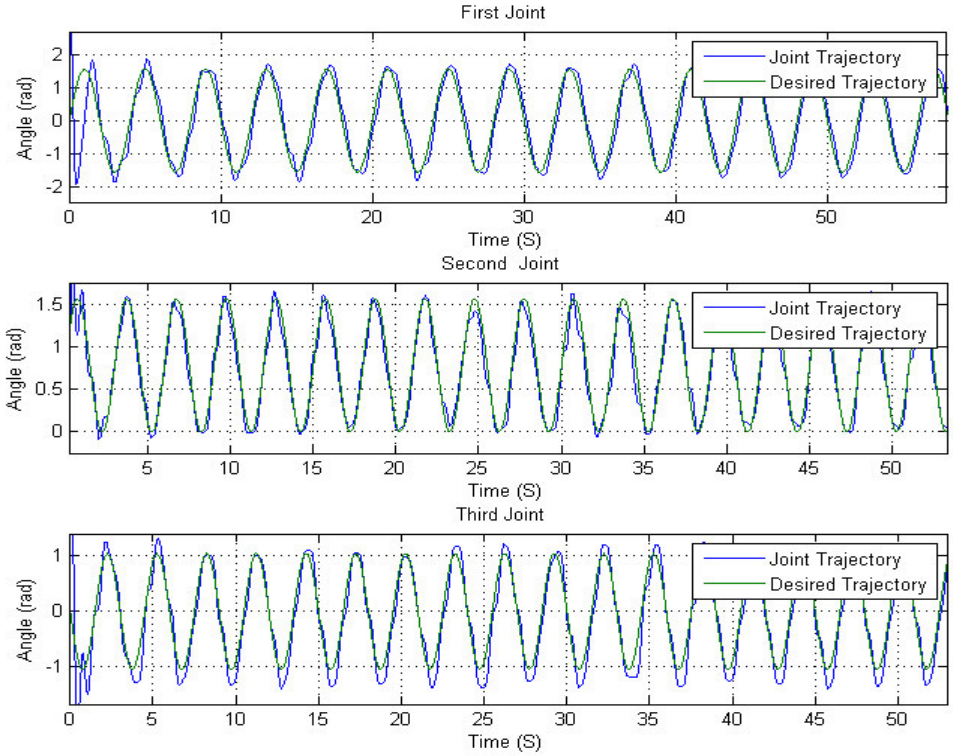


Figure (4.13): joints Angles Responses after using a computed torque control for tracking desired trajectories

As we have seen, the manipulator is tracking the desired trajectories, but there is some error due to the errors in the estimated model and the dynamics that are not included when modeling, and there is some lag because of the filter lag as we mentioned previously.

And also if we have a fast desired trajectory the manipulator cannot track, because the motors velocities are too limited.

4.5 Vision Control application:

If we want to grasp an object like pick and place application, we have to know the exact position of the object. But if the objects are coming randomly from a conveyor for example, the only solution is using a vision system to a Estimate the

objects poses. Then we use the inverse kinematics to get the desired angles, and finally apply a control law to achieve them.

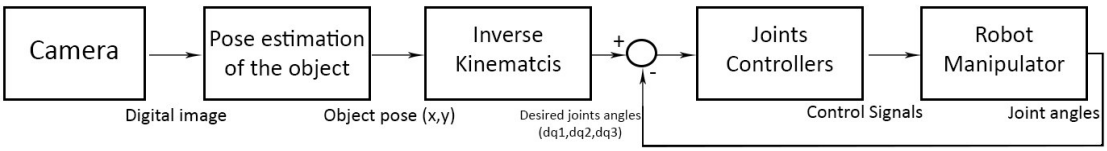


Figure (4.14): Vision Control for diagram a Robot Manipulator

In our work we try to estimate a pose of a red ball, we used image processing tools that are implemented in MATLAB to detect the red ball. But this detection allows us to know the ball position in the image which means in pixels, this result doesn't help in our application, because in our robot we can achieve a position using the real world coordinates.

In order to get the camera (webcam) model, we have to calibrate it, and then using the intrinsic and extrinsic camera parameters, we can transform the object position from image which is by pixels, to the real position by meters with respect to the world coordinate frame, then into the robot base coordinate frame.

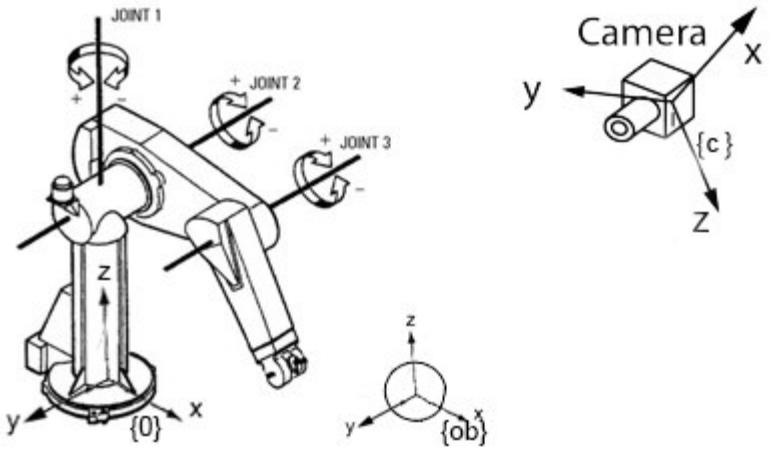


Figure (4.16): illustration of robot manipulator, camera and object coordinate frames

The previous figure shows us the different coordinate frames positions, where {0} is the robot bas coordinate frame which is the frame that we used in inverse kinematics, and we have the camera coordinate frame {c}, and finally the object coordinate frame {ob}, so using a detection algorithm we get the position of the object with respect to {c} and then we transform it to the robot frame {0}. Denote that the position of the camera with respect to the robot coordinate frame is fixed and known.

4.5.1 Object detection

In this part we worked on object detection and position estimation in the image, first thing we get the real-time image from the webcam, it's a RGB image, in order to detect the ball from its color we have to convert the image into HSV space, by selecting a Hue range and a Saturation range, these two ranges selected such a way that is cover the color of the ball in different intensity and light distribution on the ball.

So we have a red ball, by many experiences we select as follow:

$$\mathbf{360} > Hue > \mathbf{340}$$

$$\mathbf{0.80} > Saturation > 0.2$$

Then we threshold the hue and the saturation images planes using these two ranges.

After we got the binary image we applied the Hough Transform for circle detection using MATLAB function **imfindcircles**, by choosing a range for radius values [35, 65] and after some experiences. We got the following result

Figure (4.16): circle detection after thresholding and applying circle Hough transform

And we get the radius value r=18 pixel, and the position with pixels (290,181)

4.5.2 Camera calibration and pose estimation

Now after we got the ball position in the image (by pixels), we want to transform this position into the real world coordinate frame then to the manipulator coordinate frame, and use this position to get the required joints angles using the inverse kinematics.

In order to estimate any mathematical model we have to collect data represented by inputs and outputs as we did in identification, in camera calibration which is an estimation of the camera parameters we have to take several positions in the real world which represent the inputs, and estimate these positions in the digital image which represent the outputs.

Commonly in camera calibration they used a checkerboard that we know its squares sizes, so we can get the real world position of each square, and camera calibration algorithm could estimate the positions of each square in the image. Using this data, the MATLAB calibration toolbox uses some mathematical tools such SVD to estimate the camera model.

In our work we took several photos of the workplace where the checkerboard placed in a specific position with respect to the manipulator.

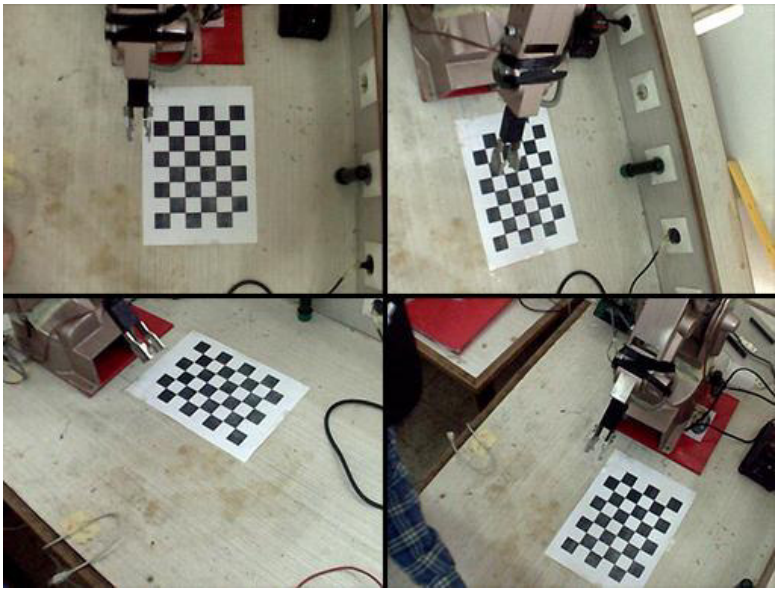


Figure (4.17): examples of several checkerboards photos that are used in camera calibration

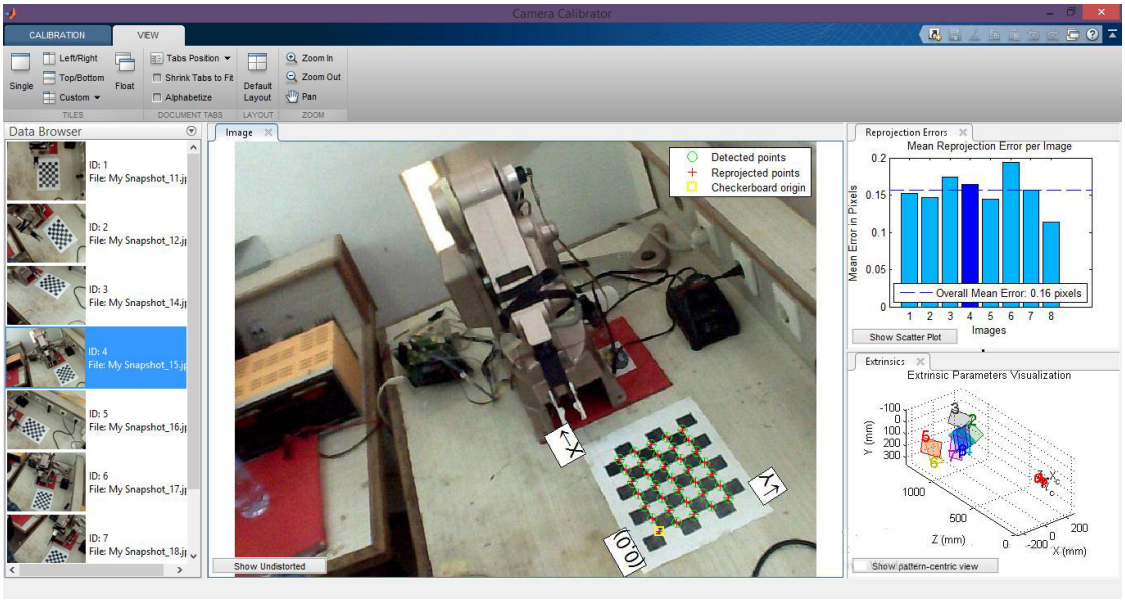


Figure (4.18) MATLAB camera calibrator app showing the world coordinate frame after calibrating the camera and showing the mean error of the estimation

As we have seen the mean error of estimation is about 0.15 pixel which is a good result.

4.5.3 Vision position control

After getting the camera parameters represented by extrinsic and intrinsic, we are able to estimate any position in the checkerboard plane with respect to the world frame.

To do that we implemented this equation using (3.5),(3.6),(3.7) and (3.8)

$$P = \begin{bmatrix} R \\ t \end{bmatrix} K$$

With $\mathbf{P}$ represent the camera parameters

$$[X \quad Y \quad Z] = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \left(\begin{bmatrix} R \\ t \end{bmatrix} K \right)^{-1}$$

With (x,y) is the position in the image and (X,Y,Z) is the position of the ball with respect to the world frame.

To compare the estimated position with the real one, we placed the ball in known positions and then apply the position estimation algorithm. We got the following results

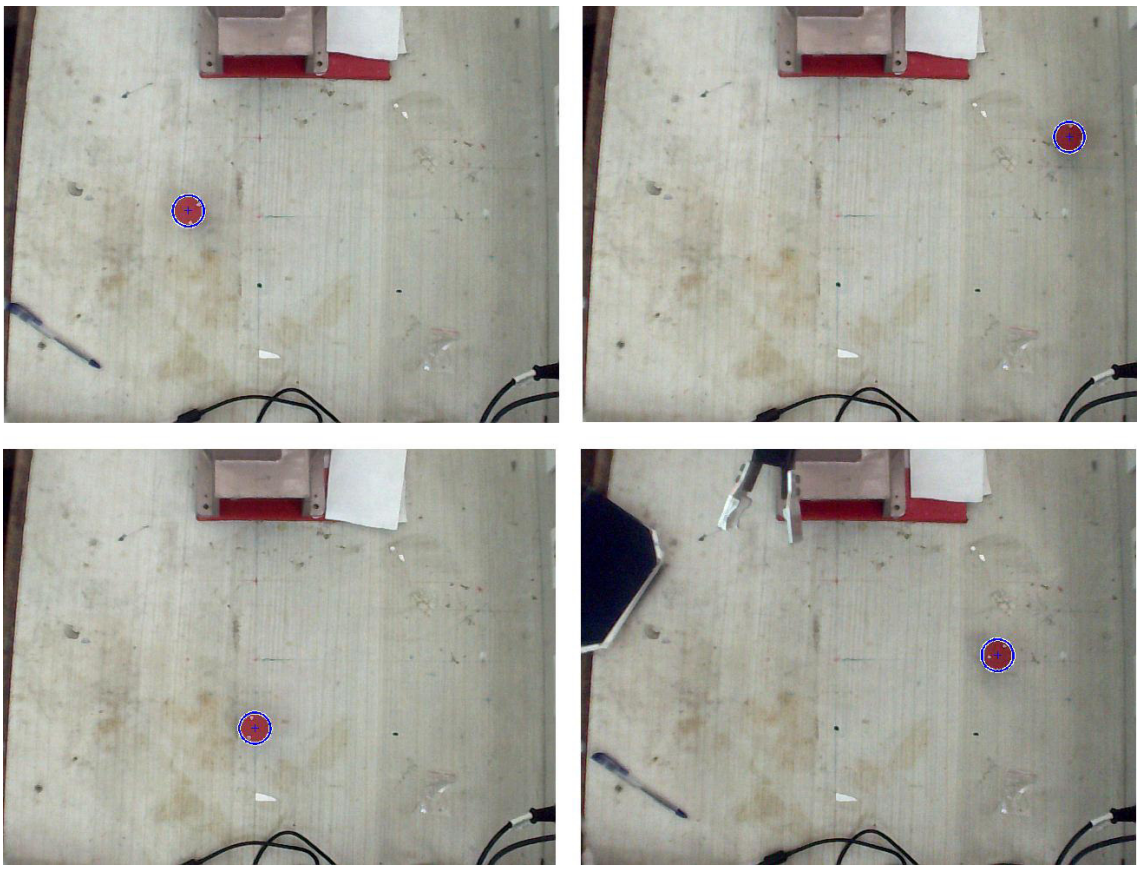


Figure (4.19): examples of real-time position estimation using MATLAB and calibrated camera

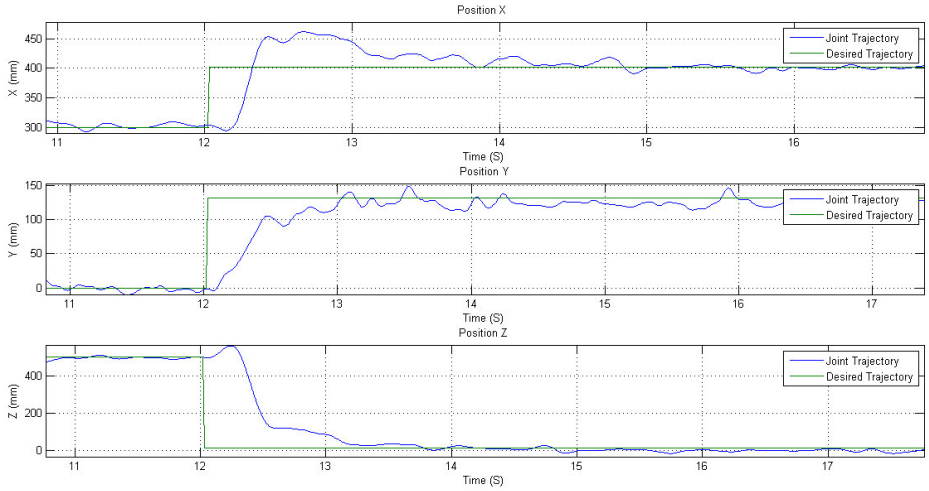
Table 4.8 : real positions of the ball compared with the estimated positions

Real position(mm)	*Estimated position(mm)*
(400,-100)	*(406,-101)*
(300,300)	*(300,302)*
(500,0)	*(492,-6)*
(400,200)	*(402,199)*

As we have seen from the table we have a good estimation of the ball position in the real world frame.

In order to grasp the object by the manipulator hand all we have to do is to send the position vector to the kinematics control SIMULINK model we demonstrated in Figure (4.5).

The figure below shows the end effector position after feeding the Set point from the vision pose estimation output.



Figure(4.20): end effector position following a desired position feeding from camera vision

Conclusion

In this thesis we implemented an efficient method for control of a manipulator with unknown dynamical parameters .The PID used for position control as independent controller for each joint. Although, these PID controllers do not guarantee a tracking of a desired trajectory. To solve this problem we applied a nonlinear controller which is known in robotics by 'Computed torque control', and in order to do this we have estimated the parameters of the manipulator. And this controller tracks the desired trajectory.

After that we have controlled the robot, we took a step further by making it more intelligent by adding vision, and now we are able to grasp an object in any arbitrary position , by detecting the object's position in the digital image ,then transform it to the world coordinate frame using a calibrated camera .

Bibliography

- [1] CONTROL STRATEGIES FOR ROBOTS IN CONTACT ,Jaeheung Park, March 2006 .
- [2] M.W. Spong, S. Hutchinson, and M. Vidyasagar. Robot modeling and control. Wiley New Jersey, 2006.
- [3] L. Sciavicco and B. Siciliano. Modeling and control of robot manipulators. Springer Verlag, 2000.
- [4]Mathworks. MATLAB - The Language of Technical Computing, Last accessed June 13, 2011. http://www.mathworks.com/products/matlab/.
- [5] Mathworks. Simulink - Simulation and Model-Based Design, Last accessed June 13, 2011. http://www.mathworks.com/products/simulink/.
- [6]MATLAB Support Package for Arduino https://www.mathworks.com/help/supportpkg/arduinoio/
- [7] Peter Corke, Robotics, Vision and Control, Fundamental Algorithms in MATLAB; pages 15-29, June 2011.
- [8] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control ,Second Edition, pages 61-64; January 28, 2004.
- [9] Paul, Richard (1981). Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators.
- [10]Spong, Mark W.; Vidyasagar, M. (1989). Robot Dynamics and Control. New York: John Wiley &Sons .
- [11]Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control ,Second Edition, pages 108-109; January 28, 2004.
- [12] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control ,Second Edition, pages 108-109; January 28, 2004.
- [13] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control ,Second Edition, pages 110-111; January 28, 2004.

- [14] Robot Manipulator Control Theory and Practice, Second Edition ,FRANK L.LEWIS

- [15] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Dynamics and Control ,Second Edition, pages 193-195; January 28, 2004.
- [16]Frautschi, Steven C.; Olenick, Richard P.; Apostol, Tom M.; Goodstein, David L. (2007). The Mechanical Universe: Mechanics and Heat, Advanced Edition (illustrated ed.). Cambridge University Press. p. 208
- [17]hyperphysics ,http://hyperphysics.phy-astr.gsu.edu/hbase/corf.html.
- [18] Ternes, Markus; Lutz, Christopher P.; Hirjibehedin, Cyrus F.; Giessibl, Franz J.; Heinrich, Andreas J. (2008-02-22). "The Force Needed to Move an Atom on a Surface". Science. 319
- [19]http://www.mogi.bme.hu/TAMOP/robot_applications/ch07.html
- [20] Robot Manipulator Control theory and Practice
- *[21] SlotineLi ,APPLIEDNONLINEAR CONTROL*
- *[22] SlotineLi ,APPLIEDNONLINEAR CONTROL*
- *[23]* Oussama Khatib Stanford University robotics course: Introduction of Robotics.
- [24]StigMoberg, Modeling and Control of Flexible Manipulators, Linköping studies in science and technology. Dissertations. No. 1349 ;pages 43-44.
- [25] StigMoberg, Modeling and Control of Flexible Manipulators, Linköping studies in science and technology. Dissertations. No. 1349 ;pages 44-46.
- [26] StigMoberg, Modeling and Control of Flexible Manipulators, Linköping studies in science and technology. Dissertations. No. 1349 ;page 52.
- [27] FRANK L.LEWIS, PH.D. Moncrief-O'Donnell Endowed Chair and Associate Director of Research Automation & Robotics Research Institute University of Texas, Arlington ,Robot Manipulator Control Theory And Practice, page 365.

- *[28]*Ngoc Dung Vuong, Marcelo H. AngJr,Dynamic Model Identification for Industrial Robot, pp 57-58
- [29] Shapiro, Linda and Stockman, George. "Computer Vision", Prentice-Hall, Inc. 2001
- [30] *Umbaugh, Scott E (2010). Digital image processing and analysis : human and computer vision applications with CVIPtools (2nd ed.). Boca Raton, FL: CRC Press.*
- [31] J. Canny (1986) "A computational approach to edge detection", IEEE Trans. Pattern Analysis and Machine Intelligence, vol 8, pages 679–714.
- [32] Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM, Vol. 15*, pp. 11–15 (January, 1972)
- [33] Mr.Naaman Abderrahmane, Vision courses.
- [34] Bouguet, J. Y. "Camera Calibration Toolbox for Matlab." Computational Vision at the California Institute of Technology. Camera Calibration Toolbox for MATLAB.
- [35] Zhang, Z. "A Flexible New Technique for Camera Calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22, No. 11, 2000, pp. 1330–1334.
- [36] Heikkila, J., and O. Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction." *IEEE International Conference on Computer Vision and Pattern Recognition*.1997.

- [37] Karl Johan Astrom,Bjorn Wittenmark.Adaptive Control (2nd Edition)