Synthetic Data Production for Machine Learning Using Robotic Arm

Final Year Project Report
Presented
by

M Muneeb Babar

Fa-2016/B. Sc. EE/059

M Hamza Liaqat

Fa-2016/B. Sc. EE/065

M Mustafa Imran

Fa-2016/B. Sc. EE/60

In Partial Fulfillment

of the Requirement for the Degree of

Bachelor of Science in Electrical (Telecommunication) Engineering /Computer Engineering

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Sir Syed CASE Institute of Technology, ISLAMABAD

July 2020

Synthetic Data Production for Machine Learning Using Robotic Arm

Final Year Project Report

Presented

by

M Muneeb Babar

Fa-2016/B. Sc. EE/059

M Hamza Liaqat

Fa-2016/B. Sc. EE/065

M Mustafa Imran

Fa-2016/B. Sc. EE/60

In Partial Fulfillment

of the Requirement for the Degree of

Bachelor of Science in Electrical (Telecommunication) Engineering /Computer Engineering

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Sir Syed CASE Institute of Technology, ISLAMABAD

July 2020

Declaration

We, hereby declare that this project neither as a whole nor as a part there of has been copied out from any source. It is further declared that we have developed this project and the accompanied report entirely on the basis of our personal efforts made under the sincere guidance of our supervisor. No portion of the work presented in this report has been submitted in the support of any other degree or qualification of this or any other University or Institute of learning, if found we shall stand responsible.

Signature:_____ **Signature:_____**

Name:_____ **Name:_____**

Signature:_____ **Signature:_____**

Name:_____ **Name:_____**

Sir Syed CASE Institute of Technology, ISLAMABAD

July 2020

Final Approval

This Project Titled

Submitted for the Degree of

Bachelor of Science in Electrical (Telecommunication)/Computer Engineering

by

Name	**Registration Number**
M Muneeb Babar	UET-16F-BSc-EE-CASE-36
M Hamza Liaqat	UET-16F-BSc-EE-CASE-17
M Mustafa Imran	UET-16F-BSc-EE-CASE-20

has been approved for

Sir Syed CASE Institute of Technology, ISLAMABAD

Supervisor	***Head***
Name	***Department of Electrical and***
Designation	***Computer Engineering***

Dedication

Dedicated To

Our Families & Friends for Supporting Us & Providing any help, which was needed, & a Special Feeling of Gratitude to Our Loving Parents For their Constant Support, Love & Encouragement.

They have been constantly providing us Spiritual, Moral & Emotional Support & have been a Source of Inspiration without which we could not have reached this Milestone.

Acknowledgements

Our gratitude for the omnipotent Allah cannot be expressed in a few words, for the strength, protection, skills and power of mind and a healthy life; whose blessings made it possible for us to have such accomplishments.

We are sincerely grateful for our supervisor, Dr. Shoab A. Khan's (The Sir Syed CASE Institute of Technology) persistent moral and intellectual support, guiding us when necessary and helping overcome the setbacks we faced.

We would like to express our gratitude to our co-supervisor, Sohaib Tallat, who guided us throughout this project and who helped us work past the obstructions we experienced. We also are thankful to him as we learned many things under his guidance.

We are very thankful to the Sir Syed CASE Institute of Technology for allowing us and assisting us in working our project at their facility and allowing us to use the necessary resources at our discretion.

Last but not least, we are thankful to have supportive friends and family who were willing to lend an extra hand whenever needed.

M Muneeb Babar

M Hamza Liaqat

M Mustafa Imran

Abstract

Due to the ongoing research in the field of computer vision, many tasks, which were deemed impossible to perform, have been made easier. As technology progresses, research is being performed on how to make life easier by allowing computers to perform tasks that can be repetitive and harmful to the human body. Such tasks include collecting garbage, disposing of nuclear waste, or any task, which requires a human to perform grasping/releasing motions. With the new work done on computer vision, now images can be processed in real-time and be edited or worked upon by any means. The scope of application for the idea behind this project would be rather huge as a simple pick and drop motion after automation can be used in a plethora of ways, from being used as an object classifier using a live image input to a full-fledge robot used to perform tasks around a fixed area.

Our idea integrates the already available libraries while working with python and using the functions, developed a program to find and track objects present on a surface. Multiple approaches were taken to identify the object but the most efficient solution was to use simple MOG background subtraction with a static background and then find the approximate measurements of the object. In our project, we make use of the python OpenCV library to perform such actions and acquire the approximate location of the object by using a 2-way system, a coordinate system, and a grid system. This helps us reduce the error by first locating the object on the grid, then to reduce error, using the x-axis and y-axis as the landmarks, locate the coordinates as accurately as possible.

Using these coordinates, we interact with the object using a robotic arm as the medium and perform actions such as picking it and dropping it at a predefined location. Another feature that was worked upon was identifying the object using a retrained ML model through which a simple classification robot can be developed. This simple process can be used to create a wide platform with multiple devices connected and used to automate some simple tasks.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

In this chapter, we discuss the details about the core idea behind our project, the market analysis performed and the contribution of our project to the society and the environment.

1.1 Introduction

In the current technological era, researches are being done just to make our daily lives more convenient and as the era of automation is moving towards its peak, work is being done to automate tasks, which are repetitive and harmful for the human body, or even jobs, which are high risk. All around the world jobs, which were conventionally performed by humans, are transitioning to automation and this has helped in creating a huge amount of new jobs for trained professionals as well as the unskilled population and with the internet having most of the resources required to become adept in the field, the industry is booming. Many simple applications of computer vision have been made open source but the actual work done on projects which actually are based on hardware are still low in number. This project takes a simplistic approach by using mainstream python libraries and software used to manipulate the hardware.

Let's take a simple problem in which we have to find the object from the image. How can we go about doing this?

Figure 1

There are multiple ways by which we can accomplish this, some of which are:

- Simple color detection: Give the program a set of values in hexa-decimal for color representation and any color other than white and black that is present on the white background can be highlighted. However, the problem is what if the object is of the same contrast as the background. Then what?
- A more sophisticated approach would be to specify a static background and if any object is being placed on the background surface, we can easily detect the change by a simple

calculation of the number of pixels, as the values will change with a new object. However, in some cases the computer may fail to detect the colors due to any external condition such as light discrepancy, shadow etc.

- The most efficient approach is to use the above methods side by side and instead of using the raw image, change the color model/space to HSV as this makes it easier for the computer to detect the change in pixels.

As you can see in figure-1 and figure-2, applying the filter makes a huge difference and the colors are emboldened which makes it easier for the computer to calculate change in pixel values. This, simple helps in identifying if there is an object present on the surface.

Figure 3-RGB Image

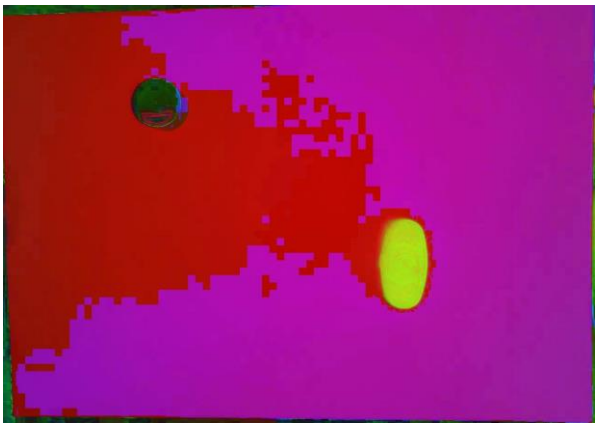

Figure 2-HSV Filter

Now as we also want to identify what the object is we have to employ some type of machine learning model, which has been extensively trained to recognize objects with a large database at the back. For this, we use the most widely known visual recognition model named AlexNet, which has a huge collection of about 1.2 million images that were used to train it. This will allow us to identify as well as recognize the object.

Taking it a step further, as we wanted a robotic arm to interact with the object, we required the object's measurements, which were simply obtained by fixing the camera to a fixed height and using a formula to obtain the values according to a ratio.

Using the same libraries and functions, we can acquire the approximate location of the objects in the form of a set of coordinates or create a grid and the output is in the form of the number of row and column. By using these both simultaneously, the error can reduced to the bare minimum and using the objects measurements we can find the center of the object, be it an irregular shaped object or a regular shaped object.

However, as our end goal is to grasp the object and move it around, we need a medium through which this can be achieved. For this, we choose a 6-DOF robotic arm, as it can be versatile as well as delicate while handling the objects and has a 270-360 degree of rotational freedom. By using Robot Operating System, otherwise known as ROS, which is robotics middleware, a virtual environment was developed with a 3D model of the robotic arm. By synchronizing the virtual model and the physical model, we achieved the precise motions for grasping/releasing and moving the arm freely.

This project shows that computer vision can be used for day-to-day activities rather easily and the solution to one task can be easily manipulated to be used for another task as the base function of grasping and releasing is the same.

1.2 Scope of the Project and Quantifiable Outcomes

As we are ultimately going towards a digital world, computer vision and automation will play a big part in achieving the goal. As technology progresses people all around the world want a more convenient way of life where menial and repetitive tasks are automated and humans perform the challenging tasks. As more and more tasks are automated, the novelty of the idea remains which helps in promoting products that use the idea behind our project as their core. The simple idea behind this project can be developed to such an extent that it can be used as an autonomous assistant in a biology/chemistry lab or as chain of robots in a fully automated car-manufacturing factory where each robot is used for a single specific task. A benefit of the project would be as everything is run by a computer, the chance of an error occurring would be much lower than the counterpart when a human is performing the same task. The hardware medium in question can be as large as a crane or as small as a microchip but the same computer vision concept can be applied.

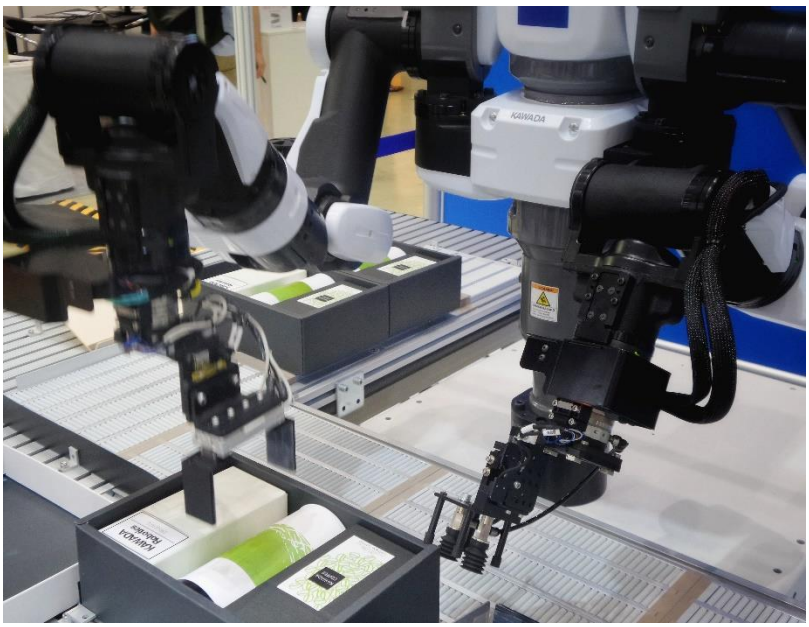

Figure 5-Robotic Arm Performing Task

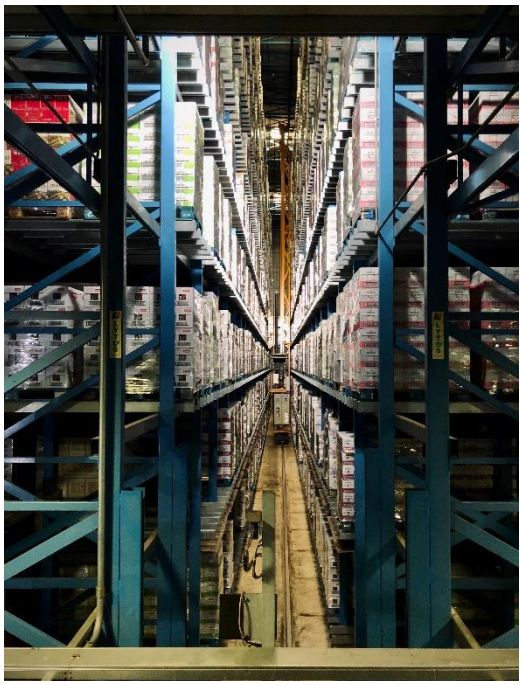

Figure 4-Robotic Arm Classifier

The project will lead to the development of an easy-to-use computer vision program which will have multiple features such as object detection, recognition (using ML classifier) and live tracking which, if required can be manipulated to be used for any specific category of objects or even humans or vehicles. With the hardware medium being a robotic arm, our goal will to develop a system where using the measurements and the location coordinates acquired by the program, the robotic arm will automatically move and interact with the object and used as a classifier or to perform any action which requires grasping and release. The 3D model as well as all the included configuration files will also be uploaded to an open source community where anyone can use our project to either learn something or create something new. The project files could also be used by any educational institute as a learning material as well.

1.3 Academic and Commercial Worth of the Project

As our project integrates multiple technologies, the commercial worth cannot be calculated but on approximated individually. According to 'Globenewswire' the total market worth of computer vision is expected to reach a staggering 48 Billion USD by 2023 and as more and more people warm up to the idea of automation and computer vision for day to day activities, the market will keep on rising. As for the robotics market, according to a study performed by 'fortunebusinessinsights' in 2019, the commercial worth of industrial robots was approximately 21.83 Billion USD. While a study performed by 'marketsandmarkets' showed that industrial and service robotics has a share of about 48.7 Billion USD in the international market. Our project incorporates these technologies and creates an idea that with the software and technology currently existing can be used to create sophisticated projects rather easily with a low budget. As seen in the figure below, as the years progress, the number of autonomous robots will increase exponentially making the market boom.

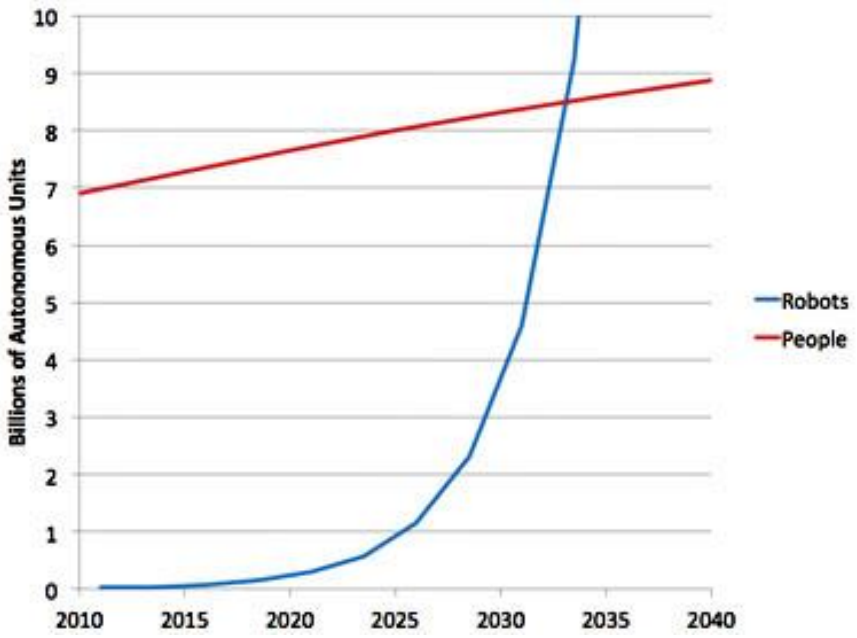

Figure 6-People VS Robots per unit relation

As the project introduces the idea about developing low cost autonomous units, which can be used to perform a plethora of tasks, the academic worth cannot be calculated as all educational institutes could introduce the subject of automation and computer vision and allow the students to work on any project with the core idea based on our project.

1.4 Contribution to the Society and Environment

This project ultimately will lead to a more convenient life which all of humanity is striving for as it can be applied in many situations. An example can be using computer vision and robotics to make a garbage collector. Instead of having a group of humans drive to each house manually and collect the garbage, the robot can be programmed to collect the garbage by itself by incorporating the idea behind our project and that of a self-driving car, and for a larger area, multiple robots could be deployed, each having their own specified area. This simple application can create more jobs such as maintenance of the robots, manufacturing, assembly and programming and the chance of the workers collecting garbage being infected from any disease while working will decrease. This is a simple application; another can be deploying the robot to dispose of chemical waste instead of a human doing it and being in a health hazardous situation. This shows that the core idea behind this project, which is computer vision and robot control, can be used in our daily life to a large extent and it can help create more jobs as well as a safer environment as with a robot, the probability of an error occurring is lower than when the same tasks is being performed by a human.

With our program, the prominent features would include detection of objects, recognition of objects and tracking in real time. This helps in a physical environment because objects being interacted with will not remain stationary in some cases. With live tracking, we obtain the center point of all objects through which the object cam be grasped as the claws of the robot simply have to close while keeping the point at center. This project can be used by anybody with a little bit of skill and knowledge about the backend of the program and can be manipulated into being a garbage collector or a classifier or for a school project.

1.5 Report Structure

This Report cover every aspect of the project in detail and the information is distributed in the chapters as follows:

- Literature Review
- Project Methodology
- Implementation Details
- Results
- Summary

The Literature Review Chapter points out the differences and similarities between previous works and researches that were performed which have a similar core idea. Later we summarize the various researches that were found to be helpful during our project.

As for the Project Methodology Chapter, we go in the depth about the milestones we achieved and the development environment that was set up. The details about work distribution, cost analysis as well as the hardware specifications are also briefly mentioned.

In the Implementation Details Chapter, we discuss in depth the complete methodology, which was adopted while undergoing the development process as well as some detail about the features of the project.

The Results Chapter concludes the project by defining the final results we achieved as well as the data that was collected.

In the Summary Chapter, we summarize the complete work that was performed as well as the results that were achieved and some detail about the versatility of the project and concludes the report.

Chapter 2

2 Literature Review

2.1 Theoretical Background

This chapter includes the Theoretical understanding of distinctive as well as novel concepts and techniques that were essential to learn for the completion of this project. In addition to this, we also reviewed multiple research papers and case studies in order to get the reference of the tasks that were needed to perform in our project

This section of the chapter includes the basic concepts related to this project and their theoretical backgrounds. We will be going through concepts such as:

- Background Subtraction
- Computer Vision
- AlexNet (Machine Learning)
- Kinect Depth Sensor
- ROS

2.1.1 MOG2 Background Subtraction:

Subtraction of Background from Foreground is a standard technique for segregating the fluid objects from a visual frame by segregating them into two different frames.

This algorithm computes the forefront mask by carrying out a deduction between the current instance and a background frame, which contains the static objects of the scene. Background modeling is based upon two main steps, which are:

- Background Initialization
- Background Update

Firstly, a model of the background is acquired, and then the model is restructured to adjust to possible changes in the scene.

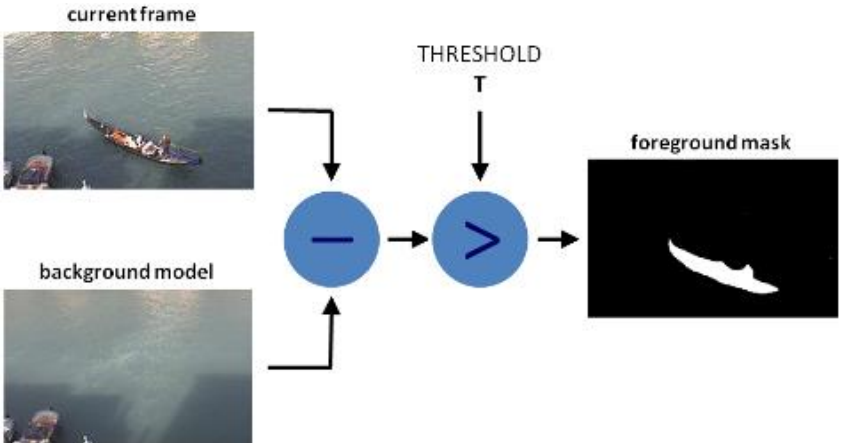

Figure 7-Thresholding

The three mainstream background subtraction algorithms that are used in Open-CV are GMG, MOG, and MOG2.

- **GMG :** The algorithm structures the background using a mixture of Kalman Filters and Bayesian deduction.
- **MOG :** This algorithm works with k Gaussians, which structures each background pixel, with k lying between 3 and 5.
- **The MOG2**, which is mapped pixel by pixel, permits multiple Gaussians distributions, which allows MOG2 to achieve an enhanced depiction of the depth of colors in every frame.

Using Python's library Open-CV we have used the technique of Background Subtraction called MOG2 Background Subtraction. It is based on Gaussian Mixture and it incorporates the Foreground Background Segmentation Algorithm. It operates through a process to transform each background pixel by multiple K Gaussian distributions where K is between three and five. The weights of the Gaussian mixture define the time proportions for colors that are stationary.

Onto the implementation of OpenCV, MOG2, the algorithm that was used, takes three inputs that are variable to calibrate for each frame.

- **History:** This parameter signifies the quantity of frames to use while modeling the background.
- **varThreshold:** It creates a relationship between the weight of the pixels on each frame and the values of the model. A Lower value will lead to the creation of false positive.
- **detectShadows:** This parameter takes input in the form of either 1 or 0, 1 enables the model to detect shadows but the downside is that the processing time increases.

Figure 9-Sample Image

Figure 8-Sample Mask

2.1.2 Computer Vision

Computer vision is a field of AI that allows a computer to infer and understand the visual domain. Computer Vision is employed to empower computers to process images in a similar way that a human does, and offer suitable output. It can be interpreted as giving a computer intelligence which is somewhat similar to a human. As of now, it is not a menial task to allow computers to differentiate between objects through simple images.

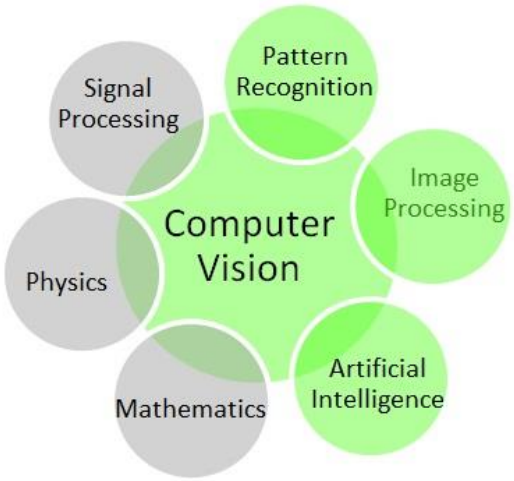

Figure 10

However, using images/videos in a digital form, which are acquired using digital cameras as an input to deep learning models, computers can accurately classify objects and then perform some predefined tasks.

2.1.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) comprises of functions which help solve problems which require ML and computer vision. OpenCV was mainly developed to sustain a platform for computer vision projects and to develop commercial products with machine perception.

The library has more than 2500 optimized functions, which range from an inclusive set of classic algorithms such as simple image processing to contemporary computer vision and ML algorithms such as cascade ML classifiers.

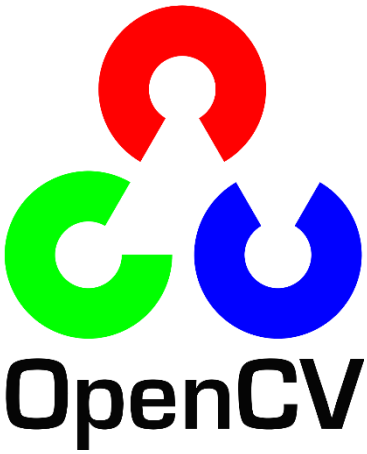

Figure 11

2.1.2.2 Image Processing

Image processing is the procedure of generating an image from an existing one, usually altering the content in one way or the other. It is somewhat similar to digital signal processing however; it does not deal with understanding what the content behind an image depicts.

A computer vision system would generally need some initial processes to be run on the input images. An example of this is pre-processing images.

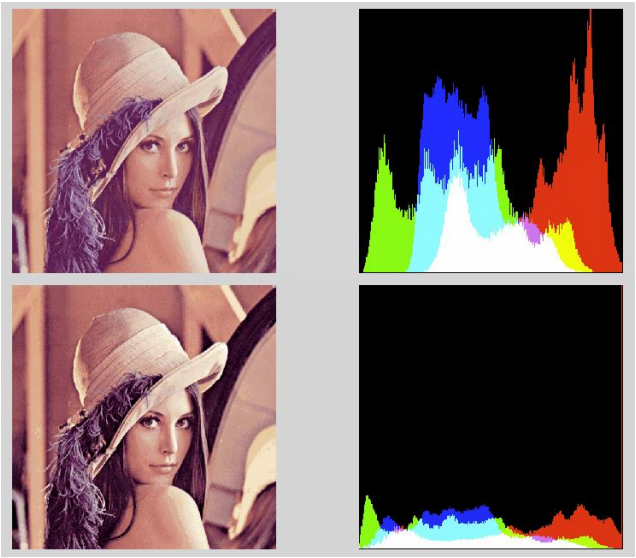

Figure 12-Color Graph

Further uses of image processing may include:

- Normalizing color scale settings of an image, such as hue, brightness.
- Reducing the size of an image and centering it
- Removing or reducing digital noise from an image.

2.1.2.3 Image Thresholding

Thresholding is a technique in which we change our pixel values to the threshold value. The thresholding process takes each pixel value and then compares it with the threshold value. If the pixel value is lower than the threshold, it is altered to 0, else, it is set to a maximum value, that is 255. It also helps in the separation of an object from its background. It orbits around two values below the threshold or above the threshold.

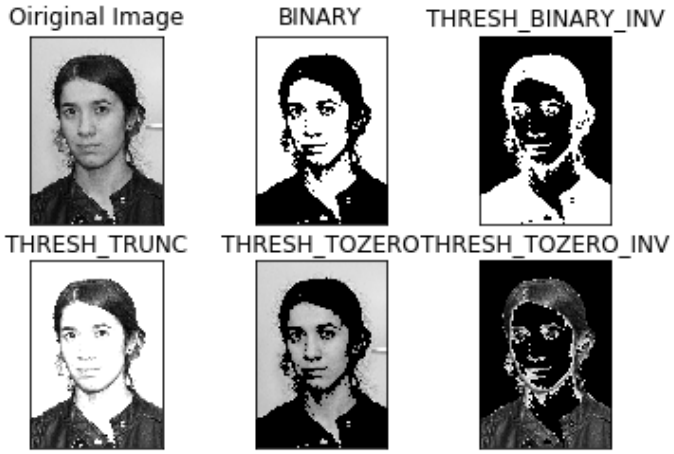

Figure 13-Open CV Filters

2.1.2.4 Image Contours

A contour is generally a closed curvature of points that symbolizes the boundaries of any object in an image. Contours are basically the shapes or figures of objects in a frame. Contours are occasionally also known as the pool of points that generally represent the shape of the object in an image.

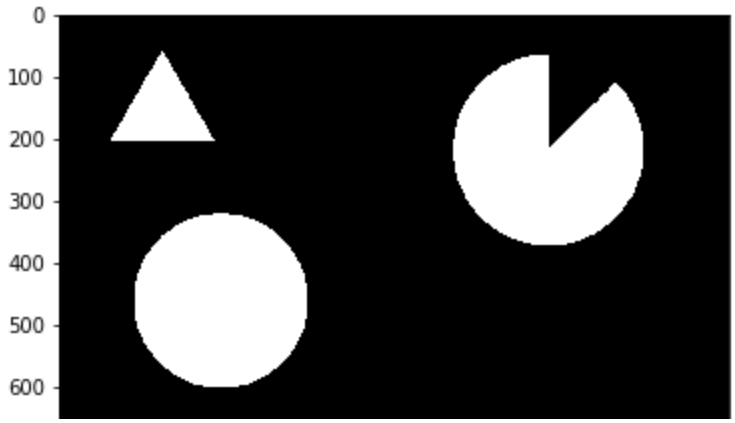

Figure 14-Contours

2.1.2.5 Object Detection

Object Detection is the procedure of finding out the location or position of a certain object in an image or frame. In Python and OpenCV, there are many algorithms that can be used for the purpose of object detection.

Object detection is a methodology, which is based upon computer vision that identifies and locates objects within a visual frame. Precisely, this algorithm creates boundaries around the detected objects, which allows us to approximately locate where a certain objects lies and how they tend to relocate through a frame.

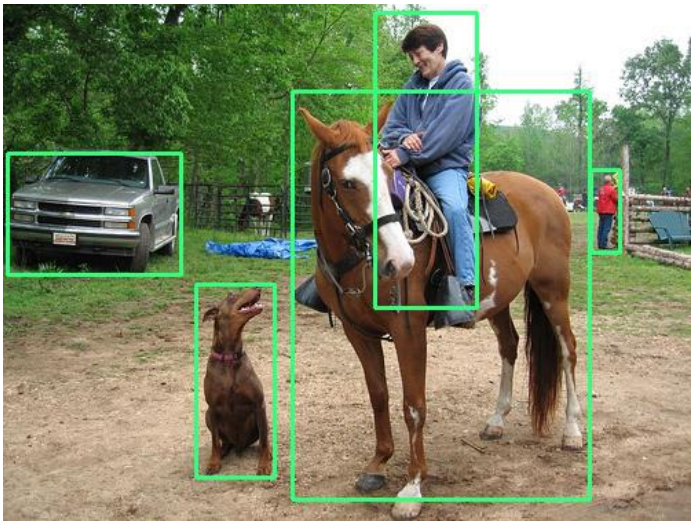

Figure 15-Object Location

2.1.2.6 Object Recognition

Object recognition is a methodology of identifying any object that is present in a visual frame. It is an application, which plays a significant role in the current applications of ML and DL.

Object Recognition algorithms tends to perform as a mixture of an image classifier and an object localization algorithm. It takes a still frame as an input and creates boundaries with a label attached to each object. These systems are proficient enough to handle images with multiple objects of different/same types.

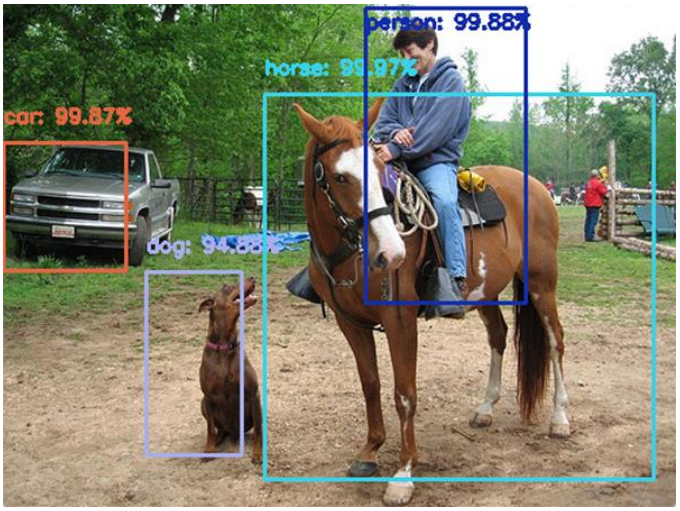

Figure 16-Object Recognition

2.1.3 Kinect Camera

Kinect is a Motion Capture camera manufactured for the Xbox 360 and Xbox One. It has been developed by Microsoft with a mixture of Microsoft built software and hardware. The Microsoft Kinect is a sensor that comprises of :

- an Infrared sensor,
- an infrared digital camera,
- an RGB based camera,
- a multi-array microphone, and
- a digital servo motor,

. The motor provides the frame a functionality to tilt within a certain limit.

The Kinect captures frames at a maximum rate of 30 fps. The Kinect has an RGB camera, which uses an 8-bit resolution with a frame rate 30 Hz while the depth sensor has a resolution of 11-bit, which leads to a total of 2048 variations of sensitivity. The depth data is collected with a pair of an IR projector and camera. The motorized pivot joint can hold the sensor up to 27° maximum either way.

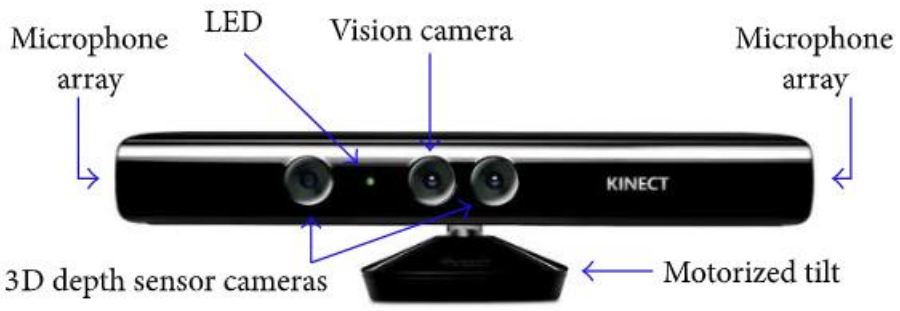

Figure 17-Kinect

There are three main hardware components that are very beneficial keeping their functionality into account. The combination of hardware working within the Kinect are:

- **VGA video recorder** - This camera assists in facial recognition and some other features by detecting the RGB components.
- **Depth sensor** - An infrared-based sensor with a projector and a monochrome CMOS (complementary metal-oxide semiconductor) sensor, which work by visualizing the room in 3-D regardless of the illumination.
- **Multi-array microphone** – Kinect has four microphones that segregate the voices of the users in a game from the noise. This allows the users to stand a few feet away from the microphone and perfectly control the device using voice commands.

The Depth sensor permits Kinect to estimate the distance of each point of the player's body by transmitting infrared rays and measuring the "time of flight" after the reflection occurs.

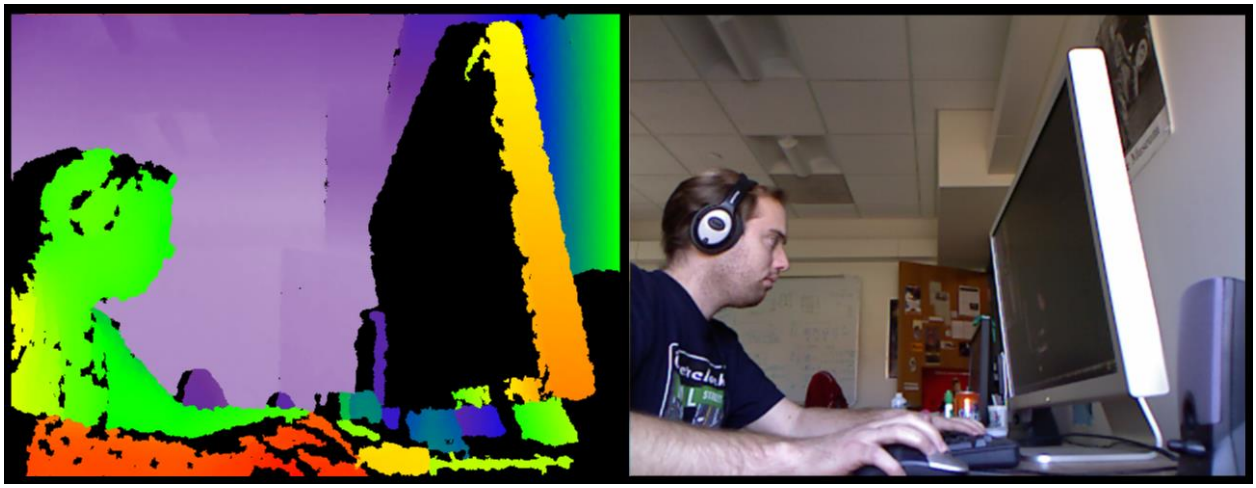

Figure 18-Kinect Depth Image

2.1.4 AlexNet

A Convolutional Neural Network (**CNN**) is a distinctive kind of multi-layer neural network, which is designed to identify and recognize visual patterns directly from pixel images with absolute minimal preprocessing.

AlexNet, a CNN that is based on eight layers, is widely known for its use as a classifier. You can use a generic version of the model which has been trained on more than a million images. The pre-trained network has the ability to classify the images into 1000 different object categories, such as book, mouse, pen, cap, stapler and many others including animals. The network has an input size of 227-by-227 for each image.

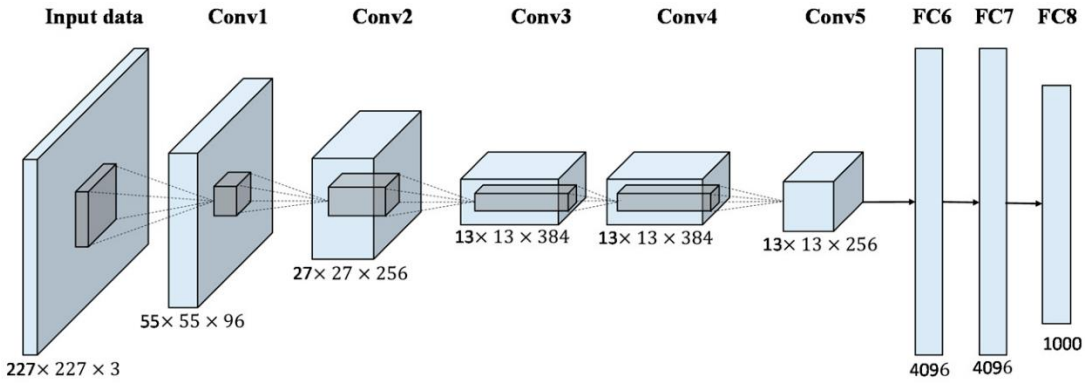

Figure 19-AlexNet Layers

AlexNet consists of a total of 8 layers, five of them being convolutional Layers while the other 3 are fully connected layers.

Multiple Convolutional Kernels, also known as filters extract defining features from an image. In a single convolutional layer, there are multiple kernels similar in size. For example, the first **Conv** Layer of the AlexNet model holds 96 kernels of size 11x11x3. Whereas the width and height are mostly the same while, the depth is the same as the quantity of channels.

- The Overlapping Max Pooling layers are receeded by two Convolutional layers
- The convolutional layers from third to fifth are interlinked with a direct connection.
- The fifth layer is trailed by a Max Pooling layer
- The output of the final layer is sent to a combination of a pair of fully connected layers.

The second fully connected layer's output is given into a softmax classifier layer which has been fitted with 1000 class labels.

Matlab is software that provides us the resources and platform to train an AlexNet model with our own database of millions of images. It provides the user a friendly GUI, that could be used to alter or manipulate its layers.

2.1.5 Robot Operating System

ROS is an open-source framework for robotics. It offers the types of assistance you would anticipate from a well renowned Operating System, including equipment planning such as hardware, low-level electronics control, communication among procedures, and package management. It also provides tools/libraries for obtaining, building, writing code across multiple computers.

Figure 20

The Robot Operating System (ROS) is but a framework and mostly comprises of tools that provide the functions an operating system normally provides. Its handiness is not restricted to just robots, however most of tools built-in are centered around working with hardware. The most widely used operating system for powering up ROS is Ubuntu.

Robot Operating System comprises of a pair of features:

- A core with tools required to communicate
- A set of built-in libraries

ROS is separated in excess of 2000 packages, each package gives a particular usefulness. The number of features of the framework is the main attraction.

The key component of ROS is the manner in which the product is run and the manner in which it conveys, permitting you to plan intricate programs without realizing how each hardware operates. ROS provides a method to connect a network of processes with a core. The processes can be operated on multiple devices through various means.

The main method of developing the network is by giving request able services, or by setting up a multi-faced connection with other nodes. Some types are in-built in the core packages, but message types can be altered by the individual packages.

2.1.5.1 How ROS Works :

In spite of the fact that ROS is not a conventional operating system, the platform provides an arrangement of nodes that permits progressions to occur within the target's platform. These processes permit communication inside a robotic architecture.

The architecture of the framework comprises of five components, which are: a Master, nodes, publishers, subscribers, and topics.

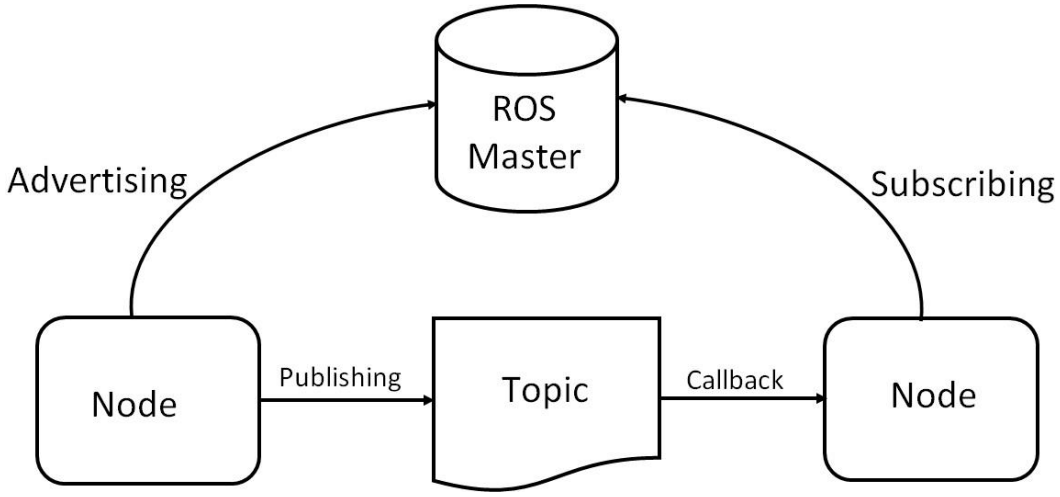

Figure 21-ROS Workflow

2.1.5.2 ROS Master:

The Master handles names and registration of the nodes within a system. Publishers and subscribers are monitored by the Master to make sure only associated topics are received. Moreover, it enables communication between nodes by means of the roscore command.

Node:

An internal ROS system executable file, which establishes a communication gateway between ROS, objects.

Publisher:

Any message transmitted by a node of an ROS system is defined as a publisher.

Subscriber:

A communication received and extracted by a node is identified as a subscriber.

Topic:

A topic is the process of sending and receiving of a message of a single name type.

2.1.5.3 ROS Distributions:

A ROS distribution contain a selected version of ROS packages. These are somewhat similar to the various Linux distributions (for example Ubuntu, Fedora, Debian). The ROS distribution allows a user to work with a relatively stable framework until a more stable version is launched or when required08000000000000.

Table 1-ROS Distribution [1]

Distribution	Release Data	Logo / Symbol	End of Life
ROS Noetic Ninjemys	May 23rd, 2020		May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018		May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017		May, 2019
ROS Kinetic Kame	May 23rd, 2016		April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015		May, 2017
ROS Indigo Igloo	July 22nd, 2014		April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013		May, 2015
ROS Groovy Galapagos	December 31, 2012		July, 2014
ROS Fuerte Turtle	April 23, 2012		--
ROS Electric Emys	August 30, 2011		--
ROS Diamondback	March 2, 2011		--
ROS C Turtle	August 2, 2010		--
ROS Box Turtle	March 2, 2010		--

2.1.5.4 ROS language(s)

ROS is mainly used side by side with C++ and Python. Those are currently one of the most popular pair of languages to work with robotics. There are also many pre-built libraries to communicate with other languages, some of which are rosjava, roslibjs and rosnodejs.

ROS is language dubious, meaning that the subprograms that we write can be programmed through any other language. An application developed through ROS can thus have a node developed in Python sending/receiving messages to a node developed in C++.

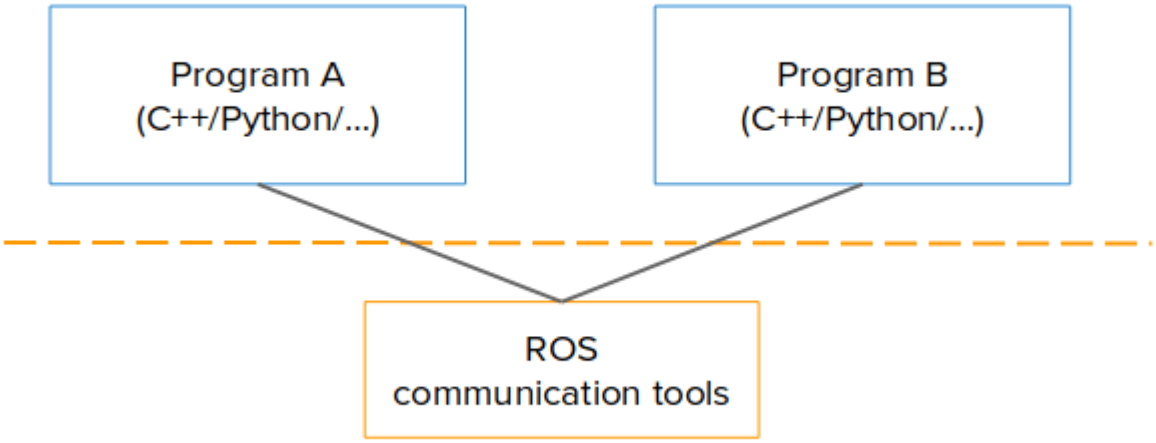

Figure 22

2.1.5.5 ROS libraries

ROS is much more than a simple communication tool for hardware. With it being open source, many libraries have been created by enthusiasts that can be used for multiple purposes. Some of these libraries are:

- roscpp,
- rospy,
- roscs,
- roseus.
- rosgo,
- roshask.
- rosjava.
- rosnodejs,
- and many more, etc

2.1.5.6 MoveIt

MoveIt is an innovative software for making use of the most recent features of motion planning, 3D manipulation and perception, kinematics and more. It gives a simple to-utilize stage for creating advanced applications, assessing modern robot architecture and building incorporated robotics products for multiple purposes.

Figure 23

MoveIt is based as a plugin of ROS. It is a prime source of the functionality for management in ROS. MoveIt builds on the ROS communication packages and makes use of some of the features in ROS like Rviz.

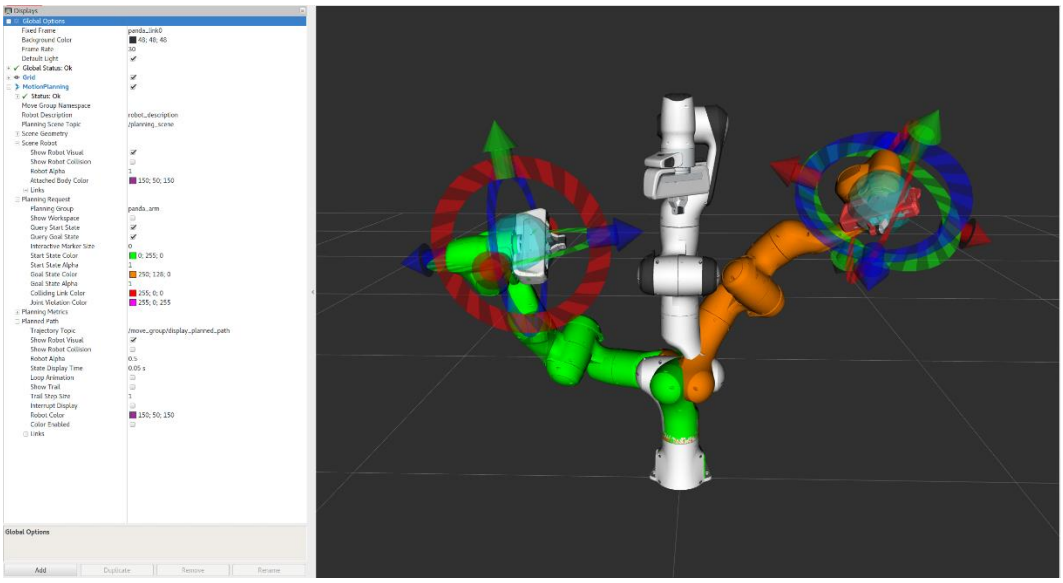

Figure 24-Moveit! Simulation

The speediest method to begin utilizing MoveIt! is through its RViz plugin. Rviz is the essential visualizer in ROS and an unfathomably helpful instrument for debugging robotics. The MoveIt! Rviz plugin permits the user to setup virtual environment such as scenes, and then create start and output states for the robot interactively, test different motion planners, and visualize the output.

2.1.5.7 URDF

The Unified Robotic Description Format (URDF) is based on the XML format, which is used in ROS to design all parameters of a robot. To model/simulate a URDF design in Gazebo/Rviz, some supplementary/specific tags must be set for it to perform properly.

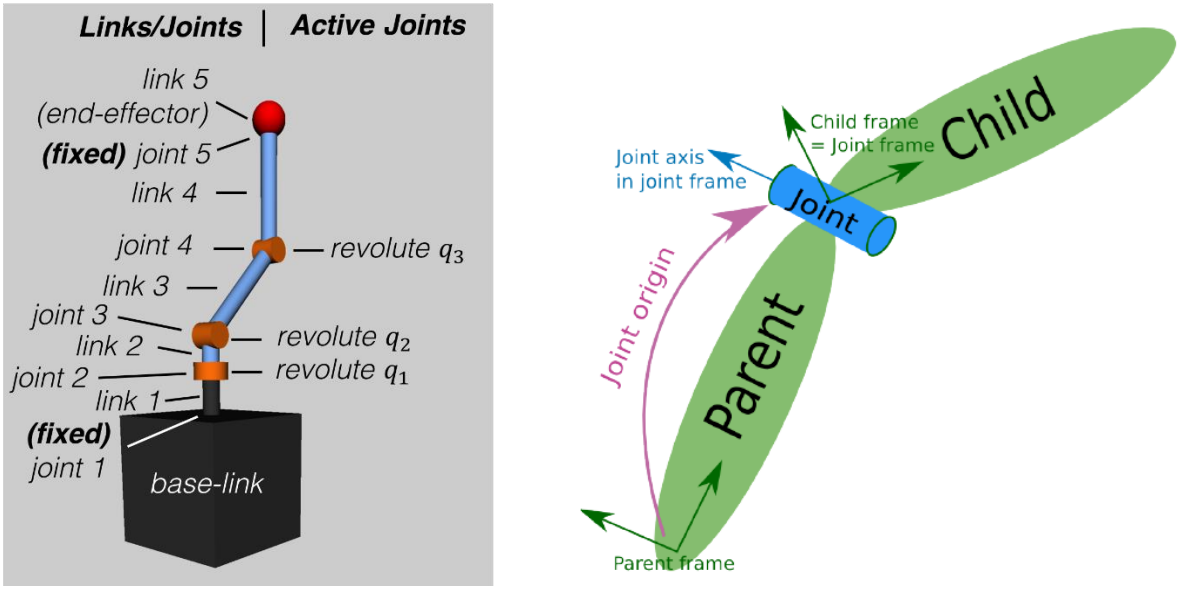

Figure 25-URDF Joints

Some of the tags used in the creation of URDF file are as following:

- Link : This element depicts a body with an inertia which has a rigid structure and describes visual and collision properties.
- Origin : This is the pose of the inertial reference frame, relative to the link reference frame.
    - rpy : Depicts the three stationary principle axis angles in radians.
    - xyz : Represents the axis and plane
- Mass : The mass of the joint is characterized by the feature of this component
- Material : The material of the visual element.
- Joint : Intermediary between two links.
- Cylinder : The origin lies in the center of the cylinder.
- Box : The origin of the box is in its center.
- Inertia : The 3x3 rotational inertia matrix, denoted in the inertia frame.
- Geometry : The shape of the visual object

2.1.5.8 Rviz

Rviz, a 3D representation software for applications centered on ROS. It gives a perspective on your robot design, catches sensor data from sensors, and replays acquired data. It can show information from any 3D and 2D peripheral devices such as digital cameras and depth sensors.

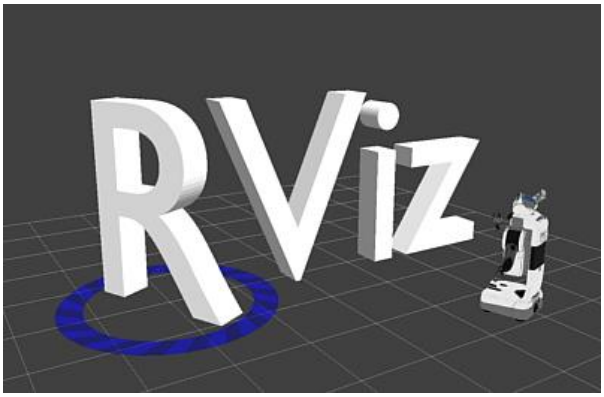

Figure 26

2.1.5.9 Gazeebo

Similar to Rviz, we also have another simulation software in ROS, which is known as Gazeebo. As a robotic simulator that is closely integrated with ROS, Gazebo is a heavy-duty virtual developer with a wide range of designs and plugins that help in communicating with a large number of sensors that are available on the market. The data collected by the sensors can be read by ROS through certain modules.

Gazebo utilizes full packages, which provide complete compatibility with ROS while developing a 3D environment. The majority of the advanced robotics simulation software are expensive and subscription based; if one cannot afford it, Gazebo is certainly the best choice.

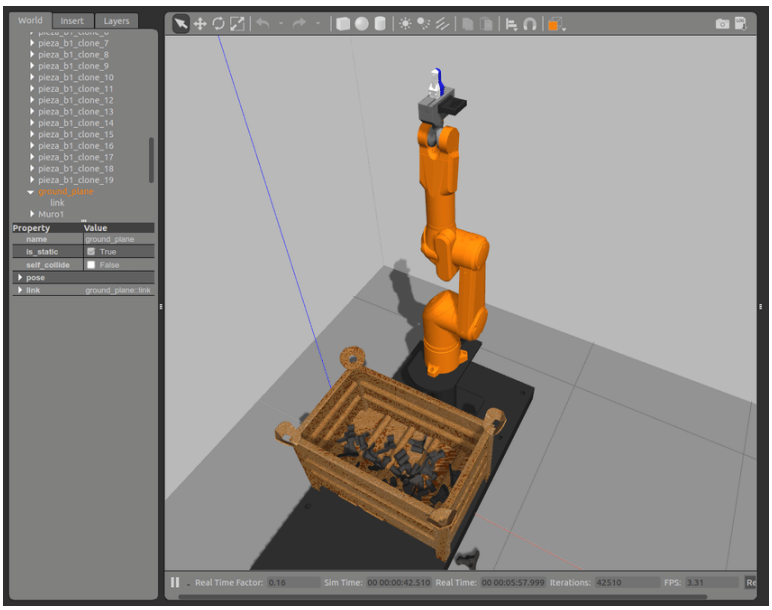

Figure 27-Rviz Simulation

2.2 Prior Work and its Relationship to Our Project

[2] A Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos :

In this paper the author describes the three background-modeling methods available in OpenCV, a free and open computer vision library that he could be used for the purpose of subtracting background from an image. The analyzed methods are GMG, MOG, and MOG2, in their respective Python implementation. The author chooses Python language considering it as an easy-to-learn programming language, with codes that are easy to understand. The

In our Project, we have used MOG2 because the results we received from it were satisfying and it is the latest background subtraction algorithm. MOG2 was developed with the objective to solve one of the limitations that MOG had; that is the fixed amount of used distributions. Initially we tried using MOG instead of MOG2, but the saturation of noise across the boundaries of the objects were pixelated and slightly more. Hence we opted for MOG2 as a Background Subtraction algorithm.

[3] Object Detection and Recognition in Images;

The author in the paper describes Object Recognition to be problematic task in the field of computer vision. Many methodologies have been researched upon in the past however no model developed has both qualities, being fast and reliable. The author focuses on EasyNet, which is a machine learning CNN model. This model goes through each image while testing so its results have greater accuracy. At the time of giving a prediction, this algorithm gives scores if an object of a particular category is detected. For this model, the problem is solved through regression for separating bounding boxes and probabilities of category classification.

The input image with a fixed size of 416 pixels squared is given to the CNN that alters some settings to output a 7x7x30 tensor, each tensor depicting a bounding box for segregated cells.

However, in our project we have used AlexNet that is a CNN based on 8 layers. The network has an image input size of 227-by-227. The first **Conv** Layer of AlexNet encompasses 96 kernels of size 11x11x3.

The difference here is usage of two different CNN based models and algorithm that have different sizes of inputs and outputs.

[4]Object Detection with Deep Learning: A Review ; Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE :

In this paper, the author provides a report on a DL framework for object detection. His review starts by introducing history of deep learning and Convolutional Neural Networks (CNN). After describing the pre-existing object detection models, some tips were mentioned on how to improve the performance further.

As detection tasks with separate classification categories show different features, the author describes several tasks, including salient object detection and facial mask detection. He Provides with results to compare multiple methodologies all the while coming to a conclusion. Various methodologies were described to create guidelines for any future research in object detection systems.

This report provides a detailed review on deep learning based object detection frameworks which handle different issues some of which are low resolution, clutter of objects, with different variations of R-CNN. The introduction includes information about generic object detection algorithms which are used as core architectures for various projects. Salient object detection, face detection and pedestrian detection are also described.

In our Project we use CNN for the purpose of Object Recognition only, whereas in this paper the author applies this deep learning approach to face detection as-well as pedestrian detection.

[5]Computer Vision Based Object Grasping 6DoF Robotic Arm Using Pi-camera

This paper presents an idea of a design for a robotic arm to perform multifunctional tasks. An Arduino mega 2560 Microcontroller has been selected to configure a controller to manipulate the arm on all axes, as it will be used to pick and drop objects at a market location. A high precision motion control with digital electric motors are required to drive the system. Further testing was performed to attach a digital camera with a computer vision framework to recognize object and locate them for grasping. The 3D simulation environment had a feature to detect objects and their position from the grasping claw. The core system was based on a Raspberry pi microcontroller for processing of the data collected through the camera, which allowed the whole system to work in parallel without any unnecessary and allowing pi to send commands to the controller in real time.

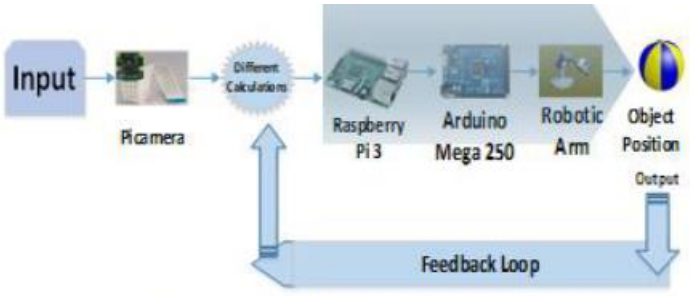

Figure 1. Block diagram of the system.

Figure 28

For this project, the author used a 6 DOF articulated robotic arm and a 3D visual optics system. A multifunctional robotic arm was developed with computer vision at its core. Its performance was

evaluated by creating a focal point of all axes of the manipulator on an object and drop it at a marked location.

A 3D simulation environment based on a digital camera and a computer vision algorithm was developed to recognize objects and their coordinates. The 3-D visualization system had features which were programmed to identify objects and their location and send them to the controller.

In our project, we have used Kinect's depth sensor to feed our MOG2 background subtraction algorithm that detects the objects in a frame. We have not used Rasberry pi 3 to control our robotic arm, instead we have used state of the art software like MoveIt! In ROS to implement the poses of our Robotic arm. Same as the paper, we have used Arduino for the controlling of the servo motors. We developed a power board to drive power to the servo motors.

[6]Robotic Grasping of Novel Objects using Vision

In this study, the developers took into account the issue of grasping novel objects through computer vision. Grasping an unknown object of which no parameters are known or 3D design is not specified is a unique as well as a baffling problem. Even in a scenario where a 3D model is known, the program will still have to identify the grasping location of the novel object. The idea that is given describes the creation of a self-learning algorithm requires no object parameters. With multiple images of an object given, the designed algorithm will try to identify/select multiple grasp locations on each object. This set of coordinates is then passed through a triangulation process to acquire a 3D coordinate for a proper grasp location. Using supervised learning, the selected algorithm is trained to identify positive grasp locations using synthetic images for training. They go along this methodology using a pair of processing platforms. The developed algorithm was successful in identifying positive grasp locations on most novel objects, be it an irregular shaped object or regular. An application for this algorithm is of unloading/loading objects to/from dishwashers.

Objects Similar to Ones Trained on

Tested on	Mean Absolute Error (cm)	Grasp-Success rate
Mugs	2.4	75%
Pens	0.9	100%
Wine Glass	1.2	100%
Books	2.9	75%
Eraser/ Cellphone	1.6	100%
Overall	1.80	90.0%

Novel Objects

Tested on	Mean Absolute Error (cm)	Grasp-Success rate
Stapler	1.9	90%
Duct Tape	1.8	100%
Keys	1.0	100%
Markers/Screwdriver	1.1	100%
Toothbrush/Cutter	1.1	100%
Jug	1.7	75%
Translucent Box	3.1	75%
Powerhorn	3.6	50%
Coiled Wire	1.4	100%
Overall	1.86	87.8%

Figure 29

In our project, we have used the same methodology for as used by the authors of this paper. We created our own dataset containing images of different types of objects. Then we Re-trained AlexNet in this training dataset for the purpose of object recognition.

[7]Controlling Robotic Arm with a Vision Based Economic System

In the mentioned report, the author presented a unique solution for the task of controlling a robotic arm. The framework makes use of 3D models of objects to create a vast amount of synthetic data to train a computer vision algorithm which will later be tested on real life images of the selected objects. By going with a semi-supervised methodology, the author takes into consideration the geometric properties of each object. The algorithm trained gives a satisfying result and using the same algorithm, 3D poses were developed for the robot to follow in the virtual 3D environment. Using reinforcement learning, a computer vision based manipulator was also designed, which will be used in the real world. Taking into account this approach, the same algorithm could also work with other rigid body dynamic systems.

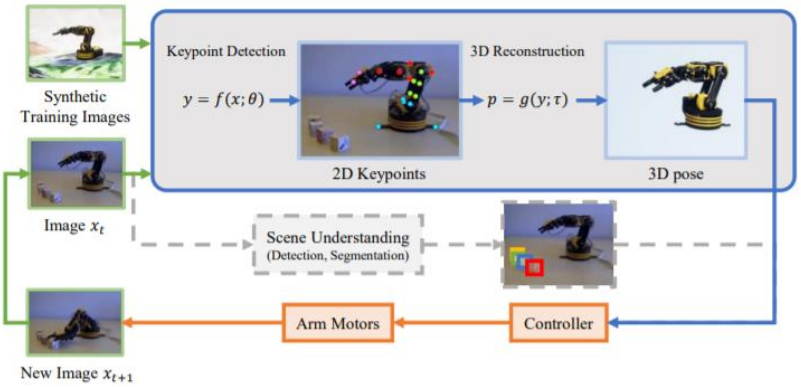

Figure 30

The author of this paper developed a system with the core idea of computer vision. The system was used to manipulate a simple robotic arm without any sensor feedback to perform selected tasks. A semi-supervised algorithm was used with data that was synthetically created with labels and real data for the training process. The constraints of the robotic arm, which were due to geometric properties, were used for domain adaptation. This framework, which uses 3D models to train the algorithm, could also be used for other rigid-body systems. Using two approaches to collect data, one is creating synthetic data and the other is collecting it from videos, all of which is used to test the algorithm for 2D point detection as well as pose estimation. The lower cost of the project will allow many researchers to study in-depth robotic manipulation using any type of machine learning and this will enable them to choose from various future directions. Educational institutes could also implement this as lab work for students at low cost.

In our project, we have not used the methodology mentioned in this paper that is the 3D modelling. Instead of this approach, we went for Computer vision with the help of Kinect's Depth Camera to track the Objects position if an environment.

[8]**A Review of Detection and Tracking of Object from Image and Video Sequences; Mukesh Tiwari, Dr. Rakesh Singhai**

The author describes in this report the various techniues used for object detection frameworks and the multiple segmentation methodologies used to work with video frames. This paper identifies and discusses the future scope of the methodologies described and the specific constraints. After working with multiple frameworks, the author came to know that some of the frameworks have greater accuracy but are more complex while some have lower accuracy but are much simpler.

The author's goal was to create a relationship between shapes and objects in the connected video frames, which leads to show that object tracking cannot be performed without object detection/classification.

In our project, using Kinect camera with our algorithms containing Opencv and Background subtraction algorithm we conduct the live tracking of objects in a space. We also use vitual grid in the video output to segment the video frame or space in multiple parts that help our algorithm to easily location and track objects.

[9] **Adaptation of System Configuration under the Robot Operating System; Jonathan M. Aitken, Sandor M. Veres, Mark Judge**

The author of this paper used ROS as the core operating system to design a reconfigurable manipulation system to control rigid body systems, in this case an autonomous robots. For any ROS framework, various libraries have been introduced. Multiple libraries for robot components were taken into account to design a system where it will try to maintain full functionality even in the case of a hardware malfunction. Tools such as Planning Domain Definition Language could be used to create multiple configuration settings. The author worked on a system with dual robotic arms where the multiple configuration settings would be used to complete a task at full capacity

.In our Project, we also used ROS for creating Automated movements of our Robotic Arm. ROS, while being a flexible framework can be used to write complex robot frameworks. ROS is a full framework with various libraries and plugins that can be used to lower the difficulty of creating complex robot manipulators. We use ROS to develop a simulation model that could be replicated on the real hardware to create the Robotic Arm poses that are needed to be performed in order to move the joints, grasp objects and then place the objects.

Table 2-Relation b/w Project & Prior Work Overview

Research Papers	Main Idea	Distinguishing factors from our Project	Inspiration

[2] **A Comparison between Background Modelling Methods.**	The author Compares methods such as GMG, MOG and MOG2 for Background Subtraction.		We choose MOG2 subtraction algorithm because in this paper, MOG2 gives noble results.
[3] **Object Detection and Recognition in Images**	The author focuses on EasyNet, which is a machine learning CNN model for the Task of Object Recognition	In our project we use AlexNet, which also a machine learning CNN model for the task of Object Recognition	
[4] **Object Detection with Deep Learning**	The author uses R-CNN for tasks such as object detection, face detection and pedestrian detection.	We use CNN for the task of object recognition.	
[5] **Computer Vision Based Object Grasping 6DoF Robotic Arm Using Pi-camera**	In this paper the author uses 6-DOF Robotic Arm with 3-D vision, Raspberry Pi 3 and Arduino.	In our Project we use 6-DOF Robotic arm with Computer vision and Robot Operating System.	
[6] **Robotic Grasping of Novel Objects using Vision**	The authors present a learning algorithm that neither requires, nor tries to build, a 3-d model of the object. Given a dataset the algorithm locates the Position of the Object.		Our idea is exactly the same as defined in this paper, which is to move away from building 3-D models of the objects instead use Computer vision and Machine Learning.
[7]**Controlling Robotic Arm with a Vision-based Economic System**	In this paper, they present an alternative solution, which uses a 3D model to create a large number of synthetic data, trains a vision model in this virtual domain, and applies it to real-world images after domain adaptation.	In our Project, we went for Computer vision with the help of Kinect's Depth Camera to track the Objects position if an environment. We use ROS to make certain Poses such as Gripping, Picking or Dropping.	
[8] **A Review of Detection and Tracking of Object from Image and Video Sequences**	This paper gives review on different object detection, tracking, recognition techniques, feature descriptors and segmentation method which is based on the video frame and various tracking technologies.		We utilize this study and comparison of different approaches. This paper helped us in learning concepts and factors related to Object Tracking.

[9] **Adaptation of System Configuration under the Robot Operating System**	This paper lays down the foundations of developing a reconfigurable control system within the Robot Operating System (ROS) for autonomous robots.		We use ROS to develop a simulation model that could be replicated on the real hardware to create the Robotic Arm poses that are needed to be performed in order to move the joints, grasp objects and then place the objects.

Chapter 3

3 Project Methodology

In the project “Synthetic Data production for Machine Learning Using Robotic Arm”, our primary aim is to allow the computer or machine to learn automatically from the given data without human interference and perform tasks accordingly. Our goal moves around the process of machine learning and training of model. Machine learning models will be implemented on Robotic arm in order to grasp object with different shapes, dimensions, sizes and weight. For this, we gather synthetic data randomly from the real world and by using the data we trained the machine and make it learn. After Machine Learning part, we use robotic arm to grasp novel objects to move it from one position to another. This whole process is categorize in steps.

3.1 Major Tasks

Our robot will train itself using self-learning algorithms and create a database for all the positive points that it will gather from the grasping hit and trial. At least 100 positive grasp locations for unique objects will be collected and stored in the database. Customized scalable intelligent grasping robot will be developed. We can increase the scale of the robot which means we can handle bigger payload and use the same datasets to interact with bigger objects. Our final outcome will be a self-learning grasping robot which does not need human input to identify, pick and drop novel objects.

Detailed discussion has been deliberated below about the methodology we have adopted during the project.

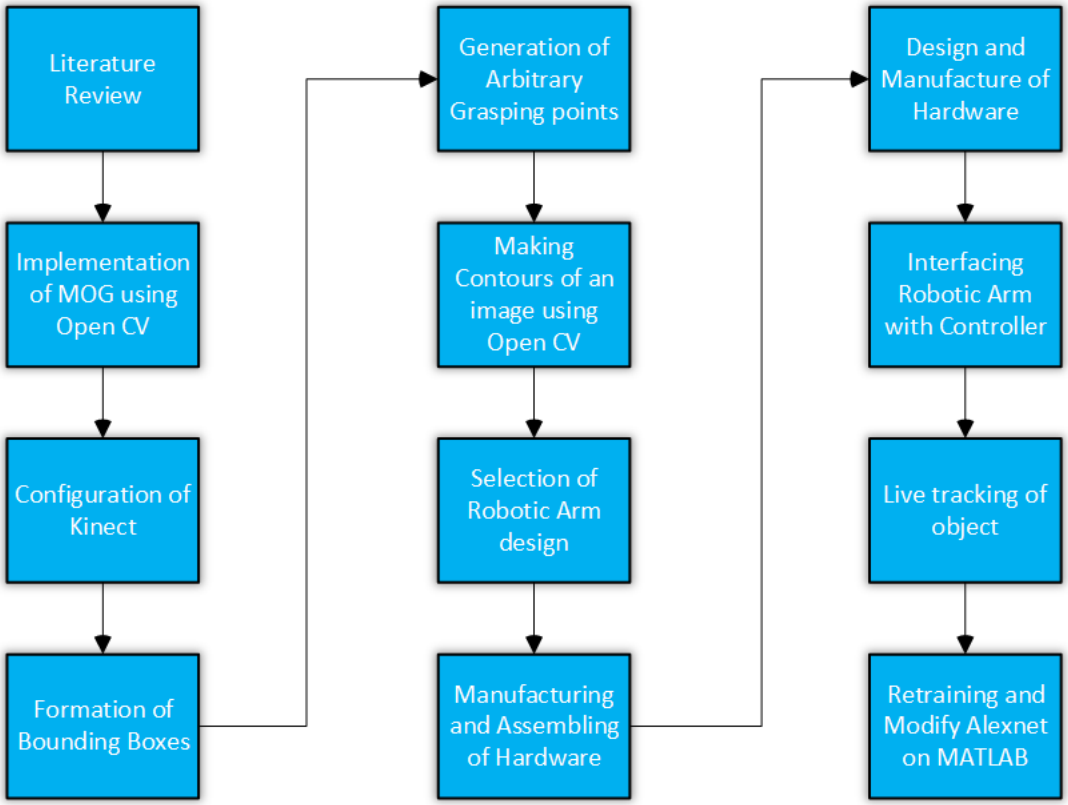

Figure 31-Major Tasks

3.1.1. Literature Review

We had defined our approach by first reviewing research papers and set the guidelines and milestones. Initially we started reviewing research papers and past projects done related to our project. Group discussions with co-supervisor was done on weekly basis, in which key points of research papers were deliberated. Core structure of project and timeline was made which had to be followed throughout the project.

In this project, we break the trend of using manually labeled grasp datasets for training grasping robots as such an approach is not scalable. Instead, inspiration taken from human experimental learning, we present a self-supervising algorithm that learns to predict grasp locations and creates a dataset with both positive and negative points via trial and error.

Our project includes working on an algorithm for identifying or recognizing objects present in an image, MOG subtraction, finding center of mass of each object and selecting random arbitrary points as grasp locations, exact position of object using live tracking. The algorithm will also give output in the form of image patches of a specific size, which will be used to retrain an Alex-Net CNN. The retrained Alex-Net CNN will allow us to choose the best grasping point for each object, relative to all other points selected for the object. When the grasp location will be selected, by using a grid system for our area of contact, robotic arm will move at that place and pick the object using jaws of the arm.

3.1.2. Implementation of MOG using Open CV

MOG uses pixel differentiation to identify new objects and using **CONTOURS**, which convert image into binary form, as a result pixels which were different from our given background will be obtained. Background subtraction is a popular method for isolating the moving parts of a scene by segmenting

it into background and foreground. The Fundamental logic for detecting moving objects is to find the difference between the background and the foreground. This method is known as “Frame Difference Method”.

OPEN CV library was used with an RGB camera to process images. By using OPEN CV Gaussian background subtraction methodology has adopted to remove the background from our images and get the separate image of our objects. We provided the Gaussian background subtraction algorithm with a static image of our background, without any objects and a camera live feed with different objects placed in front of our camera.

The problems we faced in using OPEN CV were:

- Due to **varying light conditions**, we were getting errors in identifying objects from the background
- If any object had some **matching contrasts** with our background, open cv failed to differentiate the object from our background

3.1.3. Configuration of Kinect

We use XBOX 360 KINECT instead of an RGB. The XBOX 360 KINECT has 3 different cameras, an RGB camera, a DEPTH sensing camera and an IR sensor.

With an RGB camera, while detecting objects which were similar in contrast to the background, the algorithm failed to differentiate between the background and foreground. While with a Kinect V1, the depth sensor allowed us to detect objects with greater precision as the objects with some mass were shown with a different shade of grey while the background with a different shade.

3.1.4. Formation of Bounding Boxes around each Object

Bounding boxes around each object was made in order to define or enclosed the object in a specified boundary. Bounding boxes help us to identify the exact dimensions of the object and can be easily extracted from the image with its position.

3.1.5. Generation of Arbitrary Grasping Points

Now selecting **generic point,** as we have to find the best grasping point of an object, relative to all the points, we first selected 5 points on each object. These points were selected on the basis of the particular object’s top boundary and then the points were selected equidistant to each other between the two. The points are shown on the picture in red color so as to distinguish the points from the noise. Then **center of mass** for each object was found and marked. As we know, the best grasping point for most objects is the center of mass as at that location all the forces are in a natural equilibrium.

3.1.6. Making Contours of an Image using Open CV

Afterwards **patch selection** for each grasping point. A patch of size 227x227 was made, with the grasping point at center. These patches will be used to train our model to select the best grasping point out of all points, taking in consideration some rules and measures regarding how to identify the best grasping point.

3.1.7. Selection of Robotic Arms Design

We have selected our Robotic Arm. It has been purchased from the e-commerce website (www.ewall.com). Our Robotic Arm has 6 degree of freedom. It includes 6 servo motors and a griper for gripping objects.

We prefer 6 DOF Robot Arm because it gives us multidimensional moves of the arm and it best suits our obligation. As according to the requirements, the robotic arm of our selection should be versatile, have great precision and have a 360-Deg scope of movement. With a standard aluminum bracket and 6-DOF tolerable manipulation, the arm can be fixed to any flat platform, saving a lot of complicated structures. With a standard claw opening width of 5.5 cm, the gripping claw can be opened to a maximum of 9.5 cm approximately. As our objects in question will be mini sized, the robotic arm selected fits well according to our requirements.

For selection of hardware, we ensure that robot must be reliable and perform adequately as per requirement. We tried our best to purchase the robot which best fits on our constraint and works perfectly fine as preferred and also have a minimum cost as possible.

3.1.8. Manufacturing and Assembling of Hardware

Robotic arm was made up of aluminum material. Aluminum is light weight and solid in nature. It reduces the overall weight of the structure. Robotic arm was assembled using the nuts and bolts to be in a proper shape.

Following are the features of the servo motor which we used in our robotic arm:

Model: Towerpro Servos MG995

Control System: +Pulse Width Control

Working Frequency: 20ms period/50hz (Analog Control)

(RX) Required Pulse: 3.0 ~ 5 Volt Peak to Peak Square Wave

Working Voltage: 4.8 ~ 6 V DC Volts

Working Temperature Range: -0 to + 55 Degree C

Working Speed (4.8v): 0.200 sec/60° degrees at no heap

Working Speed (6v) 0.160 sec/60° degrees at no heap

Slow down Torque (6v): 11kg/cm

Engine Type: Brushed Motor

Bearing Type: Output Bearing

Apparatus Type: Brass and Aluminum Gears

Case Material: Plastic

Dimensions: 40.7×19.7×42.9mm

Weight: 55 grams (2.64 oz)

3.1.9. Design and Manufacture Power Board

In order to control the motors of servo motor, we made a Power Board which power up the motors in parallel. Initially I listed the number of outputs needed. Then I made a schematic of the Power Board on Proteus. Voltage and Current rating was strictly maintained at each terminal in order to handle bigger payloads. Proper and complete testing had been made on schematic. After that PCB Layout was designed in such a way that it took minimum space and all our components should be placed comfortably.

3.1.10. Interfacing Robotic Arm with the Controller

Arduino microcontroller was used. This microcontroller consists of a both physical programmable circuit board and an IDE that runs on computer. Arduino is user friendly and flexible enough for advanced as well as for beginners. It is compatible on different Operating Systems like Windows, Linux and MAC.

Robotic arm consist of 6 servo motors. Each motor had been interfaced with arduino microcontroller in order to evaluate the movements of motor, speed, torque etc. While interfacing robotic arm with controller we needed a Power board too. We collectively joined robotic arm motors, power board and controller. Programming had been done repeatedly to analyze the response and behavior of the machine or robotic arm.

3.1.11. Live tracking of Object

We had to operate these system with live object coming on the allocated grid of the system. So we adopt a methodology for live tracking of the object. In which we locate the object, gets its dimensions, size and position. When the new object came, we get its dimensions as well and command our robotic arm to reach at the object and grasp it accordingly using algorithms. As soon we get the position of the object our main object objective of live tracking had been done.

3.1.12. Retraining and Modifying AlexNet on MATLAB

We used MATLAB, to modify already existing CNN, Alex Net to identify and classify objects.

AlexNet is a neural network, which has been trained with a dataset of images counting more than a million. With eight layers, the network classifies images into 1000 labels, some of which are keyboard, box, stapler, and various fruits and vegetables. Due to this, AlexNet has been trained with rich feature representation for a wide variety of visual frames and has a default image size of 227x227.

Supervised Learning technique was used to train network. For this, different image of cups, crowbars, hammers, staplers, screwdrivers etc. has gathered and properly labeled. These images were used as input to our network.

Pre-trained Alex Net had fine-tuned to classify new collection of images. This process is called transfer learning and is usually much faster and easier than training a new network. Deep network designer was used to interactively prepare a network for transfer learning.

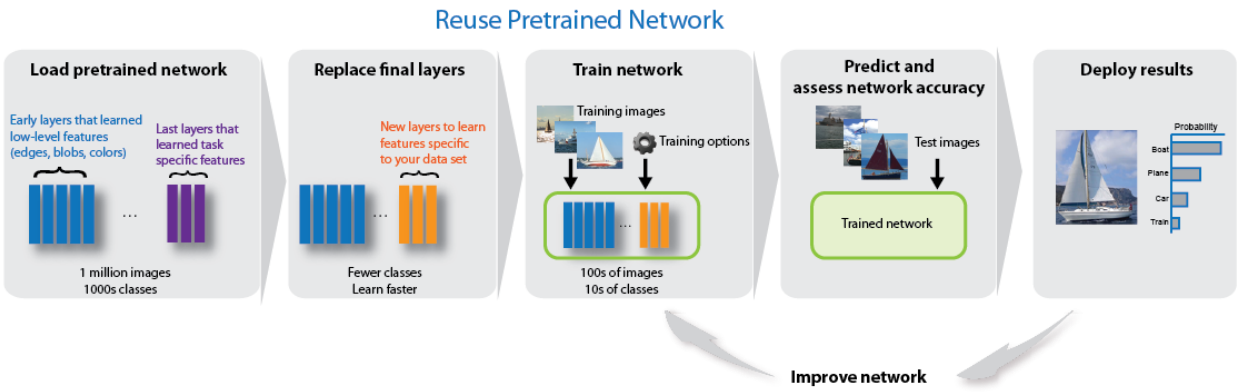

Figure 32-ML Model Structure

Data, which contain images, had to be divided in two parts i.e. Training data and testing data. Training data loaded into the network and hence trained according to the provided input. After that, we provide network the testing data and test our networks performance. This process continued until results became accurate and acceptable.

3.2 Gantt Chart

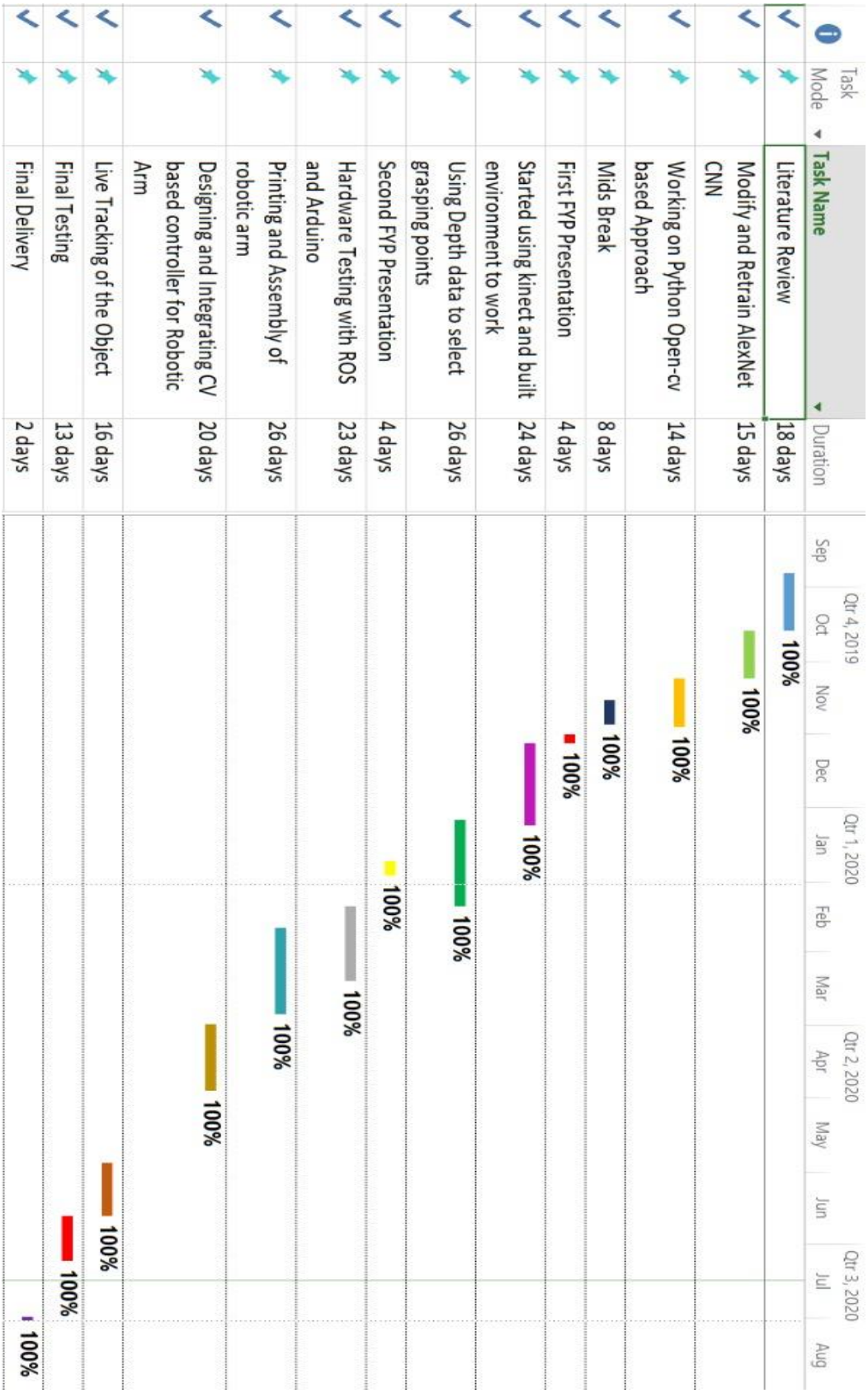

Figure 33-Gantt Chart

3.3 Work Distribution

In order to organize the work our team distributed the work in chunks in order to make it done within time. Our team includes three members.

- Hamza Liaqat
- Mustafa Imran
- Muneeb Babar

We follow our Gantt chart very firmly through the project. We categorize our work in two distinct parts hardware and software. At the start, we are only dealing with software, so initially we all are working on software part. As our project progress, Hamza and Muneeb handle the work regarding software while Mustafa managed the work regarding hardware.

3.4 Tools, Hardware, and Software Employed

In this section, we will go through the software and hardware, that we realize essential to deploy our project. Essential tools were selected keeping in mind the tasks which we want to perform. These tasks include:

- Object Detection
- Object Recognition
- MOG Subtraction
- Finding the best grasping point
- Live Detection and Position of the object
- Deployment of the project using Robotic Arm

We preferred Linux to Windows Operating system. As Linux is fast and more compatible than windows.

3.4.1. Software

We had selected software, after the analyzing the previous research and development. We tried to use the software which were user friendly and widely available and help us to perform the tasks in a comfortable zone. Following is the list of the software which we used in our project:

- Visual Studio Code
- PyCharm IDE
- MATLAB
- Kinect Studio
- Arduino IDE
- ROS and Moveit
- Proteus
- Rviz
- Gazeebo

We have done our major programming part in python. So for python programming we preferred to use VS Code and PyCharm IDE. As they were widely used in the industry and manage our work in an organized way.

Kinect Studio was used to interface Kinect with our PC in order to capture the image and for object detection.

ROS (Robot Operating System) and Moveit were used to control the movements of Robotic Arm. It assisted us to adjust the particular and exact movements of Robotic Arm to a specified place.

In our project we need a power board which will control our motors. Proteus had been used to make the schematic and PCB layout of Power Board. As Proteus was very compatible and user friendly in its functionality.

In order to check the detailed functionality and movement of motors Arduino IDE was used.

MATLAB was used for training our model and make it learn automatically from the random data assigned by us. ALEX Net was used to train the model make our machine learning from the given data.

3.4.2. Hardware

In order to detect the object we used a camera. As we need to detect the object's position with respect to the background, its distance from the camera and depth of the object. We have multiple options but we preferred Kinect. Kinect Camera has three sensors IR Sensor, RGB Sensor and Depth sensor.

There are several sensors (cameras) available in the market which can serve the purpose such as:

- The Mesa Imaging Swiss Gear 4000
- Cam Cube 2.0 from PMD Technologies
- Microsoft Kinect v1
- Microsoft Kinect v2

We had a wide variety of options however we used Microsoft Kinect V1 as it is easily available in Pakistan, has good community support and has libraries available allowing ease of usage.

3.5 Costing

In this section we will discuss about the total cost or expenditure of the project. Robotic arm that has 6 Degree of Freedom (DOF) had been ordered from the e-commerce store (www.ewall.com) for Rs 10,500. Robotic arm has 6 motors, in order to operate these motors we needed a Power Board. Essential components were also bought for the Power Board which includes ICs, T-Blocks, Wires, LED, Resistors, Fuses and Buttons. Components total cost was about Rs 1,000. Arduino Mega microcontroller was also needed to operate motors, it was cost about Rs 800.

Overall expenditure of the project has been around Rs 13,000.

S.No.	Equipments	Quantity	Price (Rs)
1	Robotic Arm	1	10,300
2	AT Mega 2560	1	800
3	PCB	1	200
4	LM 2596	2	1000
5	LM 7805	6	200
6	LM 7812	6	200
7	Push Buttons	6	60
		Total	**12,760**

Figure 34-Overall Expenditure

Chapter 4

4 Implementation Details

In this chapter, we discuss the details of the process with its steps that were performed for the implementation of this project. We have sub-divided the tasks relating to their broader category.

4.1 Image Processing

This phase of the project was performed initially in the project timeline. Image processing was performed as some operations on an image, in order to extract some useful information from it, which in our case was the Detection of an object, Recognition of an object and potential grasping points on the object in an image. These task were as following:

4.1.1. **Segregating Foreground and Background:**

In order for to identify objects in an image, we needed to provide our algorithm as an input; The Foreground and Background Frames. The algorithm then will subtract the background from the objects and the objects can be detected.

Figure 35-Foreground & Background

4.1.2. **Grey-Scaling for Threshold:**

In the second step, we converted the colored foreground and background images into Grey-Scale for the purpose of setting a Threshold. Our algorithm requires a black and white or greyscale image as an input to work.

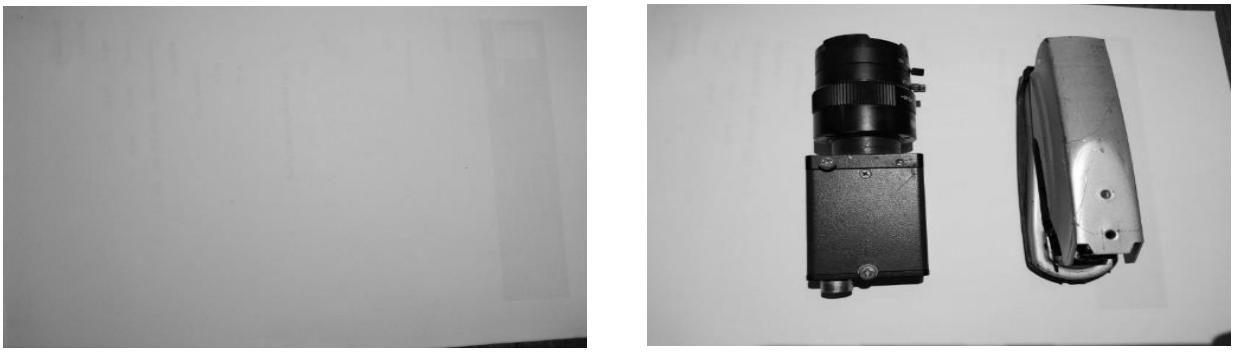

Figure 36- Foreground & Background Gray Scale

4.1.3. **Kinect v1 Depth Sensor**

We moved to using the Xbox 360's Kinect v1 for its depth sensor which provided more concise results and the output of the depth sensor was already in grey-scale. So there was no reason to manually convert into the grey-scale. Secondly We opted for Kinect v2's Depth Sensor instead of the standard RGB camera because it facilitated us in the live tracking of the object.

Figure 37-Kinect V1

4.1.4. **MOG Subtraction Algorithm:**

MOG Subtraction is a Gaussian Mixture based Background and Foreground Segmentation Algorithm. We have used MOG Subtraction Algorithm to Subtract the Background, So Objects can be segregated from this environment.

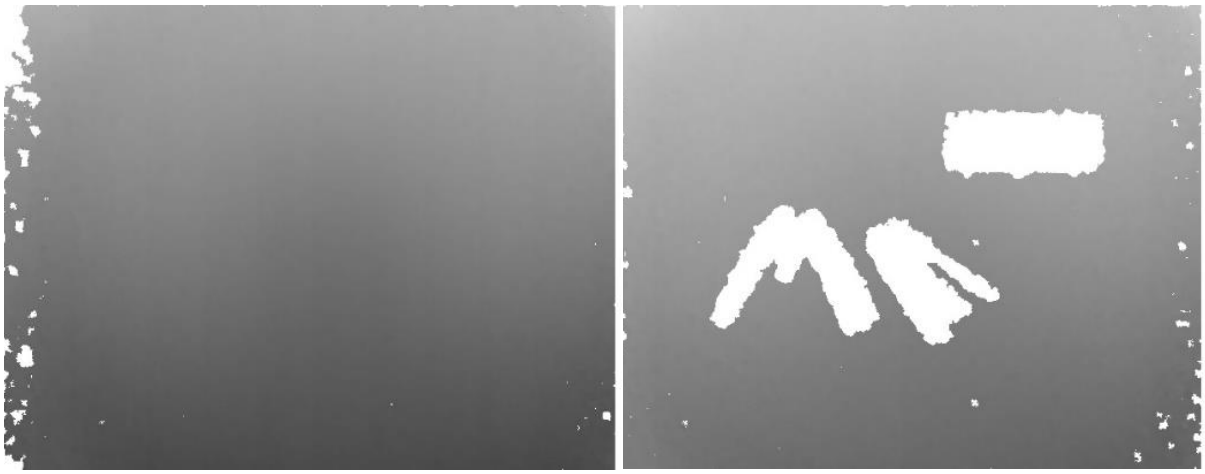

Figure 38- Foreground & Background Depth Data

4.1.5. **Object Detection:**

The Next step involved was detecting the objects. After we successfully subtracted the background from the foreground, the objects were visible in the output result. Now we had to declare the location of the objects.

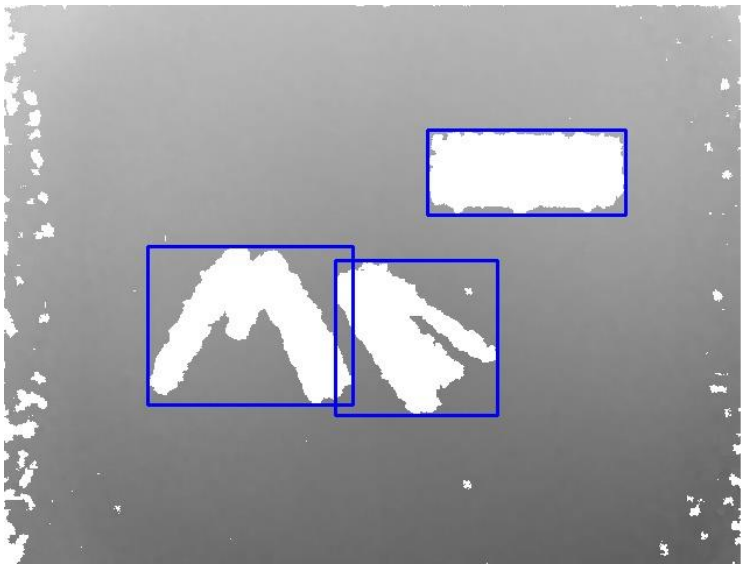

Figure 39-Object Detection and Marking

4.1.6. **Arbitrary Grasping Points:**

In this sub-task, we marked a few arbitrary potential grasping points of the object. These points were taken with respect to the coordinates starting from the top left corner of the object.

Figure 40-Arbitrary Grasping Points

4.1.7. **Centroid Point of an Object:**

As for most of the regular shaped objects have the center point as the best possible grasping point. Therefore, we identify and mark the center point of each object. The Blue point in the image shows the center point in the below figure.

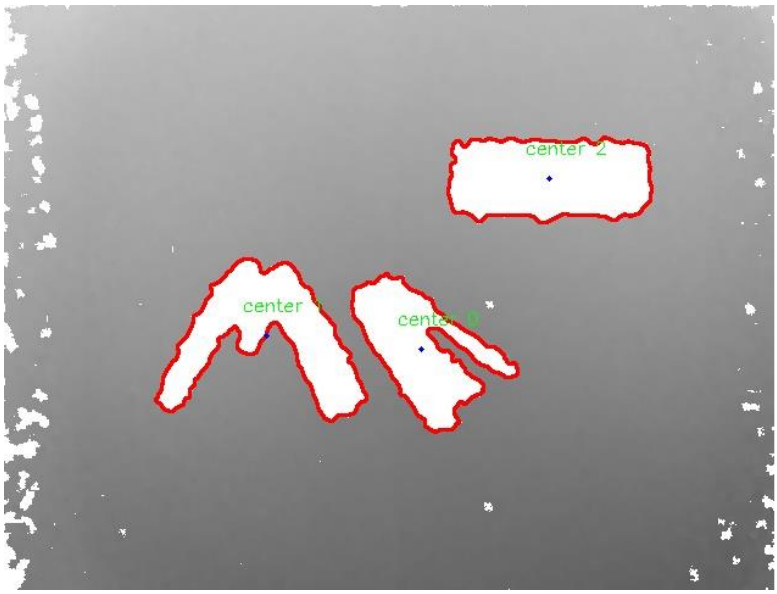

Figure 41-Centroid Point

4.1.8. **Patches/Frames w.r.t each Grasping Point:**

In this sub-task, we created bounding boxes for extracting that single cropped frame from the whole image with that particular grasping point in the center. This was done in order to give the Machine Learning model, the frames of the grasping point to train the Model.

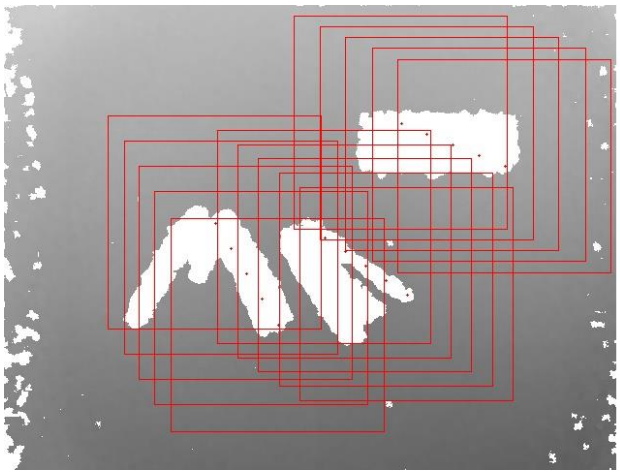

Figure 42-Patch Creation w.r.t each grasp location

4.1.9. **Formation of the Virtual Grid**

For the purpose of live tracking of the objects, we created a virtual grid using Python that would help the Grasping Robotic Arm to move to the location of the object much straightforwardly.

Figure 43-Virtual Grid

4.1.10. **Live Object Tracking**

Next task involved in live tracking of the object. Previous work was done using images or frames of video, However this live tracking moved our project into a more practical path. As the Robotic arm will use live tracking to detect the posiion of the object with the help of this vitual grid.

Figure 44-Live Object Tracking

4.2 Machine Learning:

This Machine Learning Part included the Recognition of the Object using Matlab's AlexNet Model.

4.2.1. **AlexNet Re-Training:**

This part included the Re-training of Matlab's AlexNet Machine Learning model. We first wrote the code for the re-training the model by providing the Model with multiple images.

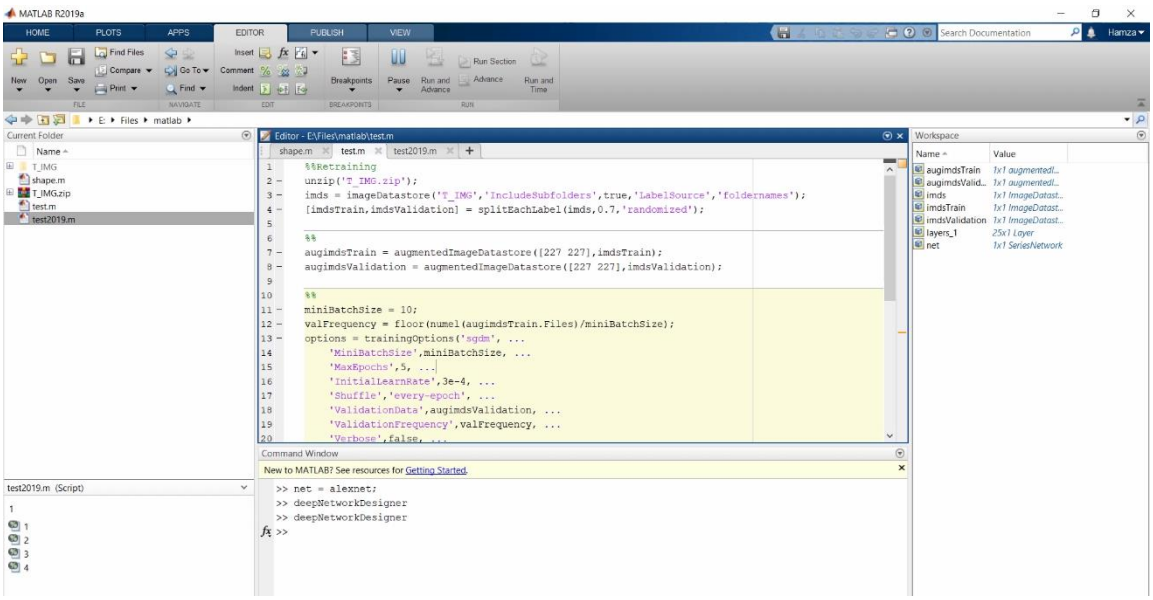

Figure 45-AlexNet Retraining

We then manipulated the layers of AlexNet by replacing the Convolution layer with a new one and altered a few settings for our convenience.

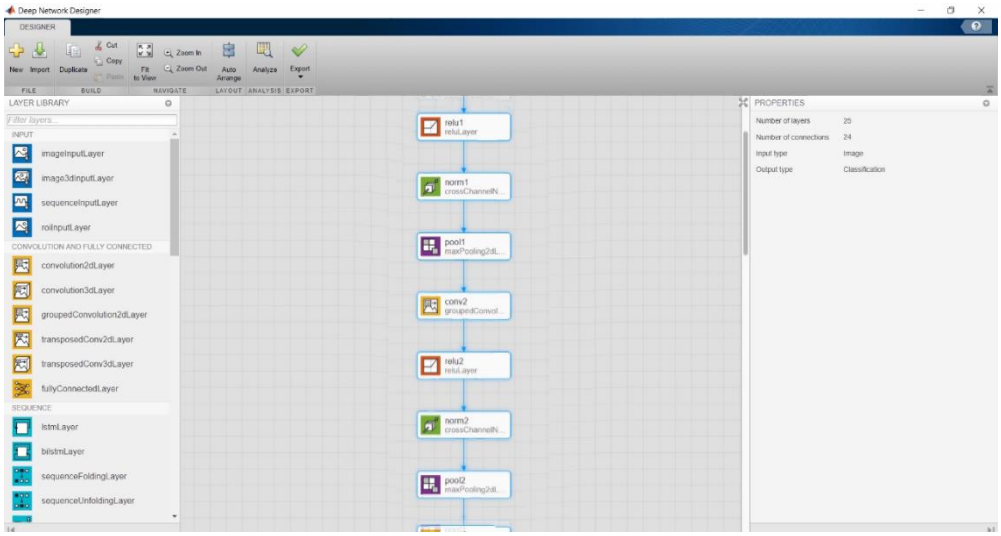

Figure 46-AlexNet Custom Layers

4.2.2. **Object Recognition:**

The purpose of using machine learning here was for the identificaion and recongnition of the objects that are presented in a frame. The output expected was that the Machine Learning model could tell us the name of the object placed after all the image processing operations were already perfomed as mentioned above.

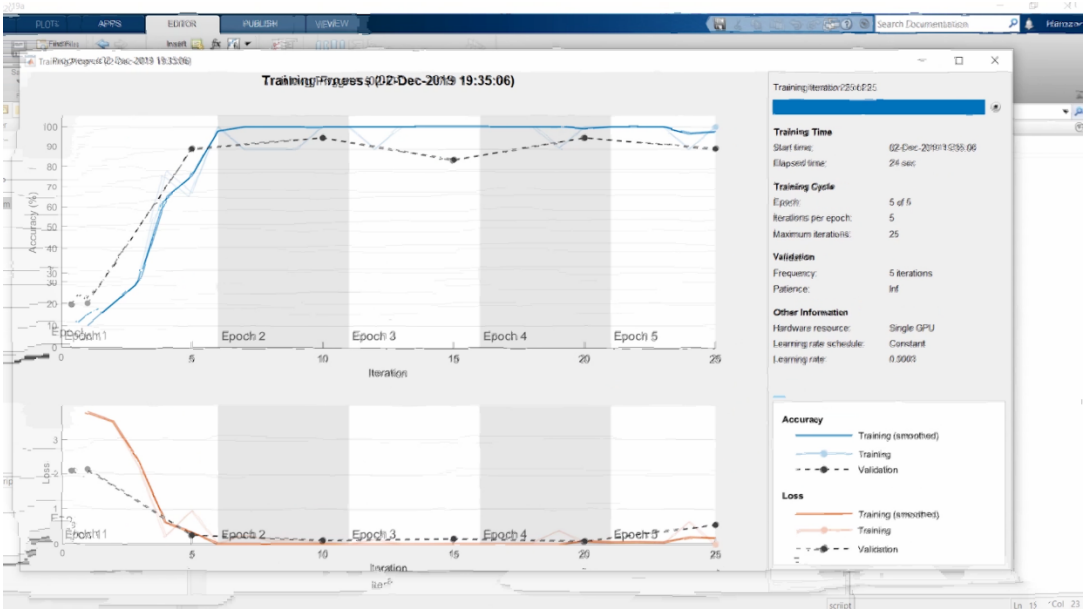

Figure 47-AlexNet Training Graph

The model was retrained and showed very precise and accurate results. In this way we were able to recognize the Objects that were placed for the testing purpose.

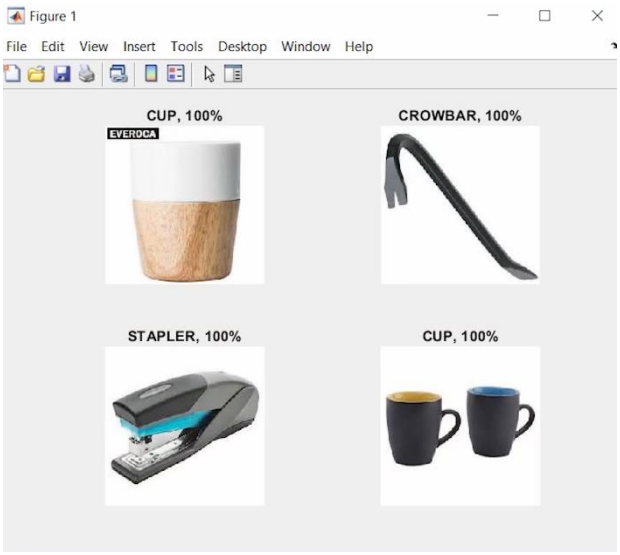

Figure 48-AlexNet Retraining Results

4.3 Hardware

After the image processing and machine learning part, we moved towards the hardware. The hardware tasks included:

4.3.1 Assembling the Robotic Arm

This task included in assembling the components such as Multifunctional brackets, Long U shape brackets, L shape brackets, Cup bearings, Screw nuts, Alloy mechanical claw, U- beam brackets, and Metal servo horn.

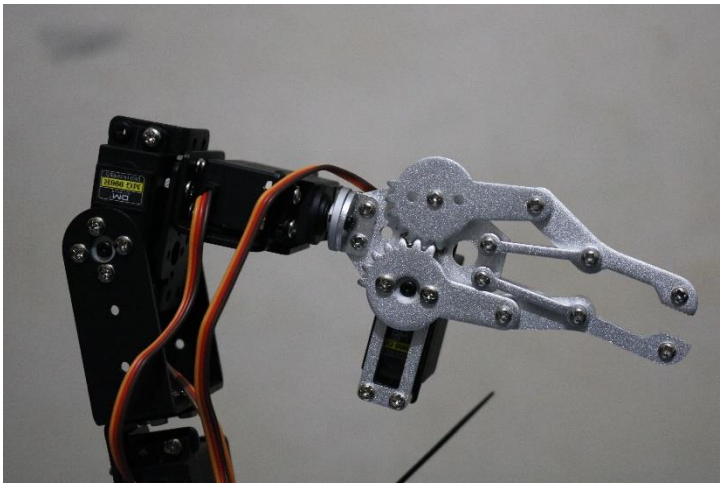

Figure 49-Robot Arm Claw

Some

Specifications of the Robotic Arm are as follows:

- Mechanical claw Material: 2mm Hard aluminum
- Mechanical claw Weight: about 68g (without motor)
- Mechanical claw maximum opening angle: 55mm
- The overall length of claw: 108mm
- The overall width of claw: 98mm

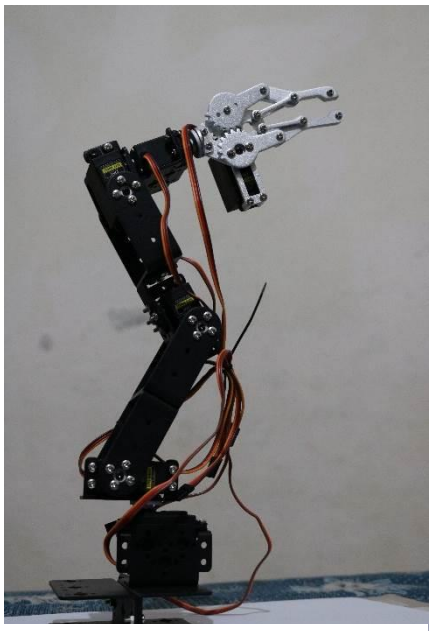

Figure 51-Robotic Arm Random Pose

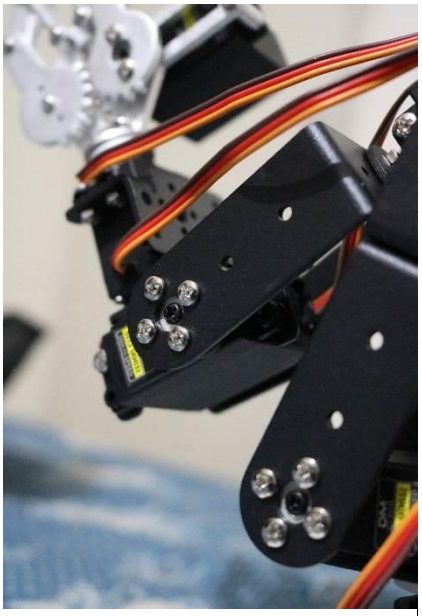

Figure 50-Robotic Arm Joints

4.3.2. **Power Board**

In order to power up the Servo motors, we designed the Schematic of the Power Board in Proteus and then manufactured its PCB board.

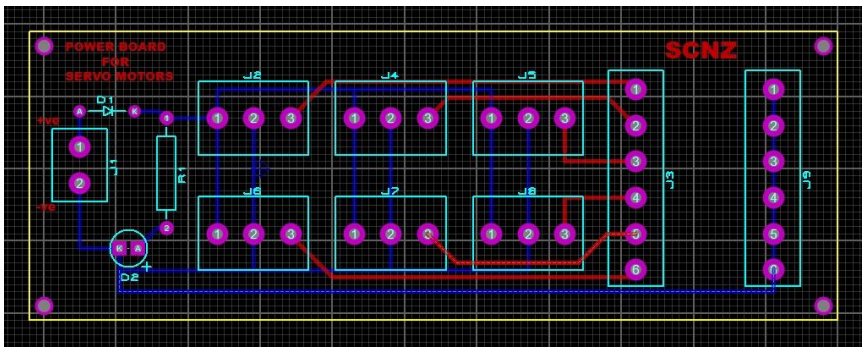

Figure 52-Power Board PCB Design

The Power board is Compromised of Voltage Regulator such as (7805 & 7812) , T-Blocks, Diodes, Resistors ,LEDs and Jumping wires.

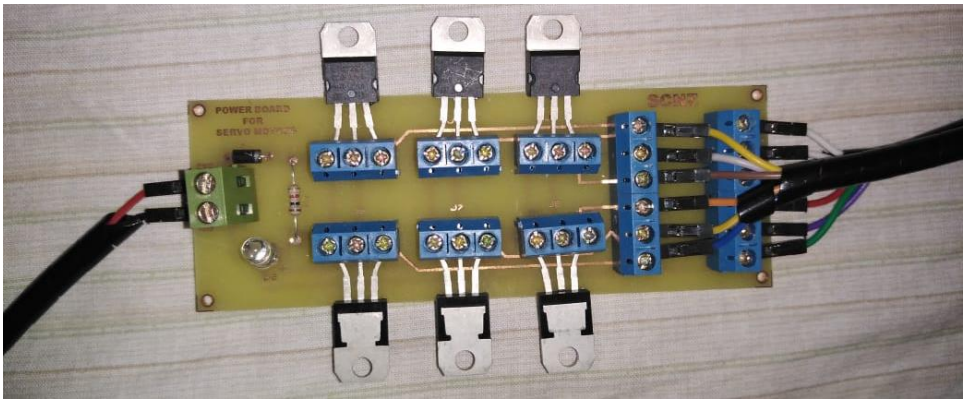

Figure 53-Power Board

4.3.3. **Integration with Power Board & Servo Motors and Testing:**

This task included subtasks such as Integrating all the hardware together, testing the servo motors, testing the poses such as grasping poses and testing the 6-DOF of the Robotic Arm.

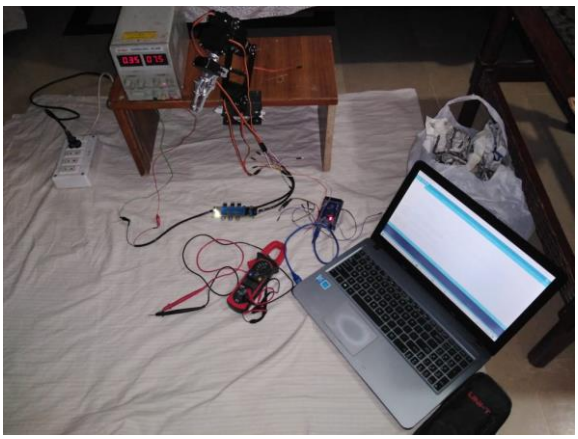

Figure 54

4.4 Robot Operating System (ROS)

We opted for ROS for the movement of the Robotic Arm instead of going for the traditional way of programming each pose manually using Arduino.

The Robot Operating System (ROS) is an advanced system that is used for creating robot manipulation software. It is a vast framework with numerous libraries and plugin that can be used to create complex robot manipulators across multiple robotic platforms.

We use ROS to develop a simulation model that could be replicated on the real hardware to create the Robotic Arm poses that are needed to be performed in order to move the joints, grasp objects and then place the objects.

The following steps were performed in this task:

4.4.1. **Setting Up Ubuntu Environment for ROS**

The environment we have setup for development is Ubuntu 18.04 LTS. The Distribution of ROS that is used is ROS Melodic Morenia that is the twelfth ROS distribution release.

This Melodic distribution is compatible with the latest Ubuntu 18.04 LTS, which is why we are using it.

```
sudo apt install ros-melodic-desktop-full

sudo rosdep init

rosdep update

echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc

source ~/.bashrc

CREATE ROS WORKSPACE

mkdir -p ~/catkin_ws/src

cd ~/catkin_ws/

catkin_make
```

Figure 55-ROS WorkSpace

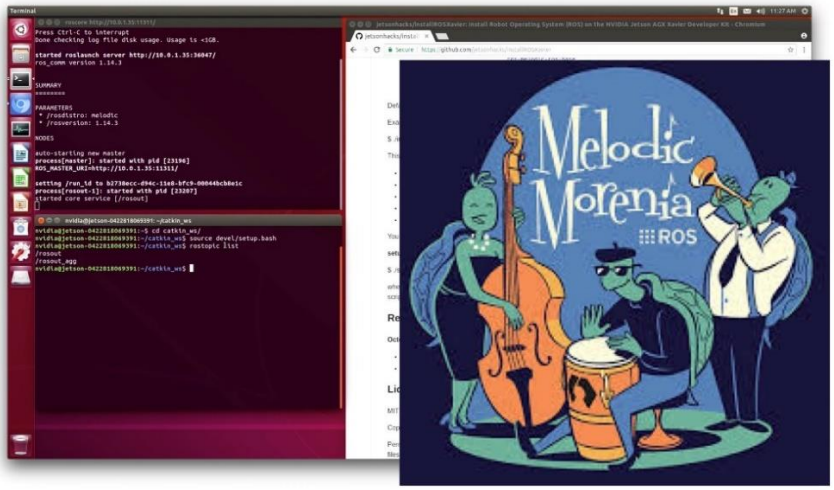

Figure 56

4.4.2. **Creating URDF File**

The Second step after setting up the environment in Ubuntu by installing Ubuntu and then ROS Melodic, we need to create a URDF file.

The Unified Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot.

```
<joint name="arm_joint_2" type="revolute">
  <parent link="arm_link_1"/>
  <child link="arm_link_2"/>
  <dynamics damping="3.0" friction="0.3"/>
  <limit effort="30.0" lower="-1.57079632679" upper="1.57079632679" velocity="5.0"/>
  <origin rpy="0 0 0" xyz="0 0 0.123"/>
  <axis xyz="0 1 0"/>
</joint>
<link name="arm_link_2">
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0.1585"/>
    <geometry>
      <box size="0.0356 0.05 0.317"/>
    </geometry>
    <material name="omni/Red"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0.1585"/>
    <geometry>
      <box size="0.0356 0.05 0.317"/>
    </geometry>
  </collision>
  <inertial>
    <!-- CENTER OF MASS -->
    <origin rpy="0 0 0" xyz="0 0 0.1585"/>
    <mass value="0.29302326"/>
    <!-- box inertia: 1/12*m(y^2+z^2), ... -->
    <inertia ixx="0.00251484771035" ixy="0" ixz="0" iyy="0.00248474836108" iyz="0" izz="9.19936757328e-05
  </inertial>
</link>
<joint name="arm_joint_3" type="revolute">
```

Figure 57-URDF Snippet

According to the measurement taken by the real hardware, we map that structure on the URDF file by using the tags such as:

- Origin
- Axis
- Box

We can link the joints with one another by the help of the tags such as:

- Joint
- Parent
- Child
- Link

We can deal with the other factors with the help of tags such as:

- Collision
- Inertia
- Limit
- Material.

Using this XML (tags format), we link up each joint and set the terms of the movement by limiting certain aspects and avoiding collision between the components. These are just some of the main tags that have been used in the creation of the URDF file.

4.4.3. **Configuring MoveIt**

Now when the URDF file is ready it is time to convert it to robot configuration package. We then Launch MoveIt Setup Assistant with the following command:

```
roslaunch moveit_setup_assistant setup_assistant.launch
```

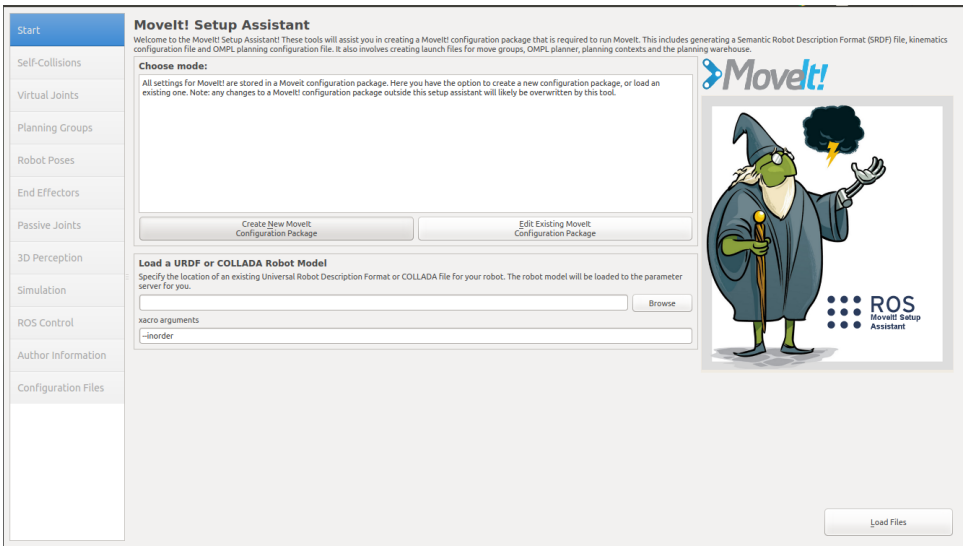

Figure 58-Moveit! Setup Wizard

After the setup assistant has been successfully launched, we configure the package file to fit the requirements of our Robotic arm model. These configuration setting includes :

- **Self Collision Matrix** : Self-Collision Matrix Generator searches for pairs of links on the robot that can safely be disabled from collision checking, decreasing motion planning processing time.
- **Add Virtual Joints :** Virtual joints are used primarily to attach the robot to the world.

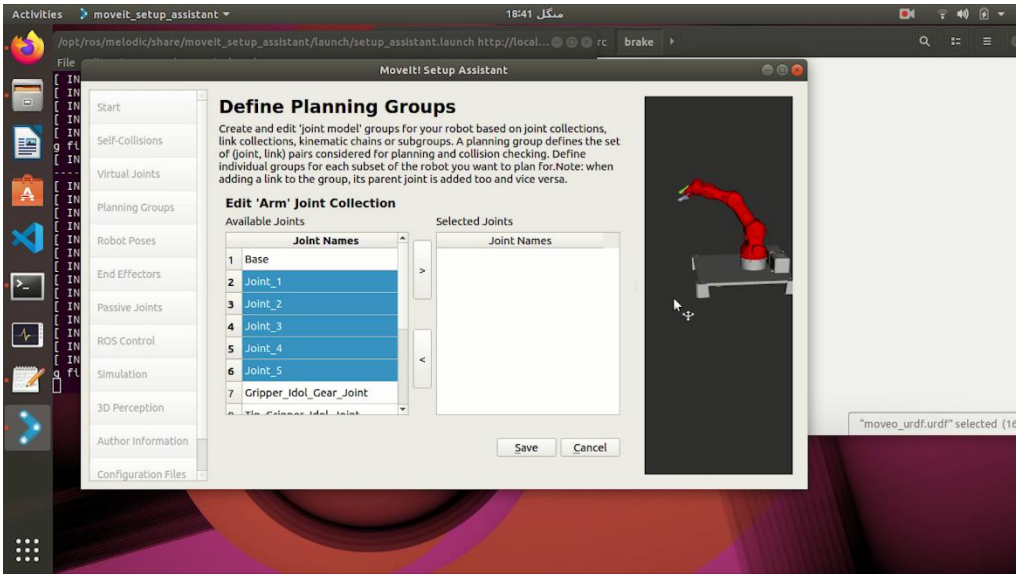

Figure 59-Moveit! Planning Groups

- **Add Planning Groups :** Planning groups are used for semantically describing different parts of your robot, such as defining what an arm is, or an end effector.
- **Add Robot Poses :** The Setup Assistant allows you to add certain fixed poses into the configuration.
- **Label End Effectors :** we designate the gripper's group that we created in planning group as a special group called **end effectors**

4.4.4. Visualizing RVIZ

Now coming to the last step as we successfully completed the configuration of the package, we now come into RVIZ in which we simulate different types of poses such as the ones that are needed in our robotic arm such as; 6-DOF movement, approaching an object that is placed within the vicinity, grasping it with its gripper or end effector, rising it upward without dropping it and then successfully placing it at a specific location.

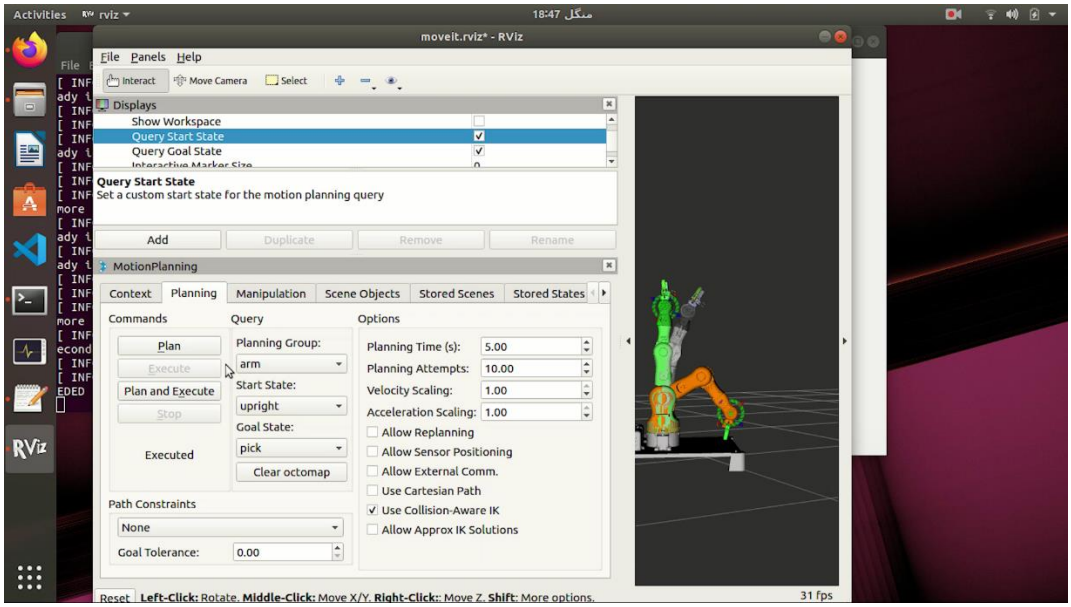

Figure 60-Rviz Simulation after Moveit! Wizard

We then performed each of these poses freely without any collision successfully. Now as all the configuration, settings and poses are set forward, we can just simply export it by using ROS Arduino library to connect the virtual Robotic arm to the real one. From that point we just need to select a certain pose in RVIZ and the hardware will actually implement it.

Chapter 5

5 Results

As discussed in the previous chapters, the project has achieved results which were sought from the initialization.

Final Outcome or deliverable has been categorize in the following points:

- Object Detection
- Live Position and Dimension of the object
- Object Recognition (Machine Learning)
- Self-learning Robotic Arm (Machine Learning)
- Grasping, pick and drop of novel objects

5.1 A flexible Object detection/recognition program with centroid/grasping-point plotting

After completing our final milestone, we achieved a program through which we can detect an object, identify it through a machine learning visual optics model, track it using a feature from the program, and acquire the measurements of the object. After getting the measurements, random points were selected as grasping points as well as the centroid was marked, as it would be one of the grasping points. After testing this part of the program, next came the developing of the virtual environment for the 3D simulation of the robotic arm, which would be synchronized with the physical model so the program can give direct inputs to the simulation, which in turn can manipulate the actual robotic arm.

5.2 A virtual simulation with 3D model and feedback looped physical model

For each object, the grasping point from which the arm was successful in gripping the object, would be stored in a database with the approximate measurements of the object, the center point which would be marked on an image and what the object is according to the ML model. As the program was designed to be flexible, any novel object can be placed in front of the surface (the object should be under the weight/length/width limit, which the robotic arm can grasp and pick) and the program will function normally.

As the virtual environment is synchronized to the actual robotic arm through a well-designed 3D model, as the arm is manipulated, live feedback is also received from the robot and displayed using the 3D model.

Chapter 6

6 Summary

After the initial section about the introduction to our project, its core idea and the performed market analysis, every major milestone was identified. Next, using previous projects with the same relative core idea, the relation between our project and cited projects was discussed and some history about the work performed on computer vision was given. Following this, the workspace that was setup for the project development was discussed in detail, which includes the software environment, the virtual space development and more.

As the project is about working around computer vision as the core, the first milestone that was set was to design a simple object detection program and as python being the most used and well-known language for computer vision and machine learning, it was chosen to go about developing our program using pre-built libraries. The methodology presented in the thesis has been made as easy to follow as possible as it simply expands on the idea of increasing the use of computer vision and automation in our daily lives.

As one of the core features of the project was to achieve flexibility, there were some constraints as to how to allow a developer to manipulate the program for different tasks. An example can be as the project primarily is an object detector, what if some task requires an object classifier based on color. This was accomplished through the versatility of the developed functions, which can be easily altered according to the requirements.

With the accessibility to a Kinect and its depth camera, the data collected had to be manipulated into something useful, which was performed through in-built functions of the OpenCV library in python. According to the data, objects were detected based on depth as the object, which had some mass, was shown in a different contrast relative to the background surface.

Any object, be it a regular shape or an irregular, were passed through the algorithm where the approximate measurements were acquired and according to a weighted argument, the center point of the object was located. As this was performed with weights included, irregular shaped objects were also countered as objects which were having mass focused on one side or divided, such as a stapler, the center point was plotted where the average of the weights lie.

After measuring the object, two random points between which the most pixels lie which were of the object were collected and in-between them, random grasping points were selected. As for most objects, the center of the object is the best grasping point as an equilibrium of forces is fixed on that point, so the centroid was also selected as a grasping location.

With the measurements of the objects in hand, the actual measurements were simply calculated through the ratio difference between the image and real world. Using a simple ROS program required to communicate with the robot, the object that was present in front of the camera could also be replicated into the 3D simulation through this method, which would perfect the virtual environment. This would help in reducing the error between the real and virtual world to a minimum due to which the program will be optimized for real world situations.

Due to the high weight of the robotic arm's frame, a delay was placed between motions of two motors, as the motors that were procured were not having the required torque to hold the frame steady. However, with the real-time feedback loop with ROS, these factors were also taken into account and countered by ROS automatically.

In the 'Project Methodology' and 'Implementation Details' chapter, the major milestones which were achieved are mentioned in detail and the exact methodology which was followed and how it was followed has also been described comprehensively. In the final chapter, the results are summarized which pertain the most for those who are technology illiterate and want to join the club and start doing projects.

7 Bibliography

[1] wiki.ros.org. [Online].

[2] A. L. C. Leandro Marcomini, " Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos".

[3] A. B. M. C. Sandeep Kumar, "Object Detection and Recognition in Images".

[4] M. P. Z. S.-t. X. a. X. W. Zhong-Qiu Zhao, "Object Detection with Deep Learning: A Review".

[5] Q. W. M. W. S. R. Vishal Kumar, "Computer Vision Based Object Grasping 6DoF Robotic Arm Using Picamera".

[6] a. A. Y. N. Justin Driemeyer, "Robotic grasping of novel objects using vision," 2008.

[7] W. Q. L. X. F. Z. Yiming Zuo, "CRAVES: Controlling Robotic Arm With a Vision-Based Economic System".

[8] D. R. S. Mukesh Tiwari, "A Review of Detection and Tracking of Object from Image and Video Sequences".

[9] S. M. V. M. J. Jonathan M. Aitken, "Adaptation of System Configuration under the Robot Operating System".

[10] Oppenheim, Signals and Systems, ABC publisher, 2009.

[11] A. Author, "How to write a [a[er," *Journal on Writing,* pp. 1482-1489, 2017.