International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 05 | May - 2021 ISSN: 2582-3930

# Object detection using open CV

Pooja kumbhar.
Onkar patil
Rushikesh Takale
Rohit kolhapure
Department of electronics engineering,DKTE Institute ichalkaranji.,Maharashtra,India.

Prof-R.N.Patil
Department of electronics engineering,DKTE Institute ichalkaranji.,Maharashtra,India.

Abstract-The aim of this thesis is to explore different methods for helping computers interpret the real world visually, investigate solutions to those methods offered by the open-sourced computer vision library, OpenCV, and implement some of these in a Raspberry Pi based application for detecting and keeping track of objects. The main focus rests on the practical side of the project. The result of this thesis is a GNU/Linux based C/C++ application that is able to detect and keep track of objects by reading the pixel values of frames captured by the Raspberry Pi camera module. The application also transmits some useful information, such as coordinates and size, to other computers on the network that send an appropriate query. The source code of the program is documented and can be developed further.

## Introduction

. Object detection and recognition are important problems in computer vision. Since these problems are meta-heuristic, despite a lot of research, practically usable, intelligent, real-time, and dynamic object detection/recognition methods are still unavailable. We propose a new object detection/recognition method, which improves over the existing methods in every stage of the object detection/recognition process. In addition to the usual features, we propose to use geometric shapes, like linear cues, ellipses and quadrangles, as additional features. Two of the famous robots are the C3PO and R2D2 machines in Star Wars made in 1977. Nowadays, the advances of hardware have created many good robots. One good example is the ASIMO from Honda.

## Problem statement-

• To implement Object detection and Recognition with Help of Raspberry Pi 4 with 8MP Camera by OpenCV.

**Project Objective** :

• To build a Object Detection Model with Raspberry pi.

**Project Scope:**

1. To research and understand the methods of Detecting Objects in OpenCV .

2. To learn how OpenCV Works with HaarCascade Files

. 3. To implement and test the model as a functional prototype.

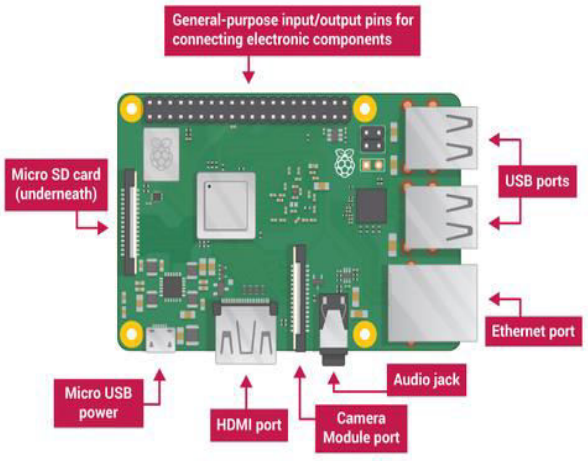
**METHODLOGY:**

The whole concept of Detection and recognition of Objects in our Project lays in the Python Library OpenCV and Coco Dataset which provides us total 91 Objects which can be used in a way in our code to Recognize the Objects in Video/Image Preference. Taking into account the relatively highperformance requirements of image processing in general and the equipment currently available to the faculty, as a relatively inexpensive and powerful embedded platform the Raspberry Pi was an obvious choice. Training and testing the custom model will be done on images of people in winter landscape. The idea is that the mainstream pretrained models out there is trained on people in a variety of landscapes. It would be interesting to see the possibility of fine-tuning the model using simple tools to increase accuracy

## hardware design:

Hardware design includes the selection / design of following hardware modules:

1. Raspberry pi 4
2. 8MP Standard Rpi CAM
3. 32 GB Memory Card
4. 3.1Amp Adaptor



**OVERCLOCKING:**

The easiest way to overclock the Raspberry Pi model B is to do it via Raspberry's configuration interface, which appears on every start-up, or can be opened using the command sudo raspi-config. Overclocking is recommended since image processing operations consume fairly large amounts of CPU power (if not optimized to harness the GPU instead) and if the Raspberry's airflow is above minimal (heat disperses easily), it will not damage the SoC. The only problems that may arise are unstability when remotely accessing the Pi on which a graphical user interface server is running. In that case, the Raspberry quickly shuts down all access via network and can even freeze completely.

**OPEN CV INTRODUCTION:**

• The application written for this thesis relies heavily on computer vision, image processing and pixel manipulation, for which there exists an open source library named OpenCV (Open Source Computer Vision Library), consisting of more than 2500 optimized algorithms. Uses range from facial recognition, object identifying, classifications of human actions in videos, achieved with filters, edge mapping, image transformations, detailed feature analysis and more Having Linux support, this is the perfect choice for developing an application specifically for a Raspberry Pi based system. Another positive aspect of this library is that it's written natively in C++ and therefore can be very smoothly implemented in a C/C++ application.

• While there are numerous methods and algorithms contained within OpenCV, the most important benefit of this library for the purposes of this thesis are its basic data structures like Mat, which can be used to store pixel values of an image in an n-dimensional array, Scalar and Point, which respectively contain pixel values and coordinates of up to 3 dimensions.

The functions provided by this library are also necessary in the development process of the object tracking application. There are numerous options, but following the scope of this thesis, the focus is set on grabbing frames from a live camera feed [8], image thresholding using HSV colour space ranges, finding blobs and using their detected contours in a binary image and, in case a graphical user interface is enabled, displaying of image frames and a control panel for changing parameters during run-time.

**OBJECT DETECTION**:

• Image Classification The task of object classification requires binary labels indicating whether objects are present in an image; Early datasets of this type comprised images containing a single object with blank backgrounds, such as the MNIST handwritten digits or COIL household objects . Caltech 101 and Caltech 256 marked the transition to more realistic object images retrieved from the internet while also increasing the number of object categories to 101 and 256, respectively. Popular datasets in the machine learning community due to the larger number of training examples, CIFAR-10 and CIFAR-100 offered 10 and 100 categories from a dataset of tiny 32×32 images .While these datasets contained up to 60,000 images and hundreds of categories, they still only captured a small fraction of our visual world. Recently, ImageNet made a

- Object detection: Detecting an object entails both stating that an object belonging to a specified class is present, and localizing it in the image. The location of an object is typically represented by a bounding box, Early algorithms focused on face detection using various ad hoc datasets. Later, more realistic and challenging face detection datasets were created . Another popular challenge is the detection of pedestrians for which several datasets have been created. The Caltech Pedestrian Dataset contains 350,000 labelled instances with bounding boxes.

- For the detection of basic object categories, a multi year effort from 2005 to 2012 was devoted to the creation and maintenance of a series of benchmark datasets that were widely adopted. The PASCAL VOC datasets contained 20 object categories spread over 11,000 images. Over 27,000 object instance bounding boxes were labelled, of which almost 7,000 had detailed segmentations. Recently, a detection challenge has been created from 200 object categories using a subset of 400,000 images from ImageNet. An impressive 350,000 objects have been labelled using bounding boxes.

## Algorithmic Analysis:

- Bounding-box detection For the following experiments we take a subset of 55,000 images from our dataset1 and obtain tight-fitting bounding boxes from the annotated segmentation masks. We evaluate models tested on both MS COCO and PASCAL, see Table 1. We evaluate two different models. DPMv5-P: the latest implementation 1.

- These preliminary experiments were performed before our final split of the dataset intro train, val, and test. Baselines on the actual test set will be added once the evaluation server is complete. of [44] (release 5 [45]) trained on PASCAL VOC 2012. DPMv5-C: the same implementation trained on COCO (5000 positive and 10000 negative images). We use the default parameter settings for training COCO models.

## Mobile Net Classifier:

Now it's time for our Mobile Nets to label each proposed image, forwarded by the region proposal system. We pre-process the current frame and generate multiple images by zero-outing all but one objects each time using the input box coordinates. As a result, we get the same number of images as the detected boxes for each input frame, we then feed all the images to the classifier to generate labels. Finally, We take the classification output and combine them with the bounding boxes, overlaying on the original frame. Readers may be curious about why we do not

use the cropped bounding box directly as input to the classifier. The reason is that the classifier is trained for a fixed size images, while in practice small objects appear very often. If we up sample the objects, the resulting images might have too low resolution to be recognized correctly. Another reason is that we assume the background information is helpful for prediction, hence we keep it.

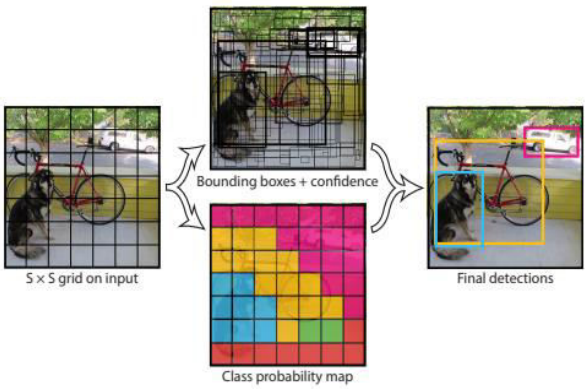
### Dataset pre-processing:

Our image data is supplied by PASCAL VOC2012 detection dataset [5]. The dataset consists of 20 classes, including objects most commonly captured in traffic cameras like bus, car, bicycle, motorcycle and person. The data has been split into 50% for training/validation and 50% for testing, with the distributions of images and objects by class are approximately equal across the training/validation and test sets. This comes to a training set of 5717 images and a validation set of 5823 images

### Mobile net:

• We also implement a detector version of Mobile Net, namely Mobile-Det, by combining Mobile Net classifier and Single Shot MultiBox Detector (SSD) framework [14]. The reason we want to do this is to further analyse the benefit of Mobile Net model, and have a fair comparison between the state-of-art detection model like VGG-based SSD and YOLO. The details of SSD is massive and beyond the scope of this project, so we will only have a brief introduction to how it works in the following parts. In short, SSD framework uses multiple feature layers as classifiers, where each feature map is evaluated by a set of different (aspect ratio) default boxes at each location in a convolutional manner, and each classifier predicts class scores and shape offset relative to the boxes.

• At training time, a default box is considered to be predicting correctly if its Jaccard overlap with the ground truth box is larger than the threshold (0.5). The loss is then measured by both the confident score and localization score.

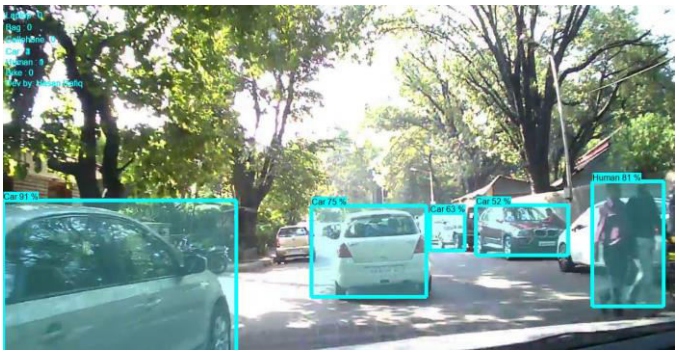


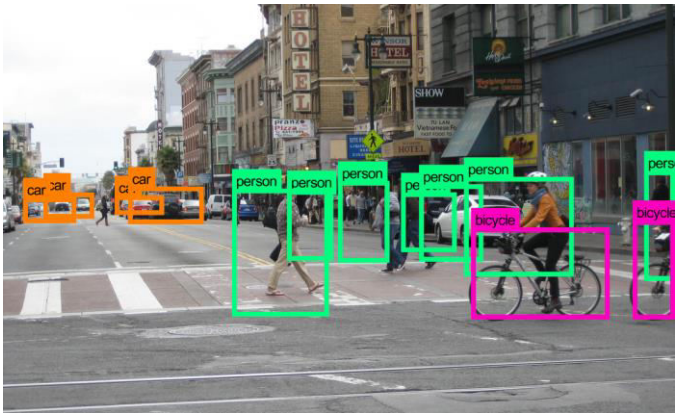

### Face attributes:

• Another use-case for MobileNet is compressing large systems with unknown or esoteric training procedures. In a face attribute classification task, we demonstrate a synergistic relationship between MobileNet and distillation [9], a knowledge transfer technique for deep networks. We seek to reduce a large face attribute classifier with 75 million parameters and 1600 million Mult-Adds.

• MobileNet can also be deployed as an effective base network in modern object detection systems. We report results for MobileNet trained for object detection on COCO data based on the recent work that won the 2016 COCO challenge [10]. In table 13, MobileNet is compared to VGG and Inception V2 [13] under both Faster-RCNN [23] and SSD [21] framework. In our experiments, SSD is evaluated with 300 input resolution (SSD 300) and Faster-RCNN is compared with both 300 and 600 input resolution (FasterRCNN 300, Faster-RCNN 600). The Faster-RCNN model evaluates 300 RPN proposal boxes per image.

### Results





## Conclusion:

The aim of this thesis was to investigate the suitability of running a real time object detection system on a Raspberry Pi. Two models, SSD and YOLO, were implemented and tested in accuracy and speed at different input sizes. The results showed that both models are very slow and that only in applications that doesn't require high speed would it be viable to use the Raspberry Pi as hardware. There is a trade-off with accuracy to be made if higher speeds are to be achieved since there is not enough computational power to have both. This lead to the conclusion that it is important to choose a proper input size to obtain the right balance of speed and accuracy that is needed for a particular application. This study could be of help for others that are looking to implement object detection on similar hardware to find that balance.

## Future Scope:

Due to time constraints there was a lot of things left out that could have been done to strengthen the results of this study such as more testing of different objects, distances and input sizes. One thing that would be interesting is to train own models on different datasets with lower resolutions and see if it could improve accuracy for smaller objects. Another thing that was left out in this project was looking at the impact that lightning has on a models ability to detect objects, this is something that could be explored in future work

## References:

Y. Amit and P. Felzenszwalb, "Object Detection", Computer Vision , pp. 537-542, 2014. ¬ "What is a Raspberry Pi?", Raspberry Pi , 2019. [Online]. Available: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. [Accessed: 06-Mar- 201] ¬ D. Velasco-Montero, J. Fernández-Berni, R. Carmona-Galán and Á.Rodríguez-Vázquez, "Performance analysis of real-time DNN inference on Raspberry Pi", Real-Time Image and Video Processing 2018 , 2018. ¬ W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. Berg, "SSD: Singlehot MultiBox Detector", Computer Vision – ECCV 2016 , pp.21-37, 2016. ¬ T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” Computer Vision – ECCV 2014 Lecture Notes in Computer Science , pp. 740–755, 201