



# **Arab Academy for Science, Technology and Maritime Transport**

## **College of Engineering and Technology**

### **Electrical & Control Engineering Department**

### **Electronics & Communications Department**

### **Mechanical Engineering Department**

B. Sc. Final Year Project

## **HIGH SPEED ROBOTIC ARM AIDED WITH COMPUTER VISION**

### **Ping Pong Perception**

Presented By:

*Ahmed Tarek Fahmy*  
*Hossam SamirArafat*

*Hesham Mohamed Fadl*  
*Moumen Yasser*

Supervised By:

*Dr. Mostafa Abdel Galil*  
*Associate Professor*  
*Head Of Electrical And Control*  
*Engineering Department*

*Dr. Mohamed Tamazin*  
*Assistant Professor*  
*Electronics and Communication*  
*Engineering Department*

## DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in *(insert title of degree for which registered)* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

**Signed:** \_\_\_\_\_

**Registration No.:** \_\_\_\_\_

**Date:** Day, xx Month Year.

## ACKNOWLEDGMENTS

This project would have not been possible without the support of various people and organizations, and as such, we would like to deeply thank the following people:

*Dr. Ismail Abd El Ghafar*, The Arab Academy for Science and Technology's president – for his continued moral and financial support

Dr. Amr Ali, The Arab Academy for Science and Technology's Dean – for providing us with a wonderful lab and not shying away from any request.

Dr. Mostafa Abdel Galil and Dr. Mohamed Tamazin, our project supervisors for their support during every part of the project.

*Dr. Abo El Magd Nour El Din* - for providing us with the AC servo motor as well as its driver.

*Dr. Ahmed El Kabbany* - for helping us in funding the ZED cam as well as Dr. Hassan El Dib for lending us his powerful PC.

ITIDA (Information and Communication Technology Outsourcing) for sponsoring the project.

In addition, a special thanks to David who was doing the same project in his university in California, USA who helped with the path prediction algorithm and another special thanks to Mohamed Lotfy and Youssef Aly for helping us in designing, simulating and manufacturing the mechanical parts of the system.

## **ABSTRACT**

The rapid advancement in computer vision and image processing has allowed machines to be more perceptive of their environments than ever before. Many industries are now looking to rely on vision-actuated robots for complex tasks that require high speed and precision. Subsequently, the goal of this project is to design a robotic arm that is able to react to its environment using computer vision. In order to prove the efficiency of this system, it will be applied on an athletic robotic arm that can play Ping-Pong against a human player.

The project comprises of various subsystems that can be summarized as follows: A stereo vision and ball detection algorithm implemented on an Xbox Kinect and ZED stereovision camera. The algorithm detects the 3D position of the ball and supplies it to a path prediction algorithm that uses aerodynamic and rebound models of the ball to predict the position of the ball at the other end of the table and, consequently, where the robot needs to be in order to hit the ball. A 4-DOF robotic arm “Dobot” is mounted on a lead screw mechanism actuated by an AC induction motor. The mechanism moves the robot to the required position where it successfully hits the ball.

## Contents

<b>List of figures .....</b>	<b>VIII</b>
<b>List of tables .....</b>	<b>X</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Motivation .....	1
1.2 Objective .....	1
1.3 Literature Survey .....	1
1.4 Organization .....	3
<b>2 Background .....</b>	<b>4</b>
2.1 Image Processing and Computer Vision .....	4
2.1.1 Image acquisition .....	4
2.1.1.1 Sensor type .....	5
2.1.1.1.1 Charge-coupled device (CCD) Sensors .....	5
2.1.1.1.2 Complementary metal oxide semiconductor (CMOS) Sensors .....	5
2.1.1.2 Sensor size .....	6
2.1.1.3 Color sensing .....	7
2.1.1.4 Resolution .....	8
2.1.1.5 Frame rate .....	9
2.1.2 Image processing .....	10
2.1.3 Depth Sensing .....	11
2.1.3.1 Time of flight .....	11
2.1.3.2 Stereoscopic .....	12
2.1.3.3 Fixed Structured light .....	13
2.1.4 Shape Detection .....	13
2.2 Path Prediction .....	14
2.2.1 Need and Path Prediction Concept .....	14
2.2.2 Aerodynamic Model .....	14
2.2.3 Rebound Model .....	16
2.3 Robotic ARM .....	17
2.3.1 Robotic Arm selection .....	17
2.3.2 Inverse Kinematics .....	19
2.3.2.1 Solutions .....	20
2.3.2.1.1 Algebraic methods .....	20
2.3.2.1.2 Iterative methods .....	21
2.3.2.1.2.1 Jacobian Inverse Methods .....	22
2.3.2.1.2.2 Jacobian Pseudo-inverse .....	23
2.3.3 Stepper motor .....	24
2.3.3.1 Theory of Operation .....	24
2.3.3.2 Stepper motor Classifications [23] .....	28
2.3.3.2.1 Motor Size .....	28
2.3.3.2.2 Step Count .....	28
2.3.3.2.3 Gearing .....	28
2.3.3.2.4 Unipolar vs. Bipolar .....	29
2.4 Servo Mecahnism .....	29

2.4.1	Servo Motor .....	29
2.4.1.1	The importance of using a motor .....	29
2.4.1.2	Motor Selection .....	29
2.4.1.3	Theory of operation of the driver .....	30
2.4.1.4	Theory of operation of the motor .....	32
2.4.2	Linear Actuator .....	33
2.4.3	Strengths and Limitations: .....	33
2.4.4	What is a Ball Screw? .....	34
2.4.5	Why use a Ball Screw? .....	34
2.4.6	Ball Screw Accuracy: .....	34
2.4.7	Ball Screw Assembly: .....	35
2.4.8	Lifetime of a Ball Screw: .....	36
2.4.9	Ball screw speed: .....	36
2.4.10	Our Ball Screw specs: .....	37
<b>3</b>	<b>Proposed Prototype .....</b>	<b>38</b>
3.1	Description .....	38
3.2	Image Processing and Computer Vision .....	38
3.2.1	Improvements on Image processing: .....	38
3.2.2	Real-Time Object Tracking Without Colour .....	39
3.2.3	Improvements on Computer vision: .....	41
3.2.4	Kinect technical specifications .....	43
3.2.5	Fast moving object detection .....	43
3.2.6	Non-registered data .....	44
3.2.7	ZED camera .....	46
3.2.8	Registered data .....	47
3.3	Path Prediction .....	49
3.3.1	Flowchart .....	50
3.4	Robotic ARM .....	52
3.4.1	Instalment and Connection .....	52
3.4.2	Motor and Robotic Arm collaboration Algorithm .....	53
3.4.3	Robotic Arm Communication Algorithm .....	53
3.4.3.1	Standard Dobot Packet form .....	54
3.4.3.2	IEEE Standard for Binary Floating-Point Arithmetic .....	55
3.4.4	Robotic Arm Control Algorithm .....	55
3.5	Servo Mecahnism .....	56
3.5.1	Servo Motor .....	56
3.5.1.1	System Connection .....	56
3.5.1.2	Motor Control .....	57
3.5.1.2.1	Motor Communication Algorithm .....	57
3.5.1.2.2	Adjusting Mode of operation .....	58
3.5.1.2.2.1	Position control mode .....	58
3.5.1.2.2.2	Speed control mode .....	59
3.5.1.2.2.3	Torque control mode .....	59
3.5.1.2.3	Required Signal .....	60
3.5.1.2.4	Motor Control Algorithm .....	60
3.5.1.2.5	Electronic Gear Ratio .....	62

3.5.2	LINEAR ACTUATOR.....	62
3.5.2.2	Coupler Design: .....	63
3.5.2.3	Assembly: .....	63
<b>4</b>	<b>Experimental Work and Results .....</b>	<b>65</b>
4.1	Results.....	65
4.1.1	Image Processing and Computer Vision .....	65
4.1.2	Colour Detection .....	66
4.1.3	Colour and Shape Detection .....	68
4.1.4	Shape Detection.....	69
4.1.5	Path Prediction.....	70
4.1.6	Robotic Arm .....	72
4.1.7	Servo Mechanism.....	73
4.1.7.1	Servo Motor .....	73
4.1.7.2	Linear Actuator.....	73
4.1.8	Reaction Force and Moment on Constraints:.....	74
4.1.9	Results:.....	74
<b>5</b>	<b>Conclusion and future work .....</b>	<b>76</b>
5.1	Conclusion.....	76
5.2	Future Work.....	76
<b>6</b>	<b>Bibliography .....</b>	<b>77</b>

## LIST OF FIGURES

Figure 2-1 CCD sensor internal structure .....	5
Figure 2-2 CMOS sensor internal structure .....	6
Figure 2-3 Standard sensor sizes .....	6
Figure 2-4 Each color system is assigned with different scheme .....	7
Figure 2-5 Sample figure represented in multiple resolutions .....	8
Figure 2-6 Frame rate represents the number of images in a single second .....	9
Figure 2-7 Motion blur can remove details from an image .....	9
Figure 2-8 Inverse relation between FPS and Resolution .....	10
Figure 2-9 TOF system topology.....	11
Figure 2-10: Stereoscopic system topology.....	12
Figure 2-11: Stereo correspondence problem.....	12
Figure 2-12: Fixed structured light system topology .....	13
Figure 2-13: The ping pong table .....	18
Figure 2-14: The Jacobian solution is a linear approximation of the actual motion of the kinematic .....	22
Figure 2-15: The position of the six-pole rotor and four-pole stator of a typical stepper motor .....	25
Figure 2-16: Movement of the stepper motor rotor as current is pulsed to the stator.....	26
Figure 2-17: Schematic of the motor driver .....	31
Figure 2-18: The Stator of Permanent Magnet Synchronous Motor .....	32
Figure 2-19: The Rotor of Permanent Magnet Synchronous Motor .....	32
Figure 2-20 Ball Screw Mechanism .....	34
Figure 2-21 Lead Error .....	35
Figure 2-22 Screw Assembly .....	36
Figure 2-23 Screw Specs .....	37
Figure 3-1: HSV, RGB and gray scale color spaces .....	39
Figure 3-2: Motion detection trial .....	39
Figure 3-3: Normal and Thresholded Image .....	40
Figure 3-4: Intersection between Frames .....	40
Figure 3-5: Intersection highlighted .....	41
Figure 3-6: Hand can be detected in motion detection .....	41
Figure 3-7: Kinect Structure .....	41
Figure 3-8: Grains of Kinect IR Grid .....	42
Figure 3-9: Kinect Ball Points.....	44
Figure 3-10: Shift Between RGB and Depth image in Kinect.....	44



Figure 3-11.....	45
Figure 3-12: Zed Camera .....	46
Figure 3-13 The ZED camera itself has different resolution modes shown above in figure. ....	46
Figure 3-14: Speciation of The Laptop Used.....	47
Figure 3-15: Socket Communication used .....	48
Figure 3-16: Socket Communication Topology.....	48
Figure 3-17: Predicted Paths on MATLAB.....	51
Figure 3-18: Predicted Paths on MATLAB.....	51
Figure 3-19: layout of Actuation Algorithms .....	52
Figure 3-20: The robotic arm is installed on a carriage.....	53
Figure 3-21: Flow Chart of Robotic Arm Control Algorithm.....	56
Figure 3-22: Motor and Driver Connection Diagram .....	56
Figure 3-23 Motor PCB Board .....	57
Figure 3-24: Type of pulses .....	59
Figure 3-25: Required Signal .....	60
Figure 3-26: The detailed flow chart of Motor Control Algorithm .....	61
Figure 3-27 Designed Assembly .....	62
Figure 3-28 Designed Assembly with Robotic Arm.....	63
Figure 3-29 Inside Coupler Design.....	63
Figure 3-30 Outside Coupler design .....	63
Figure 3-31 Designed Coupler Connected to Motor.....	64
Figure 3-32 Robotic Arm Instalment on Designed Assembly.....	64
Figure 4-1: Top View of the table .....	65
Figure 4-2: ZED Ball Points .....	65
Figure 4-3 Color Detection .....	66
Figure 4-4 Color Detection Error .....	67
Figure 4-5 Color Detection mixed to Shape Detection .....	67
Figure 4-6 Accurate Detection between 2 deferent shapes .....	68
Figure 4-7 Accurate Detection between 2 similar shapes .....	68
Figure 4-8 Shape Detection.....	69
Figure 4-9: Kalman filtered path .....	70
Figure 4-10 Valid Prediction 1 .....	71
Figure 4-11 Valid prediction 2 .....	71
Figure 4-12 Normal distribution curve and Z scores.....	72
Figure 4-13: Frequency of the generated signal .....	73
Figure 4-14 Early Design Process Design .....	74
Figure 4-15 Fixed Constraint .....	74
Figure 4-16 Displacement analysis .....	75
Figure 4-17 Stress Analysis.....	75

## **LIST OF TABLES**

Table 2-1: The Choices of Robotic Arms.....	18
---	----

## *Chapter One*

# **1 INTRODUCTION**

## **1.1 MOTIVATION**

Computer vision has been the propellant behind multiple revolutionary technologies, from driverless cars to virtual reality. This project aims to integrate computer vision with robotics. Successfully actuating a robotic system -which has a very slow response- in high speeds is especially challenging.

## **1.2 OBJECTIVE**

The objective of our project is to integrate computer vision with robotics to provide very high response robotic arm employing a very efficient computer vision algorithm. In order to demonstrate this idea, we will apply it on an athletic robotic arm which can play ping pong against a human player as an example. We aim to develop a new high speed vision system for a table tennis robot to provide the landing point and striking point quickly and accurately using structured light camera (Kinect) and utilizing window of interest so we can decrease the time of the image processing in the experimental system. Thus our system will be able to handle fast flying balls with better accuracy. In addition to that; we will depend on colour detection and shape detection so that we can provide a system immune to any environment with variant intensity light.

## **1.3 LITERATURE SURVEY**

One of the early trials to develop a robotic arm that can play table tennis is the trial of Andersson, who constructed a robot system using an industrial robotic arm which can play table tennis with human [1] [2]. Because of the relative low speed of the robotic arm, later scientists designed their table tennis robots with special structure. Acosta et al [3] designed a low cost table

tennis robot with 5 degrees of freedom (DOF), which can play against a human opponent.

But this robot is not for standard table because of its limited hitting torque. Miyazaki et al [4] [5] [6] [7] developed a 4-DOF robot mounted on the table.

Vision system serves as eyes for table tennis robot, which provides the information of the ball. There are three categories of vision systems according to the number of cameras used. For example, the vision system of the table tennis robot in [2] adopted four cameras. Multi-camera brings many problems such as the complex calibration of the cameras and the synchronization. In addition, the vision system with multi-camera has to process more frames simultaneously, which decreases its real-time performance. The vision systems in [5] [8] all adopted binocular vision. For example, a stereo vision system named Quick MAG III was used in [5].

The binocular vision systems above have two common characteristics. Firstly, their ball tracking algorithms depend on the colour information of the ball which is sensitive to the light of the environment. Secondly, the frame rate of cameras used in these systems is not more than 60 frames per second (FPS), so fast flying ball may cause a failure because of the relatively low frame rate. Acosta [3] used monocular vision system, which computed the 3D position of the ball according to the image coordinates of the ball and its shadow on the table. Its real-time performance was improved since only single frame was needed to be processed at the same time. However, it required a stable light – controlled environment.

The efficient algorithms of image processing including the recognition and the feature extraction of the ball are essential [9] to ensure high-speed visual measurement and prediction of the ball trajectory for table tennis robot. Indeed, there are many other works on the dynamic object recognition [10] [11] [12] [13]. But they are not suitable for the ball recognition and tracking in the images over 100 FPS because of the high time cost in image processing.

It is really a hard job for robot to play table tennis against a person. Firstly, the vision system must detect the ball accurately in very short time. Secondly,

the landing and striking points of the ball should be predicted accurately and quickly with limited measured positions of the ball.

## **1.4 ORGANIZATION**

Chapter 1: Introduction: covers the motivation and objective of this project and provides the literature survey conducted in the beginning of our research process.

Chapter 2: Background: Provides the theoretical basis on which the project is built upon; this chapter goes into details regarding the 3 main parts of the project, namely, computer vision, path prediction, and actuation.

Chapter 3: Proposed Prototype: Details of the prototype designed to integrate computer vision with athletic robotics.

Chapter 4: Experimental Work and Results: Showcases the practical and numerical results achieved and goes about explaining the various scientific methods followed in order to achieve them.

Chapter 5: Conclusion and Future Work: Concludes the paper and outlines the future improvements and applications for the proposed prototype.

## **2 BACKGROUND**

### **2.1 IMAGE PROCESSING AND COMPUTER VISION**

In this section we will illustrate the Image acquisition, Image processing and 3D depth sensing techniques. The aim of this stage is to enhance the acquired image and perform successful ball recognition through several color filters as a first stage. As a second stage the ball center 3D position is acquired using Kinect structured light depth sensor and then multiple coordinates are passed to the path prediction algorithm.

#### **System elements**

Image acquisition	Image processing	Depth sensing
-------------------	------------------	---------------

#### **2.1.1 Image acquisition**

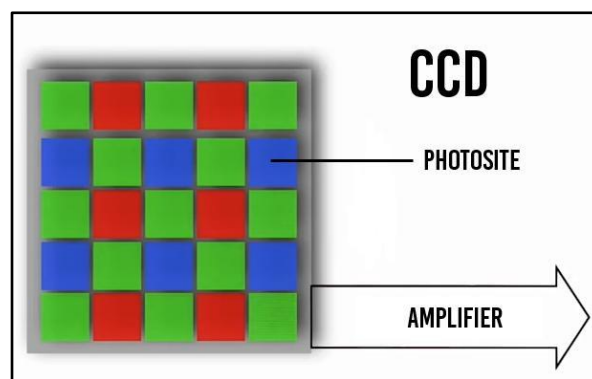
The core player of any vision system is the camera. Parameters that discriminate different cameras are numerous but we will consider here some of these parameters that are mostly commonly taken into consideration which are:

1. Sensor type ( CMOS or CCD )
2. Sensor size
3. Colour sensing ( Monochromatic or Chromatic)
4. Resolution ( Megapixels )
5. Frame rate / FPS ( Frame Per Second )

### 2.1.1.1 Sensor type

#### 2.1.1.1.1 Charge-coupled device (CCD) Sensors

CCD sensors composite of multiple photosite, the photosites are passive elements and they are used as receiving elements for image, once the photosite acquire the image, Photosite content is sent to a dedicated amplifier. No local processing is done inside a photosite. CCD are most common in scientific imaging applications, They are not recommended for high speed imaging application, due to the little utilization for CCD-Based cameras they are most often expensive.

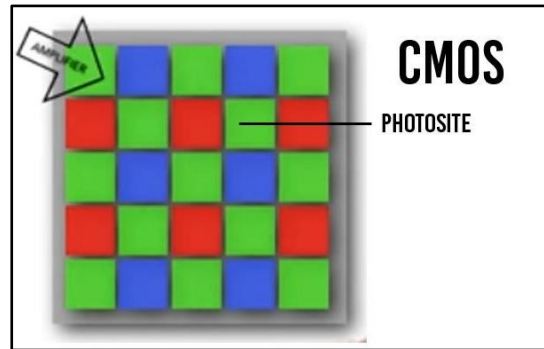


**Figure 2-1 CCD sensor internal structure**

#### 2.1.1.1.2 Complementary metal oxide semiconductor (CMOS) Sensors

CMOS sensors are composited of a grid of photosites with and integrated amplifier beneath each photosite as shown. These sensors are used in wide range of applications. The high utilization level for this fabrication technology is based on several reasons:

- CMOS sensors are faster with less noise
- CMOS sensors utilize amplifier with less bandwidth compared to those used by CCD
- Compared to CCD, They are smaller in size ( High portability )
- Mass production of CMOS sensors reduces its cost.

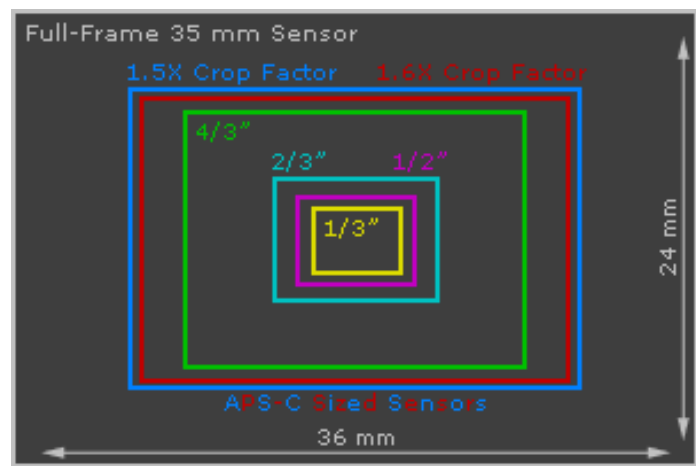


**Figure 2-2 CMOS sensor internal structure**

### 2.1.1.2 Sensor size

Sensor size can tell us how much of the scene the camera could see, large sensor size means large portion of scene is captured. The highest sensor size is called Full-frame sensors, below this type there several types of various crop factor.

The size of the camera itself is a function of sensor size, so for small and compact cameras small sensors are utilized.



**Figure 2-3 Standard sensor sizes**



### 2.1.1.3 Color sensing

Monochromatic sensors (are also called grayscale sensors ) they composite image as a number of single channel 8 bit pixels with a range of 0 - 255, Which correspond to different greyscale values from pure black at 0 to pure white at 255.

Monochromatic imaging is used due to the less image content in a grayscale picture. The fewer amounts of data enhance the speed of processing due to the reduced number of computation cycles need.

Chromatic sensors composed of a 3 main components which are Red ,Green and Blue (RGB), The mixture of these 3 values can lead to any possible color combination, On the other side, more computation power is needed as the image content had been tripled compared to grayscale. The choice between both types is process dependent, if image processing applied is color based then color sensors are preferred.

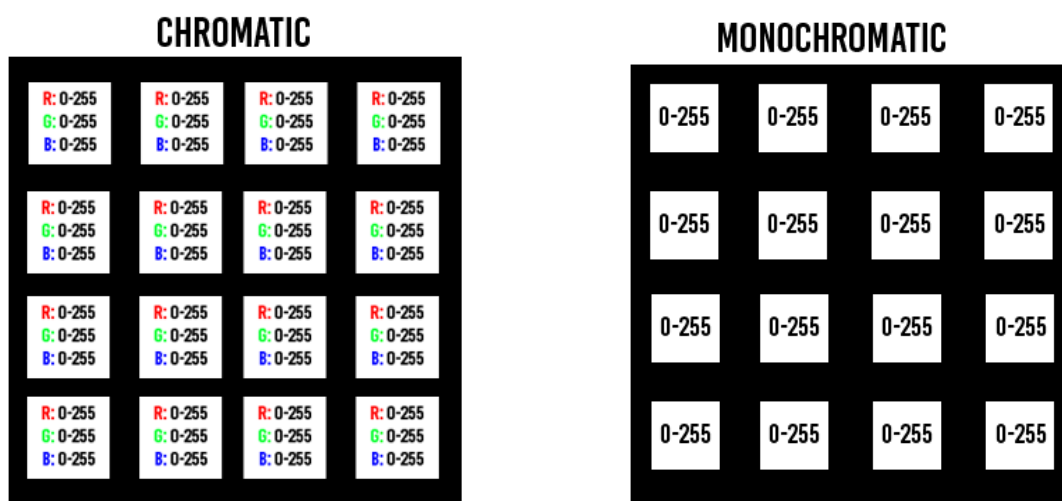
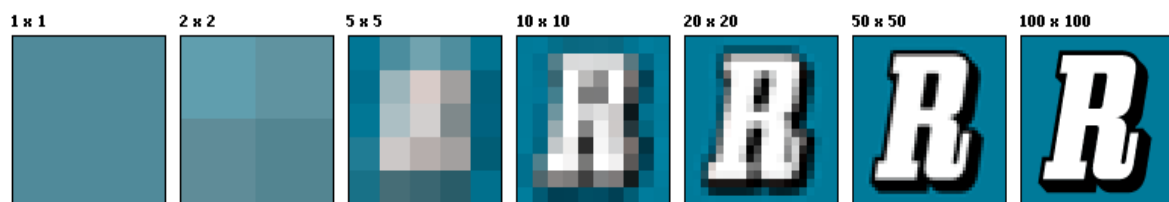


Figure 2-4 Each color system is assigned with different scheme

### 2.1.1.4 Resolution

Resolution is the capability of the sensor to observe or measure the smallest object clearly with distinct boundaries. There is a difference between the resolution and a pixel. A pixel is actually a unit of the digital Resolution depends upon the size of the pixel. Usually, with any given lens setting, the smaller the size of the pixel, the higher the resolution will be and the clearer the object in the image will be. Images having smaller pixel sizes might consist of more pixels. The number of pixels correlates to the amount of information within the image.



**Figure 2-5 Sample figure represented in multiple resolutions**

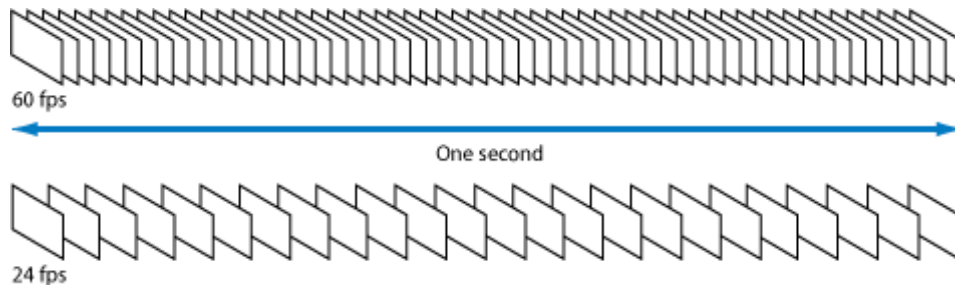
Higher resolution means more image clarity and more pixel count. An image that is 2048 pixels in width and 1536 pixels in height has a total of  $2048 \times 1536 = 3,145,728$  pixels or 3.1 megapixels.

One could refer to it as 2048 by 1536 or a 3.1-megapixel image. Or, you can think of it as a very low quality image (72ppi) if printed at about 28.5 inches wide, or a very good quality (300ppi) image if printed at about 7 inches wide.

Pixel count or Resolution is directly related to utilized FPS as we will discuss in a following section, As resolution increase image quality increase but processing time for a given frame also increase, So higher resolutions can cause reaching a CPU bound condition.

### 2.1.1.5 Frame rate

Frame rate can be simply defined as the number of images grabbed by the camera each second.



**Figure 2-6 Frame rate represents the number of images in a single second**

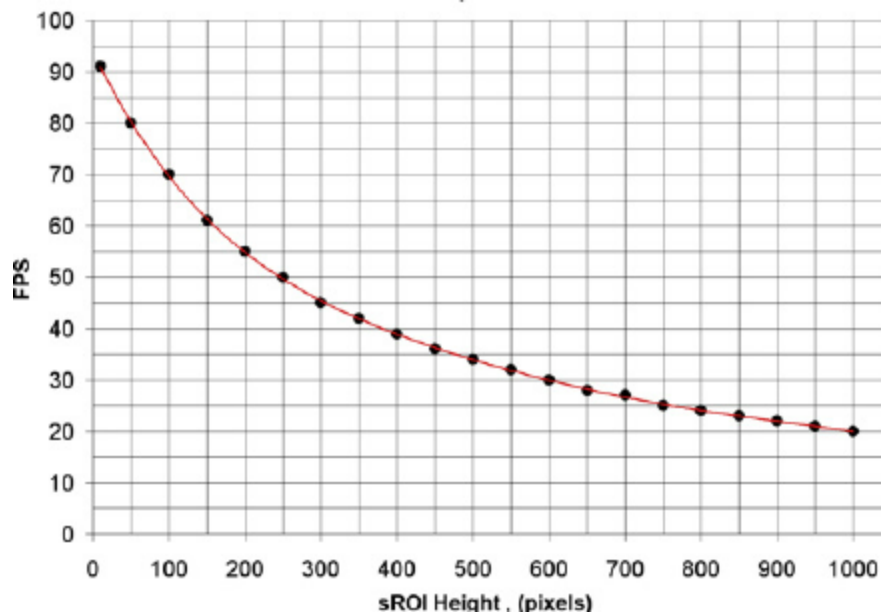
The frame rate is an important factor in high speed applications as more frame rate means less motion blur, the motion blur can reflect negatively on the results obtained after image processing is done. As shown two cameras with different FPS can lead to different results. Motion blur can reduced the details level in an image and this can lead to increase the complexity for any further feature extraction as shape detection or even color filtering.

Higher FPS is recommended in most vision systems but here comes an important trade-off of FPS versus resolution.



**Figure 2-7 Motion blur can remove details from an image**

Each camera has its own FPS/Pixels curve but all graphs are characterized by the nearly inverse relation, which means that increase of FPS lead to decrease in resolution and this is a logical result since in lower resolutions number of pixels to be transmitted is less hence the time of transmitting is less which allow here FPS.



**Figure 2-8 Inverse relation between FPS and Resolution**

### **2.1.2 Image processing**

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time vision. Originally developed by Intel's research center in Nizhny Novgorod (Russia), it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform which means it can be used on different operating systems.

#### **OpenCV have multiple advantages which are:**

- One of the powerful, platform independent library and supports C++, Java, Python.
- Provides good support for basic shape detection e.g. circle, rectangle etc.
- Video support is good, compatible with almost all the webcams.

- If you want to further go into feature detection like detection exact shape of the face like a contour representing lips or eyelash there are many algorithm available like Active shape model, active appearance model which gives pretty decent results.
- Widely used in augmented reality application.
- Support camera calibration.
- Online support is good. (Stackoverflow.com)

So OpenCV is considered the most probable candidate for any image processing and Computer vision project due to its high integrity with several libraries and hardware.

### 2.1.3 Depth Sensing

#### 2.1.3.1 Time of flight

This depth technique is based on the emission of a modulated light signal and measuring the reflected signal and measures its phase shift, the phase shift is then compared to a phase map to determine the depth of each point in 3D space, this most is accurate but it is expensive.

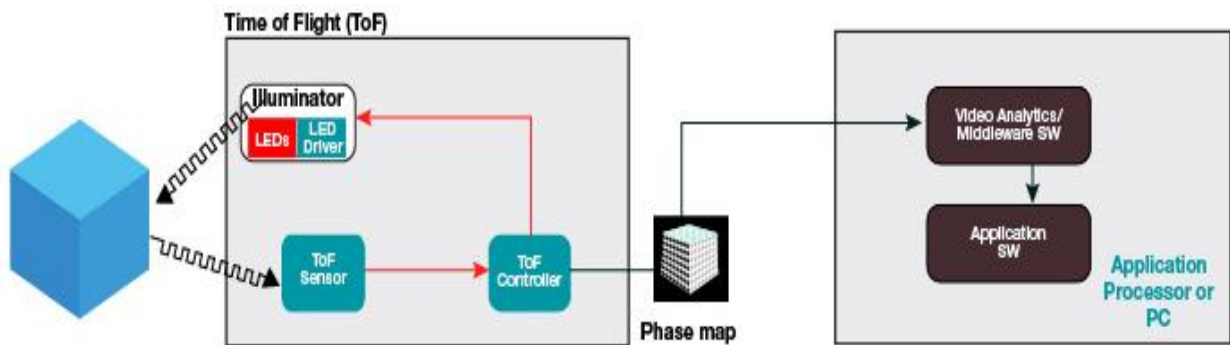
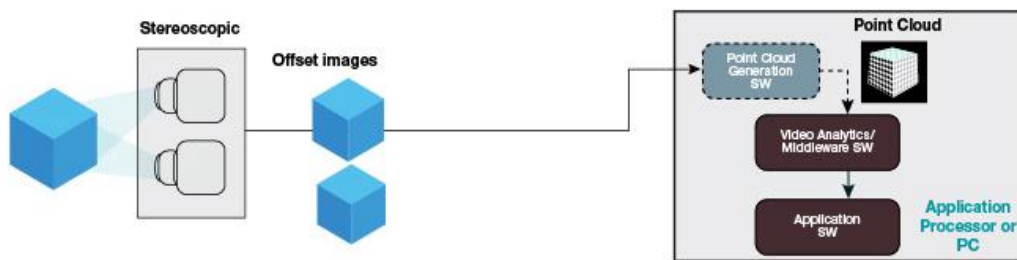


Figure 2-9 TOF system topology

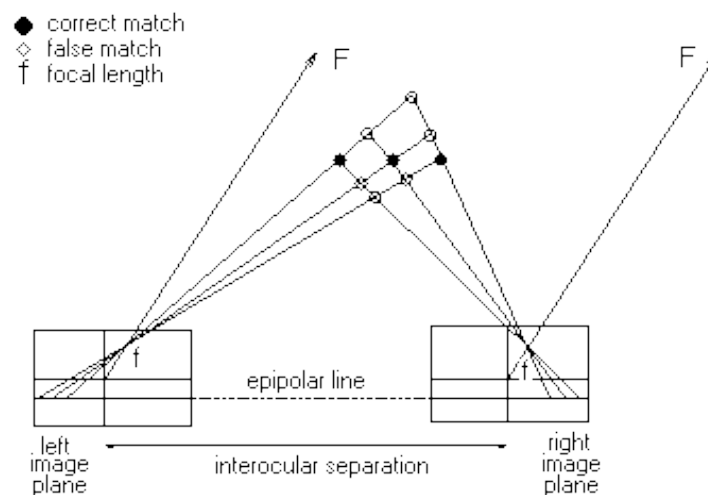
### 2.1.3.2 Stereoscopic

Stereoscopic depth sensing is based on the disparity between two fixed cameras, this system mimics the human vision system, and by the utilization of the epipolar geometry the true 3D position for a single point can be obtained.



**Figure 2-10: Stereoscopic system topology**

The most common problem for this method is a popular problem called stereo correspondence problem, several algorithm are devoted to enhance the pixel correspondence in two different image planes, the more accurate the algorithm the more correct matches occur. ZED camera and BumbleBee are commercial stereovision cameras.



**Figure 2-11: Stereo correspondence problem**

### 2.1.3.3 Fixed Structured light

The structured light method is dependent on projection of a fixed IR pattern, the reflected image of this pattern is processed to deduce the change path and pattern size, According to this changes depth map is formed. Most popular structured light camera is the Kinect.

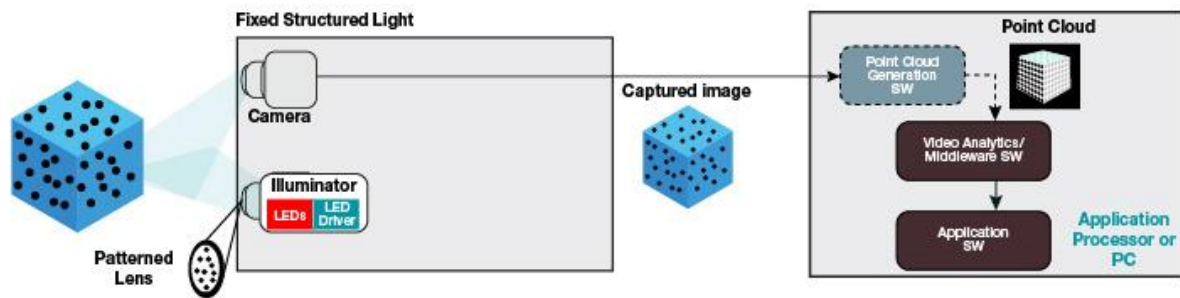


Figure 2-12: Fixed structured light system topology

### 2.1.4 Shape Detection

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. Circle Hough Transform (CHT) The Hough transform can be used to determine the parameters of a circle when a number of points that fall on the perimeter are known. A circle with radius  $R$  and center  $(a, b)$  can be described with the parametric equations  $x = a + R \cos(\theta)$  and  $y = b + R \sin(\theta)$ . When the angle  $\theta$  sweeps through the full 360 degree range the points  $(x, y)$  trace the perimeter of a circle. If an image contains many points, some of which fall on perimeters of circles, then the job of the search program is to find parameter triplets  $(a, b, R)$  to describe each circle. The fact that the parameter space is 3D makes a direct implementation of the Hough technique more expensive in computer memory and time.

## 2.2 PATH PREDICTION

### 2.2.1 Need and Path Prediction Concept

Due to the short length of our robotic arm with respect to the ping pong table, and the high speed nature of the ping pong sport compared to the response time of robotic actuators, we had to implement a prediction algorithm that would output the final position of the ball, only seconds after the human player hits it.

The algorithm employs two differential equations: one that describes the motion of the ball during flight and another for when the ball rebounds off the table, in order to calculate the position of the ball, at any given time:

$$x(t)$$

The ping pong ball can be easily assumed to be a projectile. The equation that describes the motion of a projectile is simply Newton's second law of motion:

$$\sum F = m x(t)'' \quad (1)$$

Continuing with our assumption, and ignoring air resistance, the only force acting on any falling body is the force of gravity  $F_g = mg$  :

$$\sum F = mg = m x(t)'' \quad (2)$$

However, it did not take several testing scenarios to help conclude that an algorithm that ignores air resistance and the subsequent drag force -especially for a body of small mass moving at high speeds- will never output accurate results, which defeats the purpose of what we are trying to achieve. Thus, we had to consider all kinds of aerodynamic forces that act on the ball during flight [18], [19].

### 2.2.2 Aerodynamic Model

Aerodynamics tells us that a translating and rotating ball moving through a fluid with a given speed  $x(t)'$  will be under the influence of two main forces:



- A resistive force, that depends on the fluid properties, specifically its density  $\rho$  and viscosity  $\mu$  and the ball's surface area.
- A lift force, due to the Magnus effect, named after the German physicist who accurately described the effect, although it had already been observed by Sir Isaac Newton as he studied the motion of tennis balls in Cambridge College.

The resistive force is the drag force:

$$F_d = -\frac{1}{2} C_D \rho_a \pi r^2 ||v|| v'$$

Where  $C_D$  is empirical constant that is measured by experimentation,  $\rho_a$  is the air's density and  $\pi r^2$  is the surface area of the ball.

Since all of these values are constant for the ping pong ball, we can group set constants into a single constant  $K_d$ , which yields:

$$F_d = -\frac{1}{2} K_d ||v|| v' \quad (3)$$

The Magnus force can be described by the equation:

$$F_m = C_m \rho_a \pi r^3 \omega \times v$$

Where  $C_M$  is an empirical constant measured by experimentation,  $\rho_a$  is the air's density and  $\frac{4}{3} \pi r^3$  is the volume of the ball. Since all of these values are constant for the ping pong ball, we can group set constants into a single constant  $K_m$ , which yields:

$$F_m = K_m \omega \times v \quad (4)$$

Supplementing the effect of both forces in equation (2) yields:

$$\sum F = F_g + F_d + F_m = m x(t)''$$

Substituting the expression of each force and solving for the acceleration:

$$x(t)'' = g + K_m \omega \times x(t)' - \frac{1}{2} K_d ||x(t)'|| x(t)' \quad (5)$$

It is important to note that the second term of the equation, the Magnus force, is a function of the ball's angular velocity,  $\omega$ , and on implementing the aerodynamic model, this paper did not adopt any sensory equipment or empirical methods to measure the ball's angular velocity,  $\omega$ . As such, after removing the Magnus force from the aerodynamic model, equation (5) becomes:

$$\ddot{x}(t) = g - \frac{1}{2} K_d \left| \dot{x}(t) \right| \dot{x}(t) \quad (6)$$

We now have a differential equation that describes the acceleration of the ping pong ball as a function of the different forces acting upon it during flight. In order to compute the position of the ball at any given time,  $x(t)$ .

We simply integrate equation (5) twice numerically to compute the position:

$$x(t) = x(t) + \dot{x}(t) (\Delta t) \quad (7)$$

The aerodynamic model fails to account for the Impulse force that affects the ball as it hits the table and the subsequent change in kinetic energy [14] [15].

### 2.2.3 Rebound Model

A different model is needed in order to accurately predict the change of the ball's velocity after it rebounds off the table. The ball's rebound can be modeled as an inelastic collisions between a small, fast moving body and a massive, frictionless, immobile one.

According to the laws of Impact mechanics, the ball's emergent velocity  $v_e$  can be related to its incident velocity  $v_i$  through the coefficient of restitution  $e$  using the law:

$$v_e = v_i e \quad (8)$$

As the collision is an inelastic one, the coefficient of restitution is consequently less than 1. According to the International Table Tennis Federation's (ITTF) rules, a standard table tennis (or ping pong) ball should

bounce up 24-26 cm when dropped from a height of 30.5 cm onto a standard steel block.

This translates into a coefficient of restitution from 0.89 to 0.92. However, there is not a strict and standardized system across manufacturers; those numbers vary greatly depending on where and when the ball was manufactured.

This paper accepts the results of another paper entitled “Aerodynamic Effects in a Dropped Ping-Pong Ball Experiment” which calculated the coefficient of restitution to be:

$$\begin{aligned} 0.878 \leq e \leq 0.914 & \text{ when dropped onto a standard steel block and} \\ 0.849 \leq e \leq 0.905 & \text{ when dropped onto an approved table tennis} \\ & \text{table} \end{aligned}$$

The average value of the first range is 0.896 and the average of the second range is 0.877. Averaging both values –to ensure obtain a number with minimum variation- yields 0.887. This is the value that will be used throughout this paper.

## **2.3 ROBOTIC ARM**

### **2.3.1 Robotic Arm selection**

The ping pong table has a length of 2.74 meters and a width of 1.525 meters, as shown in Figure 2-13 and if we require that our robot hits any ball that falls in his half with a height of 20 cm above the table, it that would mean that we require a 6-degree-of-freedom robot that can move:

- 1.525 meters left and right
- 1.37-meters forward

- Stands 1-meter high



**Figure 2-13: The ping pong table**

Consequently, industrial robots would be perfect for our application as their large workspace and fast, accurate motion can hit the ping pong ball from virtually anywhere on the table. However, industrial robots such as the KUKA costs around \$90,000 and even the used ones cost around \$10,000 which is significantly above our budget.

Therefore, we had to search the market for robotic arms with acceptable length, performance and pricing. After a comprehensive search, we found three convenient options shown in Table 2-1:

**Table 2-1: The Choices of Robotic Arms**

<b>Robot</b>	<b>Eva (Automata) [16]</b>	<b>7-Bot [17]</b>	<b>DoBot [18]</b>
<b>Reach</b>	600 mm full span radius	434 mm	320 mm full span radius
<b>Repeatability</b>	1 mm	-	$\pm 0.1$ mm
<b>Price</b>	\$3,000	\$350	900\$
<b>Max. Speed</b>	100° / sec	-	115° / sec
<b>Payload</b>	0.75Kg	0.5Kg	0.5Kg
<b>Weight</b>	3Kg	2.5Kg	2.7Kg

**Picture of  
the Robot**



It is evident in the comparison that the DoBot is our best option, as it can carry our payload with the highest speed, and the maximum accuracy (0.2mm repeatability) using stepper motors, moreover, it is the lightest robotic arm in relation to the alternatives and has extensive backing from the open source community, as the company launched their product on Kickstarter and based their firmware on the highly reviewed –and widely used Marlin and grbl.

All in all, the DoBot offers the best value for money and the easiest control interface, as it can be controlled using an Arduino; a microprocessor that was primarily made for the non-technical. The company's website and the Github community offer multiple sample projects where the DoBot was controlled using Python and a Raspberry Pi microcontroller, the powerful graphical language Processing, even a RAMPS driver board on an external Arduino Mega, overriding the main controller [18].

### **2.3.2 Inverse Kinematics**

Inverse Kinematics (IK) is the problem of determining a set of appropriate joint configurations for which the end effectors move to desired positions as smoothly, rapidly, and as accurately as possible. During the last decades, several methods and techniques, sophisticated or heuristic, have been presented to produce fast and realistic solutions to the IK problem. However, most of the currently available methods suffer from high computational cost and production of unrealistic poses.

Inverse Kinematics (IK) is a method for computing the posture via estimating each individual degree of freedom in order to satisfy a given task; it plays an

important role in the computer animation and simulation of articulated figures. Inverse Kinematics finds applications in several areas. IK methods have been implemented in many computer graphic and robotics applications, aiming to animate or control different virtual creatures. They are also very popular in the video games industry.

The field of computer-aided ergonomics is also concerned with articulated figures, especially human models developed for simulation and prediction purposes. The need for accurate biomechanical modelling and body sizing based on anthropometric data make IK methods a popular approach for fast and reliable solution. IK has been used in rehabilitation medicine in order to observe asymmetries or abnormalities. Recently, IK techniques have also been applied in protein science for protein structure prediction [19].

The state of the articulated structure is represented by a state vector  $\theta(\theta_1, \dots, \theta_n)$  in the joint (angular) space. The position and the orientation of the end effector are represented by the end effector position  $X_{EE}(x, y, z, \theta_x, \theta_y, \theta_z)$  in Cartesian space. The relations between the state vector  $\theta$  and the end effector position  $X_{EE}$  are expressed these equations:

#### **Forward Kinematics**

$$X_{EE} = f(\theta)$$

#### **Inverse Kinematics**

$$(\theta) = f^{-1}(X_{EE})$$

### **2.3.2.1 Solutions**

A number of methods and their combinations could be used to solve the inverse kinematics. Using each of them separately brings some advantages and also disadvantages. Therefore it is useful to combine them together and often combine them with additional approaches.

#### **2.3.2.1.1 Algebraic methods**

The algebraic solution exists only for a restricted class of cases. The joint angles could be expressed using the end effector position. The number of nonlinear equations increases with the DOFs. Each joint angle - one by one - could be solved by the system of n equations:

$$\prod_{i=2}^n A_{i-1}^i(\theta_i) = A_1^{-1}(\theta_1) \cdot A_n$$

The  $m+1$ th joint angle is expressed using the  $m$  previous angles. The last joint angle is expressed only by the end effector position and orientation.

The forward kinematics solution for the end effector position  $X_{EE}(x, y)$  in 2 DOF structure could be expressed as follows:

$$X_{EE} = (l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2), (l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)))$$

Thus by applying elementary trigonometry the inverse solution can be found. As the DOFs for most cases are higher the state vector is not analytically expressible using such easy way. Therefore more sophisticated approaches are necessary [19].

#### 2.3.2.1.2 Iterative methods

Iterative methods solve the inverse kinematics problem by using a sequence of steps leading to incrementally better solution for the joint angles. The goal is to minimize the difference between the current and desired positions of the end effector. There are several iterative methods which can solve inverse kinematics problems as:

- Jacobian Inverse Methods
  - Jacobian Pseudo-inverse
  - Jacobian
  - Singular Value Decomposition
  - Damped Least Squares
- Newton Methods
- Heuristic Inverse Kinematics Algorithms
  - Cyclic Coordinate Descent
  - Triangulation Inverse Kinematics
  - Sequential Inverse Kinematics
- FABRIK: Forward And Backward Reaching Inverse Kinematics

### 2.3.2.1.2.1 Jacobian Inverse Methods

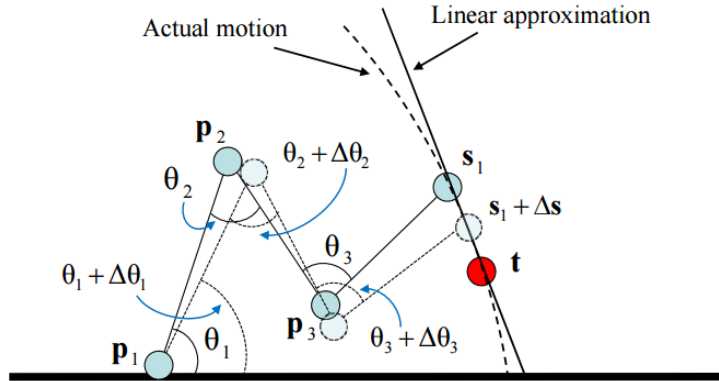
The Jacobian  $J$  is a matrix of partial derivatives of the entire chain system relative to the end effectors  $s$ . The Jacobian solutions are a linear approximation of the IK problem as shown in Figure 2-14 they linearly model the end effectors' movements relative to instantaneous system changes in link translation and joint angle. The Jacobian matrix  $J$  is a function of the  $\theta$  values and is defined by

$$J(\theta)_{ij} = \left( \frac{\partial s_i}{\partial \theta_j} \right)_{ij}$$

Where  $i = 1, \dots, k$  and  $j = 1, \dots, n$ . The Jacobian matrix entries for the  $j$ -th rotational joint can be calculated as follows

$$\left( \frac{\partial s_i}{\partial \theta_j} \right) = v_j * (s_i - p_j)$$

Where  $p_j$  is the position of the joint, and  $v_j$  is the unit vector pointing along the current axis of rotation for the joint.



**Figure 2-14: The Jacobian solution is a linear approximation of the actual motion of the kinematic**

Forward dynamics can now be written as

$$\dot{\vec{s}} = J(\theta) \dot{\theta}$$

Where the dot notation specifies the first derivative with respect to time. Using the current values  $\theta$ ,  $\vec{s}$  and  $\vec{t}$ , the Jacobian  $J = J(\theta)$  can be computed. We



then seek an update value  $\Delta\theta$  for the purpose of incrementing the joint angles  $\theta$  by  $\Delta\theta$ :

$$\theta := \theta + \Delta\theta$$

The change in end effector positions caused by this change in joint angles can be estimated as

$$\Delta\vec{s} \approx J\Delta\theta$$

The idea is that the  $\Delta\theta$  value should be chosen so that  $\Delta\vec{s}$  is approximately equal to  $\vec{e}$ , although it is also common to choose  $\Delta\theta$  so that the approximate movement  $\Delta\vec{s}$  in the end effectors (partially) matches the velocities of the target positions.

Thus, the FK problem can be expressed as  $\vec{e} = J\Delta\theta$  and the IK problem can be rewritten as  $\Delta\theta = J^{-1} \vec{e}$ . In most cases, the IK equation cannot be solved uniquely. Indeed, the Jacobian  $J$  may not be square or invertible, and even if it is invertible,  $J$  may work poorly as it may be nearly singular. Several approaches have been proposed to overcome these problems [20].

#### 2.3.2.1.2.2 Jacobian Pseudo-inverse

The Jacobian Pseudo-inverse, also known as the Moore-Penrose inverse of the Jacobian, sets the value  $\Delta\theta$  equal to

$$\Delta\theta = J^{-1} \vec{e}$$

Where  $J^{-1}$  is an  $n \times m$  matrix and is called the pseudo-inverse of  $J$ . It is defined for all matrices  $J$ , even ones which are not square or not of full row rank. The pseudo-inverse gives the best possible solution to the equation  $J\Delta\theta = \vec{e}$  in the least squares sense [20].

The pseudo-inverse has the property that the matrix  $(I - J^{-1} J)$  performs a projection onto the null space of  $J$ . Therefore, for all vectors  $\phi$ ,  $J(I - J^{-1} J)\phi = 0$ . This means that we can set  $\Delta\theta$  by

$$\Delta\theta = J^{-1} \vec{e} + (I - J^{-1} J)\phi$$

For any vector  $\phi$  and still obtain a value for  $\Delta\theta$  which minimises the value  $J\Delta\theta - \vec{e}$ . Several authors have used the nullspace method to help avoid singular. The pseudo-inverse method can be derived as follows:

$$J^T J \Delta\theta = J^T \vec{e}$$

Then let  $\vec{z} = J^T \vec{e}$  and solve the equation

$$J^T J \Delta\theta = \vec{z}$$

It can be shown that  $\vec{z}$  is always in the range of  $J^T J$ , hence the above equation always has a solution. When  $J$  is full row rank,  $J^T J$  or  $J J^T$  is guaranteed to be invertible. In this case, the minimum magnitude solution  $\Delta\theta$  can be expressed as

$$\Delta\theta = (J^T J)^{-1} J^T \vec{e} \equiv J^T (J J^T)^{-1} \vec{e}$$

### 2.3.3 Stepper motor

Maybe the stepper motor is not the best option to be used in robotic arms as it has an open loop control and doesn't provide feedback of position. But according to our budget a robotic arm with a servo motor was not affordable. So we had to use the stepper motor and understand its theory of operation and its control method in order to get the maximum benefit from it.

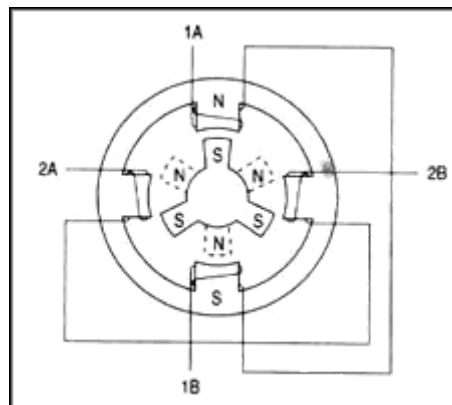
A stepper motor is an electro mechanical device, which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation of input pulses applied [21].

#### 2.3.3.1 Theory of Operation

Stepper motors provide a means for precise positioning and speed control without the use of feedback sensors. The basic operation of a stepper motor allows the shaft to move a precise number of degrees each time a pulse of

electricity is sent to the motor. Since the shaft of the motor moves only the number of degrees that it was designed for when each pulse is delivered, you can control the pulses that are sent and control the positioning and speed. The rotor of the motor produces torque from the interaction between the magnetic field in the stator and rotor. The strength of the magnetic fields is proportional to the amount of current sent to the stator and the number of turns in the windings.

The stepper motor uses the theory of operation for magnets to make the motor shaft turn a precise distance when a pulse of electricity is provided. Figure 2-15 shows a typical cross-sectional view of the rotor and stator of a stepper motor. From this diagram you can see that the stator (stationary winding) has four poles, and the rotor has six poles (three complete magnets). The rotor will require 12 pulses of electricity to move the 12 steps to make one complete revolution. Another way to say this is that the rotor will move precisely  $30^\circ$  for each pulse of electricity that the motor receives. The number of degrees the rotor will turn when a pulse of electricity is delivered to the motor can be calculated by dividing the number of degrees in one revolution of the shaft ( $360^\circ$ ) by the number of poles (north and south) in the rotor. In this stepper motor  $360^\circ$  is divided by 12 to get  $30^\circ$ .



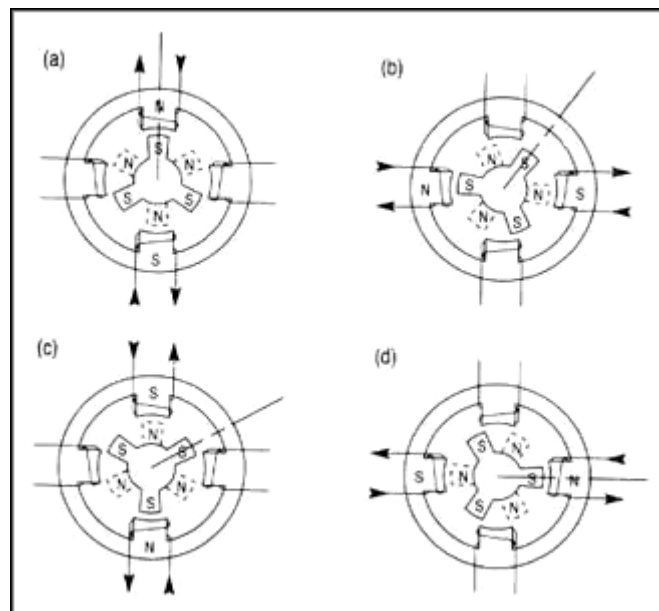
**Figure 2-15: The position of the six-pole rotor and four-pole stator of a typical stepper motor**

When no power is applied to the motor, the residual magnetism in the rotor magnets will cause the rotor to detent or align one set of its magnetic poles with the magnetic poles of one of the stator magnets. This means that the rotor will have 12 possible detent positions. When the rotor is in a detent

position, it will have enough magnetic force to keep the shaft from moving to the next position. This is what makes the rotor feel like it is clicking from one position to the next as you rotate the rotor by hand with no power applied.

When power is applied, it is directed to only one of the stator pairs of windings, which will cause that winding pair to become a magnet. One of the coils for the pair will become the North Pole, and the other will become the South Pole. When this occurs, the stator coil that is the North Pole will attract the closest rotor tooth that has the opposite polarity, and the stator coil that is the South Pole will attract the closest rotor tooth that has the opposite polarity. When current is flowing through these poles, the rotor will now have a much stronger attraction to the stator winding, and the increased torque is called holding torque.

By changing the current flow to the next stator winding, the magnetic field will be changed  $90^\circ$ . The rotor will only move  $30^\circ$  before its magnetic fields will again align with the change in the stator field. The magnetic field in the stator is continually changed as the rotor moves through the 12 steps to move a total of  $360^\circ$ . Figure 2-16 shows the position of the rotor changing as the current supplied to the stator changes.



**Figure 2-16: Movement of the stepper motor rotor as current is pulsed to the stator**

Figure 2-16-a you can see that when current is applied to the top and bottom stator windings, they will become a magnet with the top part of the winding being the North Pole, and the bottom part of the winding is the south pole. You should notice that this will cause the rotor to move a small amount so that one of its south poles is aligned with the north stator pole (at the top), and the opposite end of the rotor pole, which is the north pole, will align with the south pole of the stator (at the bottom). A line is placed on the south-pole piece that is located at the 12 o'clock position in Fig. II-58a so that you can follow its movement as current is moved from one stator winding to the next.

Figure 2-16-b current has been turned off to the top and bottom windings, and current is now applied to the stator windings shown at the right and left sides of the motor. When this occurs, the stator winding at the 3 o'clock position will have the polarity for the south pole of the stator magnet, and the winding at the 9 o'clock position will have the north-pole polarity. In this condition, the next rotor pole that will be able to align with the stator magnets is the next pole in the clockwise position to the previous pole. This means that the rotor will only need to rotate  $30^\circ$  in the clockwise position for this set of poles to align itself so that it attracts the stator poles.

Figure 2-16-c you can see that the top and bottom stator windings are again energized, but this time the top winding is the south pole of the magnetic field and the bottom winding is the North Pole. This change in magnetic field will cause the rotor to again move  $30^\circ$  in the clockwise position until its poles will align with the top and bottom stator poles. You should notice that the original rotor pole that was at the 12 o'clock position when the motor first started has now moved three steps in the clockwise position.

Figure 2-16-d you can see that the two side stator windings are again energized, but this time the winding at the 3 o'clock position is the north pole. This change in polarity will cause the rotor to move another  $30^\circ$  in the clockwise direction. You should notice that the rotor has moved four steps of  $30^\circ$  each, which means the rotor has moved a total of  $120^\circ$  from its original position. This can be verified by the position of the rotor pole that has the line on it, which is now pointing at the stator winding that is located in the 3 o'clock position [22].

### **2.3.3.2 Stepper motor Classifications [23]**

#### **2.3.3.2.1 Motor Size**

One of the first things to consider is the work that the motor has to do. As you might expect, larger motors are capable of delivering more power. Stepper motors come in sizes ranging from smaller than a peanut to big NEMA 57 monsters.

NEMA 17 is a common size used in 3D printers and smaller CNC mills. Smaller motors find applications in many robotic and animatronic applications. The larger NEMA frames are common in CNC machines and industrial applications.

The NEMA numbers define standard faceplate dimensions for mounting the motor. They do not define the other characteristics of a motor. Two different NEMA 17 motors may have entirely different electrical or mechanical specifications and are not necessarily interchangeable.

#### **2.3.3.2.2 Step Count**

The next thing to consider is the positioning resolution you require. The number of steps per revolution ranges from 4 to 400. Commonly available step counts are 24, 48 and 200. Resolution is often expressed as degrees per step. A  $1.8^\circ$  motor is the same as a 200 step/revolution motor.

The trade-off for high resolution is speed and torque. High step count motors top-out at lower RPMs than similar size. And the higher step-rates needed to turn these motors results in lower torque than a similar size low-step-count motor at similar speeds.

#### **2.3.3.2.3 Gearing**

Another way to achieve high positioning resolution is with gearing. A 32:1 gear-train applied to the output of an 8-steps/revolution motor will result in a 512 step motor.

A gear train will also increase the torque of the motor. Some tiny geared steppers are capable of impressive torque. But the trade-off of course is speed. Geared stepper motors are generally limited to low RPM application.

#### **2.3.3.2.4 Unipolar vs. Bipolar**

Unipolar drivers, always energize the phases in the same way. One lead, the "common" lead, will always be negative. The other lead will always be positive. Unipolar drivers can be implemented with simple transistor circuitry. The disadvantage is that there is less available torque because only half of the coils can be energized at a time.

Bipolar use H-bridge circuitry to actually reverse the current flow through the phases. By energizing the phases with alternating the polarity, all the coils can be put to work turning the motor.

A two phase bipolar motor has 2 groups of coils. A 4 phase unipolar motor has 4. A 2-phase bipolar motor will have 4 wires - 2 for each phase. Some motors come with flexible wiring that allows you to run the motor as either bipolar or unipolar.

## **2.4 SERVO MECAHNISM**

### **2.4.1 Servo Motor**

#### **2.4.1.1 The importance of using a motor**

Using a sliding mechanism in our system was very essential to provide a very fast and smooth robotic system. In addition to that, it enables us to use a short robotic arm, which would have less cost than tall ones. Thus we can provide a low cost solution. In order to use of a sliding mechanism, a motor should be used to drive this system with a very high speed and accurate position.

#### **2.4.1.2 Motor Selection**

Choosing the type of motor used was a very important decision. This decision would affect the speed and the accuracy of the whole system. Normal

DC or AC motor was the first choice as it has the lowest cost, but its main problem that position control cannot be achieved by these motors. As a result of that we moved to the stepper motor which can move for a specific number of steps, predefined before. However, it is an open loop system, where we can't have a feedback about the position of the motor [24].

Thus we moved to the servo motor where an encoder is placed on the motor to provide the accurate position in real time. Another decision needed to be taken where to use AC Servo motor or DC servo motor. It was found the AC servo motor is much better due to the following reasons:

- Does not need external power supply to provide DC voltage
- Less maintenance due to absence of brushes
- Compared to DC servo motor it is relatively stable and smooth operation

After choosing the use of AC servo motor, we chose to use a motor from delta electronics, as it is a well-known respectable company with a very good community and all its products are well documented so it is very easy to debug any problem.

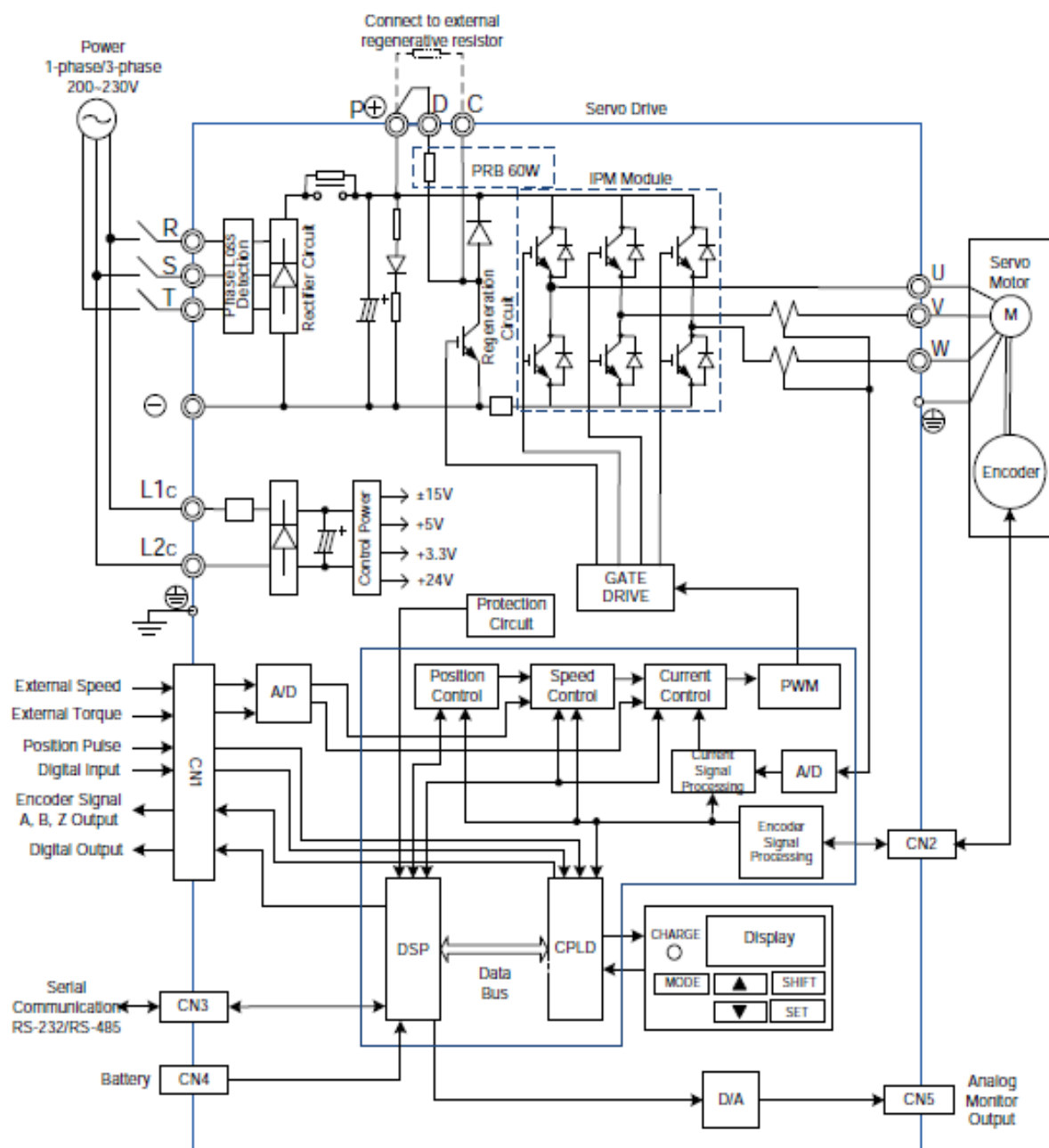
#### **2.4.1.3 Theory of operation of the driver**

The servo mechanism used has a motor driver; the schematic of this driver is shown in Figure 2-17 .The input power is fed to the driver whether 3 phases or 1 phase through the terminals R, S, T. This AC power is rectified to be DC power, then it is turned again to AC using a 3 phase Inverter after passing by a chopper circuit to adjust the voltage magnitude. This controlled AC power is the feed from the motor. Also a regenerative resistor with a regenerative circuit is added to provide a path for the current in case of braking [25].

Two current sensors are placed on two of the motor phases to measure the current; also an encoder is coupled with the motor to measure the exact position of the motor. The signal of the current for the two sensors and the encoder are fed to the control circuit which controls all the motor parameters [25].



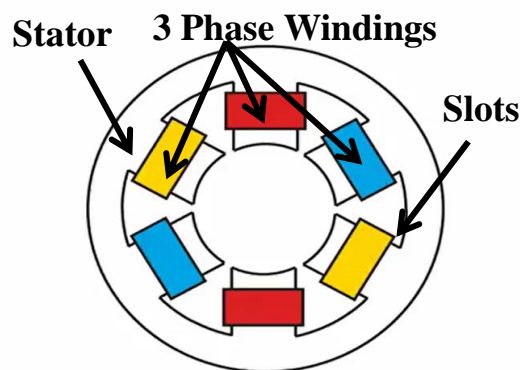
All input signals from the external controller is fed through terminal, these signals as well as the signal from the encoder are processed in the Digital Signal Processor (DSP) and the Complex Programmable Logic Device (CPLD). After that these controllers control the position control loop and the speed control loop, which initiates the current control loop. The current control loop compares the needed current with the actual current measured and provides the suitable PWM signals. Finally, this signal is fed to the gate drive circuit, which controls the gate of the inverter circuit. This controls the exact actuation of the motor [25].



**Figure 2-17: Schematic of the motor driver**

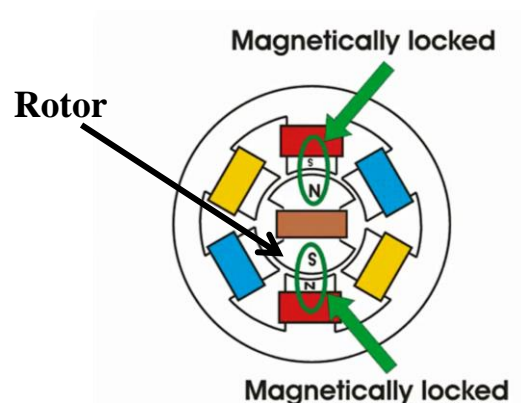
#### 2.4.1.4 Theory of operation of the motor

The Permanent Magnet Synchronous Motor (PMSM) is a mix between the AC Synchronous motor and the DC brushless Motor, where its stator is similar to the AC Synchronous motor and its rotor is similar to the DC brushless Motor. The stator is made of laminations of silicon steel to reduce hysteresis and eddy current Losses. The Inner part of the stator is provided with number of slots. 3 phase windings are distributed similarly among these slots, as shown Figure 2-18, such that when a 3 phase power is supplied a rotating magnetic field is produced. This magnetic field rotates at synchronous speed [26].



**Figure 2-18: The Stator of Permanent Magnet Synchronous Motor**

The rotor is made of permanent magnet. The poles of this magnet are magnetically locked with the rotating magnetic field as shown in Figure 2-19. Ergo; the rotor will rotate at same speed of the rotating magnetic field [26].



**Figure 2-19: The Rotor of Permanent Magnet Synchronous Motor**

### **2.4.2 Linear Actuator**

The ping pong table dimensions are 274 x 152.5 cm; the dimension of the width is the most important (152.2), because the robotic arm should deal with the ball which is coming from the other side of the table, as the robotic arm reach is 32 cm only, so we need to move the robotic arm along the table width with high accuracy, speed, efficiency and the response time should be highly considered to our calculations.

The challenge of our mechanical automation system is selecting an appropriate actuator for a desired functionality designed to move the robotic arm into precise positions.

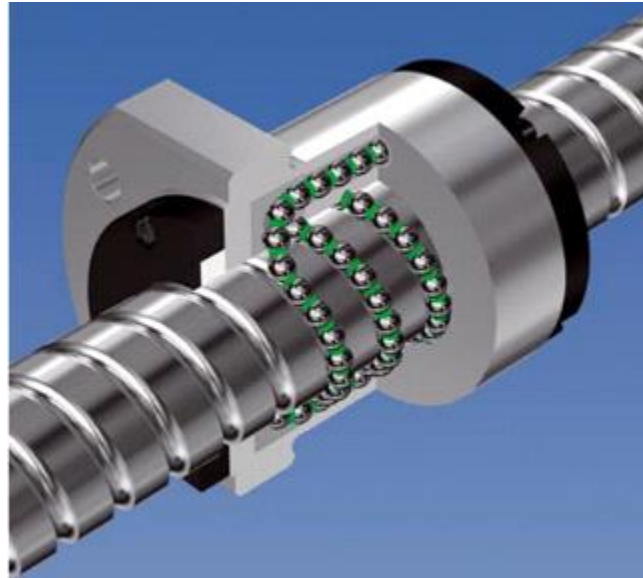
Among many linear actuators options, we should select and design the right one; so, we selected the ball screw option with supporting units, as this system is perfect for us in different aspects such as: cost-wise and efficiency.

### **2.4.3 Strengths and Limitations:**

The ball screw unit closely resembles a rolling ball bearing system and thus is capable of carrying higher loads and achieving a higher thrust force. For this reason, ball screw driven actuators are ideal for our project where the positioning of large, heavy loads may be necessary to a high level of precision. Periodic lubrication of the ball screw may be required depending on the specific actuator design.

#### **2.4.4 What is a Ball Screw?**

A ball screw, like all screws, is a rod with a helical groove that translates rotational motion into linear motion.



**Figure 2-20 Ball Screw Mechanism**

#### **2.4.5 Why use a Ball Screw?**

Common screws slide on their threads and pull the connecting piece with them. This sliding motion is easy to be produced, but extremely inefficient; around 40% of the input power is transferred to the work piece. In order to improve this idea, the sliding contact must be replaced with rolling elements. Ball Screws use ball bearings in the nuts to roll along the screw. The balls must be recirculated or they will fall out of the nut. This leads to the current designs for recirculating ball screws that have efficiencies greater than 90%.

#### **2.4.6 Ball Screw Accuracy:**

Accuracy is one of the crucial components in any ball screw operation, as ball screws are utilized primarily for their precision and accuracy retention. Lead error and mounting error are two calculated error parameters that are important to consider in our ball screw application.

### 2.4.6.1.1 Lead Error:

Lead Error is defined by two ways: the difference between the expected length ( $L$ ) and the actual length ( $L \pm e$ ), and the sum of all the  $e$ 's ( $\sum e = E$ ). For example, if the  $L$  is 5mm and the  $e = \pm 0.005$ mm the possible length could be 4.995-5.005mm.

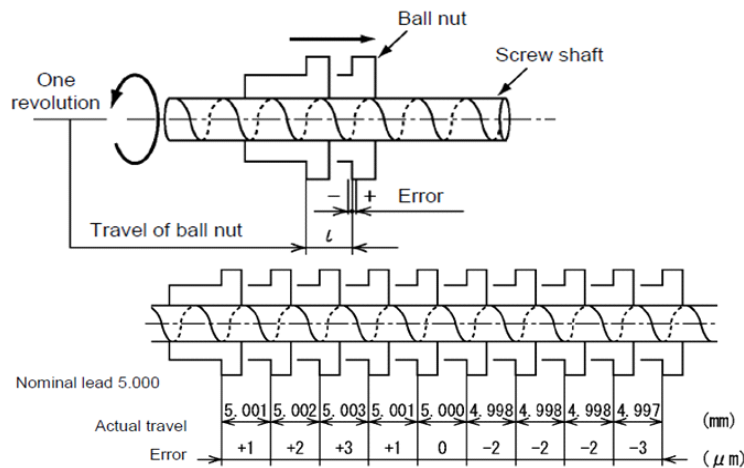


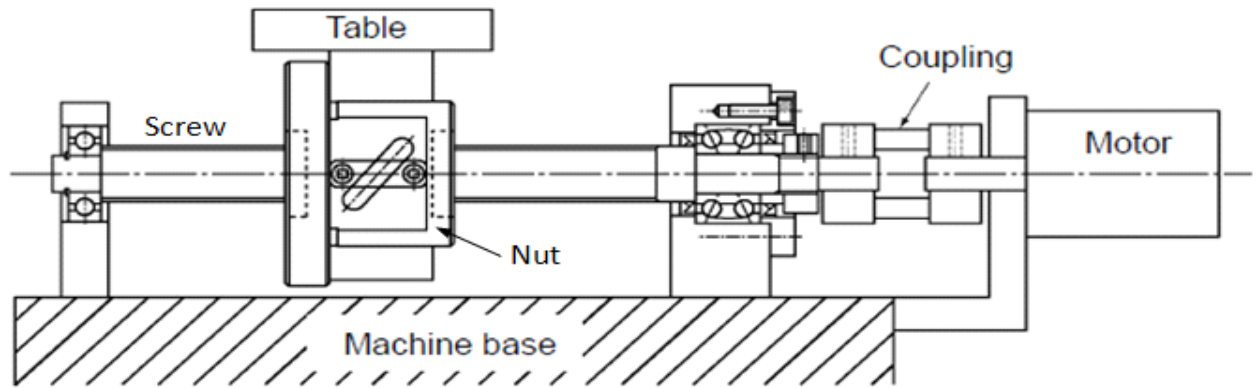
Figure 2-21 Lead Error

### 2.4.6.1.2 Mounting Error:

Mounting Error is caused by the way in which the screw is supported and the installation of the ball nut. The three most likely mounting errors include: bearing misalignment, coupling misalignment and nut misalignment. Mounting error can create noise, shorten life and cause positioning errors.

### 2.4.7 Ball Screw Assembly:

A ball screw must be supported so that the shaft can be rotated. Usually this involves roller bearings and the holding brackets. Some longer screws need to be supported for critical speed and bending restraints. End machining may be required to fit into the bearing or connect to the motor. Connecting to the motor requires a coupling and the appropriate machining. In order for the nut to translate accurately, it must be restrained by a table and support rails. In the end, a ball screw assembly should look something like the picture below.



**Figure 2-22 Screw Assembly**

#### **2.4.8 Lifetime of a Ball Screw:**

Ball screws have very low friction coefficients, which is the key to both their accuracy and their longevity. The actual life is determined by load, speed, lead and screw size. Common values are 20,000 hours, and 50,000 km or 5 years.

#### **2.4.9 Ball screw speed:**

It is necessary to check if the ball screw rotation speed is lower than the critical lap speed, 80% or less of the critical lap speed are recommended as rotation speed, the allowable speed is defined according to the method of supporting the ball screw, So the more we supported the system the higher speed we can have.

In conclusion, ball screw driven actuators are the better choice to our application where high loading and/or thrust forces are required, along with highly accurate positioning and acceptable speeds.

## 2.4.10 Our Ball Screw specs:

Ball Screw 1605 – DVF model

Nut Type: F-Type

Material: Chromo Steel – GCR 15

Length: 1500 mm.

Ball Screw Diameter: 16 mm.

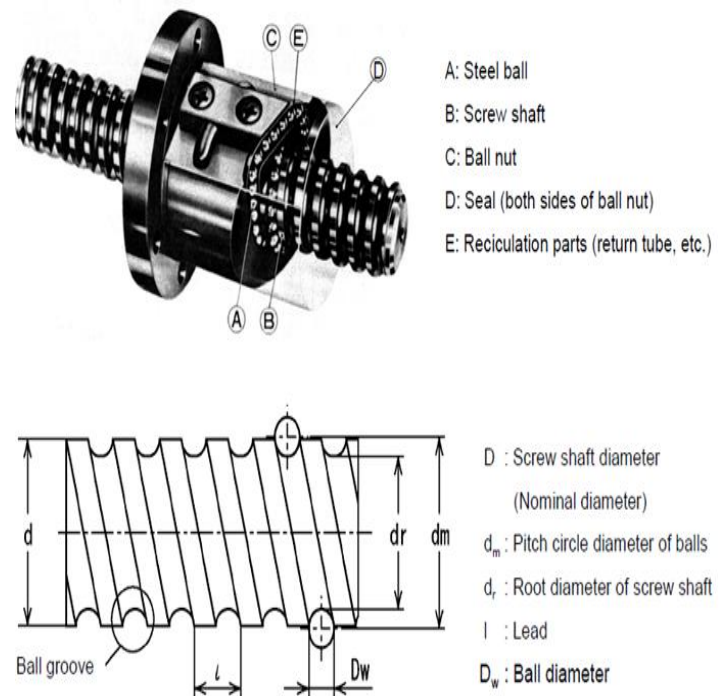
Ball Turns: 3

Lead: 5 mm.

Ball Diameter: 3.175 mm.

Basic Dynamic Load: 7.65 KN.

Basic Static Load: 13.2 KN.



**Figure 2-23 Screw Specs**

### **3 PROPOSED PROTOTYPE**

#### **3.1 DESCRIPTION**

One of the makers of good design is smart prototyping. In order to properly test and validate the feasibility of our design we decided to design a working several prototypes that –realistically and cheaply- mimics everything in our final system. Several prototypes were made along the project to test single parts of the system. This is the final prototype which mimics the actual system, with all its components and algorithms. This prototype consists of four main subsystems:

- Image Processing and Computer Vision
- Path Prediction
- Robotic Arm
- Servo Mechanism

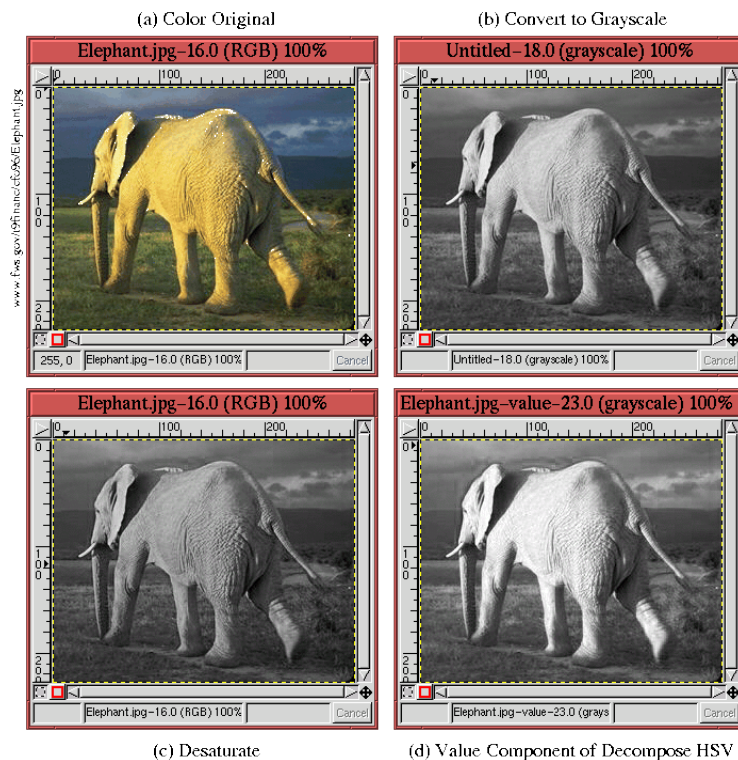
#### **3.2 IMAGE PROCESSING AND COMPUTER VISION**

##### **3.2.1 Improvements on Image processing:**

All our previous trials were colour based, as we have converted between RGB colour space into HSV colour space and included all pixels within a specific HSV value range. This approach although it is straightforward and heavily documented through OpenCV documentation, but this option has a large consuming drawback. The first drawback is the obligation of using chromatic camera (Hardware limitation) and the second drawback is the computation time needed for doing the image processing in the HSV colour space (Software limitation ).



The choice of a monochromatic hardware and software scheme is more efficient in terms of the cameras used, the cameras could be monochromatic, Also the computation cost will be reduced a lot when using monochromatic images.

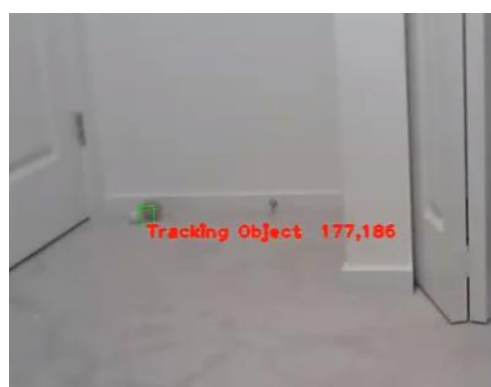


**Figure 3-1: HSV, RGB and gray scale color spaces**

So, we in this section we will begin to explain the improvement introduced into the system and how efficient was it.

### 3.2.2 Real-Time Object Tracking Without Colour

The technique is heavily tutored by Kyle Hounslow who works as Computer Vision Engineer and part time educator specializing in deep learning and traditional computer vision.



**Figure 3-2: Motion detection trial**

**The algorithm used is composed of two main steps:**

1. The first step is to read two sequential frames from the camera feed, this feed could be chromatic or monochromatic.
2. The second step is to compare these two sequential frames, once this comparison is made pixels which have changed between the two frames is determined

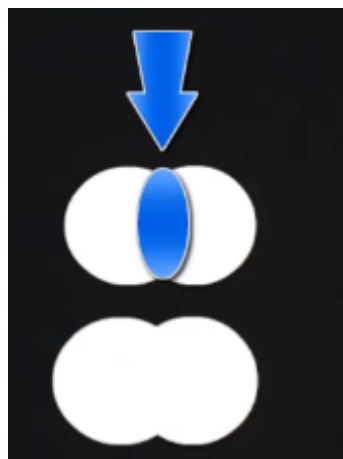
The algorithm can be also identified as Motion detection algorithm. Let's investigate how this algorithm performs. Consider our first frame of the ball, in this frame the image is converted to a thresholded version of the image, this is called frame 1.

The second frame is sequentially acquired and the absolute difference is computed between the two frames, once this difference is computed then we acquire an image that describes the position of the changed pixels.



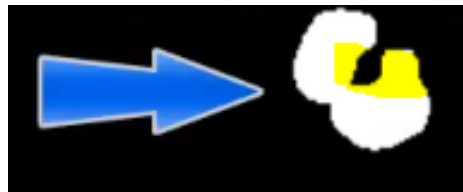
**Figure 3-3: Normal and Thresholded Image**

The blue arrow shown in Figure 3-4 shows the common point between the two frames. So when we compute the difference the parts in common will be excluded and parts not in common will be included in the new absolute difference image, an experiment can easily proof this.



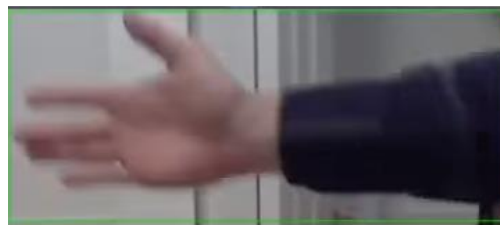
**Figure 3-4: Intersection between Frames**

In this experiment, the ball is thrown very fast and the black dot in middle of the two white spots can be easily identified, Highlighted in yellow.



**Figure 3-5: Intersection highlighted**

So, this improvement in the image processing scheme gave us a great opportunity to reduce both hardware and computation overheads, But I have a great drawback since the algorithm used is naive enough to detect all moving objects as the hands of the player.



**Figure 3-6: Hand can be detected in motion detection**

### **3.2.3 Improvements on Computer vision:**

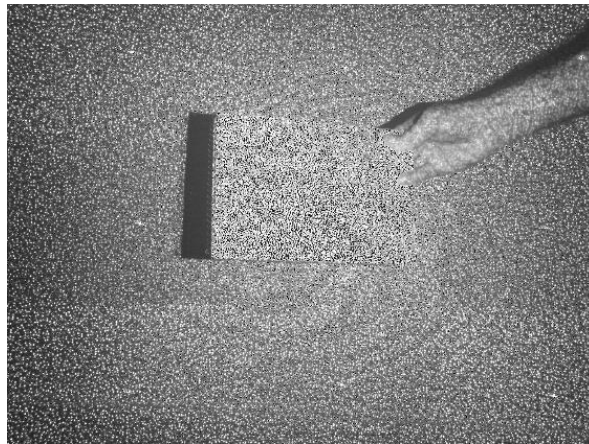
Kinect v 1.0 (Xbox 360 version)



**Figure 3-7: Kinect Structure**

The technology behind The Kinect is described in a patent licensed for Prime Sense as:

Apparatus for mapping an object includes an illumination assembly which includes a single transparency containing a fixed pattern of spots. A light source trans illuminates the single transparency with optical radiation to project the pattern onto the object. An image capture assembly captures an image of the pattern that is projected onto the object using the single transparency. A processor processes the image captured by the image capture assembly to reconstruct a three-dimensional (3D) map of the object - Google Patents.



**Figure 3-8: Grains of Kinect IR Grid**

The technology patented to Prime sense is called Structured light, Kinect seems to have three eyes: the two in its center and one off all the way to one side. That “third eye” is the secret to how the Kinect works. Like most robot “eyes,” the two protuberances at the center of the Kinect are cameras, but the Kinect’s third eye is an infrared projector. Infrared light has a wavelength that’s longer than that of visible light so we cannot see it with the naked eye. Infrared is perfectly harmless—we’re constantly exposed to it every day in the form of sunlight.

The Kinect’s infrared projector shines a grid of infrared dots over everything in front of it. These dots are normally invisible to us, but it is possible to capture a picture of them using an IR camera. We can show an example of what the dots from the Kinect’s projector look like.

This image is captured using the Kinect itself. One of those two cameras are IR cameras. It's a sensor specifically designed to capture infrared light.

### 3.2.4 Kinect technical specifications

#### Specifications for the Kinect

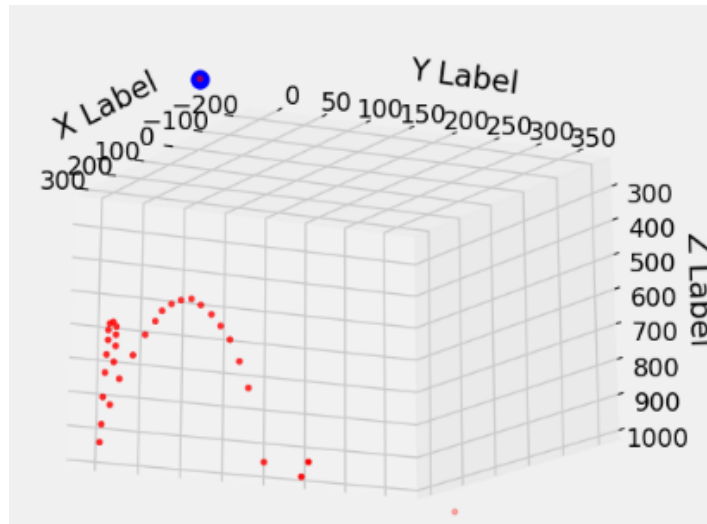
Kinect	Array Specifications
Viewing angle	43° vertical by 57° horizontal field of view
Vertical tilt range	±27°
Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppression
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.

The technical specifications of the Kinect show us a variety of great features, the viewing angle is sufficient for our application, Also the FPS was initially assumed as sufficient. Accelerometer and other audio-related features were not used in our application.

### 3.2.5 Fast moving object detection

The technology the Kinect based on, Structured light, is non-accurate for small objects specially if this object is fast moving object like a ping pong ball, this drawback led to low number of points per each trajectory which directly affect the accuracy of prediction algorithm, several tests were made by our team to determine the number of points per trajectory at different speeds.

PyOpenNI and Matplotlib are both used to communicate and plot Kinect data, Although the setup was tough procedure and required to shift to Ubuntu 16.04 OS instead of Windows, But the once the libraries, drivers and middleware is installed everything works fine.



**Figure 3-9: Kinect Ball Points**

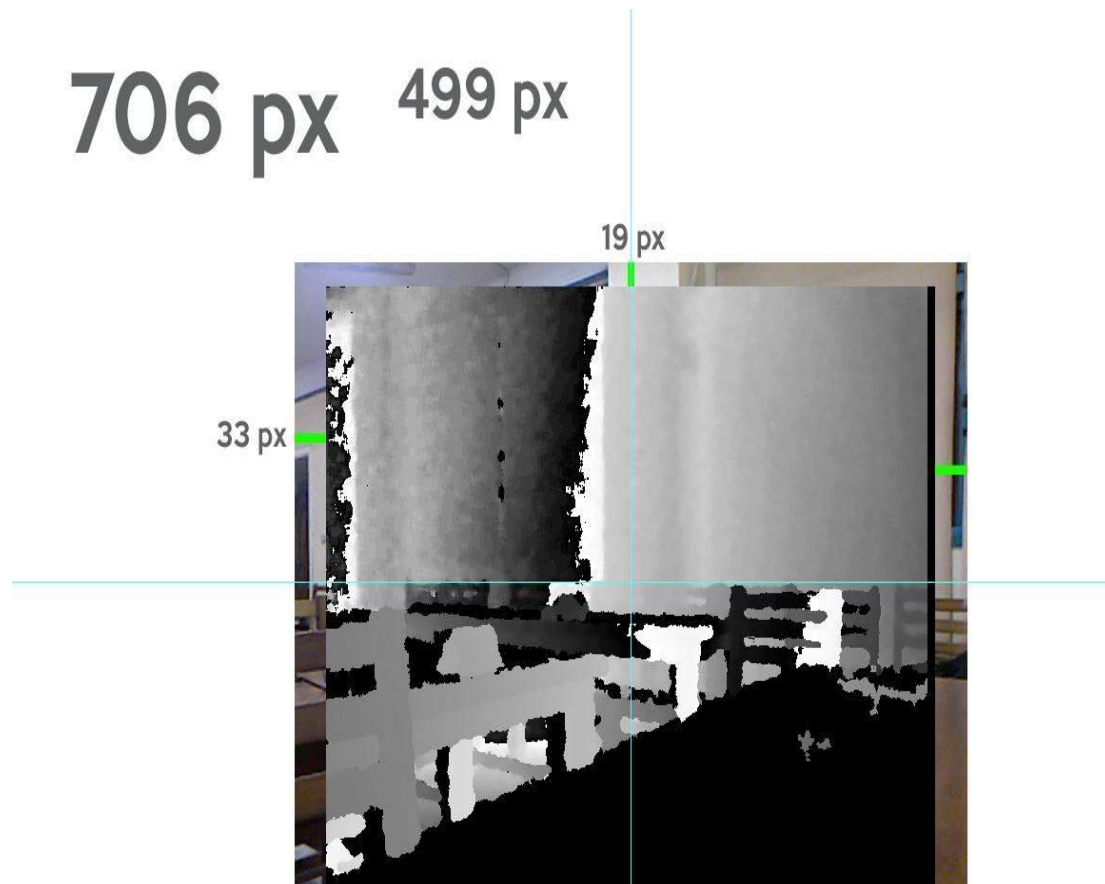
This graph is a simple demonstration for low speed bounce although initially it was thought that these number of points are sufficient, But the prediction algorithm needs more points to be accurate, also at higher speeds the number of points decrease dramatically as the Kinect depth sensing capability fail to recognize the blurry ball at 30 FPS.

### **3.2.6 Non-registered data**

One of the most difficult problems while dealing with Kinect is registration, Registration is simply mapping between RGB coordinates and Depth coordinates, the is a geometrical shift between the RGB and Depth camera in Kinect, this shift can be illustrated in the following figure:



**Figure 3-10: Shift Between RGB and Depth image in Kinect**



**Figure 3-11**

It is obvious here that the (x, y) coordinates in RGB reference frame should be mapped to corresponding depth point, this what is called registration, The OpenNI library has no registration option. The solution was to make the alignment manually by calculating the resize and bias ratio at each axis.

This manual alignment is a naive solution, but it worked for our application, but we must mention that the camera must be stationary, as change in position would need re-alignment.

Apple acquired PrimeSense on November 24, 2013 and OpenNI project was shut down, so there is no further improvements or technical support for the OpenNI, So this lead to poor documentation and non-updated information.

### 3.2.7 ZED camera

#### Technical specifications



**Figure 3-12: Zed Camera**

ZED perceives the world in three dimensions. Using binocular vision and high-resolution sensors, the camera can tell how far objects are around you from 0.5 to 20m at 100 FPS, indoors and outdoors.

The ZED camera has a group of great features :

- **Dual 4MP Camera** Capture stunning 2K 3D video with best-in-class low-light sensitivity to operate in the most challenging environments.
- **High Frame-Rate ZED** is the world's fastest depth camera. Capture 1080p HD video at 30FPS or WVGA at 100FPS and get a crisp image.
- **Wide Field of View** With its 16:9 native sensors and ultra sharp 6 element all glass lenses, you can capture 110° wide-angle video and depth




Video Mode	Frames per second	Output Resolution (side by side)
2.2K	15	4416x1242
1080p	30	3840x1080
720p	60	2560x720
WVGA	100	1344x376

**Figure 3-13 The ZED camera itself has different resolution modes shown above in figure.**



The ZED camera is GPU dependent, it utilizes CUDA for efficient rectification and depth generation, we have used Visual Studio 2013 as our IDE, The ZED SDK install both OpenCV 3.1, CUDA libraries and drivers, So the SDK is straightforward to use. The ZED cam has a user friendly interface.

The image processing program was converted from Python to C++ as the ZED SDK is limited to C++ only, This migration was easy to make with tons of available tutorials online, The PC used to interface with ZED camera had the following features:

Name	NVIDIA GeForce GT 650M	Lookup		 
GPU	GK107	Revision	A2	
Technology	28 nm	Die Size	118 mm <sup>2</sup>	
Release Date	Mar 22, 2012	Transistors	1300M	
Component	Details	Subscore	Base score	
Processor	Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz	7.5		 Determined by lowest subscore
Memory (RAM)	8.00 GB	7.7		
Graphics	Intel(R) HD Graphics 4000	7.0		
Gaming graphics	1696 MB Total available graphics memory	7.0		
Primary hard disk	6GB Free (150GB Total)	7.7		
Windows 7 Ultimate				

**Figure 3-14: Speciation of The Laptop Used**

### 3.2.8 Registered data

The ZED SDK present both intrinsic and extrinsic parameters at 640p, Using these parameters we can get depth information easily of the ball, So the registered data is easily accessed thanks to the SDK.

### 3.2.11 Socket Communication

Socket communication is used in our system, The idea is useful as it apply efficient parallel processing for distributed processing on multiple cores of the computer, This cause high efficient processing for the whole program

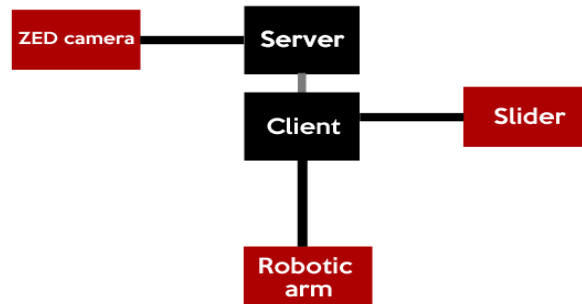


Figure 3-15: Socket Communication used

The client and Server execute simultaneously once there is a connection request the connection occur between the two sides and data interchange occur.

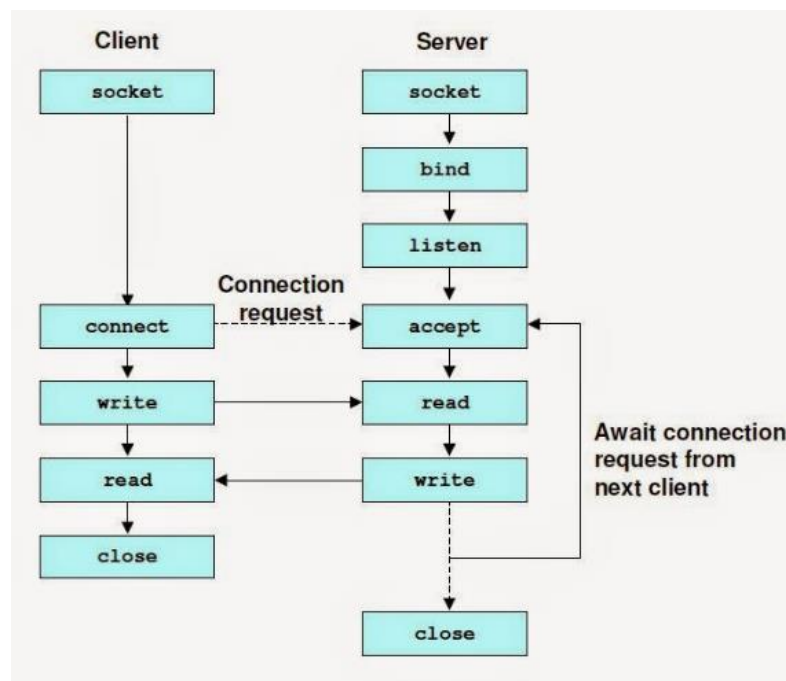


Figure 3-16: Socket Communication Topology

### 3.3 PATH PREDICTION

Having successfully built a descriptive mathematical model for our system, the set of differential equations needed to be converted into a computational algorithm in order to be utilized in our system.

The algorithm was implemented in MATLAB as a first step and then converted to C++.

MATLAB was chosen due to its strength in dealing with equations, vectors and matrices. Moreover, MATLAB facilitated the graphical representation of the output which allowed us to readily view whether or not the predicted paths converged onto the actual one. This made testing and debugging extremely easy. After all errors and pitfalls were discovered and properly remedied, the algorithm was converted to C++.

By using a linear algebra C++ library called Eigen, the path prediction code performed much faster than it did using MATLAB and was seamlessly integrated with the entire project's code.

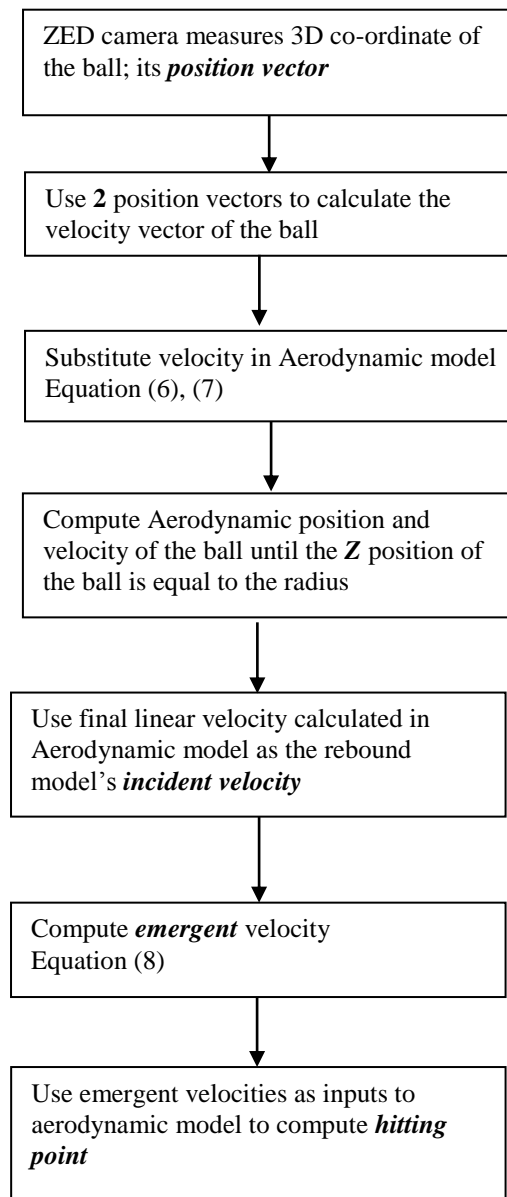
The algorithm takes as an input a set of 3 dimensional position vectors that contain the x,y,z coordinates of the ball and represent its location in 3D space relative to the camera. It then uses the aerodynamic and rebound differential equations to compute a set of predicted paths. The closest point within our robot's workspace at the end of each path is the "hitting point" which is sent to the actuation algorithm.

Consequently, our prediction algorithm can be split into 4 main parts:

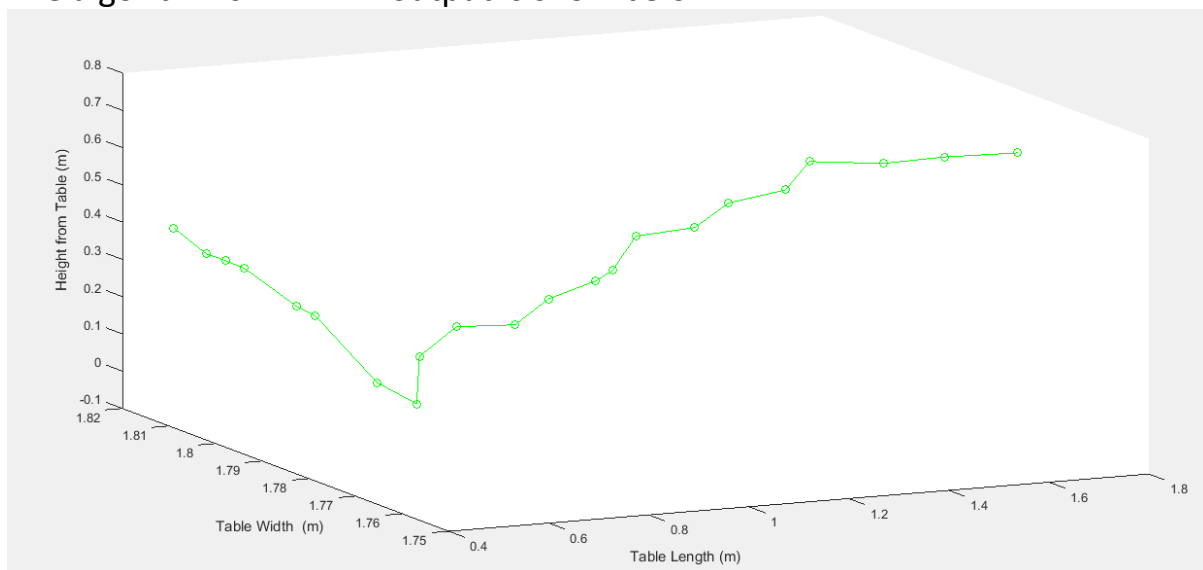
1. Measurement stage, where the ZED camera measures the 3D position of the ball and inputs it into the algorithm
2. Aerodynamic model, to predict where the ball will hit the table
3. Rebound model, to predict the change in velocity –and subsequent change in position- of the ball when it hits the table
4. Aerodynamic model, to predict the "hitting point"; the final position of the ball, which will be the position at which our robot will move to, in order to successfully hit the ball back.

### 3.3.1 Flowchart

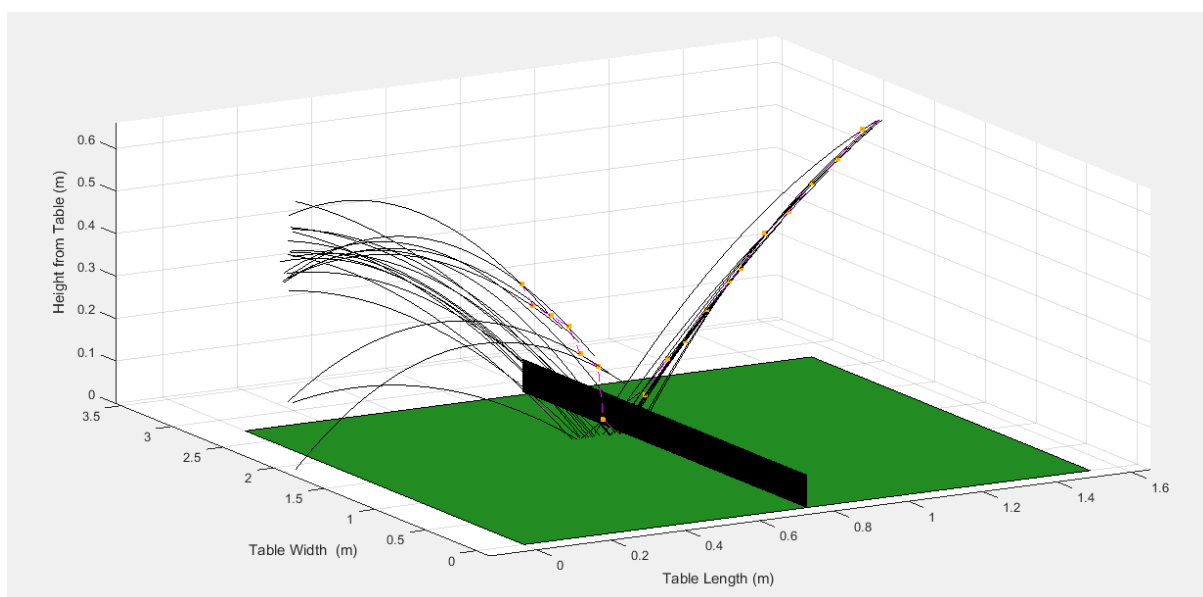
A flowchart of the code details the process:



The algorithm's MATLAB output is show below:



**Figure 3-17: Predicted Paths on MATLAB**

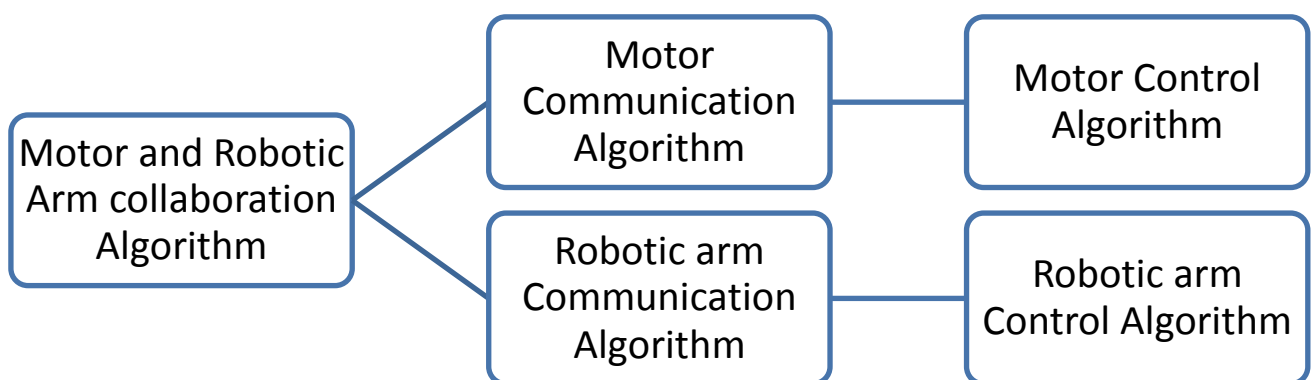


**Figure 3-18: Predicted Paths on MATLAB**

Figure 3-17 and Figure 3-18 outline the Path Prediction algorithm's output on MATLAB. Figure 3-17 is the ball's actual path on the table. A set of 3 dimensional vectors, each vector has 3 points that represent the x, y, z coordinates of the ball in 3D space. Data from the ZED camera is unfiltered and is noticeably different from the ball's parabolic path. Figure 3-18 depicts the path of the ball on the table, as well as all the predictions that are generated every two points. While the "hitting points" at the end of each path are close to the actual value, they vary greatly from one path to the next and do not provide a clear point for the actuation system to navigate to.

### 3.4 ROBOTIC ARM

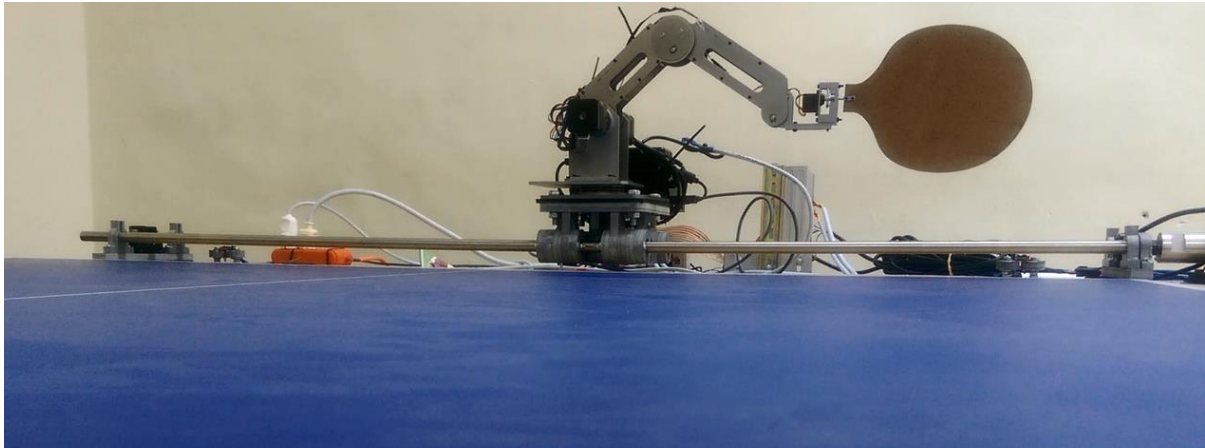
The Robotic arm as well as the servo mechanism forms the Actuation part of this system. The actuation part receives the coordinates of the ball in order to be able to hit it. Thus the algorithms which control the actuation parts are very crucial in order to achieve the system goal. The actuation part consists of five algorithms as shown in Figure 3-19. They deal with the arm and the servo mechanism in order to hit the ball successfully.



**Figure 3-19: layout of Actuation Algorithms**

#### 3.4.1 Instalment and Connection

The robotic arm is installed on a carriage moving along the linear screw driven by the motor. The control box of the robotic arm is installed behind the arm on the carriage as shown in Figure 3-20. The control box contains a PCB consists of an Arduino Mega Controller, 3 stepper motor drivers and a Bluetooth module. The 3 motor motors and the servo are connected to the control box to receive power and signal. The control Box in connected to the PC via USB cable, in addition to power cable for power feeding.



**Figure 3-20: The robotic arm is installed on a carriage**

### **3.4.2 Motor and Robotic Arm collaboration Algorithm**

Motor and Robotic Arm Collaboration Algorithm is the First algorithm used in Actuation. It is installed on the PC. It aims to receive the required position from the Path Prediction Algorithm then divide the required position among the Robotic Arm and the Motor. This division is based on the capabilities of each instrument. After that it sends this position to each of the Robotic Arm Communication Algorithm and the Motor Algorithm installed on the PC. It receives the required position from the Path Prediction Algorithm using socket programming which ensures high speed communication.

### **3.4.3 Robotic Arm Communication Algorithm**

Robotic Arm Communication Algorithm is the Second algorithm used in Actuation. It is installed on the PC. It aims to receive the required position after decoding it from Motor and Robotic Arm collaboration Algorithm and send it to the Robotic Arm Control Algorithm installed on the Arduino. It received the required position from Motor and Robotic Arm collaboration Algorithm using socket programming which ensures high speed communication.

Then the algorithm applies safety limit to make sure that the Robotic Arm doesn't receive and order out of the range of the linear actuator. At the same time a timer interrupt is working in the background every 100 ms. It uses serial communication to send the position to the Robotic Arm Control Algorithm.

The position is sent using the Standard Dobot Packet form, using IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754).

### **3.4.3.1 Standard Dobot Packet form**

Data packet with a fixed length is adapted. The packet consists of 42 bytes, one byte for the packet head and one byte for the packet end; and rest 40 byte is valid data. Data is packed with little ending byte order.

Each packet consists of 42(2+4×10) bytes, including packet head 0xA5, packet end 0x5A and 10 float parameters. Each float parameter is structured with 4 bytes data (little ending byte order). The first float indicates the mode of operation. The do bot has many modes of operation but the most suitable one for our application is the target moving mode (mode 3). The float 2, 3 and 4 are used to enter the target coordinate X, Y and Z respectively. Float 5 is used to adjust the rotation angle of the end effector. Float 8 is used to adjust the moving style to be one of the following:

- JUMP: The end effector will lift High and move horizontally to point B and then move down High to reach point B. The height of lift can be configured
- MOVJ : Each joint will run smoothly from initial angle to its target angle, regardless the trajectory. The requirement is for each joint, the running time equals, so that each joint will start and finish the movement at the same time.
- MOVL : The joints will cooperate in order to perform a line trajectory from point A to point B.

The most suitable mode for our application was MOVL. Float 9 indicates the final speed when Dobot reach its target point while float 10 indicates the Pause time after the action in sec. [27]



### **3.4.3.2 IEEE Standard for Binary Floating-Point Arithmetic**

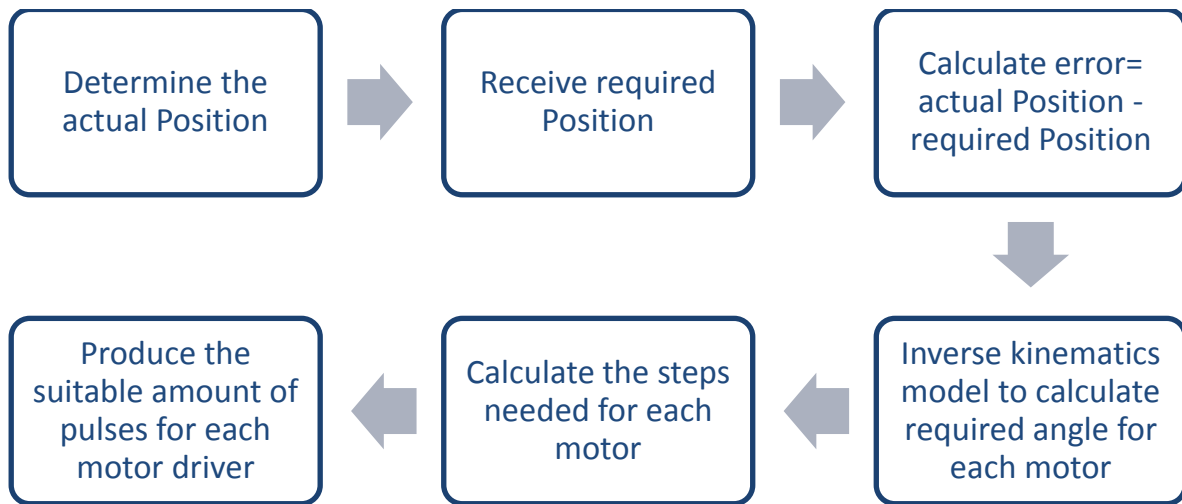
The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and portably. Many hardware floating point units now use the IEEE 754 standard [28]. There are two types of standard:

- The IEEE 754 single precision format is 32 bits, which are (from left to right):
  - One bit for the sign
  - 8 bits for the exponent
  - 23 bits for the mantissa
- The IEEE 754 double precision format is 64 bits, which are (from left to right):
  - One bit for the sign
  - 11 bits for the exponent
  - 52 bits for the mantissa

In our robot The IEEE 754 single precision format is 32 bits is used

### **3.4.4 Robotic Arm Control Algorithm**

Robotic Arm Control Algorithm is the Second algorithm used in Actuation. It is installed on an Arduino Mega micro controller placed inside the control box of the DoBot. It aims to generate the required signal in order to drive the 3 stepper motors and the servo motor to move the end effector to the desired position. The Algorithm receives the reference value form the Robotic Arm Communication Algorithm, and then moves the motor with the required speed until reaching the required position accurately. The flow chart of the system is shown in Figure 3-21.



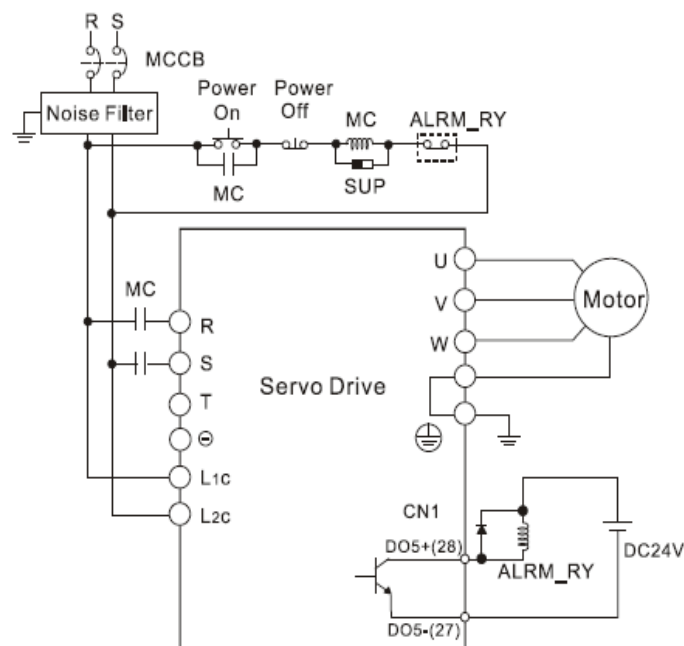
**Figure 3-21: Flow Chart of Robotic Arm Control Algorithm**

## 3.5 SERVO MECAHNISM

### 3.5.1 Servo Motor

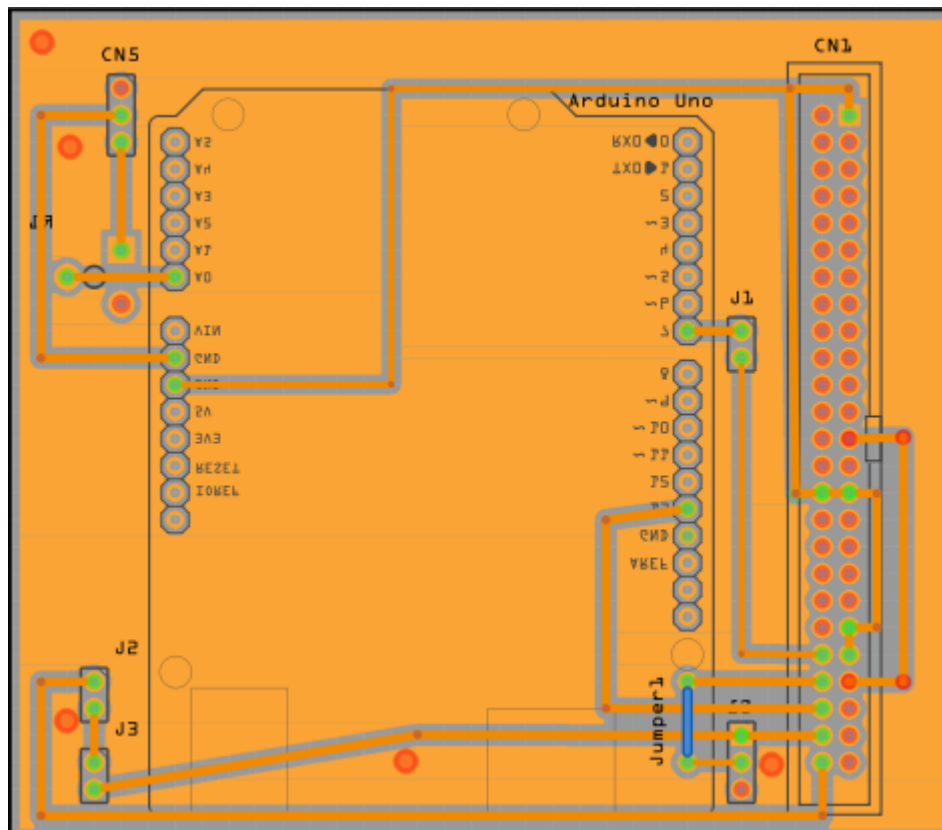
#### 3.5.1.1 System Connection

The motor and the driver are connected as shown in Figure 3-22, where a circuit breaker is attached in the main power circuit feeding current to the contactor which will in turn, feed power to the driver. The proper sizing of the CB and the cables were made according from the motor and the driver manufacturer. Moreover, a start button latched with one of the contacts of the contactor and stop button are attached to the control circuit to energize the coil of the contactor.



**Figure 3-22: Motor and Driver Connection Diagram**

PCB Board was also designed and manufactured to connect the all driver terminals with the Arduino Uno. The details of the board are shown in Figure 3-23. The PCB contains Arduino Uno, 44 pin connector (CN1) which sends and received all the required signals form and to the motor driver. It contains 3 pins connector (CN5) which received an analog voltage equivalent to the position of the motor to act as a feedback for motor position. Also two limits switch are connected via (J2) and (J3) to act as a safety precaution to stop the motor if it reached one of the ends.



**Figure 3-23 Motor PCB Board**

### **3.5.1.2 Motor Control**

#### **3.5.1.2.1 Motor Communication Algorithm**

Motor Communication Algorithm is the fourth algorithm used in Actuation. It is installed on the PC. It aims to receive the required position after decoding it from Motor and Robotic Arm collaboration Algorithm and send it to the Motor Control Algorithm installed on the Arduino. It received the required

position from Motor and Robotic Arm collaboration Algorithm using socket programming which ensures high speed communication.

Then the algorithm applies safety limit to make sure that the motor doesn't receive and order out of the range of the linear actuator. At the same time a timer interrupt is working in the background every 250 ms. It uses serial communication to send the position to the Motor Control Algorithm. Moreover a manual control option can be selected at the beginning of the code to adjust the position manually to the origin position.

### **3.5.1.2.2 Adjusting Mode of operation**

The servo mechanism provided including the driver and the motor can only operate in 3 modes <sup>[23]</sup>:

#### **3.5.1.2.2.1 Position control mode**

In this mode a reference position is given as an input to the driver and the motor keeps working till it reaches this position. This mode is the most suitable for our application, as we always need the motor to drive the slider to a defined position.

##### **3.5.1.2.2.1.1 LEVEL OF PULSES VOLTAGES**

There are two levels of pulse voltages:

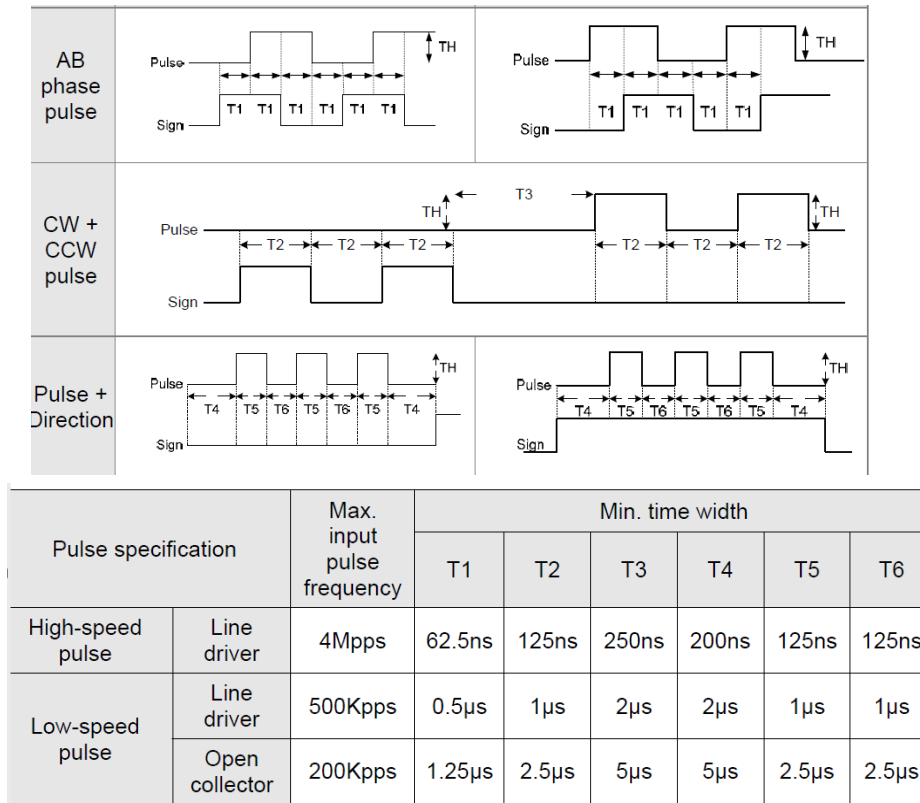
- Line driver input, it requires 5V power supply. This kind of pulse is used usually with Microcontrollers, it has two types.
  - High Frequency (Max: 4Mpps)
  - Low Frequency (Max: Kpps)
- Open-collector input, it requires 24V power supply and Max input pulse frequency: 200kpps. This kind of pulse is used usually with PLC

##### **3.5.1.2.2.1.2 ELECTRONIC GEAR RATIO**

It is a parameter in the driver, which indicates the number of input pulses which makes the motor have one complete turn. Adjusting this parameter is a trade-off between resolution and controller capability, as the speed of the motor is directly proportional to the frequency of the pulses.

### 3.5.1.2.2.1.3 TYPES OF PULSES

There are three types of pulses as shown in Figure 3-24. The most suitable type of them is (pulse + direction) type, as it is less complicated. Moreover, the pulses with variable frequency are generated only on one pin and this reduces a lot of computation effort from the controller.



**Figure 3-24: Type of pulses**

### 3.5.1.2.2.2 Speed control mode

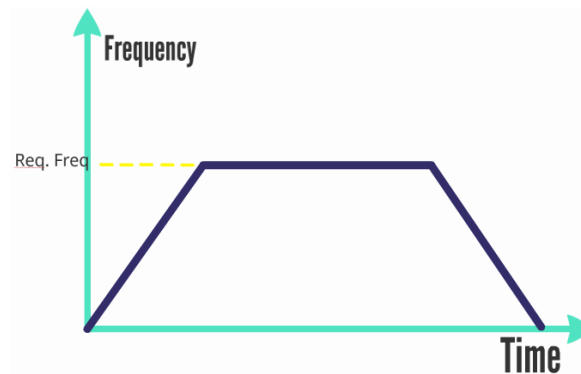
In this mode a reference speed is given as an input to the driver and the motor keeps running at this speed.

### 3.5.1.2.2.3 Torque control mode

In this mode a reference torque is given as an input to the driver and the motor keeps producing this torque.

### 3.5.1.2.3 Required Signal

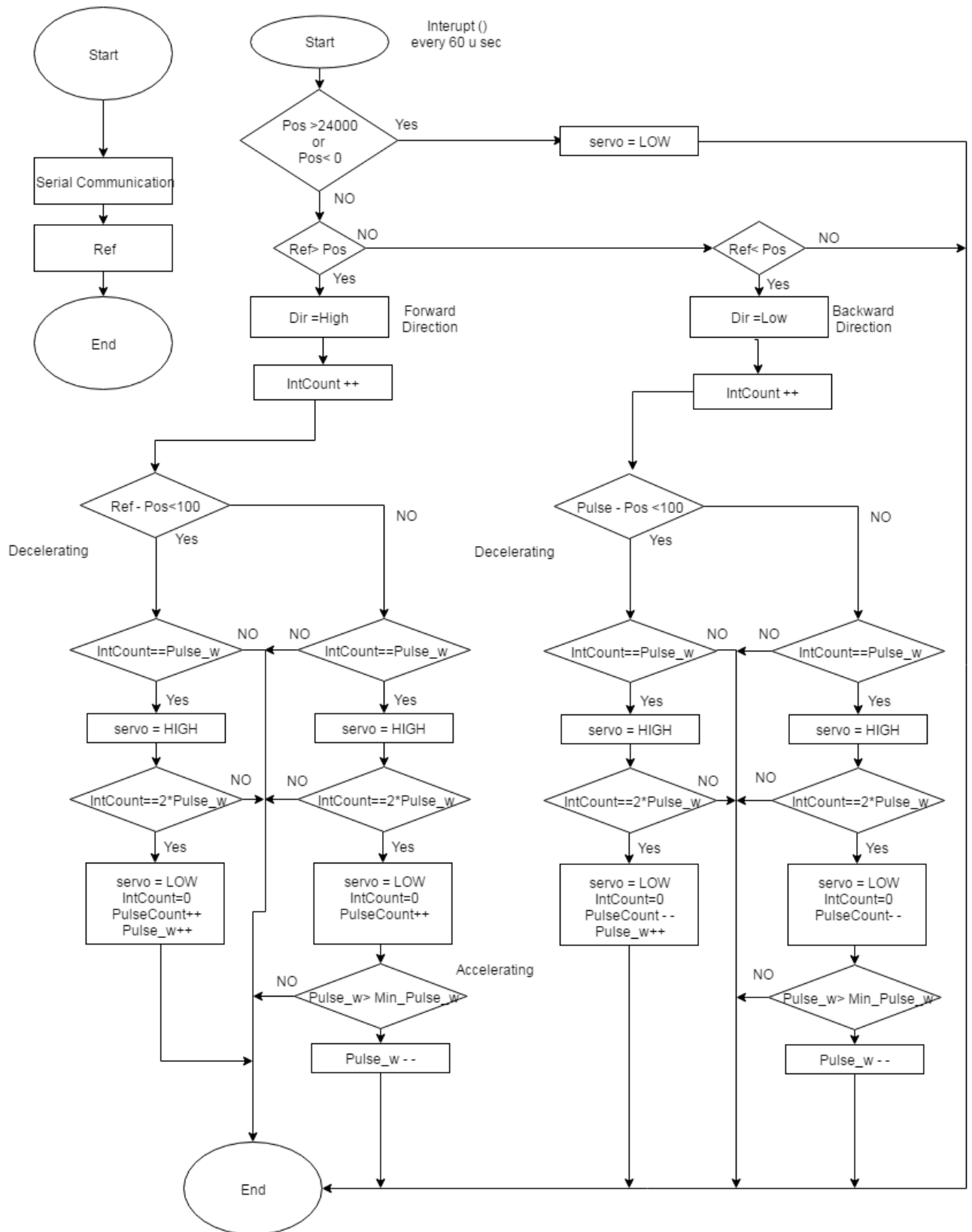
In order to get the most benefit of the motor capabilities, and to provide a very high response system, the motor was adjusted to rotate at 4800 rpm. This speed will suit the application and it is within the rating of the motor which is 5000 rpm. It was found that to operate the motor on this speed the frequency of the generated signal should be 8.33 KHz, as shown in Figure 3-25 .



**Figure 3-25: Required Signal**

### 3.5.1.2.4 Motor Control Algorithm

Motor control Algorithm is the fifth algorithm used in Actuation. It is installed on an Arduino Uno micro controller. It aims to generate the required signal in order to control the motor. The Algorithm receive the reference value form the main controller, and then move the motor with the required speed until reaching the required position accurately. The Algorithm also provides smooth start and stop for the motor by increasing and decreasing the frequency of the signal gradually. The detailed flow chart of the Algorithm is shown in Figure 3-26



**Figure 3-26: The detailed flow chart of Motor Control Algorithm**

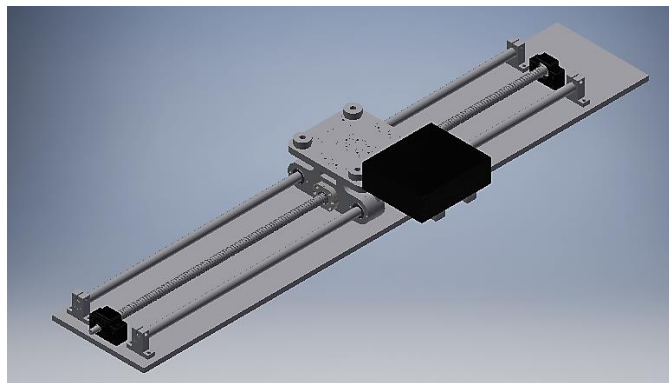
#### **3.5.1.2.5 Electronic Gear Ratio**

The electronic Gear ratio is adjusted such that 200 pulses would make one motor cycle. This was adjusted such that a number less than may result unacceptable error and resolution and a number more than this may be beyond the computation power of the controller.

### **3.5.2 LINEAR ACTUATOR**

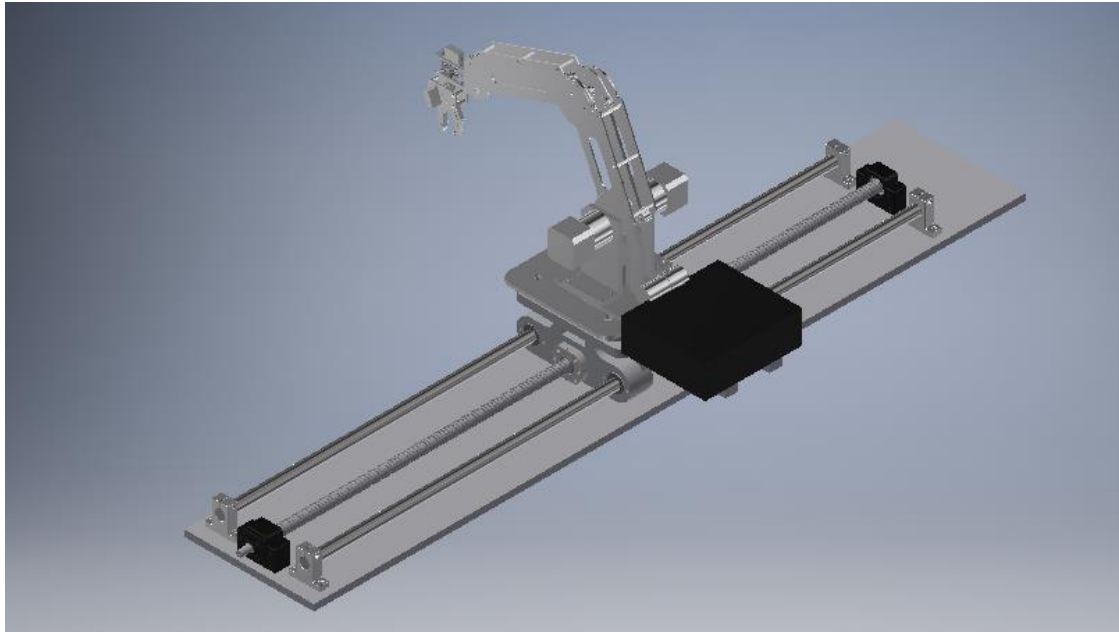
#### **3.5.2.1 Assembly Design:**

We designed the slider assembly in AutoDesk Inventor to ensure proper fitting of the parts as well as generate the necessary files for manufacturing either by a CNC router or to be turned on the lathe. The design features the ball screw in the middle between the two guide rails to offload the loads perpendicular to the screw to the guide rails to reduce friction and thus reduce wear on the ball screw and nut. The ball screw is fixed at both ends with its specialized end supports, one of which is simply a bearing and the other has an additional thrust bearing to bear the pushing and pulling forces on the screw when it moves the sliding block, thus increasing the lifetime of the motor bearings which are more expensive to replace. The linear bearings are fastened in place using retaining plates and bolts and the guiderails are fastened to the base of the assembly with the help of the specialized aluminum supports.



**Figure 3-27 Designed Assembly**

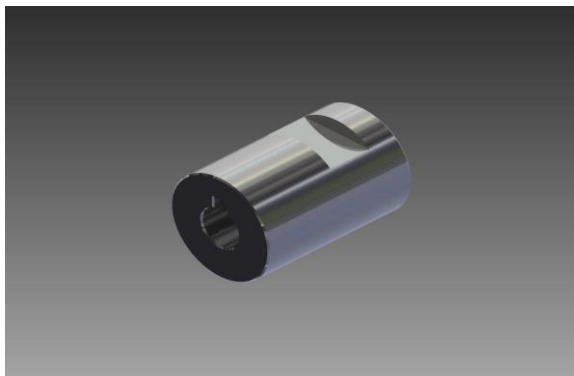




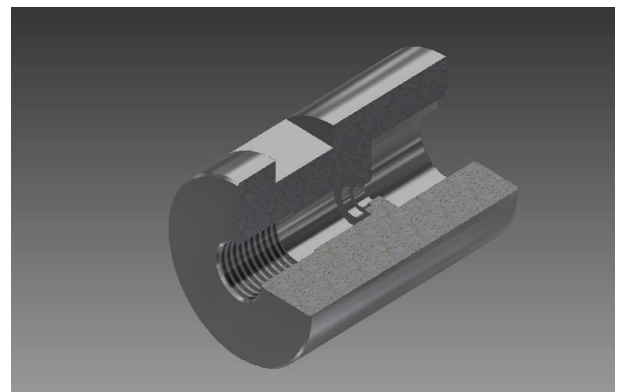
**Figure 3-28 Designed Assembly with Robotic Arm**

#### 3.5.2.2 Coupler Design:

A ball screw must be connected to the servo motor, so, we designed a coupler to connect the motor to the screw.



**Figure 3-30 Outside Coupler design**



**Figure 3-29 Inside Coupler Design**

#### 3.5.2.3 Assembly:

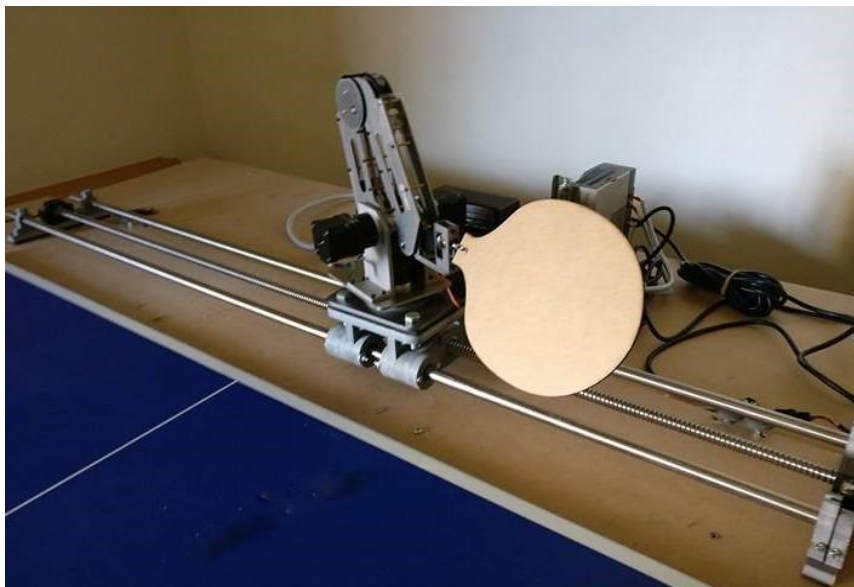
##### 3.5.2.3.2 Motor and Robotic Arm Instalment:

A coupler was designed to attach to both the motor shaft and the ball screw using a combination of a grub screw and a key-slot to attach to the motor side and a combination of a grub screw and threading the coupler on the M10 thread part of the ball screw.



**Figure 3-31 Designed Coupler Connected to Motor**

The Robotic arm and his control box are installed on the screw so the arm could move along the table's width with the required speed, accuracy and efficiency.



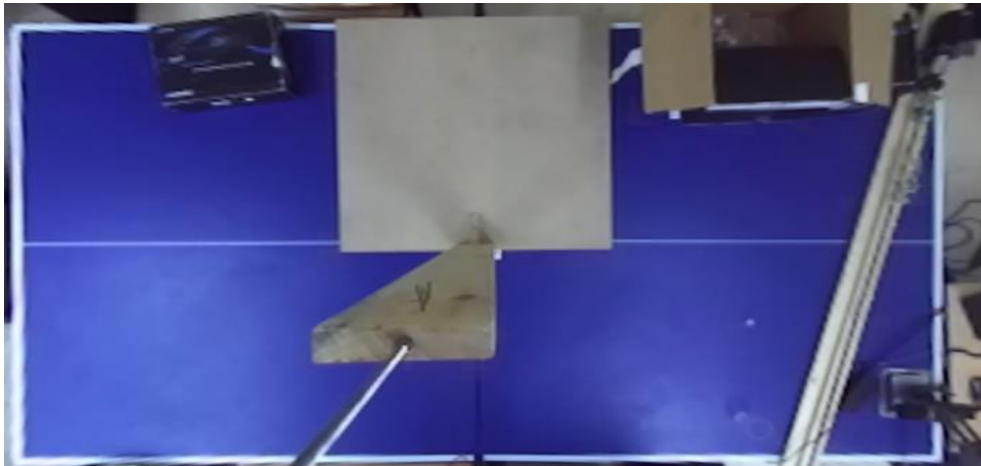
**Figure 3-32 Robotic Arm Instalment on Designed Assembly**

## 4 EXPERIMENTAL WORK AND RESULTS

### 4.1 RESULTS

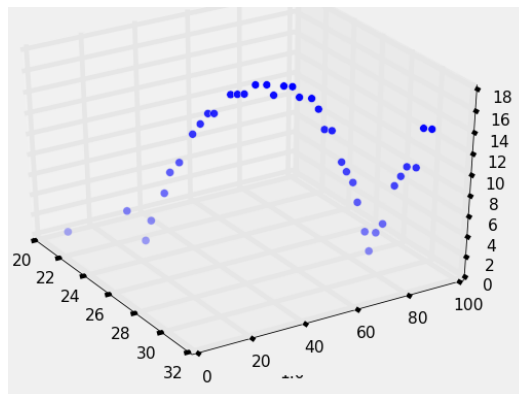
#### 4.1.1 Image Processing and Computer Vision

We have utilized the ZED camera at 640p resolution for maximum FPS, This high FPS is sufficient for detecting fast moving ping pong ball The rectified view of the table shows the ability of the camera to capture all the events that occur on the ping pong table, Several tests were made to check the ability to track the fast moving ball.



**Figure 4-1: Top View of the table**

The plot of the acquired ball position can be shown using Matplotlib, The trajectory is more accurate than Kinect result but the number of points per trajectory need to be increased for more precise prediction.



**Figure 4-2: ZED Ball Points**

### 4.1.2 Colour Detection

We used colour detection to identify and track the ball but it proved inefficient regarding speed as it processed the image for each of the three primary colours of light R, G and B.

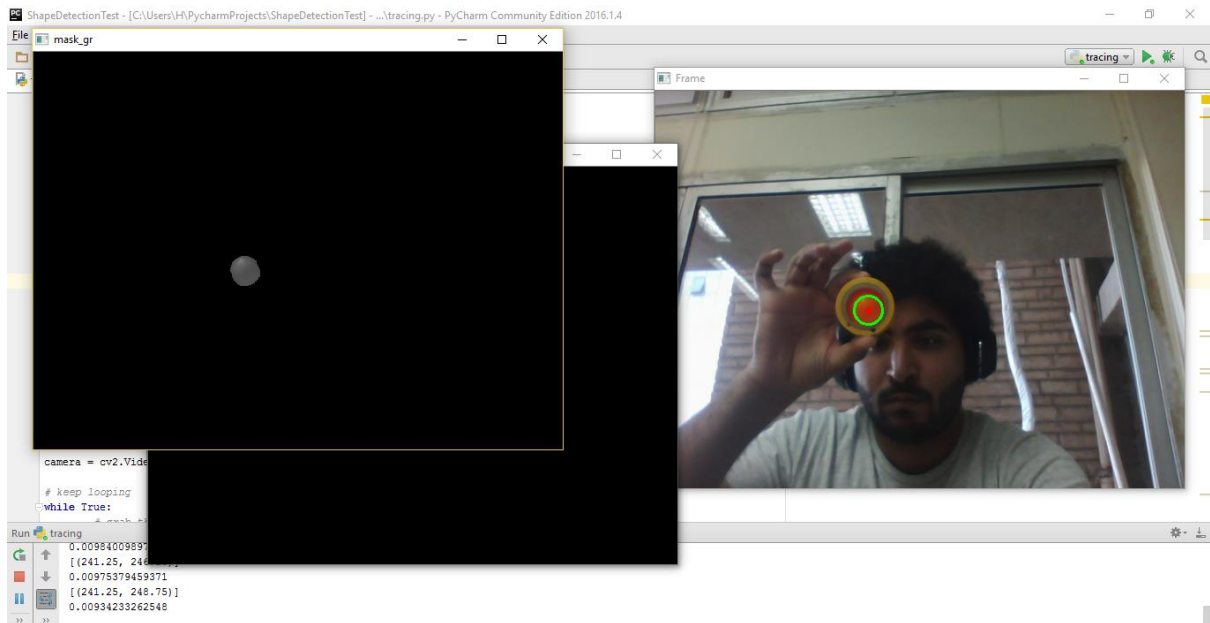
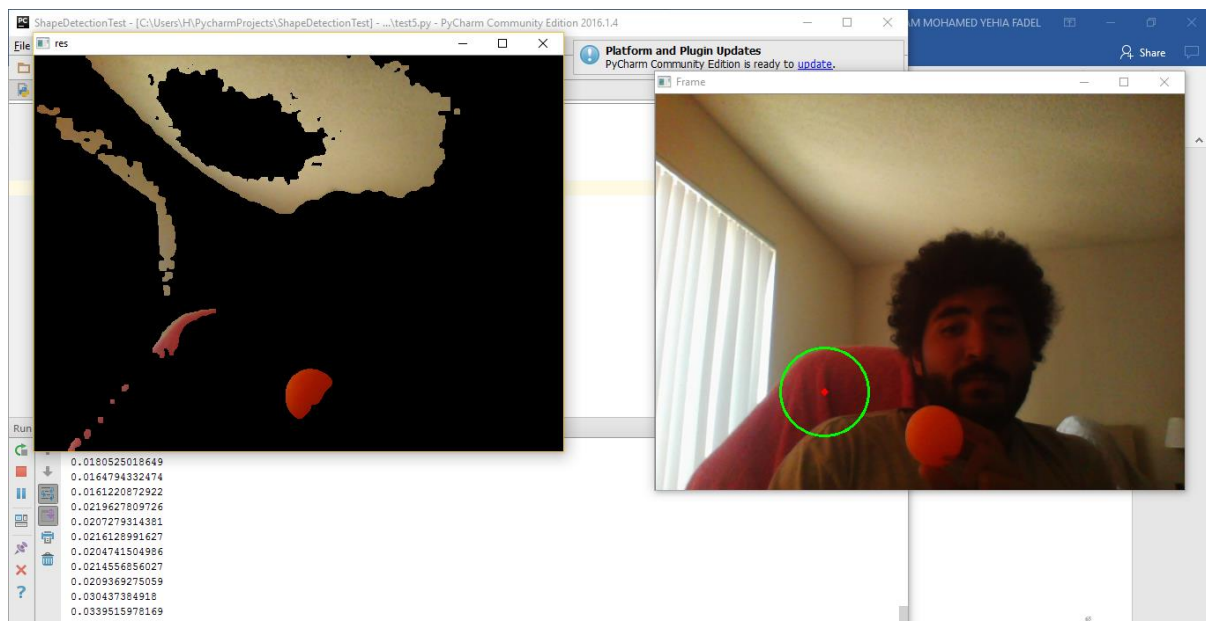
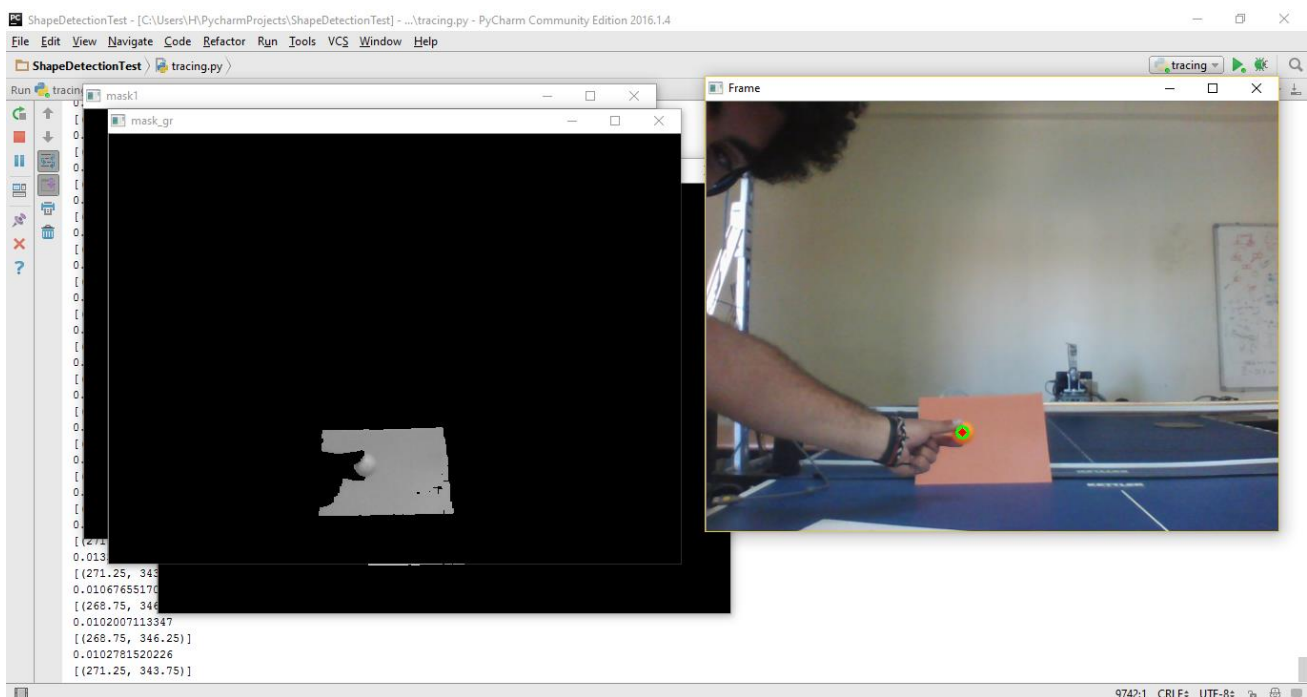


Figure 4-3 Color Detection

It also was not very accurate as any change in the lighting or anything with a colour close to the ball's colour including the player's paddle would throw it off.

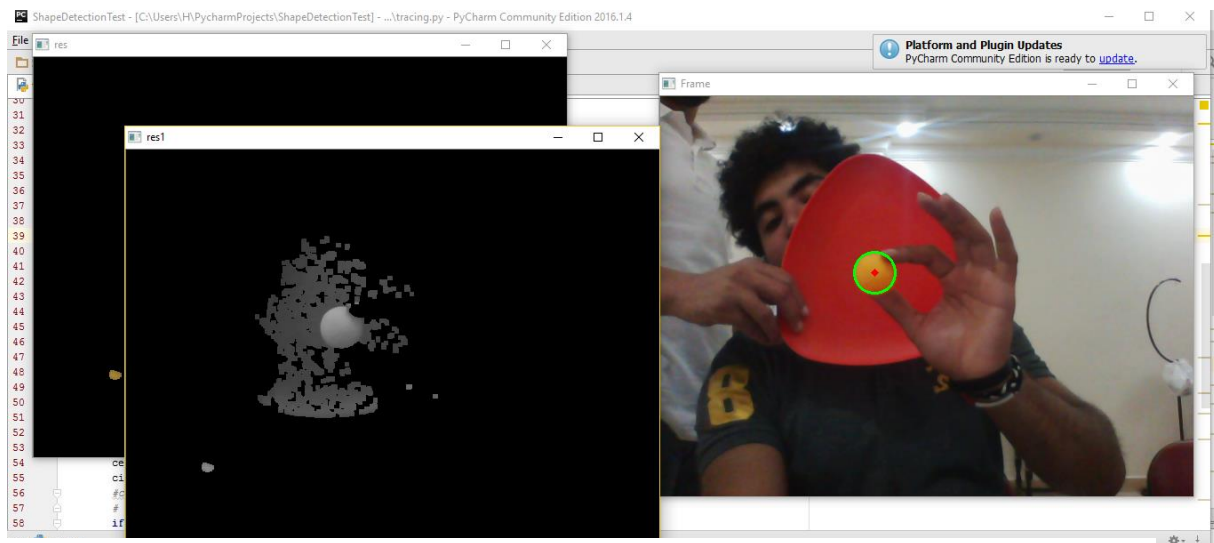


**Figure 4-4 Color Detection Error**



**Figure 4-5 Color Detection mixed to Shape Detection**

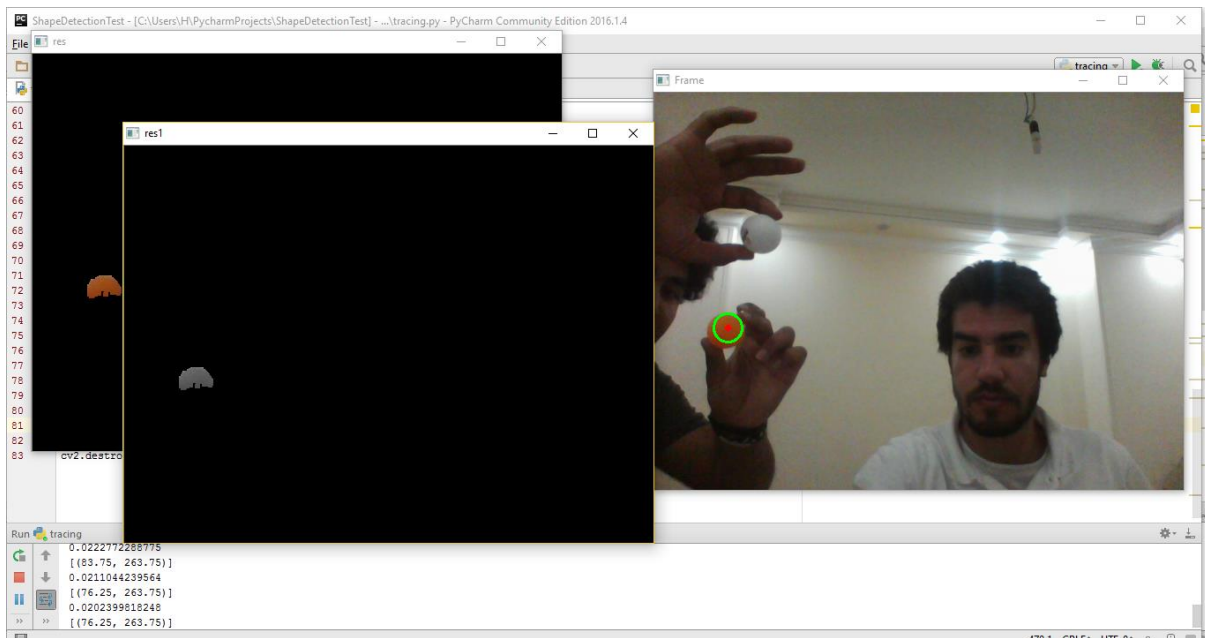
### 4.1.3 Colour and Shape Detection



**Figure 4-6 Accurate Detection between 2 deferent shapes**

We then added a shape detection algorithm along with the colour detection to restrict it to detect only balls with a specific colour. This proved to be extremely accurate in detecting the ball.

But it reduced the processing speed quite a bit that the output stream to the path prediction algorithm had a very low FPS that was practically unusable.

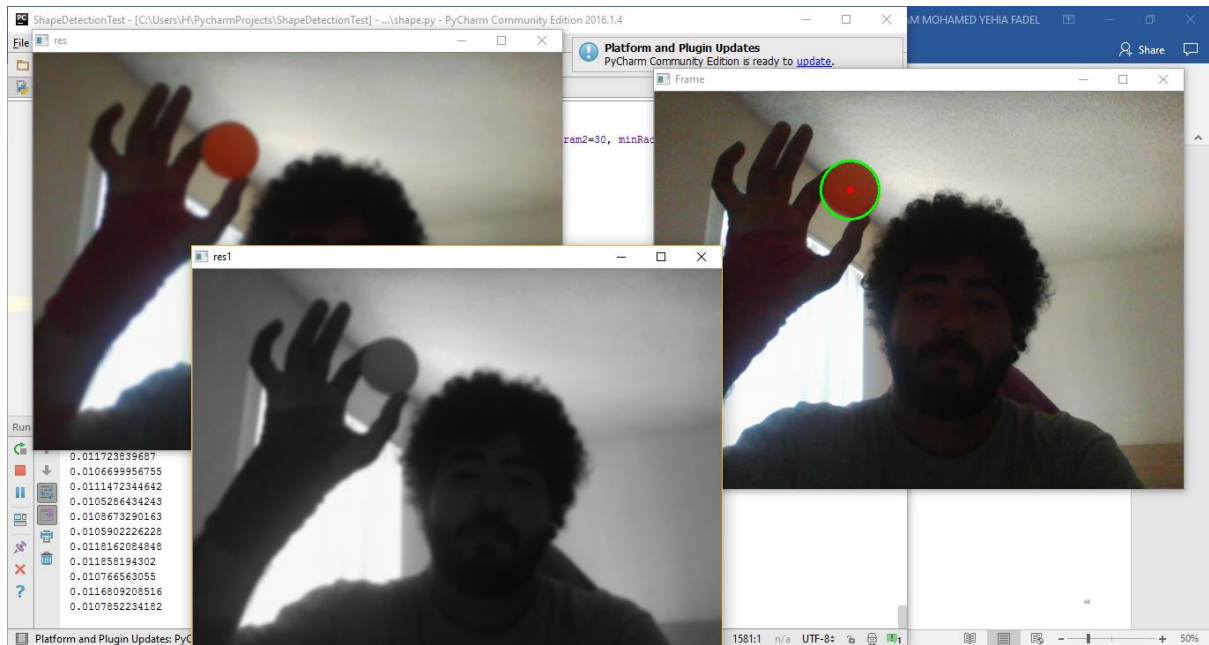


**Figure 4-7 Accurate Detection between 2 similar shapes**



### 4.1.4 Shape Detection

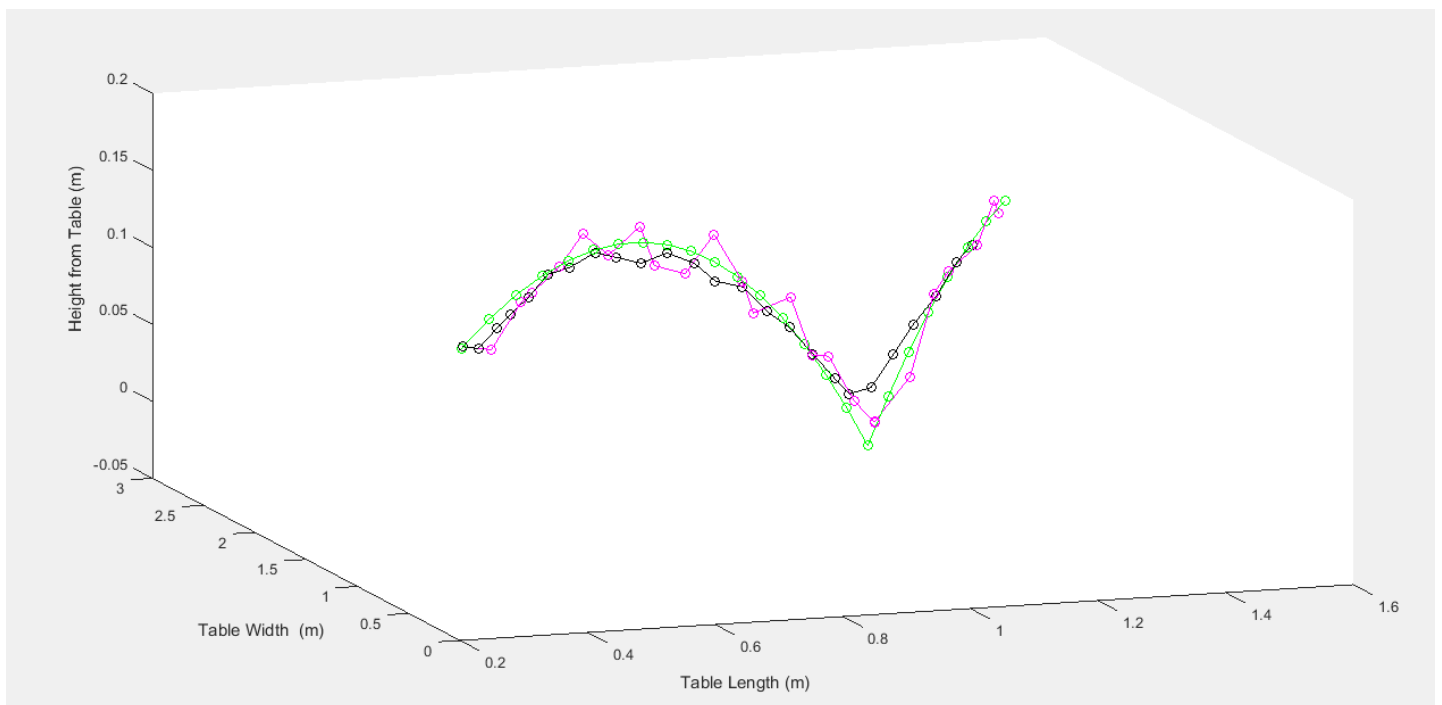
Finally, we decided on grayscale colour detection along with the shape detection algorithm for a good compromise between accuracy and processing speed.



**Figure 4-8 Shape Detection**

### 4.1.5 Path Prediction

In order to improve the algorithm's performance, two improvements were introduced. Firstly, a Kalman filter was utilized to improve the camera's readings and provide a better estimate which is closer the true location of the ball. Secondly, instead of sending the actuation system every single hitting point, statistical methods were employed to find the mean, standard deviation and Z score whenever the number of hitting points is more than 3.



**Figure 4-9: Kalman filtered path**

The results of introducing the Kalman filter can be shown below:

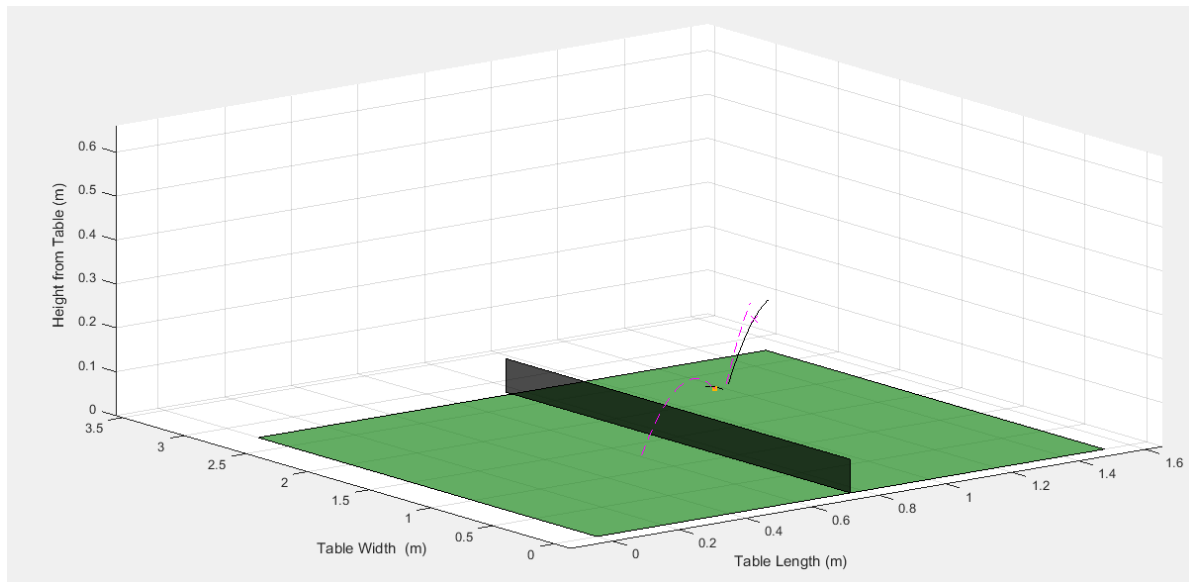
Figure 4-9 shows three curves:

1. The magenta curve represents the camera's readings
2. The green curve represent the true, parabolic path the ball took
3. The black one represents the Kalman filtered path.

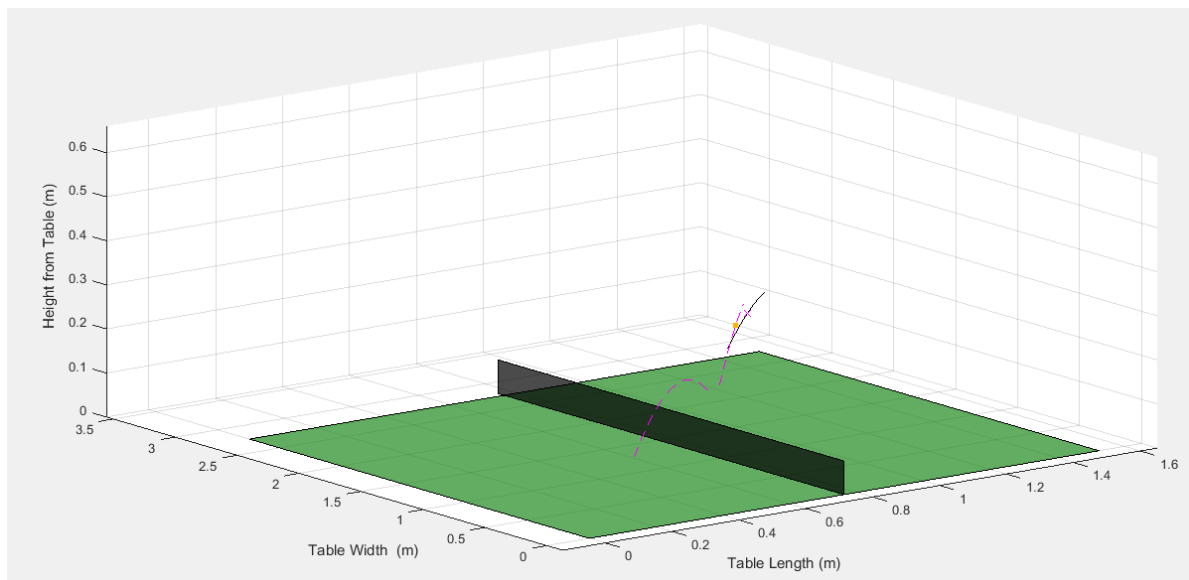


Applying both results and having MATLAB remove the plot of any invalid path yields the following output:

Shown in magenta is the filtered path of the ball. The position of the ball in the simulation is in orange and the subsequent predicted path of the ball is in black. At the end of the predicted path, the hitting point is depicted using a magenta cross.



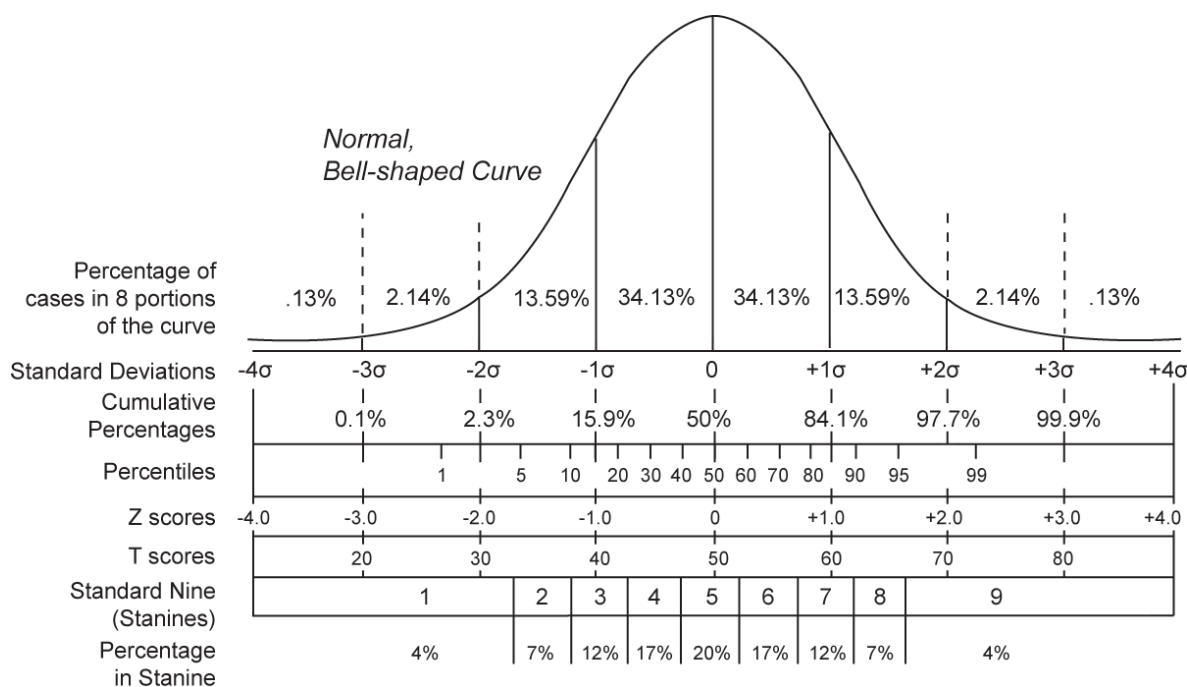
**Figure 4-10 Valid Prediction 1**



**Figure 4-11 Valid prediction 2**

The second method used to improve the algorithm's output was using statistical methods to improve the hitting point's validity and ignore any outliers.

Every prediction's hitting point is stored and after 3 predictions, the average of the resulted hitting points is calculated as well as their standard deviation. Using the average and standard deviation, the standard score or Z score is calculated for each hitting point. If the absolute value of any point's Z score is more than 1.5 (the hitting point is only 5% probable), that point is ignored and the previous hitting point is sent to the actuation system.



**Figure 4-12 Normal distribution curve and Z scores**

#### 4.1.6 Robotic Arm

Several tests were conducted on the Robotic arm (DoBot) to ensure its capabilities. First it was tested using its default software in all modes. It proved its success in the operation. Payload test was conducted, it was found that it can carry payload 500 gm. Hence, we were able to design a light weight racket much less than this payload, so that the robotic arm can move smoothly and hit the ball. After that accuracy and repeatability test were conducted to find that it has a repeatability of less than 1mm.

After the success of testing the DoBot with its default software, another test was performed with our proposed software using IEEE Standard for Binary

Floating-Point Arithmetic (IEEE 754). The Dobot performed great and it could reach any point within its workspace with a repeatability of less than 1mm and a linear speed of 1500 m/sec.

## 4.1.7 Servo Mechanism

### 4.1.7.1 Servo Motor

Several tests were conducted to find the most suitable speed for the motor, as by increasing the speed the vibration increases which affects the whole system. The optimum speed was 4800 rpm. During practical use, it was found that it will not be suitable to start the motor suddenly by applying the wanted frequency, as this causes vibrations, stresses and may cause cracks. Thus, it was found that the best solution was to start and stop the motor smoothly an algorithm was designed to achieve this task and it was tested and it proved its success as shown in Figure 4-13.

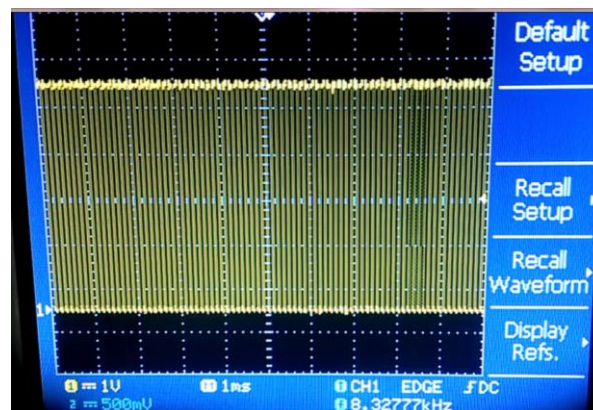


Figure 4-13: Frequency of the generated signal

### 4.1.7.2 Linear Actuator

Upon testing the linear actuator, we achieved a speed of 0.42 meters per second at 5000 RPM without any problems. Due to manufacturing defects, there is a significant amount of vibration noticeable when operating at high speed, however it does not affect the operation of the system. We concluded that using a belt system would be faster but would lose its precision and would wear out much more quickly as the belt becomes more elastic.

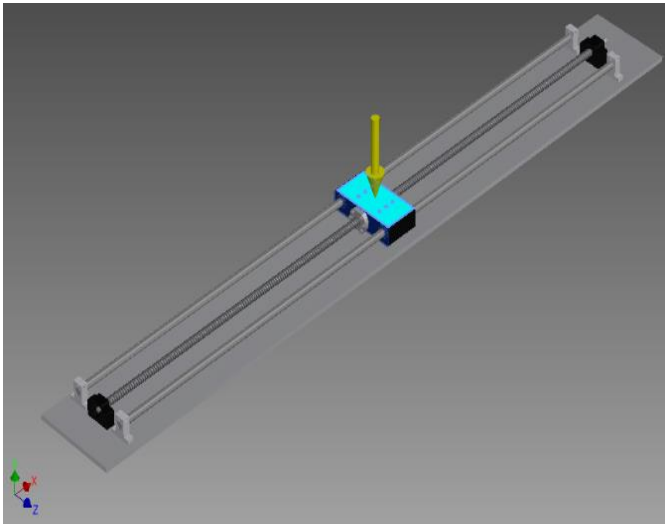


Figure 4-15 Fixed Constraint

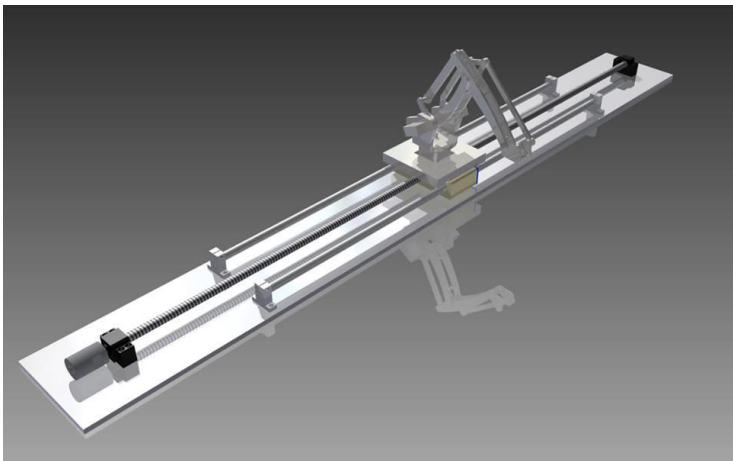


Figure 4-14 Early Design Process Design

#### 4.1.8 Reaction Force and Moment on Constraints:

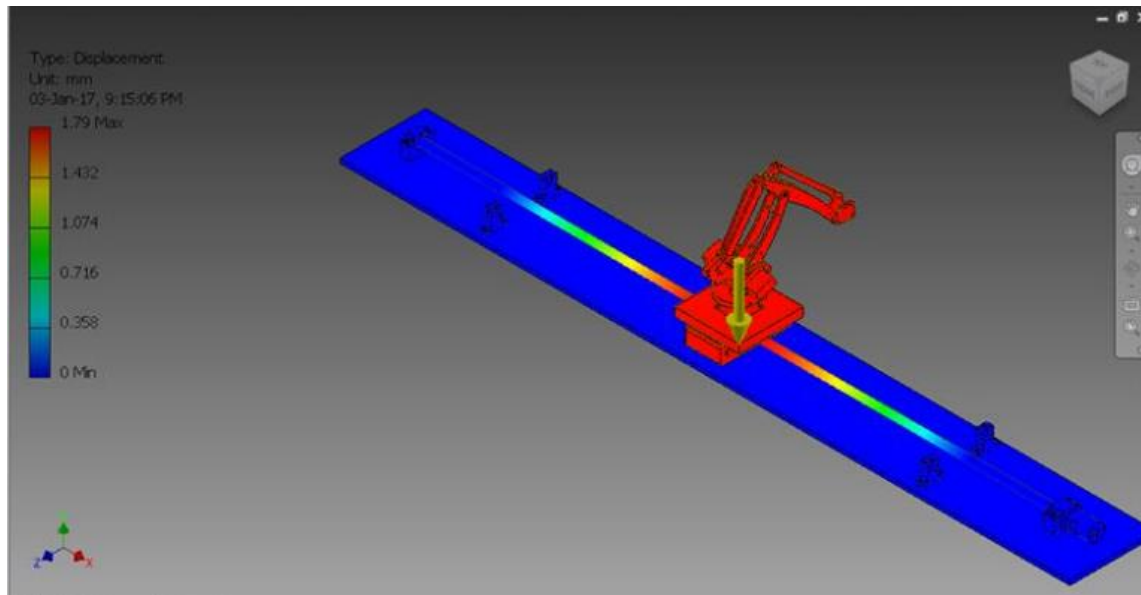
Name	Minimum	Maximum
Volume	4979020 mm <sup>3</sup>	
Mass	17.8243 kg	
Von Mises Stress	0.00000575574 MPa	22.3644 MPa
1st Principal Stress	-7.7312 MPa	22.6104 MPa
3rd Principal Stress	-21.9706 MPa	9.09751 MPa
Displacement	0 mm	0.491857 mm
Safety Factor	11.1785 ul	15 ul

#### 4.1.9 Results:

Constraint Name	Reaction Force		Reaction Moment	
	Magnitude	Component (X,Y,Z)	Magnitude	Component (X,Y,Z)
Fixed Constraint:1	48.4551 N	0 N	7.50401 N m	0 N m
		48.4551 N		0 N m
		0 N		-7.50401 N m

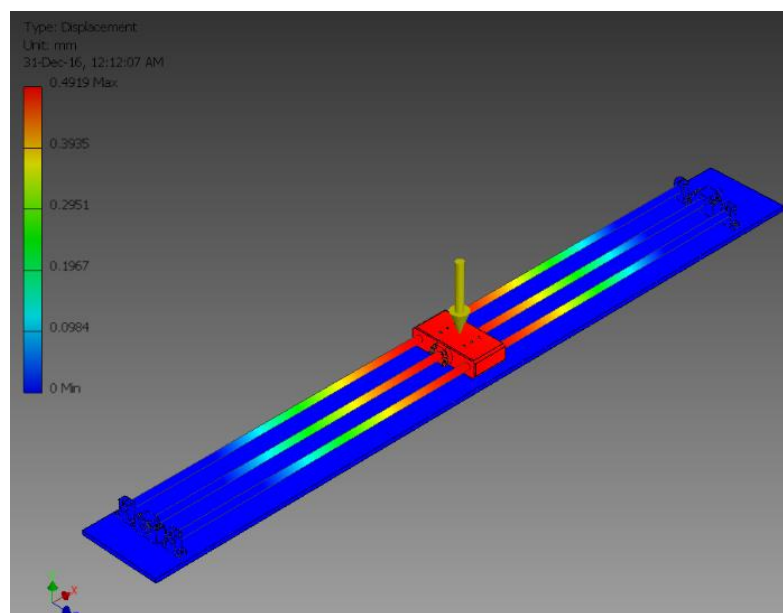
After designing the mechanical system on Autodesk Inventor, we then used the stress analysis environment to perform some basic analysis on how the system will hold up to the loads applied to it while working. The analysis results include displacement of the parts as they bend when they are loaded as well as the stresses they experience from the loading. It also includes a visual

representation of the distribution of the stresses in the different materials as well as the displacement distribution. The program was able to calculate the stresses based on the material information we input such as the alloy number of the stainless-steel rods and the alloy number of the aluminium supports. The accurate models of the different parts have a huge contribution to the efficiency of the analysis.



**Figure 4-16 Displacement analysis**

We can conclude from this stress analysis that the ball screw in this assembly and under the simulated loads can rotate at 6000 RPM according to manufacturer specifications and limiting load ratings. The simulation also shows that the maximum displacement of the parts due to the loads is acceptable for operation.



**Figure 4-17 Stress Analysis**

## **5 CONCLUSION AND FUTURE WORK**

### **5.1 CONCLUSION**

We were able to build the system as planned. As an initial step, every sub system, was built stand alone, where we could combine efficient image processing and depth data to get accurate 3D coordinate for the ball.

Moreover, we could reach highly accurate Aerodynamic Model for the ball; this enabled us to predict the ball path. Also, we built a horizontal slider to deliver the robot arm in any position with high speed and accuracy. Furthermore, we could achieve full position Control on the slider mechanism using the AC servo motor.

### **5.2 FUTURE WORK**

- Multiple vision systems will be used to localize the ball in order to achieve the most accurate result.
- Using Artificial Intelligence to increase the capability of the robot to deal with any possible scenario.
- Using GPU instead of CPU.

## 6 BIBLIOGRAPHY

- [1] R. L. Anderson, "Aggressive trajectory generator for a robot ping-pong player," *IEEE Control Systems Magazine*, *MIT Press*, vol. 9, 1989.
- [2] R. L. Andersson, "A robot ping-pong player: experiment in real-time intelligent control," *MIT Press*, 1988.
- [3] J. J. Rodrigo, J. A. Mendez, G. N. Marichal, M. Sigut L. Acosta, "Ping-pong player prototype," *IEEE Robotics & Automation Magazine*, vol. 10, 2003.
- [4] T. Hashimoto, F. Miyazaki M. Matsushima, "Learning to the robot table tennis task-ball control & rally with a human," *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2003.
- [5] T. Hashimoto, M. Takeuchi, F. Miyazaki M. Matsushima, "A learning approach to robotic table tennis," *IEEE Transactions on Robotics and Automation*, vol. 21, 2005.
- [6] M. Takeuchi, M. Matsushima, T. Kusano. T. Hashimoto F. Miyazaki, "Realization of the table tennis task based on virtual targets," *IEEE International Conference on Robotics and Automation*, vol. 4, 2002.
- [7] F. Miyazaki, M. Matsushima, M. Kawatani, T. Hashimoto M. Takeuchi, "Dynamic dexterity for the performance of 'wall- bouncing' tasks," *IEEE International Conference on Robotics and Automation*, vol. 2, 2002.
- [8] S. Masoumzadeh, M. R. Ghahroudi R. Sabzevari, "Employing ANFIS for object detection in robo-pong," in *International Conference on Artificial Intelligence*, 2008.
- [9] K. Hashimoto, M. Ishikawa Namiki, "Hierarchical control architecture for high-speed visual servoing," *The International Journal of Robotics Research*, vol. 22, 2003.
- [10] N. H. C, Yung, P. Y. S. Cheung H. Y. Chung, "A novel quadrilateral-based tracking method," in *the 7th International Conference on Control, Automation, Robotics and Vision*, 2002.
- [11] A. K. Jain, M. P. D Y. Zhong and ubuisson-Jolly, "Object tracking using deformable templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, 2000.
- [12] A. Mecocci, F. Moschetti L. Favalli, "Object tracking for retrieval applications in MPEG-2," *IEEE Transactions on circuits and systems for video technology*, vol. 10, 2000.
- [13] N. Paragios. R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, 2000.
- [14] by J.W.M. Bush. The aerodynamics of the beautiful game. [Online]. <http://math.mit.edu/~bush/wordpress/wp-content/uploads/2013/11/Beautiful-Game-2013.pdf>

- [15] Xi. Chen, "A Robust Vision Module for Humanoid Robotic Ping Pong Game," *International Journal of Advanced Robotic Systems*, 2015.
- [16] Automata Technologies Limited. (2015) <http://www.getautomata.com/>.
- [17] 7bot. (2016) [Online]. <http://www.7bot.cc/>
- [18] Shenzhen Yuejiang Technology Co. Ltd. Do Bot User Manual. [Online]. <http://dobot.cc/>
- [19] Ing. Roman Berka LuKas Barinka. Inverse Kinematics - Basic Methods. [Online]. <http://old.cescg.org/CESCG-2002/LBarinka/paper.pdf>
- [20] Andreas Aristidou and Joan Lasenby, "Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver," University of Cambridge, Cambridge, CUED/F-INFENG/TR-632, 2009.
- [21] University of Central florida. Stepper Motor and Its Driver. [Online]. [http://pegasus.cc.ucf.edu/~cham/eas5407/project/Intro\\_stepper.pdf](http://pegasus.cc.ucf.edu/~cham/eas5407/project/Intro_stepper.pdf)
- [22] National Instruments. Stepper Motor Theory of Operation. [Online]. <http://www.ni.com/white-paper/14877/en/>
- [23] Bill Earl. All About Stepper Motors. [Online]. <https://cdn-learn.adafruit.com/downloads/pdf/all-about-stepper-motors.pdf>
- [24] Rajvir Singh. Advanced AC Drive- VFD, Servo & Stepper - Powerflex & Delta. [Online]. [udemy.com](https://www.udemy.com/)
- [25] Delta Electronics, DELTA Standard AC Servo Drive for General Purpose Applications (ASDA-B2 Series User Manual), 2013.
- [26] Ramu\_Krishnan, *Permanent Magnet Synchronous and Brushless DC Motor Drives*.: CRC Press, 2010.
- [27] Shenzhen Yuejiang Technology Co. Ltd. Dobot Communication Protocol. [Online]. <http://www.dobot.cc/download/protocol-for-v1-0/>
- [28] "IEEE Standard for Binary Floating-Point Arithmetic," *IEEE Standards* , 1985.
- [29] D. Xu, J. Yu Z. Zhang, "History and latest development of robot ping-pong player," *The 7th World Congress on Intelligent Control and Automation*, 2008.
- [30] Adrian Rosebrock. Pyimagesearch. [Online]. <http://www.pyimagesearch.com/>
- [31] Greg Welch and Gary Bishop, "An introduction to the Kalman filter," Department of Computer Science University of North Carolina at Chapel Hill Chapel Hill, 2006.
- [32] Adrian Rosebrock, *Practical Python with OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision.*, 2014.