



*Symbiosis Skills and Professional University*

Village – Kiwale, Adjoining Pune Mumbai Expressway, Pune –412101, Maharashtra, India

4-Months Internship Report

On

Industrial Training

At

*Technodune Pvt. Ltd.*

Submitted by:

*Pradnya Shinde*

PRN: {*1900601005*}

Final Year B.Tech Mechatronics {SEM VII}

A.Y. 2022-2023

Under the guidance of

Industry Mentor:

*Mr. Parth Sharma*

Faculty Mentor:

*Prof. Jahida Subedar*



# CERTIFICATE

**TECHNODUNE** ROBOTICS • IoT • DRONES


Ref #: 0912202201

To Whomsoever It May Concern

This is to certify that Ms. Pradnya Shinde, PRN No. 1900601005, student of School of Mechatronics Engineering, Symbiosis Skills & Professional University, Kiwale Pune, has successfully completed (B-Tech 4th Year SEM VII) internship program at Technodune Private Limited, Marunji Pune under the guidance of Mr. Parth Sharma (Founder, CEO).

The duration of this internship was from 8<sup>th</sup> August 2022 to 8<sup>th</sup> December 2022.  
We wish her all the success in her future endeavours.  
Her performance during training can be graded as exceptional.

For Technodune Pvt Ltd Dated: 9<sup>th</sup> Dec '22

  
Parth Sharma  
Founder and CEO

Address: Avenue R2, Unit 44, Life Republic Township, Hinjewadi, Marunji, Pune - 41057  
Email: info@technodune.com | Website: technodune.com



## TECHNODUNE

Technodune is a young startup, started by Mr. Parth Sharma aimed at churning out next-gen technology through innovative products which shape the future. Channeling our passion through constant innovation and expertise, they help take you one step ahead into the future. Technodune paves the way for wide-ranging verticals and use-cases.

Technology has constantly been evolving and changing the lives of people around the world.

Technodune Team was born from the need of providing an answer for common and underlying necessities. They make life simple one step at a time. They provide you with the best solutions for problems pertaining to the industrial and commercial environment.

They exist to bring about change, they exist to inspire, and they exist to create something that goes beyond the ordinary. Not a single product has been made for the sake of conventional existence, it is to introduce a new cutting-edge technology that leaves behind a legacy. By upgrading existing technology to a more efficient one, they ensure minimum wastage maximum operating ability.

They believe in empowering the masses- including both businesses and people. For mass empowerment, state-of-the-art technology is a prerequisite. An independent and tight-knit business focused headquartered in Pune, India; Technodune is a 100% founder-owned firm. We build communities, create solutions, and truly believe in collaborating, networking, and socializing.



## ABSTRACT

This report consists of a brief explanation of everything that I learned during the internship period from 8<sup>th</sup> August, 2022 to 8<sup>th</sup> December 2022 as a **Robotic and IoT intern** at Technodune Private Limited.

For the 4 months of my internship, I worked on variety of concepts of Robotics Engineering – Robot Kinematics, Mechanical Design, Machine Learning Fundamentals, Sensor and Microcontroller Integration, Robot Sensing and Perception.

With regards to my long-term goal of specializing in Robotics and AI, Technodune has provided me with ample amount of theoretical as well practical knowledge related to the same. I was also introduced to IoT related applications, thereby providing me with knowledge of embedded systems.

## PROJECT DEXTER

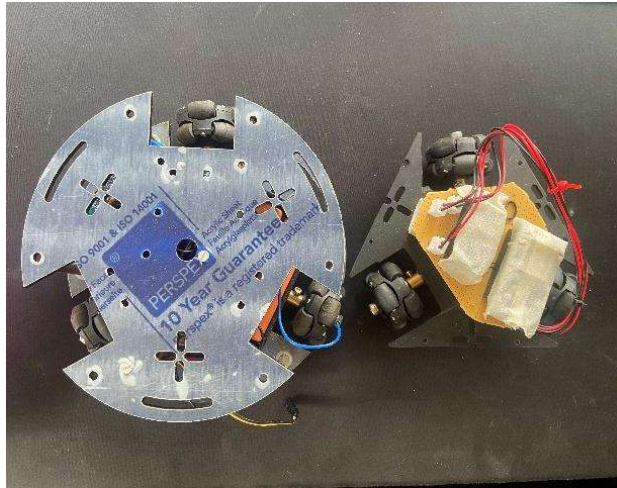
- Initial application of the project was to design and build a souvenir robot, intended to be used on desks.
- The design was to be inspired by R2D2 robot.



- However, the application was changed to Swarm Robotics and the design has to be changed accordingly.
- Size reduction was done with regards to the spacing occupied by electronics.

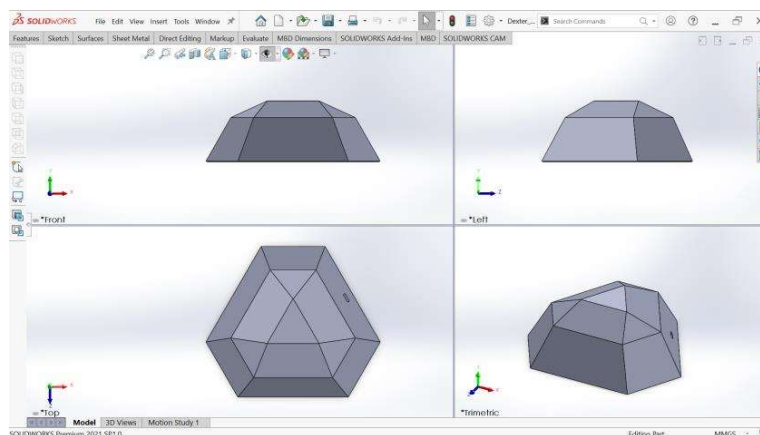


Robot Enclosure



Robot Chassis

- The robot enclosure is designed in Solidworks 2021.



- The robot consists of:
  - ESP WROOM-32
  - Micro USB 18650 Lithium Battery Charger
  - LM2596S DC-DC Step-Down Power Supply
  - 2 x DRV8833 2 Channel DC Motor Driver
  - 3 x 1.5inch 38mm 90° Omni Wheels
  - N20 3V 100 RPM DC Gear Motors



## SWARM ROBOTICS:

- Swarm communication should allow the robots in Swarm to stay aware of their local environment.
- Factors like robustness, scalability and adaptability were considered while designing the robots.
- In the adapted context, the robots are DUMB. They are not aware of their environment and moreover, they cannot take decisions on their own.
- With this in mind, we came up with a centralized control system to implement swarm application.
- Beforehand, we also tested a decentralized approach to make the robots aware of their local environment and take decisions on their own.

## SWARM COMMUNICATION:

### RSSI:

- RSSI – Received Signal Strength Indication is the measured quality of Bluetooth signal on a device.
- The strength of the signal can be use to calculate the distance between two Bluetooth devices.
- We allowed the robots to act as Bluetooth anchors and tested them for their RSSI values during static and dynamic situations.
- The RSSI approach was rejected due to the instability observed in signal strength and the disturbance obtained due to factors like environmental noise.
- The distance calculated using the RSSI value was deemed to be inefficient.

$$Distance = 10^{((Measured\ Power - Instant\ RSSI)/10*N)}.$$

*N is the constant for the environmental factor. It takes a value between 2-4. The measured power is the RSSI value at one meter.*

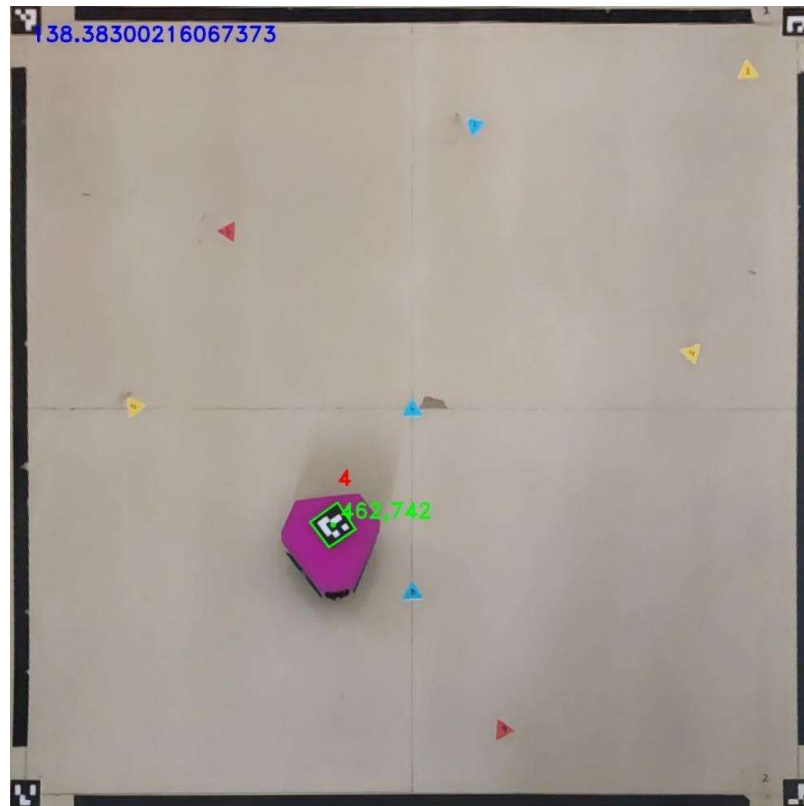
### CENTRALIZED CONTROL:

- Next, we chose a centralized approach to establish Swarm communication.
- We setup an overhead camera, a controller and a robot arena.
- The controller is thereby responsible for localizing a robot in its environment and manipulating the swarm of robots by sending over orientation and speed parameters such that the robot moves to the destination location.

- To implement localization of robots, two approaches were taken:
  1. ArUCO Marker Detection :
- OpenCV provides an ArUCO Library for the implementation of associated functions.
- The library consists of various functions such as marker creation, detection, pose estimation etc.
- The library also allows to access variables such as markerCorners, markerIds etc.
- Each robot will have its own ArUCO marker with a unique marker ID, thereby allowing the camera to perceive different robots.
- Based on the detection obtained, center coordinates of each of the robot(marker) frames were obtained.
- These coordinates were then used to localize the robots in their environment.







Orientation wrt camera frame in **BLUE**.  
Center coordinates of the robot in **GREEN**.  
Marker ID in **RED**.

```

8  ARUCO_DICT = {
9      "DICT_4X4_50": cv2.aruco.DICT_4X4_50,
10     "DICT_4X4_100": cv2.aruco.DICT_4X4_100,
11     "DICT_4X4_250": cv2.aruco.DICT_4X4_250,
12     "DICT_4X4_1000": cv2.aruco.DICT_4X4_1000,
13     "DICT_5X5_50": cv2.aruco.DICT_5X5_50,
14     "DICT_5X5_100": cv2.aruco.DICT_5X5_100,
15     "DICT_5X5_250": cv2.aruco.DICT_5X5_250,
16     "DICT_5X5_1000": cv2.aruco.DICT_5X5_1000,
17     "DICT_6X6_50": cv2.aruco.DICT_6X6_50,
18     "DICT_6X6_100": cv2.aruco.DICT_6X6_100,
19     "DICT_6X6_250": cv2.aruco.DICT_6X6_250,
20     "DICT_6X6_1000": cv2.aruco.DICT_6X6_1000,
21     "DICT_7X7_50": cv2.aruco.DICT_7X7_50,
22     "DICT_7X7_100": cv2.aruco.DICT_7X7_100,
23     "DICT_7X7_250": cv2.aruco.DICT_7X7_250,
24     "DICT_7X7_1000": cv2.aruco.DICT_7X7_1000,
25     "DICT_ARUCO_ORIGINAL": cv2.aruco.DICT_ARUCO_ORIGINAL,
26     "DICT_APRILTAG_16h5": cv2.aruco.DICT_APRILTAG_16h5,
27     "DICT_APRILTAG_25h9": cv2.aruco.DICT_APRILTAG_25h9,
28     "DICT_APRILTAG_36h10": cv2.aruco.DICT_APRILTAG_36h10,
29     "DICT_APRILTAG_36h11": cv2.aruco.DICT_APRILTAG_36h11
30 }

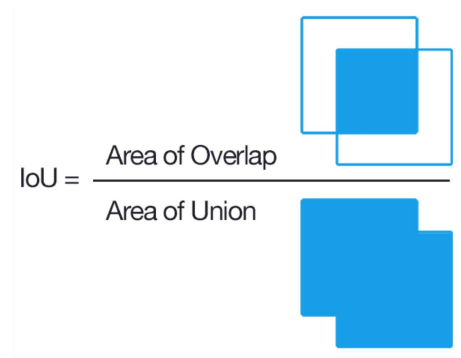
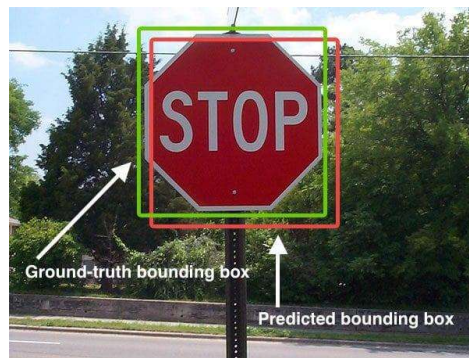
```

**ARUCO Dictionary**

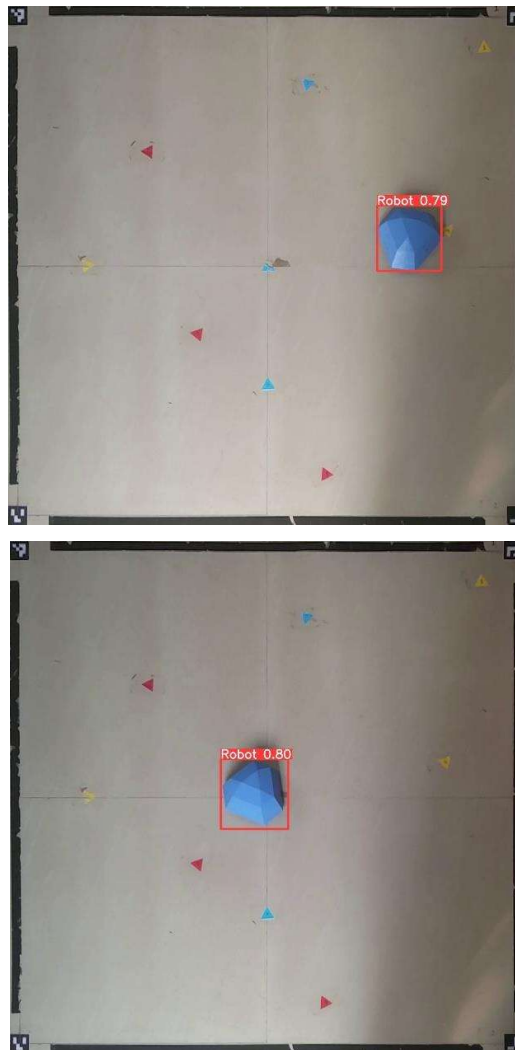
## 2. Object Detection using YoloV5 :

- ArUCO markers were found to be less robust as often they were not detected due to lighting conditions, motion blur, height specifications etc.

- Machine learning thus came into the picture.
- A set of 100 images were collected and trained using YoloV5 algorithm in Google Collab.
- The obtained weights were then used in the end-program for object detection.
- Yolo V5:
  - YOLO an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself.
  - YOLO is one of the most famous object detection algorithms due to its speed and accuracy.
- Intersection over Union (IoU):
  - Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.
  - Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU.
  - In order to apply Intersection over Union to evaluate an (arbitrary) object detector we need:
    1. The ground-truth bounding boxes (i.e., the hand labeled bounding boxes from the testing set that specify where in the image our object is).
    2. The predicted bounding boxes from the model.



**An Intersection over Union score > 0.5 is normally considered a “good” prediction.**



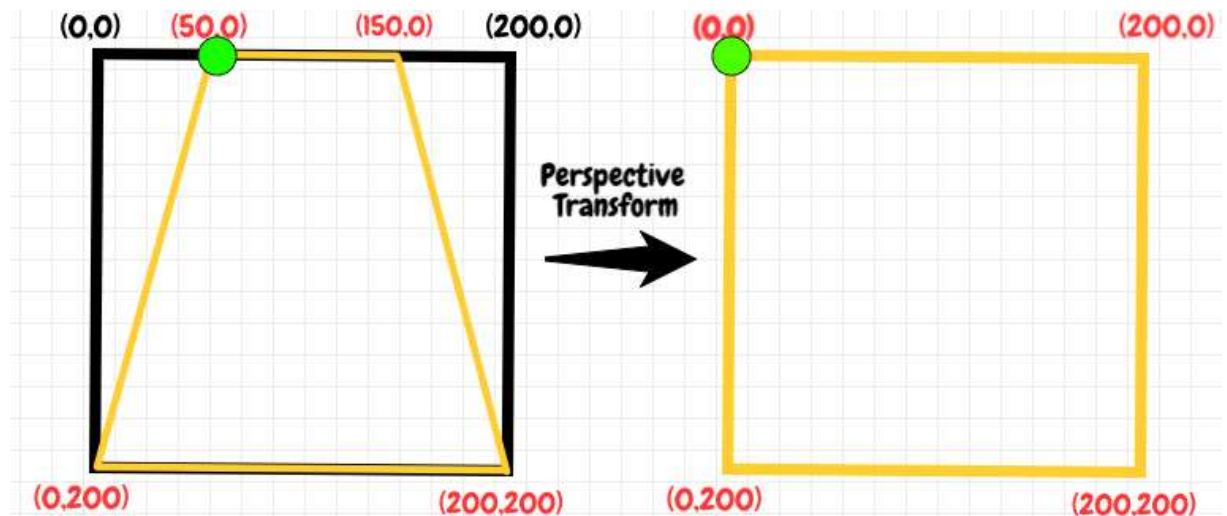
$IoU = 0.79$

$IoU = 0.80$

## PERSPECTIVE CORRECTION

- The camera provides a live stream which is distorted.
- In order to maintain accuracy, each frame of the live video needs to be corrected for perspective.
- The frame is straightened such that there is less error obtained in fetching real-world data.
- Function:

```
matrix = cv2.getPerspectiveTransform(pts1, pts2)
frame1 = cv2.warpPerspective(frame, matrix, (int(length), int(length)))
```



- Working:

$$\begin{aligned} \text{Source} &= [ [50, 0], [150, 0], [200, 0], [200, 0] ] \\ \text{Destination} &= [ [0, 0], [200, 0], [0, 200], [200, 200] ] \end{aligned}$$

Transformation Matrix  
(Based on the above Source, Destination Values)

$$\begin{bmatrix} 2 & 0.5 & -100 \\ 0 & 2 & 0 \\ 0 & 0.005 & 1 \end{bmatrix}$$

To Calculate the location of values of each pixel on Destination. Source Pixel values are multiplied with Transformation Matrix as given below.

#### Step 1 -

Transformation Matrix is multiplied with Pixel x,y coordinate values.

$$\begin{bmatrix} 2 & 0.5 & -100 \\ 0 & 2 & 0 \\ 0 & 0.005 & 1 \end{bmatrix} \cdot \begin{bmatrix} 50 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

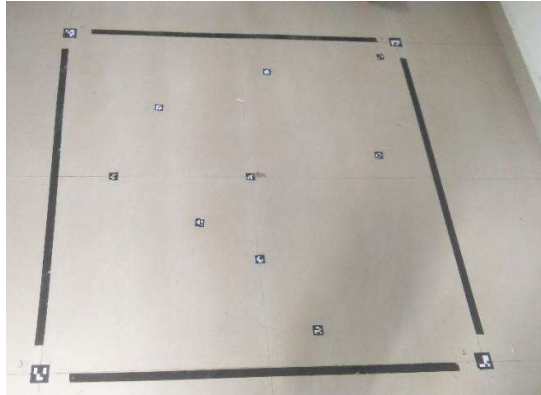
#### Step 2 -

Values obtained of first two rows are divided by third row to obtain (x,y) as given below.

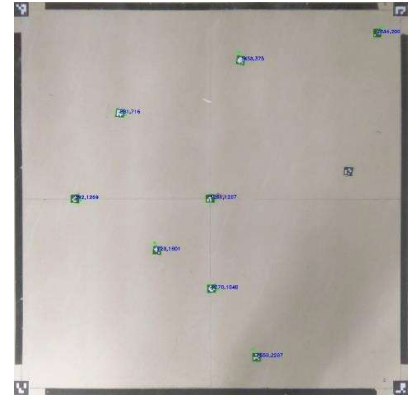
$$[0/1, 0/1] = [0, 0]$$

Therefore source to Destination Pixel location would be as Below

$$\text{Source } [50, 0] = \text{Destination } [0, 0]$$



*Distorted Input Camera Frame*



*Transformed Frame*

## PYTHON PROGRAMMING:

### Data Structures:

Following data structures were mainly used in given application.

#### 1. Tuples

Tuples are data types enclosed within round brackets.

They are immutable in nature.

Eg: ('Pradnya', 'Shinde')

#### 2. Lists

Python lists are like arrays, declared within {square brackets}.

They are mutable data types and hence the elements in lists can be modified.

Eg: [1, 2, 3, 4]

#### 3. Dictionary

Python dictionaries are defined in "key:pair" format.

They are mutable and ordered data types defined within {curly brackets}.

Eg: {1: 'Robot', 2: 'Car', 3: 'Plane'}

## NumPy:

- Numerical Python
- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- Import NumPy library using

```
import numpy as np
```

np acts as an alias

- Creating arrays:

```
array1 = np.array([1,2,3,4])
```

Creates an array

- Arrays with zeroes:

```
np.zeros((4,2),dtype='float32')
```

Returns a new array of given shape and type, filled with zeros.

- Dot product

```
res_arr = np.dot(arr1, arr2)
```

Returns dot product of two arrays

## OpenCV:

- OpenCV is an open-source library that includes several hundreds of computer vision algorithms.
- CV stands for Computer Vision.
- Import OpenCV library:

```
import cv2
```

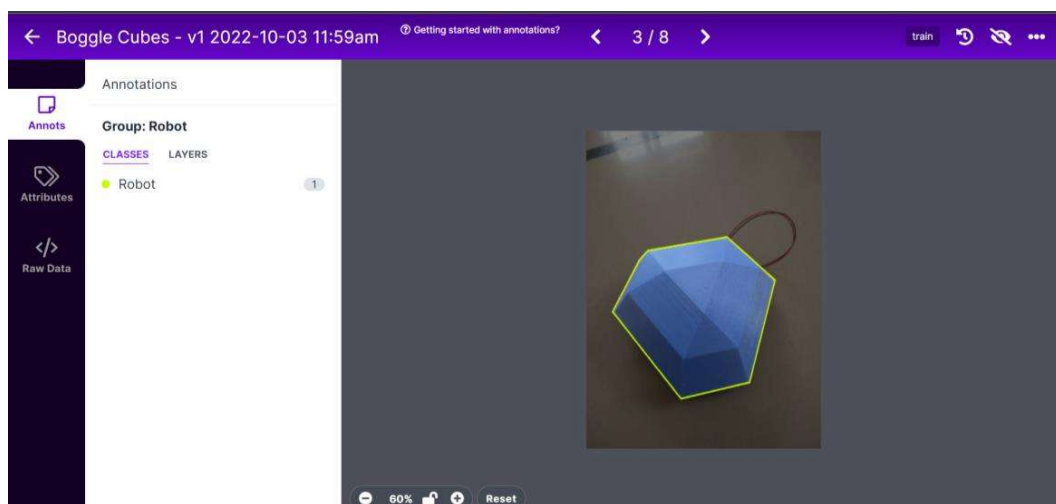
cv2 acts as an alias

- Reading, displaying and saving an image:

```
image = cv2.imread('sample.jpg')
cv2.imshow('Display Window', image)
cv2.waitKey(0)      //image displayed until user
                    presses a key
cv2.imwrite('save.jpg', image)
```

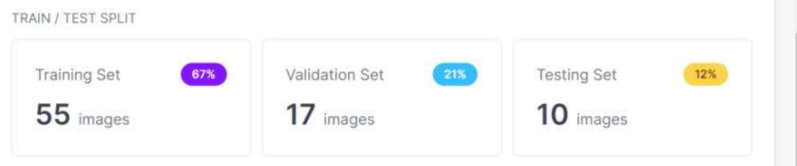
## MACHINE LEARNING:

- Trained a custom object detection model using YoloV5 and PyTorch. Following are the steps:
1. Annotation of Objects using Roboflow





2. Divide data set in 3 parts:



3. Export .zip file

Name	Date modified	Type	Size
test	14-12-2022 21:43	File folder	
train	14-12-2022 21:43	File folder	
valid	14-12-2022 21:43	File folder	
data.yaml	14-12-2022 21:43	Yaml Source File	1 KB
README.dataset.txt	14-12-2022 21:43	Text Document	1 KB
README.robotflow.txt	14-12-2022 21:43	Text Document	1 KB

*Extracted .zip file of the model*

4. Training on Google Collab:

Step 1:

```
# clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5 # clone repo
%cd yolov5
!git reset --hard fbe67e465375231474a2ad80a4389efc77ecff99
!pip install -qr requirements.txt
```

Output:

```
Cloning into 'yolov5'...
remote: Enumerating objects: 13441, done.
remote: Counting objects: 100% (243/243), done.
remote: Compressing objects: 100% (76/76), done.
remote: Total 13441 (delta 177), reused 226 (delta 167), pack-reused 13198
Receiving objects: 100% (13441/13441), 13.24 MiB | 17.90 MiB/s, done.
Resolving deltas: 100% (9241/9241), done.
/content/yolov5
HEAD is now at fbe67e4 Fix `OMP_NUM_THREADS=1` for macOS (#8624)
| 1.6 MB 32.9 MB/s
```

Step 2:

```
%cd /content/yolov5/
```

Step 3:

```
# install dependencies as necessary
# install dependencies (ignore errors)
import torch
```

```
from IPython.display import Image, clear_output # to display images
from utils.downloads import attempt_download # to download models/datasets

# clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))
```

Output:

```
Setup complete. Using torch 1.12.1+cu113 _CudaDeviceProperties(name='Tesla T4', major=7, minor=5, total_memory=15109MB, multi_processor_count=40)
```

Step 4:

Clone the zip file of annotated dataset

```
%cd /content
!curl -
L "https://app.roboflow.com/ds/dP99ZDzvQw?key=k8lJvv5yUA" > roboflow.zip; un
zip roboflow.zip; rm roboflow.zip
```

Step 5:

```
# this is the YAML file Roboflow wrote for us that we're loading into this
notebook with our data
%cat data.yaml
```

Output:

```
train: ../train/images
val: ../valid/images

nc: 1
names: ['Robot']
```

Step 6:

Train the dataset

```
import yaml
with open( "data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream) ['nc'])
%%time
%cd /content/yolov5/
!python train.py --img 640 --batch 16 --epochs 100 --data '../data.yaml' --
cfg ./models/yolov5s.yaml --weights '' --name yolov5s_results --cache
```

Step 7:

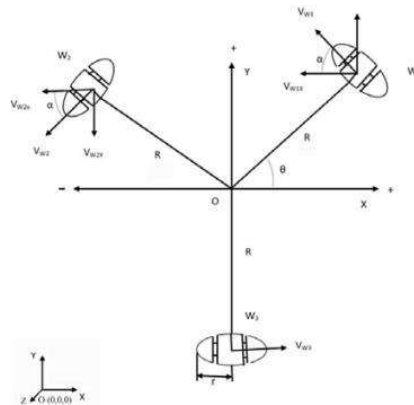
Export best.pt weights for further use

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
%cp /content/yolov5/runs/train/yolov5s_results/weights/last.pt /content/gdrive/My\ Drive
```

- These weights were then used for the intended application.

### 3-WHEEL OMNI DRIVE KINEMATICS



**3 wheeled omni drive**

$V_{w1}$ ,  $V_{w2}$ ,  $V_{w3}$  are linear velocities of wheel 1, 2 & 3 respectively.

Wheel 1 (W1):

$$VW1X = (-) VW1 \cos(\alpha)$$

$$VW1Y = (+) VW1 \sin(\alpha)$$

Wheel 2 (W2):

$$VW2X = (-) VW2 \cos(\alpha)$$

$$VW2Y = (+) VW2 \sin(\alpha)$$

Resultant chassis velocity:

$$VWX = VW3 - VW1 \cos(\alpha) - VW2 \cos(\alpha)$$

$$VWY = VW1 \sin(\alpha) - VW2 \sin(\alpha)$$

$$VWi (1,2,3) = \omega \cdot r$$

$\omega$  = angular velocity of wheel

$r$  = radius of wheel