Engineering Projects Portfolio - II

Programming, Simulation & Analysis

Yohann Panthakee

(MSc Motorsport Engineering Graduate, Cranfield University)

LinkedIn Profile

GradCAD Page

GradCAD Group

Contact Details

Introduction

AIM: The idea behind creating this Simulation and Programming Portfolio is to give the reader a deeper insight into my engineering simulation and software skills which I have self-learnt over the last 5+ years. Most of these Simulation projects were in fact created, well before I started my Automotive Engineering degree at Coventry University. I strongly believe that this differentiating presentation will allow the reader to objectively assess my engineering skills and abilities which would be relevant to your company. I would be delighted to be given the opportunity to discuss this portfolio in more detail, in person and I can be reached through the contact information on the front page.

Contents

Excel Projects

- Electric Power Calculator — Page 4
- Gearbox Design Calculator — Page 5
- Vehicle Dynamics Calculator — Page 9

ANSYS Simulation Projects

- Vertical Wind Turbine Simulation — Page 13
- Fluent 6DOF Induced Wheel Spin — Page 16
- Rigid Body Dynamics Beam Engine — Page 19
- Explicit Dynamics Impact — Page 21

MATLAB Projects

- Colour Tracking Algorithm — Page 23
- Automated Generation of Synthetic Training Data — Page 25
- Transfer Learning Image Classification Algorithm using a CNN Model — Page 28
- Transfer Learning Image Classification Algorithm using a RCNN Model — Page 29
- Using either CNN or RCNN with Live Image Acquisition — Page 30

Python Projects

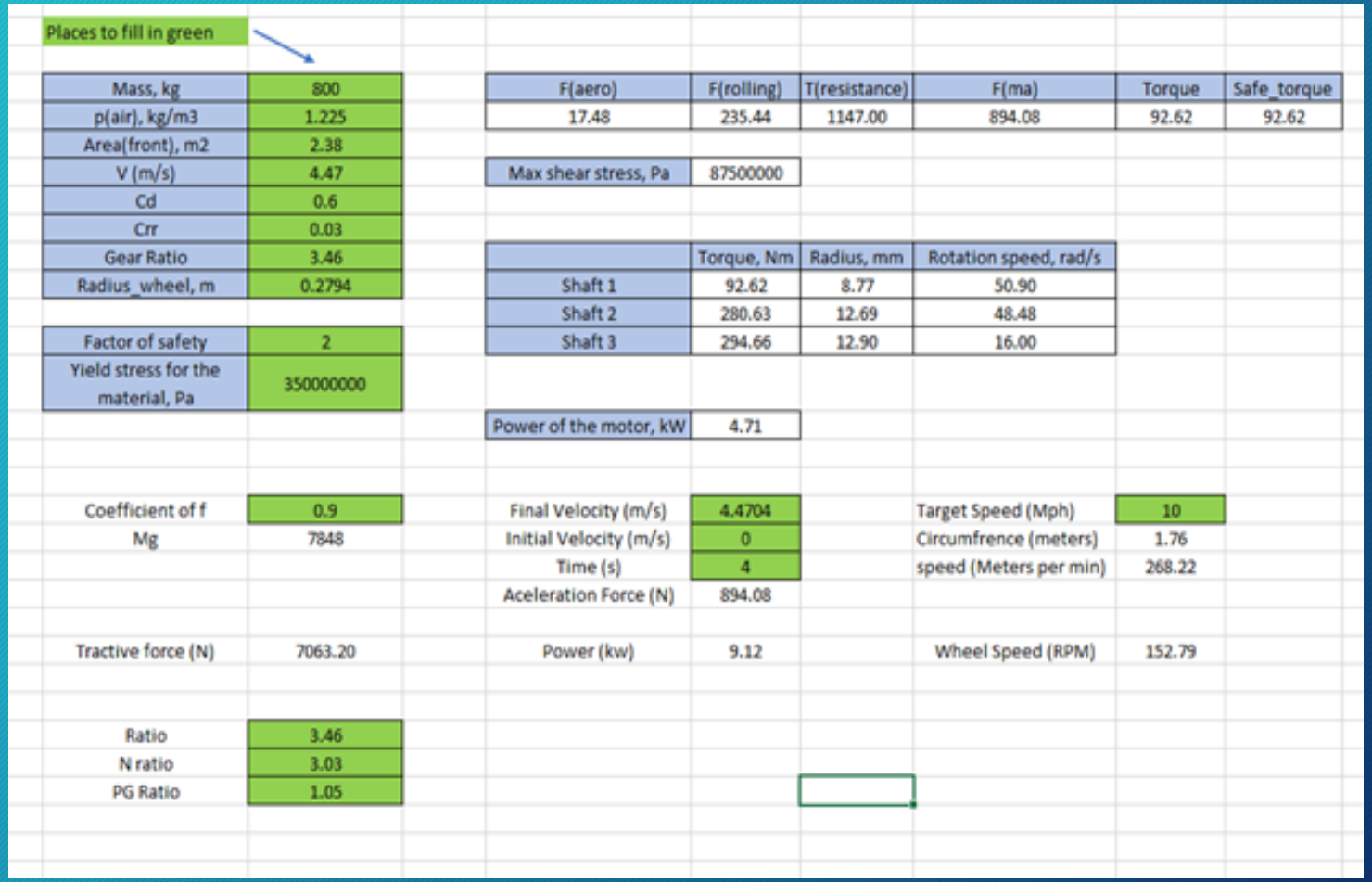
- Engineering Coursework in Python — Page 31
- Drawing App — Page 33
- Lane Detection — Page 34
- Colour Tracking Algorithm using Image Thresholding — Page 35

Electric Powertrain Calculator

Excel Project

- **Software:** Excel
- **Aim:** To develop a excel calculator to calculate the required motor power for electric vehicle applications.
- **Challenges:** Applying my theoretical knowledge.

Solution:



Places to fill in green

Mass, kg	800
p(air), kg/m3	1.225
Area(front), m2	2.38
V (m/s)	4.47
Cd	0.6
Crr	0.03
Gear Ratio	3.46
Radius_wheel, m	0.2794
Factor of safety	2
Yield stress for the material, Pa	350000000
Coefficient of f	0.9
Mg	7848
Tractive force (N)	7063.20
Ratio	3.46
N ratio	3.03
PG Ratio	1.05

F(aero)	F(rolling)	T(resistance)	F(ma)	Torque	Safe_torque
17.48	235.44	1147.00	894.08	92.62	92.62

Max shear stress, Pa	87500000

	Torque, Nm	Radius, mm	Rotation speed, rad/s
Shaft 1	92.62	8.77	50.90
Shaft 2	280.63	12.69	48.48
Shaft 3	294.66	12.90	16.00

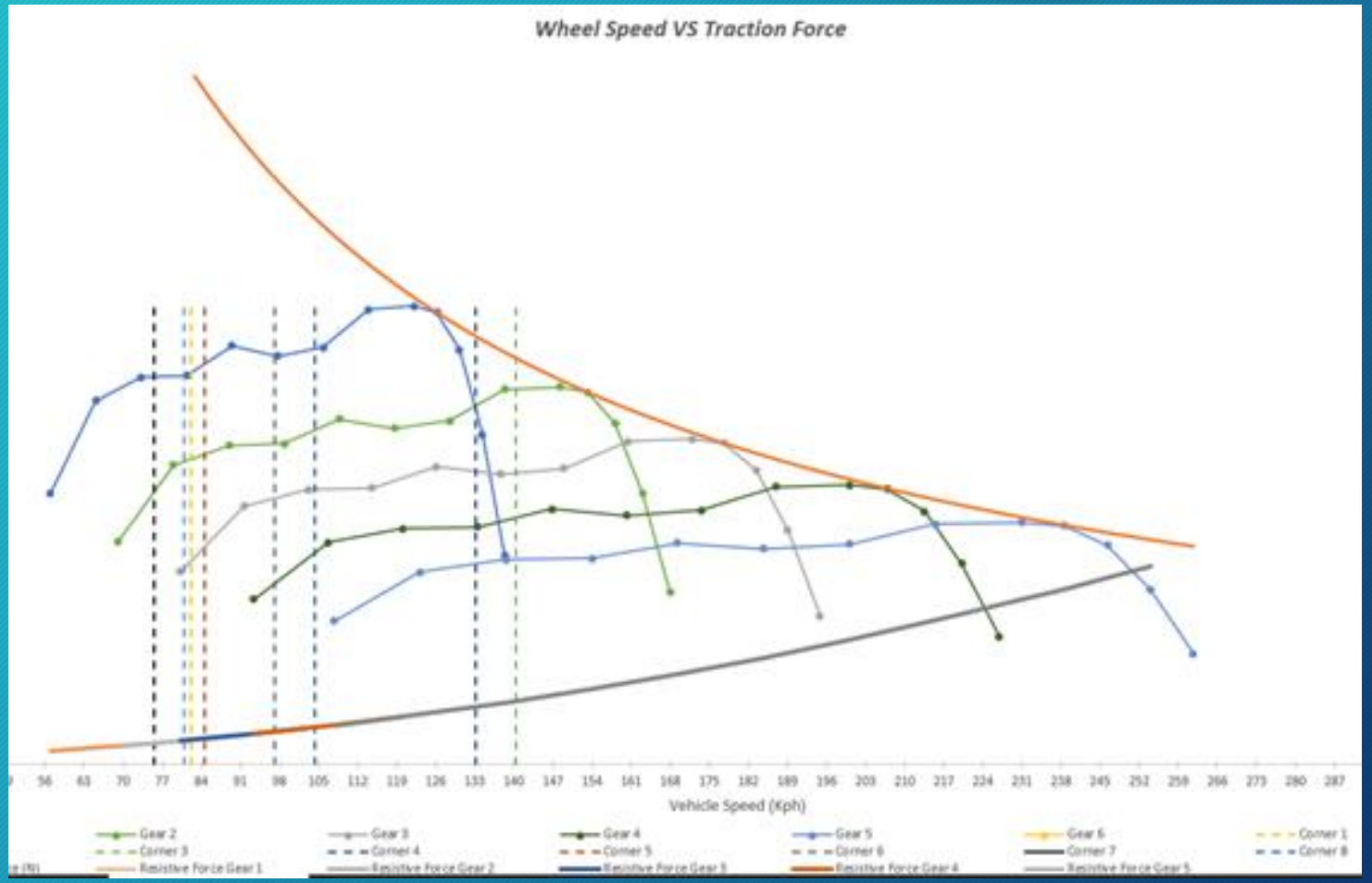
Power of the motor, kW	4.71

Final Velocity (m/s)	4.4704	Target Speed (Mph)	10
Initial Velocity (m/s)	0	Circumfrence (meters)	1.76
Time (s)	4	speed (Meters per min)	268.22
Aceleration Force (N)	894.08		
Power (kw)	9.12	Wheel Speed (RPM)	152.79

Gearbox Design Calculator

Excel Project

- **Software:** Excel
- **Aim:** To develop an excel calculator to optimise a 5 speed transmission based on set gear and final drive ratios.
- **Challenges:** Making it user friendly to use.
- **Solution:** Added drop down menus making it quick and simple to update parameters.



Gearbox Design Calculator

Continued

Excel Project

Transmission Data

	Input	
	Output	

Gear Ratios	Ratio	Spur gear (input)	Pinion gear (output)	Drop (in %)
1st	2.071	14	29	0.00%
2nd	1.706	17	29	17.62%
3rd	1.471	17	25	13.77%
4th	1.263	19	24	14.14%
5th	1.095	21	23	13.30%
6th				100.00%
Differential ratio	3.62	Qualfe		

Vehicle Data

Vehicle Mass	955	Kg		
Frontal Area	2.31	m^2		
Rolling Resistance Coeff	0.01			
Aerodynamic Resistance Coeff	0.6			
Weight Distribution (Front)	0.45	%		
Weight Distribution (Rear)	0.55	%		
Weight Dirstibution			X	Y
Front Weight Dirstribution	4215.85	Kg	0.00	4215.8475
			254.01	4215.8475
Rear Weight Dirstribution	5152.70		0.00	5152.7025
			261.70	5152.7025

Wheel Data

Wheel Diameter	0.6	M
Wheel effective radius	0.3	M
Wheel Circumference	2.03	M

Vehicle Resistance

Rolling Resistance	93.69	N
Gradient Resistance (Angle)	0.00	N
Gradient Resistance (%)	0.00	N

Engine Data

	RPM	
Max Power (BHP)	430	7200
Max Torque (Nm)	390	5750
RPM Limit	8500	

Engine Speed (RPM)	Engine Torque (N)	Torque (N/m) Test Engine Data
3500	255	281
4000	342	301
4500	364	352
5000	366	363
5500	394	389
6000	384	360
6500	392	365
7000	428	335
7500	431	321
7750	425	
8000	390	303
8250	310	207
8500	197	196

Gearbox Design Calculator

Continued

Excel Project

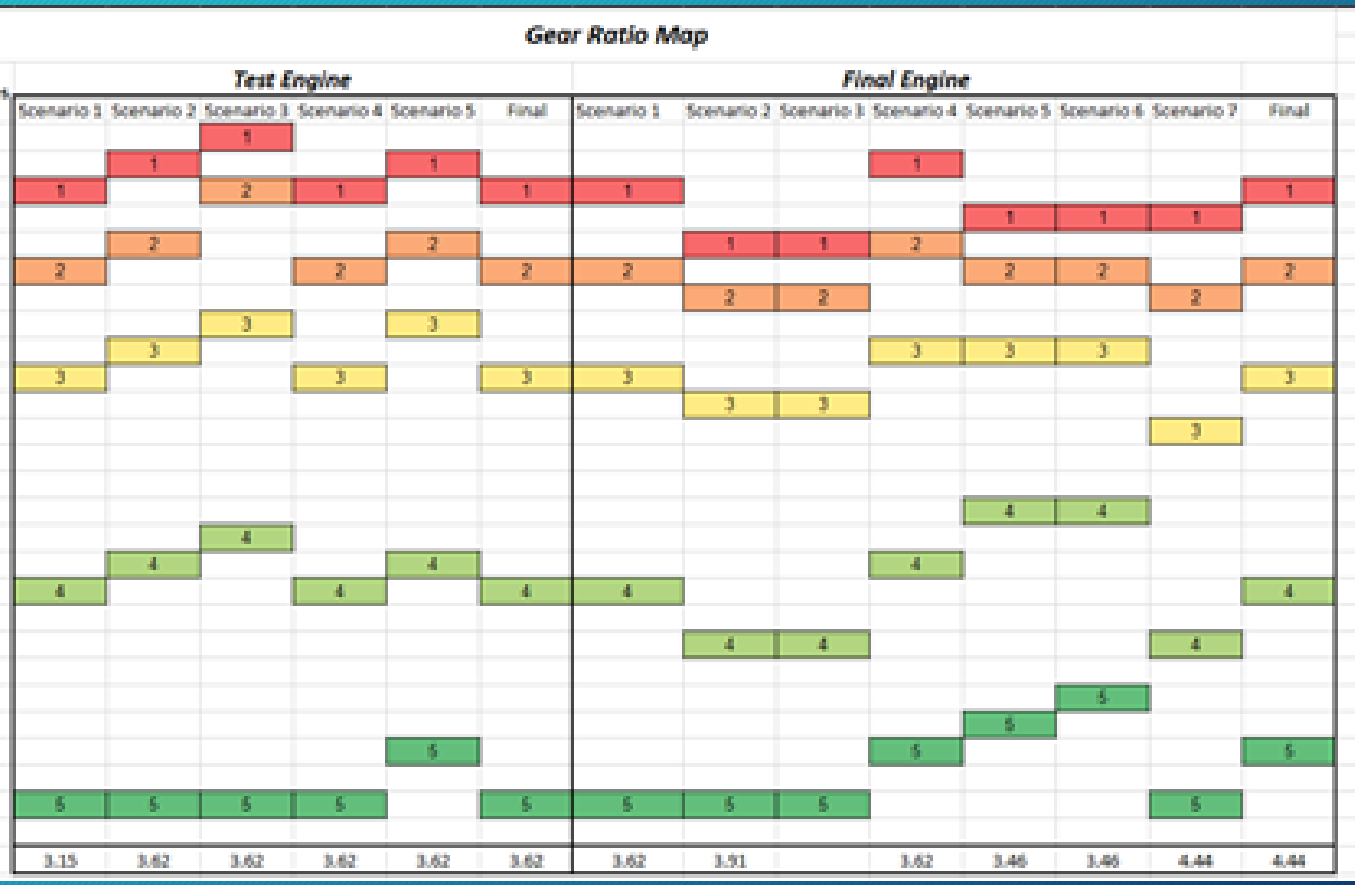
Given Transmission Ratios

Ratio O/I	Ratio I/O	Input	Output
2.692	0.371	13	35
2.214	0.452	14	31
2.071	0.483	14	29
1.933	0.517	15	29
1.813	0.552	16	29
1.706	0.586	17	29
1.611	0.621	18	29
1.556	0.643	18	28
1.471	0.68	17	25
1.421	0.704	19	27
1.381	0.724	21	29
1.35	0.741	20	27
1.316	0.76	19	25
1.286	0.778	21	27
1.263	0.792	19	24
1.238	0.808	21	26
1.211	0.826	19	23
1.182	0.846	22	26
1.167	0.857	24	28
1.143	0.875	21	24
1.12	0.893	25	28
1.095	0.913	21	23
1.077	0.929	26	28
1.053	0.95	19	20
1.038	0.963	26	27
1	1	24	24

Diff Ratio

Ratio	Manufacturer
3.06	Mercedes 380 Coupe
3.07	M3 Diff
3.15	BMW
3.15	BMW M3 Diff E92
3.25	M3 Diff
3.46	M3 Diff
3.538	Nissan 350z
3.62	Qualfe
3.73	M3 Diff
3.91	M3 Diff
4	M3 Diff
4.06	Ford Focus ST
4.083	Nissan 200sx
4.11	Qualfe
4.44	Subaru

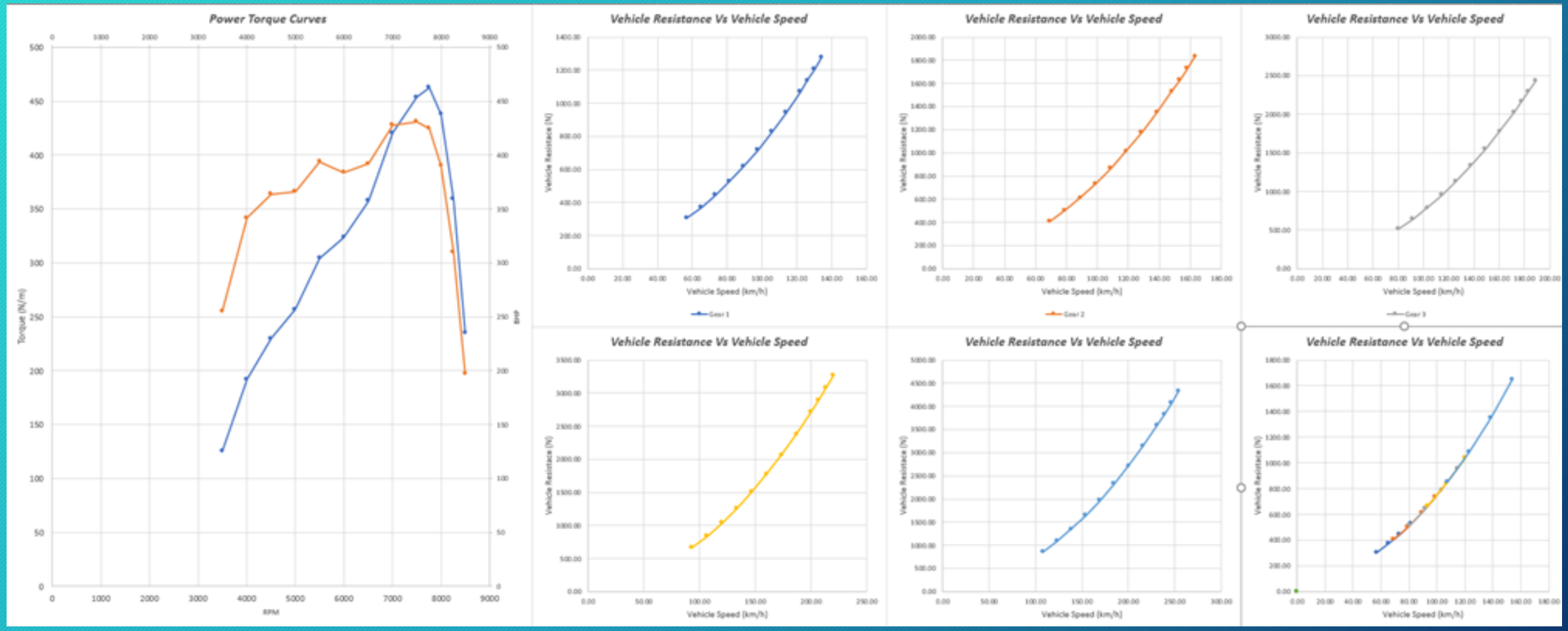
Gear Ratio Map



Gearbox Design Calculator

Continued

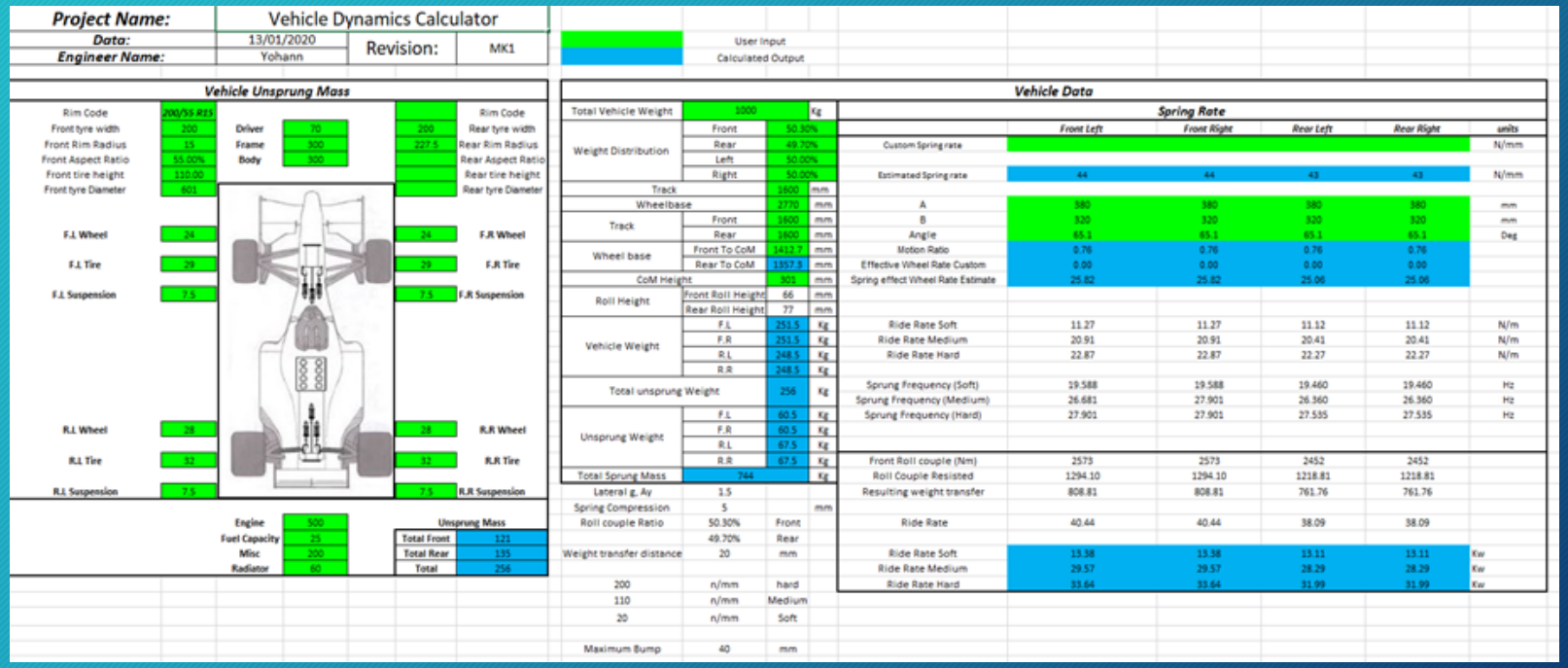
Excel Project



Vehicle Dynamics Calculator

Excel Project

- **Software:** Excel
- **Aim:** To create an excel calculator to develop and analyse vehicle dynamic systems for motorsport applications.
- **Challenges:** Understanding the background theory, developing the dynamic 2D suspension models and having them updated in real time.
- **Solution:** I did a lot of research to build upon the basics learnt in my university course. My key information sources were books such as “Race Car Design” by Derek Seward & “Vehicle Dynamics” by Martin Meywerk.



User Input

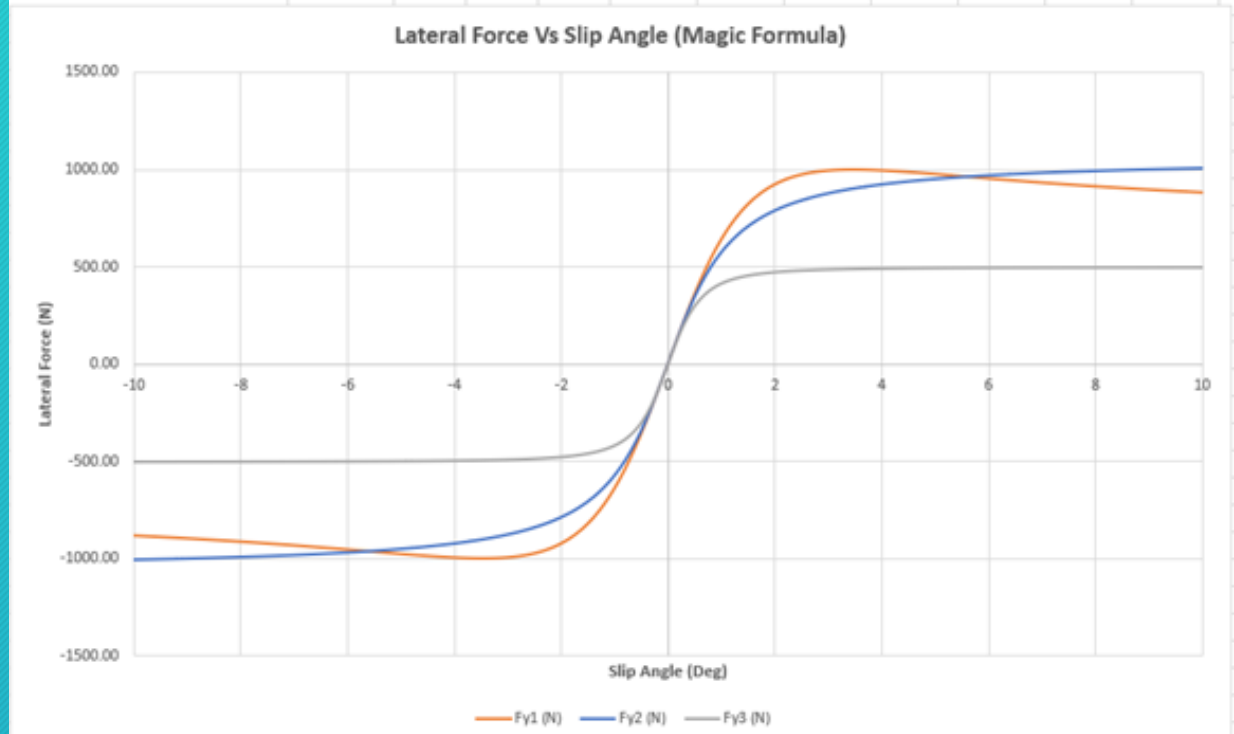
Vehicle Dynamics Calculator

Continued

Excel Project

Lateral Forces & Slip Angle

	Parameters	Fy1 (N)	Fy2 (N)	Fy3 (N)
Magic Formula Base Parameters	B	0.50	1.00	1.50
	C	1.50	0.50	1.00
	D	1.00	1.50	0.50
Vertical Force (N)	Fz	1000.00	1000.00	1000.00



Slip Angle (deg)	Lateral Force Fy1 (N)	Lateral Force Fy2 (N)	Lateral Force Fy3 (N)
-10	-882.66	-1006.51	-498.89
-9.9	-884.03	-1005.95	-498.87
-9.8	-885.41	-1005.38	-498.85
-9.7	-886.81	-1004.80	-498.82
-9.6	-888.24	-1004.21	-498.80
-9.5	-889.68	-1003.61	-498.77
-9.4	-891.14	-1002.99	-498.75
-9.3	-892.63	-1002.36	-498.72
-9.2	-894.13	-1001.72	-498.69
-9.1	-895.66	-1001.06	-498.66
-9	-897.21	-1000.38	-498.63
-8.9	-898.78	-999.69	-498.60
-8.8	-900.37	-998.99	-498.57
-8.7	-901.98	-998.27	-498.54
-8.6	-903.62	-997.53	-498.50
-8.5	-905.28	-996.77	-498.47
-8.4	-906.96	-996.00	-498.43
-8.3	-908.67	-995.20	-498.39
-8.2	-910.40	-994.39	-498.36
-8.1	-912.15	-993.56	-498.32
-8	-913.93	-992.70	-498.27
-7.9	-915.73	-991.83	-498.23
-7.8	-917.56	-990.93	-498.18
-7.7	-919.41	-990.01	-498.14
-7.6	-921.29	-989.06	-498.09
-7.5	-923.19	-988.09	-498.04
-7.4	-925.12	-987.09	-497.98
-7.3	-927.07	-986.06	-497.93
-7.2	-929.04	-985.01	-497.87
-7.1	-931.04	-983.92	-497.81
-7	-933.07	-982.80	-497.75
-6.9	-935.12	-981.65	-497.68
-6.8	-937.19	-980.47	-497.61

Vehicle Dynamics Calculator

Continued

Excel Project

Intial Parameters:

Scrub radius	10	mm
Kingpin inclination angle	0	deg
Camber Angle	0	deg
Front track	1600	mm
Ride Height	35	mm
Chassis Height Vertical	300	mm

Left Upright Width large	70	mm
Left Upright Width Small	20	mm
Left Upright Height Large	305	mm
Left Upright Height Small	152.5	mm

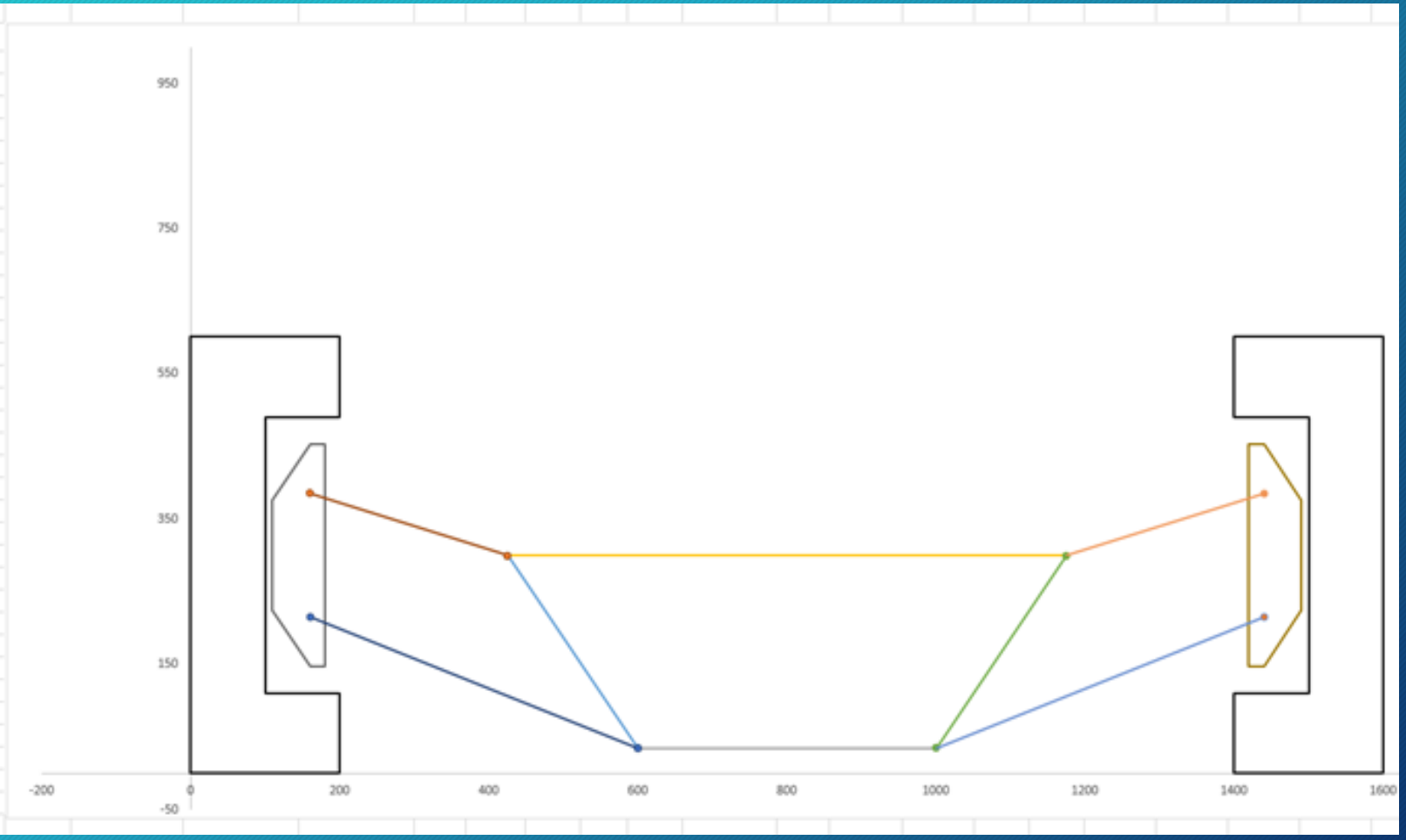
	X	Y	
Left Upright top Control mount point	50	85	mm
Left Upright Lower Control mount point	50	-85	mm

Right Upright Width large	70	mm
Right Upright Width Small	20	
Right Upright Height Large	305	mm
Right Upright Height Small	152.5	mm

	X	Y	mm
Right Upright top Control mount point	-50	85	
Right Upright Lower Control mount point	-50	-85	

Chassis Width Lower	400	
	x	Y
	1000	35
	600	35

Chassis Width Upper	750	
	x	Y
	1175	300
	425	300



Dynamically Updating 2D Suspension

Vehicle Dynamics Calculator

Continued

Excel Project

Backround Calculations

Left Control Arms

Original

	x	y
Lower outer suspension hard point	160.00	215.50
Lower inner forward suspension hard point	600.00	35.00

Rotated

x	Y
160	385.5
160	215.5

Translate 110 300.5

Original

	x	y
Upper outer suspension hard point	160.00	385.50
Upper inner forward suspension hard point	425.00	300.00

Right Control Arm

Original

	x	y
Lower outer suspension hard point	1175.00	300.00
Lower inner forward suspension hard point	1440.00	385.50

Rotated

x	Y
1440	385.5
1440	215.5

Translate 1490 300.5

Original

	x	y
Upper outer suspension hard point	1000.00	35.00
Upper inner forward suspension hard point	1440.00	215.50

Left Wheel

Original		Rotated	
x	y	x	y
-100	-300.5	0	0
100	-300.5	200	0
100	-190.50	200	110
0	-190.50	100	110
0	190.50	100	491
100	190.50	200	491
100	300.50	200	601
-100	300.50	0	601
-100	-300.5	0	0
Translate		100	300.5

Right Wheel

Original		Rotated	
x	y	x	y
-100	-300.5	1400	0
100	-300.5	1600	0
100	300.5	1600	601
-100	300.5	1400	601
-100	190.5	1400	491
0	190.5	1500	491
0	-190.5	1500	110
-100	-190.5	1400	110
-100	-300.5000	1400	0
Translate		1500	300.5

Left Upright

original		Rotated	
x	Y	x	Y
0	-76.25	110	224.25
50	-152.5	160	148
70	-152.5	180	148
70	152.5	180	453
50	152.5	160	453
0	76.25	110	376.75
0	-76.25	110	224.25
76.25	52.5	186.25	353
Translate		110	300.5

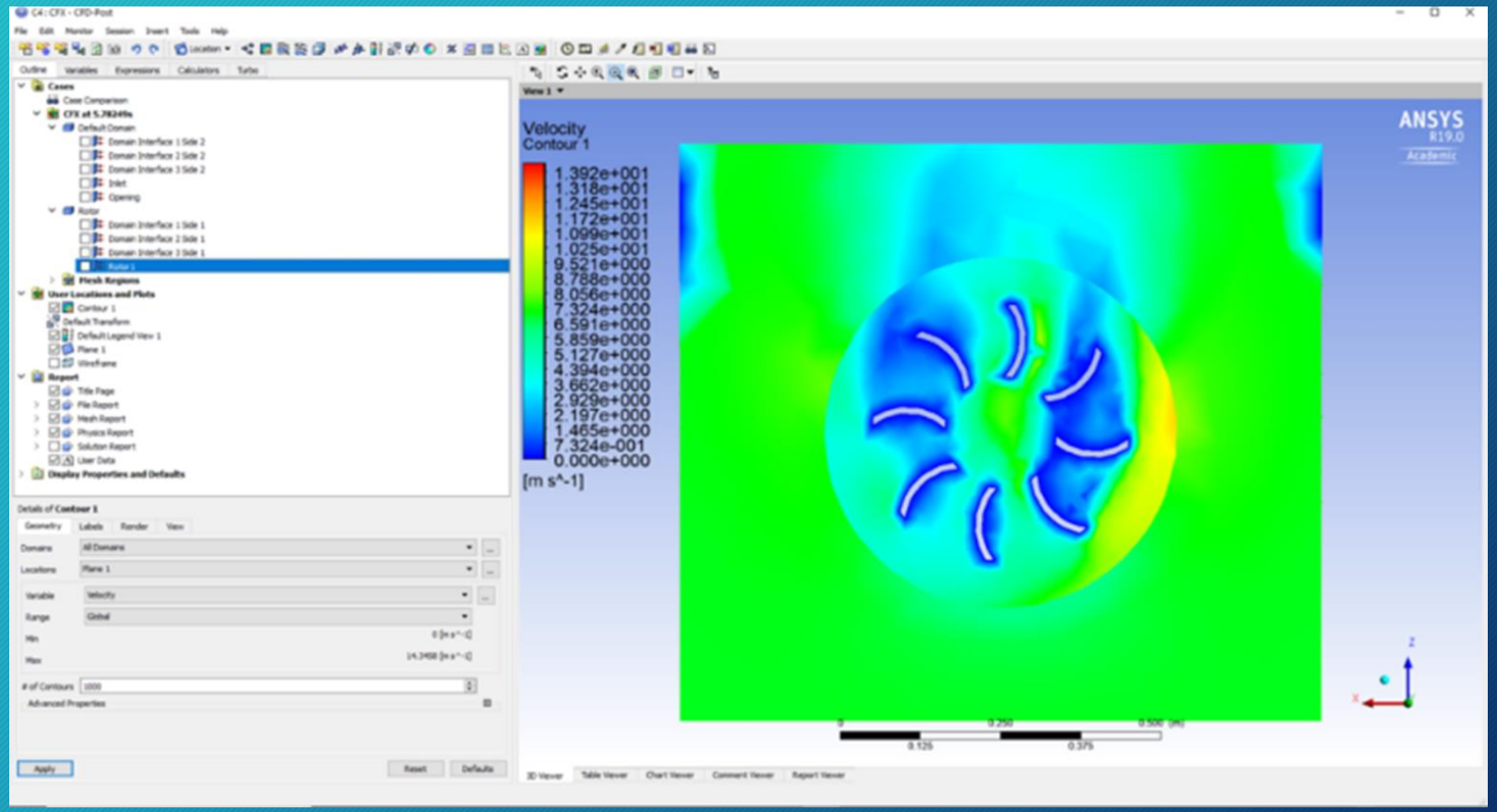
Right Upright

original		Rotated	
x	Y	x	Y
0	-76.25	1490	224.25
50	-152.5	1440	148
70	-152.5	1420	148
70	152.5	1420	453
50	152.5	1440	453
0	76.25	1490	376.75
0	-76.25	1490	224.25
76.25	52.5	186.25	353
Translate		1490	300.5

Vertical Wind Turbine Simulation

ANSYS Simulation Project

- **Software:** ANSYS, Solidworks
- **Aim:** This is an ANSYS simulation where I was studying the effects of the air, surrounding a rotating VAWT by a specified RPM. This was a dynamic 3D CFX simulation using the transient blade rotor function, thus allowing me to easily visualise the effects of the pressure and velocity of the surrounding air.
- **Challenges:** The main challenge that I faced was learning how to use the software and the specific pre-processing and setup steps before running the simulation.
- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox that I was using and what the requirements were for a successful simulation. Through this project, I was constantly learning new concepts and theories which would be useful for my university projects and future career.

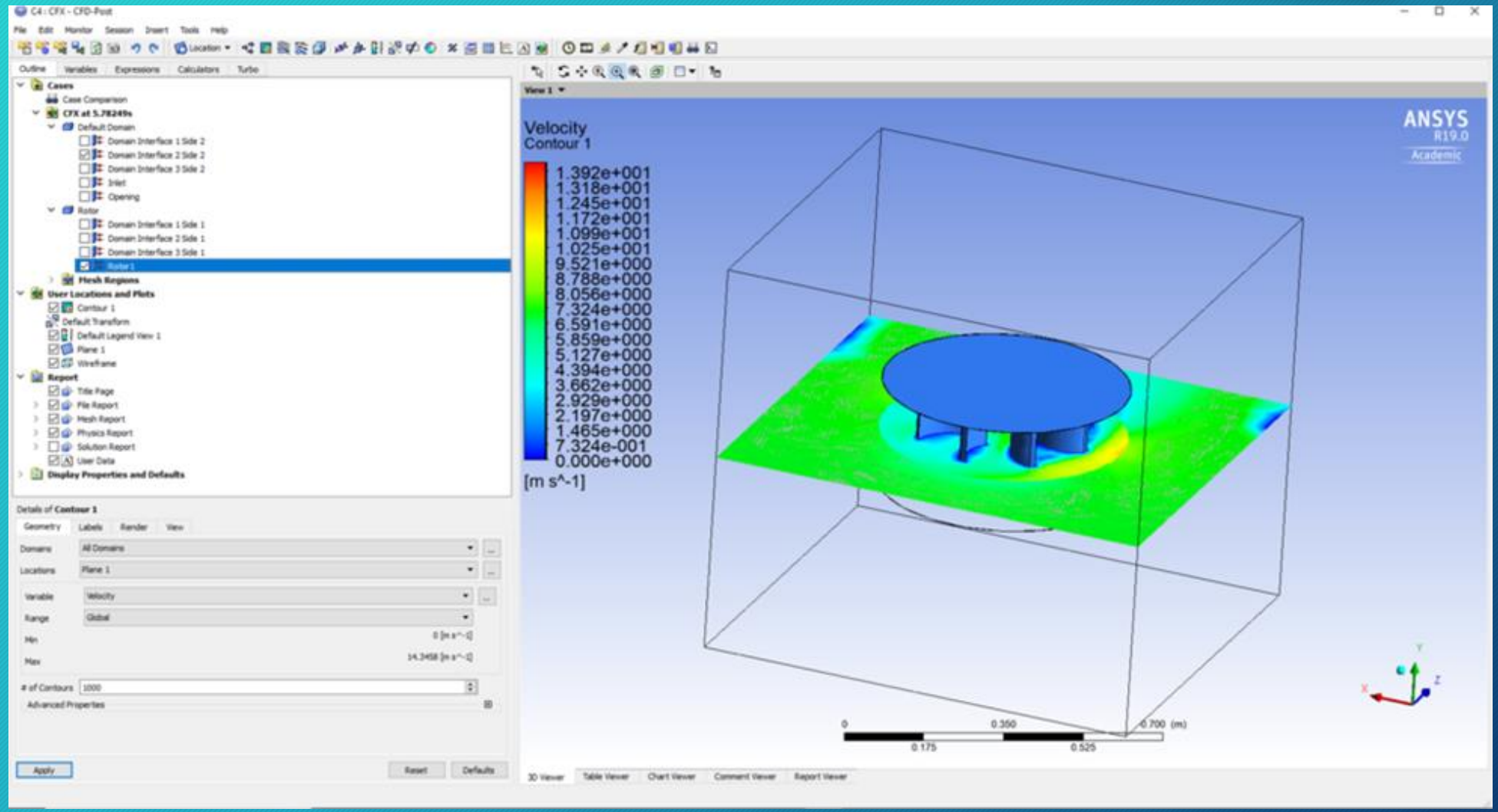


Simulation Results 2D View

Vertical Wind Turbine Simulation

Continued

ANSYS Simulation Project

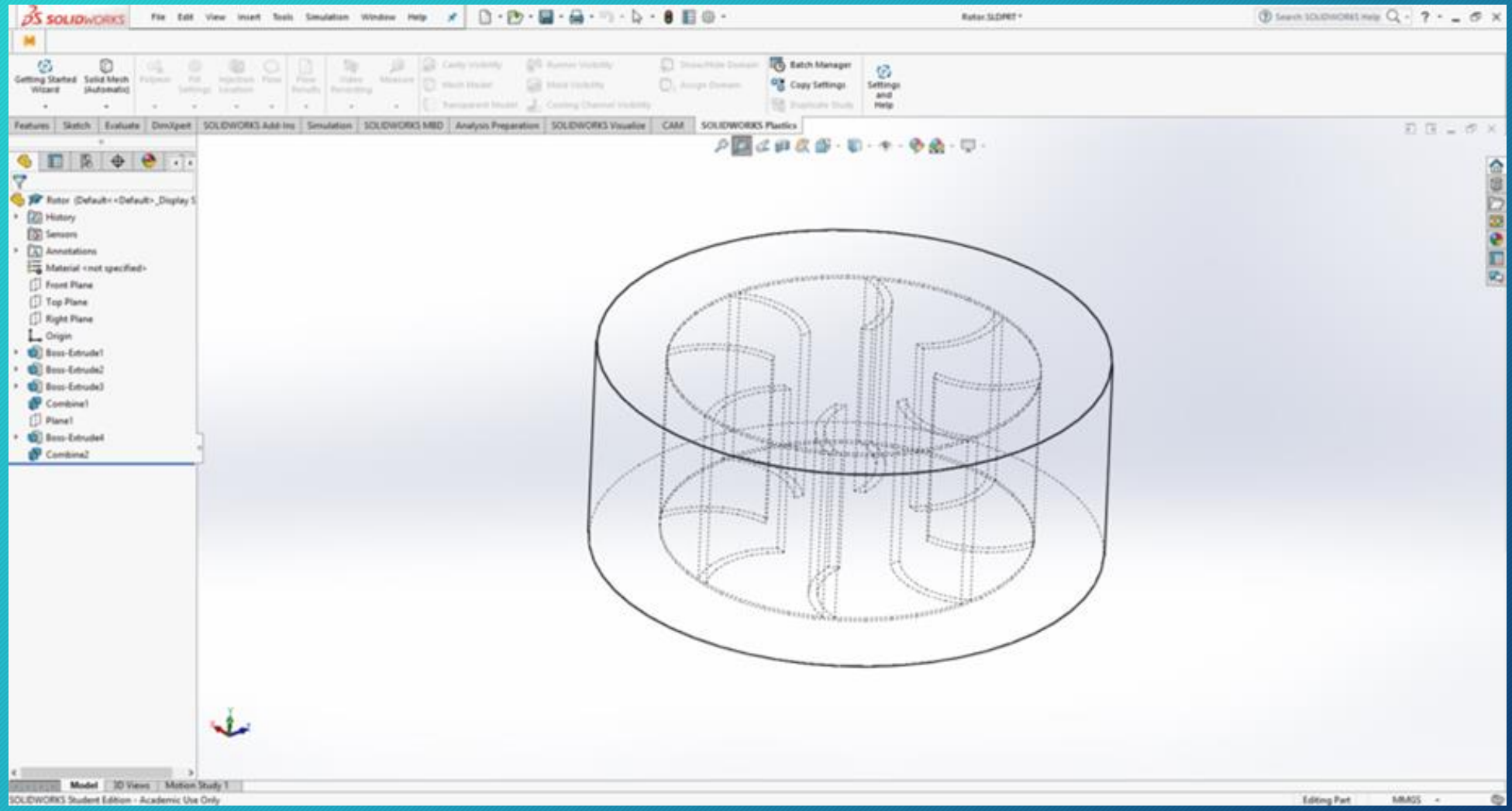


Simulation Results 3D View

Vertical Wind Turbine Simulation

Continued

ANSYS Simulation Project

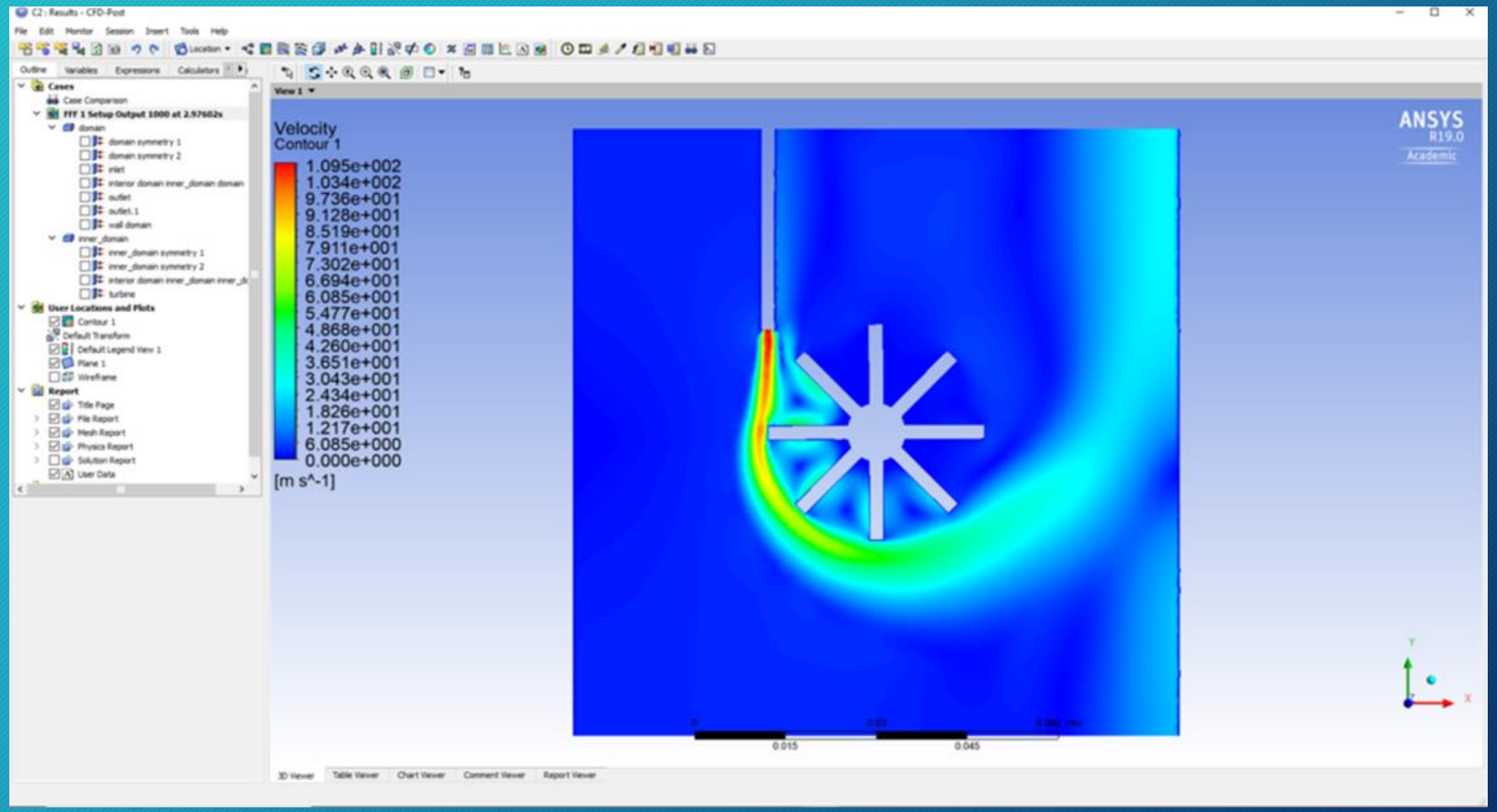


Solidworks Model

Fluent 6DOF Induced Wheel Spin

ANSYS Simulation Project

- **Software:** ANSYS, Solidworks
- **Aim:** This was an ANSYS simulation where I was studying the spin of a wheel induced by a specified inlet velocity. During this simulation, I used the dynamic mesh tool which was one of the three choices one has in ANSYS Fluent. Unlike my prior simulation where I used the sliding mesh technique, this was a new method of using ANSYS fluent and a different approach to a problem.
- **Challenges:** The main challenge that I faced was learning how to use the software and the specific pre-processing and setup steps before running the simulation.
- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox that I was using and what the requirements were for a successful simulation. Through this project, I was constantly learning new concepts and theories which would be useful for my university projects and future career.

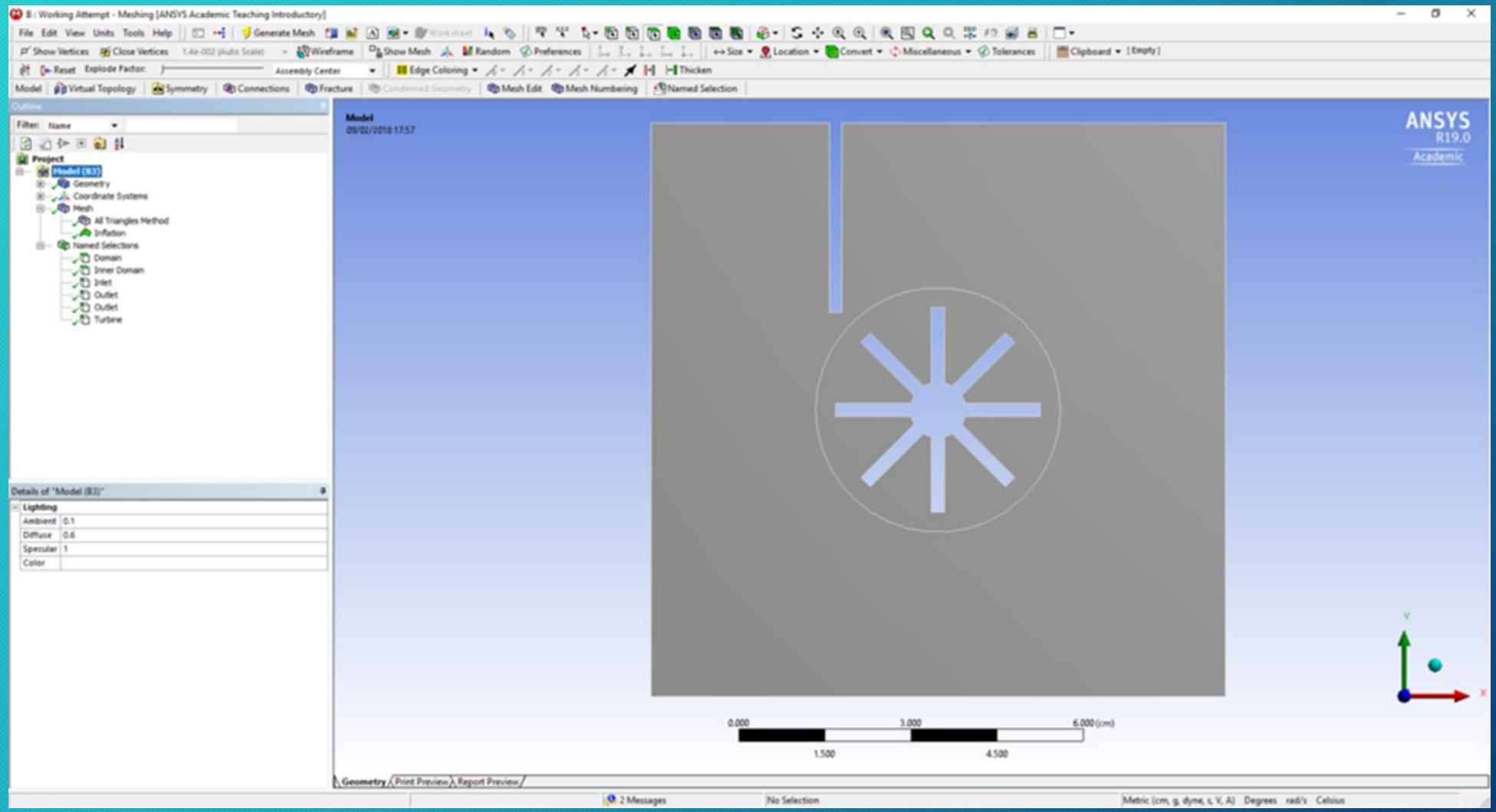


Fluent 6DOF Induced Wheel Spin Results

Fluent 6DOF Induced Wheel Spin

Continued

ANSYS Simulation Project

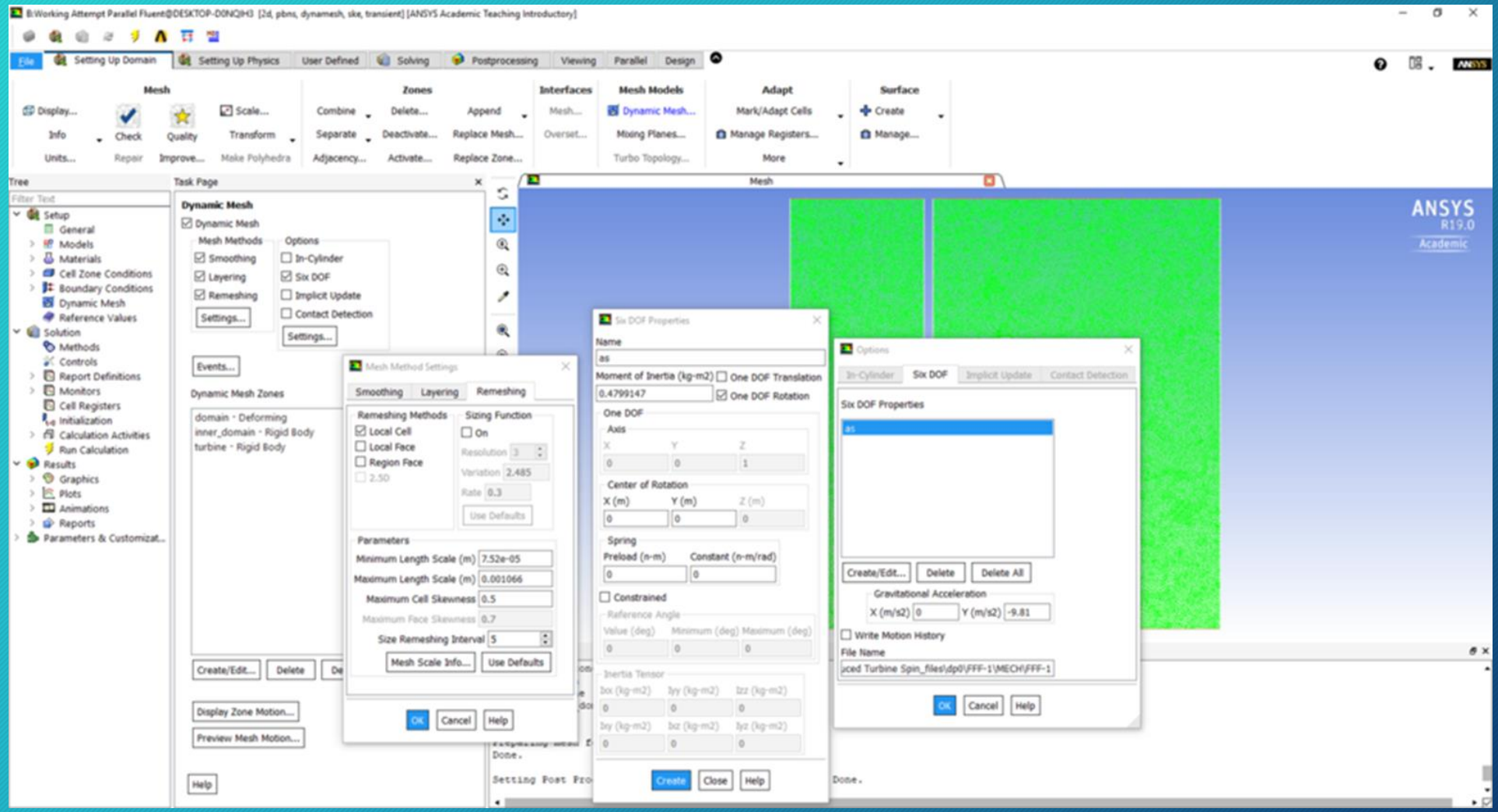


Fluent 6DOF Induced Wheel Spin Setup

Fluent 6DOF Induced Wheel Spin

Continued

ANSYS Simulation Project

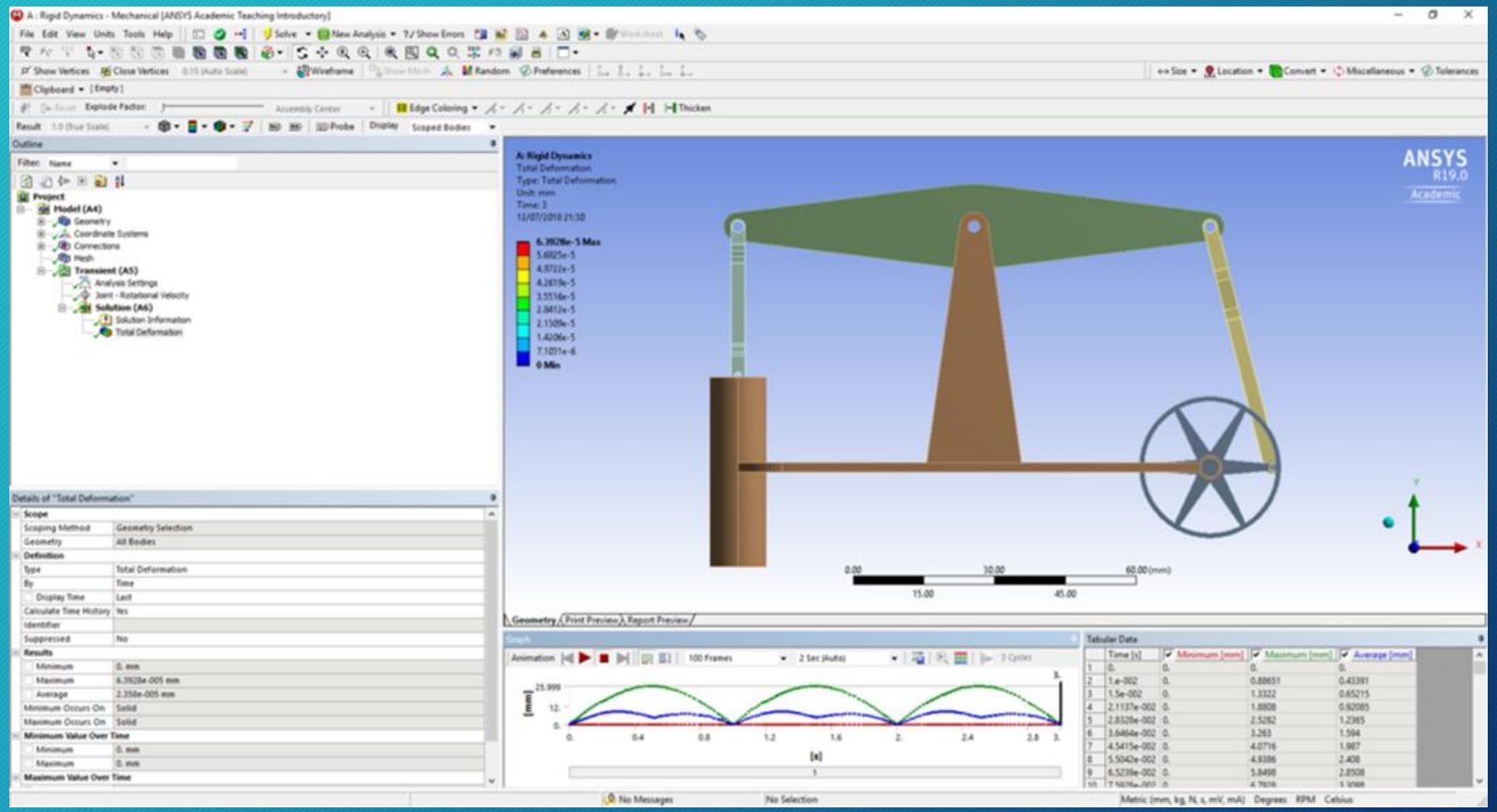


ANSYS 6DOF Setup

Rigid Body Dynamics Beam Engine

ANSYS Simulation Project

- **Software:** ANSYS, Solidworks
- **Aim:** This was an ANSYS rigid dynamic simulation where it was simulating a beam engine. This was a new component of the Ansys software that I was currently learning and I plan to do more complex ones in the future. This method of analysing a mechanism was very handy because during the setup, one had to define every joint and joint type; therefore, allowing one to fully understand the mechanism by having to break it down into its parts and see how they interact with each other.
- **Challenges:** The main challenge that I faced was learning how to use the software and the specific pre-processing and setup steps before running the simulation.
- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful simulation. Through this project, I was constantly learning new concepts and theories which would be useful for my university projects and future career.

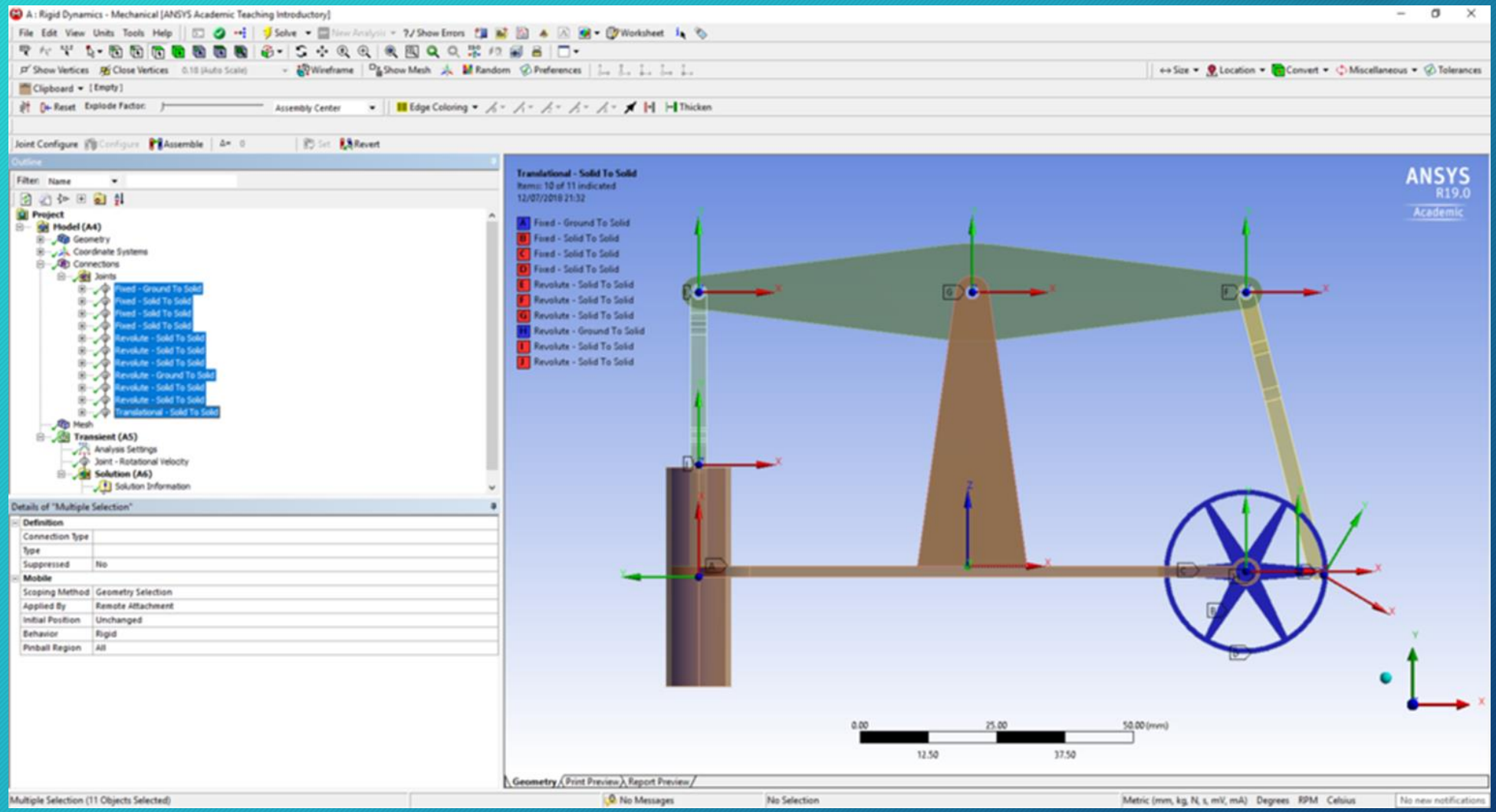


ANSYS Rigid Body Dynamics Beam Engine Results

Rigid Body Dynamics Beam Engine

Continued

ANSYS Simulation Project



ANSYS Defined Joints

Explicit Dynamic Impact

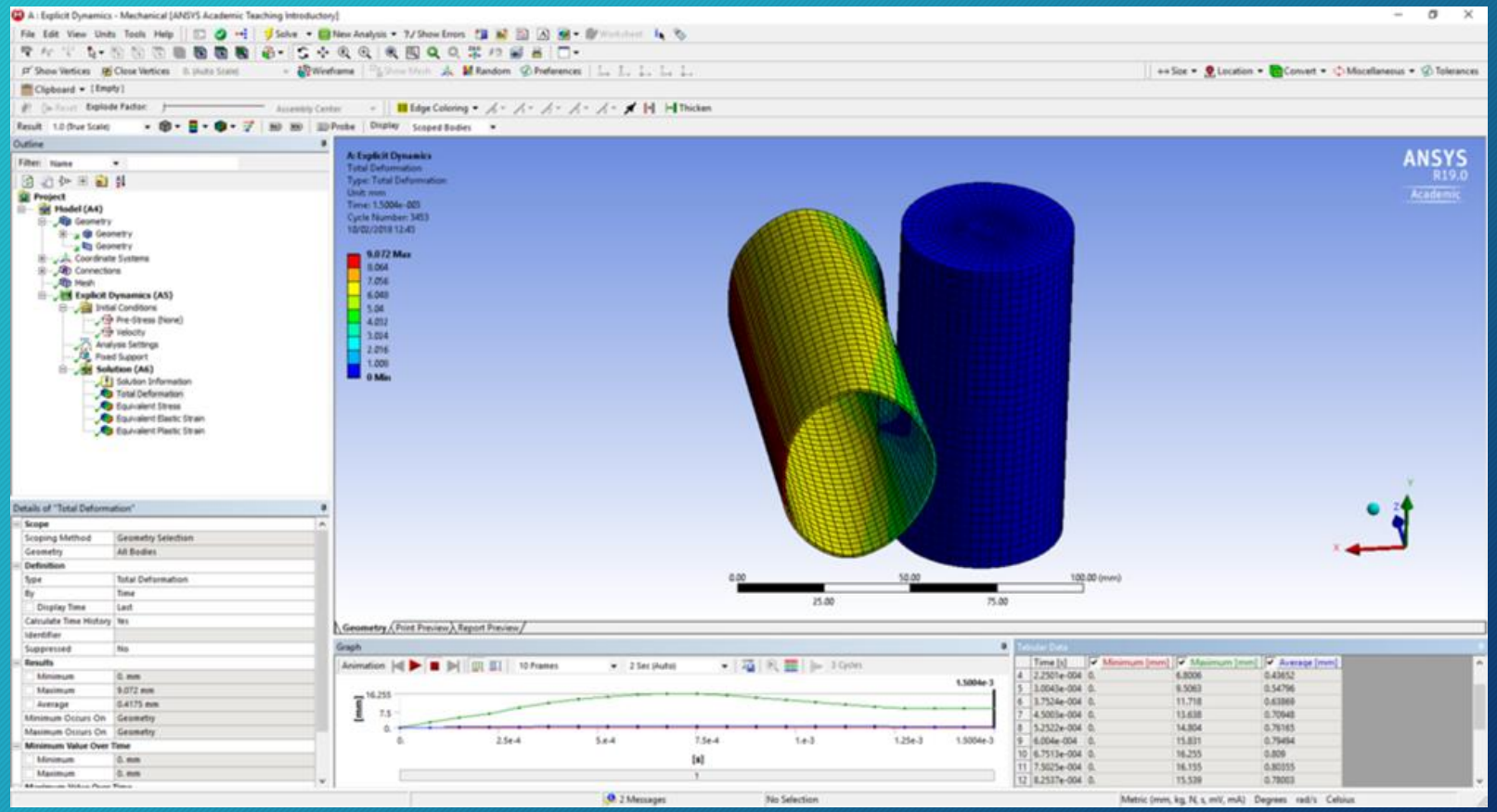
ANSYS Simulation Project

- **Software:** ANSYS, Solidworks

- **Aim:** This was an ANSYS simulation where I was studying the impact of an aluminium tube when it smashes into a support structure. During this simulation, I added the plastic strain failure parameter from the material toolbox which added a point of failure for the material. This clarified whether the aluminium tube would fail after the collision. This was a new method of using ANSYS Explicit Dynamic and a different approach to a problem.

- **Challenges:** The main challenge that I faced was learning how to use the software and the specific pre-processing and setup steps before running the simulation.

- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful simulation. Through this project, I was constantly learning new concepts and theories which would be useful for my university projects and future career.

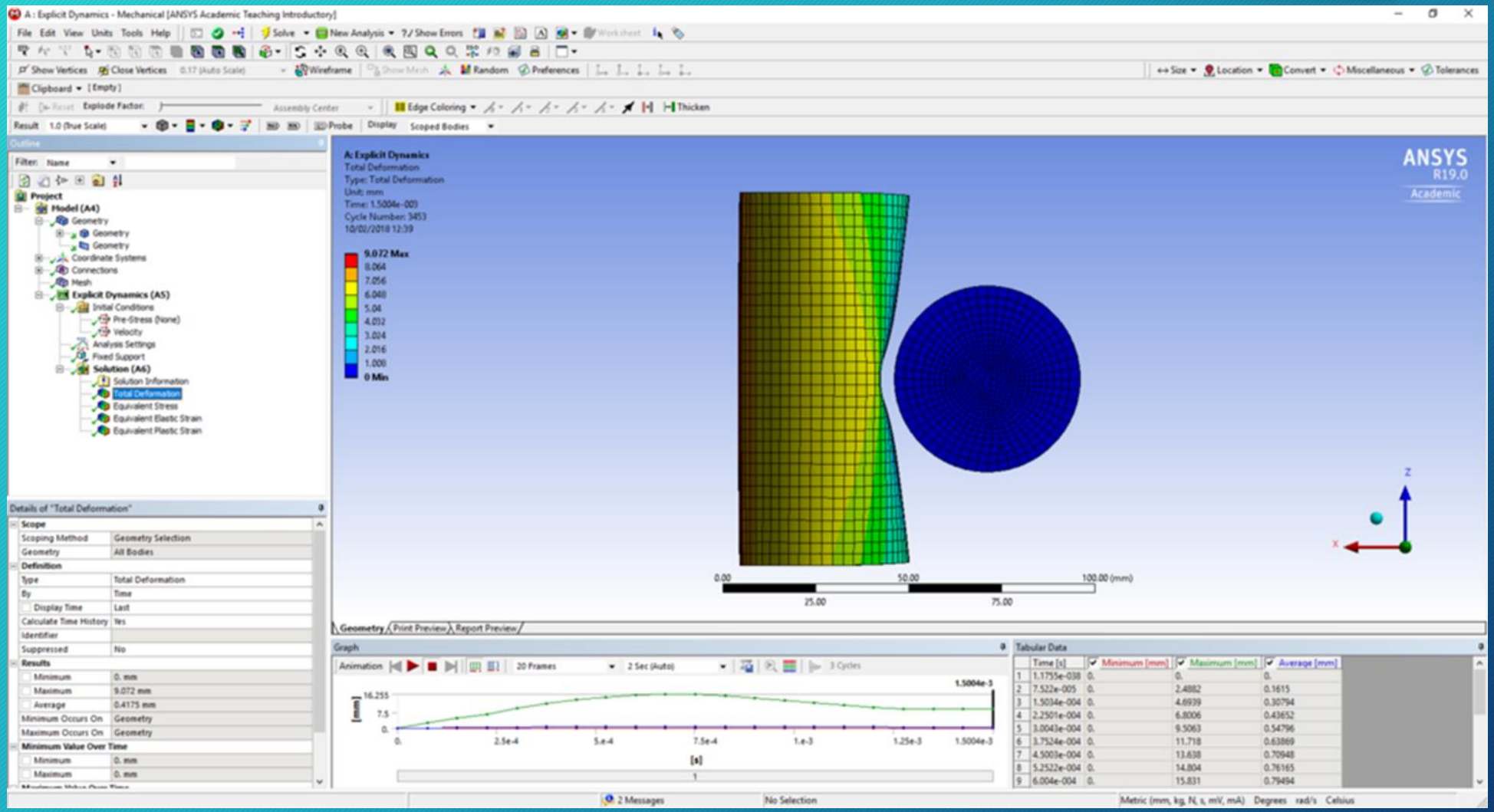


ANSYS Explicit Dynamic Impact Results 1

Explicit Dynamic Impact

Continued

ANSYS Simulation Project



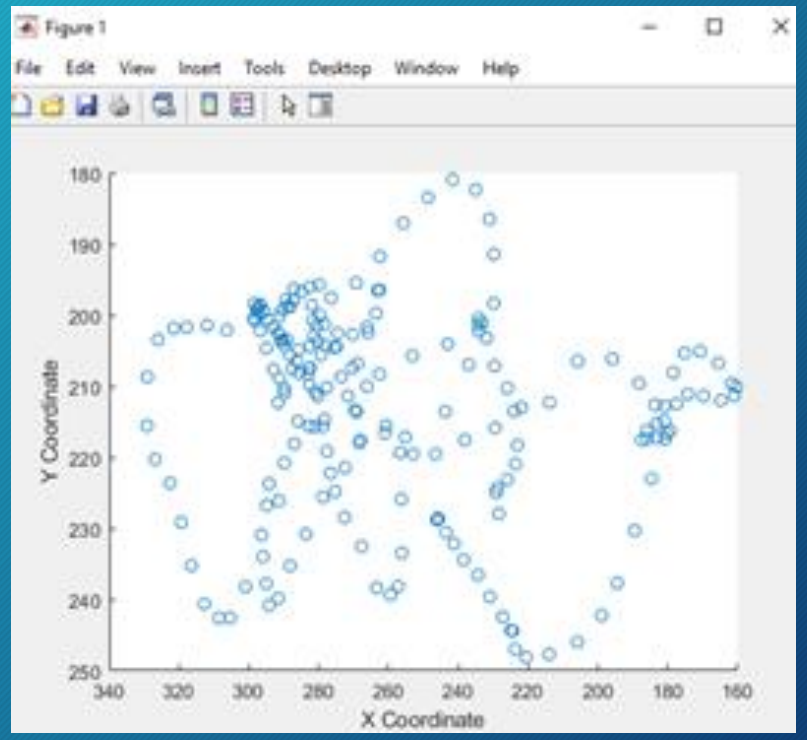
ANSYS Explicit Dynamic Impact Results 2

Colour Tracking Algorithm

MATLAB Project

- **Software:** Matlab
- **Aim:** To develop an algorithm to track a specified colour and using live image acquisition to locate the point of centre of mass and plot a circle. As the program ran over time, the centre of mass coordinate was plotted on a dynamic graph.
- **Challenges:** The main challenge that I faced was learning how to use the computer vision toolbox within MATLAB software and the specific functions associated with it.
- **Solution:** : I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful program.

Colour Tracking Real Time



Colour Tracking Plot

Colour Tracking Algorithm

Continued

MATLAB Project

WebcamTrack.m

```
%%Clean up any left over variables

clear,clc

% Create a video device to acquire data from a camera
if ~exist('myCam','var')
  myCam = imaq.VideoDevice('winvideo',1,'MJPG_640x360','ReturnedDataType','uint8');
end

% Create video player for visualization
vidPlayer = vision.DeployableVideoPlayer;
%Edge_Detection = vision.DeployableVideoPlayer;

% Define the time for the simulation
Time = 200;


%% LOOP
for i = 1:Time
    % Grab images
    img = step(myCam);
    resizeScale = 0.5;

    %% Load all the color models
    [B_centerX,B_centerY,B_circleSize] = DetectPenBlue(img,resizeScale);
    %[Y_centerX,Y_centerY,Y_circleSize] = DetectPenYellow(img,resizeScale);
    %[G_centerX,G_centerY,G_circleSize] = DetectPenGreen(img,resizeScale);

    %% VISUALIZE Color

     B_img = insertObjectAnnotation(img,'circle',[B_centerX B_centerY B_circleSize/2],'Blue thing','LineWidth',4,'Color','Blue','TextColor','black','FontSize',20,'TextBoxOpacity',0);
     %Y_img = insertObjectAnnotation(img,'circle',[Y_centerX Y_centerY Y_circleSize/2],'Yellow thing','LineWidth',4,'Color','Yellow','TextColor','black','TextColor','black','FontSize',20
     %G_img = insertObjectAnnotation(img,'circle',[G_centerX G_centerY G_circleSize/2],'Green thing','LineWidth',4,'Color','Green','TextColor','black','TextColor','black','FontSize',20,'

    %% Update the video player
    step(vidPlayer,B_img);
    %step(vidPlayer,Y_img);
    %step(vidPlayer,G_img);

    %Edge Detection
    %Edge_Detect = edge(rgb2gray(img),'canny');
    %step(Edge_Detection,Edge_Detect);
```

Colour Tracking Code

Automated Synthetic Training Data Generator

MATLAB Project

- **Software:** Matlab
- **Aim:** To develop different programs to generate synthetic training data to train Machine Learning algorithms.
  - Program 1: To split an input video into separate frames.
  - Program 2: To load all the images from program 1 and using a “for loop” to rename all the images and save them to a new folder.
  - Program 3: Using various image processing techniques, the program first takes an image, then re-sizes and randomly rotates it, then randomly translates the image and finally saves it into a new folder. Then, using pre-defined labels, it concatenates the images and the relevant labels into a training data folder which can be parsed into the function to train the Machine Learning algorithm.
- **Challenges:** The main challenge that I faced was learning how to use the computer vision toolbox within MATLAB software and the specific functions associated with it.
- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful program.

Program 1: Video Frame Extraction

VF_Extraction.m | Loop_Rename_IMG.m | Multi_Image_Processing.m

```
%% Extract the frames from a video

%Load the video into the script
[file,path] = uigetfile({"*.avi";"*.MP4"});
Vid = VideoReader(file);

% User selects the path to save the images
S_folder = uigetdir('C:\');

%%For loop to iterate through the video
for i = 1:Vid.NumberOfFrames
    filename = strcat('Image_frame',' ',num2str(i),'.jpg');
    b = read(Vid,i);
    fullFileName = fullfile(S_folder, filename);
    imwrite(b, fullFileName);
end
```

Program 1 Code

Command Window

```
Image_frame326.jpg
Image_frame327.jpg
Image_frame328.jpg
Image_frame329.jpg
Image_frame330.jpg
Image_frame331.jpg
Image_frame332.jpg
Image_frame333.jpg
Image_frame334.jpg
Image_frame335.jpg
Image_frame336.jpg
Image_frame337.jpg
Image_frame338.jpg
Image_frame339.jpg
Image_frame340.jpg
Image_frame341.jpg
Image_frame342.jpg
Image_frame343.jpg
fx >>
```

Program 1 Console Output

Automated Synthetic Training Data Generator

Continued

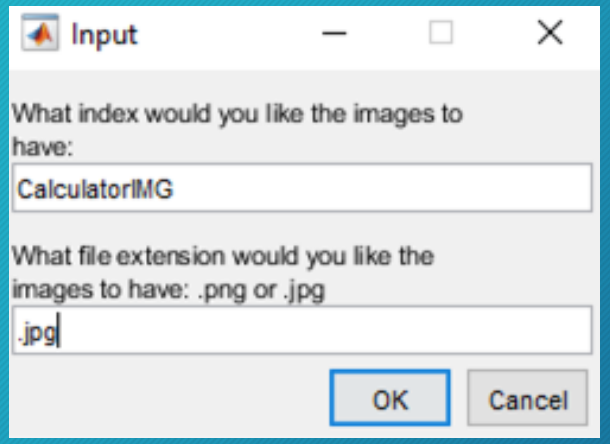
MATLAB Project

Program 2: Image Rename

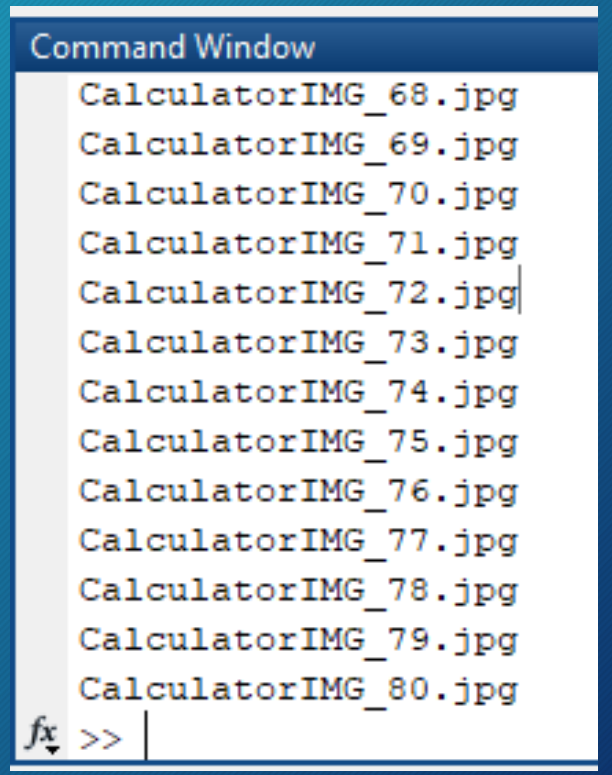
VF_Extraction.m | Loop_Rename_IMG.m | Multi_Image_Processing.m

```
%% User input Master file
uiwait(msgbox('Pick a starting folder'));
folder1 = uigetdir;
fileInfo1 = dir(fullfile(folder1, '*.png'));
%% User input Save file
uiwait(msgbox('Pick a save folder'));
S_folder = uigetdir('C:\');

PromptA = 'What index would you like the images to have: ';
PromptB = 'What file extension would you like the images to have: .png or .jpg';

prompt = {PromptA,PromptB};
dlgtitle = 'Input';

dims = [1 40];
answer = inputdlg(prompt,dlgtitle,dims);
Ans2 = string(answer(2));

%% Loop to load , Rename , saved
for k = 1 : length(fileInfo1)
    thisFilename = fullfile(fileInfo1(k).folder, fileInfo1(k).name);
    load_images{k} = imread(thisFilename);
    index = num2str(k);
    Str = [answer(1),index];
    String = strjoin(Str,'_');
    filename = strcat(String,Ans2);
    fullFileName = fullfile(S_folder, filename);
    imwrite(load_images{k},fullFileName);
    disp(filename)
end
```

Program 2 Code



Program 2 User Input



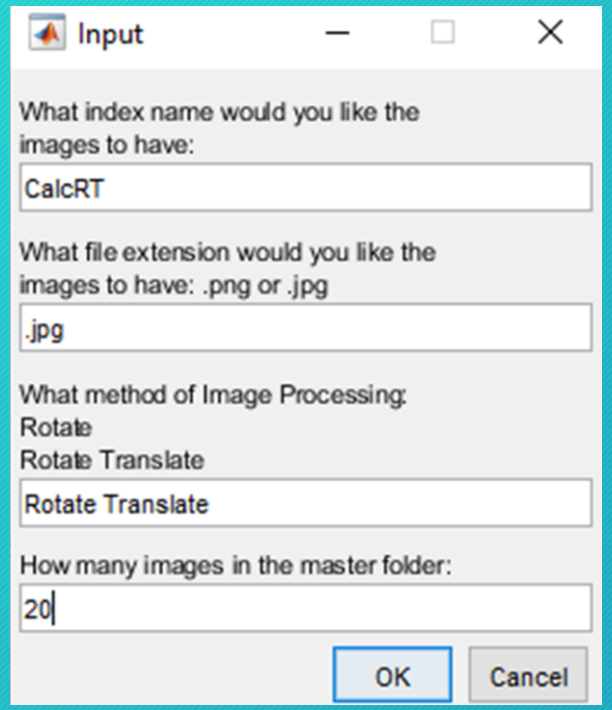
Program 2 Console Output

Automated Synthetic Training Data Generator

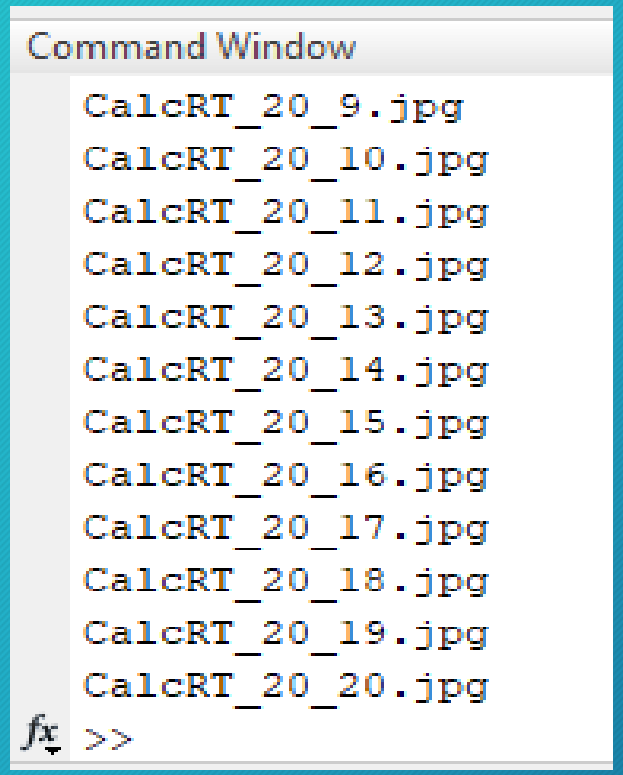
Continued

MATLAB Project

Program 3: Image Processing



Program 3 User Input



Program 3 Console Output

Program 3 Code

Transfer Learning Image Classification Algorithm using a CNN Model

MATLAB Project

- **Software:** Matlab
- **Aim:** To develop CNN Machine Learning algorithm to read an image, classify objects within that image and then output what that object is.
- **Challenges:** The main challenge that I faced was learning how to use the computer vision toolbox within MATLAB software and the specific functions associated with it.
- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful program.

CNN.m

```
%% Clear the workspace of all variables
clear
clc
%% Load AlexNet
net  = alexnet;
layers = net.Layers;
inlayer = net.Layers(1);
insz = inlayer.InputSize;

Neurons = 1;
%% Create a Data store of all the images
imds = imageDatastore("Video Frame Extraction",'IncludeSubfolders',true,'LabelSource','foldernames','FileExtensions', {'.jpg'});
fname = imds.Labels;
numTrainFiles = length(fname);
%% Split the data into training and testing
[TrainIMG,TestIMG] = splitEachLabel(imds,0.7);

%% Resize all the images to a suitable size
augTrain = augmentedImageDatastore([227 227],TrainIMG);
augTest = augmentedImageDatastore([227 227],TestIMG);
%% Adjust the new network Arcitecture
fc = fullyConnectedLayer(Neurons);
layers(23) = fc;
layers(end) = classificationLayer;

options = trainingOptions('sgdm','InitialLearnRate',0.01,...
    'Plots','training-progress',...
    'Shuffle','every-epoch');

%% Train the new network and validate
[newnet,info] = trainNetwork(augTrain,layers,options);

%% Test the CNN
NetPreds = classify(newnet,augTest);

plotconfusion(TestIMG.Labels,NetPreds)

save('CNN_Network','newnet')
```

CNN Model Code

Transfer Learning Image Classification Algorithm using a RCNN Model

MATLAB Project

- **Software:** Matlab

- **Aim:** To develop RCNN Machine Learning algorithm to read an image, classify objects within that image and then output what that object is and where it is located within the image.

- **Challenges:** The main challenge that I faced was learning how to use the computer vision toolbox within MATLAB software and the specific functions associated with it.

- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful program.

RCNN.m | Untitled2*

```
%% Clear the workspace of all variables
clear
clc
%% Load AlexNet
net  = alexnet;
layers = net.Layers;
inlayer = net.Layers(1);
insz = inlayer.InputSize;

%% Load gTruth data and Add the image directory to the MATLAB path.
load ('Training Data.mat')

%% Adjust the new network Arcitecture
Neurons = width(Training_Data);
fc = fullyConnectedLayer(Neurons);
layers(23) = fc;
layers(end) = classificationLayer;

options = trainingOptions('sgdm','InitialLearnRate',0.001,'MiniBatchSize',1,'MaxEpochs',10,'ExecutionEnvironment','auto');

%% Train the new network and validate
detector = trainFastRCNNObjectDetector(Training_Data,layers,options)
```

RCNN Model Code

Using either the CNN or RCNN with Live Image Acquisition

MATLAB Project

- **Software:** Matlab

- **Aim:** To develop a script which uses live image acquisition and a “For loop” to feed data into a Machine Learning model. If the CNN is used, there will be a title of object class suggesting the model has been classified . If the RCNN is used, a labelled bounding box will surround the object and track it in real time.

- **Challenges:** The main challenge that I faced was learning how to use the computer vision toolbox within MATLAB software and the specific functions associated with it.

- **Solution:** I followed a useful tutorial online and read different PowerPoint presentations around the specific toolbox I was using and what the requirements were for a successful program.

WebcamTrack.m

```
%% Clean up any left over variables
clear,clc
%% Load the CNN
load CNN_Network
%% Create a video device to acquire data from a camera
if ~exist('myCam','var')
  myCam = imaq.VideoDevice('winvideo',1,'MJPG_640x360','ReturnedDataType','uint8');
end

% Create video player for visualization
vidPlayer = vision.DeployableVideoPlayer;

%% Define the time for the simulation
Time = 200;

%% Loop to Run CNN in real Time
for i = 1:Time
    img = step(myCam);

    picture=imresize(img,[227,227]);

    label = classify(newnet,picture);
    image(picture);
    title(char(label));
    drawnow;

    step(vidPlayer,img);
    i = 1+i;
end
%% Clean and release
release(myCam)
release(vidPlayer)
```

Real Time CNN Code

WebcamTrack.m

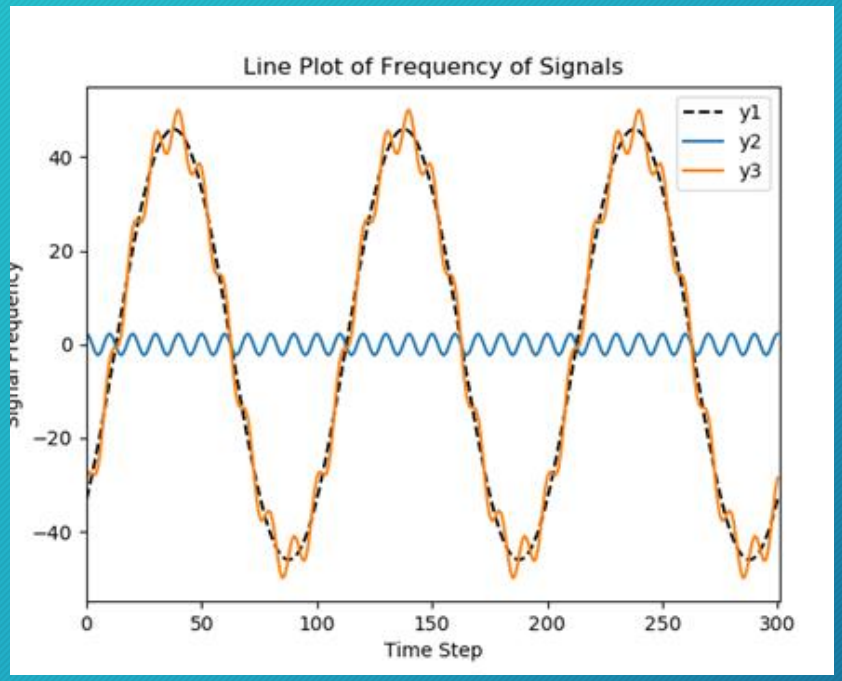
```
%% Clean up any left over variables
clear,clc
%% Load the CNN
load Detector
%% Create a video device to acquire data from a camera
% if ~exist('myCam','var')
%   myCam = imaq.VideoDevice('winvideo',1,'MJPG_320x240','ReturnedDataType','uint8');
% end

vidDevice = imaq.VideoDevice('winvideo', 1, 'YUY2_320x240', ...
                             'ReturnedColorSpace', 'rgb', ...
                             'DeviceProperties.Brightness', 130, ...
                             'DeviceProperties.Sharpness', 7);

% Create video player for visualization
vidPlayer = vision.DeployableVideoPlayer;

%% Loop to Run CNN in real Time
tic

while toc < 10
%     img = step(myCam);
    img = vidDevice();
%     img = imresize(img,[227,227]);
    [bbox, score, label] = detect(detector, img);
    annotation = ['Confidence: ' num2str(round((score*100),2)),'%'];
    detectedImg = insertObjectAnnotation(img, 'rectangle', bbox, annotation,...
        'TextBoxOpacity',0.9,'FontSize',18);
    step(vidPlayer,detectedImg);
end
%% Clean and release
release(myCam)
release(vidDevice)
```

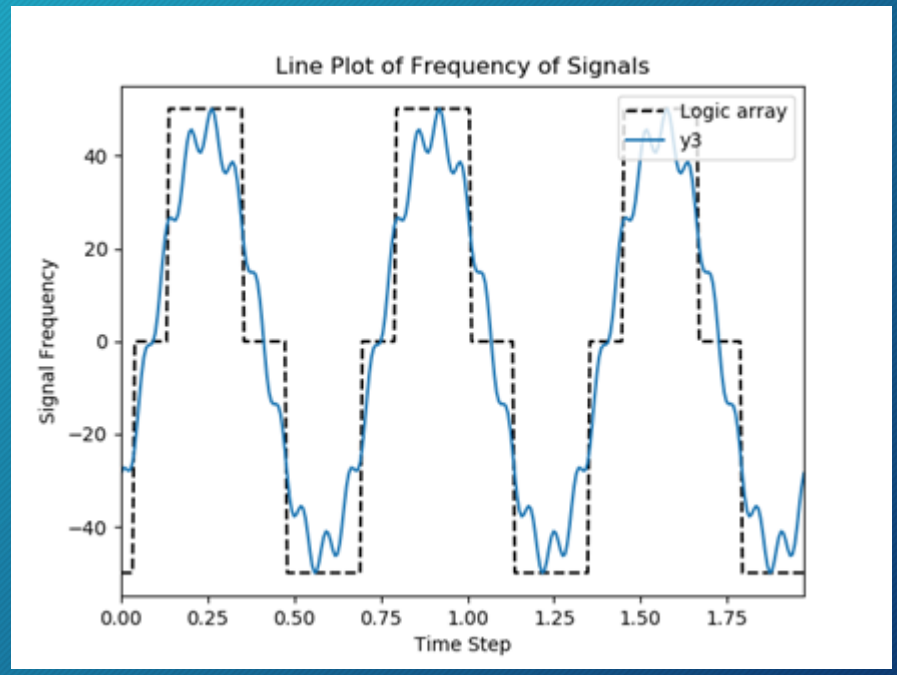
Real Time RCNN Code

Engineering Coursework in Python

Python Project

- **Software:** Python
- **Aim:** I was given programming problems that I was required to complete in MATLAB. However, as a personal challenge, I completed the course work a second time using Python.
    1. Task one was to use loops, functions and plots based off predefined equations.
    2. Task two was to develop a function for matrix indexing.
- **Challenges:** Since this was my first program in Python, nearly everything was a challenge. However, understanding different data types and the use of loops was particularly challenging.
- **Solution:** I planned out what my program was going to do beforehand and learnt as I went along.





Engineering Coursework in Python

Continued

Python Project

Task 1 Code:

```python
import matplotlib.pyplot as plt
import numpy as np

def function(SID,ts):
    A = sum(SID)
    B = A/20
    T1 = (np.mean(SID)/10)
    T2 = T1/10
    Phi_1 = np.radians(sum(SID))
    Phi_2 = np.radians((np.pi)/4)
    time_step = T2/ts
    times = np.arange(0,3*T1,time_step)

    y1 = []
    y2 = []
    y3 = []
    new = []

    for i, time in enumerate(times):
        y1.append( A * np.sin((((2*np.pi)/T1)*time)-Phi_1) )
        y2.append( B * np.cos((((2*np.pi)/T2)*time)-Phi_2) )

    for (a,b) in zip(y1,y2):
        y3.append( a + 2 * b )

    for values in y3:

        if values >= max(y3)/2:
            values = np.max(y3)
            new.append(values)

        elif np.absolute(values) < np.max(y3)/2:
            values = 0
            new.append(values)

        elif values <= np.min(y3)/2:
            values = np.min(y3)
            new.append(values)


    plt.figure(1)

    plt.plot(y1,"k--",label='y1')
    plt.plot(y2,label='y2')
    plt.plot(y3,label='y3')
    plt.xlabel('Time Step')
    plt.ylabel('Signal Frequency')
    plt.title('Line Plot of Frequency of Signals')
    plt.legend(loc='upper right')
    plt.xlim(0, len(times))

    plt.figure(2)

    plt.plot(times,new,"k--",label='Logic array')
    plt.plot(times,y3,label='y3')
    plt.xlabel('Time Step')
    plt.ylabel('Signal Frequency')
    plt.title('Line Plot of Frequency of Signals')
    plt.legend(loc='upper right')
    plt.xlim(0, np.max(times))

    plt.show()


if __name__ == "__main__":
    SID = np.array([8,5,0,9,6,9,9])
    ts = 10

    function(SID,ts)
```

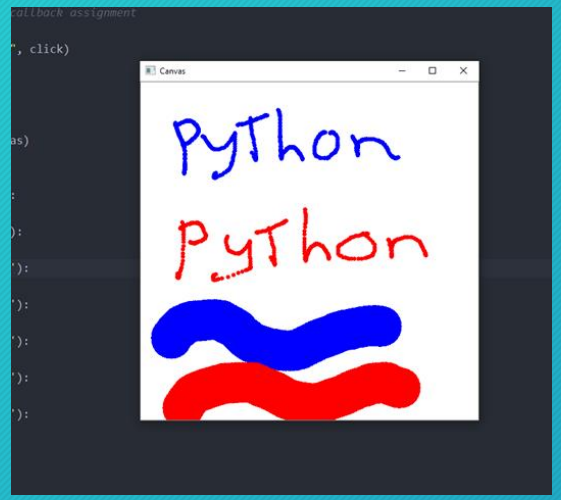
Task 2 Code:

```python
import numpy as np

def Indexing(m_row,m_col,row,col):

    matrix = np.random.randint(5, size=(m_row,m_col))

    print('The randomly generated matrix: \n \n {}'.format(matrix))

    Method_1 = matrix[row-1,col-1]
    M1_string = "The value in row: {} col: {} = {}".format(row,col,Method_1)
    print("")
    print(M1_string)

    Method_2 = []

    for c_val in range(6):
        for r_val in range(4):
            if matrix[r_val,c_val] == matrix[row-1,col-1]:
                matrix[r_val,c_val] = matrix[row-1,col-1]
                Method_2.append(matrix[r_val-1,c_val-1])
            else:
                matrix[r_val,c_val] = 0
                Method_2.append(matrix[r_val,c_val])

    print('')
    print('All the indexed values in the matrix: \n \n {}'.format(matrix))


Indexing(m_row=4,m_col=6,row=4,col=3)
```

Drawing App

Python Project

- **Software:** Python & OpenCV
- **Aim:** To develop a simple drawing GUI using Python.
- **Challenges:** Understanding the key capture events and how to manipulate them to suit my application.
- **Solution:** Followed a similar example online and adapted it to suit my application.



Main Script Code Snippet

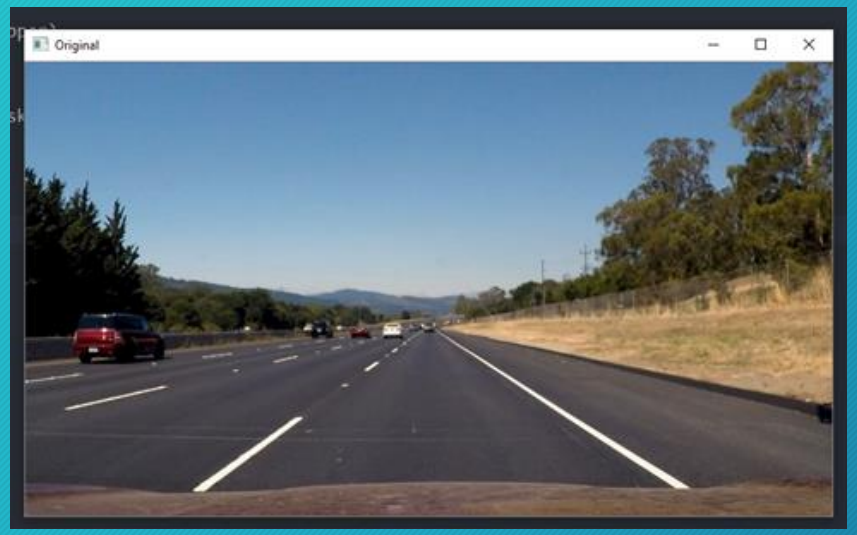
```python
import numpy as np
import cv2

# Global variables
Canvas = np.ones([500,500,4])*255

#circle Data
color = (0,255,0)
line_width = 3
radius = 3
point = (0,0)
pressed = False

# click callback
def click(event, x, y, flags, param):
    global pressed
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(Canvas, (x,y),radius,color,-1)
        pressed = True
    elif event == cv2.EVENT_MOUSEMOVE and pressed == True:
        cv2.circle(Canvas, (x,y),radius,color,-1)
    elif event == cv2.EVENT_LBUTTONUP:
        pressed = False

# window initialization and callback assignment
cv2.namedWindow("Canvas")
cv2.setMouseCallback("Canvas", click)
```

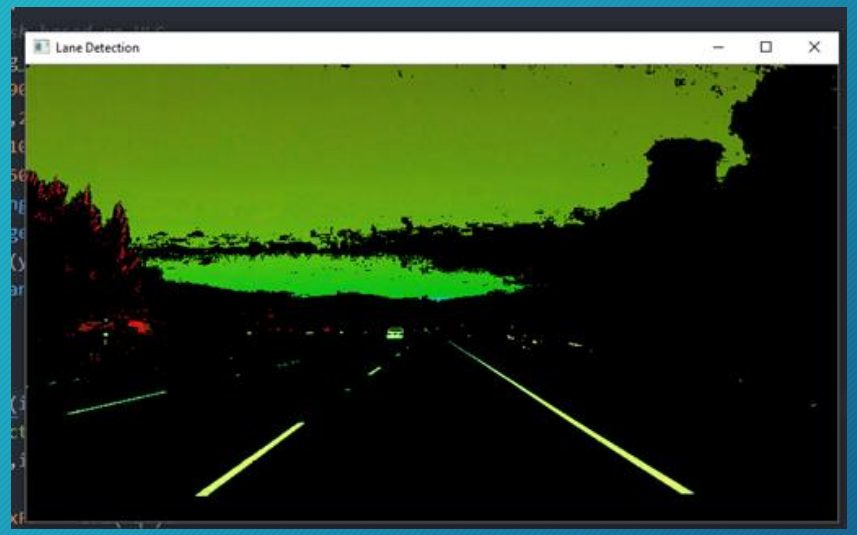
```python
# window initialization and callback assignment
cv2.namedWindow("Canvas")
cv2.setMouseCallback("Canvas", click)

# Forever draw loop
while True:

    cv2.imshow("Canvas",Canvas)
# key capture every 1ms
    ch = cv2.waitKey(1)
    if ch & 0xFF == ord('q'):
        break
    elif ch & 0xff ==ord('b'):
        color = (255,0,0)
    elif ch & 0xff == ord('r'):
        color = (0,0,255)
    elif ch & 0xff == ord('1'):
        radius = 10
    elif ch & 0xff == ord('2'):
        radius = 30
    elif ch & 0xff == ord('3'):
        radius = 50
    elif ch & 0xff == ord('4'):
        radius = 100
cv2.destroyAllWindows()
```

Lane Detection

Python Project

- **Software:** Python & OpenCV
- **Aim:** To create a Python program to take an input image and using thresholding to detect the while lanes in an image.
- **Challenges:** Since this was my first Computer Vision program in Python, learning the Python module was a challenge. However, understanding how to correctly process an image especially dynamically was particularly challenging.
- **Solution:** I planned out what my program was going to do beforehand and learnt as I went along.





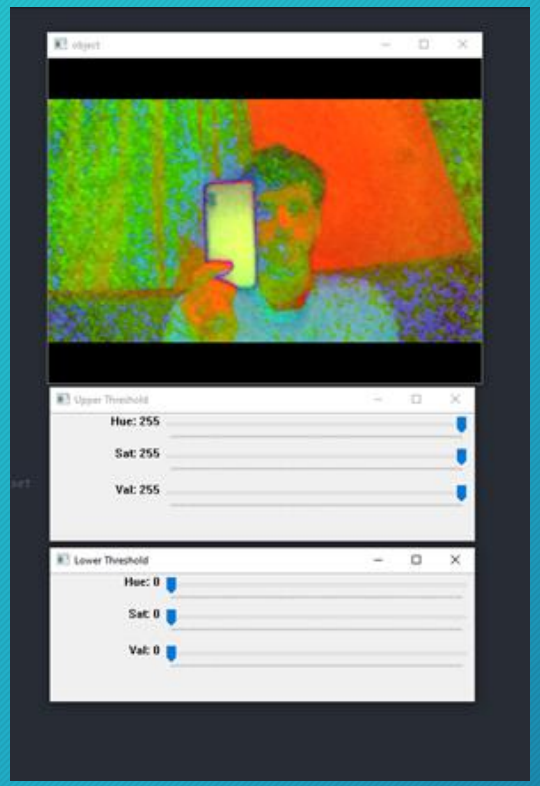
Main Script Code Snippet

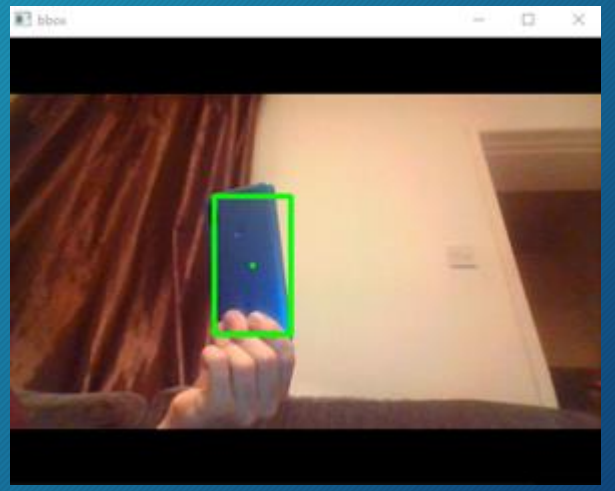
```python
import cv2
import numpy as np
import matplotlib

img = cv2.imread("straight_lines2.jpg")

def color_filter(img):
    #convert to HLS to mask based on HLS
    hls = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
    lower = np.array([0,190,0])
    upper = np.array([255,255,255])
    yellower = np.array([10,0,90])
    yelupper = np.array([50,255,255])
    yellowmask = cv2.inRange(hls, yellower, yelupper)
    whitemask = cv2.inRange(hls, lower, upper)
    mask = cv2.bitwise_or(yellowmask, whitemask)
    masked = cv2.bitwise_and(hls, hls, mask = mask)
    return masked

while(True):

    masked = color_filter(img)
    cv2.imshow("Lane Detection",masked)
    cv2.imshow("Original",img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Colour Tracking Algorithm using Image Thresholding

Python Project

- **Software:** Python & OpenCV
- **Aim:** Developed a program to take a snapshot from a webcam, then dynamically threshold it in the HSV colour space and finally track the object in real time.
- **Challenges:** Since this was my first Computer Vision program in Python, learning the python module was a challenge. However, understanding how to correctly process an image especially dynamically was particularly challenging.
- **Solution:** I planned out what my program was going to do beforehand and learnt as I went along.





Colour Tracking Algorithm using Image Thresholding

Continued

Python Project

Main Script Code Snippet

```python
import numpy as np
import cv2
import time
from Image_Threshold import *
import numpy as np

def img_res(img, scale):

    ## Reasizes an input image by a percentage
    #:param img: image to be reasized
    #:param scale: percentage to scale the image by
    #:return: the reasized image

    width = int(img.shape[1] * scale / 100)
    height = int(img.shape[0] * scale / 100)

    dim = (width, height)

    resized_img = cv2.resize(img, dim)

    return resized_img

def Alert_limit_zones(Centre_Point, Lower_Limit, Upper_Limit):

    ## generates the alert limit lists
    #:param Centre_Point: point zero where the limit boundries are offset
    #:param Lower_Limit: lower boundry limit
    #:param Upper_Limit: upper boundry limit
    #:return: the concatinated limt list

    above = []
    below = []
    centre = [Centre_Point]
    limits = []

    for i in range(Lower_Limit,Upper_Limit):
        above.append(Centre_Point+i)
        below.append(Centre_Point-i)

    limits.extend(below)
    limits.extend(centre)
    limits.extend(above)
```

```python
while(True):
    for i in range(5, 0, -1):
        print (i)
        time.sleep(0.5)

    print("taking snap say Cheese :)")
    ret, frame = cap.read()
    Res_Frame = img_res(frame, 80)
    Lower_lim, Upper_lim = Dynamic_thresh(Res_Frame)

    break

print("This is the Min val Lower: {}".format(Lower_lim[0]))
print("This is the Mid val Lower: {}".format(Lower_lim[1]))
print("This is the Max val Lower: {}".format(Lower_lim[2]))
print("This is the Min val Upper: {}".format(Upper_lim[0]))
print("This is the Mid val Upper: {}".format(Upper_lim[1]))
print("This is the Max val Upper: {}".format(Upper_lim[2]))

while(True):

    ret, frame = cap.read()

    # Reasize the image based on a percentage
    Res_frame = img_res(frame, + 85)

    width = Res_frame.shape[1]
    height = Res_frame.shape[0]

    # Parse the image into the threshold function
    mask = img_thresh(Res_frame,Lower_lim,Upper_lim)

    # Get the bounding Box Coordinates
    x, y, w, h = cv2.boundingRect(mask)
    # Draw the bounding box
    bbox = cv2.rectangle(Res_frame,(x,y),(x+w,y+h),(0,255,0),3)

    # Centre Coordinates of Bounding Box
    Centre_CodX_bbox = int(x+(w/2))
    Centre_CodY_bbox = int(y+(h/2))
```

Colour Tracking Algorithm using Image Thresholding

Continued

Python Project

Threshold Function Code Snippet

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def Dynamic_thresh(img):

    ## Threshold the image dynamically
    #:param img: takes the input image
    #:return: the lower and upper HSV threshold limit arrays

    def nothing(x):
      pass

    cv2.namedWindow('Lower Threshold')
    cv2.resizeWindow('Lower Threshold', 500,150)

    cv2.namedWindow('Upper Threshold')
    cv2.resizeWindow('Upper Threshold', 500,150)

    cv2.createTrackbar("Hue", "Lower Threshold",0,255,nothing)
    cv2.createTrackbar("Sat", "Lower Threshold",0,255,nothing)
    cv2.createTrackbar("Val", "Lower Threshold",0,255,nothing)

    cv2.createTrackbar("Hue", "Upper Threshold",0,255,nothing)
    cv2.createTrackbar("Sat", "Upper Threshold",0,255,nothing)
    cv2.createTrackbar("Val", "Upper Threshold",0,255,nothing)

    while(1):
##################################  Lower Threshold  ##########
        # Created the empty arrays for the Hue saturation and values
        Hue_A_Lower = []
        Saturation_A_Lower = []
        Value_A_Lower = []

        # Use the TrackbarPos and save it to the relevant variables
        Hue_Lower = cv2.getTrackbarPos("Hue", "Lower Threshold")
        sat_Lower = cv2.getTrackbarPos("Sat", "Lower Threshold")
        val_Lower = cv2.getTrackbarPos("Val", "Lower Threshold")

        # Append the values to the empty arrays
        Hue_A_Lower.append(Hue_Lower)
        Saturation_A_Lower.append(sat_Lower)
        Value_A_Lower.append(val_Lower)


        # Convert the BGR image to HSV
        hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

        # Create the Lower and upper limit arrays
        lower_limit = np.array([Hue_Lower, sat_Lower, val_Lower])
        upper_limit = np.array([Hue_Upper, sat_Upper, val_Upper])

        # Create the mask object
        mask = cv2.inRange(hsv, lower_limit, upper_limit)

        cv2.imshow("object", cv2.bitwise_and(hsv,hsv, mask=mask))

        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):

            Lower_lim = [Hue_A_Lower[-1],Saturation_A_Lower[-1],Value_A_Lower[-1]]
            Upper_lim = [Hue_A_Upper[-1],Saturation_A_Upper[-1],Value_A_Upper[-1]]
            break
    return Lower_lim, Upper_lim

def img_thresh(img,Lower_lim,Upper_lim):

    ## generates HSV mask for the vision algorithm
    #:param img: takes the input image
    #:param Lower_lim: lower boundry limit
    #:param Upper_lim: upper boundry limit
    #:return: mask

    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    lower_limit = np.array([Lower_lim[0], Lower_lim[1], Lower_lim[2]])
    upper_limit = np.array([Upper_lim[0], Upper_lim[1], Upper_lim[2]])
    mask = cv2.inRange(hsv, lower_limit, upper_limit)

    return mask
```