Surveillance System for Autonomous Survey Vehicles

by

Jeyalakshmi Thoppe Subramanian

Master Thesis in Electronics- Microelectronics and Sensor Technology

Number of Credits:60

Department of Physics

The Faculty of Mathematics and Natural Sciences

Spring 2021

Summary

Autonomous Survey Vessels, also known as Autonomous Survey Vehicles (ASV) are used for surveying of lakes in Norway. This Master thesis explains about development of a surveillance system for Autonomous Survey Vehicles.

This system consists of three parts – detection of the objects in the direction of ASV, measurement of the obstacle distance and deviation of ASV away from the obstacle to avoid collision.

The objects in the ASV path are classified and located in the image by using Raspberry Pi, PiCamera and machine learning with pre-trained object detection model from TensorFlow object detection repository. The object detection model is trained with custom data set to detect objects in lake environment.

Distance of the obstacle in ASV path is estimated by measuring and combining data from Ultrasound and Laser sensor. The Ultrasound sensor measurement includes temperature and humidity correction. Influence of obstacle shape, surface and ambience light on sensor measurement is studied.

By using information from object detection and obstacle distance measurement, ASV is steered away from obstacle to avoid collision.

Forward

Thanks to my supervisor Helge Balk for steering the thesis in correct direction with inputs and intervention at required stages.

Thanks to Sverre Holm for inputs on Ultrasound sensors and surface roughness.

Thanks to UiO for all the facilities and Espen Murtnes for answering all my queries immediately without fail.

Thanks to Norway for prioritising lives.

Jeyalakshmi

Contents

List of Figures

List of Tables

Abbreviations

AI - Artificial Intelligence
ASV - Autonomous Survey Vessel or Vehicle
API- Application Programming Interface
BLE-BLuetooth Low Energy
cm- centimetre
CNN-Convolutional neural network
CPU-Central Processing Unit
CSI- Camera Serial Interface
COCO-Common Objects in Context
fps-frame per second
FPN-Feature Pyramid Network
GPIO-General Purpose Input Output
GPU-Graphical Processing Unit
GUI- Graphical User Interface
GPS-Global Positioning System
GNSS- Global Navigation Satellite System
I2C-Inter-Integrated Circuit
IDE -Integrated Development Environment
IR- InfraRed
IMU-Inertial Measurement Unit
kNN- k Nearest Neighbours
LED- Light Emitting Diode
LIDAR- Light Detection and Ranging
ML- Machine Learning
mm- millimetre
SSD- SingleShot Multibox Detector
Std.Dev -Standard Deviation
OS-Operating System
OpenCV- Opensource Computer Vision
OpenMV- Opensource Machine Vision
OS- Operating System
YOLO- You Only Look Once
PLC-Programmable Logic Controller
PSD- Photo Sensitive Detector
PWM- Pulse Width Modulation
PoE- Power-over-Ethernet
ReLU- Rectified Linear Unit
SLAM- Simultaneous Localisation And Mapping
SSH- Secured Shell Protocol
SDA-Signal Data Line
SCL-Signal Clock Line
TOF- Time Of Flight
UiO- University I Oslo
USD- US Dollar
YOLO- You Only Look Once

1 Introduction

In accordance with "EU water frame work directive- Directive 2000/60/EC of the European Parliament and of the Council of 23 October 2000 establishing a framework for Community action in the field of water policy" [2], Norway is obliged to undertake survey of all the lakes above a certain size to make survey of fish and ecosystem of lakes. These surveys are usually carried out with Survey vessels with hydro-acoustic measurements using echo sounders. As manual surveying of lakes may not be feasible due to economic constraints and suitability of surveying methods and equipment, it is preferred to undertake the survey using autonomous methods. Also hydro-acoustic surveying of lakes may be mostly undertaken at night due to minimum disturbance, better data collection about fish life etc.

The Department of Electronics at UiO (University I Oslo) is developing an Autonomous Survey Vehicle (ASV) which will be deployed for lake survey. Earlier work in this direction is a low cost automatic survey vessel platform [3], which can be fitted on any boat and this boat can be remotely controlled during survey or pre-plan the route for this boat. It consists of two electric outboard engines, motor control system for the engines and a simple autopilot system.

An important requirement for such a survey vessel will be to avoid obstacles on its path. For this purpose, a simple and cost effective Surveillance system is designed in this thesis. The main purpose of this system will be to get the details of the objects in its direction, estimate the object distance and develop an autopilot system with obstacle avoidance.

To prove the concept and check the feasibility of such system on land, an existing Hardware system (Model:Mars Rover PiCar- B, Make: Adeept) [9] at Electronics Department was chosen. This system is based on Raspberry PI controller and is fitted with different sensors like Camera, Ultrasound sensor and it has a motor to move the system forward or backward with servomotors to steer the PiCar B. As the PiCar-B (hereafter referred as PiCar) moves, the images in direction of PiCar movement is captured through the camera and Machine learning concepts are used to detect the object and classify the obstacles. The Ultrasound sensor measures the distance of the PiCar to the obstacle and the PiCar is steered away from the obstacles by controlling the servomotor connected to front wheels.

Along with the Ultrasound sensor, a Laser sensor was also used for obstacle distance measurement and data fusion between both the sensor data was used for better estimate of object distance. Temperature and relative humidity measurements from a sensor were used for sound velocity calculation in Ultrasound sensor measurement.

Organisation of thesis report:

The thesis report is organised as follows:

Chapter 2 explains the theory behind Object Detection, Distance Measurement and Obstacle avoidance.

Section 2.1 explains about Machine Learning, Object Classification, Object Localisation, Feature Extraction, CNN, different layers in CNN, training a CNN model, different types of object detector models, datasets, TensorFlow API, OpenCV library.

Section 2.2 explains about working principle of distance measurement sensors such as Ultrasound sensor, IR sensor and comparison between Ultrasound and Laser sensor.

Section 2.3 explains the basic theory behind steering the vehicle for obstacle avoidance.

Chapter 3 covers in detail the referred literatures about ASV, Object Detection, Image sensors, Distance measurement sensors, Distance Measurement, Obstacle avoidance, Mobile robots and others.

Chapter 4 covers the Materials and Methods used in this thesis. Section 4.1 and 4.2 gives details of the hardware and sensors used in this thesis such as PiCar, RaspberryPi, Arduino Uno, Arduino Nano, RaspberryPi camera module, servomotors, Ultrasound and laser sensor.

Section 4.3 lists the methods used in this thesis

Chapter 5 explains in detail the method for Object detection.

Chapter 6 explains in detail the methods used for distance measurement using ultrasound sensor, laser sensor on different targets and under different ambient lighting. Section 8.2 summarises the results for object distance measurement.

Chapter 7 explains the method used for navigating the PiCar for Obstacle avoidance.

Chapter 8 lists the results of object detection, Obstacle distance measurement and Obstacle avoidance.

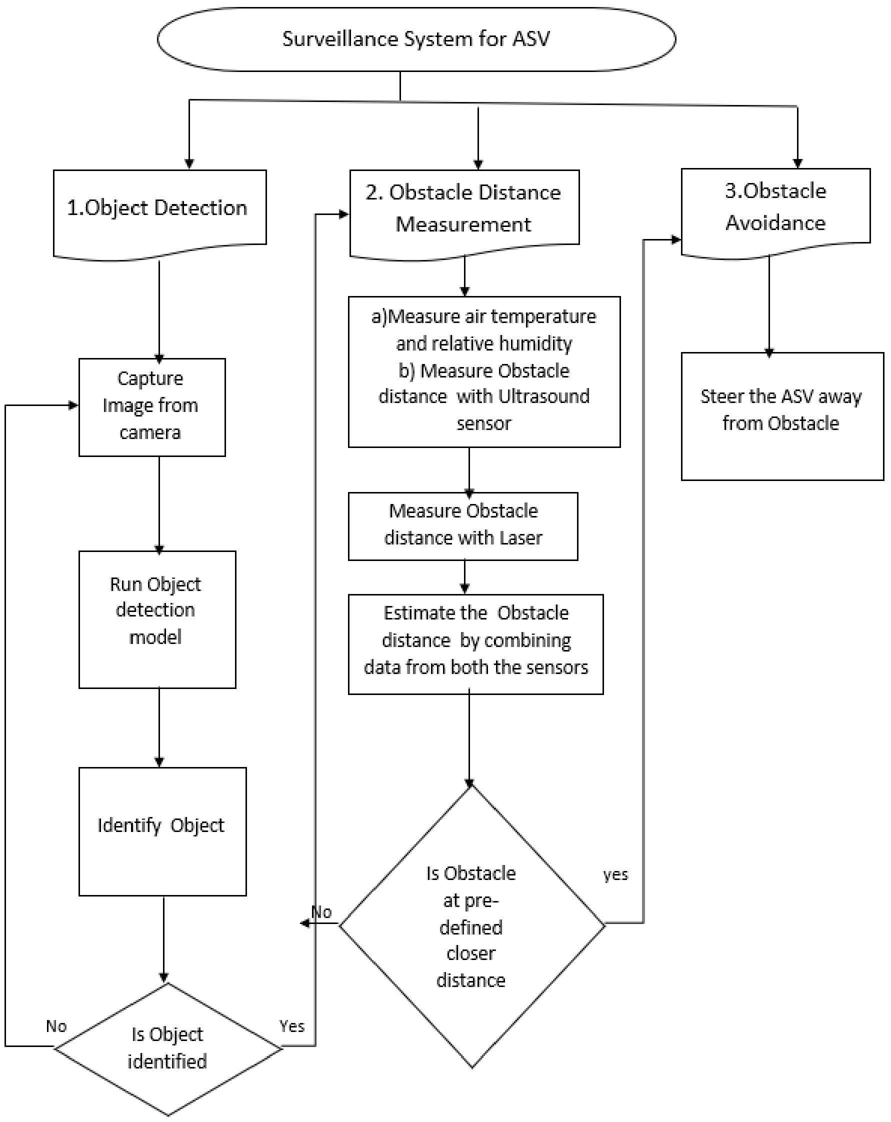
Chapter 9 discusses the results in detail, risk management and possible future extensions of this thesis.

2 Theory

This master thesis-Surveillance system for Autonomous Survey Vehicles consists of following main tasks :

- Object detection on the ASV path.
- Obstacle distance measurement
- Obstacle avoidance

The following flow chart shows the main processes:



Surveillance System for ASV
1.Object Detection
2. Obstacle Distance Measurement
3.Obstacle Avoidance
Capture Image from camera
Run Object detection model
Identify Object
Is Object identified
No
Yes
a)Measure air temperature and relative humidity
b) Measure Obstacle distance with Ultrasound sensor
Measure Obstacle distance with Laser
Estimate the Obstacle distance by combining data from both the sensors
Is Obstacle at pre-defined closer distance
No
yes
Steer the ASV away from Obstacle

Figure 1: Flowchart of Surveillance system of Autonomous Survey Vehicles

The theory and principle involved in each of the three main parts are presented in the following sections.

2.1 Theory about Object Detection

For developing a surveillance system, it is important to identify the objects in the environment. The images are first captured through the camera. In this project, PiCamera in the PiCar is used to capture the images. The objects in these images are identified from pre-trained object list. Machine learning with Convolution Neural Network (CNN) is used to identify the objects in these images. Specifically, Deep Learning with CNN is used. In this thesis, the term Machine Learning with CNN is used for broad understanding.

Main concepts about Machine Learning and Object detection are presented in this section.

2.1.1 Machine Learning

A machine is considered intelligent if it can perform tasks which are considered intelligent when carried out by a human being. Artificial Intelligence(AI) is a broad science and engineering category of building smart machines which are capable of performing tasks that typically would require human intelligence, for example, natural language processing.

Machine learning (ML) is a branch of artificial intelligence and computer science. It focuses on the use of data and algorithms to imitate the way the humans learn, gradually through experience [6].

There are three main branches of ML and each has its own area of application as shown below.

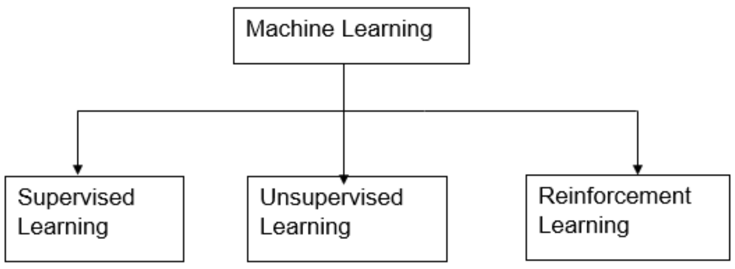


Figure 2: Branches of Machine Learning

Deep Learning (DL) is a subset of Machine Learning. The relationship between Artificial Intelligence (AI), Machine Learning(ML) and Deep Learning (DL) can be represented as below.

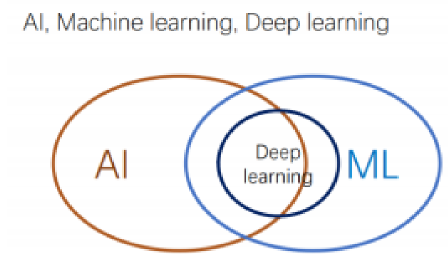


Figure 3: Relationship between AI, ML, DL. Image Reference [1]

In computer vision applications like object detection, the main difference between Machine Learning and Deep Learning is in feature engineering. More details are given in section 2.1.6

2.1.2 Algorithm

In general, an algorithm can be termed as a method to carry out a particular task or a set of steps to solve a particular problem. In machine learning, an algorithm can be defined as a set of well-defined steps which learns from input data, applies this learning on new input data. These algorithms can be implemented using any modern programming language. There are different algorithms for each category of Machine Learning.

2.1.3 Supervised Learning

In Supervised Learning, the machine is trained with labelled data and when new data is presented to the machine, it tries to relate the new data to any of the previously learned label.

Applications: Some of the applications of Supervised Learning are

Classification
- E-mail spam detection
- Image Classification

Regression
- Risk Assessment
- Weather forecast

Supervised Learning is the key element in Image classification. Figure 4 illustrates how Supervised Learning is used to train the computer by using pictures of animals and corresponding label like duck, dog and cat to generate an object classifier model. When this model is presented with a new picture of duck which was not used for training, the model identifies and labels the picture as duck.

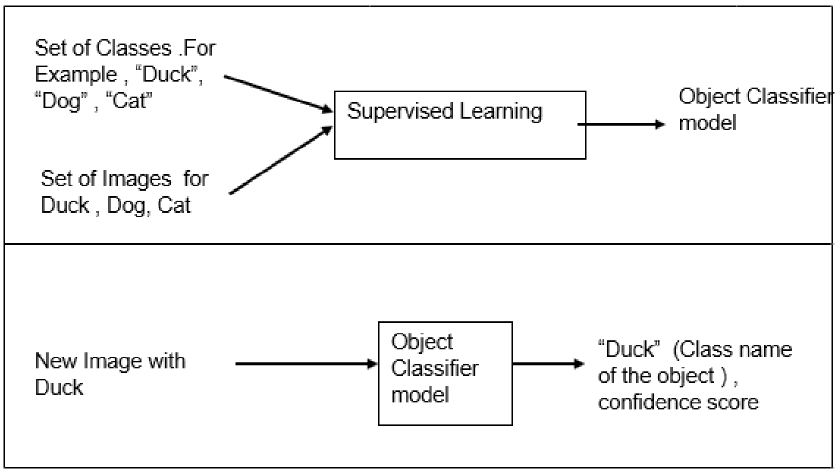

Set of Classes .For Example , "Duck", "Dog" , "Cat"
Set of Images for Duck , Dog, Cat
Supervised Learning
Object Classifier model
New Image with Duck
Object Classifier model
"Duck" (Class name of the object) , confidence score

Figure 4: Illustration of Supervised Learning. Model is trained with Images and Label. The trained model is used to predict label of new image.

Algorithms: Some of the algorithms for Supervised Learning are

Perceptron
kNN (K Nearest neighbours)
Linear Regression
Support Vector Machine(SVM)
Neural Networks

2.1.4 Unsupervised Learning

In Unsupervised Learning, the machine is trained with unlabelled data and it tries to find the similarities among the data and group the similar data together. Figure 5 illustrates this.

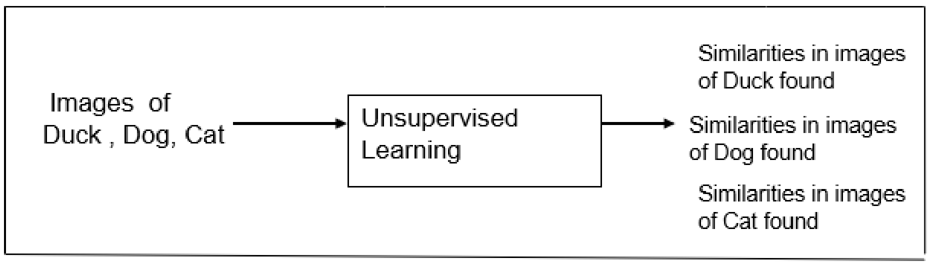

Images of Duck , Dog, Cat
Unsupervised Learning
Similarities in images of Duck found
Similarities in images of Dog found
Similarities in images of Cat found

Figure 5: Illustration of Unsupervised Learning. The model is trained to find similarities and groups the presented images of Duck, Dog, Cat.

Applications: Some of the applications of Unsupervised Learning are

Clustering
Targeted Marketing
Customer Segmentation
Dimensionality Reduction
Feature elicitation
Big data visualisation

Algorithms: Some of the algorithms for Unsupervised Learning are

k-means clustering
Principal Component Analysis

2.1.5 Reinforcement Learning

In Reinforcement Learning, the machine learns from its experience of interacting with the environment. The goal is to maximize an accumulated reward which is defined and given by the environment. The machine interacts with the environment via states, actions and rewards. It takes action in the environment and gets the feedback from the environment.
Applications: Some of the applications of Reinforced Learning are

Gaming
Robot Navigation

Algorithms: Some of the algorithms for Reinforced Learning are

Markov Decision Process
Q-Learning

As this thesis uses Object detection, which is one of the main applications of supervised learning, we will focus more on that.

2.1.6 Object detection

Object detection is a computer vision technique. It is one of main applications of supervised learning. Neural network algorithm can be used for creating an object detection model. Traditional machine Learning algorithms for supervised learning as Support Vector Machine (SVM) can also be used for object detection. But these require proper features for correct object classification. Hence, proper feature extraction should be designed for these algorithms to perform well. But neural network algorithm, like Convolutional Neural Network (CNN) extracts the features directly from the data at various levels. More details are given at section 2.1.7 and 2.1.8

When an image is given as input to an object detection model, it classifies one or more objects found on the image with a class probability score. Also, it gives the location of the objects in that image. It is a combination of two tasks - object classification and object localisation.

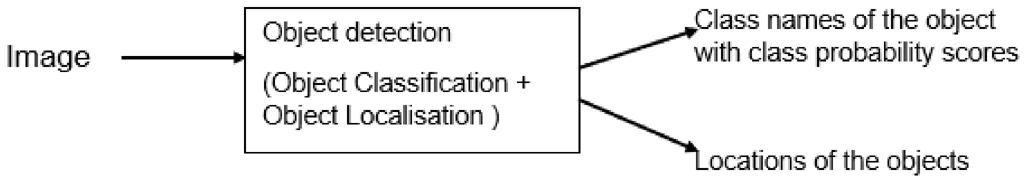


Image
Object detection
(Object Classification + Object Localisation)
Class names of the object with class probability scores
Locations of the objects

Figure 6: Object detection process.

As shown in the example below, the object detection model classifies two objects found on the given image - bottle and book, with a score. This probability score is termed as confidence score. Also, it locates those objects in the image with a bounding box.

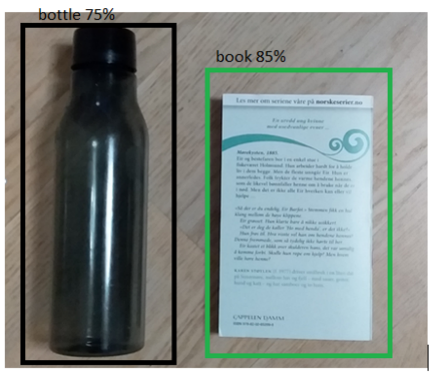


Figure 7: Example of Object detection from an Image -Two objects "book" and " bottle" are identified with some confidence score and object locations are shown on the image. Image reference [65]

Object classification Object classification technique predicts only the class of objects in the image with confidence scores.

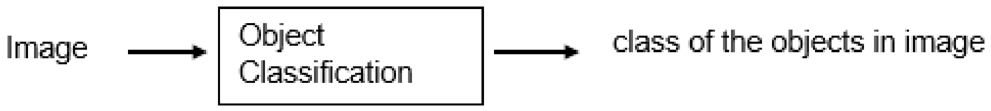


Figure 8: Object classification process.

As the example shown below, the object classification model predicts the class of the objects found on the image with confidence score.

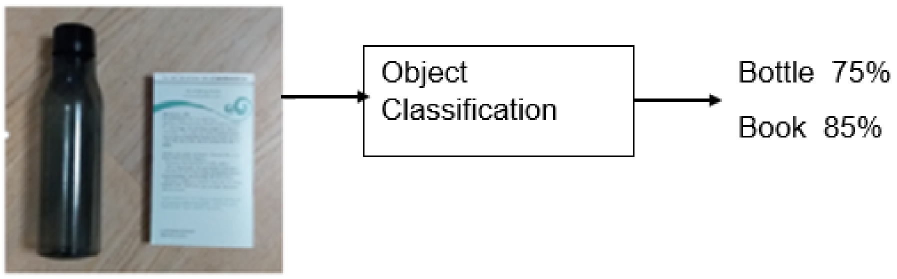


Figure 9: Example of Object Classification- two objects on the image "book" and "bottle" are predicted with some confidence scores.

Object localisation Object localisation technique involves identifying the location of one or more objects in an image and drawing bounding box around them.

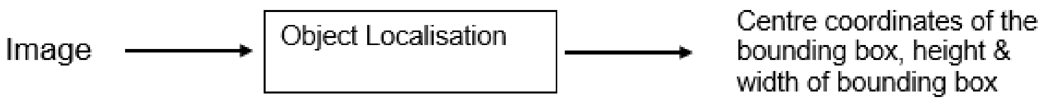
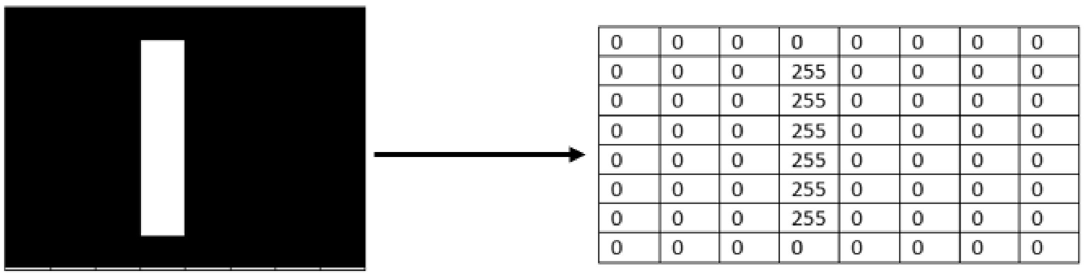


Image
Object Localisation
Centre coordinates of the bounding box, height & width of bounding box

Figure 10: Object Localisation process.

As mentioned in section 2.1.6, object detection models can be based on Neural Networks. One popular realisation of Neural networks is the Convolutional Neural Network. The basics of CNN and image feature extraction are given below.

In digital images with formats as jpeg, the details of images are given by the pixel information of the digital image. The pixels are represented from value 0 to 255 i.e by one byte. The value of the pixel describes the brightness of the pixel. For example, if it is a greyscale image, value 0 represents black and value 255 represents white. The digital image is normally represented as a matrix of pixel value numbers. The following figure illustrates pixel representation of a greysclae image as matrix of pixel values.

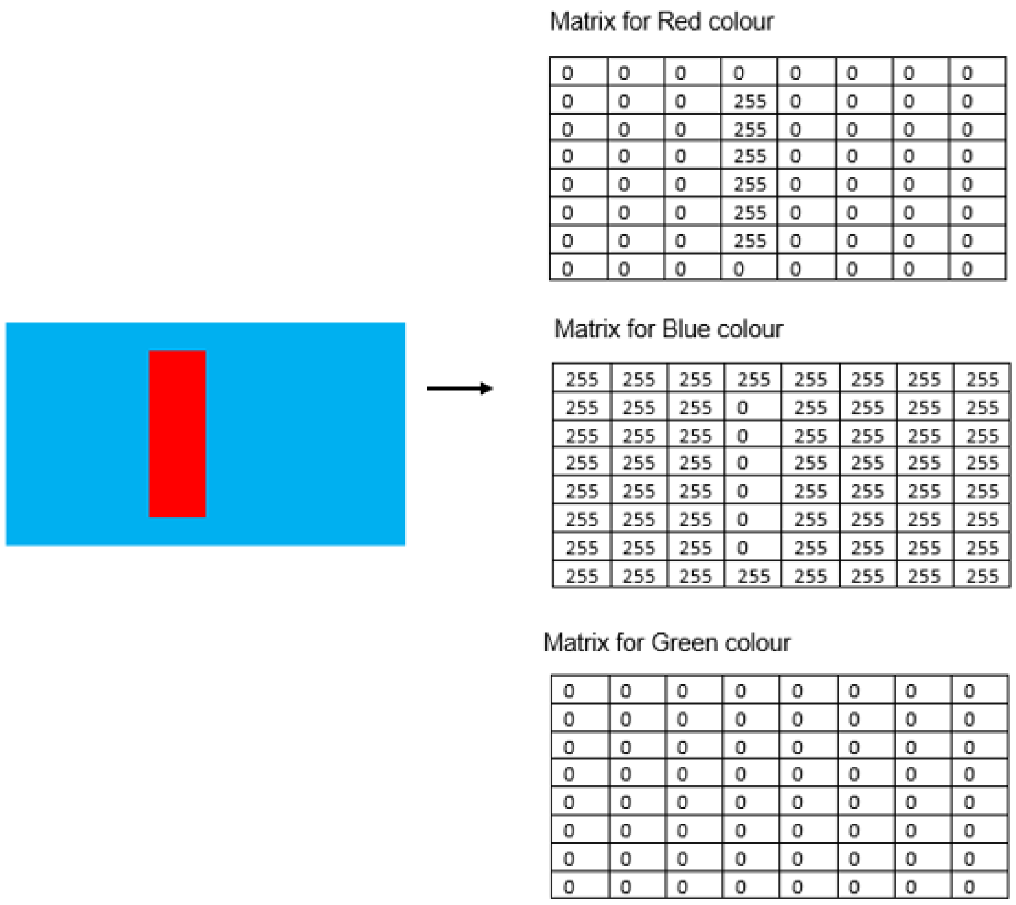


0	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	0	0	0	0	0

Figure 11: Illustration of GreyScale image pixel representation in 8 X 8 matrix.

In the above image, a black colour pixel is represented as 0 and white colour pixel as 255.

Similarly for representing colour images, three matrices are used for colours- Red, Green and Blue. Each matrix has values between 0-255 representing the intensity of the color for that pixel. The following figure illustrates representation of a colour image as Red, Green and Blue (R, G, B) matrices.



Matrix for Red colour

0	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	255	0	0	0	0
0	0	0	0	0	0	0	0

Matrix for Blue colour

255	255	255	255	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	0	255	255	255	255
255	255	255	255	255	255	255	255

Matrix for Green colour

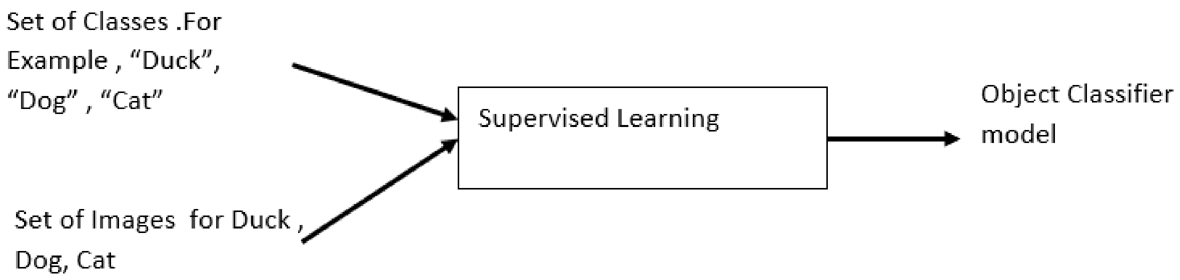
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 12: Illustration of Colour image pixel representation in R,G,B 8 X 8 matrices.

Consider a pixel with only blue colour in the above image. That pixel is represented by 255 in matrix for Blue colour and by 0 in other two matrices. Similarly, a pixel with only red colour is represented by 255 in matrix for Red colour and by 0 in other two matrices. Since the image does not contain any green pixels, the matrix for Green colour has all zeroes.

As shown in figure 4, classification of image with supervised learning involves two steps. First step is to create a Image Classification model using Supervised Learning and second step is to use this model to predict the class of a new image.

To generate an image classification model, we need to have a defined set of images and a set of classes under which we want to classify the image objects. So, the goal of Object classifier can be specified as mapping from set of images to set of classes.



Set of Classes .For Example , "Duck", "Dog" , "Cat"
Set of Images for Duck , Dog, Cat
Supervised Learning
Object Classifier model

Figure 13: Generating an Object classifier model.

Once this model is created, it is used to predict the object class in a new image.

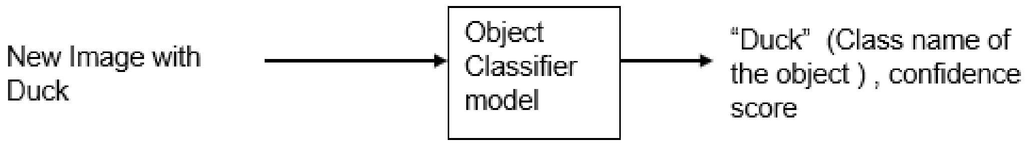
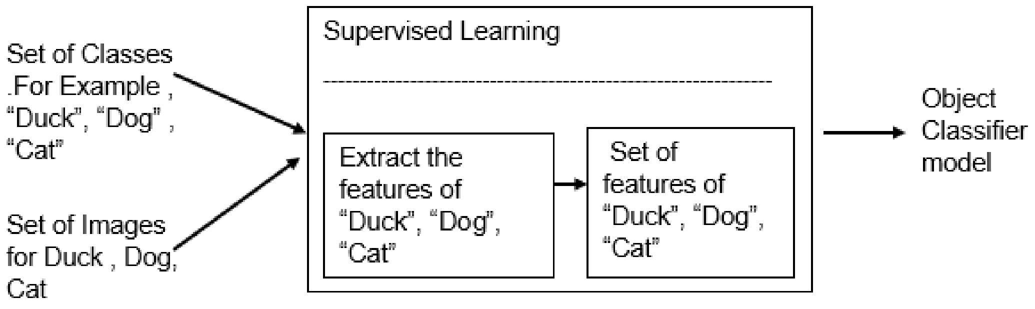


Figure 14: Predicting class of a object in new image using the generated object classifier model.

To classify a given image as "Duck" it is required to extract the features of this image and classify these features as "Duck".

Now the figure 13 can be shown as,



Set of Classes .For Example , "Duck", "Dog" , "Cat"
Set of Images for Duck , Dog, Cat
Supervised Learning
Extract the features of "Duck", "Dog", "Cat"
Set of features of "Duck", "Dog", "Cat"
Object Classifier model

Figure 15: Generating an Object classifier model.

In traditional Machine learning, we need to hand-craft the features.i.e to explicitly specify the important features. In Deep learning, feature extraction is carried out automatically[8]. It is about extracting relevant information from presented data automatically and applying it to analyse new data.

Deep learning is based on Artificial Neural Networks (ANN) which are inspired by the neuron model of human brain. Convolutional Neural Networks (CNN) are special type of ANN. These are widely used for computer vision applications.

The supervised learning models can be based on Convolution Neural Network(CNN) which is explained in section 2.1.7 and 2.1.8.

2.1.7 Feature Extraction

Consider the following picture of two ducks swimming in water. We can identify the ducks mainly due to the shape of the duck and also by the colours.

Figure 16: Picture of Two ducks in water. Image reference [65]

If we need to train a machine to identify a duck, we need to extract the shape features of the duck such as beak shape, head shape etc. These shapes are made of basic shapes like horizontal lines, vertical lines and so on.

These basic shape details are represented by matrices called **kernels** or **filters**. To extract the features of the image, sections of the input image are convoluted with these kernel matrices. Convolution is an operation where all elements in one matrix are multiplied by all elements in another matrix in a sequence. Due to this, these models are called **Convolutional Neural Network (CNN)** models. The result of such multiplication can show whether a particular section of the image contains the feature represented by a specific kernel or not.

An application of feature extraction is given in the Master Thesis of Halvard Yri Adriaenssens **"Automatisk utsetting og innhenting av sensorer i innsjø"** [4]. Here, Halvard uses a sobel filter [7] to detect the vertical lines from the camera image. Short description of the thesis is given under the Chapter 3 Literature Survey.

2.1.8 Convolutional neural network (CNN)

Consider the highlighted section of the input image as shown in the figure 17 and assume that we have a filter which represents an arc as shown in figure 18.

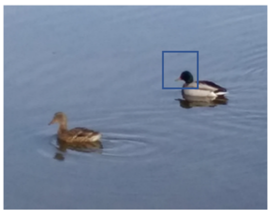


Figure 17: Picture of Two ducks in water with a section of the image highlighted.

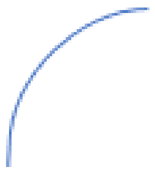


Figure 18: Depiction of a feature - Right-side curve.

If we convolute the image matrix in figure 17 with filter matrix of arc in figure 18, we will get a resulting matrix with high values at sections matching the arc and low values elsewhere. The resulting matrices from convolution are evaluated by **activation functions**. This helps in non-linear transformation of input image. There are different activation functions available. For example, the activation function **ReLU** (Rectified Linear Unit) which is defined as **max(o,x)** outputs either the positive maximum or zero among the given inputs. These convolution and activation operations are represented as **convolution layers** in CNN models as shown in figure 20. Output of one convolution layer is given as input to the next convolution layer.

As specified previously, the input colour image contains R, G, B (Red, Green, Blue) matrices. The sizes of these matrices are determined by the image resolution and these sizes grow as the resolution of the image grows. Thus convolution process on images involves lot of array operations. To reduce the number of computations and to reduce the spatial size of the CNN model, **pooling layers** are applied. A number of pooling operations are available. For instance **Maxpooling** function selects only the maximum value among the given inputs. For example, Let us assume that output feature map after the activation function in a convolution layer is given as 4X4 matrix in figure 19. If we define a maxpooling function which can identify maximum value of 2 X 2 square, then the output after Maxpooling operation will be as shown in figure 19.

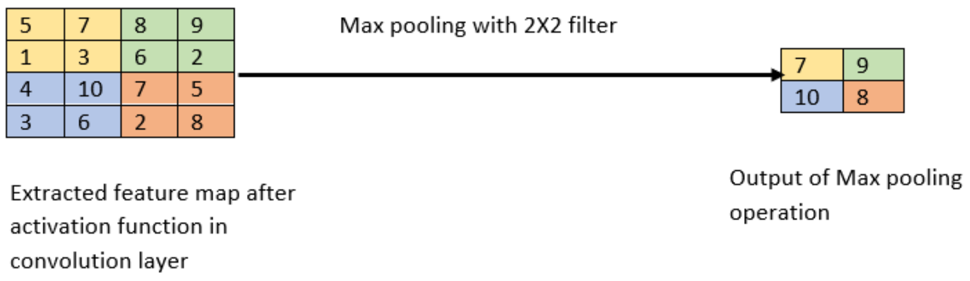

5 7 8 9
1 3 6 2
4 10 7 5
3 6 2 8
Max pooling with 2X2 filter
7 9
10 8
Extracted feature map after activation function in convolution layer
Output of Max pooling operation

Figure 19: Explanation of Maxpooling operation

The last layer in the CNN architecture is the **fully connected layer**. The probability of final extracted features belonging to a particular object class is given by this layer, by using ground truth labels and through variables called **weights**. An activating function called **softmax activation** function is used for this.

With all the layers as explained above, CNN network or model can be depicted as following:

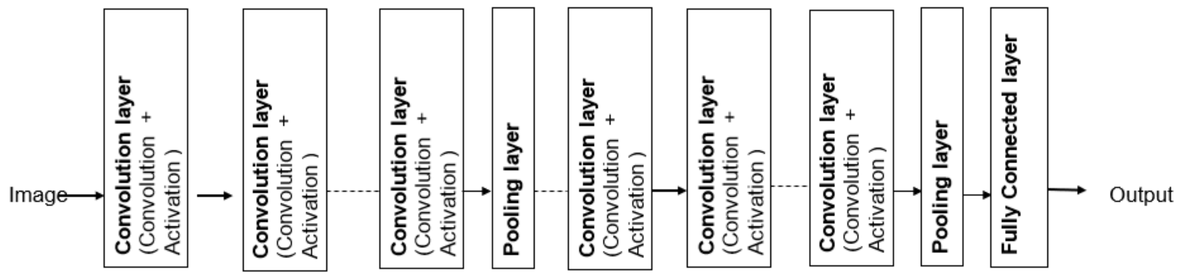


Image
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Pooling layer
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Pooling layer
Fully Connected layer
Output

Figure 20: CNN model with convolution, activation, pooling and fully connected layers

2.1.8.1 Training a CNN model

At the initial stage, we have images with labels. Let us assume that we have different images of "Duck" labelled as "Duck". With the CNN model we are trying to extract the features of "Duck" by using kernels. By training a CNN model we mean that the model is progressively trained to extract features of the images to match the label to a large extent.

At the beginning, these kernels are randomly initialised. Hence, if we compare the final extracted feature map as given by the fully connected layer to the given label, it may have lot of variation. This variation or difference between the given label data and extracted feature data is calculated by **Loss or Error** function. There are a number of Loss functions available. But the Most popular one is **MSE- Mean Square Error** function which is given as

$$MSE = \frac{1}{2}(expectedoutput - calculatedoutput)^2 \tag{1}$$

All the steps from left to right in figure 21 i.e from convolution to calculation of loss, is termed **Forward Propagation**.

As mentioned previously, at initial stages the loss will be high since the kernels are initialised randomly, and the kernels may not represent the features to be extracted correctly. As a result, the label predicted by the CNN model will vary largely from the actual ground truth label. But we want to reach a state where the predicted label matches with the provided ground truth label. For this, the weights which contribute more to the loss need to be identified and adjusted in such a way that the loss decreases (as in an optimisation problem). One of mostly used Optimisation algorithm is the **gradient descent optimisation algorithm**. The weights are updated from the Fully Connected layer to first convolution layer. This process is called as **Backward Propagation**.

The Forward and Backward propagation steps are carried out till the loss value decreases and falls within some acceptable range. This is termed as training the CNN model. The process of forward pass, loss function, backward pass, and parameter update is one training iteration. At the beginning of training, the loss value will be high and as the training progresses and the weights get updated, the loss value will reduce progressively. Parameter like learning rate is used for such updates. The following figure shows the above process :

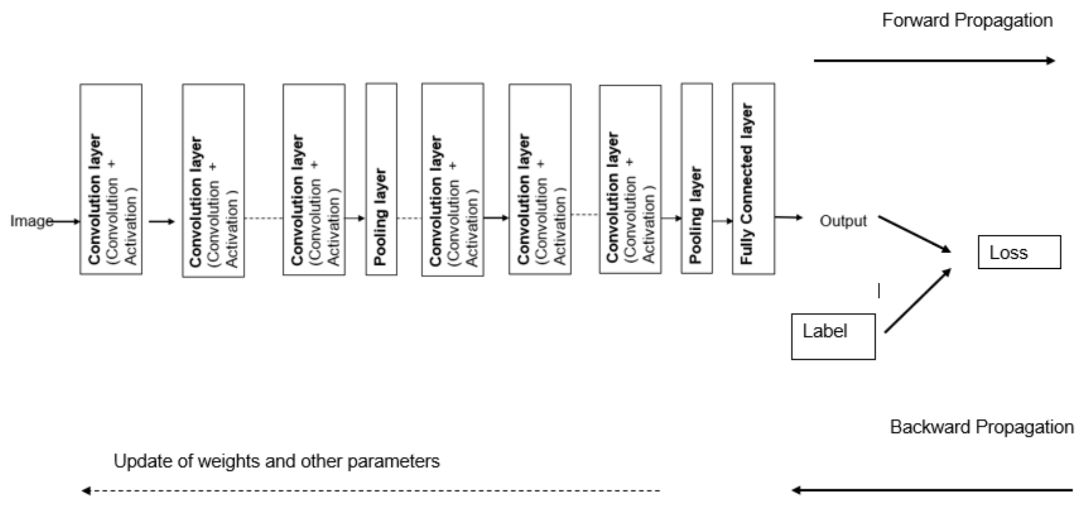

Forward Propagation
Image
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Pooling layer
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Convolution layer (Convolution + Activation)
Pooling layer
Fully Connected layer
Output
Loss
Label
Backward Propagation
Update of weights and other parameters

Figure 21: CNN model shown with Forward and Backward Propagation process

2.1.8.2 Testing a CNN model

After the training of CNN model is completed, we need to check whether our CNN model works correctly or not. To do so, we apply a different set of images and labels other than the training set. These images and labels are passed through the trained CNN model and outputs from the model is compared to the ground truth labels.

2.1.8.3 Feature extraction layer

As mentioned in section 2.1.7, feature extraction layer captures interesting parts of the image. In this project, **MobileNet** architecture is used. MobileNets are a family of neural network architectures released by Google. It is a CNN architecture model for Image Classification which consumes less computation power to run. Organisation of convolution operation in MobileNet architecture is the main reason for less computation power. Here, the convolution operation is organised as separable modules as depthwise and pointwise convolutions. This reduces the amount of computation[11]. Hence, this architecture can be used on machines with limited computing power, like mobile devices, embedded systems and computers without Graphical Processing Unit(GPU). These provide reasonable accuracy, while requiring less memory and computing power. This makes them a very fast family of networks to use for image processing[59]. It supports classification, detection, and semantic segmentation. There are also other architectures as LeNet, Alexnet, ResNet etc.

2.1.8.4 Object detection layer

As shown in figure 6, Object detection involves object classification and localisation. Localisation layers are used on top of the feature extractor layers for object classification. These are fine-tuned to learn how to predict bounding boxes and class probabilities for the objects inside these bounding boxes. In this project, **SSDLite** architecture is used. This architecture uses MobileNet as base architecture followed by several convolutional layers for single shot object detection(SSD). Some other type of object detector models are presented below.

2.1.9 Types of object detectors

Multi-stage object detector models like Region based CNN(R-CNN) detects the objects in two stages. First, they find interesting regions in the image which may contain objects. These are called region proposals. After that, for every such region, they make a prediction of probable objects. These models are more accurate than Single stage object detector model, but slow as they need to run the detection and classification portion of the model multiple times.

Single stage object detector models require only a single pass through the image. They divide the image in fixed size of regions and for every region they predict all possible objects in one go. These models are fast and suitable for mobile devices and for real time applications. Some examples are Single Shot Detector(SSD), You Only Look Once(YOLO), SqueezeDet and DetectNet. The single stage object detector models use only one stage and are faster than multi-stage detector models, while multi-stage object detector models are more accurate.

2.1.10 Datasets

For training object detector models, we need a large number of images which are also labelled. These are grouped as datasets. There are a number of common datasets for training object detectors (for example, COCO dataset, Kitti dataset etc.) Common Objects in Context (COCO) is one of the most popular image datasets with applications like object detection, segmentation and captioning. It has around 90 classes. It contains 2.5 million labelled images and text. There are Kaggle datasets for fashion and accessories. In this project, **COCO** data set is used.

2.1.11 TensorFlow

In this project, an object detection Application Programming Interface (API), called TensorFlow is used with Python Programming. It is an opensource framework from Google for creating a deep learning network to solve object detection problems. There are already pre-trained models in the framework which is referred as Model Zoo ([55]). This includes a collection of pre-trained models trained on the COCO dataset, the KITTI dataset, and the Open Images Dataset. TensorFlow handles data in the form of multidimensional arrays called Tensors. So it can handle large amounts of data. The popularity of the TensorFLow is given by the following figure.

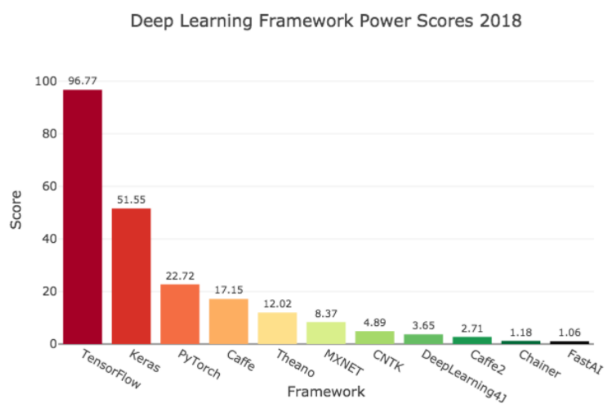


Figure 22: Deep learning framework power scores [10]

In this project, pre-trained SSDLite-MobileNetV2-COCO model from Tensorflow detection model zoo is used for object detection on the images with Python programming. Also opensource Python library OpenCV - is used.

2.1.12 OpenCV

Open Source Computer Vision Library (OpenCV) is an open source computer vision and machine learning software library. It provides a common infrastructure for computer vision applications. The library includes a comprehensive set of both computer vision and machine learning algorithms.[12]

2.1.13 Object detection with a pre-trained CNN model

Pre- trained model means that the object detection model is already trained to detect certain objects. We can directly use this to detect these objects from images. For example, model SSDLite-MobileNetV2-COCO is trained on the COCO dataset to detect 90 objects like "orange", "banana", "book", "tree", "person", "cup " etc. This model is downloaded from the TensorFlow model zoo and the training procedure is followed as given at [55].

Method of object detection using pre-trained SSDLite-MobileNetV2-COCO is given in section 5.1

2.1.14 Training CNN model with custom dataset

It is possible to train the object detection model with custom dataset. Generally, large amount of data is required for efficiently training a model. For example, consider that we want to train a model to detect five objects like boat, ball, bird, bus, train. Then, We need to collect hundreds of images for each these objects. We need to label each of these image as boat, ball, bird, bus, train in accordance with the object in the image. Labelling is done for all these images manually using labelling applications. The data is then split into training data set which typically contains 80 % of the data, and test data set which contains 20 % of the data. Training data set is used to train the object detection model. In the training process, loss values are calculated via forward propagation and the model parameters are updated via backward propagation. Test data set is used at the very end to evaluate the performance of the final model.

For example, as the ASV is developed for lake environment, the common objects or obstacles can be small boats, buoys, floating swim toys, thick tree branches etc. So it is possible to train our model to detect these objects with a dataset containing pictures of these objects.

2.2 Theory about Object Distance Measurement

Once an obstacle is classified from the captured image, distance between the obstacle and the PiCar needs to be measured. If the obstacle is very close, the PiCar should be steered away from the obstacle to avoid collision. Standard configuration of PiCar includes one number of Ultrasound sensor(Model: HC-SR04). There are different types of sensors available for distance measurement.

The theory, advantages, and disadvantages of different sensor types are given below:

2.2.1 Ultrasound sensor

An Ultrasound sensor has a transmitter and receiver. Ultrasound waves are transmitted from the transmitter. Part of the transmitted ultrasound energy is absorbed by the target in front of the sensor and part is reflected from the target which is sensed by the receiver of the sensor. The Time of Flight(TOF) between transmission of Ultrasound wave and receiving, is measured and used for calculation of distance of the target.

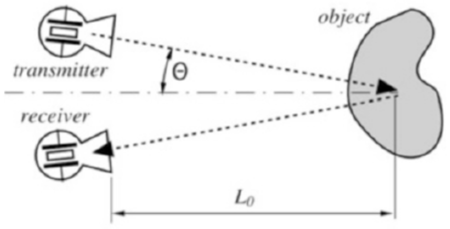


Figure 23: Working principle of Ultrasound sensor. Image reference. [13]

The distance Lo between object and Ultrasound sensor is calculated with the following equation

$$Ł_0 = (vtcos(\theta))/2 \tag{2}$$

Where *v* is speed of Ultrasound waves in media (here in air).

t is the time for ultrasound waves to travel from transmitter to the object and back to receiver (hence, the division factor 2).

$cos(\theta) = 1$ (if transmitter and receiver are positioned close to each other as compared with distance to the object).

Advantage of Ultrasound Sensors

• Not affected by object colour, transparency and environment lighting.

Disadvantage of Ultrasound Sensors

•Difficult to measure the distance of objects that have complex surface shapes.

2.2.2 IR(InfraRed)sensor

IR distance measurement sensor has an IR emitter(LED) and a photosensitive detector (PSD). It works through the principle of triangulation, measuring distance based on the angle of the reflected beam.

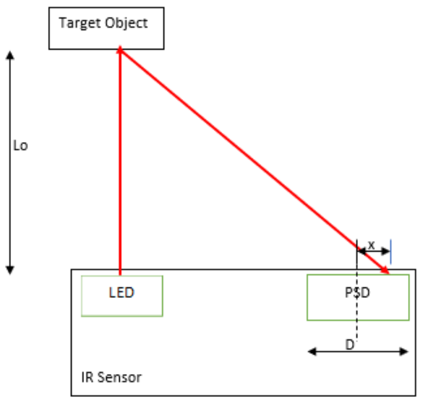


Figure 24: Working principle of IR sensor.

Figure 24 illustrates distance measurement by an IR sensor. IR beam is transmitted from transmitter. Part of this IR energy gets absorbed by the target and part is reflected back by the target and detected by the Photo Sensitive Device (PSD). The distance of the target is calculated as

$$Ł_0 = k(D/x) \tag{3}$$

Where k is sensor's geometrical constant.
D - distance between two electrodes in PSD.
x- distance of incidence of reflected IR beam at PSD from the reference centre of PSD.

As the object moves, incidence of the reflected beam at PSD changes, which changes *x*, and hence, measured distance *L_0* also changes.

Advantage of IR sensors

●Line of sight measurement

Disadvantage of IR sensors

●Affected by environmental lighting conditions and object's reflectivity of IR beam.

2.2.3 Laser sensors

Laser beam is transmitted from Laser sensor to object. By calculating the time taken between transmitting and receiving the laser beam and speed of light in air, distance between sensor and the object is calculated.

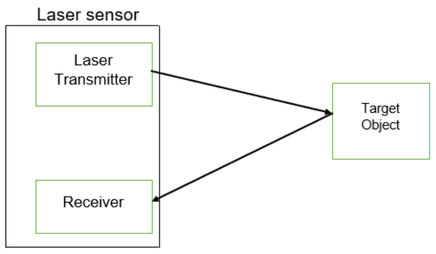


Figure 25: working principle of Laser sensor.

2.2.3.1 LiDAR

LiDAR(Light Detection and Ranging) is laser distance sensor which uses Time of Flight concept.

Advantages of LiDAR

• Realtively Long measurement range compared to Ultrasound sensor.

•Ability to measure 3D structures

Disadvantage of LiDAR

•Higher cost as compared to Ultrasound and IR sensor.

2.2.3.2 TOF10120 sensor

For distance measurement, an Ultrasound sensor (HC-SR04) was available as standard with the PiCar. But there were challenges in measuring distance of non -flat surface objects. The primary objective of this thesis is to detect and avoid obstacles. The obstacles can have different shapes and surfaces. So, I decided to experiment with a Laser sensor Model TOF10120. It uses vertical cavity surface emitting laser(VCSEL). VCSELs are semiconductor lasers, more specifically laser diodes with a monolithic laser resonator, where the emitted light leaves the device in a direction perpendicular to the chip surface [28].

2.2.4 Comparison between Ultrasound sensor (HC-SR04) and laser sensor (TOF10120)

The following table gives comparison between Ultrasound sensor(HC-SR04) and laser sensor(TOF 10120) used in this thesis.

Table 1: Comparison between Ultrasound sensor(HC-SR04)and laser sensor (TOF 10120)

Ultrasound sensor (HC -SR04)	Laser sensor (TOF10120)
Exposed to the environment	Sealed to the environment
Low cost (USD1)	High lost (USD10)
Connection : digital IO	Connection : I2C, serial
Relatively large	Relatively small
Wider field of View	Narrower field of view
Slow response	Quick response

2.3 Theory about Obstacle avoidance

To avoid obstacles in path of the vehicle, first the obstacles have to be detected. Then distance to the obstacles are measured. If an obstacle is at close distance to the vehicle, action plan to avoid the obstacle is needed. In this thesis, obstacles in the path of PiCar are detected using object detection model. Distance of the obstacle is measured using sensors. If it is within pre-defined close distance to PiCar, then PiCar is deviated from the obstacle. PiCar can be steered to left or right direction to avoid the obstacle.

3 Literature survey

Literature regarding ASV, object detection, distance measurement with different sensors and obstacle avoidance is given in following sections.

3.1 Literature on ASV

Literature about basic construction of ASV developed by a master student at UiO, description of applied sensors and path planning are found in the following work.

Development of a full-size low price Automatic Survey Vessel (ASV) for hydroacoustic work- This master thesis by Bendik S. Søvegjarto [3] presents a pilot for the ASV project at Physics Department, UiO. In this thesis, a motor control unit is designed, which can control two Electric outboard engines. This setup can be fitted to any boat and thereby turn the boat into an autonomous survey vessel. The control unit in this project includes a motor control system, communication equipment and a simple autopilot. The autopilot uses a Global Navigation Satellite System (GNSS)-receiver and a self-developed compass for navigation.

A Radio Controlled transmitter and Receiver are used to communicate from shore to the Vehicle. The steering command received by the receiver is processed by the control unit to steer the engines. Lead acid battery is used to power the engines.

Object Detection program used in my thesis can be used to enhance the features of such a Vessel. Raspberry Pi can be powered from a battery pack such as given in [56]. Python program for object detection can be executed and Raspberry Pi can be programmed to give signal out from GPIO Pins if an object is detected.

Further, the object distance also can also be measured and in case the object is closer at a predefined distance, the Raspberry Pi can be programmed to give signal out from GPIO Pins to the motor control unit.

The work by Bendik S. Søvegjarto includes a custom designed Attitude and Heading Reference System. The new Arduino nano BLE 33 Sense board has an IMU unit, which has 3 axis Gyroscope, 3 axis Accelerometer, 3 axis Magnetometer. This board can be further explored along with any Global Positioning System (GPS) to develop an autopilot system.

The Master Thesis by Halvard Yri Adriaenssens - **Automatisk utsetting og innhenting av sensorer i innsjø** [4] describes the development of a crane system to lower the Hydro-acoustic measurement probe in the lake and lift it up after the survey. The depth of the measurement probe is measured and the direction of cord holding the measurement probe is estimated using a camera system. This information can be given as input to the autopilot system so that the Vessel can be adjusted according to the probe position.

An article at [5] explains building of an ASV with sensors for bathymetry and oceanic currents, which follows predefined GPS waypoints and avoids obstacles. Open-source hardware PiHawk, open -source software Mission Planner and LiDAR are used in this project.

The paper at [43] provides an analysis of the surface target detection in the water environment using automotive radar. This can be used for tracking and anti-collision systems for autonomous surface vehicles (ASV).

[14] compares different sensors used in marine environment like radar, Camera and LiDAR and presents their advantages and disadvantages. Different path planning approaches classified as global and local path planners are analysed along with obstacle avoidance. Application of Artificial Neural Network for path planning is discussed. Shortfalls of path planning approaches which do not include the vessel dynamics, weather and sea conditions are discussed.

A data fusion model to detect road boundary using LiDAR and camera is presented in [15]. The likelihood of road boundary detection from Camera only model and LiDAR only model is combined for better likelihood results.

3.2 Literature about Object detection

A vision based surveillance system using Picamera and Raspberry Pi was chosen in this thesis. The system detects objects using computer vision techniques. Literature about the basics of object detection, different vision techniques, assessment of different image sensors, object detection with Raspberry Pi platform are given below.

Object detection uses Machine learning for image analysis. Since this field is continuously evolving, lot of on-line materials were referred.

A method of Object detection with Raspberry Pi is given at [16].

Basics of Object Detection, Single Stage Object Detection (SSD), datasets, loss function, bounding boxes, training a model, evaluating a model is explained in at [19].

Comparison between MobilNetV1 and MobileNetV2 is given in this googleblog webpage [17]

The method of object detection using pre-trained SSD-MobileNetV2-COCO TensorFlow model given in [20], is referenced for my thesis. Tensorflow1 is used in this.

The repository in [21] is referenced for object detection with custom a dataset using Tensorflow2.

3.3 Literature about Image sensors

Machine vision (MV) cameras were also considered at initial stage of the thesis. Then more general Computer vision technique and camera was finalised for the thesis. The differnce between these two are given below:

Difference between Computer Vision and Machine Vision

Computer vision uses PC-based processor for deeper analysis of the image as it has a much greater processing capability of acquired visual data when compared to machine vision.

When such advanced capabilities of computer vision is not needed, machine vision is used. For most of such applications, processing capabilities of a simple PLC-based system is sufficient. Machine vision is mostly used to quickly analyse image data to facilitate simple automated choices like presence / absence of an item, whether an item is good/bad,defective/not defective.

Details of Machine Vision cameras such as Open MV Cam M7, Open MV Cam H7, OpenMV Cam H7 with FLIR Lepton 3.5 are given in [22]. Since our application mainly focuses on object detection including inanimate objects and as these objects mostly will not be thermally visible, **thermal vision cameras** are not considered.

Night vision cameras can be considered for our application and details of Raspberry Pi night vision camera is given in [23]. This camera has IR LED that allow the camera to see in dark without producing visible light. If the ASV will be used for night surveys, this type of camera can be used.

Camera Model Pi NoIR provides the same functionality as the regular Camera Model V2, except that it does not employ an infrared filter[24].

The main difference between IR camera and Thermal Camera is that thermal imagers operate in longer infrared wavelength regions than IR. So thermal imagers are not affected by oncoming headlights, smoke, haze, dust, etc.[27]

Raspberry Pi Camera V2

Raspberry Pi Camera V2 Module is fitted on PiCar and is used for this master thesis. It has a Sony IMX219 8-megapixel sensor. It supports video modes as well as still capture. It attaches via a 15cm ribbon cable to the Camera Serial Interface (CSI) port on the Raspberry Pi. Detailed specifications are given at 4.1.4

Figure 26: Raspberry Pi -camera V2. Image reference [25]

3.4 Literature about distance measurement sensors

After identifying the obstacle, the distance to the obstacle is measured so that the PiCar can avoid the object. For this, distance measuring sensors are used. Ultrasound sensor(HC-SR04) was present in PiCar. Also, to measure distance of objects with non-flat surfaces, a Laser sensor (TOF 10120) was chosen and experimented. Literature about other type of sensors and references to distance measurement techniques are given below:

Ultrasound sensor HC-SR04 (present in Adeept PiCar B): The detailed specification of this sensor is given at 4.1.3 and comparison between Ultrasound sensor and Laser sensor is given in section 2.2.4

Multiple Ultrasound sensors can be connected to find distance in different directions. [29] demonstrates a way to connect multiple Ultrasound sensors to Raspberry Pi. To avoid crosstalk between signals of different sensors, a time delay can be set between triggers of different sensors. Otherwise, next trigger can be sent once a signal is received from the first sensor.

LiDAR: LiDAR stands for Light Detection and Ranging. It sends laser light towards the target and by measuring the time for the reflected light to return to the receiver, object distance is calculated. Some models of LiDAR sensors are LIDAR Lite V3, LIDAR TFMini-Micro LiDAR Module. LiDAR is mostly used for 3-D representation of the surrounding environment.

Laser sensor TOF 10120: Laser sensor TOF 10120 sensor was chosen for object distance measurement in the thesis. Working principle of the sensor is given in section 2.2.3.2 and comparison with Ultrasound sensor is given in section 2.2.4. Technical Specification of TOF10120 sensor is given at 4.2.3

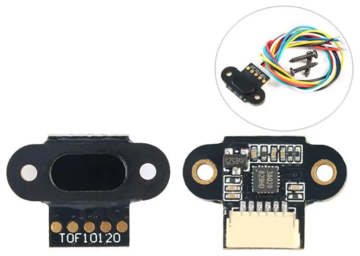


Figure 27: TOF 10120 sensor. Image reference [26]

A general literature covering three type of sensors (Camera, Radar and LiDAR) for autonomous vehicles is given at [30]

3.5 Literature about Distance Measurement

Distance measurement using Arduino Uno and Ultrasound sensor and deviating the car away from the obstacle is given in arduino projects- [31] and [32].

An autonomous vehicle made using Arduino Uno, Magnetic Compass, GPS Receiver, DC Motor is given the project [36]. GPS receiver finds the current location of the vehicle. The destination coordinates are entered manually and the vehicle moves to this location with motor. Magnetic compass gives the heading direction. Obstacle distance is given by ultrasound sensor.

An application for distance measurement with image from camera and circle marker on target is presented in [33]. A method of measuring object distance by detecting a square edge marker on the target is given in [34]. A method for measurement of object distance from image is described in [35].

3.6 Literature about Obstacle Avoidance

PiCar has servomotor which controls the front wheels. These wheels need to be steered to move the PiCar away from obstacle. Methods for obstacle avoidance using sensors with Arduino or Raspberry Pi like [37] were referred. Adeept PiCar -B v1.0 documentation at [38] gives PiCar part details and servomotor control detail. PiCar manual version 1 released in 2018 was used during this thesis. A higher and detailed version of manual is available now.

3.6.1 Mobile Robotics

Mobile Robotics has many interesting solutions and applications for obstacle avoidance and navigation. For example, consider a Robotics system such as Model Turtlebot3 burger [50]. It is fitted with Laser Displacement Sensor (360° LIDAR), Inertial Measurement Unit(IMU) and uses Robotics Operating System(ROS) [51]. ROS libraries have pre-built functions which can be readily used to estimate the current position of the robot. In this system, estimation of the current position of the Robot, planning a Robot path and navigation can be carried out with Simultaneous Localisation And Mapping (SLAM) technique(section 4,5 of [52].

Another example of autonomous navigation using mobile robot with SLAM algorithm is presented in [53]. SLAM technique is used for mapping the environment of the robot and estimating the position of the robot.

Use of mobile robot -Turtlebot3 (burger) and SLAM technique for autonomous medical application is given in [54]. Here, the local environment of the robot is mapped and the robot is successfully navigated to reach a patient and deliver medicines. Also Facial recognition is used to identify the correct patient.

Turtlebot3 – Waffle Pi model includes a Raspberry Pi camera (section 2.2-[52]). This can be used for vision applications. Cost comparison of Turtlebot3 with Adeept PiCar is presented here to give an overview [price details as on 14.January. 21] :
Price of Turtlebot 3 : USD 1399
Price of Adeept PiCar + LiDAR : USD82.49 +USD179.9

3.7 Interface between Arduino and Raspberry Pi boards

Laser sensor(TOF10120) was connected to Arduino Uno board for distance measurement. An Arduino Nano board was used to measure temperature and relative humidity. These Arduino boards were connected to Raspberry Pi using USB interface. Interface between Arduino and Rapsberry Pi is given at [39].

4 Materials and Methods

As shown in figure 1, in this thesis we are designing a surveillance system for ASV. With this system, we are trying to identify the obstacle in the movement direction of PiCar, measure the obstacle distance and avoid obstacle when it is too close to PiCar. The hardware materials are listed below:

Adeept Mars Rover PiCar-B .

It has a base for four wheels and different sensors as shown in the figures 28,29, 30 and 31.

PiCar consists of the following:
Plastic Base with four wheels
Raspberry PI 3 model B+ : 1 no.
PiCamera model V2: 1 no.
Ultrasound sensor (HC-SR04): 1 no.
DC motor: 1 no.
Servomotors: 3 no.s
(1 servo motor for front wheels steering.
1 servo motor for up and down movement of Picamera and Ultrasound sensor.
1 servo motor for left and right movement of Picamera and Ultrasound sensor.)
Adeept interface board - Adeept Motor HAT for Raspberry Pi: 1 no.
Power Adapter for Raspberry Pi3 Model B+: 1 no.

The following additional hardware and sensors were added to PiCar to enhance the functionality of PiCar for my thesis.
Arduino Uno: 1 no.
Laser Sensor (TOF 10120): 1 no.
Interconnecting cables for TOF10120 to Arduino Uno.
Arduino Nano 33 BLE Sense: 1 no.
USB interface cable between Raspberry PI and Arduino Uno: 1 no
USB interface cable between Raspberry PI and Arduino Nano: 1 no

Common Material and software:
Laptop (Make: ASUS): 1 no
Software: Python, Arduino Sketch

The following pictures shows the hardware and sensor details:

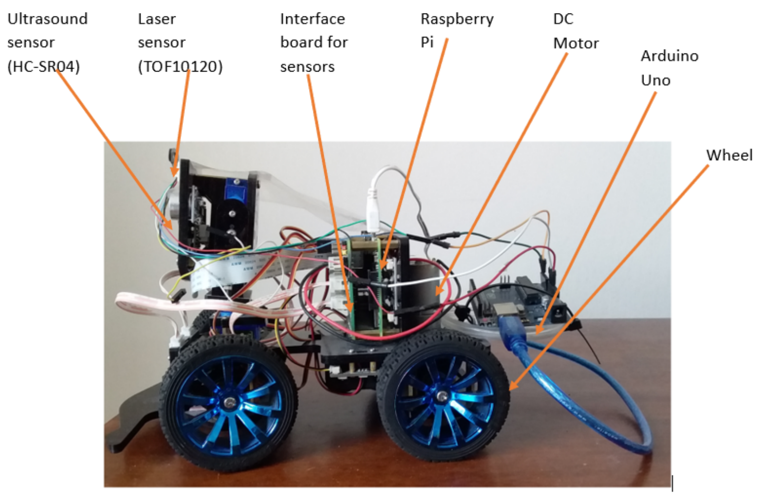


Figure 28: Adeept PiCar - side view showing different hardware details. Laser sensor and Arduino Uno are not standard part of PiCar. Image reference [65].

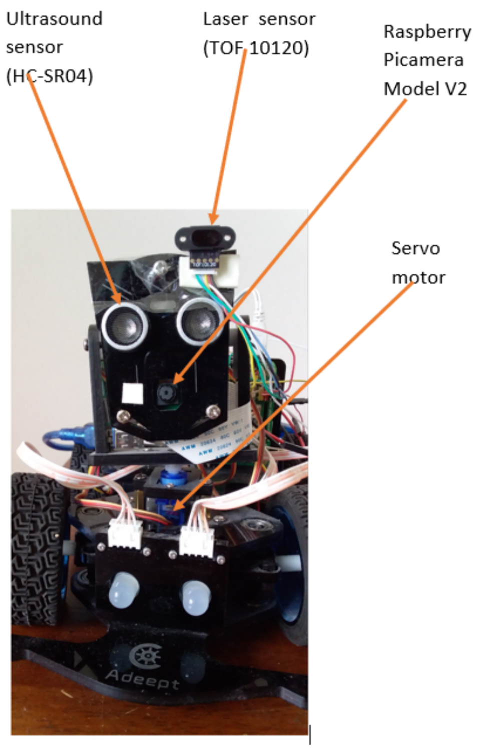


Figure 29: Adeept PiCar - Front view showing different hardware details. Laser sensor is not a standard part of PiCar. Image reference [65].

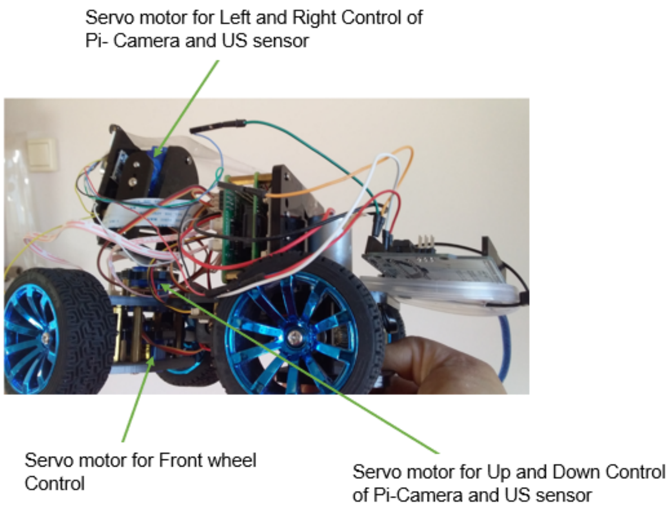

Servo motor for Left and Right Control of Pi- Camera and US sensor
Servo motor for Front wheel Control
Servo motor for Up and Down Control of Pi-Camera and US sensor

Figure 30: Adeept PiCar - Servomotor details. Image reference [65].

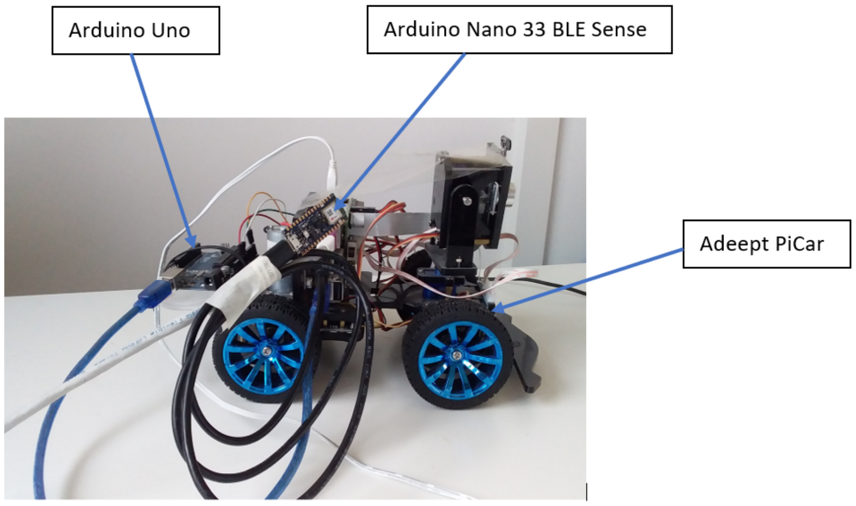


Figure 31: Adeept PiCar with Arduino Uno and Arduino Nano 33 BLE sense. Image reference [65].

4.1 Hardware and Sensors with Adeept PiCar B

4.1.1 Raspberry Pi 3 Model B+

Raspberry Pi is a small single board computer. It has ports to standard input devices as mouse, keyboard and ports to connect to display. It can be programmed for different applications. The detailed specifications are given at [57].

4.1.2 Adeept interface board - Adeept Motor HAT for Raspberry Pi

This is an interface board by Adeept for Raspberry Pi. It integrates a DC motor driver, a 16-H PWM servo controller and a DC-DC buck chip. This can be used for making Raspberry Pi based robots and smart cars.

4.1.3 Ultrasound sensor HC-SR04 specification

Range: 2 cm to 4 m
Power Supply: +5V DC
Effectual Angle: <15°
Resolution : 0.3 cm
Ultrasound Frequency: 40 kHz
Interface: digital IO

Table 2: Connection details -Ultrasound sensor(HC-SR04) and Raspberry Pi

Ultrasound sensor (HC -SR04)Pin	Raspberry Pi3 Model B+Pin
VCC	+5V
GND	GND
TRIG	GPIO Output
ECHO	GPIO Input

4.1.4 Raspberry PI camera module V2 specification:

Still Resolution : 8 MP
Video modes: 1080p30, 720p60 and 640×480 p60/90
Sensor: Sony IMX219
Focal length : 3.04 mm
Detailed specification is given at [60]

4.1.5 Servomotor -TowerPro SG90

Operating Voltage : +5V typical
Torque: 2.5kg/cm
Operating speed is 0.1s/60°
Gear Type: Plastic
Rotation : 0°-180° (90° in one direction)

Table 3: Connection details of Servo Motor - TowerPro SG90

Wire Colour	Description
Brown	Ground wire
Red	+5V
Orange	PWM signal

4.2 Additional Hardware and Sensors that I added in the PiCar

4.2.1 Arduino Uno

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. It has digital input/output pins, analog input pins, an USB connection, a power barrel jack and a reset button. It can be programmed for different applications to read input signals or to control outputs [40].

As explained earlier at Chapter 1, the Laser sensor TOF10120 was used for object distance measurement. This sensor was connected to Arduino Uno board which in turn was connected to Raspberry Pi through USB interface. The reason for not connecting the Laser sensor directly to Raspberry Pi is given at section 9.4

4.2.2 Arduino Nano 33 BLE sense

This board has sensors like IMU (Inertial measurement Unit -Accelerometer, Gyroscope, Magnetometer), pressure, temperature, humidity, proximity sensors among others. It has Bluetooth Low energy capability for wireless communication [41].

Temperature and relative humidity readings from the sensor on this board are used for more accurate measurement of object distance form Ultrasound sensor. This board is connected to Raspberry Pi through USB interface.

4.2.3 Laser sensor TOF10120

Laser sensor TOF10120 was selected for experiment of distance measurement on objects with non-flat surfaces.

Specifications:
940nm laser-Class 1 operating conditions -IEC 60825-1: 2014
Range: 100 cm to 1.8m
Working voltage: 3.5 V
Maximum accuracy: within 5% indoor.
Interface: Serial / I2C

In this thesis, the Laser sensor TOF 10120 was connected to Arduino Uno board. It is possible to connect the Laser sensor to Arduino Nano board as well, but this is not done as the Arduino Nano board was available to me only at later stage of the thesis. The connection details are as given below.

Table 4: Connection details -Laser sensor(TOF 10120) with Arduino Uno and Arduino Nano 33 BLE Sense

Laser Sensor (TOF 10120)-Pin	Arduino Uno -Pin	Arduino Nano 33 BLE Sense -Pin
GND	GND	GND
VDD (3 to 5 V)	5V Output	5V Output
RXD (Serial Receive)	Not connected	Not connected
TXD (Serial Transmit)	Not connected	Not connected
SDA (I2C data)	Analog In -A4(or PC4-SDA pin)	Analog In -A4
SCL (I2c clock))	Analog In -A5 (or PC5 -SCL pin)	Analog In -A5

4.3 Details of the methods

The main task for this thesis is to design a simple and cost effective surveillance system for an autonomous survey vessel. Such a system should be able to detect obstacles in front of the boat, estimate the distance of the obstacles and provide information so that the autopilot can take actions to avoid these. The major steps of the thesis can be listed as following and they are explained in the sections as mentioned below :

Setting up the PiCar (section 4.4).
Capturing images from the camera module(Chapter 5).
Identifying the objects from the images and defining obstacles (Chapter 5).
Finding distance of the obstacle (Chapter 6).
Deviating from the obstacle (Chapter 7).

4.4 Setting up the PiCar

Adeept PiCar is built on Raspberry Pi which operates on Raspbian Operating System (Raspian OS). Raspian OS is loaded on SD card and inserted in Raspberry Pi. Raspberry Pi boots from the SD card. Raspberry Pi is powered by power cable. With this setup, the PiCar movement is restricted by the length of power cable to Raspberry Pi. To eliminate the need of power cable, a suggested battery model UM 18650X2 was used to run the PiCar for more free movement. With continuous motor operation, the charge of the battery lasted only for 20 to 30 minutes. So, the PiCar was powered from power cable further in my experiments.

The standard setup and operating procedure for PiCar is given below [38].

The Picar is connected to Wi-Fi network. In the Laptop or Controller, Server python program is loaded which works through a Graphical User Interface (GUI). In Raspberry Pi of PiCar, Client python program is loaded. The PiCar can be controlled through data interaction between Server and Client Programs.

At the UiO campus, connecting PiCar to UiO -Guest network was not possible due to connectivity rules. Also in practical surveying environment at lake, it may not be possible to have a Wi-Fi connection. Hence, I decided to avoid Wi-Fi connection and used a network cable to connect the Laptop to PiCar over Secured Shell Protocol (SSH). With this arrangement, the movement of PiCar is restricted with the length of network cable, and there is still need of SSH connection. To avoid this, It is desirable that the PiCar starts working without any interface as soon as it is powered on. An executable program at PiCar boot up can start the PiCar, carry out object detection, obstacle distance measurement, obstacle avoidance till aborted.

A Python program was developed for the surveillance system to carry out object detection, obstacle distance measurement and obstacle avoidance. It was possible to make an executable file for the Python program and run from Raspberry Pi command prompt. when I tried to make this executable program run at boot up of Raspberry Pi, there were challenges and errors. These errors were mostly from loading of object detection model. These errors can be addressed in future work. For the rest of the thesis, the Python program was run from Thonny Python Integrated Development Environment (IDE).

5 Method of Object Detection

The objects in the path of the PiCar are identified by object detection models using machine learning. The images are continuously captured by PiCamera of PiCar through video capture. The object detection model identifies the objects in these images and produces output with object class name and location of the object in the image.

5.1 Method of Object detection with pre-trained model

First, a pre-trained model SSDLite-MobileNetV2-COCO from TensorFlow model zoo is used for detecting objects from the image (section 2.1.13). This model is trained on COCO dataset. Object detection with PiCamera is explained in [20].

Pre- trained model means that the object detection model is trained to detect certain objects. SSDLite-MobileNetV2-COCO is trained to detect 90 objects like "orange", "banana", " book", "tree", "person", "cup " etc.

The PiCamera continuously captures the images and the image captured from Pi-camera is given as input to the object detection model. This model predicts the class of objects identified in the image according to the theory discussed in section 2.1.

I started this part of the project work around January 2020 and software versions of TensorFlow, OpenCV, Python were chosen according to PiCar manual version1 [38] and loaded in Raspberry PI:

Python version : 3.7.3

TensorFlow version : 1.14.0

OpenCV : 4.1.0

The pre-trained model SSDLite-MobileNetV2-COCO20180509 from TensorFlow1 Object Detection Model Zoo [12] was loaded on Raspberry Pi. This model was chosen considering the processing capability of Raspberry Pi3 model B+ for object detection.

Algorithm for object detection with pre-trained model is shown figure 32.

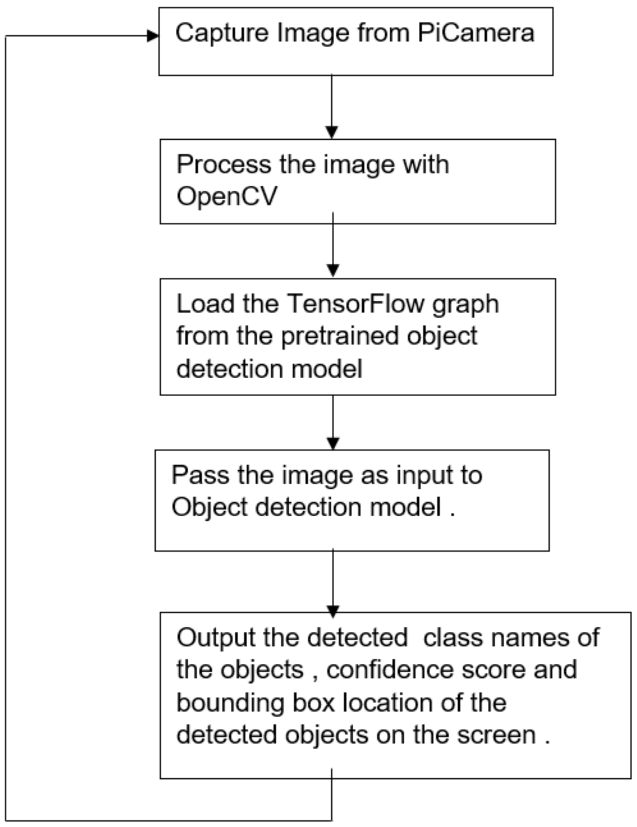

Capture Image from PiCamera
Process the image with OpenCV
Load the TensorFlow graph from the pretrained object detection model
Pass the image as input to Object detection model .
Output the detected class names of the objects , confidence score and bounding box location of the detected objects on the screen .

Figure 32: Algorithm for object detection using a pre -trained object detection model.

A Python program captures an image from the PiCamera and passes it to the object detection model SSDLite-MobileNetV2-COCO. This model analyses the image and identifies the objects in the image. It produces output such as the identified class of the object (class to which the object belongs, for example Car, TV, etc), confidence score of the object class, and the bounding boxes locating the objects. This whole process continues till the program executes. The results of object detection with pre-trained model is given in section 8.1.1.

5.2 Method of Object detection with pre-trained model using custom dataset

As discussed in section 5.1 and through results in section 8.1.1, the pretrained model gave satisfactorily results. The model is trained to detect 90 class of objects which are given under Labels [42]. Some of the classes are “cat”, “umbrella”, “snowboard”, “cup”,etc.

As the ASV will be used in the water environment, the objects more common in such environments like tree branch, thick logs, ducks, swans, lifebouys need to detected. Also, my thesis started by studying [43], which explores the use of an automotive 3D radar in ASV for obstacle detection in the water environment such as boat and floating objects. Figure 5 at [43] shows some of the common floating objects like airtoys, lifebuoys, fenders etc. Section 6 of the reference mentions that the system detected many small objects, but object like fenders were not detected. Also objects that are air inflated, such as fenders or air toys, show worse detectability than solid targets, such as lifebuoys.

This was my source of interest and curiosity to experiment whether these kind of common objects in the water environment can be detected using Machine learning in image processing.

5.2.1 Training CNN object detection model with custom dataset

As I was unaware of any dataset catering to objects in the water environment, I decided to collect images to make a small dataset and train the model with custom dataset.

I choose a "buoy" as shown below as the object to be detected.

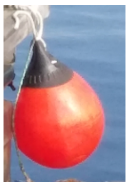


Figure 33: image of Buoy. Image reference [65].

Algorithm of the process is given in figure 34.

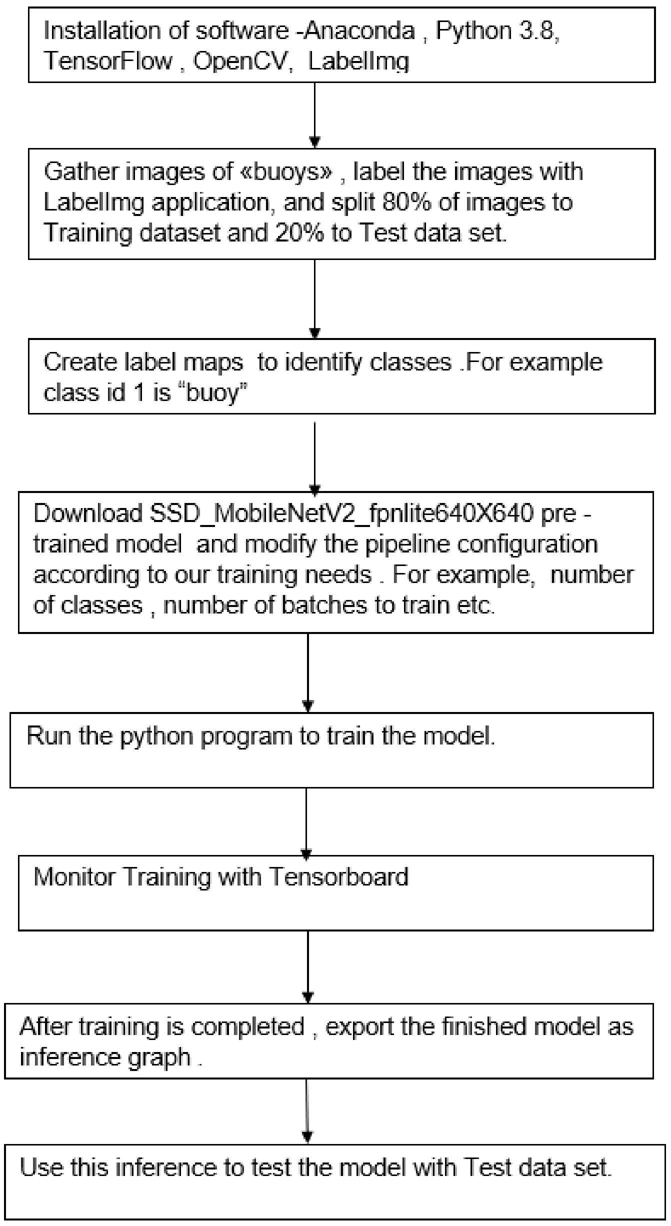

Installation of software -Anaconda , Python 3.8, TensorFlow , OpenCV, LabelImg
Gather images of «buoys» , label the images with LabelImg application, and split 80% of images to Training dataset and 20% to Test data set.
Create label maps to identify classes .For example class id 1 is "buoy"
Download SSD_MobileNetV2_fpnlite640X640 pre - trained model and modify the pipeline configuration according to our training needs . For example, number of classes , number of batches to train etc.
Run the python program to train the model.
Monitor Training with Tensorboard
After training is completed , export the finished model as inference graph .
Use this inference to test the model with Test data set.

Figure 34: Algorithm of training CNN object detection model with custom data set.

This activity of making a custom dataset of buoy started during January 2021. Because of winter season, collection of buoy images was challenging and hence, I had to refer some images of buoy from Internet. At a later stage, some buoy images were collected from nearby boat parking area at Horten, Norway and these were used for testing the model.

For the object detection model to perform better, it requires generally hundreds of images with different backgrounds. I was able to collect 36 images of buoy with different backgrounds. Even though I was sceptical of the model training and performance with this small custom dataset, I proceeded with the dataset. Dataset images need to be labelled to create the label map for the model. The procedure at [21] explains training the model with custom dataset. I planned to do the labelling and training of the model in Raspberry Pi itself. When installing labelling tool on Raspberry Pi, there were errors which could not be resolved.

So, I wanted to follow another approach of training the model on my personal laptop and then exporting the trained inference graph to Raspberry Pi and use it for detection of buoys. Machine learning in image processing is a fast and continuously evolving field. From December 2019 till January 2021, some of the software came out with newer versions. Especially, TensorFlow software had almost 49 different versions between 19th June 2019 and 21st January 2021 [44]. The major modification being TensorFlow2.

As the first step, I planned to train the model on my Laptop. My Laptop (Model:ASUS) has a Graphical Processing Unit(GPU) of model NVIDIA GEFORCE and training a model with GPU speeds up the training process as it uses multiprocessing. But I decided not to use the GPU and only use the Central Processing Unit (CPU) for the training process to assess the training time. Hardware and main software details are given below.:

Laptop: Model-Asus
OS : windows 10
Python version : 3.8
TensorFlow version: 2.4.1
LabelImg-labelling application
Anaconda virtual environment

Other supporting software tools were also loaded in the laptop. The collected images of buoy are labelled with the labelImg application. Image sizes are reduced and the images (type PNG) are opened one by one in the LabelImg application. The portion of the image containing buoy is selected and labelled. The LabelImg application generates xml files from the images.

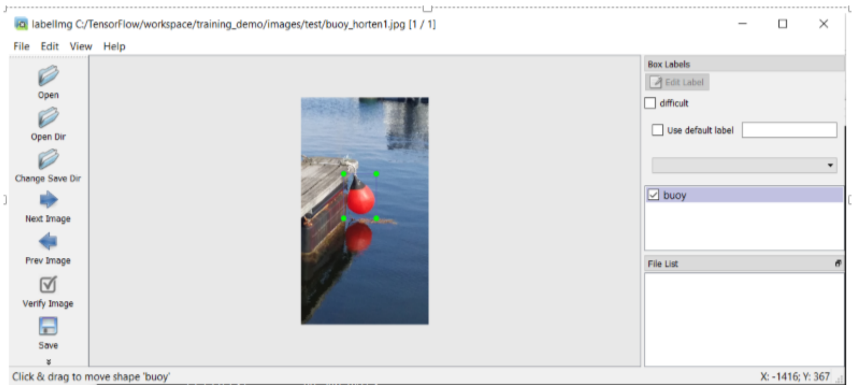
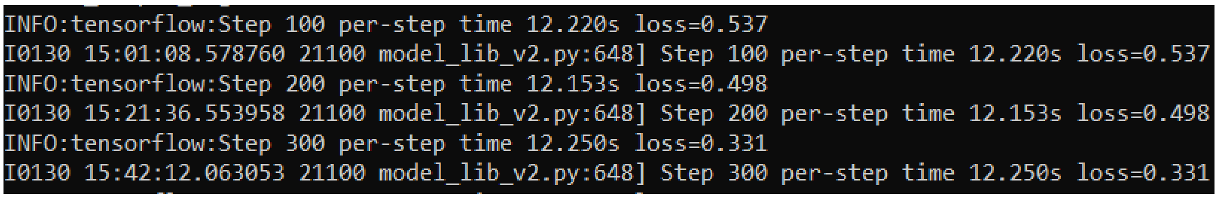


Figure 35: labelling Image with the LabelImg application – labelling single buoy. Image reference [65].

All the images were labelled in this manner and among the dataset of 36 images, 28 images (approximately 80 %) were taken as Training data set and 8 images (approximately 20%) as Test data set.

Label mapping file was created. As this model was trained as a single class object detection model, only one id, id 1 was used to denote the class " buoy". From Google's TensorFlow Model zoo for Tensorflow2 [45], SSD-MobileNet-V2-FPNLite-640x640 was selected among listed models of SSD MobileNet. The reason for selecting the SSD MobileNet model was discussed in section 2.1.13. FPN (Feature Pyramid Network) is a network type which outputs feature map of different resolutions and 640X640 is input resolution size.

The model was trained with the training dataset with a batch size of six images. One training step is one gradient update after processing the batch of six images. Example of process window was as shown below:



```
INFO:tensorflow:Step 100 per-step time 12.220s loss=0.537
I0130 15:01:08.578760 21100 model_lib_v2.py:648] Step 100 per-step time 12.220s loss=0.537
INFO:tensorflow:Step 200 per-step time 12.153s loss=0.498
I0130 15:21:36.553958 21100 model_lib_v2.py:648] Step 200 per-step time 12.153s loss=0.498
INFO:tensorflow:Step 300 per-step time 12.250s loss=0.331
I0130 15:42:12.063053 21100 model_lib_v2.py:648] Step 300 per-step time 12.250s loss=0.331
```

Figure 36: Example of process window.

The above figure shows the process window with details of the training. It shows the time taken for training each step and loss value calculated at each step.

The total loss value shows the accuracy of the training. The training was stopped at step number 1900, when the total loss value fell below 0.2, which I consider to be a reasonable accuracy. The training process took approximately six hours and it was continuously monitored with TensorBoard. TensorBoard is a visualisation tool for understanding and optimising the model.

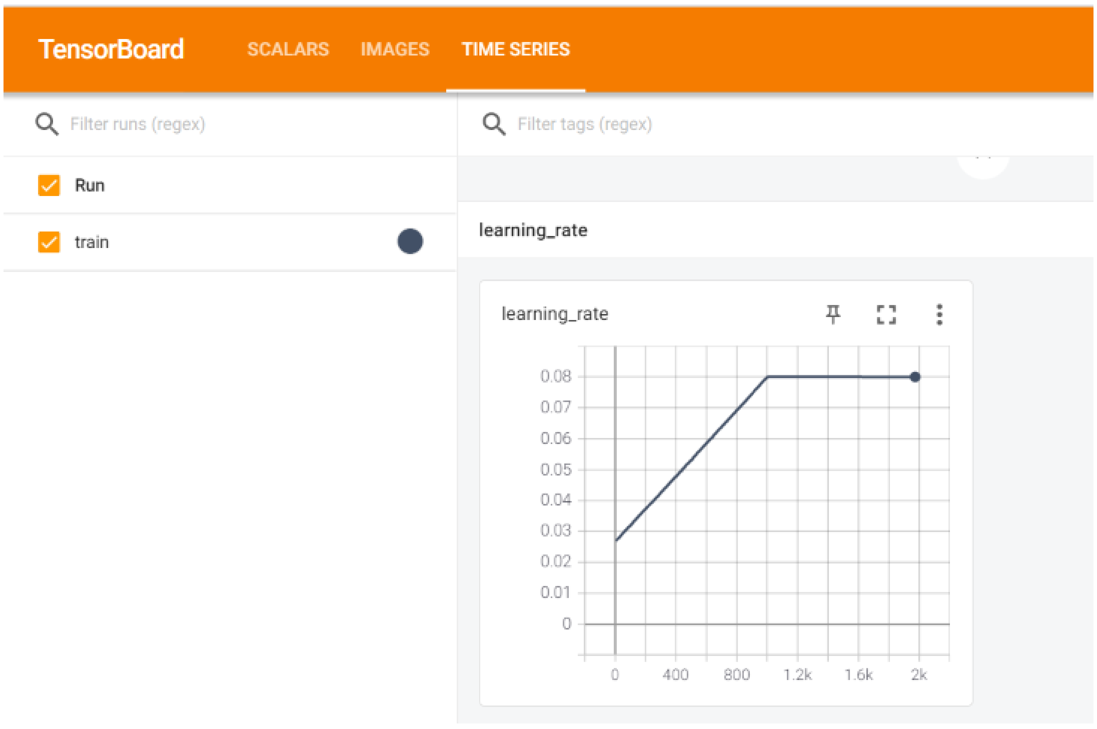


Figure 37: TensorBoard visualisation of model hyper parameter learning rate at training step 1900. X-axis is training step number. Y-axis is learning rate.

Learning rate is one of the hyper parameters in training a model. Figure 37 shows the Learning rate as a function of iteration steps. Here we can see that the learning rate starts around 0.03, gradually increases during the training and then stabilises at 0.08 as the training matures.

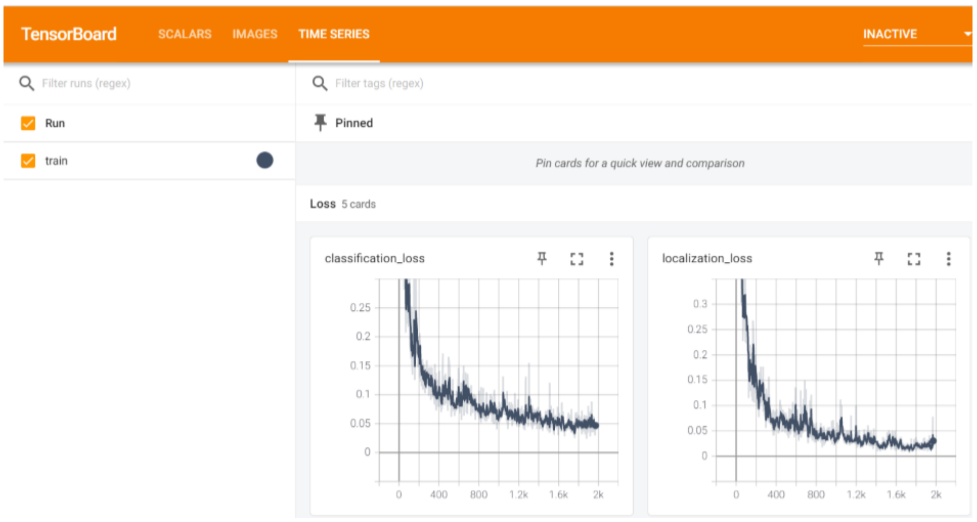
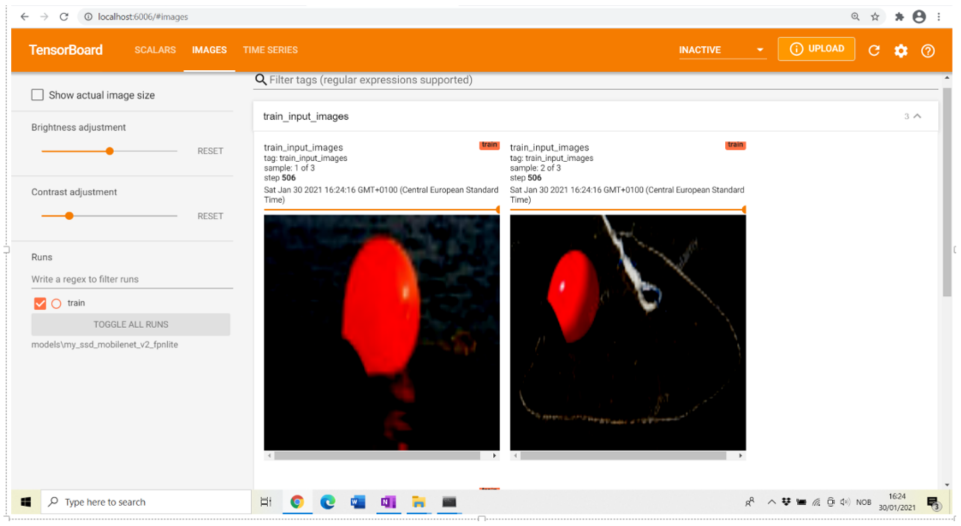


Figure 38: TensorBoard visualisation of Classification and Localisation loss at training step 1900. X-axis is training step number. Y-axis in leftside graph is classification loss, rightside graph is localisation loss.

As explained earlier, object detection involves classification of the image under the ground truth labels and localising the objects in the image. Figure 38 shows classification and localisation loss as function of training steps.



train_input_images
tag: train_input_images
sample: 1 of 3
step 506
Sat Jan 30 2021 16:24:16 GMT+0100 (Central European Standard Time)

train_input_images
tag: train_input_images
sample: 2 of 3
step 506
Sat Jan 30 2021 16:24:16 GMT+0100 (Central European Standard Time)

Figure 39: TensorBoard visualisation – training of some images at step 506.

Figure 39 shows a step in the training process with the input images.

5.2.2 Testing trained model with images

After training the model with reasonable accuracy, the model was tested. Images from test dataset were given as input to check if the model identified the object "buoy". Test results are given in section 8.1.2.

5.2.3 Testing trained model with live camera feed

In section 5.2.2, the model was tested by providing input image from test data set. The performance of the model had to be verified with live camera feed. The camera in the Laptop was used and the live image from camera was given to the model for real time object detection. As an actual buoy was not available with me to test the model, I used an orange balloon with black attachment to imitate a buoy. The results are given in section 8.1.3.

6 Method of Obstacle distance measurement

The standard setup of PiCar has an Ultrasound sensor. Distance measurement can be carried out with this sensor in accordance with its specification. Distance of any obstacle within the focus range of the ultrasound sensor can be measured. This can be done without the need of object detection from image. I have also experimented with Laser sensor TOF10120 for measuring distance of the obstacle.

Note: To measure the distance of the obstacle in front of the PiCar, the obstacle needs to be in the focus range of the Ultrasound sensor or Laser sensor.

The velocity of sound is taken at a constant value of 340 m/s without temperature and humidity correction at equation 2.2.1 for distance measurement using Ultrasound sensor for experiments 1 to 9 at table 6.

Warm up time of 15 to 20 minutes was given to sensors before taking measurement.

A number of experiments were conducted for object distance measurement and the results are presented under the sections as listed below. These experiments were carried out on different days during the thesis.

Table 5: List of Experiments done for Object Distance Measurement

Experiment	sensor	Target and test case	section
1	Ultrasound	Flat Surface, Black colour object	6.1.1
2	Laser	Flat Surface, Black colour object	6.2
3	Laser	Flat Surface, White colour object	6.2
4	Both	Curved surface object – Coffee Cup	6.3.1
5	Ultrasound	Curved surface object – candle holder(smooth side)	6.3.3
6	Ultrasound	Curved surface object – candle holder(rough side)	6.3.4
7	Both	Curved surface object – candle holder(rough side)	6.3.5
8	Both	Flat surface, White colour object – ambient light around 300 lux	6.3.6
9	Both	Flat surface, White colour object – ambient light at 0 lux	6.3.6
10	Ultrasound	Flat object-temperature and humidity correction of Ultrasound sensor data	6.3.7
11	Both	Flat object-Data fusion from Ultrasound and Laser sensor data	6.3.8

For all the above experiments, only measurement of distance to the object was carried out. Object detection was not carried out. Combined execution of Object detection and Object distance measurement is presented in section 8.3.

6.1 Method of distance measurement using Ultrasound sensor

The PiCar had a single Ultrasound sensor in standard configuration and initially it was used for distance measurement. The readings of the Ultrasound sensor were roughly checked with measuring tape for measurement of distance. Also different types of surfaces were used as targets for distance measurement. The set up and details are given in the following sections.

6.1.1 Measurement on flat surface

Test setup was made with Ultrasound sensor on PiCar, a test object with Flat surface, measurement tape, Python program for acquiring data from PiCar. The test object was moved manually in steps and Ultrasound sensor data was recorded.

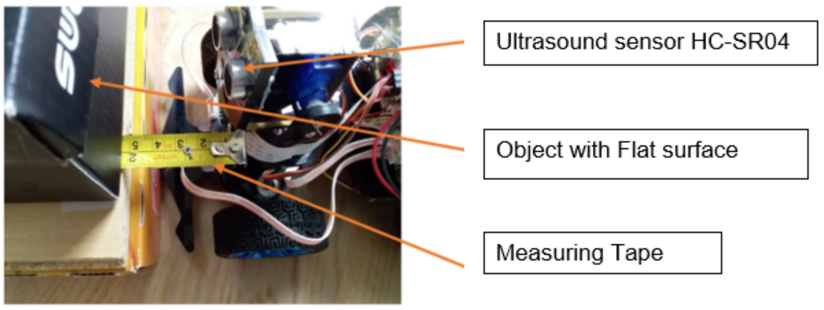


Figure 40: Test setup using Ultrasound sensor for distance measurement of target with flat surface. Image reference [65].

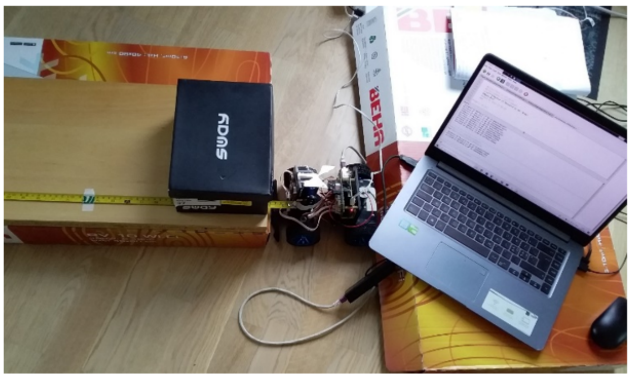


Figure 41: Test setup using Ultrasound sensor for distance measurement of flat surface-data acquisition and display with laptop.

Results of this experiment are given at section 8.2.1.

6.1.2 Measurement on curved object

This thesis involves identifying different target objects and measuring distances to these objects. To test the system on curved surfaces, I chose "Coffee Cup" as target. Also, the object detection model SSDLite-MobileNetV2-COCO is trained to detect "Coffee Cup". When measuring distance to this object, I observed that the ultrasound sensor readings varied significantly. Readings were not recorded due to significant variation.

So, I had to think of a better method to measure distance on a curved surface. One such alternate was to use a Laser sensor instead of Ultrasound sensor to measure distance and a cost effective laser sensor TOF 10120 was selected.

6.2 Method of distance measurement using Laser sensor

The Laser sensor (model TOF10120) was connected to the Arduino Uno board via I2C interface. The signal lines(SDA and SCL) and the power lines (VCC and GND) are connected to Arduino Uno as given in table 4. Arduino Uno is connected to Raspberry Pi of PiCar with USB interface. The Arduino sketch driver for TOF 10120 is referred from [61]. The test setup is shown at figures 42 and 43. Measurement results are given at section 8.2.2:

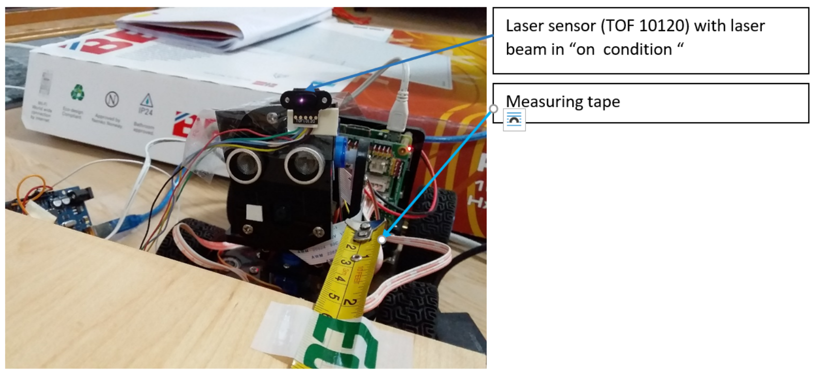


Figure 42: Test setup for distance measurement with laser sensor -front view. Image reference [65].

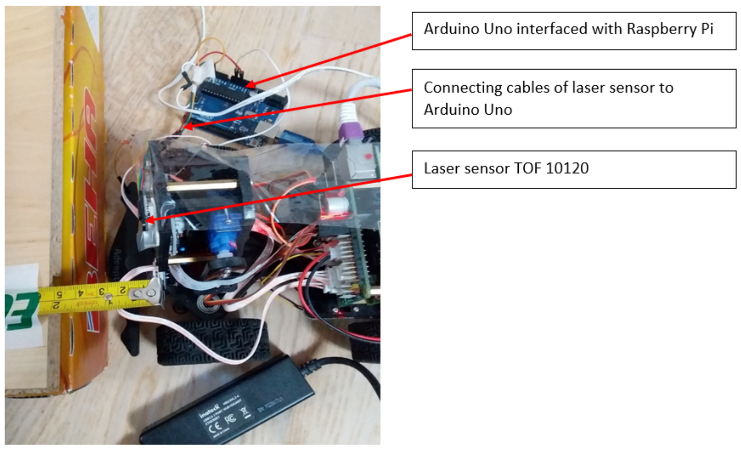

Arduino Uno interfaced with Raspberry Pi
Connecting cables of laser sensor to Arduino Uno
Laser sensor TOF 10120

Figure 43: Test setup for distance measurement with laser sensor -Top view.

6.3 Method of distance measurement with Ultrasound sensor and Laser sensor against various targets

The main purpose of choosing the Laser sensor was that Ultrasound sensor gave inconsistent measurement readings for curved surface target. To confirm this, simultaneous measurement with both Laser and Ultrasound sensor was needed on different targets. Experiments were carried out using coffee cup, candle holder and flat object. The details are given in the following sections.

6.3.1 Measurement with both sensors on object "Coffee Cup"

A test setup made with a curved surface object (a coffee cup) is shown below and the results of the measurements are given at 8.2.3.

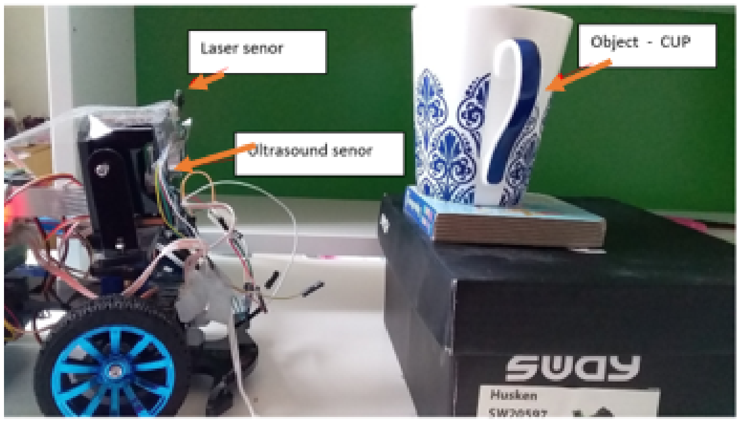


Figure 44: Test setup for distance measurement of curved object "coffee cup" – side view. Image reference [65].

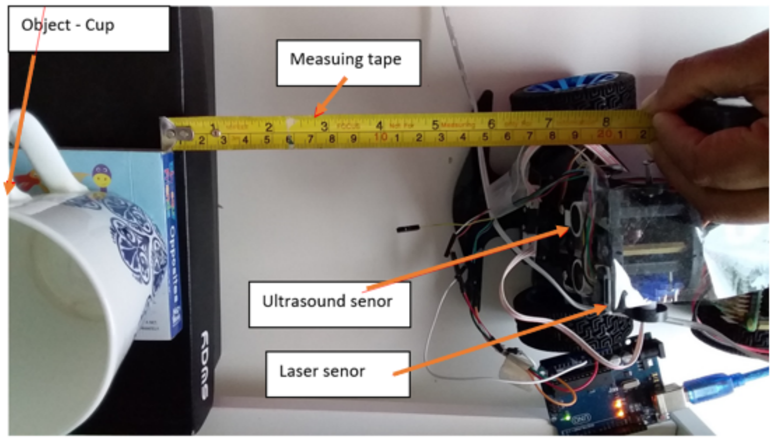


Figure 45: Test setup for distance measurement of curved object "coffee cup" – top view.

6.3.2 Measurement with Ultrasound sensor on object "candle holder"

I observed that the Ultrasound sensor readings were inconsistent for certain measurements with coffee cup as given in figures 59 and 61 of section 8.2.3. To explore the behaviour of the Ultrasound sensor on curved surface further, another curved object "candle holder" was chosen. Distance measurement with Ultrasound sensor was done on the surface of test object "candle holder". I would like to refer this original surface as "smooth side" of candle holder. The reason is that at later stage, the surface was roughened to check the effect of surface roughness on Ultrasound measurement. This roughened surface of candle holder was termed as "rough side".

Test set up with "smooth side " and "rough side" of the candle holder is shown in sections 6.3.3 and 6.3.4.

6.3.3 Measurement with Ultrasound sensor on "candle holder"-smooth side

The test setup made with Smooth side of the candle holder is shown below.

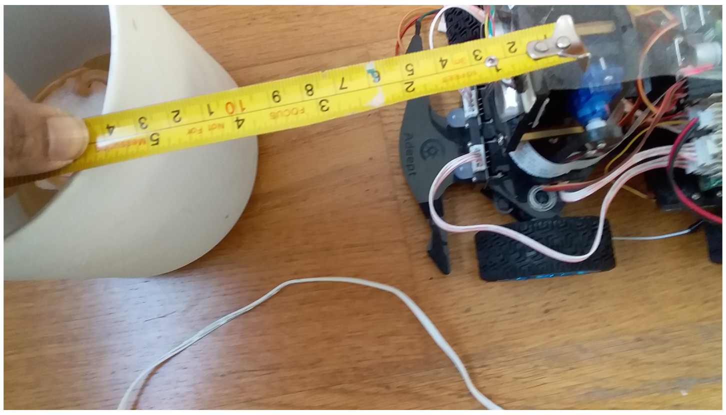


Figure 46: Distance measurement with Ultrasound sensor - test object "Candle holder" - on smooth side. Image reference [65].

The measurement results using this test setup are given at 8.2.4.

6.3.4 Measurement with Ultrasound sensor on "candle holder"-rough side

Measurements in section 6.3.3 was made on the smooth target surface of the object -"candle holder". From the result in section 8.2.4, we observe inconsistency of Ultrasound sensor measurement at 560 mm.

I wanted to experiment further about the distance from where this error in Ultrasound measurement starts. Sverre Holm -Professor in the Physics department suggested to check if the roughness of the surface has any effect on measurement range.

In [48], the authors explain about rough surface scattering models of Electromagnetic waves. It is mentioned that roughness of the surface is the dominant factor for scattering of the electromagnetic waves. Roughness of any scattering surface depends on the properties of the waves being transmitted. For Electromagnetic waves the relation for roughness and wavelength is given by ks

where s is statistical roughness parameter and

$k = 2\ \pi\ /\lambda$

λ is Electromagnetic wavelength

Though Ultrasound sensor is used in this thesis, the same relationship as given above is expected. To check this, I roughened one side of the object "candle holder" with a metal scrubber by scrubbing off some surface. Distance measurement using ultrasound sensor was made on this rough surface. Results are given at 8.2.5.

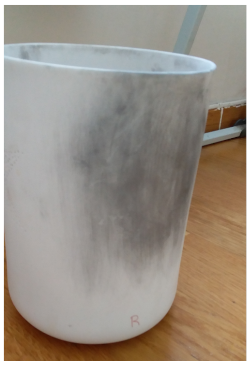


Figure 47: Object surface roughened with metal scrubber

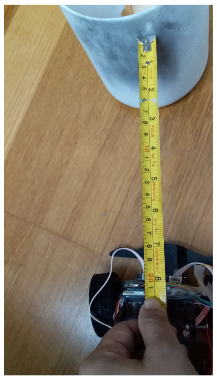


Figure 48: Distance measurement on roughed surface

6.3.5 Measurement with both sensors on "candle holder"-rough side

Experiment was conducted to measure object distance with both Ultrasound sensor sensor and laser sensor simultaneously on object -"candle holder"(rough side). The result is given at 8.2.6

6.3.6 Measurement with both sensors on Flat surface under different ambient light

To check if the Laser sensor readings are affected by ambient light, some experiments were carried out using both Ultrasound and Laser sensor. Experiment was done on flat surface of an object of white colour with ambient light around 300 lux and at 0 lux. The light in the indoor test room was blocked out by blinding the window with blinder and distance measurement was done with dark background. I downloaded Lightmeter application on phone to measure light intensity. Standard equipment can be used for accurate measurement of light intensity.

Test setup with ambient light at 0 lux is shown at figure 49 and test results are given at 8.2.8.

Test results with ambient light around 300 lux are given at 8.2.7.

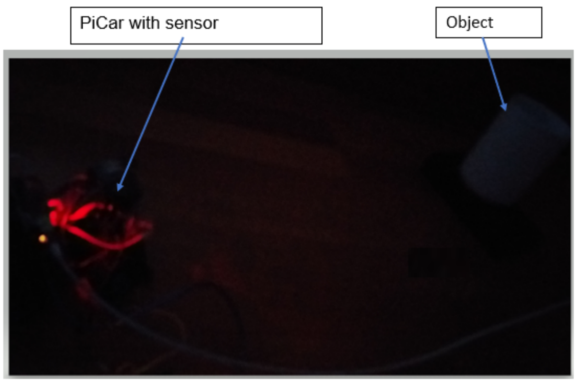


Figure 49: Test setup for distance measurement on Flat object at ambient light at 0 lux.

.

6.3.7 Ultrasound sensor data-Temperature and humidity correction

Velocity of sound was taken at a constant value of 340 m/s i.e v= 340 m/s at equation 2.2.2. This value was used without temperature and humidity correction for calculation of distance from Ultrasound sensor in measurements 1 to 9 at table 6. Arduino Nano BLE 33 sense board was available to me for LAB exercises in Course FYS 3240 at UiO. This board has HTS221- Temperature and Relative humidity Sensor. This board was connected to Raspberry Pi of PiCar. The data from these sensors were used for calculation of velocity of sound [49] according to the following equation.

$$v = 331.4 + (0.6 * Temperature) + (0.01243 * RelativeHumidity) \tag{4}$$

Now, this temperature and humidity corrected value of v was considered for velocity of sound at Equation 2.2.2. The results of Ultrasound distance measurement using this value are presented at 8.2.9.

6.3.8 Data fusion with Ultrasound and laser sensor data

Since we have two sensors, namely Ultrasound and Laser sensor to measure the distance, data from these two sensors can be combined to get better estimate of measured parameter, in our case object distance.

Measurements from two sensors can be combined as given below to get optimal estimate of object distance: [58]

$$\tilde{x} = (\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}) * z1 + (\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}) * z2 \tag{5}$$

where $\tilde{x}$ is the estimate of object distance in mm
$\sigma 1$ is standard deviation of Ultrasound sensor data
$\sigma 2$ is standard deviation of Laser sensor data
$z1$ is actual reading from Ultrasound sensor in mm
$z2$ is actual reading from Laser sensor in mm

For this, an experiment was conducted on Flat white surface object under ambient light of the test room. The standard deviation readings from similar setup in section 8.2.7 were referred. The standard deviation at 520 mm for Ultrasound sensor was 2.66 and for Laser sensor was 10.15 respectively. These readings are used for calculation at Equation 5. The standard deviation in the data from Laser sensor is higher than in the data from Ultrasound sensor. When applying these values at 5, we see that the Laser sensor data is given less weight than Ultrasound sensor data. The results are given at section 8.2.10.

7 Method of Obstacle avoidance and navigation of the PiCar

A Python program which combines Object detection, Obstacle distance measurement and Obstacle avoidance is used. As an obstacle is identified from the image of the PiCar's Camera in real time, distance to the obstacle is measured and in order to avoid collision, the PiCar is deviated from the obstacle.

The PiCar has a servomotor to steer the front wheels. The PiCar wheels are directed towards forward direction at the start. As it moves and identifies an obstacle at a predefined distance, the front wheel is steered towards left or right (defined in the program). Thus the PiCar deviates from the obstacle and continues to move again in forward direction. This was checked with the PiCar. Some of the results are shown in section 8.3. Adafruit PCA 9685 library is used to control the servo motors.

The object detection model SSDLite-MobileNetV2-COCO can identify 90 object classes and I selected a particular object from these 90 objects and defined it as an obstacle (for example, coffee cup). It is possible to redefine the obstacle definition in the program. It should be noted that all objects captured in the camera frame are not at the same distance from the PiCar. i.e Depth information of the objects is not known from the image. So, the classified obstacles can be at different distances from PiCar. Hence, it is difficult to ensure that the same classified object lies in the focal area of distance measuring sensor. The sensor will measure distance of any object which is in its focal area.

The first algorithm I wrote, measured distance to an object after it was classified by the object detection model. This worked well in many situations, but did not handle situations where targets where not classified. Hence, this order was modified so that any nearby target would raise a signal to the autopilot whether the target was classified by the object detection model or not. The additional information obtained by the camera could be used to assist the autopilot in taking actions. For example, a small buoy may easily be avoided by taking a small half circle around while a bird may move away by itself without any avoidance reaction. An additional version of the program was made to simply record the detected objects with timestamp in a text file to have an overall mapping of the objects in the area.

8 Results

Results of Object detection, Object distance measurement and Obstacle avoidance are given in following sections. All the results are from experiments conducted indoors.

8.1 Results for Object detection

As mentioned in chapter 5, the following results are shown:
•Results with pre-trained object detection model on live camera feed.
•Results with object detection model trained on custom dataset
-with different test images as input.
-on live camera feed.

8.1.1 Results of pre-trained object detection model on live camera feed

The following figure shows different objects detected from the camera feed with the pre-trained SSDLite-MobileNetV2-COCO model. This model can identify 90 object classes as given at [42]. The detected objects are displayed with object class name, confidence score of the object class and location with bounding box.

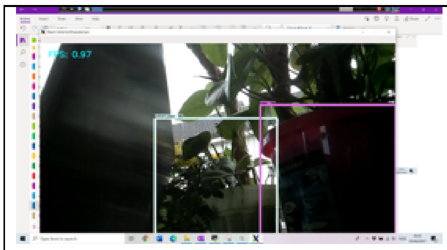
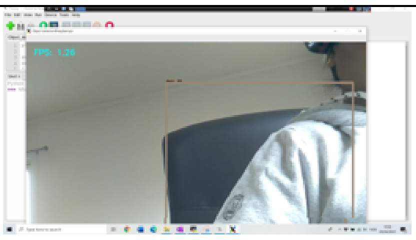
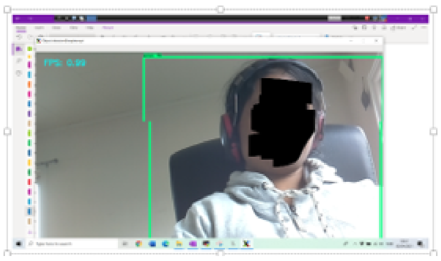
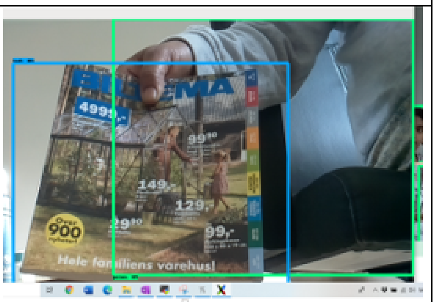
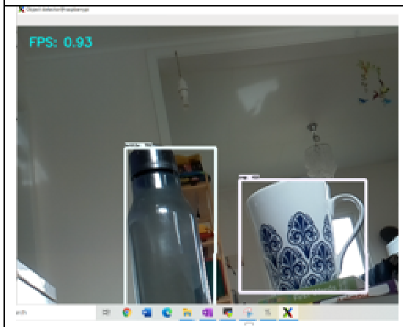
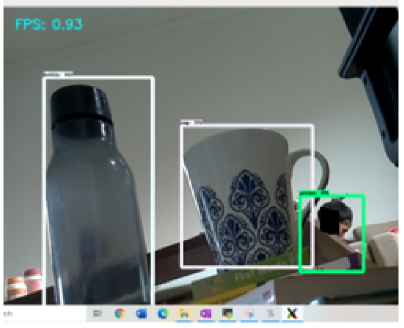
 Potted plant 80% Vase 74%	 Chair 63%
[redacted] Person 79%	 Person 60% Book 40%
 Bottle 70% Cup 41%	 Bottle 91% Cup 77% Person 47%

Figure 50: Output of the Object detection program with SSDLite-MobileNetV2-COCO model. Objects are detected and displayed with object class name, confidence score and bounding box, as described in section 5.1

8.1.2 Results of custom trained model on input images

Model SSDMobileNetV2FPNLite640x640 was trained with custom dataset of bouys, as explained in section 5.2. The trained model was tested with different input images from test data set and the outputs of the model are given in figures 52 and 54.

From test dataset, an image of buoy with water background (figure 51) was given as input to the trained model. The model was able to identify the object 'buoy' with confidence score of 59% and locate it with green bounding box. This is shown in figure 52.

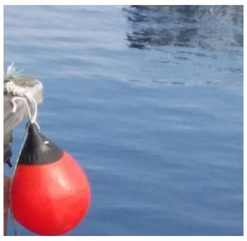


Figure 51: Input Image- buoy with water background. Image reference [65].

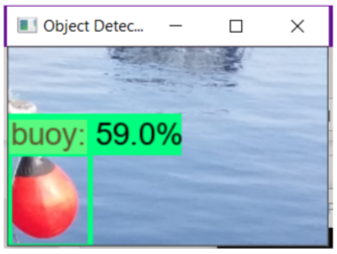


Figure 52: Output of object detection Model -buoy with water background identified

For next test case, an input image of buoy with wooden deck background was chosen from test dataset(figure 53). It was given as input to object detection model. The output of object detection model is shown in figure 54. The model output did not show any bounding box or object class name indicating that the object "buoy" was not detected.

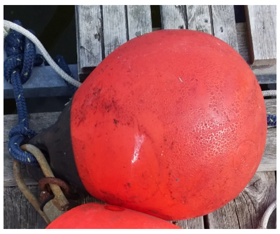


Figure 53: Input Image- buoy with wooden deck background. Image reference [65].

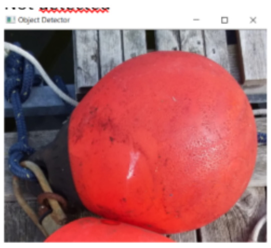


Figure 54: Output of object detection Model -buoy with wooden deck background "not detected"

8.1.3 Results of custom trained model on live camera feed

As explained in section 5.2, model SSD-MobileNetV2-FPNLite640x640 was trained with custom dataset of buoys. The trained model was tested with live camera feed and the outputs of the model are given below.

An orange balloon with black attachment was used to imitate a "buoy". The model was able to detect the object "buoy" with some confidence score and locate it with green bounding box as shown in figure 55 and 56.

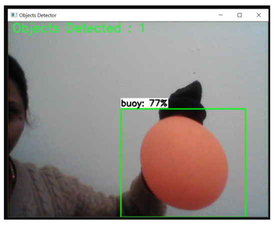


Figure 55: Camera image 1 - orange balloon with black attachment. The object detection model detected buoy with 77 % confidence score.

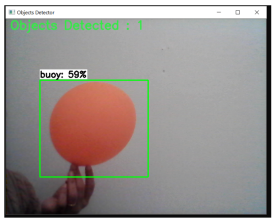


Figure 56: Camera image 2- orange balloon.The object detection model detected buoy with 59 % confidence score.

These results of all the object detection experiments are further discussed in section 9.1

8.2 Results for Obstacle distance measurement

A number of experiments were conducted for object distance measurement and the results are placed in the sections as given in the following table:

Table 6: Overview of results for Object Distance Measurement

Experiment	sensor	Target	Section reference for Result
1	Ultrasound	Flat Black object	8.2.1
2	Laser	Flat Black object	8.2.2
3	Laser	Flat White object	8.2.2
4	Both	Curved object – Coffee Cup	8.2.3
5	Ultrasound	Curved object – candle holder(smooth side)	8.2.4
6	Ultrasound	Curved object – candle holder(rough side)	8.2.5
7	Both	Curved object – candle holder(rough side)	8.2.6
8	Both	Flat object – ambient light around 300 lux	8.2.7
9	Both	Flat object – ambient light at 0 lux	8.2.8
10	Ultrasound	Flat object-temperature and humidity correction of Ultrasound sensor data	8.2.9
11	Both	Flat object-Data fusion from Ultrasound and Laser sensor data	8.2.10

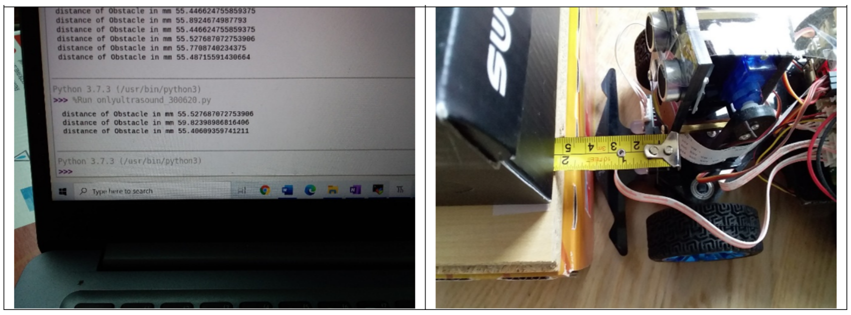
Note: I assume that there may be an error of $+/-2$ mm due to human error in reading of the measurement tape. Accuracy of Ultrasound sensor HC-SR04 is mentioned as 3 mm in datasheet [18]. So, the sensor readings may have 3 mm error. In some datasheets, resolution of Ultrasound sensor HC-SR04 is specified as 3 mm. I have considered accuracy of the Ultrasound sensor HC-SR04 as 3 mm.

The Laser sensor accuracy is mentioned as 5% indoors in section 4.2.3. So, the Laser sensor readings can have an error of +/- 5%.

Velocity of sound was taken at a constant value of 340 m/s i.e $v=$ 340 m/s at equation 2.2.2. This value was used without temperature and humidity correction for calculation of distance from Ultrasound sensor in measurements 1 to 9 at table 6.

8.2.1 Measurement with Ultrasound sensor on flat surface

The following figure shows one set of measurement on flat surface using Ultrasound sensor.



```
distance of Obstacle in mm 55.446624755859375
distance of Obstacle in mm 55.8924674987793
distance of Obstacle in mm 55.446624755859375
distance of Obstacle in mm 55.527687072753906
distance of Obstacle in mm 55.7708740234375
distance of Obstacle in mm 55.48715591430664

Python 3.7.3 (/usr/bin/python3)
>>> %Run onlyultrasound_300620.py

 distance of Obstacle in mm 55.527687072753906
 distance of Obstacle in mm 59.82398986816406
 distance of Obstacle in mm 55.40609359741211

Python 3.7.3 (/usr/bin/python3)
>>>
```

Figure 57: Left side of Image shows Sample measurement. Decimal places were rounded off at later stage. Right side shows the test setup of flat surface with Ultrasound sensor.

One set of measurement data is given in the following table:

Table 7: Measurement data from Ultrasound sensor on a flat object

Sl.No.	Ultrasound sensor (US)Reading(mm)	Measuring tape (MT) Reading(mm)	Error (US-MT) (mm)
1	55	54	1
2	201	200	1
3	300	300	0
4	398	400	-2
5	498	501	-2
6	594	600	-2
7	694	700	-6
8	988	1000	-12

The Ultrasound sensor measured object's distance to reasonable accuracy consistently for a flat object as shown above. A slightly large error of the Ultrasound sensor reading is seen at Sl.No.8. This is discussed at section 9.2.1

8.2.2 Measurement with Laser sensor

Initially a black colour Cardboard box, as shown in figure 40 was used as object to measure distance with Laser sensor. A python program was used to read the Laser sensor data.

Laser sensor measurements were accurate up to 400 mm. After that the error started to increase. When analysing the reason for the error, it was suggested by my thesis guide to check if the Laser sensor's performance was affected by object's colour. The object was changed to white colour and measurements were taken. Following figure shows test setup with new object.

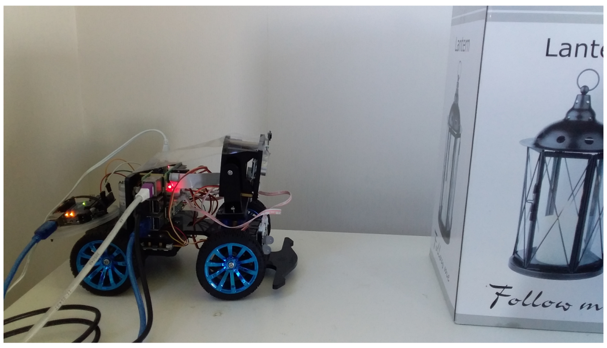


Figure 58: Test setup with White Flat object.

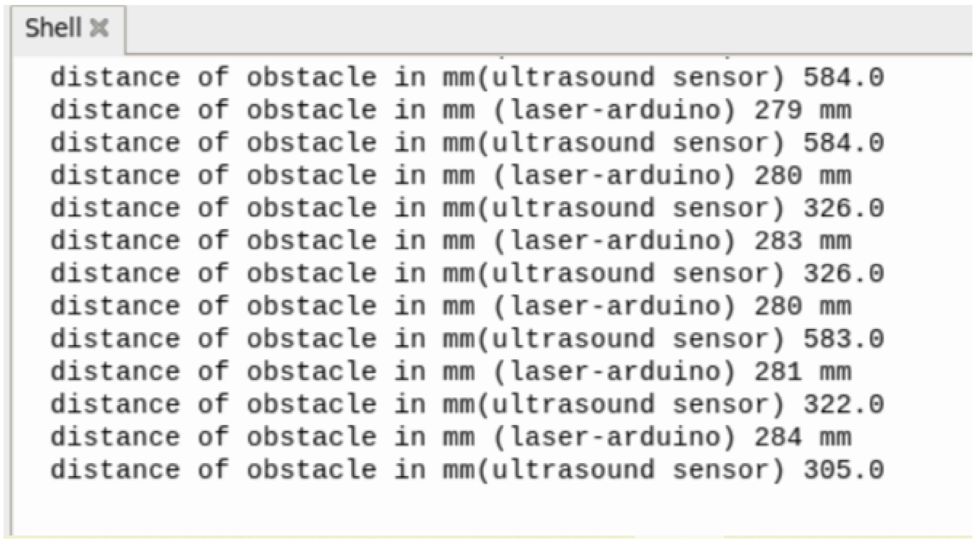
I observed that the measurement was accurate and consistent for long distance ranges on this surface and I measured object distance till 1 meter within the accuracy range of the Laser sensor.

A research paper is given at [47], which discusses how sample patches of different colours reflect the laser light with different intensity by doing experiments on two different laser scanners.

8.2.3 Measurement with both sensors on "Coffee Cup"

As explained in section 6.3.1, simultaneous distance measurement with Ultrasound and Laser sensor was carried out on an object coffee cup. In the test setup, the Ultrasound sensor was positioned approximately 8 - 10 mm behind the laser sensor.

When the object was placed at certain distances, the ultrasound sensor readings were inconsistent, as shown in figures 59 and 60.



```
distance of obstacle in mm(ultrasound sensor) 584.0
distance of obstacle in mm (laser-arduino) 279 mm
distance of obstacle in mm(ultrasound sensor) 584.0
distance of obstacle in mm (laser-arduino) 280 mm
distance of obstacle in mm(ultrasound sensor) 326.0
distance of obstacle in mm (laser-arduino) 283 mm
distance of obstacle in mm(ultrasound sensor) 326.0
distance of obstacle in mm (laser-arduino) 280 mm
distance of obstacle in mm(ultrasound sensor) 583.0
distance of obstacle in mm (laser-arduino) 281 mm
distance of obstacle in mm(ultrasound sensor) 322.0
distance of obstacle in mm (laser-arduino) 284 mm
distance of obstacle in mm(ultrasound sensor) 305.0
```

Figure 59: Example of output from working environment on Object "coffee cup". Laser sensor reading is consistent around 280 mm. Ultrasound sensor reading varies and shows some readings other than 280 mm.

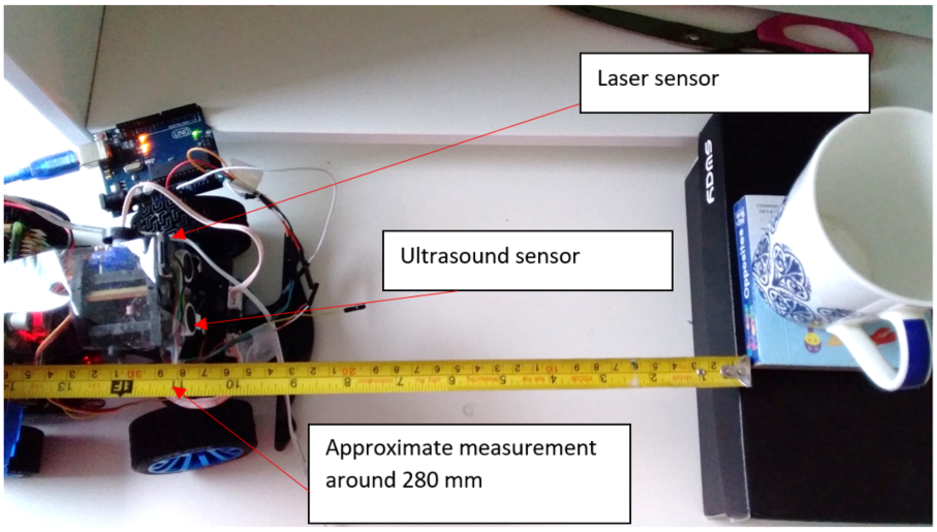
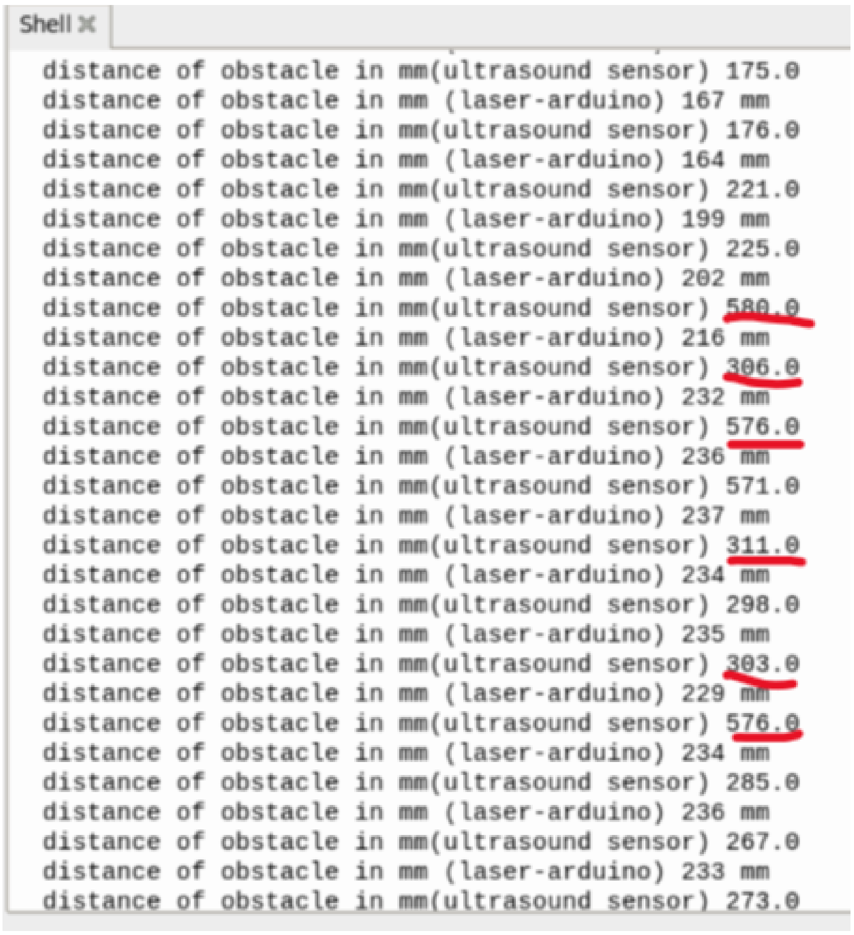


Figure 60: Actual distance measurement of 280 mm with measuring tape on object "coffee cup".

Also, when the object was slowly moved away from the sensors, the ultrasound sensor readings were inconsistent as shown in figure 61.



```
distance of obstacle in mm(ultrasound sensor) 175.0
distance of obstacle in mm (laser-arduino) 167 mm
distance of obstacle in mm(ultrasound sensor) 176.0
distance of obstacle in mm (laser-arduino) 164 mm
distance of obstacle in mm(ultrasound sensor) 221.0
distance of obstacle in mm (laser-arduino) 199 mm
distance of obstacle in mm(ultrasound sensor) 225.0
distance of obstacle in mm (laser-arduino) 202 mm
distance of obstacle in mm(ultrasound sensor) 580.0
distance of obstacle in mm (laser-arduino) 216 mm
distance of obstacle in mm(ultrasound sensor) 306.0
distance of obstacle in mm (laser-arduino) 232 mm
distance of obstacle in mm(ultrasound sensor) 576.0
distance of obstacle in mm (laser-arduino) 236 mm
distance of obstacle in mm(ultrasound sensor) 571.0
distance of obstacle in mm (laser-arduino) 237 mm
distance of obstacle in mm(ultrasound sensor) 311.0
distance of obstacle in mm (laser-arduino) 234 mm
distance of obstacle in mm(ultrasound sensor) 298.0
distance of obstacle in mm (laser-arduino) 235 mm
distance of obstacle in mm(ultrasound sensor) 303.0
distance of obstacle in mm (laser-arduino) 229 mm
distance of obstacle in mm(ultrasound sensor) 576.0
distance of obstacle in mm (laser-arduino) 234 mm
distance of obstacle in mm(ultrasound sensor) 285.0
distance of obstacle in mm (laser-arduino) 236 mm
distance of obstacle in mm(ultrasound sensor) 267.0
distance of obstacle in mm (laser-arduino) 233 mm
distance of obstacle in mm(ultrasound sensor) 273.0
```

Figure 61: Example of output from working environment on test object "coffee cup"- Laser sensor readings show gradual increment from 167 to 236 mm. Ultrasound sensor are inconsistent and show 580 mm, 306 mm as underlined.

The inconsistency in Ultrasound readings at figure 59 and 61 may be due to the reason that during movement, the object was not positioned correctly within the focal area of the Ultrasound sensor. Also, because of the object's shape and surface, the transmitted Ultrasound waves could have been scattered and were not properly received back at the sensor receiver. But since the Laser sensor has pointed beam, it was able to measure the object distance more consistently.

8.2.4 Measurement with Ultrasound sensor on "candle holder"-smooth side

Section 6.3.3 explains about an experiment carried out on smooth side of the candle holder with Ultrasound sensor. The results are plotted in figure 62. The results of experiment on rough side of the candle holder are also plotted for comparison.

Measurement data showed that the inconsistency of Ultrasound sensor measurement started when the actual object distance was 560 mm.

8.2.5 Measurement with Ultrasound sensor on "candle holder"-rough side

Section 6.3.4 explains about an experiment carried out on rough side of the candle holder with Ultrasound sensor. The results are plotted in 62 along with results from section 8.2.4.

From figure 62, we observe that the ultrasound sensor measures correct distance till 850 mm. The measurement inconsistency starts when the object is at a distance around 930 mm.

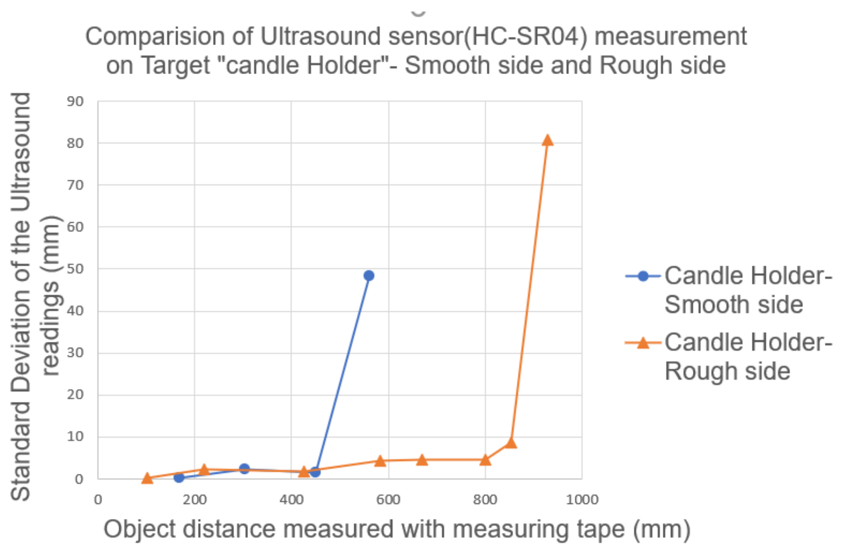


Figure 62: Comparison of Ultrasound sensor measurement on test object "Candle holder" - Smooth side and Rough side.

From the plots in figure 62, It can be seen that the detection range of the ultrasound sensor has improved by a factor of 1.6 on the rough surface. **Note:** This experiment has been done on only one object.

8.2.6 Measurement with both sensors on "candle holder"- rough side

Distance measurement with both the sensors on the rough side of the target candle holder is explained in section 6.3.5. The results of the measurement are plotted at figure 63.

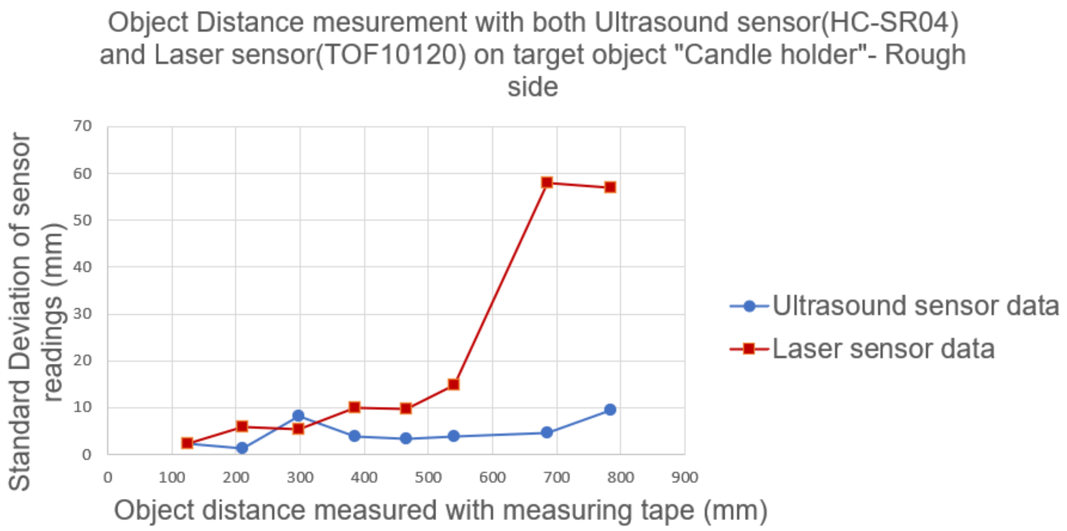


Figure 63: Object distance measurement with both Ultrasound sensor and Laser sensor on test object "Candle holder" - Rough side.

As mentioned previously, the Ultrasound sensor is positioned approximately 10 mm behind the Laser sensor. Since we are are comparing the standard deviations of readings, this factor is considered as addressed. I observed that while the ultrasound sensor readings were consistent at 685 mm and 785 mm, the laser sensor readings were inconsistent. The reason for this behaviour of Laser sensor could not be explained and may need further experiments.

8.2.7 Measurement with both sensors on Flat object surface- ambient light around 300 lux

As explained in section 6.3.6, results of distance measurement using both the sensors on Flat target surface under ambient light around 300 lux is plotted in figure 64.

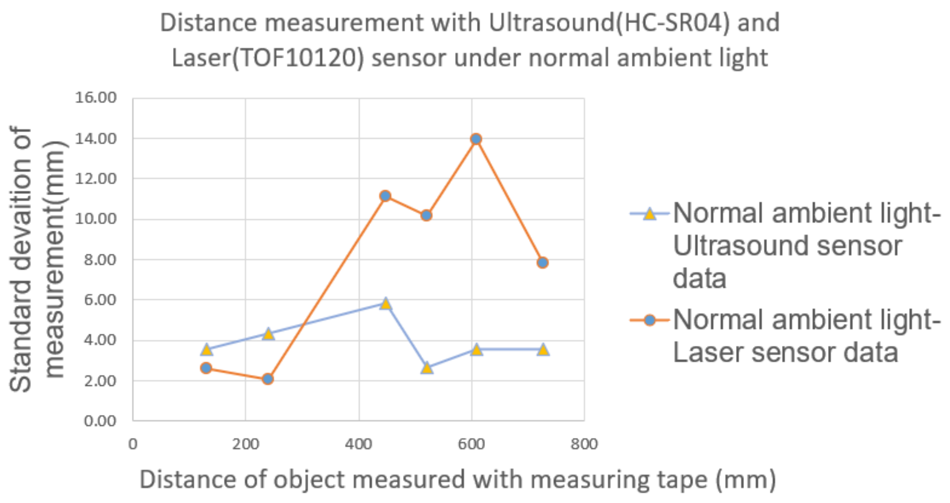


Figure 64: Object distance measurement with both Ultrasound sensor and Laser sensor on Flat object under normal ambient light. Here, normal ambient light refers light intensity around 300 lux

Standard deviation of Laser sensor readings is higher than the Ultrasound sensor data. Standard deviation of Laser sensor data at object distance of 727 mm is less than at 610 mm. The reason for this decrease could not be determined. The Ultrasound sensor has same standard deviation at both values.

Comparison of the measurement in section 8.2.7 and 8.2.8 is given in section 9.2.4

8.2.8 Measurement with both sensors on Flat object surface-ambient light at 0 lux

The results of distance measurement using both the sensors on Flat target surface with ambient light at 0 lux (section 6.3.6) is plotted in figure 65.

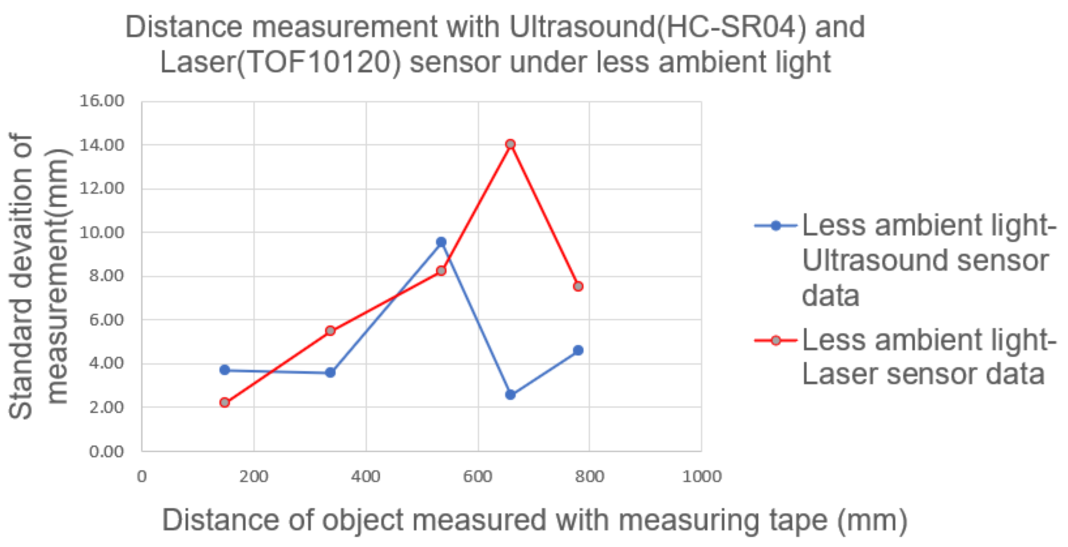


Figure 65: Object distance measurement with both Ultrasound sensor and Laser sensor on Flat object under less ambient light. Here, less ambient light refers 0 lux.

In general,standard deviation in the data of Laser sensor is higher than the standard deviation in the data of Ultrasound sensor. The standard deviation of Ultrasound sensor reading at 535 mm is higher than the Laser reading. This may be due to some measurement process error since the ultrasound sensor had good accuracy for flat surface target in earlier experiments. Some more experiments may be needed further to verify this. Standard deviation of Laser sensor data at object distance of 727 mm is less than at 610 mm for unknown reason. But this behaviour matches with the readings for ambient light around 300 lux.

Comparison of the measurement in section 8.2.7 and 8.2.8 is given in section 9.2.4

8.2.9 Ultrasound sensor data with and without temperature and humidity correction

The distance measurement of Ultrasound sensor data with and without temperature and humidity correction is explained in section 6.3.7. Measurement was carried out first at room temperature of 26°C and relative humidity of 58%. Then, the indoor environment was heated with room heater and another set of reading was taken at room temperature of 27°C and relative humidity of 52%. The results are plotted below.

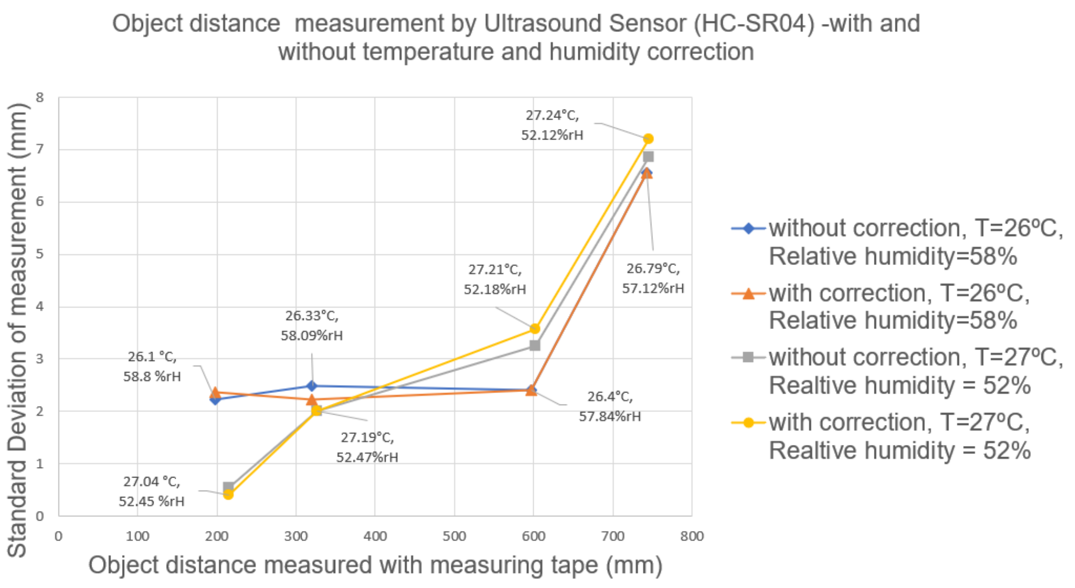


Figure 66: Distance measurement from Ultrasound sensor with and without temperature and humidity correction

From the plots, we observe that temperature and humidity correction has less influence on Ultrasound sensor measurement.

Temperature and humidity correction is possible for laser sensor data as well. It is left as a future work due to shortage of time.

Note: Here, the instantaneous value of temperature and relative humidity were used for the calculation.

The HTS221 specifies the sensor details as follows:

Humidity Accuracy : $+/-3.5\%$ at 20 to 80 $\%$ rH

Humidity hysteresis : $+/-1\%$

Humidity Response time : 10 sec

For a sample relative humidity measurement of 37.49 $\%$ rH, this accuracy and hysteresis value accounts as 1.68 $\%$ rH variation (apart from Ultrasound sensor accuracy).

Similarly for temperature sensor,

Temperature accuracy:$+/-1°C$ at 0 to 60 °C

Response time for temperature is 15 seconds.

Detailed technical specification of HTS221 sensor can be found at [62]. In practical, there appears to be more error from HTS221 sensor as in the discussion thread [63]. Hence, temperature and relative humidity readings of HTS221 should be verified with readings from another standard sensor.

8.2.10 Data fusion between Ultrasound sensor and laser sensor data

Figure 67 shows object distance estimate by combining data from both Ultrasound and Laser sensor. The distance measurement data from Ultrasound sensor and Laser sensor is also plotted. Ultrasound sensor data was temperature and humidity corrected.

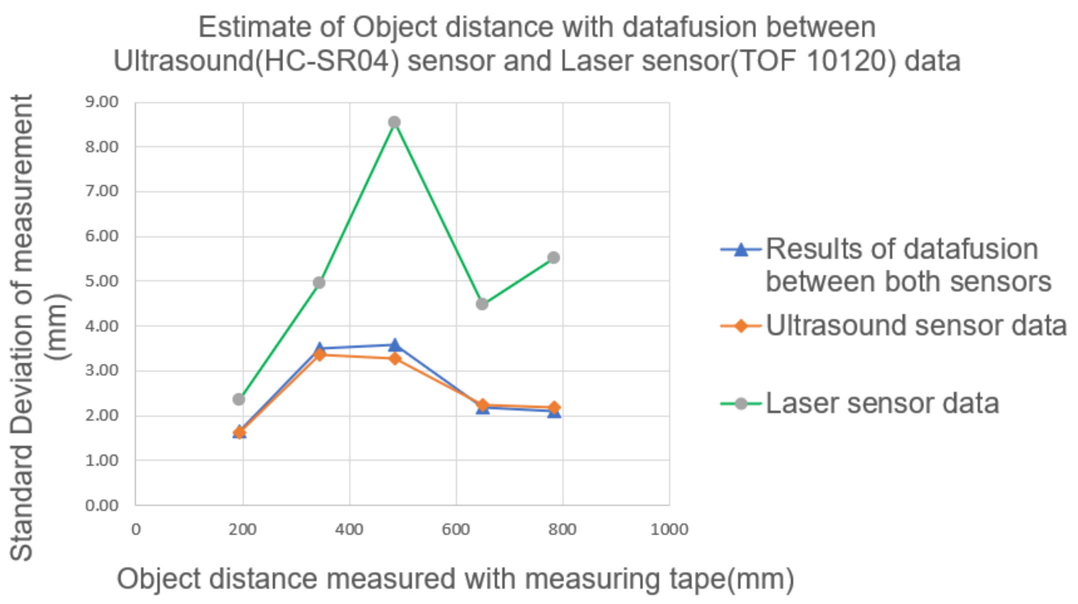


Figure 67: Distance estimate from datafusion between Ultrasound sensor and Laser sensor measurements. Here, the measurement from Ultrasound sensor is temperature and humidity corrected

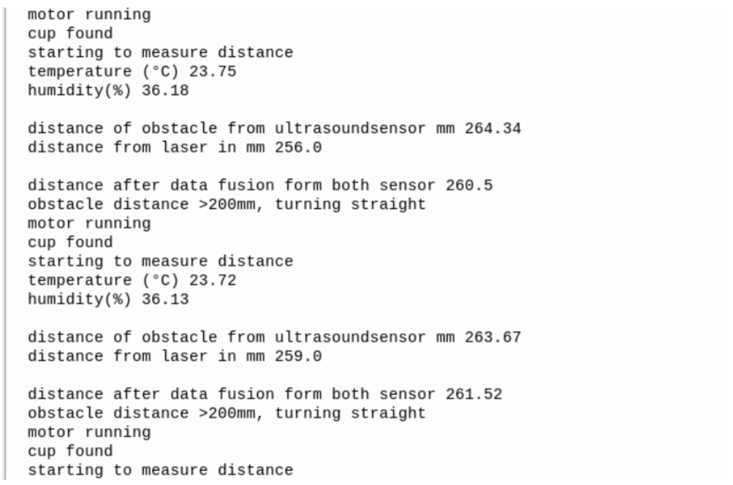
.

From the plot, we see that standard deviation of Laser sensor is more compared to Ultrasound sensor. But, optimal estimate using data fusion, is better than the Laser sensor data, almost equal to the Ultrasound sensor data.

8.3 Results for Obstacle avoidance and Navigation of PiCar

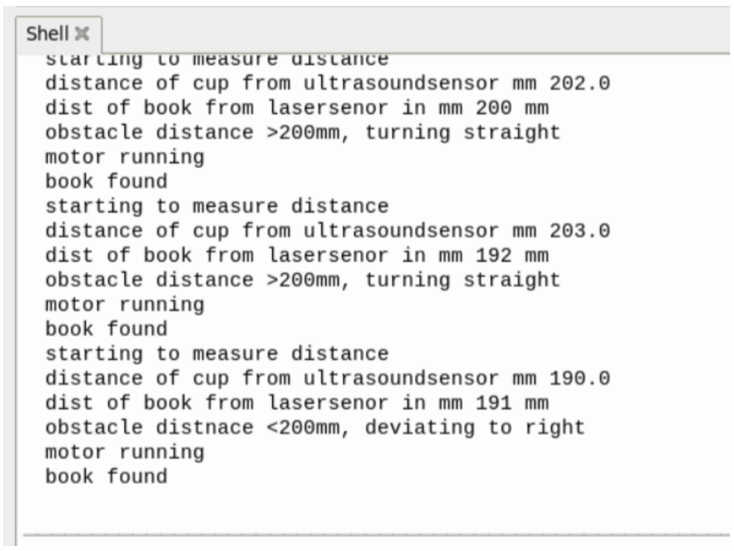
As mentioned in section 7, a Python program which combines Object detection, Obstacle distance measurement and obstacle avoidance is used. Once the Python program starts to execute, PiCar starts to move forward and Picamera continuously captures the images in this direction. The object detection part of the program detects objects in the live feed of camera. The object " coffee cup" is defined as an obstacle in the program. It is possible to define multiple objects as obstacle. The PiCar can be programmed to deviate if any of these obstacles are identified. Model SSDLite-MobileNetV2-COCO can identify 90 object classes. Once an obstacle is detected, obstacle distance measurement is carried out. If this obstacle is found within a pre-distance distance of 200 mm, PiCar is deviated form the obstacle. Otherwise PiCar travels straight ahead.

Figures 68, 69 show the results in working terminal during trials of obstacle avoidance and navigation of PiCar.



```
motor running
cup found
starting to measure distance
temperature (°C) 23.75
humidity(%) 36.18

distance of obstacle from ultrasoundsensor mm 264.34
distance from laser in mm 256.0

distance after data fusion form both sensor 260.5
obstacle distance >200mm, turning straight
motor running
cup found
starting to measure distance
temperature (°C) 23.72
humidity(%) 36.13

distance of obstacle from ultrasoundsensor mm 263.67
distance from laser in mm 259.0

distance after data fusion form both sensor 261.52
obstacle distance >200mm, turning straight
motor running
cup found
starting to measure distance
```

Figure 68: Obstacle avoidance result-1. Detected obstacle "cup". When the obstacle is at more than 200mm distance, the PiCar runs straight.



```
starting to measure distance
distance of cup from ultrasoundsensor mm 202.0
dist of book from lasersenor in mm 200 mm
obstacle distance >200mm, turning straight
motor running
book found
starting to measure distance
distance of cup from ultrasoundsensor mm 203.0
dist of book from lasersenor in mm 192 mm
obstacle distance >200mm, turning straight
motor running
book found
starting to measure distance
distance of cup from ultrasoundsensor mm 190.0
dist of book from lasersenor in mm 191 mm
obstacle distnace <200mm, deviating to right
motor running
book found
```

Figure 69: Obstacle avoidance result-2. Detected obstacle "book". When the obstacle is at less than 200mm distance, the PiCar deviates to right. Otherwise it runs forward.

9 Discussion of Results

The main task for this thesis has been to design a simple and cost effective surveillance system for an autonomous survey vessel. Such a system should be able to detect obstacles in front of the boat and provide information so that the autopilot can take actions to avoid these. Actions could be to find alternative paths around, stop and wait or simply contact an operator for help. In this thesis, the focus has been on the sensors, and on detection and classification of obstacles. We have implemented simple reactions to demonstrate proof of concept, but this is not the main topic here. To simplify the testing, we used a small model car as a substitute for the boat.

Pre-trained object detection model which was loaded on Raspberry Pi of PiCar identified objects in the Lab setup. Also, an object detection model was trained to detect objects at lake, such as buoys. This model was tested in the Lab setup and gave satisfactory results. The results are elaborated at section 9.1.

The obstacle distance was successfully estimated with datafusion between Ultrasound and Laser sensor measurement. The results are given at section 9.2.

Obstacle avoidance was demonstrated by using PiCar. The PiCar was steered away from the obstacle by controlling the front wheel movement. The results are given at section 9.3.

For a real ASV, detection range and reaction speed must be improved (e.g by Long range sensors, Multi-direction distance measurement,faster processing).

9.1 Discussion of results for Object detection

9.1.1 Object detection with pre-trained model

The pre-trained model (SSDLite-MobileNetv2-COCO) gave satisfactory results in real time object detection (section 8.1.1). Some instances of false - positives detection was noted but this is expected as the SSDLite-MobileNetV2-COCO model prioritises speed of the detection than accuracy. Instances of non detections were noticed in rare occasions. Overall, the object detection program satisfactorily identified the objects among 90 classes with reasonable accuracy at the frame rate of 1.5 fps (frame per second).

According to Raspberry Pi- Picamera V2 specification, for camera resolution of 640 X 480, the achievable frame rate can be 40- 90 fps(Frame Per Second). But, in practical experiments maximum frame rate of 1.5 only could be achieved even with a resolution of 640 X 480. The reason may be OpenCV- VideoCapture function for object detection from camera. This function continuously buffers the video frames. Alternate methods can be explored to achieve high frame rate.

9.1.2 Custom trained object detection model with image input

In section 8.1.2, the input image in figure 51 contains the object in a clear way. So, the model is able to identify the object with more confidence score.

The image in figure 53, has a slightly complex background. Hence, the model was not able to identify the object. If training of the model was done with image dataset consisting of lot of images with similar complex background then the result may improve.

These results are satisfactory and encouraging as the model is able to identify the object even with training of the model with a small dataset of only 36 images.

Since we have trained the model with single object class and small dataset, chances of overfitting the model to the trained dataset is possible. The results will definitely improve and the model will perform better by training with a large dataset of 100 or more images with different backgrounds.

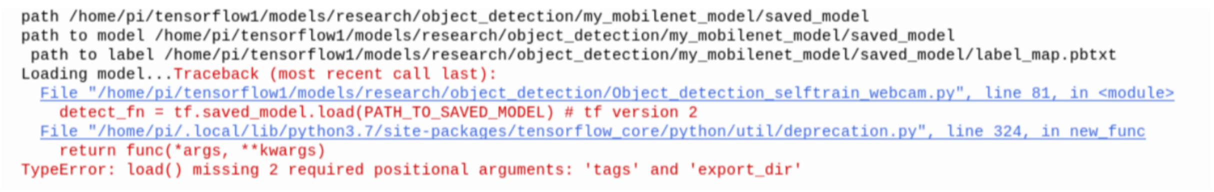
9.1.3 Custom trained object detection model with live camera feed

Figure 55 in section 8.1.3 shows that when an orange balloon with black attachment matched more to actual buoy, the model predicted the object with more confidence. But only with an orange balloon as in figure 56, the confidence score is less.

The results with live object detection are promising and the prediction can definitely be improved by training with a large dataset. Also large data set can be collected for more common objects like tree branches, thick logs, swans, ducks and the model can be trained to identify such objects. The trained model can be exported to Raspberry Pi of PiCar and checked for object detection.

9.1.4 Exporting and checking the trained model on Raspberry Pi of PiCar

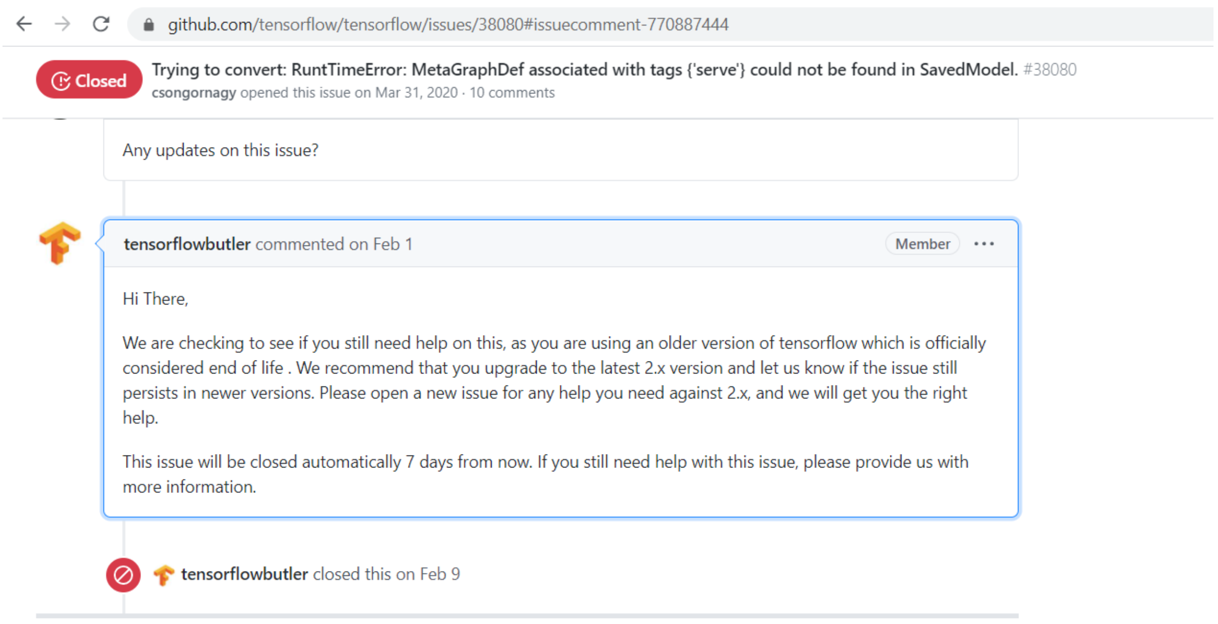
The trained model was saved in Laptop and this saved model was exported to Raspberry Pi of PiCar. A Python program was used to load this model for detecting object. Errors were encountered due to mismatch of TensorFlow versions in Laptop and Raspberry Pi of PiCar. The model was saved using TensorFlow- version 2 in Laptop and Raspberry Pi of PiCar had TensorFlow-version 1. The error is shown below:



```
path /home/pi/tensorflow1/models/research/object_detection/my_mobilenet_model/saved_model
path to model /home/pi/tensorflow1/models/research/object_detection/my_mobilenet_model/saved_model
 path to label /home/pi/tensorflow1/models/research/object_detection/my_mobilenet_model/saved_model/label_map.pbtxt
Loading model...Traceback (most recent call last):
  File "/home/pi/tensorflow1/models/research/object_detection/Object_detection_selftrain_webcam.py", line 81, in <module>
    detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL) # tf version 2
  File "/home/pi/.local/lib/python3.7/site-packages/tensorflow_core/python/util/deprecation.py", line 324, in new_func
    return func(*args, **kwargs)
TypeError: load() missing 2 required positional arguments: 'tags' and 'export_dir'
```

Figure 70: Error due to TensorFlow versions mismatch in Raspberry PI (TensorFlow1) and saved object model in Laptop(TensorFlow2). Error shows missing parameters in TensorFlow1

Attempts were made to address this error in Raspberry Pi. But referred on-line resource from TensorFlow suggests upgrading TensorFlow to version-2 as TensorFlow version-1 is officially outdated as shown below.


github.com/tensorflow/tensorflow/issues/38080#issuecomment-770887444

Closed Trying to convert: RuntTimeError: MetaGraphDef associated with tags {'serve'} could not be found in SavedModel. #38080
csongornagy opened this issue on Mar 31, 2020 · 10 comments

Any updates on this issue?

tensorflowbutler commented on Feb 1 Member

Hi There,

We are checking to see if you still need help on this, as you are using an older version of tensorflow which is officially considered end of life . We recommend that you upgrade to the latest 2.x version and let us know if the issue still persists in newer versions. Please open a new issue for any help you need against 2.x, and we will get you the right help.

This issue will be closed automatically 7 days from now. If you still need help with this issue, please provide us with more information.

tensorflowbutler closed this on Feb 9

Figure 71: On -line resource from TensorFlow suggesting up gradation of TensorFlow version to avoid error in loading saved model.- Accessed on April, 2021

It is obvious that the exported model will perform in Raspberry Pi with TensorFlow version 2. However, upgradation of TensorFlow version in Raspberry Pi was not taken up due to shortage of project time and can be taken up as future work.

If the model was trained with Tensorflow version 1 and other dependant software, the model could have been exported without error to Raspberry Pi. It can be verified in future work.

9.2 Discussion of results from obstacle distance measurement

Experiments were conducted to measure the distance of the object by using Ultrasound sensor and Laser sensor as explained in section 6 and the results are given in section 8.2. It has to be noted that object detection was not carried out in these experiments. For measuring object distance, the object should be in the focus range of the Ultrasound and Laser sensor. Only a single object was in the focal area of both Ultrasound sensor and Laser sensor.

9.2.1 Measurement with Ultrasound sensor on flat surface

For calculation of distance from Ultrasound sensor, sound velocity was taken as 340 m/s without temperature compensation and humidity correction for experiments 1 to 9 in table 6.

Ultrasound sensor measurement error at Sl.No.8 of 7 in section 8.2.1 can be analysed as below.

If small humidity correction is ignored, then temperature variation from reading at Sl.No.7 to Sl.No.8 could have been the reason for a slightly high error at Sl.No.8. Consider the following equation [46] for sound velocity.

$$v = 331.3 + (0.606 * Temperature) \tag{6}$$

From the table 7 data, we can deduce that the temperature Sl.No.7 could have been 19.16 °C and at Sl.No.8 could have been 21.08 °C.

Air temperature was not measured in this experiment. Temperature increase of 2°C would have taken considerable amount of time. Time detail was not recorded to check any time delay between Sl.No.7 and Sl.No.8. So, the given reason for the error could not be confirmed. It might have been due to error in measurement process also.

9.2.2 Measurement with Laser sensor

We observe from section 8.2.2 that Laser sensor readings are affected by the target object colour (reflectivity of the object). This may have impact on the target distance measurement in both indoor and outdoor environment.

9.2.3 Measurement with Ultrasound sensor on "Candle holder"-smooth and rough side

From the data at section 8.2.5, we observe that Ultrasound sensor readings are influenced by roughness of object surface. The detection range of the Ultrasound sensor improved by a factor of 1.6 on a rough surface as compared to the smooth surface. Experiment has been carried out on only one object. Some more experiments may be needed to confirm the observation.

9.2.4 Measurement with both sensors on Flat object surface in different ambient light

Data from sections 8.2.7 and 8.2.8 are plotted together in figure 72.

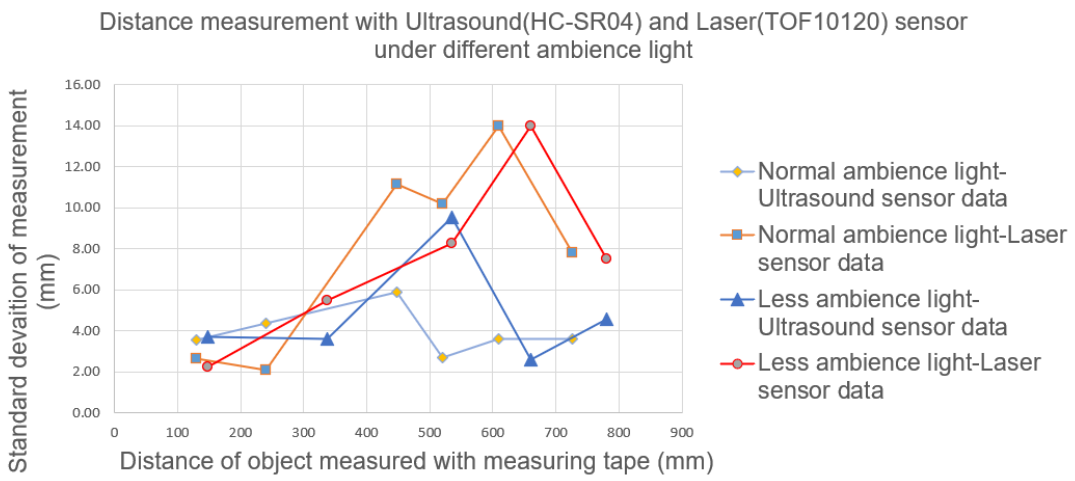


Figure 72: Plot of object distance measurement with Ultrasound and Laser sensor measurement on Flat object with different ambient lighting. Here normal ambient light refers around 300 lux and less ambient light refers to 0 lux.

The plots show that Laser sensor is able to measure the object distance under both ambient light conditions with almost same accuracy. The ultrasound sensor had more error at 535 mm. This could be due to measurement process error since the ultrasound sensor had good accuracy for flat surface target in earlier experiments. Some more experiments with different object surfaces may be needed further to verify this.

9.2.5 Ultrasound sensor data with and without temperature and humidity correction

From results at 8.2.9, we observe that the temperature and humidity correction has no significant effect on the Ultrasound sensor readings. As discussed in [63], the temperature and humidity measurements of Sensor HTS221 need to be verified with another standard sensor.

9.2.6 Data fusion between Ultrasound sensor and laser sensor data

From figure 67 in section 8.2.10, we see that standard deviation in the data from Laser sensor is higher than in the data from the ultrasound sensor and thereby given less weight in data fusion calculation. Standard deviation in the data from datafusion estimate is better than laser sensor data.

9.2.7 Measuring object distance in different directions

In this thesis, one number of Laser sensor (TOF 10120) was connected to Arduino Uno board. Arduino Uno was interfaced with Raspberry Pi in PiCar as explained in section 6.2. The Laser sensor was mounted on PiCar in a way that it points towards the object in front.

In standard setup of PiCar, the Ultrasound module(HC-SR04) is mounted on a bracket controlled by servomotor. The servomotor moves the bracket to left, right, up or down direction which in turn moves the Ultrasound sensor. With these movements, the Ultrasound sensor scans the surrounding in different directions to measure obstacle distance. Thus with single Ultrasound sensor, it is possible to scan the area surrounding the sensor. Such an arrangement can be made for Laser sensor also.

Alternatively, with an I2C multiplexer module like **Adafruit TCA9548A Multiplexer**, upto 8 numbers of TOF 10120 sensors can be connected to Arduino Uno board. These sensors can be mounted such that obstacle distance from multiple directions can be measured without a need to move the sensors. Precautions should be taken to avoid crosstalk between the sensors.

Multiple Ultrasound sensors can also be mounted in PiCar to measure distances of obstacles in different directions [29].

9.3 Discussion of results for Obstacle avoidance and Navigation of PiCar

Section 7 explains the method of obstacle avoidance and navigation of PiCar. Results of the experiments are given at section 8.3. The results show that the PiCar is successfully steered away, when identified obstacle "Coffee cup" or "book" is within a pre-defined distance of 200 mm. The Picar continues to run straight when there is no obstacle in front of it.

By controlling the servomotor connected to the front wheels of PiCar, the wheels are steered to right to avoid obstacles. The steering direction and steering angle of the wheels, Speed of the PiCar movement can be changed in the Python program as needed.

With indoor test setup and PiCamera, the object detection model could classify an object at a distance of approximately 3.2 meters, as this object was in the focal area of PiCamera. But, this object was not in focal area of the Ultrasound sensor or Laser sensor. So, the sensors could not measure the distance of this object. As given in the note of Chapter6, distance of the objects which are in focal area of the Ultrasound sensor or Laser sensor can only be measured. This means, even if multiple objects are identified in an image, these objects can be at different distances from PiCamera. Only the distance of the object in the focal distance of Ultrasound sensor and Laser sensor can be measured in the current setup.

Obstacle avoidance was checked by either moving the obstacle or PiCar. Both were not moved simultaneously. Speed of the Picar or the moving object is not accounted in the experiments.

The speed of Picar was measured at minimum of approximately 0.8 cm /sec to maximum of 1.6 cm/sec. In my Lab setup, the speed of PiCar is affected by following: additional hardware and materials mounted on PiCar as described in section4, PiCar Power cable and Interface cable length, speed setting of motor in the program. Without additional hardware, speed of PiCar was measured at maximum of 3.6 cm/sec. Also, from start of program it takes almost one minute to load the model and start object detection from image. After that, the image update rate is 1.5 frame per second and the objects are detected from the images. With maximum PiCar speed and time taken for processes like object detection and obstacle distance measurement, there is a possibility of PiCar crashing into an obstacle. Multiprocessing in the program and other alternatives can be explored to improve this.

In current setup, PiCar is steered away from the obstacle either towards left or right direction as programmed. There is no predefined path for the PiCar to follow. Hence, PiCar can deviate too much from the starting point. Also, repeated deviation in same direction may make the PiCar to move only in a limited area without exploring new area. A solution to predefine the path of PiCar may be explored. Mobile robotics platform have pre-built functions for navigation and path planning as discussed in section 3.6.1. It may be considered for ASV applications.

This surveillance system is developed for ASV. So, once an obstacle is detected at pre-defined distance, it can be programmed to send signals out from Raspberry Pi GPIO pins to Autopilot system of ASV. Autopilot system can steer the ASV from the obstacle.

Full measuring range of Ultrasound sensor(HC-SR04) is 4 meters and Laser sensor(TOF10120) is 1.8 meters. Measurement till 1 meter only has been carried out in this thesis.

A method of measuring object distance using stereo camera is explained at [64], which can be explored to eliminate the need of distance measuring sensors like Ultrasound sensor and Laser sensor.

9.3.1 Relating results from Lab to ASV in lake environment

A survey vessel should be able to run at a speed of at least 3-5 knot or 5.6 -9.3 km/hour to be able to cover a reasonable part of a lake during a night. Moreover, for a real survey vessel, it would be important to detect and avoid targets at distances of at least some meters.

In the lab, with the selected PiCar, sensors and processing units, we could achieve speed within 0.8 to 1.6 cm / sec, which is about 1/100 of the speed needed for a real ASV and a maximum measured distance of 1 meter which is at least 10 times less than needed at the lake. The limitation of speed can be overcome by placing the surveillance system on the ASV [3]. By selecting long range sensors, the measurable distance of the surveillance system can be improved. There are different long range distance sensors are available in market, which may be selected based on the needed range and accuracy. For example, the B87A-b200416 Long Range Laser distance sensor from Chengdu JRT Meter Technology Co., Ltd. can measure ranges up to 100 meter at an accuracy of +/-3 mm and it can be connected to Raspberry Pi.

In this thesis, we have applied Arduino uno and Raspberry Pi for the processing. We wanted to see if this was sufficient, and to ease the experimenting. For the real ASV it would not be difficult to port the developed algorithms and tested procedures to more powerful processing systems.

Surveying of lake is carried out at night in calm water condition normally. Hence, the achieved frame capture rate may be sufficient. Also, accelerometers on the ASV measure the vibration levels and abort the mission if these levels cross certain limit. From IMU sensor in Arduino Nano board, 3-axis acceleration can be measured and used for limit setting.

9.4 Project risk Management

Some project risks were identified and avoided as given below:

1. Procurement of Laser sensor TOF 10120:

   In addition to Ultrasound sensor in PiCar, I wanted to experiment with laser sensor(TOF10120) for measuring object distance. Technical specifications of different laser sensors were studied and the procurement was completed in April - May 2020 to avoid any delivery delays due to Covid Pandemic situation and hence, any project delay.

2. Laser sensor (TOF10120) measurement with Arduino Uno to avoid disturbance of PiCar hardware setup :

   The laser sensor(TOF10120) can be directly connected to Raspberry Pi of PiCar and data can be acquired through I2C interface. But in the existing PiCar setup, I2C interface pins of Raspberry Pi are already used for servo control. Also there is an interface board which connects sensors and motors of PiCar to Raspberry Pi. This board needs to be dismantled from PiCar or pins at this board need to be used. Any tempering with this board may result in damage of the fragile Picamera CSI cable which is located very close. As there was no spares or repair facility at my home office, all these risks were avoided by connecting the laser sensor to Arduino Uno board. It was in turn connected to Raspberry Pi of PiCar via USB interface.

9.5 Future work

1. Challenges with standard operating method of PiCar are detailed at section 4.4 along with possible alternate solutions. These solutions may be implemented.

2. Errors due to TensorFlow version mismatch may be resolved either by upgrading TensorFlow version in PiCar or training the object detection model with TensorFlow1 in Laptop.

3. With current algorithm, the PiCar starts moving forward and if there is an obstacle, then it steers to the right side of PiCar. After crossing the obstacle it starts moving forward. If there is another obstacle, then it steers to right side again. Thus, repeated deviation within short duration may make the PiCar to move again and again in the same area without exploring new area. i.e, there is no predefined path for the PiCar to follow. A solution to predefine the path of PiCar may be explored. Mobile robotics platform have pre-built functions for navigation and path planning as discussed in section 3.6.1. It may be considered for ASV applications.

4. Multi- direction obstacle distance measurement with multiple Ultrasound sensors or Laser sensors may be implemented as discussed in sections 3.4 and 9.2.7.

5. Object detection at frame rate of 1.5 fps only could be achieved in the program, where as the PiCamera specifies frame rate in range of 40-90 fps at 640 X480 resolution. The reason may be OpenCV- VideoCapture function for object detection from camera. This function continuously buffers the video frames. Alternate methods may be explored to achieve high frame rate.

6. For the purpose of object detection- Raspberry Pi, PiCamera and Python object detection program is sufficient. This system may be used as a standalone part. It may be fitted on any Survey vehicle Platform and its performance may be verified outdoors.

7. Object detection part of the program generates location of objects also as output in the image. This information is not found useful in this thesis, as we are only interested about the object classification. Only object classification may be attempted in future work. The location detail of an object may be used in applications to track the object.

8. Obstacle distance measurement may be carried out using any one of the following configurations:
   a) Python program with Raspberry PI and Ultrasound sensor.
   b) Arduino sketch program with Arduino Uno board and Ultrasound sensor.
   c) Arduino sketch program with Arduino Uno board and Laser sensor.
   Any of these configurations may be used as an independent module on Survey vehicle Platform and system performance may be verified outdoors for full measurement range of the sensor.

10 Conclusion

This thesis successfully demonstrated object detection by detecting the predefined object classes in indoor environment. Most of the objects from listed classes in the COCO dataset were identified correctly in real time with the processing power of Raspberry Pi 3 model B+. Some instances of false - positives detection occurred. This is, however, a known drawback since the chosen model SSDLite-MobileNetV2-COCO prioritises speed over accuracy. Also, this model is suited for the computing power of Raspberry Pi3 Model B+.

To detect objects in the water environment, object detection model was successfully trained to detect object "buoy" with a small data set of 36 images. Object detection with live camera feed was tested successfully on a test object. This gives confidence for further training of object detection model with small data set for any new object in lake environment.

Obstacle distance measurement with Ultrasound sensor(HC-SR04) was carried out with good accuracy on flat objects. Influence of object shape on Ultrasound measurement was shown with different experiments on Flat and curved surfaces. Influence of surface texture was also shown and I observed that detectable range of ultrasound sensor was better on rough surface by a factor of 1.6. It was also demonstrated that Ultrasound sensor measurement was not influenced much by the ambient light. Also, temperature and humidity correction did not have any significant effect on Ultrasound sensor data. This, however, should be verified with further investigation of sensor HTS 221.

Laser sensor TOF 10120 performed well for obstacle distance measurement on Flat surfaces till 0.55 % of its full range and 0.3% of full range on curved object surfaces. It was shown that ambient light did not have much influence on Laser sensor measurement. The impact of object colour on Laser sensor measurement was also demonstrated.

Estimate of obstacle distance was achieved with data fusion between Ultrasound and Laser sensor data. Among two sensors, Ultrasound sensor(HC-SR04) had better accuracy than Laser sensor(TOF10120) in my experiments at section 8.2.10. By use of data fusion, it was demonstrated that more weight is given to sensor with better accuracy. Average error from Laser sensor was reduced by factor of 2.0 in data fusion estimate. This observation can be different for other kind of sensors available in market.

Obstacles were detected through live feed of PiCamera and if the obstacle was within a pre-defined distance, then PiCar was steered away from the obstacle. Though movement and speed of the PiCar is restricted due to current hardware setup, deviation from obstacles was demonstrated with PiCar. By using battery model UM 18650X2, the PiCar was able to operate continuously for 20 minutes without additional hardware described at section 4. With the first algorithm, there was a possibility of PiCar crashing into obstacles unclassified by the object detection model. So, it was later modified such that the PiCar steers away from nearby obstacles independent of whether these obstacles were classified by object detection model or not. In such case, object detection information can be used to assist the autopilot in taking actions or to map overall survey environment.

All the experiments were conducted in a small scale Lab setup to demonstrate the concept of surveillance system using the capability of PiCar. The demonstrated range and speed are not sufficient for a real survey vessel in a lake and will have to be improved as discussed in section 9.3.1.

References

[1] Lecture 1, IN3050 course at UiO,Spring 2020

[2] Title : Directive 2000/60/EC of the European Parliament and of the Council of 23 October 2000 establishing a framework for Community action in the field of water policy, year : 2000,
https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32000L0060, last access on January, 2021

[3] MastersThesis: Author : B. S. Søvegjarto, Title : Development of a fullsize low price Automatic Survey Vessel (ASV)for hydroacoustic work, School : University i Oslo, Year : 2015

[4] MastersThesis: Author : Halvard Yri Adriaenssens, Title : Automatisk utsetting og innhenting av sensorer i innsjø, School : University i Oslo, Year : 2020

[5] Authors : Daniel F. Carlson, Alexander Fürsterling, Lasse Vesterled, Mathias Skovby, Simon Sejer Pedersen,Claus Melvad, Søren Rysgaard, Title : An affordable and portable autonomous surface vehicle with obstacle avoidance for coastal ocean monitoring, article from Science Direct,URL: https://www.sciencedirect.com/science/article/pii/S2468067219300161?via%3Dihub, Year : 2019

[6] https://www.ibm.com/cloud/learn/machine-learning

[7] https://en.wikipedia.org/wiki/Sobel_operator

[8] http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

[9] https://www.adeept.com/adeept-mars-rover-picar-b-wifi-smart-robot-car-kit-for-raspberry-pi-4-3-model-b-b-2b-speech-recognition-opencv-target-tracking-stem-kit_p0117.html,

[10] https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a, Author : Hale, Jeff, found on March 2020

[11] https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

[12] https://opencv.org/about/, found on September, 2020

[13] Author:Fraden Jacob, Book:Handbook of modern sensors, Chapter 7.1. Pearson, 2007, Last access October, 2019.

[14] Authors: IRiccardo Polvara, Sanjay Sharma, Jian Wan, Andrew Manning and Robert Sutton, Title:Obstacle Avoidance Approaches for Autonomous Navigation of Unmanned Surface Vehicles, Year : 2017, URL:https://doi.org/10.1017/S0373463317000753 found on October 2019,last accessed on April 2021

[15] Authors:Jun Jo, Yukito Tsunoda, Bela Stantic and Alan Wee-Chung Liew, Title:A Likelihood-Based Data Fusion Model for the Integration of Multiple Sensor Data: A Case Study with Vision and Lidar Sensors, Year : 2017,URL: https://doi.org/10.1007/978-3-319-31293-4_39 found on October 2019,last accessed on April 2021

[16] https://www.instructables.com/Raspberry-Pi-Object-Detection/.

[17] https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

[18] https://components101.com/sensors/ultrasonic-sensor-working-pinout-datasheet

[19] https://machinethink.net/blog/object-detection/

[20] https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi.Listed with Apache 2.0 open source licence.

[21] https://github.com/armaanpriyadarshan/Training-a-Custom-TensorFlow-2.X-Object-Detector.

[22] https://openmv.io/collections/cams

[23] https://thepihut.com/products/raspberry-pi-night-vision-camera

[24] https://www.raspberrypi.org/products/pi-noir-camera-v2/

[25] https://www.raspberrypi.org/products/camera-module-v2/

[26] https://www.banggood.com/

[27] https://www.flir.ca/support-center/oem/what-is-the-difference-between-active-ir-and-thermal-imaging/#:~:text=Active%20IR%20systems%20use%20short,or%20long%20wavelength%20IR%20energy.

[28] https://www.rp-photonics.com/vertical_cavity_surface_emitting_lasers.html, found on April 2020

[29] https://www.instructables.com/Add-6-Ultrasonic-Distance-Sensors-to-Existing-Rasp/

[30] https://www.fierceelectronics.com/components/three-sensor-types-drive-autonomous-vehicles

[31] https://create.arduino.cc/projecthub/ByronSpars/autonomous-car-bc10ac

[32] https://create.arduino.cc/projecthub/hannahmcneelyy/arduino-uno-autonomous-car-c45fd1

[33] http://emaraic.com/blog/distance-measurement

[34] https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/

[35] https://github.com/paul-pias/Object-Detection-and-Distance-Measurement

[36] https://www.instructables.com/Self-Driving-Car-Using-Arduinoautonomous-Guided-Ve/

[37] https://www.youtube.com/watch?v=1n_KjpMfVT0

[38] https://www.adeept.com/

[39] https://roboticsbackend.com/raspberry-pi-arduino-serial-communicatio n/

[40] https://store.arduino.cc/arduino-uno-rev3

[41] https://store.arduino.cc/arduino-nano-33-ble-sense

[42] https://colab.research.google.com/github/trekhleb/machine-learning-e xperiments/blob/master/experiments/objects_detection_ssdlite_mobilenet _v2/objects_detection_ssdlite_mobilenet_v2.ipynb, found on January 2021

[43] Authors: Stateczny A, Kazimierski W, Gronska-Sledz D, Motyl W.,Title: The Empirical Application of Automotive 3D Radar Sensor for Target Detection for an Autonomous Surface Vehicle's Navigation., Year: 2019,URL: https://doi.org/10.3390/rs111011 56, Last access on December 2019.

[44] https://pypi.org/project/tensorflow/#history

[45] https://github.com/tensorflow/models/blob/master/research/object_detec tion/g3doc/tf2_detection_zoo.md, found on January 2021

[46] https://la3za.blogspot.com/2014/05/temperature-compensation-for-ardu ino.html

[47] Authors: Bucksch, Alexander and Lindenbergh, Roderik and Ree, J. Title:Error budget of terrestrial laser scanning : Influence of the intensity remission on the scan quality . III International Scientific Congress, Year : 2007, URL:https://www.researchgate.net /publication_ERRO/38144496, Last access on November 2020

[48] Authors: I.Hajnesk,K.Papathanassiou.,Title:Tutorial on SAR Polarimetry-Rough Surface Scattering Models, Year : 2005, URL: https://earth.esa.int/documents/653194/6 56796/Rough_Surface_Scattering_Models.pdf, Last access on February 2021.

[49] https://nerdyelectronics.com/embedded-systems/sensors/how-to-improve -readings-of-ultrasonic-sensor-temperature-and-humidity-compensation/

[50] https://emanual.robotis.com/docs/en/platform/turtlebot3/features/#spe cifications, found on November 2020

[51] https://www.ros.org/

[52] https://emanual.robotis.com/docs/en/platform/turtlebot3/features/#com ponents

[53] Authors: Thale, Sumegh, Prabhu, Mihir, Thakur, Pranjali, Kadam, Pratik., Title: ROS based SLAM implementation for Autonomous navigation using Turtlebot, Year: 2020, URL: https://www.researchgate.net/publication/343284410, Last access on December 2020

[54] Authors: Ana Almer Casino, Miguel Ángel Sempere Vicente, Title: Autonomous Medical Robot, Year: June 2020, URL: https://www.diva-portal.org/smash/get/diva2:1446226/FULLTEXT01.pdf, Last access on December 2020

[55] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md

[56] https://www.storkz.com/vge-dt03.html?currency=NOK&country=NO&gclid=EAIaIQobChMIu4XRhtTw7wIV1hV7Ch3BOgM-EAQYESABEgIpSfD_BwE

[57] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[58] UiO course FYS 3240,Spring 2021,lecture10,Data fusion and estimation,slide 13

[59] https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html

[60] https://www.raspberrypi.org/documentation/hardware/camera/

[61] https://dronebotworkshop.com

[62] https://www.st.com/resource/en/datasheet/hts221.pdf

[63] https://forum.arduino.cc/t/sensor-accuracy/615239

[64] Authors:Abdelmoghit Zaarane, Ibtissam Slimani, Wahban Al Okaishi, Issam Atouf, Abdellatif Hamdoun, Title:Distance measurement system for autonomous vehicles using stereo camera, School: LTI Lab, Department of Physics, Faculty of Sciences Ben M'Sik, University Hassan II Of Casablanca, Morocco, Year: January 2020, URL: https://www.sciencedirect.com/science/article/pii/S2590005620300011, Last access on May 2021.

[65] Image by author of the thesis-Jeyalakshmi Thoppe Subramanian, Year 2020-21, Digital Photograph, type PNG.

Appendix

Python Source code for surveillance system

Source code for surveillance system in Python is given below:
It comprises code for the following:
Object detection
Distance measurement with Ulrasound (HC-SR04) sensor
Temperature and humidity reading from ArduinoNano33BLE sensor
Distance measurement from Laser sensorTOF10120 from ArduinoUno
Data fusion between both sensors
Assessment of object distance
Deviation of PiCar

```
#!/usr/bin/env python3
#########
## Object Detection with pre trained ssdlite mobilenet v2 coco
# model using tensorflow1.Program is made with from
#references as following:
# https://github.com/EdjeElectronics/TensorFlow-Object-Detection-
#on-the-Raspberry-Pi and from thereof
#https://github.com/tensorflow/models/blob/master/research/
#object_detection/object_detection_tutorial.ipynb
#https://github.com/datitran/object_detector_app/blob/master/
#object_detection_app.py

import os
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
import argparse
import sys
from multiprocessing import Process
import serial
import RPi.GPIO as GPIO
import time
import Adafruit_PCA9685

turn_right_max  = 250 #front wheel right turn. 100 full right
#extreme
turn_left_max   = 450 # #setting 500 stalls
# but 100 to 560 is specified as full range
turn_middle = 350 # 460 in middle. 350 in actual.
```

```
status      = 1            #Motor rotation
forward     = 0           #Motor forward
backward    = 1            #Motor backward
left_spd  =100 # due to Picar weight, cannot set less as giben in
#manual
right_spd=100
speed=100 # duty cycle 100%
spd_ad_1 = 1 # tried setting 0.5,i.e. half speed at middle
#direction,
# with 0.5, picar not moving, so setting 1
spd_ad_2 = 1

pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(50)

def right():
    pwm.set_pwm(2, 0, turn_right_max)

def left():
    pwm.set_pwm(2, 0, turn_left_max)

def middle():
    pwm.set_pwm(2, 0, turn_middle)
####
# pin detail in GPIO BOARD mode
Motor_A_EN      = 7
Motor_B_EN      = 11

Motor_A_Pin1    = 8
Motor_A_Pin2    = 10
Motor_B_Pin1    = 13
Motor_B_Pin2    = 12

pwm_A = 0
pwm_B = 0

def motorStop():
    GPIO.output(Motor_A_Pin1, GPIO.LOW)
    GPIO.output(Motor_A_Pin2, GPIO.LOW)
    GPIO.output(Motor_B_Pin1, GPIO.LOW)
    GPIO.output(Motor_B_Pin2, GPIO.LOW)
    GPIO.output(Motor_A_EN, GPIO.LOW)
    GPIO.output(Motor_B_EN, GPIO.LOW)

def setup():#Motor initialization
    global pwm_A, pwm_B
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(Motor_A_EN, GPIO.OUT)
    GPIO.setup(Motor_B_EN, GPIO.OUT)
    GPIO.setup(Motor_A_Pin1, GPIO.OUT)
    GPIO.setup(Motor_A_Pin2, GPIO.OUT)
    GPIO.setup(Motor_B_Pin1, GPIO.OUT)
    GPIO.setup(Motor_B_Pin2, GPIO.OUT)
    motorStop()
    pwm_A = GPIO.PWM(Motor_A_EN, 1000)
    pwm_B = GPIO.PWM(Motor_B_EN, 1000)


def motor_right(status, direction, speed):
    global  pwm_B
    if status == 0: # stop
        motorStop()
    else:
        if direction == forward:
            GPIO.output(Motor_B_Pin1, GPIO.HIGH)
            GPIO.output(Motor_B_Pin2, GPIO.LOW)
            pwm_B.start(100)
            pwm_B.ChangeDutyCycle(speed)
        elif direction == backward:
            GPIO.output(Motor_B_Pin1, GPIO.LOW)
            GPIO.output(Motor_B_Pin2, GPIO.HIGH)
            pwm_B.start(100)
            pwm_B.ChangeDutyCycle(speed)
def motor_left(status, direction, speed):
    global pwm_A
    if status == 0: # stop
        motorStop()
    else:
        if direction == forward:#
            GPIO.output(Motor_A_Pin1, GPIO.HIGH)
            GPIO.output(Motor_A_Pin2, GPIO.LOW)
            pwm_A.start(100)
            pwm_A.ChangeDutyCycle(speed)
        elif direction == backward:
            GPIO.output(Motor_A_Pin1, GPIO.LOW)
            GPIO.output(Motor_A_Pin2, GPIO.HIGH)
            pwm_A.start(100)
            pwm_A.ChangeDutyCycle(speed)
    return direction

def destroy():
    motorStop()
    GPIO.cleanup()# Release resource

try:
    pass
except KeyboardInterrupt:
    destroy()

## to add ultrasonic module

TRIG=23 # 11 BCM, 23 BOARD
ECHO=24  # 8 BCM, 24 BOARD
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

GPIO.setup(TRIG,GPIO.OUT)
GPIO.output(TRIG,0)
GPIO.setup(ECHO, GPIO.IN)

## til this
time.sleep(0.1)

# camera constants
IM_WIDTH = 1280
IM_HEIGHT = 720
# less resolution , little faster framerate
#IM_WIDTH = 640
#IM_HEIGHT = 480

camera_type = 'picamera'
parser = argparse.ArgumentParser()
args = parser.parse_args()
#path to working directory as the object_detection folder.
sys.path.append('..')
# Import utilites
from utils import label_map_util
from utils import visualization_utils as vis_util

#  object detection module chosen
MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'

#  get current working directory
CWD_PATH = os.getcwd()

# Path to frozen inference graph of detection trained model.
#THis graph contains the trained model info-
#ssdlite_mobilenet_v2_coco
PATH_TO_CKPT =
os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')
# Path to label map file
#label map of 90 obejcts of ssdlite_mobilenet_v2_coco
PATH_TO_LABELS =
os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')

# Number of classes or objects the object detector can identify
NUM_CLASSES = 90
## Load the label map.
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
# a index number corresponds to  specific object class
#Example index no 5 to airplane, 47 to coffee cup, 84 to book etc

category_index = label_map_util.create_category_index(categories)
# 'initlise and Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)
# Define input and output tensors
# Input tensor is the image
image_tensor detection_graph.get_tensor_by_name('image_tensor:0')
# Output tensors are the detection boxes, scores, and classes
# Each box shows a part of the image where a particular object
#was detected
detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represents level of confidence for each of the
#objects.
# The score is shown on the result image, together with the class
#label.
detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()
font = cv2.FONT_HERSHEY_SIMPLEX
```

```
stddevUS=2.66
stddevlaser=10.15
totalstddev=stddevUS**2+stddevlaser**2
setup()## added by jeya 09.03.20
## starting motor
print("motor␣running")
middle()
motor_left(status, forward,left_spd*spd_ad_1)
motor_right(status,backward,right_spd*spd_ad_1)

### Picamera ###

camera = PiCamera()
camera.resolution = (IM_WIDTH,IM_HEIGHT)
camera.framerate = 10
rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
rawCapture.truncate(0)
for frame1 in camera.capture_continuous(rawCapture,
format="bgr",use_video_port=True):
    t1 = cv2.getTickCount()
    # get frame and expand frame dimensions to have shape: [1,
    None, None, 3]
    # i.e.a single-column array,where each item in column has
    #the pixel RGB value
    frame = np.copy(frame1.array)
    frame.setflags(write=1)
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_expanded = np.expand_dims(frame_rgb, axis=0)

    # Perform the actual detection by running the model with the
    #image as input
    (boxes, scores, classes, num) = sess.run(
        [detection_boxes, detection_scores, detection_classes,
        num_detections],
        feed_dict={image_tensor: frame_expanded})

    # visulaization of the results
    vis_util.visualize_boxes_and_labels_on_image_array(
        frame,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8,
        min_score_thresh=0.40)
```

```
cv2.putText(frame,"FPS:␣{0:.2f}".format(frame_rate_calc),
(30,50),font,1,(255,255,0),2,cv2.LINE_AA)
# display result.
cv2.imshow('Object␣detector', frame)
#connecting arduino uno and nano to raspberry pi
ser0 = serial.Serial('/dev/ttyACM0', 9600, timeout=1)#arduino
#for laser
ser1 = serial.Serial('/dev/ttyACM1', 9600, timeout=1)#arduino
#for temp , humidity
ser0.flush()
# only keeping this gives better distance close to
#US # arduino

for index,value in enumerate(classes[0]):
    # mute to measure object distance independent of
    #object detection
    if scores[0,index] > 0.5:
        # finding a particular object we can find any or all
        #objects in 90 classess
        if category_index.get(value)['name']=='cup':  #cup
            print("cup␣found")
            #mute till this - independent distancemeasurement
            time.sleep(0.5)
            #measuring distance only
            #after the object is found
            print("starting␣to␣measure␣distance")
            ser0.flush()    # arduino

            #ultrasoundmeas() #cannot call a fucntion outside
            #this main program .
            #Otherwise it hangs . need to define the function
            #inside itself

            #mesuring distance from Ultrasound sensor
            GPIO.output(TRIG,1)
            time.sleep(0.00001)
            GPIO.output(TRIG,0)
            while (GPIO.input(ECHO)==0):
                pass
            starttime=time.time()

            while (GPIO.input(ECHO)==1):
                pass
            stoptime=time.time()
            tim=stoptime-starttime
            # tempearutre,humidity from ARduino nano BLE 33
            #board
            # Board connectd to Serial 1 of Raspberry pi of
```

```
#Picar
line1 = ser1.readline().decode('utf-8').rstrip()
list=line1.split(',')
#print(list)
temp=float(list[0])
#print("temperature",temp) # temp in celsius)
humidity =float(list[1])
#print("humidity(%)",humidity)
#return temp,humidity
#temp,humid=arduinoBLEread()
print()
# sound velocity after tempearure and himidity
# correction
soundvel=331.4+(0.6*temp)+(0.01243*humidity)
dist_US1=(soundvel/2)*100*(stoptime-starttime)*10
dist_US2=dist_US1 # if correction for Ultrasound
#sensor position in Picar needed then -10
#dist_US1=round((170*100*(stoptime-starttime)*10),
#0)
#dist_US=dist_US1
dist_US=float("{:.2f}".format(dist_US2))
#print("distanceof obstacle from ultrasoundsensor
#mm",dist_US)

#####mesuring distance from Ultrasound sensor

##### measuring distance from laser sensor
# Laser sensor connected to ARduino Uno
# arduino connected to Serial 0 Raspberry Pi of
#Picar
time.sleep(0.5)
if ser0.in_waiting > 0:
    line=ser0.readline().decode('utf-8').rstrip()
    distance_laser=float(line)
    #print("distance from laser in
    #mm",distance_laser)

else:
    distance_laser= 0
####### measuring distance from laser sensor
### datafusion from Ultrasound and laser

datafusion_op1=(((stddevlaser**2)/totalstddev)*
dist_US)+(((stddevUS**2)/
totalstddev)*distance_laser)
datafusion_op=
float("{:.2f}".format(datafusion_op1))
print()
```

```
                    #print("distance after data fusion form both
                    #sensor",datafusion_op)
                    if distance_laser <= 200.0: #use laser sensor op
                    #if dist_US<= 200.0: #use Ultrasound sensor op
                    #if datafusion_op<= 200.0: # use datafusionop

                         # deviating from obstacle
                         #print("obstacle distance<200mm, deviating to
                         #left")
                         left()
                         motor_left(status,forward,left_spd*spd_ad_2)
                         motor_right(status,backward,
                         right_spd*spd_ad_2)
                         time.sleep(0.5)# 2

                    else:
                         #print("obstacle distance >200mm, turning
                         #straight")
                         middle()

     ##correct expression
     dislabel=[category_index.get(value)['name'] for index,value
     in enumerate(classes[0]) if scores[0,index] > 0.5]
     #print(dislabel)
     t2 = cv2.getTickCount()
     time1 = (t2-t1)/freq
     frame_rate_calc = 1/time1

      # Press 'q' to quit
     if cv2.waitKey(1) == ord('q'):
          break
          motorStop()
     rawCapture.truncate(0)

camera.close()
GPIO.cleanup()
cv2.destroyAllWindows()
```

Python code for saving only object details to text file

A python program is developed to do only object detection. The extra window showing object detail is suppressed. The detected object name along with timestamp is stored in a list. This list is written to a text file when program stops. At present the program is stopped after 20 seconds after start. This setting can be changed.

```
#Import necessary packages
CWD_PATH = os.getcwd()
# output file where objects found are stored with timestamp
```

```
PATH_TO_file = os.path.join(CWD_PATH,"objectsfound.txt")
file2=open(PATH_TO_file,"a+")
# iniltialise list
lis=[]
listime=[]
start_time=float("{:.0f}".format(time.time()))

for frame1 in camera.capture_continuous
(rawCapture, format="bgr",use_video_port=True):
    # perform Obejct detection refer code
    #from sureveillance system program
    print("object identification starts ",start_time)
    listime.append(str(datetime.now()))
    #get detected object name
    dislabel=str([category_index.get(value)['name']
     #continued next line
    for index,value in enumerate(classes[0])
    if scores[0,index] > 0.5])
    #get timestamp
    timevalue=str(datetime.now())
    # combine both
    fullvalue=timevalue+dislabel
    # append to list
    lis.append(fullvalue)

    # to break from object detection ,
    stop_time=float("{:.0f}".format(time.time()))
    # process time  is approx. 25 sec from start time
    # stopping the program after approx. 20 seconds
    #total 45 seconds
    if stop_time >= start_time + 45:
        print(stop_time)
        break
camera.close()
cv2.destroyAllWindows()
file2.write(str(lis))
file2.close()
```

Arduino sketch for Nano BLE 33 Sense

```
/*
  HTS221 - Read Sensors

  This example reads data from the on-board HTS221 sensor of the
  Nano 33 BLE Sense and prints the temperature and humidity sensor
  values to the Serial Monitor once a second.

  The circuit:
  - Arduino Nano 33 BLE Sense

  This example code is in the public domain.
*/

#include <Arduino_HTS221.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!HTS.begin()) {
    Serial.println("Initialisation␣Failed!");
    while (1);
  }
}

void loop() {
  // read all the sensor values
  float temperature = HTS.readTemperature();
  float humidity    = HTS.readHumidity();

  // print each of the sensor values
  //Serial.print("Temperature=");
  Serial.print(temperature);
 // Serial.println("degC");
  Serial.print(",");
 // Serial.print("Humidity=");
  Serial.print(humidity);
  //Serial.println("%");

  // print an empty line
  Serial.println();

  // wait 1 second to print again
  delay(1000);
}
```

Arduino sketch for TOF10120

```
/*
    TOF10120 Distance Sensor Demonstration
    TOF10120-Demo.ino
    Demonstrates use of TOF10120 Distance Sensor
    Adapted from code from SurtrTech
    Displays results on Serial Monitor
    DroneBot  Workshop  2019
    https://dronebotworkshop.com
*/

#include  <Wire.h>
unsigned  char  ok_flag;
unsigned  char  fail_flag;
unsigned  short  lenth_val  =  0;
unsigned  char  i2c_rx_buf[16];
unsigned  char  dirsend_flag=0;
void  setup()  {
    Wire.begin();
    Serial.begin(9600,SERIAL_8N1);
      //printf_begin();
}
void  loop()  {
      int  x=ReadDistance();
      Serial.print(x);
      Serial.println();
      //Serial.println("mm");
}
int  serial_putc(  char  c,  struct  __file  *  )
{
    Serial.write(  c  );
    return  c;
}
void  printf_begin(void)
{
    fdevopen(  &serial_putc,  0  );
}
void  SensorRead(unsigned  char  addr,unsigned  char*  datbuf,
unsigned  char  cnt)
{
unsigned  short  result=0;
 //step1:instruct  sensor  to  read  echoes
Wire.beginTransmission(82);//transmit to device #82 (0x52)
//address specified in  the  datasheet  is  164  (0xa4)
//but i2c adressing uses the high 7  bits so  it's␣␣82
Wire.write(byte(addr));//sets␣distance␣data␣address(addr)
Wire.endTransmission();//stop␣transmitting
//step2:wait␣for␣readings␣to␣happen
delay(1);//datasheet␣suggests␣at␣least␣30uS
//step3:request␣reading␣from␣sensor
Wire.requestFrom(82,cnt);//request␣cnt␣bytes␣from␣slavedevice
//#82␣␣(0x52)
//step4:receive␣reading␣from␣sensor
if␣(cn␣␣<=␣␣Wire.available()){//if␣two␣bytes␣were␣received
*datbuf++=Wire.read();//receive␣high␣byte(overwrites␣␣previous␣␣reading)
*datbuf++=Wire.read();//receive␣low␣byte␣as␣lower␣8␣bits
}
}
int␣␣ReadDistance(){
SensorRead(0x00,i2c_rx_buf,2);
lenth_val=i2c_rx_buf[0];
lenth_val=lenth_val<<8;
lenth_val|=i2c_rx_buf[1];
delay(500);
return␣␣lenth_val;
}
```