HUMAN FOLLOWING SHOPPING CART

A Final Year Project Report

Presented to

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

Department of Mechanical Engineering

NUST

ISLAMABAD, PAKISTAN

In Partial Fulfillment
of the Requirements for the Degree of
Bachelors of Mechanical Engineering

By

MUHAMMAD WAQAR AKRAM

October 7th , 2020

EXAMINATION COMMITTEE

We hereby recommend that the final year project report prepared under our supervision by:
Muhammad Waqar Akram 00000130975

Titled: "HUMAN FOLLOWING SHOPPING CART" be accepted in partial fulfillment of the requirements for the award of MECHANICAL ENGINEERING degree with grade ___

Supervisor: Yasar Ayaz, Dr. (Professor) Department of Robotics & Artificial Intelligence	______
Committee Member: Muhammad Jawad Khan, Dr. (Assistant Professor) Department of Robotics & Artificial Intelligence	______

______________________ ___________

(Head of Department) (Date)

COUNTERSIGNED

Dated: ________________ ________________________

(Dean / Principal)

ABSTRACT

In this project we are going to develop a shopping cart that is capable to follow a human wearing a certain tag. The cart determines the position of the object (human wearing a tag) and then decides the way to follow it. It uses image processing to obtain the position of the object. The object following eliminates the difficulties of pushing the cart around and also eliminates the need for steering it. The project involves finding the path to follow the object and run the cart smoothly without any problem, using the laws of dynamics. Then making a control system that can be modeled in a control system. The robot cart uses two radii of turn depending upon the position of the object. Then there is a software written to operates and guides the cart to follow the human carrying the tag.

ACKNOWLEDGMENTS

Throughout the duration of this project the guidance and supervision of Dr Yasar Ayaz who is one of the most celebrated scientists in the country. This project couldn’t have been possible without the help of Dr Yasar Ayaz.He guided us throughout the project despite his very busy schedule. It was no less than an honor to work under the guidance of such a prominent leading face in the world of robotics and artificial intelligence. Then there comes the mention of Mr. Hamza Asif Nizami who has been the driving force behind the completion of this project. Mr. Hamza Asif Nizami is very approachable and always there to guide you. Every time we needed any help Mr. Asif Nizami was there to help us out. The contribution of our other committee member Mr. Dr Jawad Khan is also remarkable. Dr Jawad khan specializes robotics specifically in mobile robots I also acknowledge the efforts of our project coordinator Mr. Rehan Zahid. Lastly I thank school of mechanical and manufacturing engineering to provide us the environment to work.

ORIGINALITY REPORT

I hereby declare that this project has been originally produced and has not been copied. All the findings are original. I also declare that the project of similar work has not been presented to any other institute and only have been presented to Dr Yasar Ayaz of SMME.THE ORIGINALITY REPORT OF THE PROJECT PRODUCED BY TURNIT SHOWS 11% SIMILARITY INDEX. WITH NO SINGLE SOURCE MORE THAN ONE PERCENT SIMILARITY. REPORT IS ATTACHED AT THE NEXT PAGE.

report

ORIGINALITY REPORT

11%	3%	4%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Abdul Malik Shaari, Nur Safwati Mohd Nor. "Position and Orientation Detection of Stored Object Using RFID Tags", Procedia Engineering, 2017 Publication	1%
2	Submitted to Engineers Australia Student Paper	1%
3	tech.3si.vn Internet Source	1%
4	Submitted to Liverpool John Moores University Student Paper	1%
5	Submitted to Middlesex University Student Paper	1%
6	Submitted to FPT Academy International HoChiMinh City Student Paper	1%
7	Submitted to University of Western Sydney Student Paper	1%

Submitted to CSU, San Jose State University

CONTENTS

ABBREVIATIONS

NUST	National University of Sciences and Technology
AC	Alternating Current
DC	Direct Current
SMME	School of Mechanical and Manufacturing Engineering

NOMENCLATURE

F	Force
R	Rolling resistance
r	reaction force
l	distance from front wheel
h	height of center of gravity
m	mass
α	inclination
V	Velocity

CHAPTER 1: INTRODUCTION

The motivation of work, problem statement and objectives of the project are detailed below.

1.1 Motivation

With the increase in population and ever increasing speed of life we need robots to help us with our jobs to do them efficiently and effectively. This technology of human following robot can be used at a number of places like for example in green farms farmers need to cultivate the field and fertilize. Similarly it will provide a new world experience in the shopping malls to the customers. It can be used in industrial warehouses workshops and at airport. It will help the passengers by carrying their luggage for them. Why we have selected the shopping mall out of all these applications because in other applications the need for complexity is far greater and it requires robust design and the model to do the job. But a shopping cart is relatively simple, it's cost effective, it targets the market and it can be easily implemented.

1.2 Problem Statement

The objective is:

"A human following shopping cart that follows a human wearing the tag"

1.3 Objectives

The above problem for the automated shopping cart under consideration can be solved by breaking it down into the following three main objectives.

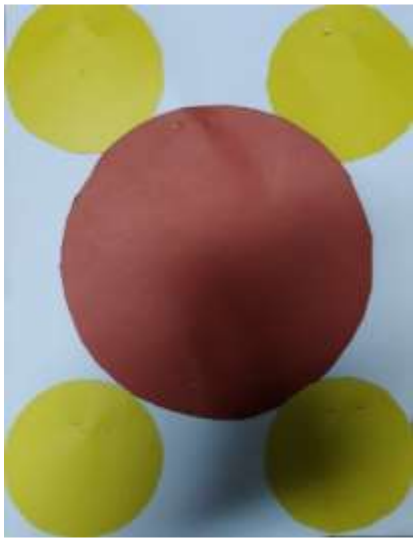
- Train Model for Machine learning
- Object Detection
- Object Following

1.3.1 Train Model for Machine Learning

Tensorflow is an open source machine learning library. And is especially useful for working with the branch of machine learning called Deep Learning. Deep learning does not require us to extract features of images manually. Instead we can use raw pixels of the images and classifier will do the rest. We can train an image classifier starting from a directory of images. We are going to train a classifier which will tell us the object when its detected. We work with a code lab called tensorflow for poets. This code lab is high level to train our object detector we just need to run a couple of scripts.to train an object detector with tensorflow we only need to provide one thing that is training data (a directory full of images) our classifier object detector will detect a custom made unique tricolor tag named as tag1. We provided more than 300 images for training. Script takes hours to train the classifier. It does not train the classifier from scratch. Instead it starts from an existing classifier called inception. Inception is googles best image classifier. And it is open source. Inception was trained on 1.2 million images from thousands of different categories. We will begin with inception and then will use a technique called retraining to adjust it to work with our images.

1.3.2 Object Detection

In deep learning we will be using the classifier called Neural Network. A neural network can learn complex functions.it doesn’t need us to extract the features of the object manually rather it uses pixels I raw form. Our classifier will not only detect the object it will also tell us the location of the object in 5 discrete intervals.



Our object named tag1

1.3.3 Object Detection

Once our object is detected we will determine its location using the information where does it lies in the view of camera. Camera has a vison of 50° and we have divided it into 5 intervals left to right which tells us about its location. We will work out for the values of pwm for accurately following the object.

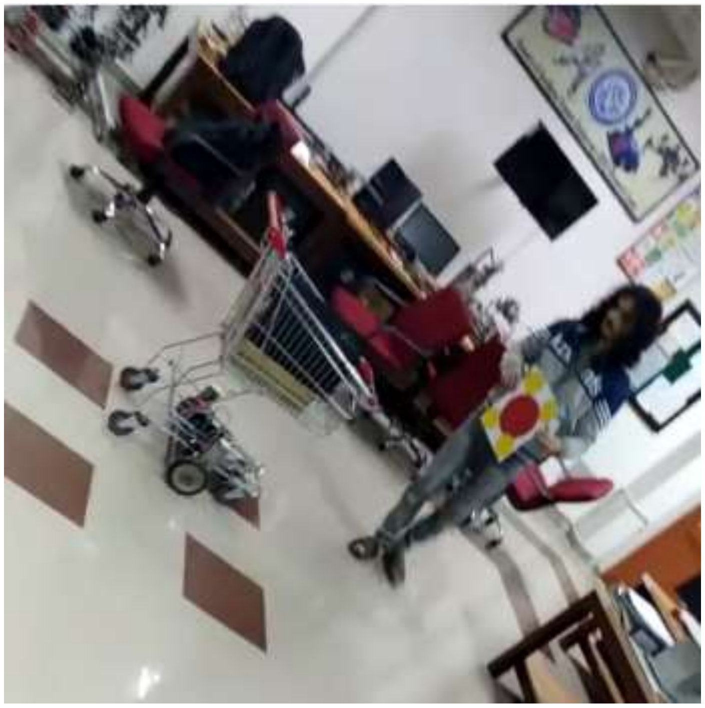


Fig. Robot Cat following the tag1

CHAPTER 2: LITERATURE REVIEW

A mobile robot is defined as the one that is able of locomotion. Since it can move it has more complex as compared to a fixed robot. It has to be aware of its surrounding in order to react to its environment. It has some systems that make it possible for it. These systems are explained below

2.1 Movement/Locomotion

The very first thing a mobile robot has to do is to move. For the correct steering and speed of the mobile robot it must have motor and batteries that not only able to manage its weight but also the resistance it faces from its environment. For that purpose we will use DC motors because they are power efficient.

Following are the type of motors that can be used for this purpose

- Permanent Magnet DC Motors
- Series DC Motors
- Shunt / Parallel DC Motors
- Compound DC Motors

2.1.1 Permanent Magnet DC Motors

These motors are used for the robots that are supposed to have low horse power. They produce field flux that produces good torque but remains inadequate afterwards.

2.1.2 Series DC Motors

They produce good torque in the start but they can be damaged by no load conditions.

2.1.3 Shunt DC Motors

These motors are great to regulate the speed and reverse controls amplified.

2.1.4 Compound DC Motors

These are like shut motors and they produce great starting torque but when it comes to variable speed control, they are not very good options.

For every motor here are advantages and disadvantages.

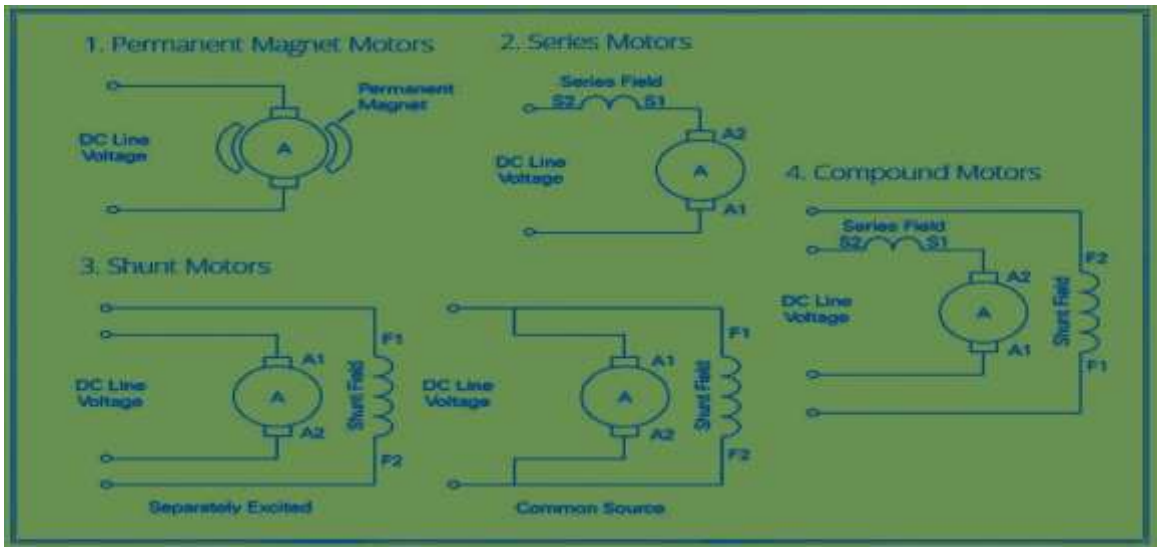


Fig Brushed DC motors

Figure I - Types of Brushed DC Motors

2.2 Perception and planning

A robot needs to know its location in order to perform most of its tasks. In order to achieve the navigation there are four blocks:

- Awareness/Perception
- Localization
- Cognizance
- Motion Control

2.3 Target recognition and distinguishing

Following are some of the options available for object recognition and following. We will go through the pros and cons of each, one by one.

1. Kinect

Kinect detects motion so it is motion sensor device. It is made by Microsoft, contains an infrared emitter, an RGB camera, a multi array microphone, an IR depth sensor, and a tilt motor.
Depth camera: It works using IR (Infrared) technology. It scans IR plots and make a 3-D map of objects and environment around it.
Color camera: a webcam that captures video. By using this information, it is possible to attain the details about items and people in the nearby environment.

IR Emitter: points a sequence of IR (Infrared) beams into the environment. When the rays hit an object, it make the pattern to become distorted. This distortion is examined by the depth camera making a 3D map.
Microphone array: It uses four mics which identify origin of voices and sounds coming on them. It also filters the background noise.
Tilt motor: As the name suggest it is a motor which make adjustments according to height of object it is facing. It is tall, it adjusts the box upside and vice versa.
Object recognition using Kinect is a complicated part and is not user friendly. We didn't used Kinect for our object detection and following system since the corresponding hardware was unavailable. Moreover there were more economical solutions available than Kinect.

2. Beacon and receiver

They are very versatile and can detect various objects in several surroundings. However, they lack precision in regular use and hence their probability varies. Still, they can be quite luxurious in terms of money. Therefore, it is preferred to make these states in single-board computers. The latest research tells that wireless signals have surfaced for making low range radar systems in less and higher mess surroundings with reasonable results. They are used mostly for short-range detection and mostly in indoor environments. Study shows target location can be projected from the CSI data which inter CSI data can also be outlined by Physical Layer Convergence Procedure (PLCP)(Stanislaw Rzewski et air.,)

3. RFid tag

Nowadays, for of its omnipresence, RFID (radio frequency identification) technology is leading in the area of object location. It works by emitting radio signals which are sensed by a tag. The tag then sends back an exclusive ID code to the emitter which is linked using an information management system. There has been a lot of research done in this area to detect objects that are tagged, placed on surfaces as tables and it has now reached a certain level of ripeness.

Database of information has been stablished by earlier mostly in the home environments using the idea of "Intelligent Space". Intelligent Space spreads opportunity of home service robot. The major application of intelligent space are automatic restoring objects and searching. The suggested system entails of following things: object box or case, RFID tag, RFID reader, and Arduino board. On object case cover, RFID tag is connected. The purpose of RFID reader is to sense tag data and receive data of the stored object box with connection of Arduino to software. The information received from tags tell us about box coordinate and through a different code, specify the right or wrong of each object's position and orientation.

In the studies done before, the range of object orientation and position is inadequate and they are not in consistency of RFID tag reading. Still, researchers put on variation of things in object case on shelf, positions and orientation of the stored object's boxes which make the cases irregular. So, the data on object orientation and position is important in order to keep the object's case on shelf in a correct position. However, the robot can easily grasp the box within exact position and orientation.

4. Vision

Computer vision makes computer to see. But, how? It does it by obtaining, processing, studying and understanding images captured by a sensor. It does it by mining high-dimensional data(i.e. it includes lots of features) from the outside world to make numerical or figurative information, i.e. in the forms of matrices to make decisions[4][5][6][7]. It is similar to humans which transform visual images that we get through retina, by making it picture of the world we understand through our thought process. By processing this data, we take appropriate decisions or actions. This image knowledge can be imagined as the separation of symbolic information from image data by means of models built with the help of physics, geometry, statistics, and learning theory. We are using computer vision for this project

<u>**2.4 camera based vision and neural networks**</u>

Computer vision is a modern field formed by the combination of computer science, robotics and other fields. Since its foundation in recent years, it has impressed upon a lot of researches and common people with applications like self-driving cars. The development of convoluted neural networks has made a great progress for it. One of the most important power of computer vision is object detection due to which it us now seen in everyday robotics application like vehicle detection, pose approximation and surveillance. In classification algorithms, it detect object by making a box bounding it, by location the object of interest in image. However, we are not limited by it in object detection with computer vision as there is no limit or many bounding boxes and gives us high versatility in detecting objects.

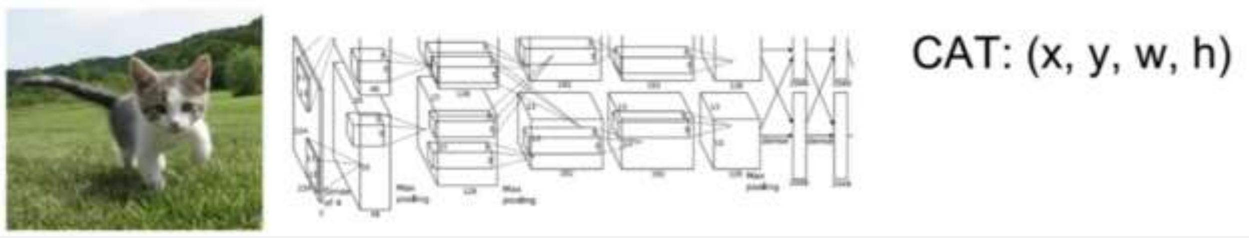


Fig how neural network works

We cannot apply convoluted neural network because of the output layer is not constant which means number of occurrences of object detection can vary which is difficult in CNN with fully connected layer. Another approach can be to capture different regions of concentration from the image, and then apply CNN for object detection in that region. This is a naïve approach as the target might have different position and its size can be varied. This would require a large quantity of regions which is very high computationally expensive and cannot work on small computers. For this, researchers have developed algorithms like R-CNN and YOLO to tackle this problem.

RCNN

In order to solve this problem, where we have to select a large number of regions, Ross Girshick et al suggested a technique in which, selective search is implemented. It involves extracting a limited number of regions (2000) taken from the image and the called these regions as regions proposals. It has advantage that it limits the number of regions, otherwise we need to classify much more region, now we can get our work done with only a few of them. For selecting these 2000 region proposals, it uses following selective search algorithm.

Selective Search:
1. Generate initial sub-segmentation, we generate many candidate regions
2. Use greedy algorithm to recursively combine similar regions into larger ones
3. Use the generated regions to produce the final candidate region proposals

3. Use the generated regions to produce the final candidate region proposals

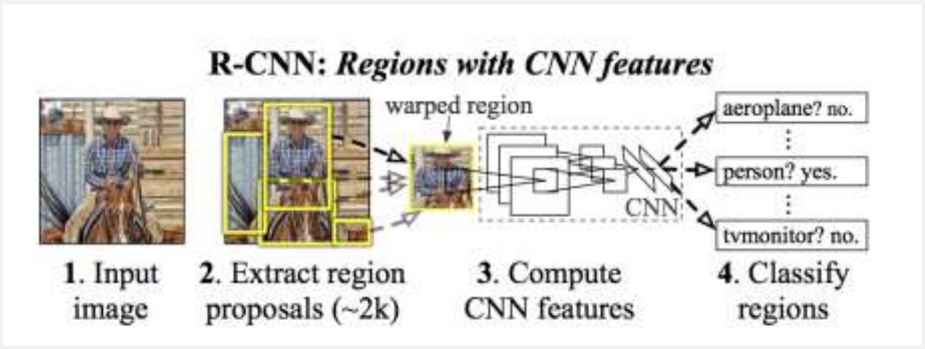


Fig How RCNN works

These selected 2000 regions are then grouped into a square and send into a CNN that make a 4096-dimensional vectors of feature as output. The purpose is feature extraction and its output is then fed into support vector machine which classify it and locate the object inside the region proposal. It has another advantage other than object detection which is it predicts four values. These values are offset values and can be used to increase the precision of algorithm to detect in bounding box. Let us suppose we have a region proposal which detects a person, algorithm can correctly point out that where person is present but it might cut it in half or make it headless. In such cases, offset values are great in helping to adjust the bounding box for the region proposal.

Recognition

The classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity. Different varieties of the recognition problem are described in the literature.

Object recognition (also called **object classification**) – one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene. Blippar, Google Goggles and Like That provide stand-alone programs that illustrate this functionality.

- **Identification** – an individual instance of an object is recognized. Examples include identification of a specific person's face or fingerprint, identification of handwritten digits, or identification of a specific vehicle.
- **Detection** – the image data are scanned for a specific condition. Examples include detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system. Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analyzed by more computationally demanding techniques to produce a correct interpretation.

Currently, the best algorithms for such tasks are based on convolutional neural networks. An illustration of their capabilities is given by the ImageNet Large Scale Visual Recognition Challenge; this is a benchmark in object classification and detection, with millions of images and hundreds of object classes. Performance of convolutional neural networks, on the ImageNet tests, is now close to that of humans.[29] The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble with images that have been distorted with filters (an increasingly common phenomenon with modern digital cameras). By contrast, those kinds of images rarely trouble humans. Humans, however, tend to have trouble with other issues. For example, they are not good at classifying objects into fine-grained classes, such as the particular breed of dog or species of bird, whereas convolutional neural networks handle this with ease.

<u>2.5 Equipment</u>

<u>Sensors and Micro-controllers</u>

Using raspberry pie instead of whole computer

Using a whole computer for the object detection and following works great but it’s neither cost effective nor computationally easy. Moreover it adds to more owner requirement and adding to the weight of the cart. Using a computer gives faster computation and more frame rate but it’s an overkill for such a scale of project. So we are using raspberry pie 3B+ for the brain of the operation. A Logitech 720 p HD gives the video feed to the raspberry pie and it does the image processing and makes the decision of driving the cart around.

Any mobile robot which is built to fulfill a requirement or a goal requires different and specific to that problem or goal sensors. Other than those, some universal or general sensors are also used typically in almost all of the mobile robots.

Some sensors are:

- Ultra-Sonic Sensors - SONARs
- Camera – Vision
- Wireless Beacons

2.5.2 Micro-controllers

The sensors are then controlled by some microcontrollers like Arduino UNO or Arduino Mega which are as a whole controlled by another microcontroller such as a Raspberry Pi.

2.5.3 Other Instrumentation

Apart from sensors and micro-controllers, some other equipment is also used for the instrumentation to work efficiently such as:

- H-Bridge
- Relay

CHAPTER 3: METHODOLOGY

3.1 Preparing the data set for training

We took more than 300 images in indoor light variations to make the data set. All the pictures were indoor and with only in artificial lights because the cart will only be operating indoors so there is no reason for the data set to have too much variety on this matter. After collecting all the pictures we placed them in training/demo folder. We installed labelimg software and made the bounding boxes around the target. All the pictures were taken from the same camera that would be used during detection. This would reduce a lot of anomalies that could come due to difference in picture quality.

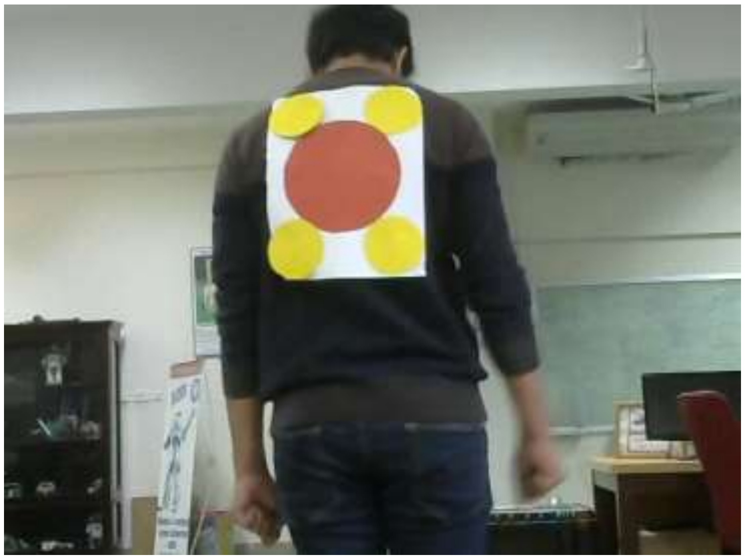


Fig Person wearing the tag

We setup tensorflow in our system. Afterword I created a under tensorflow and named it workspace .within the workspace we stored all our training set-ups. Then we created another folder under workspace and named it training demo.

The training demo folder is our *training folder*, which will contain all files related to our model training. For each training there should be a separate training folder,

Annotating images
To annotate images we used labelImg package.

- After collecting all the images we placed them in images folder inside training demo folder. There were more than 300 images and only one classier. our tag
- Then we launched labelImg
- Then we activated labelImg
- Next we went ahead and started labelling in the labelimg.

- Using the following command

```
Python labelImg.py ..\..\workspace\training_demo\images
```

- A dialogue window opens which points to training_demo\images folder.
- After that we selected the images folder and started annotating over images.

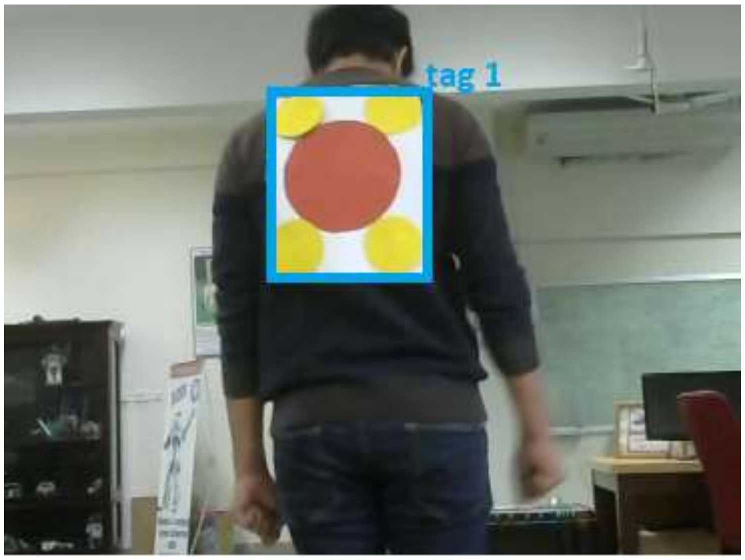


Fig. Tag detected and labelled by computer vision

Partitioning the images

After annotating all the objects only a part of it will be used for training and the rest of it will be used for testing purposes.

Most of the images are used for training task and the rest about 10% will be used for testing purpose. After deciding the number of images for training and testing task make another folder inside the images folder and place the corresponding images and their corresponding.xml files within their corresponding folders i.e. training_demo\images\train folder and training_demo\images\test.

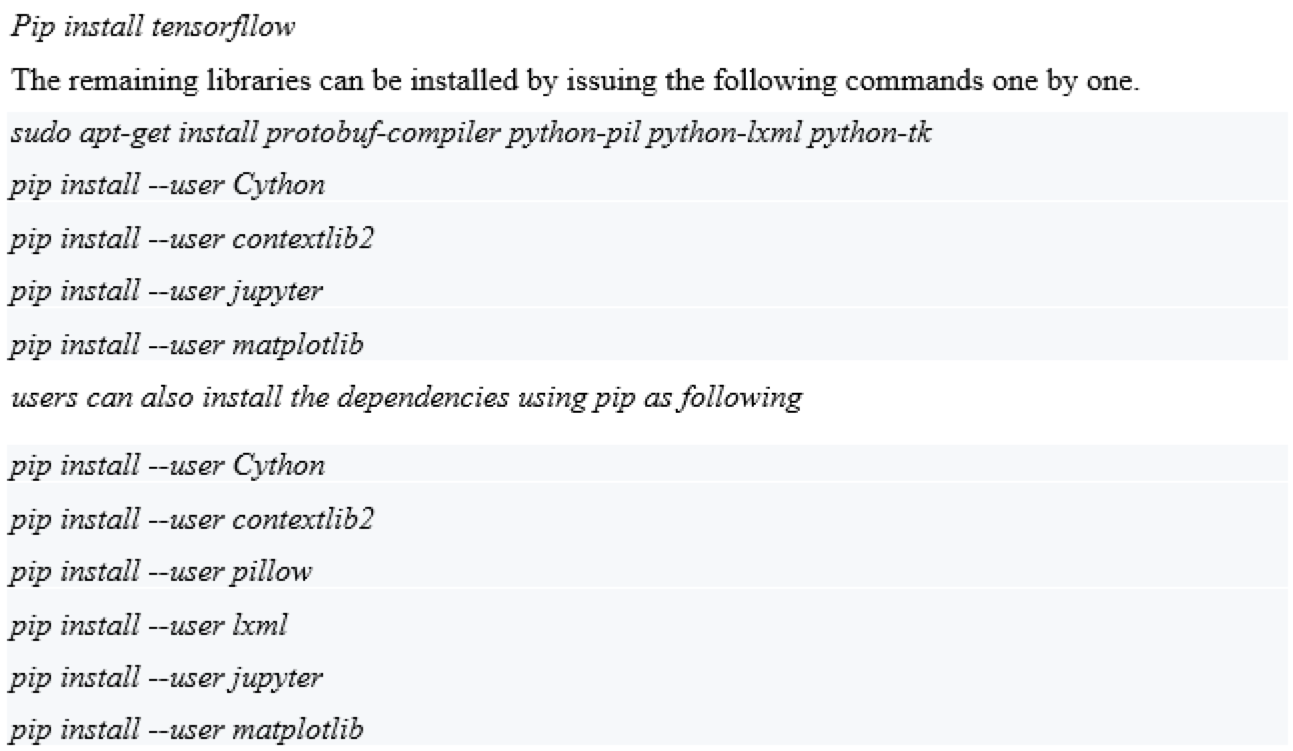
After the above process there are two folders inside images train and test .they contain 90% images and their .xml files in training folder and 10% images and their corresponding .xml files in the test folder. The script will not delete the images and will avoid the risk of losing the data. After placing images and their .xml files in the corresponding folders delete the images from images folder manually.

Setting up tensorflow on computer

Following steps were taken to setup tensorflow on the computer.

Tensorflow object detection API has various dependencies.

Use the following command to install the tensorflow



Pip install tensorfllow

The remaining libraries can be installed by issuing the following commands one by one.

```
sudo apt-get install protobuf-compiler python-pil python-lxml python-tk
pip install --user Cython
pip install --user contextlib2
pip install --user jupyter
pip install --user matplotlib
```

users can also install the dependencies using pip as following

```
pip install --user Cython
pip install --user contextlib2
pip install --user pillow
pip install --user lxml
pip install --user jupyter
pip install --user matplotlib
```

Coco API Installation

Once we have downloaded it copy the pycocotools folder to the tensorflow/models/research folder.

Protobuf Compilation

Compile the protobuf libraries by running the following command from the tensorflow/models/research/ directory:

```
# From tensorflow/models/research/
protoc object_detection/protos/*.proto --python_out
```

Adding Libraries to PYTHONPATH

Run the following command to add the libraries to pythonopath

```
# From tensorflow/models/research/
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

Testing the Installation

Test the installation by running the following command *python object_detection/builders/model_builder_test.py*

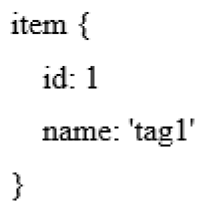
<u>Setting up tensorflow lite on raberry pi 3b+</u>

After that we setup tensorflow lite on our raspberry pi 3b+ module.We did training for the tag on our system and detection was 100% accurate at a frame rate of 5 fps.
We used Google’s SSDLite-MobileNet-v2 object detection model which is trained off the MSCOCO dataset .We converted it to run on TensorFlow Lite.

<u>Creating Label Map</u>

It is required to create a label map for tensorflow.This mapps each of the label in integer values..
The label map is not only used for training purpose but also for detection.

Below is an example label map, our dataset contains 1 label and we named it as “tag1”:



```
item {
    id: 1
    name: 'tag1'
}
```

The extension of label map file is .pbtxt and is placed inside the training_demo\annotations folder.

<u>Creating TF Record</u>

After generating the annotations and splitting our data sets for training and testing subsets we covert the annotations into tfrecord format.

Following are the steps we took to accomplish this:

- Converted *.xml files to a unified *.csv .
- Converted *.csv files to *.record .

<u>Converting *.xml to *.csv</u>

We used simple script to do this iteratively. *.xml files in
the training_demo\images\train and training_demo\images\test , and generated a *.csv

Converting from *.csv to *.record

After having the csv file we converted it into tfrecords file.

Now that we have obtained our *.csv annotation files, we will need to convert them into TFRecords.

Configuring a Training Pipeline

We did not do training from scratch rather we will use pretrained model that is made available by tensorflow.

We used ssd_mobilenet_v2_coco model, since we are using it on raspberry pi and this model can work with low computational capability.it works with good speed and performance.

since we are using the ssd_mobilenet_v2_coco model, we downloaded the corresponding ssd_mobilenet_v2_coco file.

We also downloaded the corresponding neural network file.it was pre-trained NN for the model we are using.the downloaded file would be a *.tar.gz file.

Once the *.tar.gz file was downloaded,we opened it using a decompression software. Next, we open the folder that you see when the compressed folder is opened and extract it's contents to the training_demo\pre-trained-model.

After downloading and extracting the pretrained model we applied some changes to *.config file.

Important changes

Trained Pipeline

Changing the size of images
Since we are implementing our solution on raspberry pi which has very low computational power, we cannot go with 300*300px images size. Our system will be hanged. To avoid this problem we change the images size from 300*300 to 150*150px. In the file pipeline.config change (in the line 6 and 7) the height and width to 150*150. Also change the same height and width in ssd mobilnet folder in the line 45 and 46.it will help in faster calculation and efficient operation.

Number of Classes
On the top of both the files we have num classes, this is the number of classes we need to detect in our operation. In our case it is 1, so it is kept 1 in both the files.

Batch Size
We have a batch size line at 134 in the pipleline.config and same thing on 141 line of the file other. We reduced the batch size to 2 at a time, otherwise our system would stuck. Don't reduce to less than 2 or it will result in wrong calculations. It depends upon the computation power of your device.

Number of files in data set
In the pipeline.config line 168 and line 182 in ssd.mobilnet file there are num-examples, it is the number of pictures in training folder. It can be left unchanged.
Change it data set size.

Other changes
In the line 156 of ssd.mobilnet there address of the model.ckpt files. It points to the model.ckpt of the pre trained model we downloaded.
See line 173 and 188 of ssd.mobilnet both of them start a scope. Train means training files and evals means testing. Point to the address of train.record and test.record we created, at line 175 and 190 respectively.
Now look at the line 181, this line originally didn't existed in the file we downloaded. We wrote it ourselves. So if cocoapi is installed, and you wish to use it, you write this line inside the eval_config as written below:

```
  eval_config: {
 metrics_set:"coco_detection_metrics"
 num_examples: 160
 # Note: The below line limits the evaluation process to 10 evaluations.
 # Remove the below line to evaluate indefinitely.
 max_evals: 10
}
```

Creating Frozen graph

A simple script will create and export the frozen graph.We used Tensorflow frozen graph with compatibility.

Then we converted the frozen graph to the Tensorflow lite format. It created a file named detect.tflite in tflite folder. The we renamed the detect.tflite to ***detectx.tflite***. then we ran the object detection.

3.3 Discretization

It's a process to divide a continuous feature into small intervals or portions. So we used discretization in our cart operation. We have not used all the continuous values but we divided the view of camera that is 50° to 5 portions each containing 10°.why do we need to discretize the view of camera. It makes the computations simpler and easy. That is it is computationally less costly as compared to using the whole continuous values the discretization we used is uni-variation that is we only applied it to a single feature.

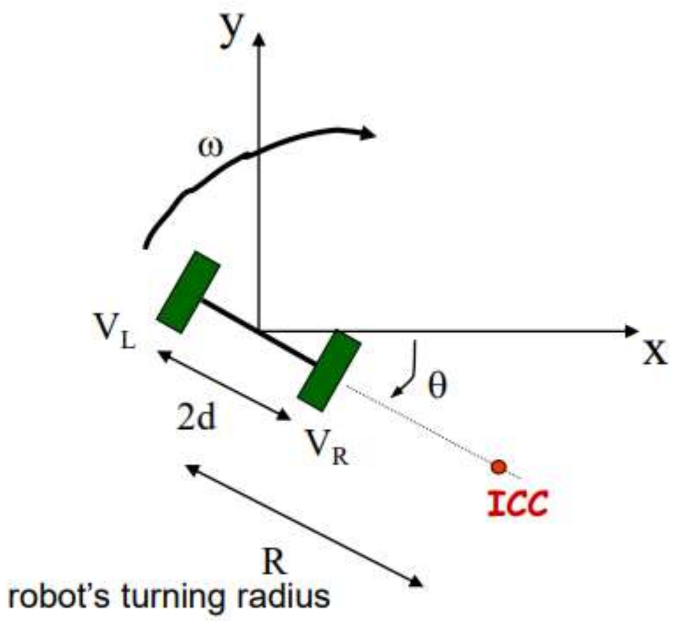
We could have come up with more continuous function but it would have made it impractical.

Each interval "the central 10 is for the cart to move straight. The right 10 means the cart turn clockwise in a bigger radius. The farther right 10 require the cart to move at a smaller radius" something along those lines

3.4 Radius of turn

First the coordinate system of the robot and the universal system are established, then kinematic analysis is done upon them to determine the radius of turn and turn rate.

For determining the radius of turn, we need to find out the instantaneous center of curvature ICC.



Reference image is given [5]

Figure IIdifferential drive parameters

To achieve the no slippage condition, it is necessary that the ICC lies on the same axis as the driving wheels. This means both wheels are turning at the same angular velocity. Note that this is not the wheels' rotation speed, but the rate at which the wheel moves around ICC. This condition gives us the information to write our initial equations of motion:

$$\omega(R + d) = V_L$$

$$\omega(R - d) = V_R$$

Solving these two gives us the location of ICC as

$$R = d\frac{V_R + V_L}{V_R - V_L}$$

And the overall velocity of the robot becomes

$$V = \frac{V_R + V_L}{2}$$

Where the Vx and Vy can be found out using the angle at the given time frame. Where angle is calculated by

$$\vartheta(t) = \int \omega(t)dt$$

In the condition where both the wheels have equal and opposite velocities V, the term V_R+V_L takes the form V-V=0. Which means R=0, consequently the radius of curvature lies between the two wheels, thus achieving a small radius of turn.

But this was the radius of turn on just 2 wheels. The total radius of turn becomes different on 4 wheel, or more. The radius now incorporates the diagonal of the polygon made by the wheels. The easiest way to reduce the radius, is to reduce the length of diagonal.

If the castor wheels have already adjusted, the resulting radii of turn are shown for two configurations.

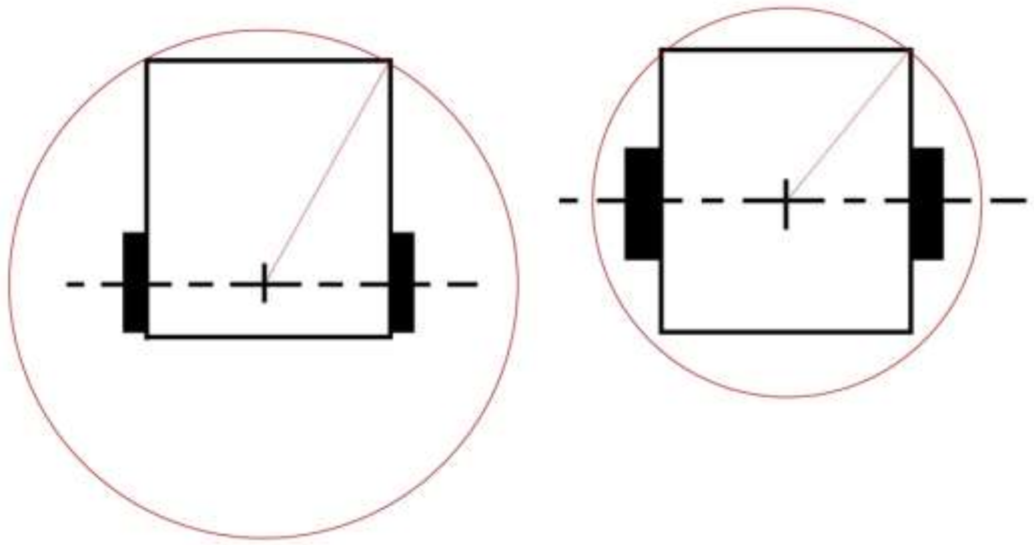


Figure III- Demonstration of diagonal relation with turning radius

The shortest diagonal, and hence the radius, is possible if the differential drive wheels are placed in the center of the vehicle. Therefor the approach to place the driving wheels in center was taken. But such configuration will introduce instability of the center of gravity shifts beyond both the wheel pairs, hence the 6 wheel design, with 4 castors and 2 fixed drivers.

<u>3.5 Path to the target :Optimized</u>

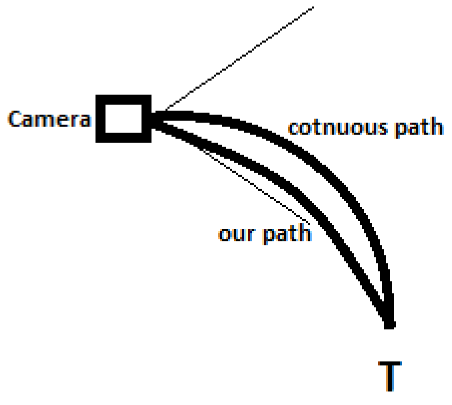


Fig: Continuous path and optimized path

It is seen with practical application that with continuous oath the target goes out of the vision of the cart easily but in our optimized path cart vision des not lose the target.

Working out the radii experimentally:

We marked two end points on the ground, first the cart ran straight through them and we recorded the time then iteratively set the cart on a path such that the circular path intercepts the end point of the straight path, So we drew a line of 1.5m. Then we performed the experiment. The radius of turn would then be 0.75m. When we figured the pwm values for it, we doubled the length. The line was now 3n long. Now the radius would be 1.5m. And then we figured the pwm values again.

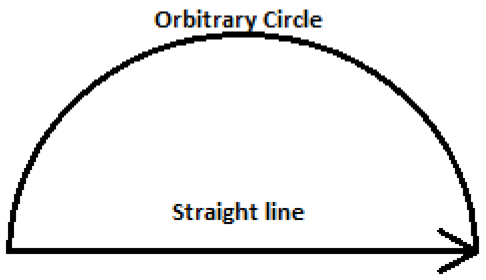


Fig: Demonstration of working out turning radii

3.6 Turning information

Given the cart's length of approximately 0.7m and the optimal distance from the target to be 1m. We decided that if cart reaches within 1.7m of the target, it would register as "target successfully followed".

Using the discrete intervals from the angles we decide next. That if cart were within the range, but it's not in the central interval, it would spin on it's own axis to bring the target to central interval. If it were out of range and in central interval, it would move straight. And if out of range and not in central interval, it would move while turning. And so on. Decisions are revised every instant. Each instant being the time taken for one successful detection.

Finally you can mention the mode of communication between the pi and the Arduino. Which was digital pins. 4 pins that turn high or low. And what information does each pin carry. Possibility for decisions is 16 but we are using only 10. Pin 1 carries far or close. Pin 2 carries: right. Pin 3 carries left

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Voltage Cycles for motors

Nominal values of torque and rpm are given but it has internal worm gear with gear ratio of 1:15. Our requirement per motor is 11 Nm for our weight limits and using torque speed curve the speed is 176 rpm. Our driving wheel diameter is 22 cm and overall movement speed is approximately 2m/s. The average walking speed for a human is 1.4m/s but inside the mall is about 1m/s. So we achieve this value of speed at 50% pwm value that is 128 pwm. The practically comfortable values are 110pwm.

Table 1 Motor properties

nominal torque	3 Nm
nominal rpm	3200
gear ratio	1:15
torque with gear ratio	52 Nm
rpm with gear ratio	222
Voltage	30 V
Current	4.9 A
Requirement of torque per motor	11 Nm
Required rpm	176
Diameter of driving wheel	22cm
Speed of cart	2m/s
Average human walking speed	1.4m/s

Human movement Speed inside a mall	~1m/s
For 1m/s pwm is required	50%
Speed to be achieved at pwm	128/255
Practical value of pwm	110 voltage cycles

In order to calculate the velocity for both the motors we have used the following formulas were used. These are used to calculate the velocity for right and left wheel for any given conditions' the half distance between two wheels and r is the radius of turn to be used. We used theoretical pwm for achieving the required velocities but the actual practical velocities show anomalies. The reason for different response to theoretical pwm values because motors are not new. They are in fair conditions to use but not new because of that they show anomalies.

$$\omega(R + d) = V_L$$

$$\omega(R - d) = V_R$$

$$R = d\,\frac{V_R + V_L}{V_R - V_L}$$

$$V = \frac{V_R + V_L}{2}$$

Turn	Theoretical speed	Values pwm	Practical Values Pwm
Acw	V_R=0.982	`126	105

0.75	V_L=0.730	`93	95
Cw 0.75	V_R=0.730 V_L=0.982	`93 `126	75 125
Acw 1.5	V_R=1.108 V_L=0.605	`142 `77	115 85
Cw 1.5	V_R=0.605 V_L=1.108	`77 `142	60 130

4.2 Detections and Adjustment

We trained the model for nearly 1 million iterations. It dtects more than 95% accuracy

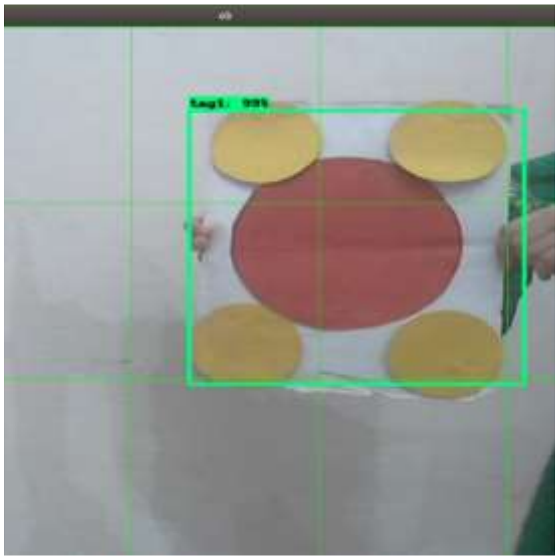
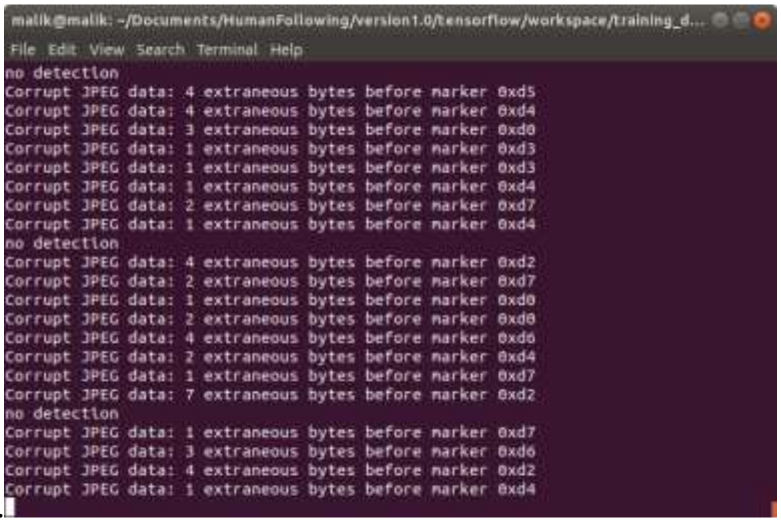


Fig showing detections with 99% accuracy

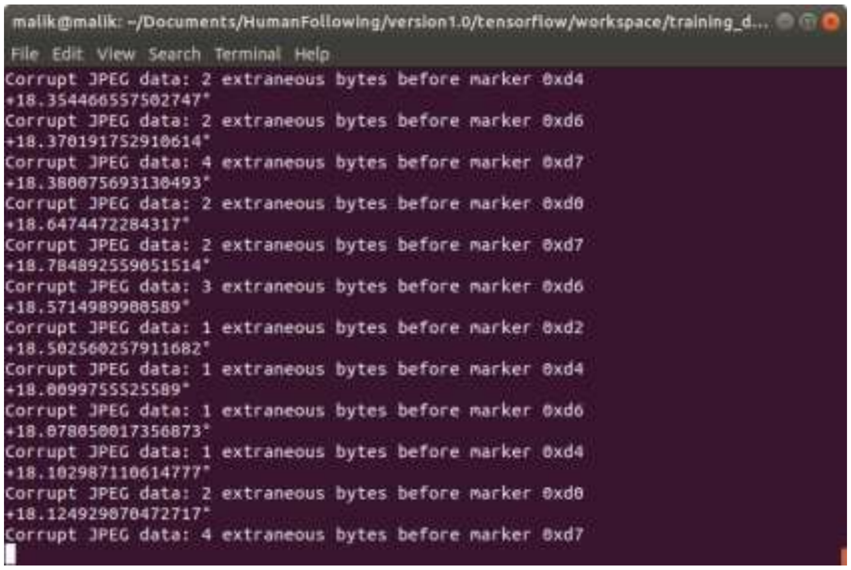
It can also tell us when the object is not detected



```
malik@malik: ~/Documents/HumanFollowing/version1.0/tensorflow/workspace/training_d...
File  Edit  View  Search  Terminal  Help
no detection
Corrupt JPEG data: 4 extraneous bytes before marker 0xd5
Corrupt JPEG data: 4 extraneous bytes before marker 0xd4
Corrupt JPEG data: 3 extraneous bytes before marker 0xd0
Corrupt JPEG data: 1 extraneous bytes before marker 0xd3
Corrupt JPEG data: 1 extraneous bytes before marker 0xd3
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
Corrupt JPEG data: 2 extraneous bytes before marker 0xd7
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
no detection
Corrupt JPEG data: 4 extraneous bytes before marker 0xd2
Corrupt JPEG data: 2 extraneous bytes before marker 0xd7
Corrupt JPEG data: 1 extraneous bytes before marker 0xd0
Corrupt JPEG data: 2 extraneous bytes before marker 0xd0
Corrupt JPEG data: 4 extraneous bytes before marker 0xd6
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
Corrupt JPEG data: 1 extraneous bytes before marker 0xd7
Corrupt JPEG data: 7 extraneous bytes before marker 0xd2
no detection
Corrupt JPEG data: 1 extraneous bytes before marker 0xd7
Corrupt JPEG data: 3 extraneous bytes before marker 0xd6
Corrupt JPEG data: 4 extraneous bytes before marker 0xd2
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
```

Fig: showing no detection

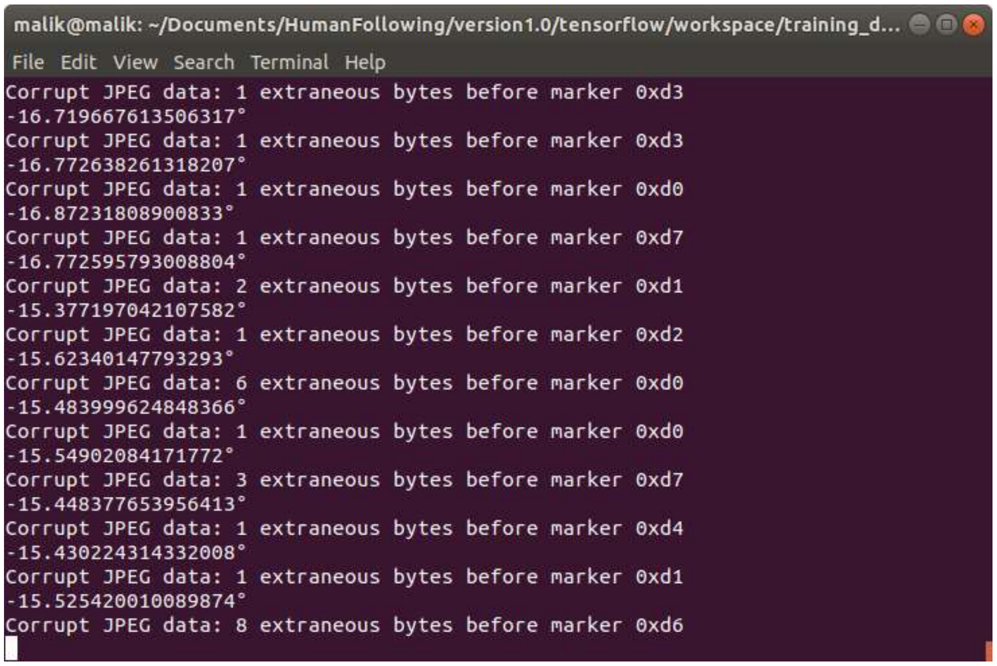
Detections on the positive angle



```
malik@malik: ~/Documents/HumanFollowing/version1.0/tensorflow/workspace/training_d...
File  Edit  View  Search  Terminal  Help
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
+18.354466557502747°
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
+18.370191752910614°
Corrupt JPEG data: 4 extraneous bytes before marker 0xd7
+18.380075693130493°
Corrupt JPEG data: 2 extraneous bytes before marker 0xd0
+18.6474472284317°
Corrupt JPEG data: 2 extraneous bytes before marker 0xd7
+18.784892559051514°
Corrupt JPEG data: 3 extraneous bytes before marker 0xd6
+18.5714989900589°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd2
+18.502560257911682°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
+18.0099755525589°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd6
+18.078050017356873°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
+18.102987110614777°
Corrupt JPEG data: 2 extraneous bytes before marker 0xd0
+18.124929870472717°
Corrupt JPEG data: 4 extraneous bytes before marker 0xd7
```

Fig showing detections at positive angle

Detections with negative angle.


malik@malik: ~/Documents/HumanFollowing/version1.0/tensorflow/workspace/training_d...
File Edit View Search Terminal Help

```
Corrupt JPEG data: 1 extraneous bytes before marker 0xd3
-16.719667613506317°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd3
-16.772638261318207°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd0
-16.87231808900833°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd7
-16.772595793008804°
Corrupt JPEG data: 2 extraneous bytes before marker 0xd1
-15.377197042107582°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd2
-15.62340147793293°
Corrupt JPEG data: 6 extraneous bytes before marker 0xd0
-15.483999624848366°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd0
-15.54902084171772°
Corrupt JPEG data: 3 extraneous bytes before marker 0xd7
-15.448377653956413°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd4
-15.430224314332008°
Corrupt JPEG data: 1 extraneous bytes before marker 0xd1
-15.525420010089874°
Corrupt JPEG data: 8 extraneous bytes before marker 0xd6
```

Fig showing detection at negative angle

We divided camera vision into 5 divisions given them each 10° but later practically we saw that higher turn rate regions require larger segments . So we gave 15, 10,10,10,15 degrees left to right respectively. Overall fps is slow that gives delay and human turning are not smooth rather abrupt.

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

We were able to successfully transfer rcnn network to raspberry pi 3b+ and we were able to produce enough fps on the detection. We had a success rate of more than 95%" and finally we were able to translate this information into motion decision in reality based time.

5.1 Application and Uses and Future Advancements

Obstacle Avoidance

If we use a double camera/"time of flight" camera, which gives us possibility to measure distance of objects. Then we can use multiple tensorflow models and detect if objects were in between the cart and the tag, and thus deploy collision avoidance.

Tagless Following

It can be trained to follow a human without the need of a tag. Still a long way from development.

Line Following Robot

If we train a single robot for multiple tags, we can make it run through an entire course using tags as checkpoints and thus make line following robots obsolete (as this one can do more stuff).

REFERENCES

[1] Object following control of six-legged robot using Kinect camera Amruta Vinod Gulalkari ; Giang Hoang ; Pandu Sandi Pratama ; Hak Kyeong Kim ; Sang Bong Kim ; Bong Huan Jun

[2] https://medium.com/@teyou21/convert-a-tensorflow-frozen-graph-to-a-tflite-file-part-3-1ccdb3874c4a

[3] https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html

[4] https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[5] Discretization: An Enabling Technique. Editors: Fayyad, Mannila, Ramakrishnan