DUM-E – PROJECT REPORT

MENTOR - RISHIKA

TEAM – SHIVSHANKAR, ABHAY, SIDDHANT, AYUSH

PART I –
What all we did!!

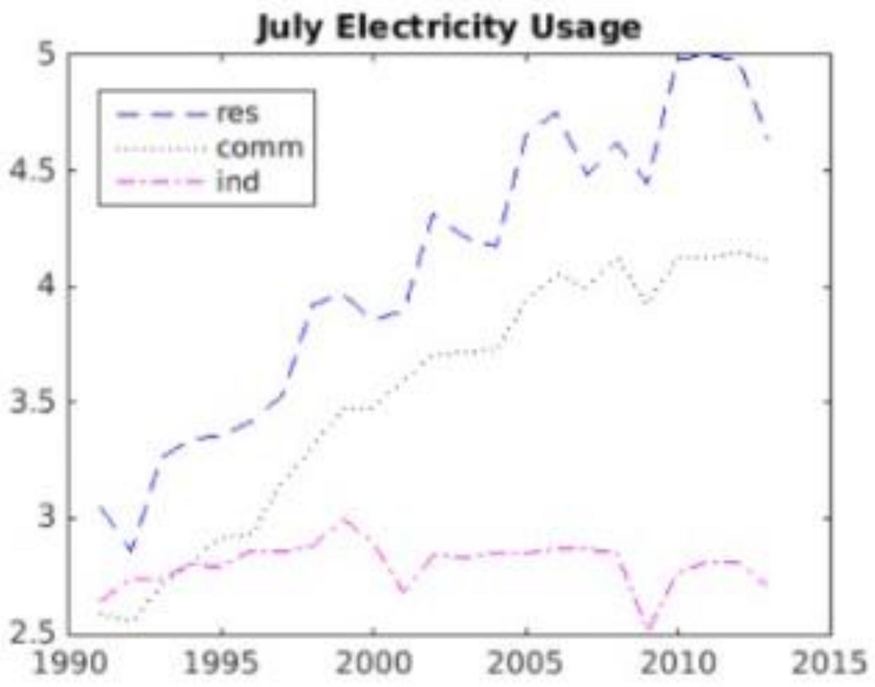
PART II –
All about Dum-e

PART 1 – WHAT ALL WE DID!!

1. BASICS OF MATLAB FRAMEWORK
2. SIMULINK
3. SIMSCAPE
4. RIGIDBODYTREE – CONSTRUCTING ROBOTIC MANIPULATOR IN MATLAB
5. 2-D PATH TRACING WITH INVERSE KINEMATICS
7. IMAGE PROCESSING – OBJECT DETECTION AND CLASSIFICATION
8. HAAR CASCADE – ANOTHER METHOD FOR OBJECT DETECTION AND CLASSIFICATION

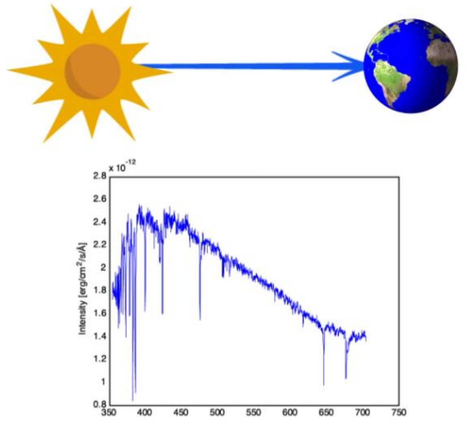
I. BASICS OF MATLAB FRAMEWORK

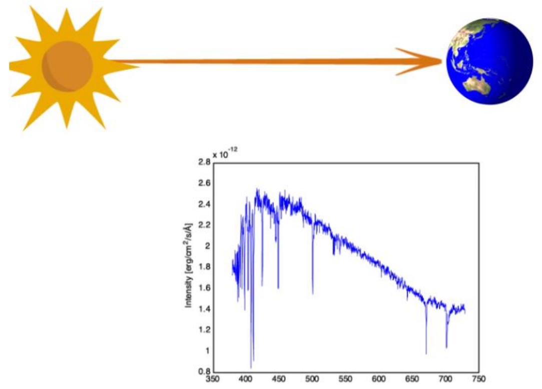
- Firstly, we familiarize ourselves with the syntax of MATLAB. Though arrays play a significant role in any high-level language, it is especially true in the case of MATLAB. And so, we digged a little deeper in those concepts like we learn how to generate arrays of random elements and how to extract information from array. All this later came handy when we were learning Image Processing as images are nothing but numerical array, where each element is equivalent to the intensity of the corresponding pixel.
- We learnt how to plot data graphically and extract useful information from it. We did a couple of sub-projects too on the basis of the concepts we learned.
    - ***Electricity usage on graph* –** In this project, we imported (.mat) files containing US electricity consumption data, and from that we were able to graph the electricity consumption of the three economic sector, i.e residential, commercial and industrial.



I. BASICS OF MATLAB FRAMEWORK

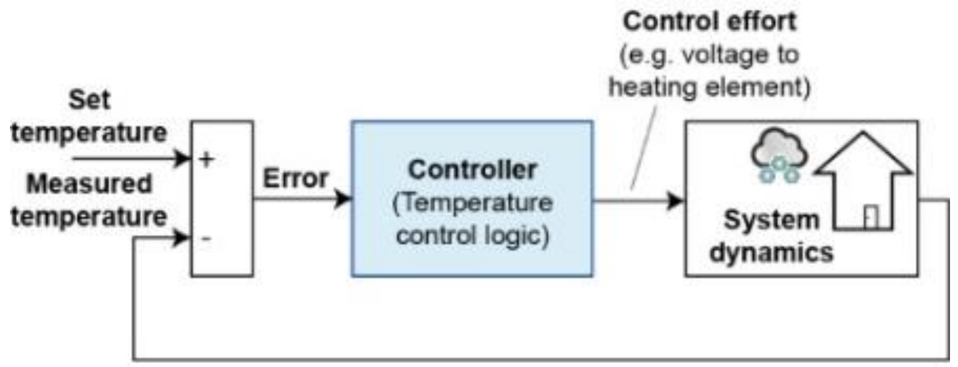
- ***Stellar Motion* –** In this project, we were given data files containing the spectral data of multiple star. Our job was to determine which stars were receding from us and which were coming towards us. The physical concept behind the project was simple, a star emits light of various wavelengths and each wavelength has different intensity. This property gives each star an identity, which is called its spectra. A good analogy for spectra would be fingerprints. Now, if we use the concept of doppler effect from EM waves with the spectral data, we will conclude that the star who moves towards earth undergoes blue-shift and opposite in the case of moving farther.





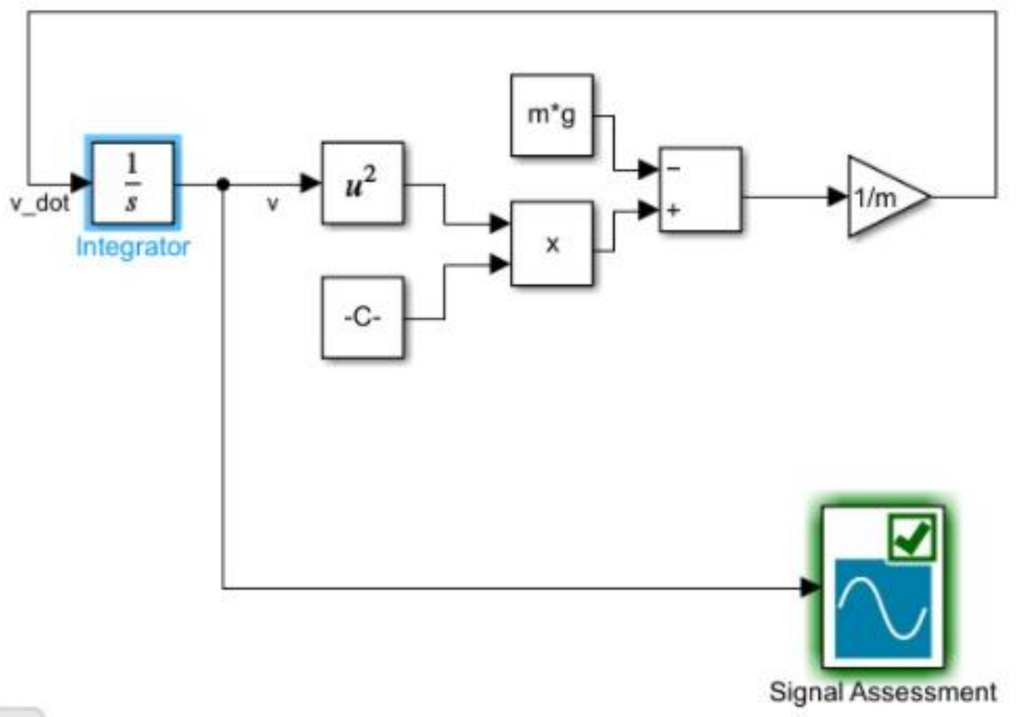
SIMULINK

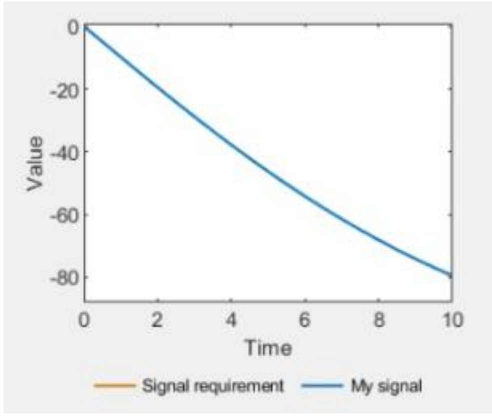
- Simulink is a MATLAB tool, which is used to model, simulate and analyze multi-domain dynamical systems. In simpler words, we can model equations using graphical block diagrams. We also did the following sub-projects while studying this framework.
- ***Thermostat* –** We designed a control system through which we were able to control the temperature of an environment, by monitoring the current temperature of the room and comparing it with the desired one.


Set temperature
Measured temperature
+
-
Error
Controller (Temperature control logic)
Control effort (e.g. voltage to heating element)
System dynamics

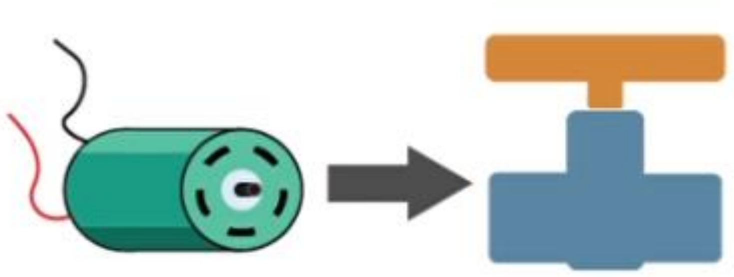
SIMULINK

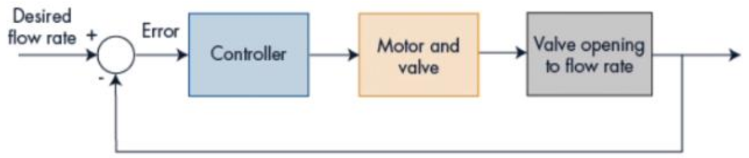
$$m\dot{v} = \frac{1}{2}\rho C_d A v^2 - mg$$





SIMSCAPE

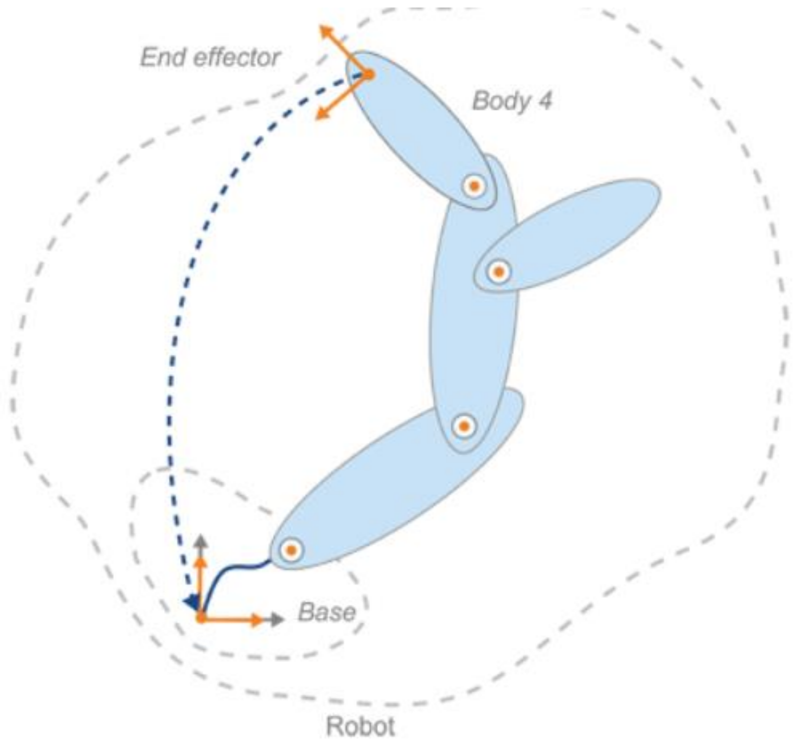




- Simscape is a significant part of the MATLAB framework, which helps one to model physical systems, interpret them and accordingly give the graphs and calculations as output.
- It contains various libraries through which one can make various electrical and mechanical system and even electro-mechanical systems. For example, you can model bridge rectifiers, electric motors, refrigerator systems, etcetera.
- We also modeled an ***electronic valve*** as our hands-on project. By using the various electrical, mechanical and electro-mechanical block elements, we modeled our electric motor, then we designed a feedback controller to around this plant to maintain a desired flow rate through the valve.

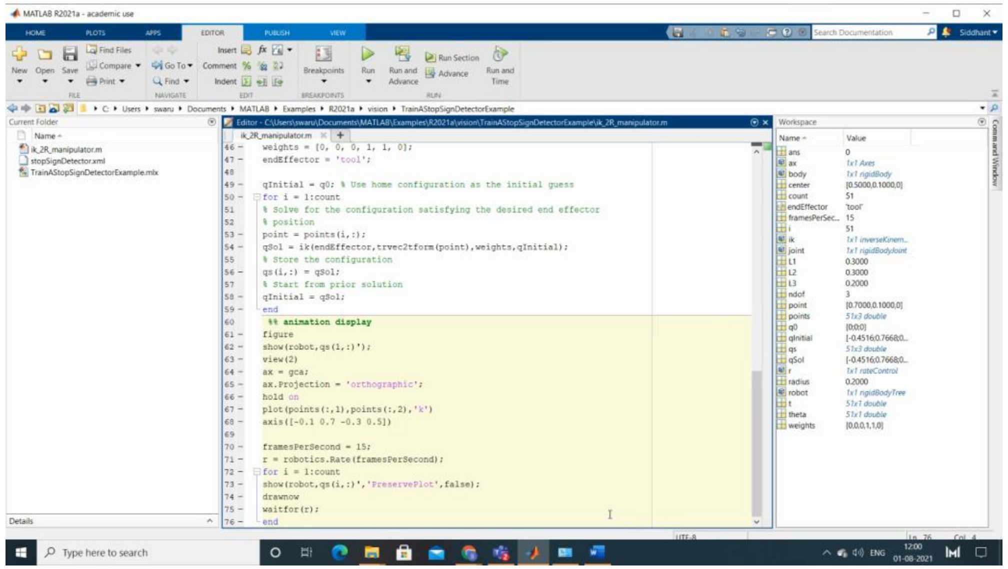
RIGID BODY TREE

- The rigid body tree model is a representation of robotic manipulator. We use ***rigidBodyTree*** to create a robotic manipulator. It is made of rigid bodies and joints, which are created using ***rigidBody*** and ***rigidBodyJoints*** respectively.
- By learning this framework, we were able to understand the basic mechanical concepts related to robotic manipulator. Like degrees of freedom of a robot, types of joints, which are by the way, **fixed, revolute and prismatic**, etcetera.



2-D PATH TRACING WITH INVERSE KINEMATICS

Inverse kinematics is just opposite to forward kinematics. It refers to process of obtaining joint angles from known coordinates of end effector. For path tracing this process happens at each point of time for each co-ordinate of end effector.


MATLAB R2021a - academic use
Editor - C:\Users\swaru\Documents\MATLAB\Examples\R2021a\vision\TrainAStopSignDetectorExample\ik_2R_manipulator.m
ik_2R_manipulator.m
Current Folder
ik_2R_manipulator.m
stopSignDetector.xml
TrainAStopSignDetectorExample.mlx

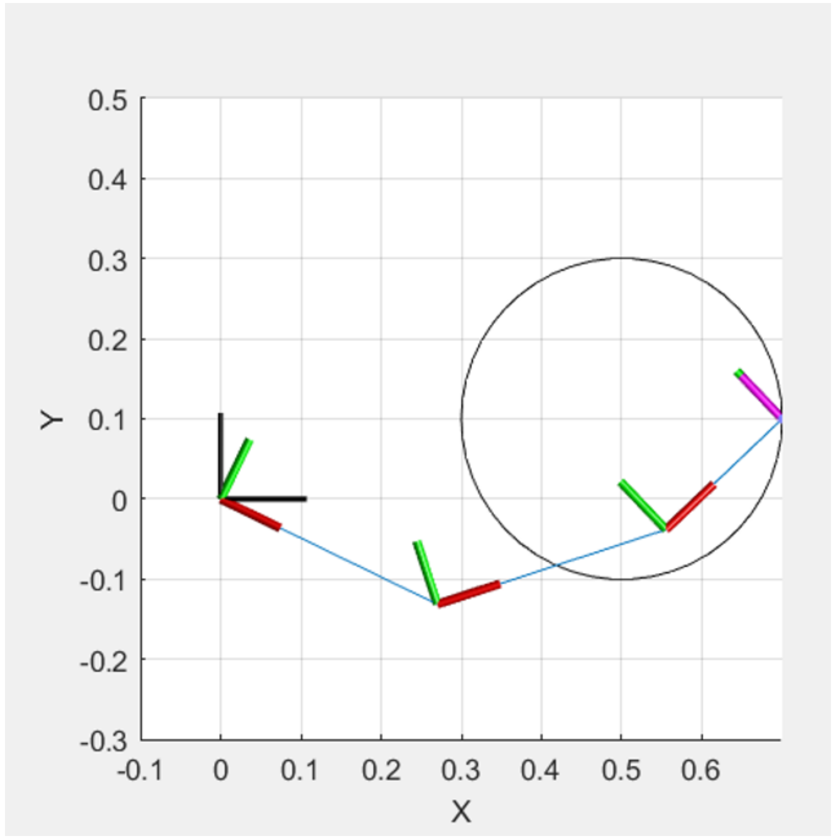
```
weights = [0, 0, 0, 1, 1, 0];
endEffector = 'tool';

qInitial = q0; % Use home configuration as the initial guess
for i = 1:count
% Solve for the configuration satisfying the desired end effector
% position
point = points(i,:);
qSol = ik(endEffector,trvec2tform(point),weights,qInitial);
% Store the configuration
qs(i,:) = qSol;
% Start from prior solution
qInitial = qSol;
end
 %% animation display
figure
show(robot,qs(1,:)');
view(2)
ax = gca;
ax.Projection = 'orthographic';
hold on
plot(points(:,1),points(:,2),'k')
axis([-0.1 0.7 -0.3 0.5])

framesPerSecond = 15;
r = robotics.Rate(framesPerSecond);
for i = 1:count
show(robot,qs(i,:)','PreservePlot',false);
drawnow
waitfor(r);
end
```

Workspace

Name	Value
ans	0
ax	1x1 Axes
body	1x1 rigidBody
center	[0.5000,0.1000,0]
count	51
endEffector	'tool'
framesPerSec...	15
i	51
ik	1x1 inverseKinem...
joint	1x1 rigidBodyJoint
L1	0.3000
L2	0.3000
L3	0.2000
ndof	3
point	[0.7000,0.1000,0]
points	51x3 double
q0	[0;0;0]
qInitial	[-0.4516;0.7668;0...
qs	51x3 double
qSol	[-0.4516;0.7668;0...
r	1x1 rateControl
radius	0.2000
robot	1x1 rigidBodyTree
t	51x1 double
theta	51x1 double
weights	[0,0,0,1,1,0]



2-D PATH TRACING WITH INVERSE KINEMATICS

- In this the output inverse-kinematic solution is fed back as the initial guess for the next solution. This initial guess helps track the end-effector pose and generate smooth configurations

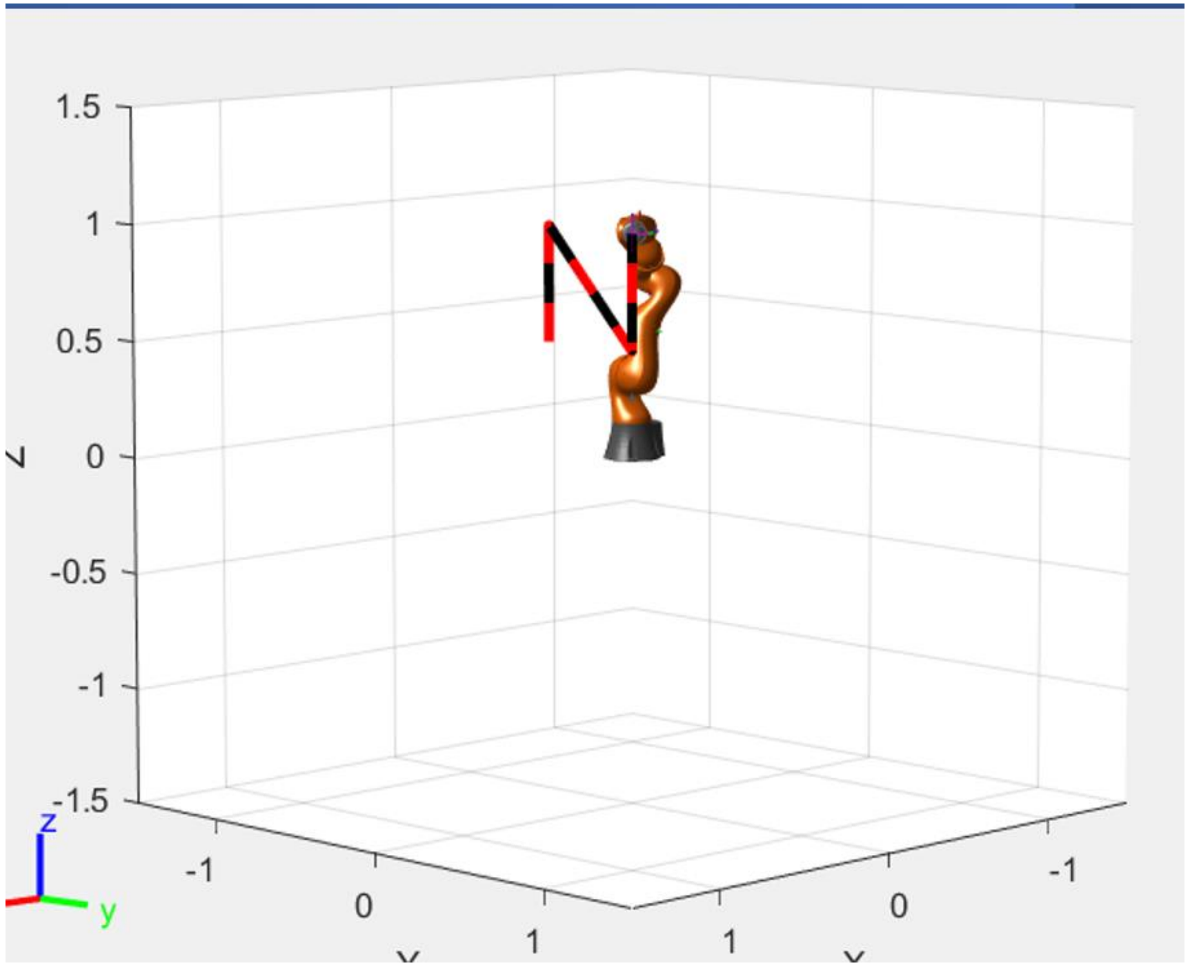


IMAGE PROCESSING

- Image Processing in general is a tool which can be used to monitor, discover and identify information in images. Putting it more simply, it is a set of techniques to extract information from images or to modify it.
- We decided to do our first course in IP in MATLAB by learning in detail a certain example. So, we did a tutorial which taught us how to make an IP algorithm to distinguish images of receipts from other images.
- We learnt that an image is nothing but a numerical array, where each of its elements contains the value of intensity of light in the corresponding pixel.
- We also learnt the single most important concept in any IP algorithm, which is how to convert colored pictures into grayscale image and then how to convert them into binary images.
- We also learnt how to increase the accuracy of our analysis by improving the clarity and decreasing the noise in the image and setting threshold to extract useful objects and to remove unnecessary details.
- Finally, in making the aforementioned algorithm, we were able to distinguish the images of receipts by calculating the *rowsum* of the rows of pixels and calculating the number of minimas for it, and then we set a threshold for the number of minimas. The algorithm classified the images with greater value than *rowsum threshold* as Receipt images.

HAAR CASCADE

- The vision.CascadeObjectDetector System object detects objects in images by sliding a window over the image. The detector then uses a cascade classifier to decide whether the window contains the object of interest. The size of the window varies to detect objects at different scales, but its aspect ratio remains fixed. The detector is very sensitive to out-of-plane rotation, because the aspect ratio changes for most 3-D objects. Thus, you need to train a detector for each orientation of the object. Training a single detector to handle all orientations will not work.
- The cascade classifier consists of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

PART II – ALL ABOUT DUM-E!!

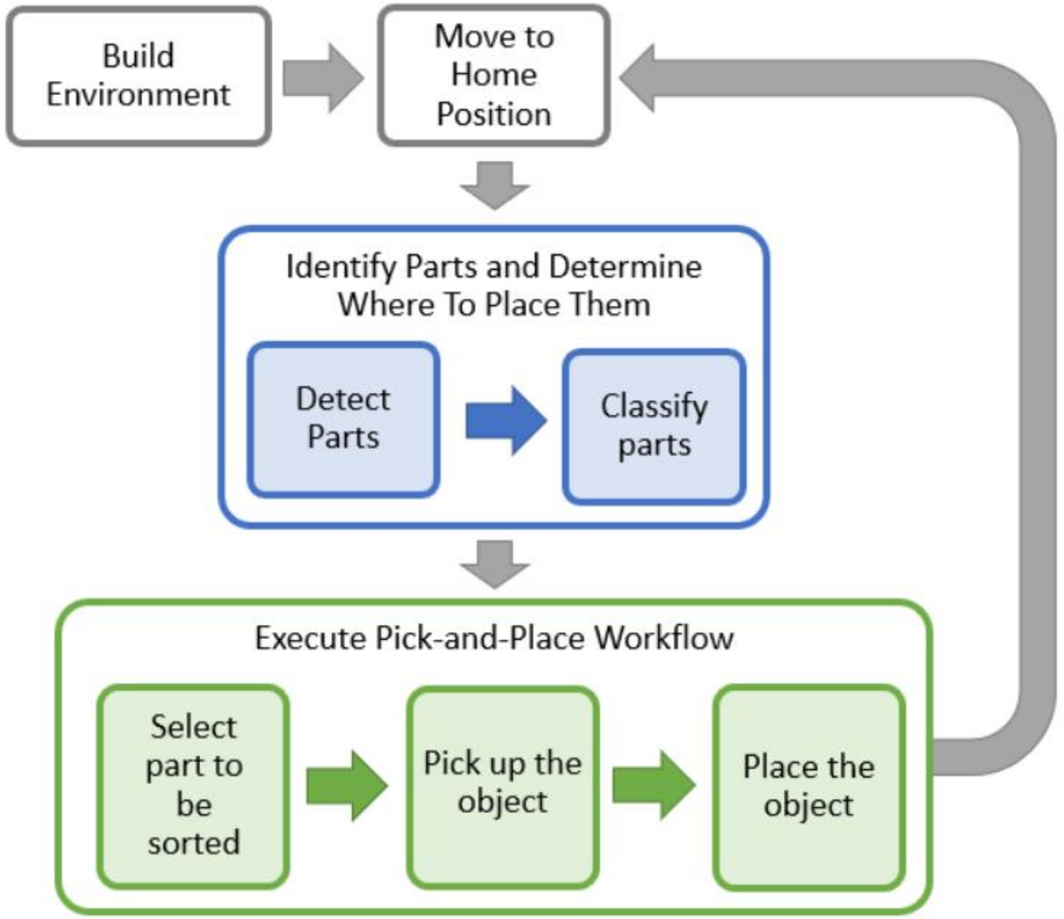
1. HISTORY OF DUM-E
2. OUR DUM-E
3. TOOLBOXES USED
4. FUTURE OF DUM-E

HISTORY OF DUM-E

- As you all probably already know, Dum-E was a robotic arm which repeatedly appears in the movie, Iron Man I. Though the robotic arm keeps creating mess for Tony, it had saved his life too.
- When Obadiah Stane decided to steal Tony's Arc Reactor, Tony almost died and went to his garage to get the arc reactor on a mantle that Pepper Potts made for him. He was unable to reach it and gave up, but Dum-E gave it to him, thus saving Tony's life.
- This was 'the great' Tony Stark's version of Dum-E, next we will talk about ours.

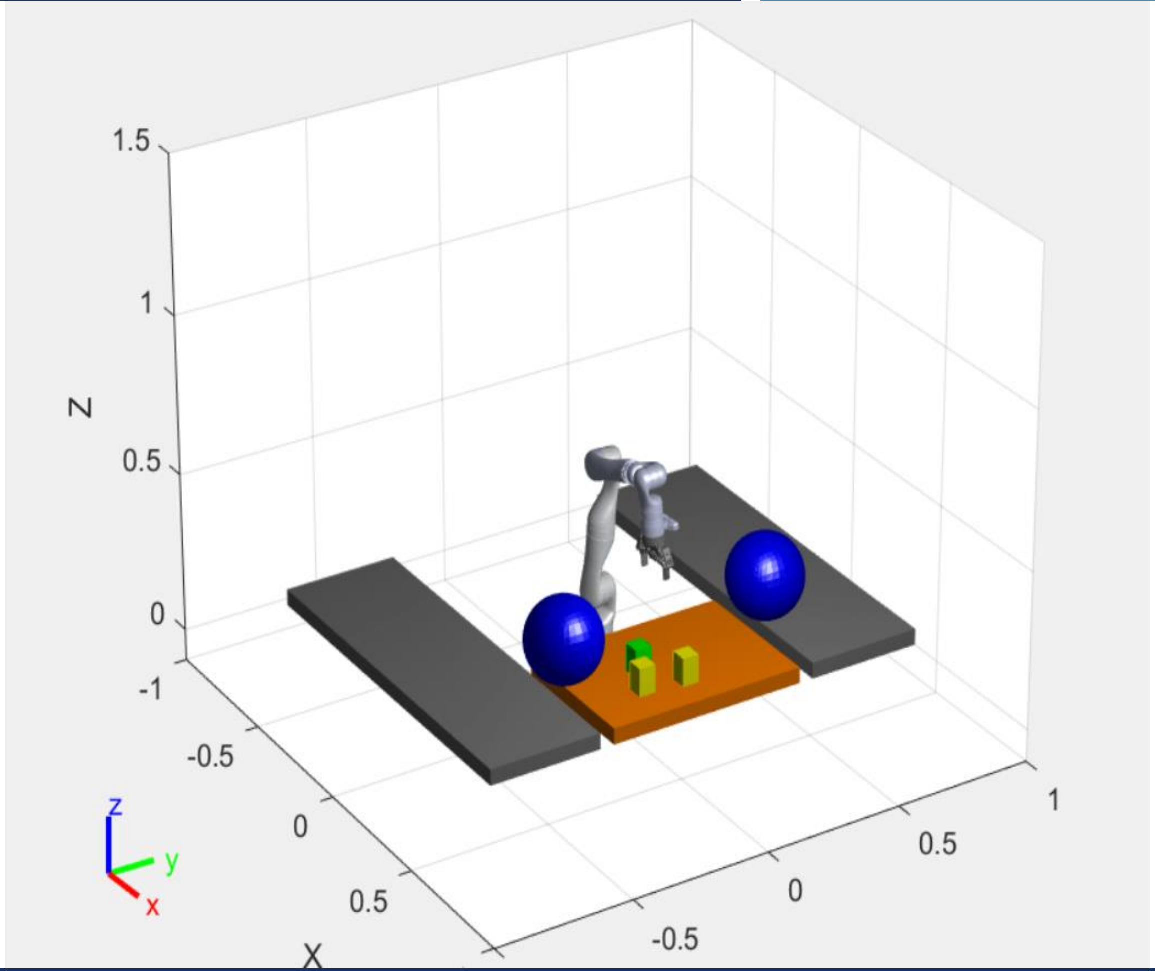
OUR DUM-E

- Our project was basically designed to perform Pick and Place functions. For now, we have successfully created the pick and place function in predefined static environment.
  Also, with the help of Haar Cascade From Computer vision toolbox, we have trained it to detect objects like ball in real environment.



A Stateflow chart is a MATLAB tool used to model a machine in a graphical manner and for scheduling its different tasks.

STATEFLOW MODEL OF DUM-E



Setting up the Environment -
Firstly, the robotic manipulator is imported to the environment ,using *loadrobot* function. Then objects, shelves and obstructions are also loaded in the same environment.
Now, initial and final joint parameters are determined by using inverse kinematics.

Object Detection and Classification –
This is done by using RGB model. An IP algorithm is made for distinguishing and classifying different colored objects based on the color planes of RGB model of each image.
For each differently colored object, different final positions are also specified.

WORK ENVIRONMENT

Static and Dynamic Environment –

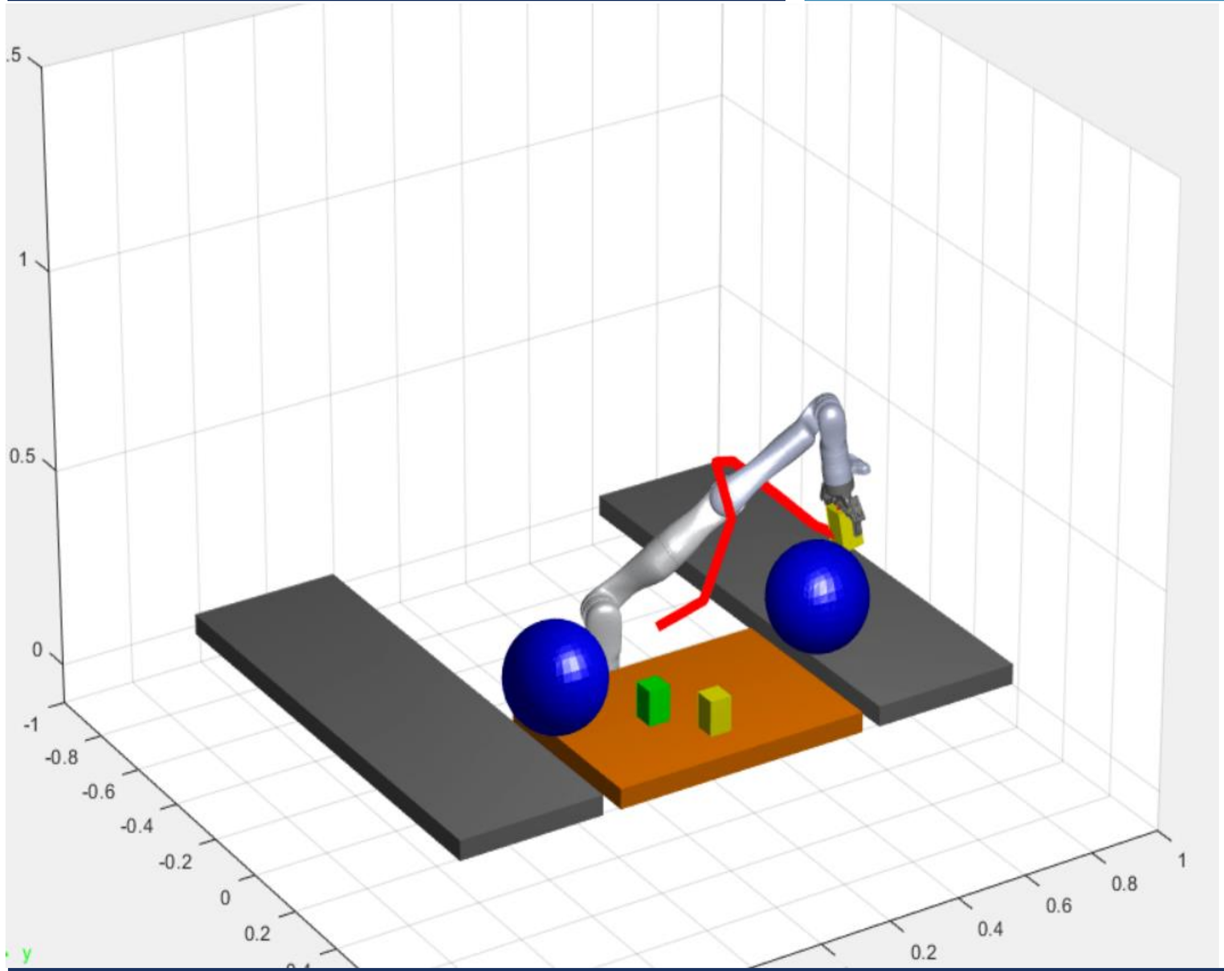
The *isMovingObst* function value is set to be ‘false’ if the environment is static and its opposite if the environment is dynamic. For the latter, one must make something called *collision world,* and also add *collision meshes* into it.

With the help of Model Predictive Control Toolbox, a Nonlinear Model Predictive Controller is designed for determining a collision free trajectory.

A function called *cost function* is central in designing the necessary algorithm for collision free trajectory.

$$J = \int_0^T (p_{\mathrm{ref}} - p(q(t)))' Q_r (p_{\mathrm{ref}} - p(q(t))) + u'(t) Q_u u(t)\, \mathrm{dt} + (p_{\mathrm{ref}} - p(q(T)))' Q_t (p_{\mathrm{ref}} - p(q(T))) + \dot{q}'(T) Q_v \dot{q}(T)$$

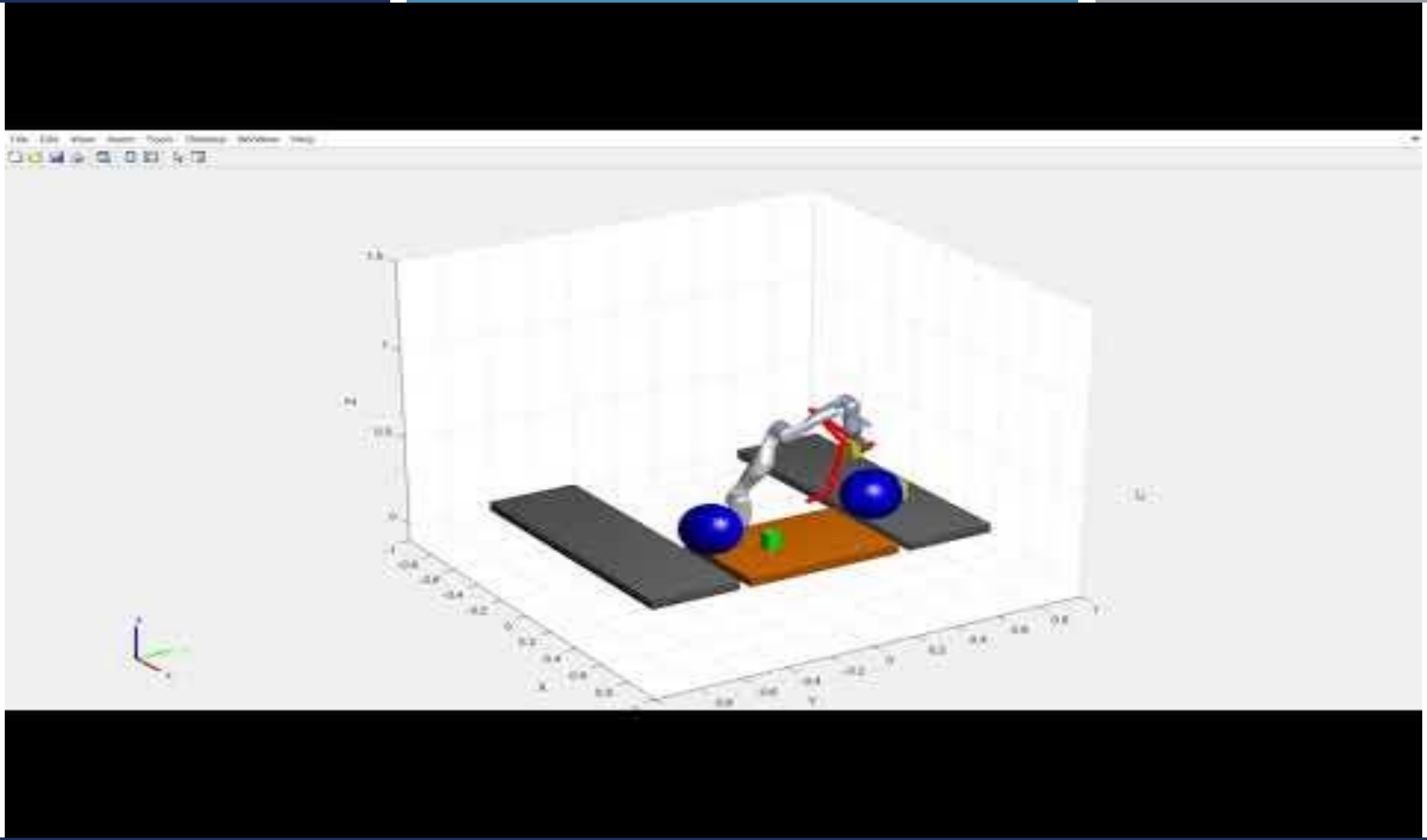
STATIC VS DYNAMIC ENVIRONMENT



For deciding the perfect collision-free trajectory, a number of iterations are performed by the manipulator to finalize the best path. The function *nlmpcmove* from Model Predictive Toolbox comes handy for generating closed loop trajectory.

Each iteration calculates the position, velocity, and acceleration of the joints to avoid obstacles as they move towards the goal.

DESIGNING CLOSED LOOP TRAJECTORY FOR DYNAMIC ENVIRONMENTS



SIMULATION

TOOLBOXES AND ADD-ONS USED

Robotics System Toolbox is used to model, simulate, and visualize the manipulator, and for collision-checking.

Model Predictive Control Toolbox and **Optimization Toolbox** are used to generate optimized, collision-free trajectories for the manipulator to follow.

Stateflow is used to schedule tasks in the following model and step from task to task.

FUTURE OF DUM-E

Our future resolution for this project are following,

1. To integrate a path planning algorithm, which lets the robotic arm as a whole to move in a static environment,
2. To make it function in a dynamic environment using computer vision and statistical and machine learning toolbox,
3. To make both the end-effector as well as the robotic arm as a whole to move in a dynamic environment,

THANK YOU