A
PROJECT REPORT

on

OBJECT DETECTOR FOR VISUALLY IMPAIRED WITH DISTANCE CALCULATION FOR HUMANS

Submitted in partial fulfilment for the Award of Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

DIXITHA KASTURI (160116737003)
GOUTHAMI V (160116737005)

Under the guidance of

Ms. E. RAMALAKSHMI
Assistant Professor
Dept. of IT, CBIT.

DEPARTMENT OF INFORMATION TECHNOLOGY
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
(Affiliated to Osmania University; Accredited by NBA (AICTE) and NAAC (UGC),ISO Certified 9001:2015), Kokapet (V), GANDIPET(M), HYDERABAD – 500 075
Website: www.cbit.ac.in

2019-2020

CERTIFICATE

This is to certify that the project work entitled "**OBJECT DETECTOR FOR VISUALLY IMAPIRED WITH DISTANCE CALCULATION FOR HUMANS**" submitted by **DIXITHA KASTURI (160116737003)** and **GOUTHAMI V (160116737005)** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING** in **INFORMATION TECHNOLOGY** to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)**, affiliated to **OSMANIA UNIVERSITY**, Hyderabad, is a record of bonafide work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Project Guide
Ms. E. Ramalakshmi
Asst. Professor, Dept. of IT,
CBIT, Hyderabad.

Head of the Department
Dr. K. Radhika
Professor & Head, Dept. of IT,
CBIT, Hyderabad.

DECLARATION

We declare that the project report entitled "**OBJECT DETECTOR FOR VISUALLY IMPAIRED WITH DISTANCE CALCULATION FOR HUMANS**" is being submitted by us in the Department of Information Technology, Chaitanya Bharathi Institute of Technology (A), affiliated to Osmania University, Gandipet, Hyderabad, under the guidance of **Ms. E. Ramalakshmi**, Assistant Professor, Dept of IT, C.B.I.T.Hyderabad .

No part of the report is copied from books / journals and wherever the portion is taken, the same has been duly referred. The reported results are based on the project work done entirely by us and not copied from any other source.

Dixitha Kasturi (160116737003)

Gouthami V (160116737005)

ACKNOWLEDGEMENT

We would like to express our gratitude to our guide **Ms. E. Ramalakshmi,** Assistant Professor, Dept. of IT, C.B.I.T, for her kind co-operation and encouragement which help us in completion of this project.

We are highly indebted to **Dr. K. Radhika,** Head of Department, Information Technology and **Dr. P. Ravinder Reddy**, The Principal, C.B.I.T., Hyderabad and for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We have taken efforts in this project work. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

Our thanks and appreciations also go to the people who have willingly helped us out with their abilities.

Dixitha Kasturi (160116737003)

Gouthami V (160116737005)

OBJECT DETECTOR FOR VISUALLY IMPAIRED WITH DISTANCE CALCULATION FOR HUMANS

ABSTRACT

Object detection is a computer vision technique for locating instances of objects in videos. When we as humans look at images or videos, we can recognize and locate objects within a matter of moments. The goal of this project is to replicate the intelligence of human in doing that using Deep Neural Networks and Raspberry Pi and a camera. This model could be used for visually disabled people for improved navigation and crash free motion.

YOLO(You Only Look Once) is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. A Haar is just like a kernel in CNN where the kernel values are determined by training while in Haar they are determined manually. Haar is a cascading classifier Haar Classifier is used to enhance the human detection and calculate distance for humans. As the objects are identified they will be read out using a text-to-speech engine known as gTTS(google text-to-speech) and ,which stores the text in an mp3 file. The package known as Pygame will load and play the mp3 file dynamically as the objects are detected. This developed Deep Learning model is integrated with Raspberry Pi using OpenCV.

Technologies used

Software:

Language: Python

Libraries: OpenCV, gTTS, pygame

Hardware:

Raspberry Pi

Camera

TABLE OF CONTENTS

	Page No
Abstract	v
1. INTRODUCTION	1
1.1 Overview	1
1.2 Motivation	1
1.3 Applications	1
1.4 Problem Statement	2
1.5 Objective	3
1.6 Organization of Report	3
2. SYSTEM REQUIREMENT SPECIFICATION AND LITERATURE SURVEY	4
2.1 Hardware Requirements	4
2.2 Software Requirements	6
2.3 Literature Survey	7
3. METHODOLOGY	29
3.1 Existing System	29
3.2 Proposed System	30

3.3 Architecture	35
4. IMPLEMENTATION	37
4.1 RCNN implementation	37
4.2 YOLO implementation	38
4.3 OpenCV functions	39
4.4 Other functions	44
4.5 Distance calculation for Humans	45
4.6 Voice assistance	47
4.7 Integration with PI	47
5. RESULTS	49
6. CONCLUSION AND FUTURE SCOPE	53
BIBLIOGRAPHY	54
ANNEXURE 1	57

LIST OF FIGURES

fig 2.1 Raspberry Pi board 4

fig 2.2 Camera for Raspberry Pi board 5

fig 2.3 Dataset Representation 6

fig 2.4 Multistage Object Detection Network 8

fig 2.5 Proposed video surveillance pipeline 11

fig 2.6 Overview of Random Forest 14

fig 2.7 Extraction of plant objects 18

fig 2.8 Result of classification by random forest and SVM classifier respectively 18

fig 2.9 Distance measure 20

fig 2.10 Cow detection 21

fig 2.11 Proposed framework of human face detection cascade 27

fig 2.12 Results for test images 28

fig 3.1 Walking aide for the blind 29

fig 3.2 Obstacle detector sensors in cars 30

fig 3.3 Fast RCNN Working 32

fig 3.4 YOLO Working 34

fig 3.5 Design 35

fig 4.1 Haar Classifier 45

fig 5.1 Running Object detector on Laptop 49

fig 5.2 Connecting Raspberry Pi to laptop 50

fig 5.3 object detection running on Raspberry Pi viewed through laptop 50

fig 5.4 Object detection running on Raspberry Pi viewed through Tv monitor 51

fig 5.5 Distance measurement for humans along with detection 51

fig 5.6 Distance measurement for two humans along with detection 52

fig 5.7 Multiple Objects detected with distances for 2 humans 52

LIST OF TABLES

Table 2.1 YOLO vs RCNN's 24

1. INTRODUCTION

1.1 OVERVIEW

The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. With the availability of large amounts of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy. In this project we will explore Object detection and localization using convolutional networks and integrate it with Raspberry Pi and camera. The distance calculation in inches will be done for humans using a separate classifier which considers the contours of human face.

1.2 MOTIVATION

According to the survey conducted in 2011, about 1% of the human population is visually impaired (about 70 million people) and amongst them, about 10% are fully blind (about 7 million people) and 90% (about 63 million people) with low vision. The main problem with blind people is how to navigate their way to wherever they want to go. Such people need assistance from others with good eyesight. Accounting to the rapidly growing technology and shift towards automation specifically driverless cars, the need for an efficient and fast obstacle detector is prevalent .This motivated us to think about the solution of developing an obstacle detector which would not only help visually impaired people to navigate but also has numerous other applications in the real world.

1.3 APPLICATIONS

There are an endless number of applications for object detection. Few of the applications are explained below:

1. Tracking objects**:** Video tracking is the process of locating a moving object over time using a camera. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality
2. Video surveillance**:** Video cameras are used to observe an area, connected to a recording device or IP Network and monitored in a Control Room. Video surveillance technologies are used for far more than their roots in crime detection, with video analytics solutions able to serve a variety of applications such as traffic and crowd control management.
3. Pedestrian detection**:** Pedestrian detection is an essential and significant task in any intelligent video surveillance system, as it provides the fundamental information for semantic understanding of the video footages. It has an obvious extension to automotive applications due to the potential for improving safety systems.
4. Anomaly detection: In data mining, anomaly detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data.
5. People Counting**:** It is a technique that is used to measure the number of people traversing a certain passage or entrance. Examples include simple manual clickers, infrared beams, thermal imaging systems, WiFi trackers
6. Self driving cars**:** A self-driving car, also known as an autonomous car, driverless car, or robotic car, is a vehicle that is capable of sensing its environment and moving safely with little or no human input.
7. Face detection**:** Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene.

1.4 PROBLEM STATEMENT

With the world moving towards automation, there's a rising demand for an efficient Obstacle (Object) detector .There are no low cost object detectors available which are

efficient and most of the available obstacle detectors use ultrasonic sensor which doesn't take a broader frame into its line of sight. This project focuses on integrating Deep learning algorithms with Raspberry Pi and Camera to detect the type of object and its distance efficiently.

1.5 OBJECTIVE

The main objective of this project is to develop a model on raspberry using deep learning that efficiently detects the objects as they are captured by the camera and also calculate distance of humans from the camera. This model could be used by the blind for crash free motion.

1.6 ORGANIZATION OF PROJECT

The detailed description about the organization of report is given below.

Chapter 1 consists of Introduction to the project.

Chapter 2 consists of Literature survey.

Chapter 3 consists of System design.

Chapter 4 consists of detailed description of the Implementation.

Chapter 5 consists of the Results obtained

Chapter 6 consists of Conclusion and Future scope.

2. SYSTEM REQUIREMENT SPECIFICATION AND LITERATURE SURVEY

2.1 HARDWARE REQUIREMENTS

2.1.1 Raspberry Pi Board

The Raspberry Pi, as shown in fig 2.1, is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything a desktop computer can do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. The Raspberry Pi has the ability to interact with the outside world using camera and other sensors. It in sort is a computer on a chip.

fig 2.1 Raspberry Pi board

2.1.2 Camera for Raspberry Pi

The Raspberry Pi Camera Module v2,as shown in fig 2.2, replaced the original Camera Module in April 2016. The v2 Camera Module has a Sony IMX219 8-megapixel sensor

(compared to the 5-megapixel OmniVision OV5647 sensor of the original camera).The Camera Module can be used to take high-definition video, as well as stills photographs

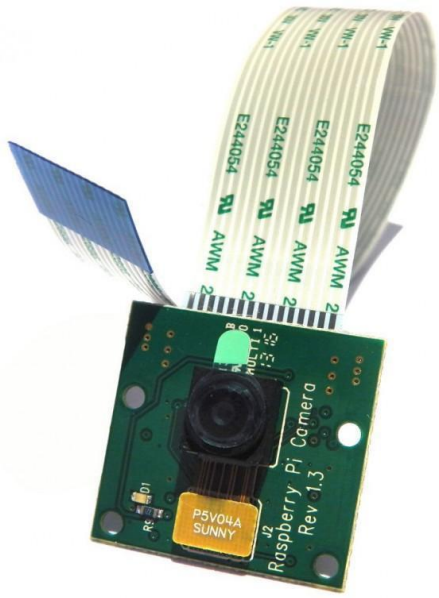

fig 2.2 Camera for Raspberry Pi board

2.1.3 Speaker

The Raspberry Pi has two audio output modes: HDMI and headphone jack. If your HDMI monitor or TV has built-in speakers, the audio can be played over the HDMI cable, but you can switch it to a set of headphones or other speakers plugged into the headphone jack.

2.2 SOFTWARE REQUIREMENTS

2.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database.The video being captured is divided into frames and the images are processed for object detection.

2.2.2 Python

Python is the language used for machine learning, ie., building a fast RCNN and to program the Raspberry Pi board to capture the videos for object detection.

2.2.3 DataSet

COCO is large scale images with Common Objects in Context(COCO) for object detection, segmentation, and captioning dataset. COCO has 1.5 million object instances for 80 object categories. COCO stores annotations in a JSON file. Sample of objects is shown in fig 2.3

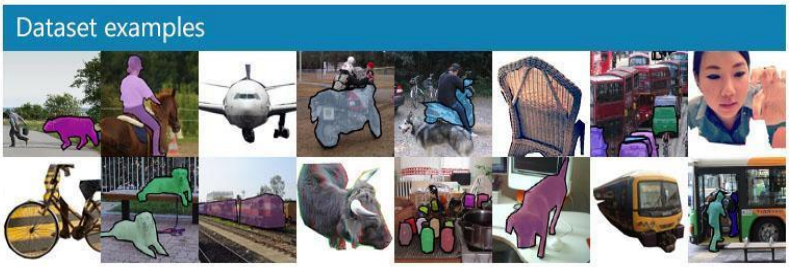

fig 2.3 Dataset Representation

2.3 LITERATURE SURVEY

2.3.1 Survey Paper 1

Title

Multistage Object Detection with Group Recursive Learning

Authors and Year of Publication

Authors**:** Jianan Li , Xiaodan Liang, Jianshu Li , Yunchao Wei , Tingfa Xu , Jiashi Feng, and Shuicheng Yan.

Year of Publication: 2018

Problem Statement

Most existing detection algorithms treat object proposals independently and predict bounding box locations and classification scores over them separately. The main aim is to create a new EM-like group recursive learning approach to iteratively refine object proposals by incorporating such contexts of surrounding proposals and provide an optimal spatial configuration of object detections.

Methodology/ Algorithm

The proposed object detection architecture consists of a cascade of multiple CNN networks, each of which focuses on a specific task, i.e., weakly-supervised semantic segmentation, proposal generation and recursive detection refinement respectively. The three networks share convolutional features learned from the entire image. The input image first passes through several convolutional and max pooling layers to produce convolutional feature maps. Then the semantic segmentation network learns semantic segmentation features for the entire image from the convolutional feature maps. The produced features are then fed into the proposal generation network to generate candidate object proposals. Finally, the recursive detection network iteratively refines the scores and locations of generated object proposals via a group recursive learning strategy.

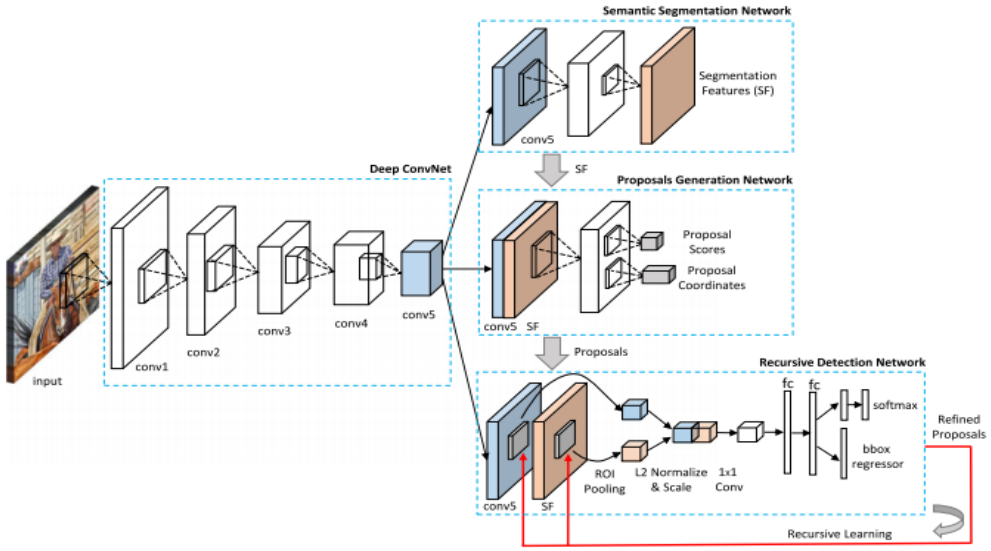

fig 2.4 Multistage Object Detection Network

Specifically, the ground truth bounding boxes of an image are projected on the last hidden layer of the Fully Convolutional Network as shown in fig 2.4. The "pixels" inside the projected boxes are labelled as foreground while the rest are labelled as background. This process is performed independently for each class. After the Fully Convolutional Network has been trained on the class-specific foreground segmentation task, the last classification layer is dropped and the convolutional feature maps are extracted to the output by the last convolutional layer as semantic segmentation features for the input images. The extracted semantic segmentation features consist of 512 channels and are of the same resolution with the last shared convolutional feature maps of the input image. It can be seen that the extracted features carry significant semantic segmentation information, which is enough to separate the object from its background.

Datasets

Two datasets have been used to develop this model. The two datasets consist of 9,963 and 22,531 images respectively, and they are divided into training, validation and test subsets. The model evaluated on PASCAL VOC 2007 is trained based on the train-val split from PASCAL VOC 2007, including 5,011 images, and the trainval split from

PASCAL VOC 2012, including 11,540 images. The model evaluated on PASCAL VOC 2012 is trained based on all images from PASCAL VOC 2007, including 9,963 images, and the trainval split from PASCAL VOC 2012. Standard evaluation metrics such as Average Precision (AP) and mean of AP (mAP) following the PASCAL challenge protocols have been used for evaluation.

Results

Given an input image, the proposed framework first generates initial object proposals using the proposal generation network and then recursively passes them into the recursive detection network. At the t-th iteration, the recursive detection network predicts the category-level confidences pt and bounding-box regression offsets rt for each proposal. The category of the proposal is predicted as the class with the maximum score in pt. For the proposals predicted as a specific object class, the locations of the proposals are updated by refining the previous location lt−1 with the predicted bounding-box regression offsets rt,g and then performing the group confidence pooling scheme as previously mentioned. For the proposals predicted as the background class, the locations of the proposals are not updated. The final outputs for each proposal are the results in the last iteration t = T, including the predicted category-level confidences pT and the refined locations lT

Limitations

The major limitation or drawback to while using a standard convolutional network followed by a fully connected layer is that, the length of the output layer is variable not constant, this is because the number of occurrences of the objects of interest is not fixed. A naive approach to solve this problem would be to take different regions of interest from the image, and use a CNN to classify the presence of the object within that region. The problem with this approach is that the objects of interest might have different spatial locations within the image and different ratios. Hence, you would have to select a huge number of regions and that would result in a large computational time and complexity which further accounts to less accuracy.

2.3.2 Survey Paper 2

Title

New Object Detection, Tracking, and Recognition Approaches for Video Surveillance Over Camera Network

Authors and Year of Publication

Authors: Shuai Zhang (Member, IEEE), Chong Wang (Member, IEEE), Shing-Chow Chan (Member, IEEE), Xiguang Wei, and Check-Hei Ho

Year of Publication: 2015

Problem Statement

Multi Camera Surveillance includes two fundamental tasks: Object detection and tracking. The main challenge of surveillance is to achieve these tasks in a nonoverlapping multiple camera network.

Methodology/Algorithm

The conventional K-means clustering method is used to cluster the segments obtained by MS into classes with the predefined object number. It can be seen that the occluded object is successfully segmented into two independent objects. Please note that the proposed object detection algorithm can be used for both stereo and monocular surveillance cameras. In contrast, the occluded objects cannot be separated by using monocular camera alone since the depth is unavailable. The methodology of video surveillance is shown in fig 2.5 and explained below.

The probabilistic object tracking problem can be modelled by the discrete-time linear state-space model as follows:

$$x_k = A_k x_{k-1} + w_k ,$$

$$z_k = C_k x_k + v_k,$$

where xk and zk denote respectively the state and observation (measurement) vectors at time k. Ak denotes the state transition matrix and Ck constitute the observation

model which relates the measurement with the state. wk and vk denote respectively process and observation noise vectors, and are assumed to be mutually independent.

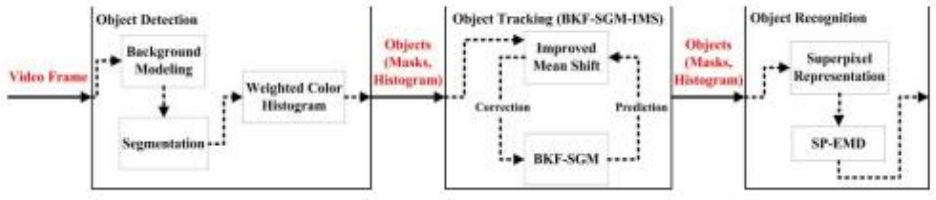

fig 2.5 Proposed video surveillance pipeline

Dataset

In the video surveillance dataset, four different objects cross both cameras 1 and 2 and five different objects cross both cameras 3 and 4. All the objects are tracked independently in the stereo cameras. The objective of our recognition algorithm is to identify the interested object from one view to another. The angle between cameras 3 and 4 is 30 degrees and it suffers from

serious lighting change. The angle between cameras 1 and 2 is 90 degrees. Therefore, object appearance will change even more than views 3 and 4.

Results

The performance of the proposed method is evaluated and compared with other methods in terms of correct object recognition rate and processing times. It was noticed that the method has the highest recognition rate in both tests. The high recognition rate is due mainly to the local representation of the appearance of the object. On the contrary, histogram based methods cannot fully utilize the shape and edge information and they model the appearance as global representation. Compared to other methods, our algorithm is very efficient. On average, it only takes 0.08 second to process one object. Therefore, it is practical to be used in real-time video surveillance system after appropriate acceleration using multiple thread processing or GPU.

Limitations and Drawbacks

The clustering method mainly focuses on dividing the image and pre-processing it to generate sub clusters to extract important features from the image. The location of the object and contrast is considered but the actual type of object is not. The problem arises when multiple objects have to be detected. The time for pre-processing is high. When relative motion is considered ,the time lost due to slow pre processing and clustering might lead to inaccurate and slow detection of objects.

2.3.3 Survey Paper 3

Title

Salient Object Detection via Random Forest

Authors and Year of Publication

Authors: Shuze Du and Shifeng Chen

Year of Publication: 2014

Problem Statement

Salient object detection plays an important role in image pre-processing. Existing approaches often neglect the contours of salient objects, thus resulting in inaccurate detection for large objects. Besides, they mainly focus on detecting only a single object. Salient object from the view of the object contour needs to be detected and to measure patch rarities and compute similarities among patches random forest should be exploited.

Methodology/Algorithm

As shown in fig 2.6, the input image is taken and rarity analysis outputs a global saliency map by using the random forest is generated. Based on the rarity map, the rough contour of the salient object is extracted. A local saliency map, which represents the similarity between an outer patch and all inner patches of the contour, and the dissimilarity between an inner patch and all outer patches, is computed.

The final map is obtained by refining the local map with graph-cut based segmentation.

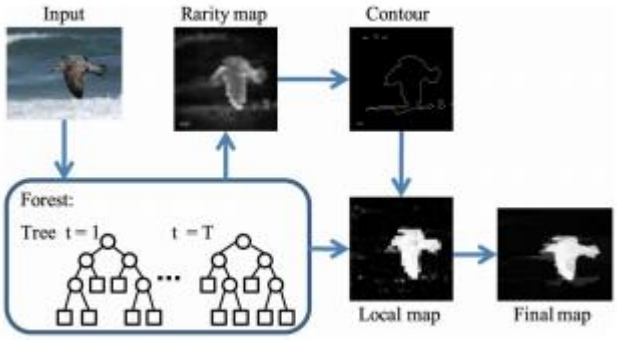

fig 2.6 Overview of Random Forest

As the approach is patch-based, the detection result at object boundary is not accurate. We use image segmentation to alleviate this issue. The final saliency map is obtained by averaging saliency values within image regions which are generated by segmenting the original color image using graph-cut. This step can effectively remove noise or remedy the false detection.

In this manner, the results can be further refined.

Dataset

Two datasets have been used where one was used to train the model to detect one object and the other was used to train to detect two objects in a frame. The model detects the two objects well, while the other methods only detect one object or highlight the objects non-uniformly, i.e., pay more attention to one object than the other. This validates that the other methods work well on the single salient object dataset but worse on multiple objects dataset, while this model achieves good performance on both sets.

Results

In this work, we introduce a novel method to evaluate the saliency of an image based on the rarities of patches and contour-based contrast analysis. Our model achieves currently

higher performance on two datasets compared to various state-of-the-art works.

Limitations and Drawbacks

i. Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decisions on the majority of votes.

ii. Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.

2.3.4 Survey paper 4

Title

Raspberry Pi as Visual Sensor Nodes in Precision Agriculture: A Study

Authors and Year of Publication

Authors: Radhika Kamath, Mamatha Balachandra (Member, IEEE), and Srikanth Prabhu(Member, IEEE)

Year of Publication: 2019

Problem Statement

To use wireless sensor network and classifier to sense the important agricultural field parameters, such as temperature, humidity, soil moisture level, nitrite content in the soil, groundwater quality, and so on. These sensed parameters will be sent to a remote station, where it will be processed and analysed to build a decision support system.

Methodology/Algorithm

The huge success of the Raspberry Pi boards led to the development of Raspberry Pi camera module v1 to be used together with the Raspberry Pi boards. The Raspberry Pi camera is a low-power high-definition small camera that comes with a flat flexible cable which is to be connected into the CSI (Camera Serial Interface) connector. In the year 2016, the camera module v2 was released. For both the iterations, there are visible light and infrared versions. Many programming libraries are available for image processing using the Raspberry Pi camera which can be used for various applications. In this research work, Python APIs have been used to acquire and process images from Raspberry Pi camera.

Two different algorithms were adopted to test the classifiers one was Random Forest and the other was SVM classifier. The system consisted of static nodes deployed at a fixed location. One of the key design issues in implementing wireless visual sensor network is the placement of the sensor nodes in the field and to decide the number of sensor nodes required. There are two different types of coverage areas in WVSNs

namely radio coverage area and sensing coverage area. The radio coverage area gives the distance covered by the communication technology used and the sensing coverage area is the area visible from the visual sensor or image sensor. It is not practical to cover the entire field by the visual sensors due to various factors like cost, the terrain of the land, and so on. This is perfectly fine when implementing wireless visual sensor network for monitoring crops for pests like weeds or diseases. Because weeds usually will be spread throughout the field. So it will be captured by at least one image sensor node and highly unlikely that weeds go unnoticed due to the uncovered regions. In this study, an attempt is made to keep a minimal number of uncovered regions. The field where the system was set-up was about 10m2.

The camera was mounted at a distance of 2m above the ground facing down towards the crops on the field. This resulted in a square sensing coverage area of about 2m X 2m approximately. A good coverage and deployment strategy is necessary for the optimum resource allocation in a WSN thereby reducing the overall cost of the network. Random deployment of visual sensor nodes results in some regions densely or sparsely covered by visual sensor nodes.

Shape features are one of the important features used for the discrimination of crop and weed. Individual leaves or the whole plant can be considered for the shape feature extraction. In this study, the whole plant was considered for shape feature extraction. As shown fig 2.7 Grass-type weed and paddy belong to the same family and hence their shape features also resemble to a great extent. Even sedges(a type of weed) also have close similarities with the paddy crop with respect to shape features. The shape of the plants keeps varying as they grow.

Random Forest Classifier has been used to do this. The tree-based supervised learning algorithm is considered to be one of the best as it provides high accuracy and maps the non-linear relationships effectively

fig 2.7 Extraction of plant objects

Dataset

The size of the training set is 606 and the size of the test data is about 182. Out of 182 plant objects, 133 belonged to paddy plants and 49 were weed plants.

Results

The result of the Random Forest classifier and SVM classifier is given in the fig 2.8 respectively. The misclassifications can be attributed to the similarity of grass-type weed and the paddy crop. It is difficult to compare the result of paddy crop and weed discrimination carried out in this study with other published work in the literature because each research work has been carried out for the different crops under different field conditions, different lab conditions and boundary conditions.

Predicted Class	Actual Class	
	Weed	Crop
Weed	37	27
Crop	12	106

Predicted Class	Actual Class	
	Weed	Crop
Weed	35	58
Crop	14	75

fig 2.8 Result of classification by random forest and SVM classifier respectively

Drawbacks and Limitations

i. Overfitting
ii. Large complexity
iii. Difficult to understand and interpret the final model

2.3.5 Survey Paper 5

Title

A Practical Animal Detection and Collision Avoidance System Using Computer Vision Technique

Authors and Year of Publication

Authors: Sachin Umesh Sharma, Dharmesh J. Shah

Year of Publication: 2017

Problem Statement

There are an endless number of accidents that take place on highways, everyday in India. Collision of an Animal with the vehicle is a big issue which leads to road accidents. To prevent that a low cost automatic animal detection system must be developed.

Methodology/Algorithm

Road accidents are increasing due to the increase in a number of vehicles day by day and also the due to the absence of any intelligent highway safety and alert system. According to the report given by the Society for Prevention of Cruelty to Animals (SPCA), around 270 cattle had been brought to their hospital-cum-animal-shelter in the year 2013, most of whom were accident victims. Applications built on detection of animals play a very vital role in providing solutions to various real-life problems. The base for most of the applications is the detection of animals in the video or image.

Animals can be detected using the knowledge of their motion. The fundamental assumption is that the default location is static and can simply be subtracted. Intelligent highway safety and driver assistance systems are very helpful to reduce the number of accidents that are happening due to vehicle-animal collisions. On Indian roads, two types of animals – the cow and the dog are found more often than other animals on the road. The primary focus of the proposed work is for detection of animals on roads which can have the potential application of preventing an animal-vehicle collision on highways. Specific objectives of the research work are:

i. To develop a low-cost automatic animal detection system in context to Indian roads.

ii. Finding the approximate distance of animal from the vehicle in which camera is mounted.

iii. To develop an alert system once the animal gets detected on the road which may help the driver in applying brakes or taking other necessary action for avoiding collision between vehicle and animal

A histogram of oriented gradients (HOG) is used in computer vision applications for detecting objects in a video or image, which by definition is a feature descriptor. First the input image is given to colour normalization block. Colour normalization is used for object recognition on colour images when it is important to remove all intensity values from the picture while preserving colour values. After colour normalization, the second step of calculation is the computation of the gradient values. The most common method is to apply the 1D centred point discrete derivative mask in both the horizontal and vertical directions. Specifically, this method requires filtering the grey scale image.

I. DISTANCE CALCULATION OF THE DETECTED ANIMAL: The video is taken and converted into frames (image of size 640 ∗ 480). Following is the procedure for calculating the distance of the detected animal from the camera-mounted vehicle:

i. Image resolution is 640 × 480

ii. X range is 0 to 640

iii. Y range is 0 to 480

Let the right bottom coordinate of the detected cow be (x, y). As shown in fig2.9.Then the distance of cow from the lower edge (car/camera) is 480 – y.

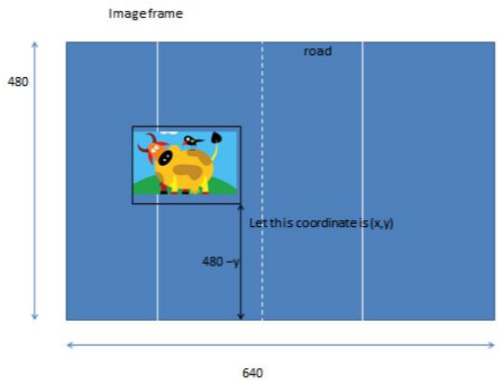

fig 2.9 Distance measure

II. CONVERSION FROM PIXELS TO METERS*:* There is some relationship between the depth of the object in pixel and depth in real world units (meters) from the camera mounted vehicle once the object (animal) gets detected in the frame. As the depth of the object in meters from the camera mounted vehicle increases (size of the object decreases), the depth in pixels also increases and vice versa This hinted us to find a relationship between the depth of the object in pixels and meters. Once the camera position in the car and height of the camera from the ground was fixed (camera calibration done), we took different images of the same object kept at various depths from the camera centre .The depth of the object from the camera centre in meters was known to us. We then noted the corresponding depth of the object in pixels. The best fitting second order polynomial equation is

$y = 0.0323x\ 2 + 22.208x + 1.3132$ (1) where y is the depth in pixels and x is depth in meters

Dataset

A good source for the animal images is the KTH dataset and NEC dataset that included pictures of cows and other animals. Some more animal images have been clicked (during different weather conditions i.e. morning, afternoon and evening) for creating a robust database of almost 2200 images

Results

Fig 2.10 shows that a cow is detected using and distance is calculated.

FIGURE 14. Camera mounted vehicle.

fig 2.10 Cow detection

Limitations

Though our proposed system can detect the animals (cow) on roads and highways as well as gives alert to the driver, it has some limitations too. The proposed system can detect animal up to a distance of 20 meters only when a vehicle is stationary. The system can prevent collision of the vehicle with the animal when driving at a speed in between 30 to 35 kmph. Beyond this speed, though animal gets detected time is not sufficient to prevent animal-vehicle collision.

2.3.6 Survey Paper 6

Title

You Only Look Once: Unified, Real-Time Object Detection

Authors and Year of Publication

Authors: Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

Year of Publication: 2016

Problem Statement

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection.

Methodology/Algorithm

The YOLO design enables end-to-end training and real time speeds while maintaining high average precision. This system divides the input image into an $S \times S$ grid. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) $*$ IOUtruth pred. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth. Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the centre of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. Each grid cell also predicts C conditional class probabilities, Pr(Class$i$|Object). These probabilities are conditioned on the grid cell containing an object.

Dataset

The model was evaluated on PASCAL VOC 2007 and PASCAL VOC 2012

Results

First the developed YOLO was compared with other real-time detection systems on PASCAL VOC 2007. To understand the differences between YOLO and R-CNN variants the errors were explored on VOC 2007 made by YOLO and Fast R-CNN, one of the highest performing versions of R-CNN. Based on the different error profiles it was shown that YOLO can be used to rescore Fast R-CNN detections and reduce the errors from background false positives, giving a significant performance boost. VOC 2012 results were also presented and compare mAP to current state-of-the-art methods. Finally, it was shown that YOLO generalizes to new domains better than other detectors on two artwork datasets. The complete comparison between RCNN's and YOLO has been shown in Table 2.1

Table 2.1 YOLO vs RCNN's

	mAP	**Combined**	**Gain**
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	66.9	72.4	0.6
Fast R-CNN (VGG-M)	59.2	72.4	0.6
Fast R-CNN (CaffeNet)	57.1	72.1	0.3
YOLO	63.4	75.03	0.2

Limitations and Drawbacks

YOLO imposes spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that the model can predict. The model struggles with small objects that appear in groups, such as flocks of birds. Since the model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. The model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down sampling layers from the input image.

2.3.7 Survey Paper 7

Title

Human face detection algorithm via Haar cascade classifier combined with three additional classifiers

Authors and Year of Publication

Authors: Li Cuimei1, Qi Zhiliang, Jia Nan 2, Wu Jianhua

Year of Publication: 2017

Problem Statement

Human face detection has been a challenging issue in the areas of image processing and patter recognition. A new human face detection algorithm by primitive Haar cascade algorithm combined with three additional weak classifiers is proposed in this paper.

Methodology/Algorithm

The typical cascade classifier is the very successful method of Viola and Jones for face detection. Generally, many object detection tasks with rigid structure can be addressed by means of this method, not limited to face detection. The cascade classifier is a tree-based technology, in which Viola and Jones used Haar-like features for human face detection. The Haar-like features by default can be used with all scales in the boosted classifier and can be rapidly computed from an integral version of the image to be detected in.

On account of the aforementioned background and discussions, a new human face detection algorithm is proposed to implement a stronger whole detection system by appending three weak classifiers, i.e., a classifier based on skin tone histogram matching, a classifier based on eyes detection. The proposed human face detection as shown in fig 2.11 is implemented with the help of OpenCV. However, at the stage of skin hue histogram matching, the prototype of histogram should be obtained before the test phase of human face detection. The block F0 is itself a cascade of classifiers using Haar-like features. The cascade classifier supplied OpenCV 3.3.0 is used.

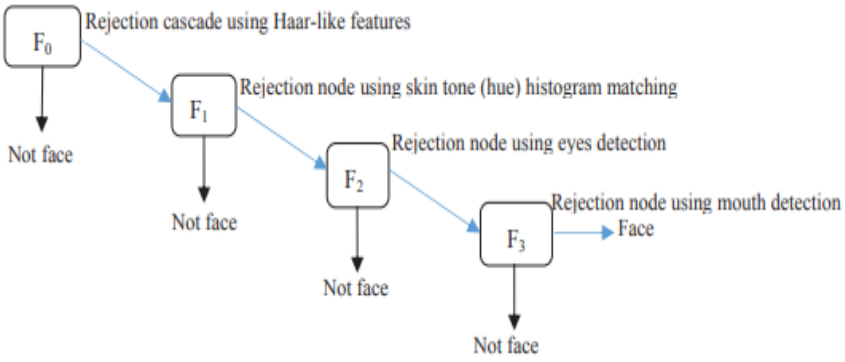

fig 2.11 Proposed framework of human face detection cascade

Dataset

30 color images of people of our friends, students, colleagues, and family members, with their consent were collected. All people were of yellow race, including 344 human faces. These were manually verified from initially detected faces (440 true and false faces). After removing 96 wrongly detected faces in initial phase, the prototype of skin hue histogram is determined by remaining 344 skin hue histograms of true faces. Then, these 30 color images of people are again processed by the proposed human face detection system.

Results

A statistic of human face detection results are shown in fig 2.12, where PPV means positive prediction value, defined as PPV =TP/(TP+ FP) * 100% where TP means true positives, i.e., the number of correct detected faces, and FP means false positives, i.e., wrongly detected faces (non-faces as faces). Because the true negatives (TN) is very large and the false negatives (FN) is very small, comparison of other measures, such as sensitivity and specificity partly in terms of FN or TN is of little significance. Thus, only the results of measure of PPV are given in fig 2.12.

Number of images	30	
Real human faces	344	
Faces detected by primitive cascade of classifiers	440	PPV: 78.18%
Faces detected by proposed detection system	351	PPV: 98.01%

fig 2.12 Results for test images

3. METHODOLOGY

3.1 EXISTING SYSTEM

The existing models for obstacle detection are mostly based on sensors i.e., ultrasonic sensors, IR sensors etc. The ultrasonic sensor sends ultrasonic waves, as shown in fig 3.1 and fig 3.2 and records the echo which will further be used to calculate the distance. This will work only if the obstacle is within the range/path of the ultrasonic waves. If any obstacle is below the path, it will not be detected as the waves don't hit the obstacle and are not reflected back thereby not detecting the obstacle. Another drawback of using the above stated method is that the type of obstacle will not be known.

In high end luxury cars, a rear-view camera is used to detect obstacles, the underlying technology again goes back to using sensors and not the actual camera. The camera display is just for the driver to see and get a better view while reversing the car.

Obstacle detection is a key problem in computer vision for navigation. Existing methods could be categorized into two main approaches. The first approach learns object model then verifies if a pixel or an image patch satisfies the learnt model. The second approach is based upon a definition of objectness and detects regions with the highest objectness measures. In method for obstacle avoidance based on stereovision and a simplistic ground plane detection

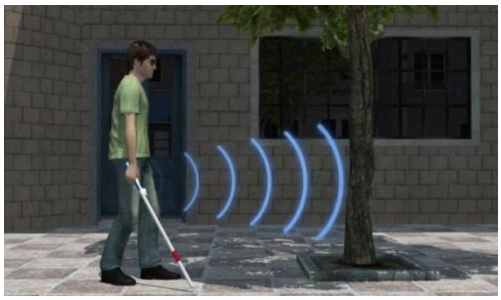

fig 3.1 Walking aide for the blind

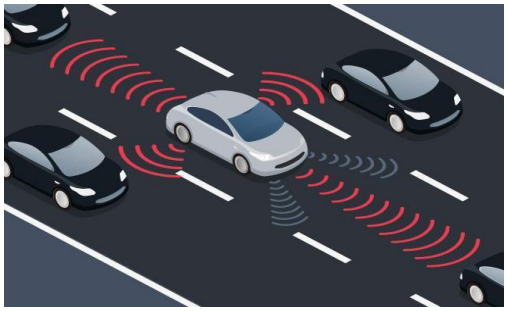

fig 3.2 Obstacle detector sensors in cars

3.2 PROPOSED SYSTEM

To overcome the drawbacks of using an ultrasonic sensor, a system using camera as a sensor and a speaker to alert the user is proposed.

A raspberry pi board with camera and speaker will be used to detect the obstacle and alert the user by reading out the distance and type of obstacle/object in the path of motion. Machine Learning algorithm will be used to train the system. The underlying algorithm is CNN(Convolutional Neural Network). The model will be trained to identify the type of object and calculate the approximate, if not exact, distance of the object from the camera. Once the object is detected, the type of object and the distance for humans will be read out using a speaker and if the distance is less than a 1meter, then an message will be read out alerting the user and giving them information about the type of object and the distance for humans identified thereby they can avoid the obstacle or reduce the impact of collision. Objects in motion will be detected as well by using the concept of relative motion.

This technique can be used in automated or driverless cars, where instead of just measuring the distance and reading out an alert message, the speed of the vehicle can be automatically adjusted to either avoid the obstacle, reduce the impact or avoid accidents by triggering the application of breaks. The proposed system will be implemented using 2 algorithms Fast RCNN and YOLO

Algorithm 1 : Fast RCNN

The in object detection algorithms, one tries to draw a bounding box around the object of interest to locate it within the image. Not necessarily draw just one bounding box in an object detection case, there could be many bounding boxes representing different objects of interest within the image.

RCNN-A method where selective search is used to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions, you can just work with 2000 regions. The selective search algorithm which is written below.

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

Drawbacks of RCNN:

i. It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.

ii. It cannot be implemented real time as it takes around 47 seconds for each test image.

iii. The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

Some of the drawbacks of R-CNN are solved by a faster object detection algorithm called Fast R-CNN, as shown in fig 3.3. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, the input image is fed to the CNN to generate a convolutional feature map. From the convolutional feature map, the identified region of proposals are wrapped into squares and a RoI pooling layer is used to reshape them into a fixed size so that it can be fed into a fully connected

layer. From the RoI feature vector a softmax layer is used to predict the class of the proposed region and also the offset values for the bounding box.

The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

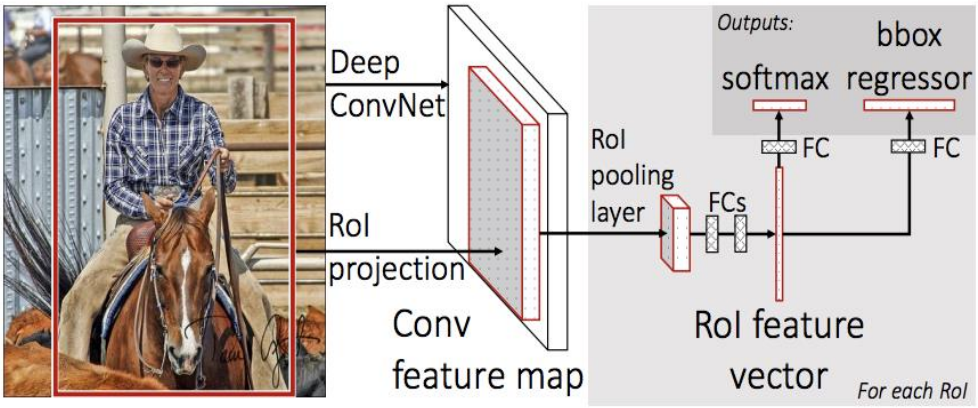

fig 3.3 Fast RCNN Working

Algorithm 2 : You Only Look Once(YOLO)

A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

First, YOLO is extremely fast. Since one frames detection as a regression problem a complex pipeline is not needed. A neural network is simply run on a new image at test time to predict detections. The network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means the streaming video in real-time can be processed with less than 25 milliseconds of latency.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

The network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means the network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.

The following points explain the process of object detection and how bounding boxes are drawn.

1. As shown in figure 3.4, An image is split into SxS grid, within each of the grid m bounding boxes are taken.
2. For each of the bounding box, the network outputs a class probability and confidence scores values for the bounding box.
3. These confidence scores reflect how confident the model is that the box contains an object and how accurate the prediction is.

4. Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell.
5. The width and height are predicted relative to the whole image.
6. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

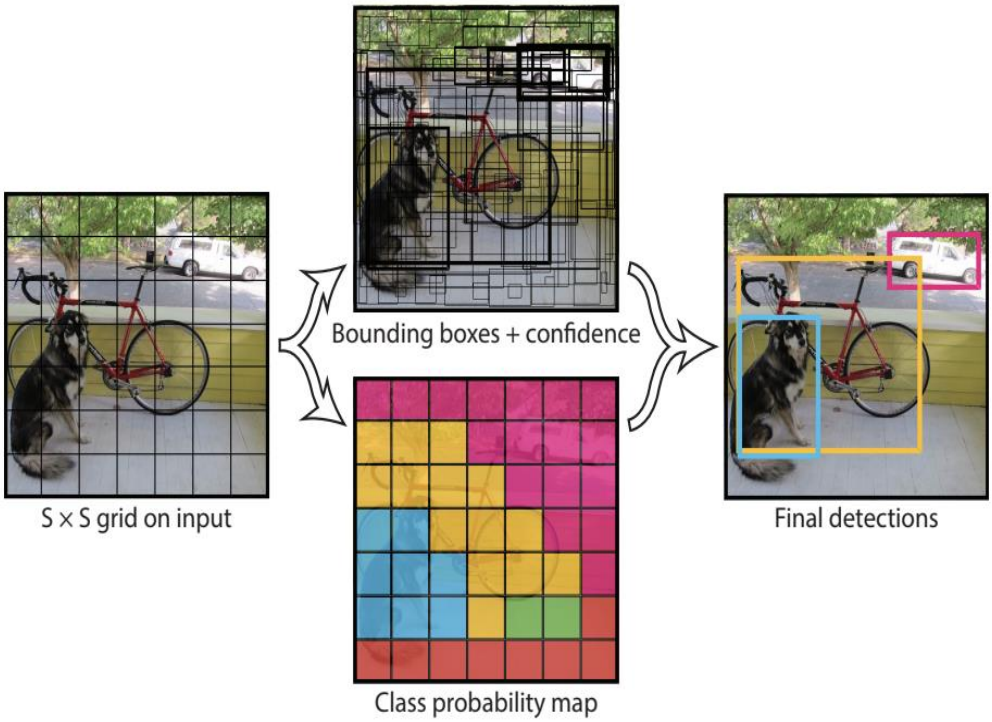

fig 3.4 YOLO Working

3.3 ARCHITECTURE

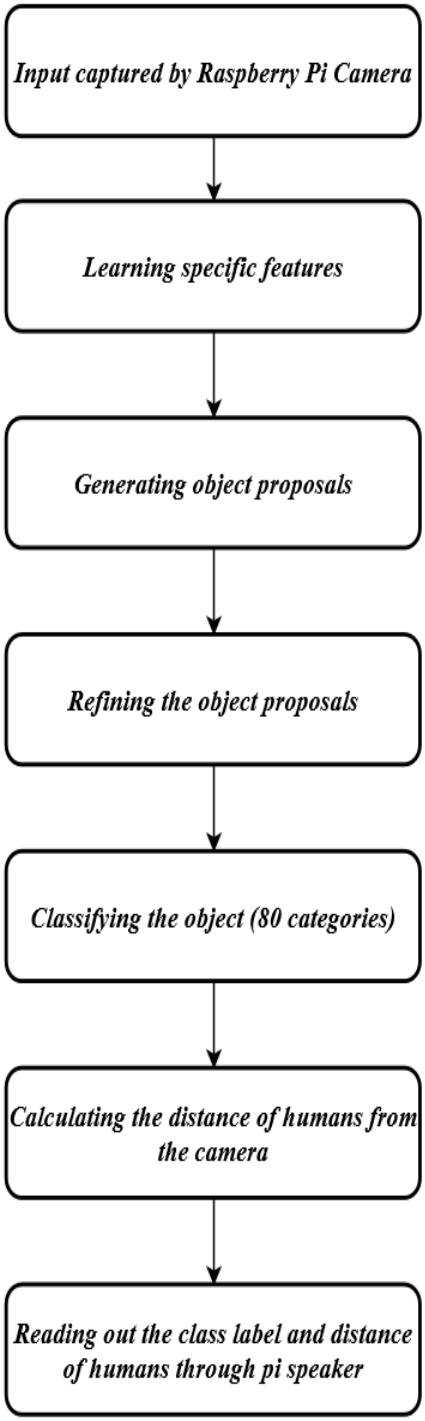

fig 3.5 Design

As shown in fig 3.5, the phases of the proposed system design are described as follows :

Phase 1: The raspberry pi camera captures the input video live to detect objects in real time.

Phase 2: The video captured is broken down into frames and the model trained extracts specific features from the frames.

Phase 3: The boundary boxes are drawn and object proposals are generated.

Phase 4: Object proposals from phase 3 and the surrounding proposals from the vicinity are grouped together. Each object proposal grouped together detects a particular part of the object. These object proposals are refined to get the actual object that needs to be detected.

Phase 5: The refined objects from phase 4 are classified according to the trained categories. The number of categories that the model can identify us 80.

Phase 6: Based on the contours of the humans detected, the distance between the detector and human is calculated. A Haar classifier is used to identify the contours

Phase 7: The type of object along with the distance for humans is read out using the speaker attached to the Raspberry Pi.

4. IMPLEMENTATION

The code was written in Python. Implemented on a Laptop CPU first to compare the speeds. On observation it was found that RCNN is slower than YOLO. The differences between RCNN and YOLO are stated below.

RCNN

Advantage:

Faster RCNN generalizes well even when objects in the image show rare aspects of ratio. So, it detects small objects well with good amount of accuracy.

Limitation:

Faster RCNN is faster than a lot of other existing algorithms but still is slower than YOLO in processing the frames.

YOLO

Advantages:

YOLO is a lot faster than other object detection algorithms. It has the capability to process 45 frames per second. It makes classification and bounding box regression at the same time.

Limitation:

The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

4.1 RCNN IMPLEMENTATION

Using OpenCV PCNN was implemented for simple object detection and found that it identifies unnecessary smaller objects. This increases the computation time by seconds.

4.2 YOLO IMPLEMENTATION

YOLO was first implemented using Keras and tensorflow. The model was trained on an online GPU and executed. When the same model was deployed on the CPU it had a time lag which eventually led to termination of the program. To find an alternative solution, the CFG(configuration) and Weight files of the YOLO algorithm which was trained for the COCO data set were considered.

The underlying CNN for YOLO had the following layers in the CFG file :

[net] section

1. batch=64 - number of samples (images, letters, ...) which will be processed in one batch
2. subdivisions=16 - number of mini_batches in one batch, size mini_batch = batch/subdivisions, so GPU processes mini_batch samples at once, and the weights will be updated for batch samples (1 iteration processes batch images)
3. width=6-8 - network size (width), so every image will be resized to the network size during Training and Detection
4. height=608 - network size (height), so every image will be resized to the network size during Training and Detection
5. channels=3 - network size (channels), so every image will be converted to this number of channels during Training and Detection
6. inputs=256 - network size (inputs) is used for non-image data: letters, prices, any custom data

Optimizator:

7. momentum=0.9 - accumulation of movement, how much the history affects the further change of weights.
8. decay=0.0005 - a weaker updating of the weights for typical features, it eliminates dysbalance in dataset.
9. learning_rate=0.001 - initial learning rate for training

10. burn_in=1000 - initial burn_in will be processed for the first 1000 iterations, current_learning rate = learning_rate * pow(iterations / burn_in, power) = 0.001 * pow(iterations/1000, 4) where is power=4 by default
11. max_batches = 500200 - the training will be processed for this number of iterations (batches)
12. policy=steps - policy for changing learning rate: constant (by default), sgdr, steps, step, sig, exp, poly, random (f.e., if policy=random - then current learning rate will be changed in this way = learning_rate * pow(rand_uniform(0,1), power))
13. steps=8000,9000,12000 - if policy=steps - at these numbers of iterations the learning rate will be multiplied by scales factor
14. scales=.1,.1,.1 - if policy=steps - f.e. if steps=8000,9000,12000, scales=.1,.1,.1 and the current iteration number is 10000 then current_learning_rate = learning_rate * scales[0] * scales[1] = 0.001 * 0.1 * 0.1 = 0.00001

Data Augmentation:

15. angle=0 - randomly rotates images during training (classification only)
16. saturation = 1.5 - randomly changes saturation of images during training
17. exposure = 1.5 - randomly changes exposure (brightness) during training
18. hue=.1 - randomly changes hue (color) during training

The .weights file contains weights of the YOLO model generated for the COCO dataset by transfer learning.

4.3 OpenCV FUNCTIONS

1. **Net cv::dnn::readNetFromDarkenet(const String& cfgFile, const String& darknetModel = string()) -** Returns the Net object which contains the network model.
   cfgFile – path to .cfg file with description of network architecture.

darknetModel – path to .weights file with learned network.

2. **Void cv::dnn:Net::setPreferableBackend(int backendId)** – To instruct network to use specific computation backend.
   backendId – backend identifier(cv.dnn.DNN_BACKEND_OPENCV)

3. **Void cv::dnn::Net::setPreferabaleTarget(int targetID)** – To instruct network to make computations on specific target device
   targetId – target identifier (cv.dnn.DNN_TARGET_CPU)

4. **cv2.namedWindow(winname[, flags])**
   name – Name of the window in the window caption that may be used as a window identifier.
   flags – WINDOW_NORMAL: If this is set, the user can resize the window (no constraint).

5. **cv2.resizeWindow(winname, width, height)**
   winname – Window name
   width – The new window width
   height – The new window height

6. **cv2.VideoCapture()**
   opens camera and starts recording

7. **cv2.waitKey([delay])**
   delay – Delay in milliseconds. 0 is the special value that means “forever”.

8. **cv2.VideoCapture.read([image])**
   This is the most convenient method for reading video files or capturing data from decode and return the just grabbed frame. If no frames has been grabbed (camera has been disconnected, or there are no more frames in video file), the methods return false and the functions return NULL pointer.

9. **cv::dnn::blobFromImage(image[, scalefactor[, size[, mean[, swapRB[, crop[, ddepth]]]]]]) -** Returns 4-dimensional Mat with NCHW dimensions order.

   image- input image (with 1-, 3- or 4-channels).

   size - spatial size for output image.

   mean - scalar with mean values which are subtracted from channels.

   scalefactor - multiplier for image values.

   swapRB - flag which indicates that swap first and last channels in 3-channel image is necessary.

   crop - flag which indicates if the image will be cropped after resize or not

   ddepth - Depth of output blob. Choose CV_32F or CV_8U.

10. **cv::dnn::Net::forward(const String & *outputName* = String())**- returns blob for first output of specified layer.

    outputName - name for layer which output is needed to get

11. **cv::dnn::Net::setInput(blob[, name[, scalefactor[, mean]]])**

    blob - A new blob should have CV_32F or CV_8U depth.

    name - A name of input layer.

    ncalefactor - An optional normalization scale.

    mean - An optional mean subtraction values.

12. **cv2.imshow(winname, mat)**

    winname – Name of the window.

    image – Image to be shown.

13. **cv2.destroyAllWindows()** - The function destroyAllWindows destroys all of the opened HighGUI windows.

14. **Void cv::dnn::NMSBoxes(const std::vector< Rect > & bboxes, const std::vector< float > & scores, const float score_threshold, const float**

nms_threshold, std::vector< int > & indices, const float eta = 1.f, const int top_k = 0)

bboxes - a set of bounding boxes to apply NMS.

scores- a set of corresponding confidences.

score_threshold- a threshold used to filter boxes by score.

nms_threshold-a threshold used in non maximum suppression.

indices - the kept indices of bboxes after NMS.

Eta-a coefficient in adaptive threshold formula :nms_thresholdi+1= eta· nms_thresholdi.

top_k-if >0, keep at most top_k picked indices.

15. **cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]])**

    img – Image.

    pt1 – Vertex of the rectangle.

    pt2 – Vertex of the rectangle opposite to pt1 .

    rec – Alternative specification of the drawn rectangle.

    color – Rectangle color or brightness (grayscale image).

    thickness – Thickness of lines that make up the rectangle.

    lineType – Type of the line.

    shift – Number of fractional bits in the point coordinates

16. **cv.CvtColor(src, dst, code)** – Returns an image

    src – input image: 8-bit unsigned, 16-bit unsigned, or single-precision floating-point.

    dst – output image of the same size and depth as src.

    code – color space conversion code

17. **cv.CascadeClassifier(const String & filename) -** Loads a classifier from a file.

    filename-Name of the file from which the classifier is loaded.

18. **cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) -** returns a list of rectangles

    cascade – Haar classifier cascade

    image – Matrix of the type CV_8U containing an image where objects are detected.

    objects – Vector of rectangles where each rectangle contains the detection

    scaleFactor – Parameter specifying how much the image size is reduced at each image scale.

    minNeighbors – Parameter specifying how many neighbors each candidate rectangle should have to retain it.

    flags – Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.

    minSize – Minimum possible object size. Objects smaller than that are ignored.

    maxSize – Maximum possible object size. Objects larger than that are ignored.

19. **cv2.putText(image, text, org, font, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]) -** Returns an image

    image- It is the image on which text is to be drawn.

    text- Text string to be drawn.

    org- It is the coordinates of the bottom-left corner of the text string in the image. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

    font- It denotes the font type. Some of font types are FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN, , etc.

    fontScale- Font scale factor that is multiplied by the font-specific base size.

    color- It is the color of text string to be drawn. For BGR, pass a tuple. eg: (255, 0, 0) for blue color.

    thickness- It is the thickness of the line in px.

    lineType- This is an optional parameter.It gives the type of the line to be

used.

bottomLeftOrigin- This is an optional parameter. When it is true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

4.4 OTHER FUNCTIONS

1. **numpy.argmax(array, axis = None, out = None) -** Returns indices of the max element of the array in a particular axis.
   Array- Input array to work on
   Axis- [int, optional]Along a specified axis like 0 or 1
   Out- [array optional]Provides a feature to insert output to the out array and it should be of appropriate shape and dtype

2. **numpy.append(array, values, axis = None) :** appends values along the mentioned axis at the end of the array
   Array- [array_like]Input array.
   Values- [array_like]values to be added in the arr
   Axis-Axis along which the values are to be inserted. By default, array is *Flattened*

4.5 DISTANCE CALCULATION FOR HUMANS

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then features are to be extracted from. For this, Haar features shown in the fig 4.1 are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

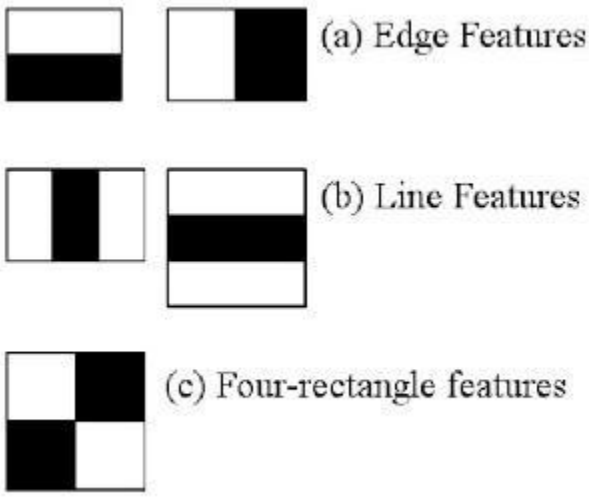

fig 4.1 Haar Classifier

All possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, find the sum of the pixels under white and black rectangles.

Creating Haar Cascade XML file for classification :

To generate the haar XML file "positive" images, and "negative" images are needed. The "positive" images are images that contain the object one wants to find. This can

either be images that just mainly have the object, or it can be images that contain the object, and one can specify the ROI (region of interest) where the object is. With these positives, a vector file is built that is basically all of these positives put together.

From single positive image, the opencv_createsamples command is used to actually create a bunch of positive examples, using negative images. Positive image will be superimposed on these negatives, and it will be angled. Similar to the WordNet, ImageNet is used for image training, so their images are specific and categorised. Using OpenCV commands, the URL lists are taken from which the links are grabbed and visited, pull the images, resize them, save them, and repeat until all required images are obtained.

Using DigitalOcean paid server, XML file was generated using the OpenCV commands for humans.

Form the positive images, a vector file is created which basically stitches all the positive images together. To do that opencv_createsamples is used.
-->opencv_createsamples -info info/info.lst -num 1950 -w 20 -h 20 -vec positives.vec

To train the haar cascade classifier, opencv_traincascade command is used. First give the number of positive images and number of negative images where number of positive images are double the number of negative images. Then give the number of stages, this decides how many times the classifier should run to train the model. After that provide the command with the width and height of the images that are used for training.
To train this classifier that detects humans, 800 positive images that contain humans and 400 negative images that contains no humans at all in any of the images were chosen.

-->opencv_traincascade -data data -vec positives.vec -bg bg.txt -numPos 1800 -numNeg 900 -numStages 10 -w 20 -h 20

After the above command is executed, an xml file is generated which contains information about all the features that is required to detect a human. This classifier is integrated into the YOLO object detection algorithm to enhance human detection and calculate distance of humans from camera using the formula:
(2*3.14 * 180)/(width+height*360)*1000 + 3

4.6 VOICE ASSISTANCE

The objects that are identified will be stored in a set as they are getting detected. Similarly the distance that is calculated will also be read out.

1. GTTS engine was used to read out the labels. (*Google Text-to-Speech*), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Writes spoken MP3 data to a file, a file-like object (bytestring) for further audio manipulation, It features flexible pre-processing and tokenizing, as well as automatic retrieval of supported languages.
2. Pygame is a Python wrapper module for the multimedia library to play Audio files dynamically as they get stored.

4.7 INTEGRATION WITH PI

The Raspberrypi was connected to the laptop by getting the IP address of the PI. This IP address was used to connect to it through the putty. The default login credentials have to be entered while accessing the pi. SSH and Camera interfaces have to be enabled. The python script , cfg file and .weight files must be placed in a single folder. The python file has to be run on the optimised IDE known as Thonny.

1. Multiple packages like OpenCV, gTTS, SSH, pygame were installed
2. The object detection program was run using optimised Python IDE, Thonny
3. Pi was accessed on the laptop using Putty , Xming Server, Windows Remote Control

4. Camera was interfaced to the pi through the CSI port
5. For audio output, a speaker/headphone was connected through the audio jack provided on the Raspberrypi

5. RESULTS

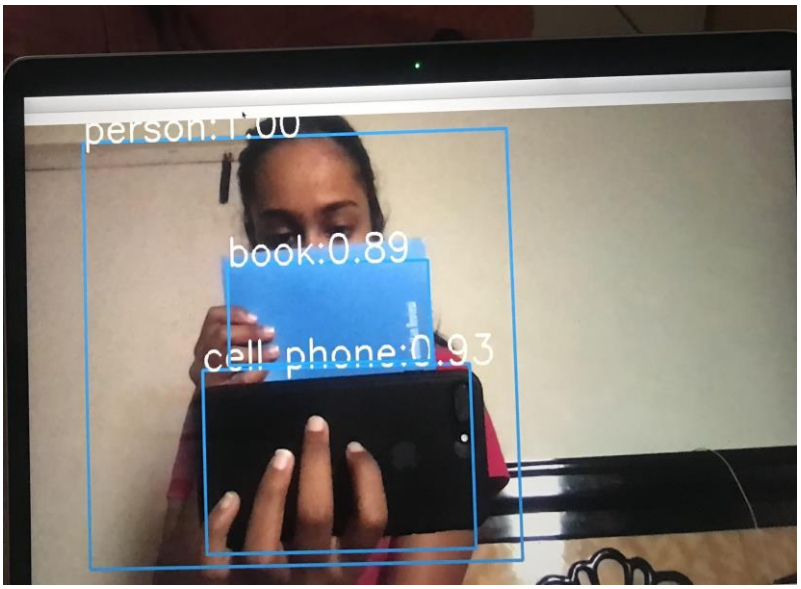

fig 5.1 Running Object detector on Laptop

YOLO object detector was run on a CPU. For an object to be identified and classified, the threshold value was set to a standard value of 0.25. If for the object detected the threshold value was greater than or equal to 0.25, bounding boxes were generated and the class label was predicted according to the trained network. In fig 5.1 the objects detected were person, book, cell phone with their corresponding confidence values of 1.00, 0.89,0.93 respectively. On the CPU the frames were processed at a rate of 3 FPS.

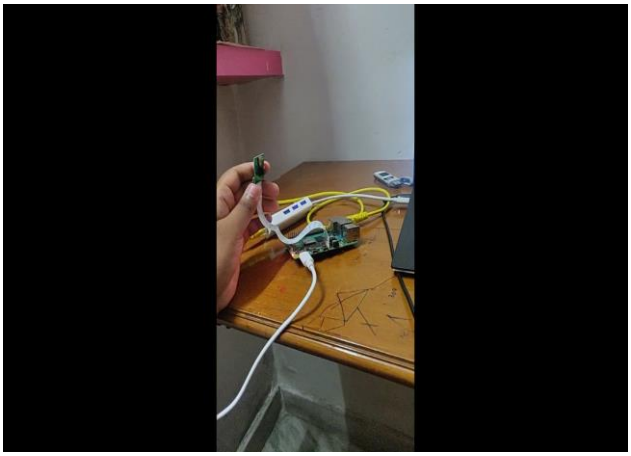

fig 5.2 Connecting Raspberry Pi to laptop

To access the Operating System of Raspberry Pi and run the object detection model, it was connected to the laptop as shown in fig 5.2. The camera was connected by enabling the Camera Serial Interface.

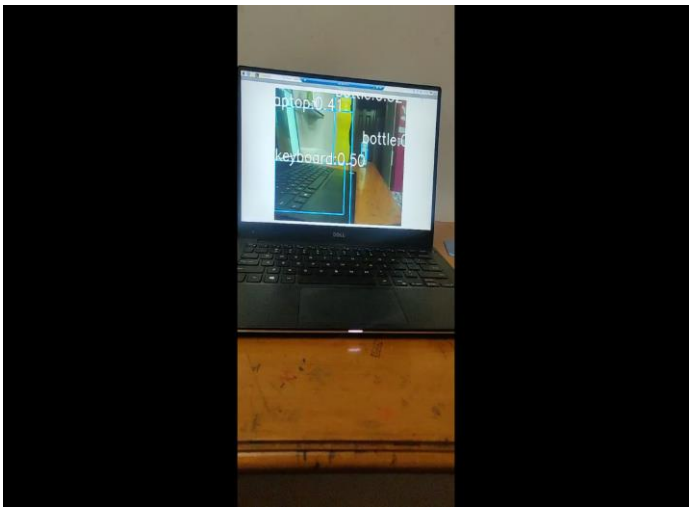

fig 5.3 object detection running on Raspberry Pi viewed through laptop

The object detection model was run on Raspberry Pi using the python IDE ‘Thonny’. The frames were processed at a rate of 0.6fps. As shown in fig 5.3 the objects identified were laptop with a confidence of 0.41,keyboard with a confidence of 0.50, two bottles with a confidence of 0.32 each. The time taken to identify objects was much slower compared to the regular CPU as the fps on Raspberry Pi is much lower than that of the CPU

fig 5.4 Object detection running on Raspberry Pi viewed through Tv monitor

As shown in fig 5.4, the object detection was viewed using the TV monitor,connected using the HDMI slot.

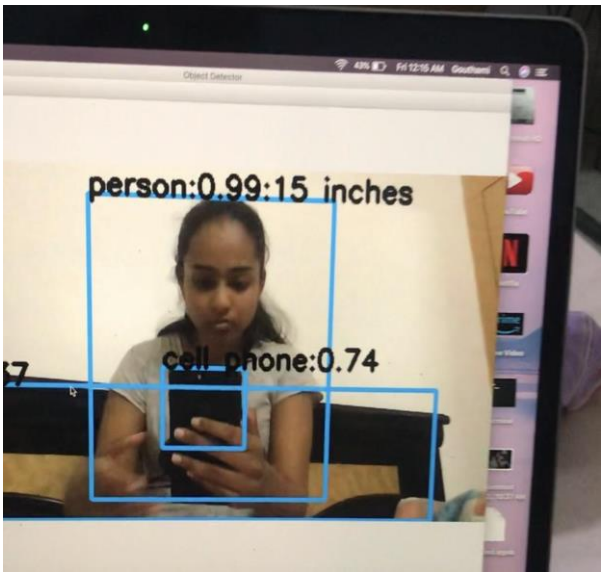

fig 5.5 Distance measurement for humans along with detection

As shown in fig 5.5 the objects detected were cell phone, bed and person with confidences of 0.74, 0.99, 0.67 respectively. The distance for a human from the monocular camera was calculated by running the Haar classifier which takes the contours of the human. The distance is shown in inches. For the person above the distance calculated by the classifier was 15 inches whereas in real time it was between 16-17inches. Therefore the accuracy of distance calculation is high.

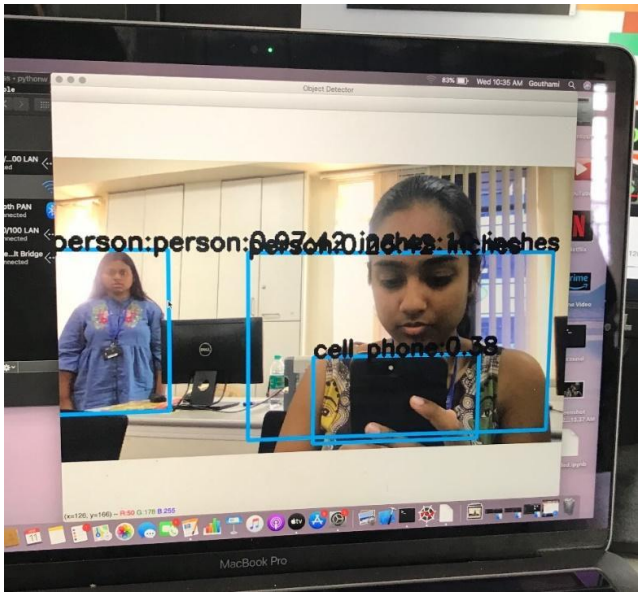

fig 5.6 Distance measurement for two humans along with detection

As shown in fig 5.6, the distances for two humans detected were 42 inches and 11 inches respectively.

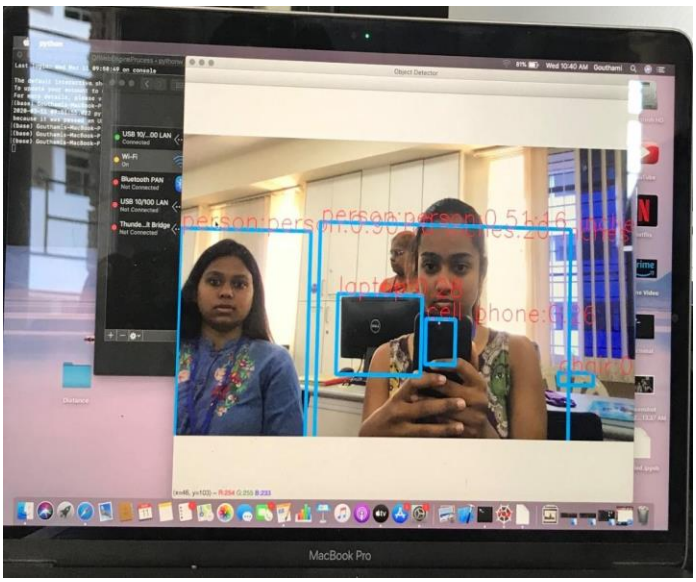

fig 5.7 Multiple Objects detected with distances for 2 humans

As shown in fig 5.7, the distances for three humans were detected and other objects like monitor(laptop) and phone were identified.

6. CONCLUSION AND FUTURE SCOPE

To conclude the report, after testing under different circumstances, the objects were accurately identified by the model developed using YOLO .The distance calculation and human detection enhancement was done using Haar Cascading Classifier which worked fairly well for multiple distances. As the objects were identified, a text-to-speech engine was used to store the labels of objects and distance in inches for humans as an MP3 file and read out dynamically using Pygame. The obstacle detector that will be implemented will have diverse uses the main one being an obstacle detector for the blind as their walking aide for smooth navigation. The other one being object detection in automated/driver less cars. Under the future scope of this project the following two modules have to be looked into and improvised

i. An efficient method for the distance calculation for multiple objects monocular camera is yet to be formulated. 3D reconstruction which is under research ,can be used to calculate the distance for multiple objects.
ii. The processing of frames on the Raspberry Pi is less compared to that of a CPU and GPU. This makes it very slow. A method to increase the speed of processing of the frames must be deviced.

BIBLIOGRAPHY

1. Jianan Li , Xiaodan Liang, Jianshu Li , Yunchao Wei , Tingfa Xu , Jiashi Feng, and Shuicheng Yan, “Multistage Object Detection with Group Recursive Learning”, IEEE TRANSACTIONS ON MULTIMEDIA, Vol. 20, Pages 1645-1655, 2018.
2. Shuai Zhang (Member, IEEE), Chong Wang (Member, IEEE), Shing-Chow Chan (Member, IEEE), Xiguang Wei, and Check-Hei Ho, “New Object Detection, Tracking, and Recognition Approaches for Video Surveillance Over Camera Network”, IEEE SENSORS JOURNAL, Vol. 15, Pages 2679-2692, 2015.
3. Shuze Du and Shifeng Chen, “Salient Object Detection via Random Forest”, IEEE SIGNAL PROCESSING LETTERS, Vol. 21, Pages 51-56, 2014.
4. Radhika Kamath, Mamatha Balachandra (Member, IEEE), and Srikanth Prabhu(Member, IEEE), “Raspberry Pi as Visual Sensor Nodes in Precision Agriculture: A Study”, IEEE Access, Vol. 7, Pages 45110 - 45122, 2019.
5. Sachin Umesh Sharma, Dharmesh J. Shah, “A Practical Animal Detection and Collision Avoidance System Using Computer Vision Technique”, IEEE Access, Vol. , Pages 347-359, 2017
6. Duy Thanh Nguyen, Tuan Nghia Nguyen, Hyun Kim, Hyuk-Jae Lee, “A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 27, Pages 1861 - 1873, 2017
7. Xing Wang, Tingfa Xu, Jizhou Zhang, Sining Chen, Yizhou Zhang, “SO-YOLO Based WBC Detection With Fourier Ptychographic Microscopy”, IEEE Access, Vol. 6, Pages 51566 - 51576, 2018
8. Wei Fang, Lin Wang, Peiming Ren, “Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments”, IEEE Access, Vol. 8, Pages 1935 - 1944, 2020
9. Hualin Yang ; Long Chen ; Miaoting Chen ; Zhibin Ma ; Fang Deng ; Maozhen Li ; Xiangrong Li, “Tender Tea Shoots Recognition and Positioning for Picking Robot Using Improved YOLO-V3 Model”, IEEE Access, Vol. 7, Pages 180998 - 181011, 2019
10. Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, Xindong Wu, “Object Detection with Deep Learning: A Review”, IEEE Paper, Volume 30, Pages 20, 2019

11. Li Cuimei, Qi Zhiliang , Jia Nan , Wu Jianhua, “Human face detection algorithm via Haar cascade classifier combined with three additional classifiers”, IEEE paper, volume 13,Pages 5, 2017
12. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once:
    Unified, Real-Time Object Detection”, IEEE Paper, Volume , Pages 10, 2016
13. Qichang Hu, Sakrapee Paisitkriangkrai, Chunhua Shen, Anton van den Hengel, Fatih Porikli, “Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework”, IEEE Transactions on Intelligent Transportation Systems, Vol. 17, Pages 1002 - 1014, 2016
14. Bastian Leibe, Konrad Schindler, Nico Cornelis, Luc Van Gool, “Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 30, Pages 1683 - 1698, 2008
15. Yunhang Shen, Rongrong Ji, Changhu Wang, Xi Li, Xuelong Li, “Weakly Supervised Object Detection via Object-Specific Pixel Gradient”, IEEE Transactions on Neural Networks and Learning Systems, Vol. 29, Pages 5960 - 5970, 2018
16. Stefania Matteoli, Giovanni Corsini, Marco Diani, Giovanna Cecchi, Guido Toci, “Automated Underwater Object Recognition by Means of Fluorescence LIDAR”, IEEE Transactions on Geoscience and Remote Sensing, Vol. 53, Pages 375 - 393, 2015
17. S. Nadimi, B. Bhanu, “Physical models for moving shadow and object detection in video”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, Pages 1079 - 1087, 2004
18. Jun Nishimura, Tadahiro Kuroda, “Versatile Recognition Using Haar-Like Feature and Cascaded Classifier”, IEEE Sensors Journal, Vol. 10, Pages 942 - 951, 2010
19. C. Gao, S.-l.l. Lu, T. Suh, H. Lim, “Field programmable gate array-based haar classifier for accelerating face detection algorithm”, IET Image Processing, Vol. 4, Pages , 2010
20. Jiaqiu Ai, Ruitian Tian, Qiwu Luo, Jing Jin, Bo Tang, ”Multi-Scale Rotation-Invariant Haar-Like Feature Integrated CNN-Based Ship Detection Algorithm of Multiple-Target Environment in SAR Imagery”, IEEE Transactions on Geoscience and Remote Sensing, Vol. 57, Pages 10070 - 10087, 2019

21. Aiwen Luo, Fengwei An, Xiangyu Zhang, Hans Jürgen Mattausch, ”A Hardware-Efficient Recognition Accelerator Using Haar-Like Feature and SVM Classifier”, IEEE Access, Vol. 7, Pages 14472 - 14487, 2019
22. https://ieeexplore.ieee.org
23. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
24. https://storage.googleapis.com/openimages/web/download.html?source=post_page
25. https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/
26. https://opencv.org/about/

ANNEXURE 1

Paper Title : Object Detector for Visually Impaired with Distance Calculation for Humans

Publication : Blue Eyes Intelligence Engineering and Sciences

Publication Date : April 2020

Object Detector_Paper_103.pdf