A DEVICE PERFORMING OBJECT DETECTION TO HELP THE BLIND PEOPLE

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

B.PAVAN KUMAR
Roll No. 16551A0517

K.MUKTA
Roll No. 16551A0538

V.ESWAR KUMAR REDDY
Roll No. 16551A05B1

RUPESH SHAH
Roll No. 16551A05B5

Under the Supervision of

Ms. A. VANDANA PETER
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GODAVARI INSTITUTE OF ENGINEERING & TECHNOLOGY
CHAITANYA KNOWLEDGE CITY, NH-16, RAJAMAHENDRAVARAM, AP

Jawaharlal Nehru Technological University, Kakinada, AP, India
APRIL-2020

GODAVARI INSTITUTE OF ENGINEERING & TECHNOLOGY

(Autonomous)

CHAITANYA KNOWLEDGE CITY, NH-16, RAJAMAHENDRAVARAM 533296, AP

BONAFIDE CERTIFICATE

Certified that this project report **"A DEVICE PERFORMING OBJECT DETECTION TO HELP THE BLIND PEOPLE"** is the Bonafide work of "**B.PAVAN KUMAR (16551A0517), K.MUKTA (16551A0538), V.ESWAR KUMAR REDDY (16551A05B1) and RUPESH SHAH(16551A05B5)"**, who carried out the project work under my supervision during the year 2019 to 2020, towards partial fulfilment of the requirements of the Degree of Bachelor of Technology in Computer Science & Engineering as administered under the Regulations of Godavari Institute of Engineering & Technology, Rajamahendravaram, AP, India and award of the Degree from Jawaharlal Nehru Technological University, Kakinada. The results embodied in this report have not been submitted to any other University for the award of any degree.

Signature of the Head of the Department

DR.B. SUJATHA
HEAD OF THE DEPARTMENT
Department of Computer Science & Engineering

Signature of the Supervisor

Ms. A.VANDANA PETER
SUPERVISOR
Professor,
Department of Computer Science & Engineering,
Godavari Institute of Engineering & Technology.

Date:

External Viva voice conducted on _____

Internal Examiner **External Examiner**

GODAVARI INSTITUTE OF ENGINEERING & TECHNOLOGY

(Autonomous)
CHAITANYA KNOWLEDGE CITY, NH-16, RAJAMAHENDRAVARAM 533296, AP

CERTIFICATE OF AUTHENTICATION

We solemnly declare that this project report **"A DEVICE PERFORMING OBJECT DETECTION TO HELP THE BLIND PEOPLE" is** the bonafide work done purely by us, carried out under the supervision of **Ms. A.VANDANA PETER**, towards partial fulfilment of the requirements of the Degree of Bachelor of Technology in Computer Science & Engineering as administered under the Regulations of Godavari Institute of Engineering & Technology, Rajamahendravaram, AP, India and award of the Degree from Jawaharlal Nehru Technological University, Kakinada during the year 2019- 2020.

We also declare that no part of this document has been taken up verbatim from any source without permission from the author(s)/publisher(s). Wherever few sentences, findings, images, diagrams or any other piece of information has been used for the sake of completion of this work, we have adequately referred to the document source. In the event of any issue arising hereafter about this work, we shall be personally responsible.

It is further certified that this work has not been submitted, either in part of in full, to any other department of the Jawaharlal Nehru Technological University Kakinada, or any other University, institution or elsewhere, in India or abroad or for publication in any form.

Signature of the Student(s)

B.PAVAN KUMAR	(16551A0517)
K.MUKTA	(16551A0538)
V.ESWAR KUMAR REDDY	(16551A05B1)
RUPESH SHAH	(16551A05B5)

ACKNOWLEDGEMENT

We feel immense pleasure to express our sincere thanks and profound sense of gratitude to all those people who played a valuable role for the successful completion of our **project.**

We would like to express my gratitude to **Dr. K.V.V. SATYANARAYANA RAJU**, chairman, Chaitanya group of Institutions, **Sri. K. SASI KIRAN VARMA**, Vice Chairman, GIET group of Institutions for providing necessary infrastructure.

We are thankful to Principal **Dr. P.M.M.S.SARMA** for permitting and encouraging us in doing this project.

Our special thanks to **Dr. N. LEELAVATHY**, Vice Principal (Academics), for their content guidance and motivation and also for providing an excellent environment. We are truly grateful for their valuable suggestions and advice.

We are proudly grateful to express my deep sense of gratitude and respect towards **Dr. B. SUJATHA**, professor and Head of the Department, Computer Science and Engineering, whose motivation and constant encouragement has led to pursue a project in the field of software development. We are very fortunate, for having her to enlighten us in all the aspects of life and transforming us to be an ideal individual in this field.

We are much obliged and thankful to our internal guide **Ms**. **A.VANDANA PETER** professor, Department of CSE, for providing this opportunity and constant encouragement given by her during the course. We are grateful to her valuable guidance and suggestions during my project work.

Our parents have put us ahead of themselves, because of their hard work and dedication, we have had opportunities beyond our wildest dreams. Our heart full thanks to them for giving us all support we needed to be successful student and individual.

Finally, we express our special thanks to **Dr. J. M. S. V. Ravi Kumar** and other Professors, classmates, friends and non-teaching staff who helped us for the completion of the project and without this infinite love and patience it would never have been possible.

ABSTRACT

The visually impaired and blind people face various challenges in their day to day life. The objective of the proposed work is to develop an application for visually challenged persons based on the Android smart phone. It will eliminate the need for dedicated devices and other wearable devices to assist them to recognize objects as they move around. The Android application helps the visually impaired to navigate independently using real-time object detection and identification technology. The application makes use of the image processing technique to detect the object and speech synthesis to produce the voice output. The objective of the system is to detect real time objects which are scanned through the mobile camera and notify the blind persons about the object through audio or vocal information. The detection of images on moving objects has been a significant research area in computer vision which has been highly worked upon, and integrated with residential, commercial and industrial environments. Due to lack of data analysis of the trained data, and dependence of the motion of the objects, inability to differentiate one object from the other has led to various limitations in the existing techniques which include less accuracy and performance. Hence, Fast R-CNN (Region-based Convolutional Neural Networks) algorithm has been implemented to detect the object with high accuracy and processing speed. The detected image information is provided as a voice output using a speech synthesizer to the visually challenged persons to assist them in their mobility.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

1.1 Introduction of Project

Millions of people suffer from vision impairment in one or other way. One of the important senses that is very essential for a human being to lead a normal life is the vision. Many People suffer from blindness face quite difficulties while moving around the surrounding environment. This condition leads to the need for guidance or assistance for every action of the handicapped person. Blindness makes the normal, professional and social life of the people very difficult. Human vision has the extraordinary capacities of storing billion of images in the brain and realizing the images by comparing with the pre images. But still some people are not blessed with the vision and some suffer from the retinal diseases. By using computer vision techniques, the quality of life of blind people could be improved. One of the major research areas in computer vision is the object detection and identification technology which is used for object recognition and prediction of moving objects. Objects like intensity, edge, and shape are used to recognize objects from the input image by using object detection technology. The Fast R-CNN algorithm and the trained Tensorflow models are used to detect objects in the image. The computer vision application is deployed in the Android platform due to the wide popularity and usage of the Android-based devices. According to Gartner's survey, the Android mobile platform has gained 70% share of the smartphone market at the end of 2012[2]. The application is easy to use and it is equipped with speech synthesis so that the detected object is communicated to the blind people as voice output.

The project comes under the domain Machine Learning which is the part of Artificial Neural Network. Machine Learning concepts makes the system learn on its own from the experiences it gains, without the interference of the external factors. The YOLO (You Only Look Once) algorithm using Convolutional Neural Network is used for the detection purpose. It is a Deep Neural Network concept from Artificial Neural Network. Artificial Neural Network is inspired by the biological concept of Nervous System where the neurons are the nodes that form the network. Similarly, in Artificial Neural Network perceptrons act like the nodes in the network. Artificial Neural Network has three layers that are, Input Layer, Hidden Layer and the output Layer. Deep Learning is the part of the Artificial Neural Network that has multiple Hidden Layer that can be used for the Feature Extraction and Classification purposes. Convolutional Neural Network (CNN) is the part of Deep Learning that is used in analysis

of visual imagery. It has four different kinds of layers, they are, Convolutional Layer, Pooling Layer, Activation Layer and Fully Connected Layer. Convolution Layer uses filter and strides to obtain the Feature Maps. These Feature Maps are the matrix that is obtained after the Convolution Layer. It can be simplified using ReLU (Rectified Linear Unit) that maps negative values to 0. The resulted Feature Map is reduced by sending it into the Pooling Layer where it is reduced to the smaller sized matrix. This is how the features are extracted. At the end of the convolutional neural network is the Fully Connected Layer where the actual Classification occurs.

1.2 Introduction of Domain

1.2.1 Machine Learning

In the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance. Machine Learning involves a high level of generalization in order to get a system that performs well on yet unseen data instances.

Machine learning is a relatively new discipline within Computer Science that provides a collection of data analysis techniques. Some of these techniques are based on well established statistical methods (e.g. logistic regression and principal component analysis) while many others are not.

Most statistical techniques follow the paradigm of determining a particular probabilistic model that best describes observed data among a class of related models. Similarly, most machine learning techniques are designed to find models that best fit data (i.e. they solve certain optimization problems), except that these machine learning models are no longer restricted to probabilistic ones.

Therefore, an advantage of machine learning techniques over statistical ones is that the latter require underlying probabilistic models while the former do not. Even though some machine learning techniques use probabilistic models, the classical statistical techniques are most often too stringent for the oncoming Big Data era, because data sources are increasingly complex and multi-faceted. Prescribing probabilistic models relating variables from disparate data sources that are plausible and amenable to statistical analysis might be extremely difficult if not impossible.

Machine learning might be able to provide a broader class of more flexible alternative analysis methods better suited to modern sources of data. It is imperative for statistical agencies to explore the possible use of machine learning techniques to determine whether their future needs might be better met with such techniques than with traditional ones.

1.2.2 Classes of Machine Learning

There are two main classes of machine learning techniques:

a) **Supervised machine learning**
b) **Unsupervised machine learning**
c) **Reinforcement Learning**

Examples of supervised learning Logistic regression (statistics) vs Support vector machines (machine learning) Logistic regression, when used for prediction purposes, is an example of supervised machine learning. In logistic regression, the values of a binary response variable (with values 0 or 1, say) as well as a number of predictor variables (covariates) are observed for a number of observation units. These are called training data in machine learning terminology. The main hypotheses are that the response variable follows a Bernoulli distribution (a class of probabilistic models), and the link between the response and predictor variables is the relation that the logarithm of the posterior odds of the response is a linear function of the predictors. The response variables of the units are assumed to be independent of each other, and the method of maximum likelihood is applied to their joint probability distribution to find the optimal values for the coefficients (these parameterize the aforementioned joint distribution) in this linear function. The particular model with these optimal coefficient values is called the "fitted model," and can be used to "predict" the value of the response variable for a new unit (or, "classify" the new unit as 0 or 1) for which only the predictor values are known. Support Vector Machines (SVM) are an example of a non-statistical supervised machine learning technique; it has the same goal as the logistic regression classifier just described: Given training data, find the best-fitting SVM model, and then use the fitted SVM model to classify new units. The difference is that the underlying models for SVM are the collection of hyper planes in the space of the predictor variables. The optimization problem that needs to be solved is finding the hyper plane that best separates, in the predictor space, the units with response value 0 from those with response value 1. The logistic regression optimization problem comes from probability theory whereas that of SVM comes from geometry.

Other supervised machine learning techniques mentioned later in this briefing include decision trees, neural networks, and Bayesian networks. B. Examples of unsupervised learning Principal component analysis (statistics) vs Cluster analysis (machine learning). The main example of an unsupervised machine learning technique that comes from classical statistics is principal component analysis, which seeks to "summarize" a set of data points in high-dimensional space by finding orthogonal one-dimensional subspaces along which most of the variation in the data points is captured. The term

“unsupervised” simply refers to the fact that there is no longer a response variable in the current setting. Cluster analysis and association analysis are examples of non-statistical unsupervised machine learning techniques. The former seeks to determine inherent grouping structure in given data, whereas the latter seeks to determine co-occurrence patterns of items

1.3 Objective of the Problem

Object Classification is a principle task in image and video processing. It is exercised over a multitude of applications ranging from test and number classification to traffic surveillance. The primitive machine learning concepts had provided the pedestal for carrying out umber of image processing tasks. Nowadays requirement of detection algorithm is to work end to end and take less time to compute. Real-time detection and classification of objects from video provide the foundation for generating many kinds of analytical aspects such as the amount of traffic in a particular area over the years or the total population in an area. In practice, the task usually encounters slow processing of classification and detection or the occurrence of erroneous detection due to the incorporation of small and lightweight datasets. To overcome these issues, YOLO (You Only Look Once) based detection and classification approach (YOLOv2) for improving the computation and processing speed and at the same time efficiently identify the objects in the video. Classifier such as Haar cascade which uses Haar like features was primitively used for face detection.

Figure 1.1 Objective of the problem

1.4 Scope of the project

The main aim of this project is to build a system that detects objects from the image or a stream of images given to the system in the form of previously recorded video or the real time input from the camera. Bounding boxes will be drawn around the objects that are being detected by the system. The system will also classify the object to the classes the object belongs. Python Programming and a Machine Learning Technique named YOLO (You Only Look Once) algorithm using Convolutional Neural Network is used for the Object Detection.

CHAPTER-2
LITERATURE SURVEY

2.1 Mutual Learning Between Saliency and Similarity: Image Cosegmentation via Tree Structured Sparsity and Tree Graph Matching Yan Ren ; Licheng Jiao ; Shuyuan Yang ; Shuang Wang IEEE 2018.

This paper proposes a unified mutual learning framework based on image hierarchies, which integrates structured sparsity with tree-graph matching to conquer the problem of weakly supervised image cosegmentation. We focus on the interaction between two common-object properties: saliency and similarity. Most existing cosegmentation methods only pay emphasis on either of them. The proposed method realizes the learning of the prior knowledge for structured sparsity with the help of tree-graph matching, which is capable of generating object-oriented salient regions. Meanwhile, it also reduces the searching space and computational complexity of tree-graph matching with the attendance of structured sparsity. We intend to thoughtfully exploit the hierarchically geometrical relationships of coherent objects. Experimental results compared with the state-of-the-arts on benchmark data sets confirm that the mutual learning framework is capable of effectively delineating co-existing object patterns in multiple images.

2.2 SaCoseg: Object Cosegmentation by Shape Conformability Wenbing Tao ; Kunqian Li ; Kun Sun IEEE 2018.

In this paper, an object cosegmentation method based on shape conformability is proposed. Different from the previous object cosegmentation methods which are based on the region feature similarity of the common objects in image set, our proposed SaCoseg cosegmentation algorithm focuses on the shape consistency of the foreground objects in image set. In the proposed method, given an image set where the implied foreground objects may be varied in appearance but share similar shape structures, the implied common shape pattern in the image set can be automatically mined and regarded as the shape prior of those unsatisfactorily segmented images. The SaCoseg algorithm mainly consists of four steps: 1) the initial Grabcut segmentation; 2) the shape mapping by coherent point drift registration; 3) the common shape pattern discovery by affinity propagation clustering; and 4) the refinement by Grabcut with common shape constraint. To testify our proposed

algorithm and establish a benchmark for future work, we built the CoShape data set to evaluate the shape-based cosegmentation. The experiments on CoShape data set and the comparison with some related cosegmentation algorithms demonstrate the good performance of the proposed SaCoseg algorithm.

2.3 Clothing Cosegmentation for Shopping Images With Cluttered Background Bo Zhao ; Xiao Wu ; Qiang Peng ; Shuicheng Yan IEEE 2018.

In this paper, we address an important and practical problem of clothing cosegmentation (CCS): given multiple fashion model photos with natural backgrounds on e-commerce websites, to automatically and simultaneously segment all images and extract the clothing regions. However, cluttered backgrounds, variations in colors and styles, and inconsistent human poses all make it a challenging task. In this paper, a novel CCS algorithm is proposed to improve the accuracy of clothing extraction by exploiting the properties of multiple clothing images with the same apparel. First, the co-salient objects are computed by detecting the upper bodies of fashion models and transferring their locations within multiple images. Based on the coarse clothing regions determined by the upper body localization and co-salient object detection, the foreground (clothing) and background Gaussian mixture models are estimated, respectively. Finally, the clothing region in each image is extracted through energy minimization based on graph cuts iteratively. The proposed cosegmentation algorithm is mainly designed for multiple clothing images. As a byproduct, it can also be applied to single image segmentation without any modification. The experiments demonstrate that the proposed approach outperforms the state-of-the-art cosegmentation methods as well as traditional single image segmentation solution for shopping images.

2.4 Cosegmentation for Object-Based Building Change Detection From High-Resolution Remotely Sensed Images Pengfeng Xiao ; Min Yuan ; Xueliang Zhang ; Xuezhi Feng ; Yanwen Guo IEEE 2018.

This paper presents a cosegmentation-based method for building change detection from multitemporal high-resolution (HR) remotely sensed images, providing a new solution to object-based change detection (OBCD). First, the magnitude of a difference image is calculated to represent the change feature. Next, cosegmentation is performed via graph-based energy minimization by combining the change feature with image features at each phase, directly resulting in foreground as multitemporal changed objects and background as unchanged area. Finally, the spatial correspondence between

changed objects is established through overlay analysis. Cosegmentation provides a separate and associated, rather than a separate and independent, multitemporal image segmentation method for OBCD, which has two advantages: 1) both the image and change features are used to produce foreground segments as changed objects, which can take full advantage of multitemporal information and produce two spatially corresponded change detection maps by the association of the change feature, having the ability to reveal the thematic, geometric, and numeric changes of objects and 2) the background in the cosegmentation result represents the unchanged area, which naturally avoids the problem of matching inconsistent unchanged objects caused by the separate and independent multitemporal segmentation strategy. Experimental results on five HR datasets verify the effectiveness of the proposed method and the comparisons with the state-of-the-art OBCD methods further show its superiority.

CHAPTER-3

PROBLEM STATEMENT AND METHODOLOGY

3.1 Problem Definition

The main advances in object detection were achieved thanks to improvements in object representations and machine learning models. A prominent example of a state-of-the-art detection system is the Deformable Part-based Model (DPM). It builds on carefully designed representations and kinematically inspired part decompositions of objects, expressed as a graphical model. Using discriminative learning of graphical models allows for building high-precision part-based models for variety of object classes. Manually engineered representations in conjunction with shallow discriminatively trained models have been among the best performing paradigms for the related problem of object classification as well. In the last years, however, Deep Neural Networks (DNNs) have emerged as a powerful machine learning model. DNNs exhibit major differences from traditional approaches for classification. First, they are deep architectures which have the capacity to learn more complex models than shallow ones. This expressivity and robust training algorithms allow for learning powerful object representations with- out the need to hand design features. This has been empirically demonstrated on the challenging ImageNet classification task across thousands of classes.

3.2 Methodology

3.2.1 Classification-Based Segmentation

In classification-based segmentation, voxels are classified and labeled as belonging to a particular tissue class according to a certain criterion. The simplest technique is based on thresholding. Thresholding algorithm attempts to determine a threshold value which separates the desired classes. Iterative thresholding used to distinguish brain tissues from others in axial IMAGE slices. Starting at set values, thresholds for the head and the image are then iteratively adjusted based on the geometry of resulting masks. Although thresholding algorithm is simple and computationally very fast, it is very sensitive to INU artifact and noise in IMAGE images. The automatic determination of a suitable threshold could be problematic if there is severe overlap between the intensities of different tissue types due to noise and intensity in homogeneities.

Instead of using simple thresholding in earlier classification-based segmentation work, statistical classification based segmentation has been the method of choice in more recent time. Statistical classification has the advantage of being more robust, as well as having a rigorous mathematical foundation in stochastic theory. In statistical classification methods, the probability density function of tissue intensity for different tissue classes are often modeled parametrically as a mixture of Gaussians, usually one Gaussian function per tissue class. In order to incorporate local contextual information, IMAGE regularization is often employed as well. The bias field estimation problem is cast in a Bayesian framework and the expectation-maximization (EM) algorithm is used to estimate the inhomogeneity and the tissue classes. However, their method needs to be supplied with the tissue class conditional intensity models, which are typically constructed manually from training data. They also did not consider neighborhood dependencies for the tissue segmentation. algorithm by using IMAGE to introduce context or dependency among neighboring voxels. Propose to use a 3-step EM algorithm, which interleaves voxel classification, class distribution parameter estimation, and bias field estimation. Instead of using manually constructed tissue class conditional intensity models, their method employs digital brain atlas with a priori probability maps for each tissue class to automatically construct intensity models for each individual scan being processed. The image tissue classes are modeled as finite Gaussian mixtures with MRF regularization to account for contextual information and the bias field is modeled as a fourth order least square polynomial fit. It also use the Gaussian mixture to model the three brain tissue classes. The biological variations of a particular tissue class are accounted for in their statistical model by assuming that the mean intensities of the tissue classes are slowly varying spatial functions. The magnetic field in homogeneities modify both the mean tissue intensities and the noise variances in a similar manner. To account for the smoothness and piecewise contiguous nature of the tissue regions, they use a 3D MRF as a prior. Consider the statistical segmentation of multispectral IMAGE image. In their work, the intensity distributions of the brain tissues are again modeled as a mixture of Gaussians. Another major class of voxel classification techniques uses custering-based method. Clustering is a popular unsupervised classification method and has found many applications in pattern classification and image segmentation. Clustering algorithm attempts to classify a voxel to a tissue class by using the notion of similarity to the class.

3.2.2 Region-Based Segmentation

The shape of an object can be described in terms of its boundary or the region it occupies. Image region belonging to an object generally have homogeneous characteristics, e.g. similar in

intensity or texture. Region-based segmentation techniques attempt to segment an image by identifying the various homogeneous regions that correspond to different objects in an image. Unlike clustering methods, region-based methods explicitly consider spatial interactions between neighboring voxels. In its simplest form, region growing methods usually start by locating some seeds representing distinct regions in the image . The seeds are then grown until they eventually cover the entire image. The region growing process is therefore governed by a rule that describe the growth mechanism and a rule that check the homogeneity of the regions at each growth step. Region growing technique has been applied to IMAGE segmentation. A semi-automatic, interactive IMAGE segmentation algorithm was developed that employ simple region growing technique for lesion segmentation. In , an automatic statistical region growing algorithm based on a robust estimation of local region mean and variance for every voxel on the image was proposed for IMAGE segmentation. The best region growing parameters are automatically found via the minimization of a cost functional. Furthermore, relaxation labeling, region splitting, and constrained region merging were used to improve the quality of the IMAGE segmentation. The determination of an appropriate region homogeneity criterion is an important factor in region growing segmentation methods. However, such homogeneity criterion may be difficult to obtain a priori. An adaptive region growing method is proposed where the homogeneity criterion is learned automatically from characteristics of the region to be segmented while searching for the region.

3.2.3 Split-and-merge based segmentation

In the split-and-merge technique, an image is first split into many small regions during the splitting stage according to a rule, and then the regions are merged if they are similar enough to produce the final segmentation.

3.2.4 Watershed-based segmentation

In the watershed-based segmentation, the gradient magnitude image is considered as a topographic relief where the brightness value of each voxel corresponds to a physical elevation. An immersion based approach is used to calculate the watersheds. The operation can be described by imagine that holes are pierced in each local minimum of the topographic relief. Then, the surface is slowly immersed in water, which causes a flooding of all the catchment basins, starting from the basin associated with the global minimum. As soon as two catchment basins begin to merge, a dam is built. The procedure results in a partitioning of the image in many catchment basins of which the borders

define the watersheds. To reduce over-segmentation, the image is smoothed by 3D adaptive anisotropic diffusion prior to watershed operation. Semi-automatic merging of volume primitives returned by the watershed operation is then used to produce the final segmentation.

3.2.5 Contour-Based Segmentation

Contour-based segmentation approach assumes that the different objects in an image can be segmented by detecting their boundaries. Whereas region-based techniques attempt to capitalize on homogeneity properties within regions in an image, boundary-based techniques rely on the gradient features near an object boundary as a guide. Hence, contourbased segmentation methods that rely on detecting edges in the image is inherently more prone to noise and image artifacts. Sophisticated pre- and post-processing is often needed to achieve a satisfactory segmentation result.

3.2.6 Edge detection segmentation

Image segmentation based on edge detection has been proposed , where a combination of Marr-Hildreth operator for edge detection and morphological operations for the refinement of the detected edges is used to segment 3D MR images. A boundary tracing method is proposed, where the operator clicks a pixel in a region to be outlined and the method then finds the boundary starting from that point. The method is, however, restricted to segmentation of large, well defined structures, but not to distinguish fine tissue types. Edge-based segmentation methods usually suffer from over or under-segmentation, induced by improper threshold selection . In addition, the edges found are usually not closed and complicated edge linking techniques are further required.

3.2.7 Active contour based segmentation

Active contour deforms to fit the object's shape by minimizing (among others) a gradient dependent attraction force while at the same time maintaining the smoothness of the contour shape. Thus, unlike edge detection, active contour methods are much more robust to noise as the requirements for contour smoothness and contour continuity act as a type of regularization. Another advantage of this approach is that prior knowledge about the object's shape can be built into the contour parameterization process. However, active contour based algorithms usually require initialization of the contour close to the object boundary for it to converge successfully to the true boundary. More importantly, active contour methods have difficulty handling deeply convoluted boundary such as CSF, GM and WM boundaries due to their contour smoothness requirement. Hence,

they are often not appropriate for the segmentation of brain tissues. Nevertheless, it has been applied successfully to the segmentation of intracranial boundary , brain outer surface , and neuro-anatomic structures in images.

3.2.8 Feature Extraction

Feature extraction is done after the preprocessing phase in character recognition system. The primary task of pattern recognition is to take an input pattern and correctly assign it as one of the possible output classes. This process can be divided into two general stages: Feature selection and Classification. Feature selection is critical to the whole process since the classifier will not be able to recognize from poorly selected features. Criteria to choose features given by Lippman are: “Features should contain information required to distinguish between classes, be insensitive to irrelevant variability in the input, and also be limited in number, to permit, efficient computation of discriminant functions and to limit the amount of training data required” Feature extraction is an important step in the construction of any pattern classification and aims at the extraction of the relevant information that characterizes each class. In this process relevant features are extracted from objects/ alphabets to form feature vectors. These feature vectors are then used by classifiers to recognize the input unit with target output unit. It becomes easier for the classifier to classify between different classes by looking at these features as it allows fairly easy to distinguish. Feature extraction is the process to retrieve the most important data from the raw data. Feature extraction is finding the set of parameter that define the shape of a character precisely and uniquely. In feature extraction phase, each character is represented by a feature vector, which becomes its identity. The major goal of feature extraction is to extract a set of features, which maximizes the recognition rate with the least amount of elements and to generate similar feature set for variety of instance of the same symbol. The widely used feature extraction methods are Template matching, Deformable templates, Unitary Image transforms, Graph description, Projection Histograms, Contour profiles, Zoning, Geometric moment invariants, Zernike Moments, Spline curve approximation, Fourier descriptors, Gradient feature and Gabor features.

3.3 Existing System

Numerous researchers and experts have done many prominent and effective researches and given various effective approaches. We all know that the key point related to co-segmentation is to automatically extract common/multi-class information from multiple images by forcing the segments to be consistent. Early research focused on performing pixel-level segmentation and concentrate on

improving the global energy term and corresponding optimization methods. Rother et al, was the first researcher who put forward the term cosegmentation, which received a lot of attention. This method used the color histogram matching in cooperate with corresponding additional constraint energy into the MRF framework. In this method segmentation is done for common objects through adding foreground similarity constraint into traditional MRF based segmentation methods. L1-norm was used to present the foreground similarity, and the co-segmentation energy. Mukherjee et al. replaced L1 by L2-norm and the Pseudo-Boolean optimization was used for the minimizatio. Later on, other types of features have been also utilized to exploit the relationship between image foregrounds information, such as SIF and Gabor features, but focused on segmenting a pair of images with one common object. After this, many researches in have been done to simplify the optimization by using other types of histogram consistency terms and extended the application as well but color histogram consistency based method limits their application because these approaches only suitable for the targets which share the same color distribution. To address the co-segmentation of multiple images, set up the co-segmentation task as a discriminative clustering problem by clustering the image pixels into foreground and background. Vicente in formulated to extract objects from a group of images by using an object recognition scheme to generate a pool of object like segmentations, and then selecting the most likely segmentations using a learned pair wise consistency term. In contrast, Chang et al. proposed an MRF optimization model, by introducing a co-saliency prior as a hint about possible consistent foreground locations. The proposed model was then optimized using graph-cut techniques. Rubio et al. proposed a method based on establishing correspondences between regions in the images, and then estimating the appearance distributions of both the foreground and the background for better joint segmentation. It was later extended for multiple images containing common object in by using more effective and ideal approaches enforcing inter-image consistency

3.4 Proposed System

Ranking of Segmentation Easiness An image is easy to segment if the foreground stands out from the homogeneous background. For such images, there should be a clean separation between foreground and background with clear boundaries, and the resultant segments should contain complete foreground objects. This paper proposes a saliency-based continuous measure for segmentation easiness Rsal as follows: Rsal = P i∈fg S(i) P i S(i) , (1) where P i∈fg S(i) is the sum of saliency scores over a foreground region, and P i S(i) is the sum over the whole image. The saliency score of every

image region S(i) is estimated via a global contrast saliency score. This score is based on the region's color contrast with respect to the whole image, with weighted sum contributions from the neighboring regions. Subsequently, more salient regions are segmented out using a graph cut. Upon segmentation, the saliency ranking Rsal is computed. An image with a high Rsal score should be easy to segment. Segmentation Propagation Simple images can be readily segmented to produce good segmentation masks due to a clear separation between foreground and background in these images. The well segmented object masks are then propagated to more difficult images as a segmentation prior.

Even in some images that may not be well segmented, the results can be further improved by passing them to the propagation step. The propagation step is elaborated as follows. Let the image set be {I1, I2, ..., It, It+1, ...}, where Ik is an image according to our ranking Rsal. Images in {I1, ..., It} have been segmented, and image It+1 is the next to be segmented. The object in image It+1 may not be as salient and the background is more cluttered. This is where the well segmented images can help by propagating the segmentation masks to the similar unsegmented object regions. Since multiple objects could exist in an image, we extract possible object patches from image It+1 forcomparison.

The image patches are extracted based on objectness as defined in , and they may overlap. Upon extraction, each patch is then matched to the closest K patches in the segmented set {I1, ..., It}. The resultant segmentation prior of patch x in image It+1 is defined as follows: P(x) = 1 K X K l=1 exp −d 2 (x, l)/2σ 2 , (2) where P(x) is the prior probability of patch x being in the foreground, d(x, l) is the distance between patches x and l, and σ is a parameter to set. The patch distance d(x, l) is computed based on their corresponding GIST features. In this manner, every pixel on the test patch will have a probability of being in the foreground and being in the background. Segmentation with Prior Information After propagating segmentation masks, we then segment image It+1 via a graph cut, which solves the following energy minimization problem: E(L) = X i U(Li) +X i,j V (Li , Lj), (3) where E(L) is the energy to minimize, U(Li) is the unary potential of pixel i being labelled as Li , and V (Li , Lj) is the potentials term modeling the spatial coherence between two neighboring pixels i and j. The unary potentials term is defined as U(Li) = − P k log(P(Li |Ck)P(Ck)), where P(Li |Ci) is the probability of pixel i belonging to class k, k ∈ {0, 1}, and P(Ck) is the prior probability of class k computed from (2). P(Li |Ci) is computed based on a Gaussian mixture model. The pair-wise potentials are defined as: V (Li , Lj) ∝ d(i, j) −1 exp −γ X k=R,G,B |Ii(k) − Ij (k)| 1 , where d(i, j) is the pixel spatial distance and |Ii(k) − Ij (k)| 1 is the intensity difference across RGB channels, and

γ is a constant. The minimization and pixels labelling are carried out iteratively until there is no change in pixel labels.

3.4.1 System Architecture

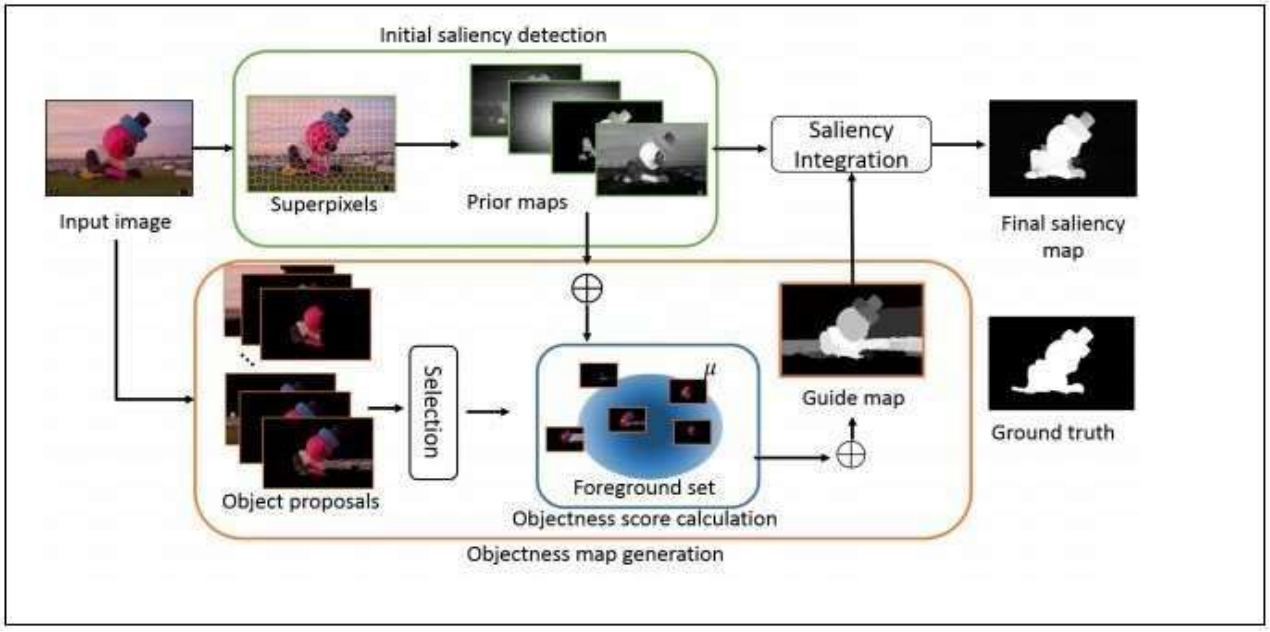


Figure 3.1 System architecture

3.4.2 Advantages of Proposed System

- Real time video surveillance.
- Using open cv.
- More speed and accuracy

CHAPTER - 4
SYSTEM STUDY

4.1 Feasibility Study

In successful business feasibility study of projects is a critical factor. Sometimes we noticed that several projects be unsuccessful because of their incorrect facts or incorrect assumptions. Therefore, the basic features in any feasibility study must be ensuring that we are working with accurate information, exact statement, and the latest financial records. The main objective of feasibility study is to evaluate three types of feasibility; technical, operational and economic feasibility. In this stage, only the project cost of the development and implementation cost of the project is considered. Feasibility study may also depend on the numerous factors, like the risks of the project before and after project and potential returns of the investments. It is in simply verify the changes before the project design. Importance of performing a Feasibility Study. Some major benefits of conducting a feasibility study are the following. Conduct a feasibility study is always important because of it gives a clear idea of the dreamed project. It helps to identify the valid and proper reason to select the project. With the support of feasibility study project teams' members become more focused. It helps to decision-making on the project. Improve the achievement time by calculating several restrictions.

General criteria:

- The user's demonstrable requirements and how the proposed system meets them?
- What are possible resources are available and the present project utilized theseresources?
- Likely impact of the proposed system on the organization and it is really fit with the organization.

The proper investigation and evaluation procedure of the proposed project is mainly identification of the possible problem, specification, expected performance and, the cost of each sub-system and selection of the best and efficient procedures. The outcome of the study is generally considered as formal proposal of the expected system. This is basically a report about the scope of the projected solution. The proposals are summarizing with the expected so. It consists of the following:

1. Brief statement of the proposed project –a carefully worded and may be the graphical statement of the proposed problem.

Summary of the outcome - a list of the major outcome from the study.

2. Details of findings- an outline of the proposed methods and procedures of the desired plan undertaken to complete task with all possible finding. This report also included the possible structures, costs estimation and benefits of the desire projects.
3. Recommendations and final conclusions- Few specific advice and recommendations based on its outcome along with the personnel assignments, projected costs and target dates. Basically, it is a predictable project that demonstrate conclusion of the assessment, examine and evaluation of a designed method and terminate if thismethod is accurately feasible, profitable and productive.

4.1.1 Technical Feasibility

Technical feasibility mainly associated with the technologically evaluates the project. In this subject area generally a group of engineers or technical expert study the whole projects and technical aspects. This study facilitates said organizations to proper assess. The industrial possessions may assemble capability. Based on the results it decides whether the technical team can convert the idea into real.

4.1.2 Economic Feasibility

Economic feasibility study related with price, and all kind of expenditure related with the scheme before the project start. This study also improves project reliability. It is also helpful for the decision-makers to decide the planned scheme processed latter or now, depending financial condition of the organization. This evaluation process also studies the price benefits of the proposed scheme.

4.1.3 Operational Feasibility

Operational Feasibility may employ the responsibility to examine and decide whether the proposed methods fulfil all kind of business requirements. Its actions forecast all possible schemes to recognized and resolves troubles. These studies may also examine and verify how the project planned guarantee the method development is feasible or not.

4.1.4 Social Feasibility

Social feasibility is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.

His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 5

SYSTEM DESIGN

5.1 System Architecture

- **Raspberry Pi 3 Model B+**

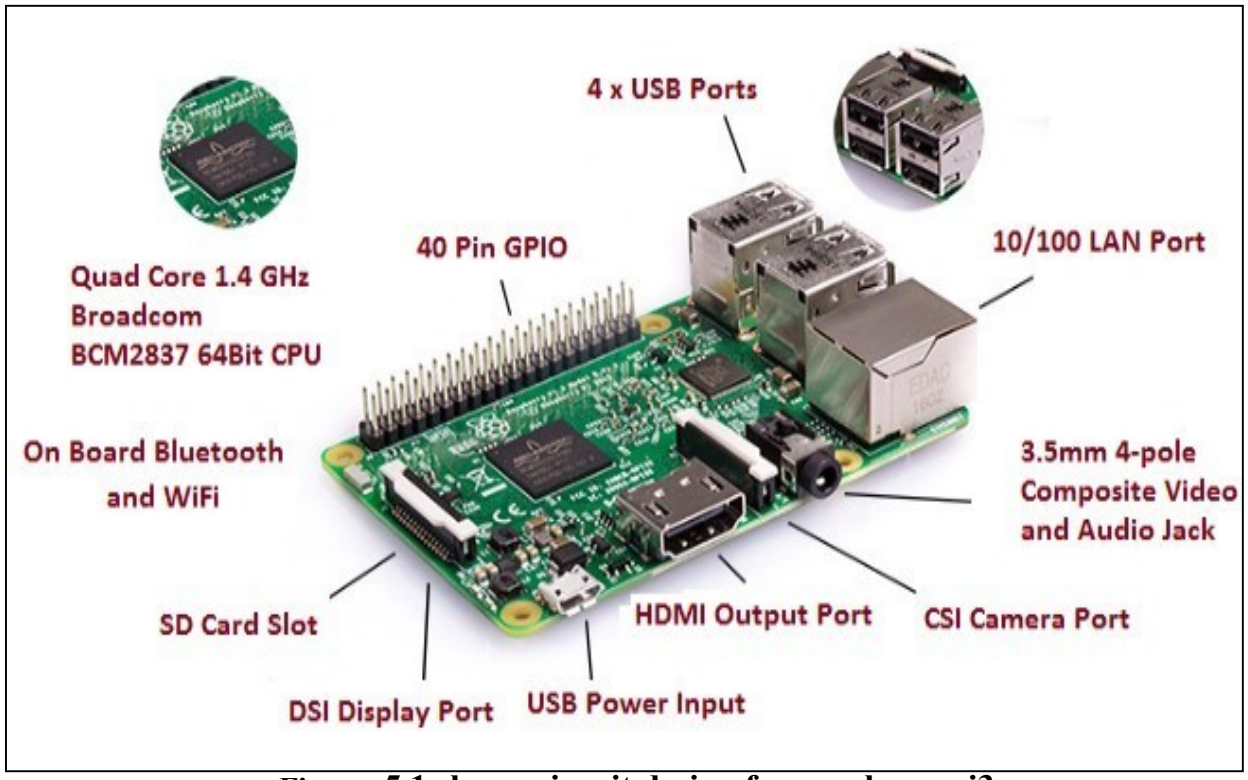


Figure 5.1 shows circuit design for raspberrypi3

5.1.1 Technical Specification:

- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2GHz
- 1GB RAM
- BCM43143 Wi-Fi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screendisplay
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)

5.1.2 Block Diagram

A **block diagram** is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic devise, software design, and process flow diagram. Block diagrams are typically used for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation. Contrast this with the schematic diagram and layout diagram used in electrical engineering, which show the implementation details of electrical components and physical construction.

Block diagrams derive their name from the rectangular elements found in this type of diagram. They are used to describe hardware and software systems as well as to represent processes. Block diagrams are described and defined according to their function and structure as well as their relationship with other blocks.

Block diagrams are generally used when the visualization of information or control flows is important – or when processes are involved. In this way we can represent complex algorithms or flows of information or communication among individual components within a large system as with, for example, in a facility designed for mass production. A graphical representation is often easier to understand than a textual representation.

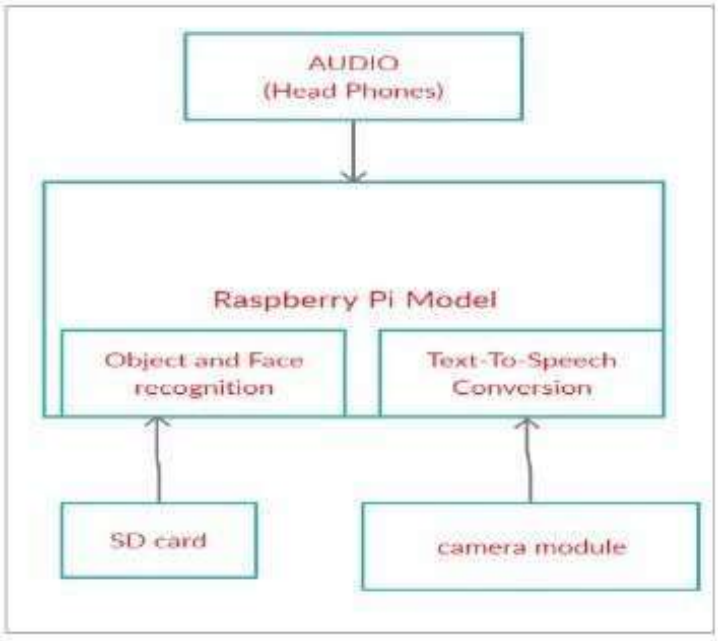


Figure 5.2 Shows the Block Diagram for Object Detection

5.2 Flow Chart Diagram

A simple flowchart representing a process for dealing with a non-functioning lamp. A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields.

Flowcharts are used in designing and documenting simple processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps find flaws, bottlenecks, and other less-obvious features within it. There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

- A processing step, usually called activity, and denoted as a rectangular box
- A decision usually denoted as a diamond.
- The start and ending process of flowchart is denoted by oval
- The flow of the process in the flowchart is indicated by Arrows

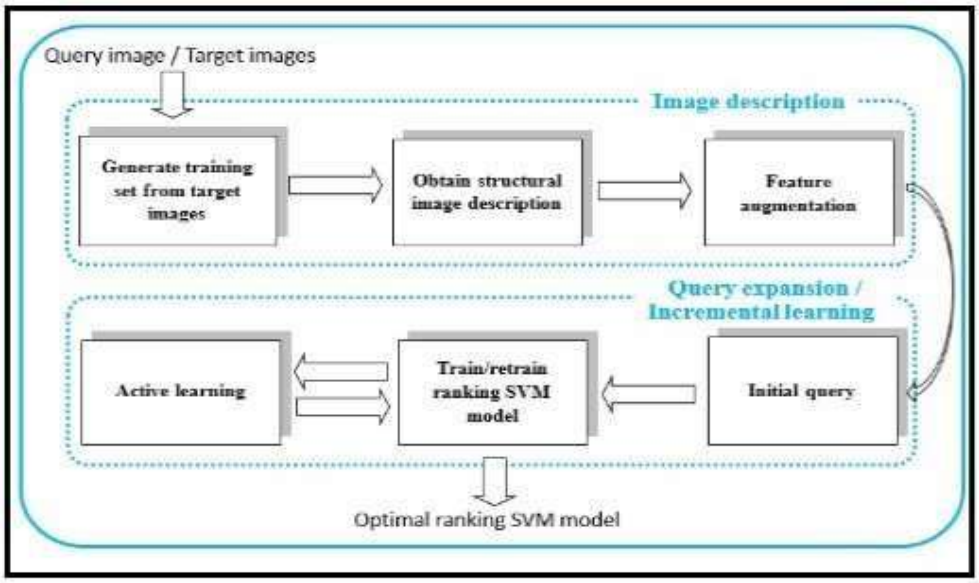


Figure 5.3 flow chart for Object Detection

5.3 Uml Diagrams

UML **(Unified Modelling Language)** is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

5.3.1 Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as **actors**. Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities.

They also help identify any internal or external factors that may influence the system and should be taken into consideration.

They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

Various Notations in Use Case Diagram:

Use-case:

Use cases are used to represent high-level functionalities and how the user will handle the system. A use case represents a distinct functionality of a system, a component, a package, or a class. It is denoted by an oval shape with the name of a use case written inside the oval shape.

Actor:

It is used inside use case diagrams. The actor is an entity that interacts with the system. A user is the best example of an actor.

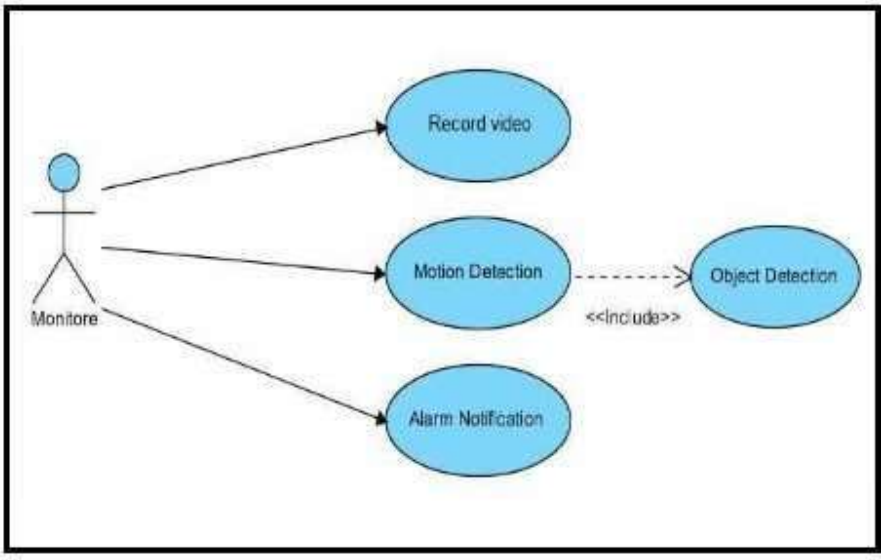


Figure 5.4 use case for Object Detection

5.3.2 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

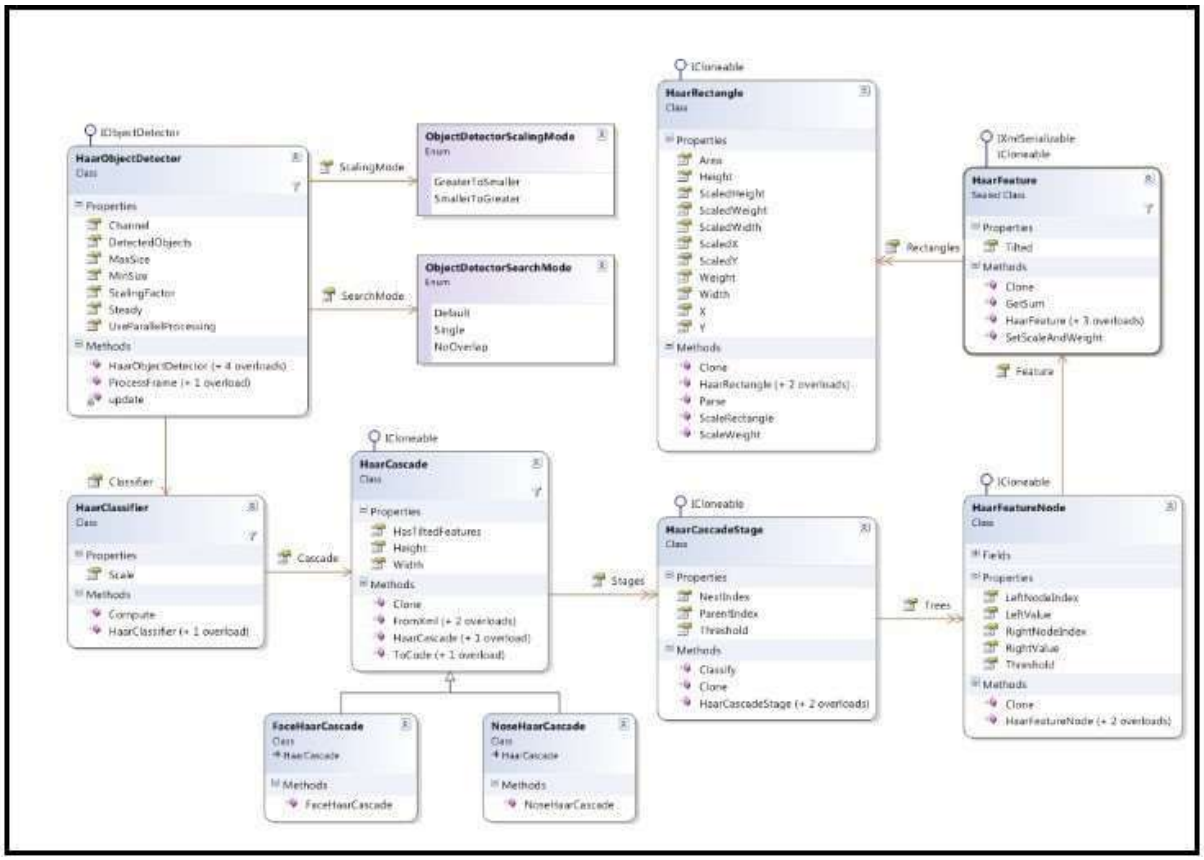


Figure 5.5 Class Diagram for Object Detection

5.3.3 Sequence Diagram

A **sequence diagram** is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another. Interaction among the components of a system is very important from implementation and execution perspective.

Sequence diagrams describe interactions among classes. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system

Various Notations in Sequence Diagrams:

Class Roles or Participants:

Class roles describe the way an object will behave in context.

Activation or Execution Occurrence

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, we use a thin gray rectangle placed vertically on its lifeline.

Messages: Messages are arrows that represent communication between objects. We use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

Lifelines: Lifelines are vertical dashed lines that indicate the object's presence over time.

Destroying Objects: Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, we can place an X at the end of its lifeline to denote a destruction occurrence.

Loops: A repetition or loop within a sequence diagram is depicted as a rectangle. We place the condition for exiting the loop at the bottom left corner in square brackets [ ].

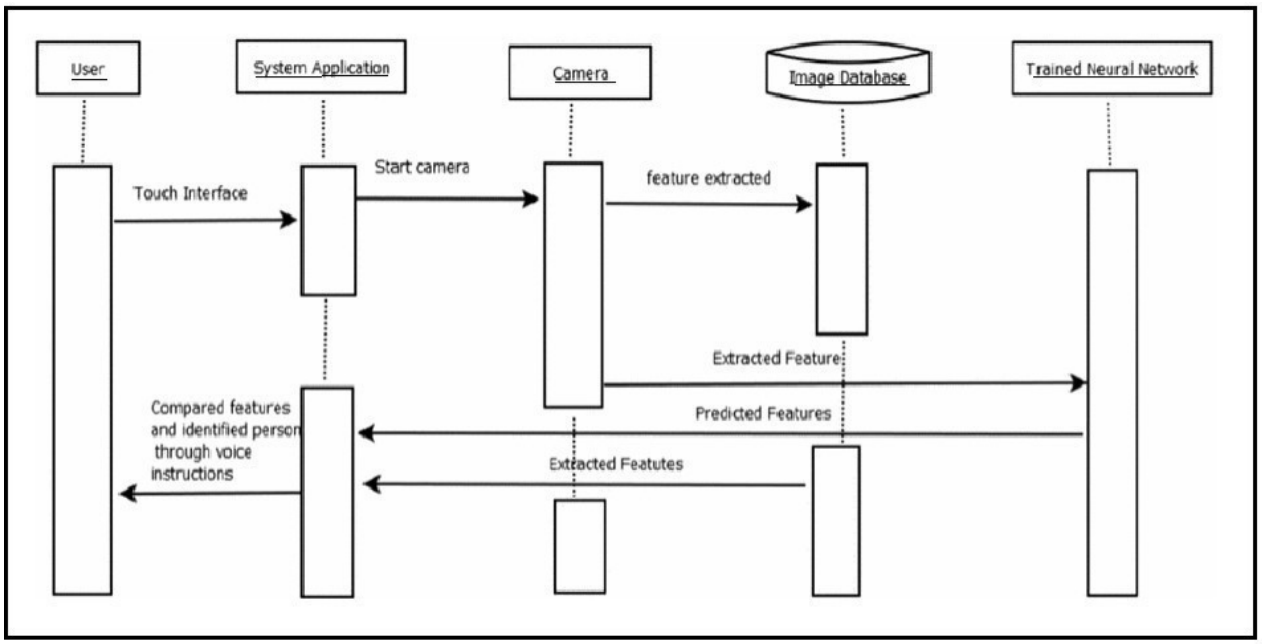


Figure 5.6 Sequence Diagram for Object Detection

5.3.4 Activity Diagram

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

Various Notations in Activity Diagram:

Initial State or Start Point

A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swimlanes, make sure the start point is placed in the top left corner of the first column.

Activity or Action State

An action state represents the non-interruptible action of objects. We can draw an action state in Smart Draw using a rectangle with rounded corners.

Action Flow

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.

Decisions and Branching

A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."

Guards

In UML, guards are a statement written next to a decision diamond that must be true before moving next to the next activity. These are not essential, but are useful when a specific answer, such as "Yes, three labels are printed," is needed before moving forward.

Synchronization

A fork node is used to split a single incoming flow into multiple concurrent flows. It is represented as a straight, slightly thicker line in an activity diagram.

A join node joins multiple concurrent flows back into a single outgoing flow.

A fork and join mode used together are often referred to as synchronization.

Time Event

This refers to an event that stops the flow for a time; an hourglass depicts it.

Merge Event

A merge event brings together multiple flows that are not concurrent.

Sent and Received Signals

Signals represent how activities can be modified from outside the system. They usually appear in pairs of sent and received signals, because the state can't change until a response is received, much like synchronous messages in a sequence diagram. For example, an authorization of payment is needed before an order can be completed.

Interrupting Edge

An event, such as a cancellation, that interrupts the flow denoted with a lightning bolt.

Swimlanes

Swimlanes group related activities into one column.

Final State or End Point

An arrow pointing to a filled circle nested inside another circle represents the final action state.

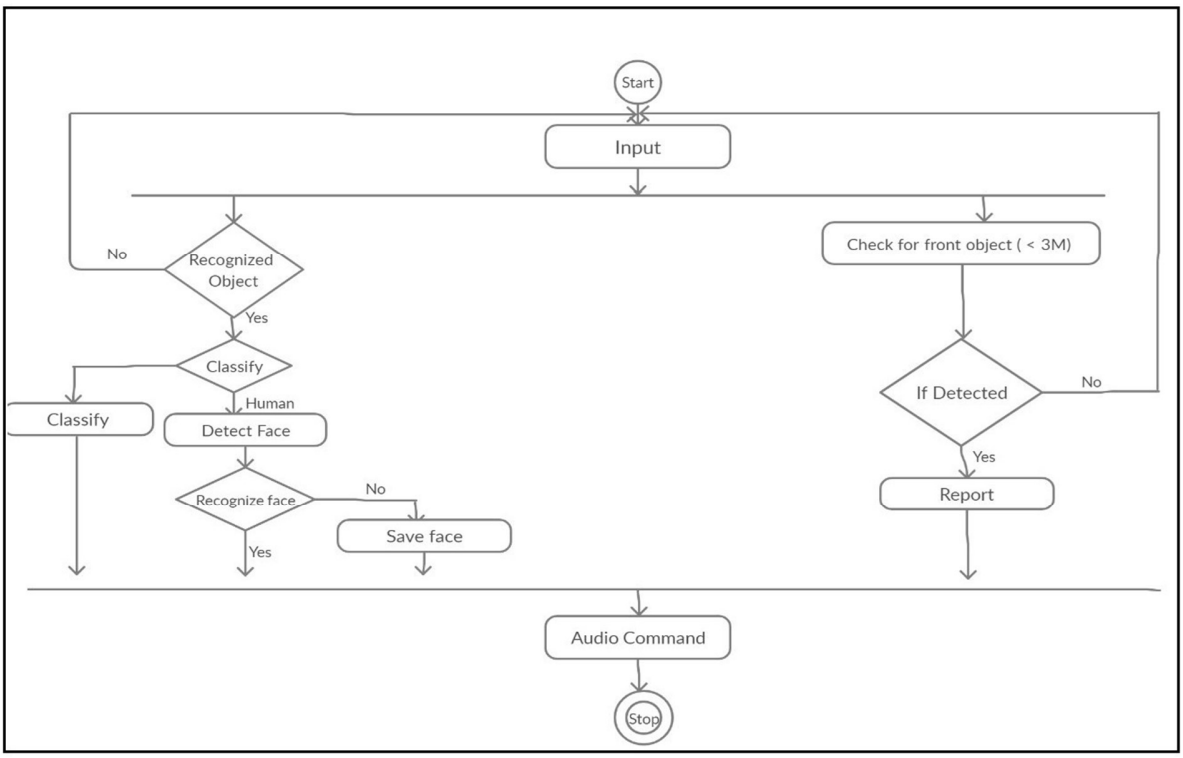


Figure 5.7 Activity Diagram for Object Detection

CHAPTER 6
SOFTWARE ENVIRONMENT

6.1 Python Description:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python - Environment Setup

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS
- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual machines

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python **h ttps://www.python.org/**

You can download Python documentation from **https://www.python.org/doc/**. The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to **https://www.python.org/downloads/.**
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run ./configure script
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to **https://www.python.org/downloads/.**
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.

- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix −

- **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
- **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.
- **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
- **Note** − /usr/local/bin/python is the path of the Python directory

Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

At the command prompt − type path %path%;C:\Python and press Enter.

Note − C:\Python is the path of the Python directory

Python Environment Variables

Here are important environment variables, which can be recognized by Python −

Sr.No.	Variable & Description
1	**PYTHONPATH** It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer.
2	**PYTHONSTARTUP** It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.
3	**PYTHONCASEOK** It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.
4	**PYTHONHOME** It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

Running Python

There are three different ways to start Python −

Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

```
$python # Unix/Linux
or
python% # Unix/Linux
or
C:> python # Windows/DOS
```

Here is the list of all the available command line options −

Sr.No.	Option & Description
1	**-d** It provides debug output.
2	**-O** It generates optimized bytecode (resulting in .pyo files).
3	**-S** Do not run import site to look for Python paths on startup.
4	**-v** verbose output (detailed trace on import statements).
5	**-X** disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6.
6	**-c cmd** run Python script sent in as cmd string
7	**file** run Python script from given file

Script from the Command-line

A Python script can be executed at command line by invoking the interpreter on your application, as in the following −

```
$python script.py # Unix/Linux

or

python% script.py # Unix/Linux

or

C: >python script.py # Windows/DOS
```

Note − Be sure the file permission mode allows execution.

Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

- **Unix** − IDLE is the very first Unix IDE for Python.
- **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
- **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

Note − All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

PANDA

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including

finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Audience

This tutorial has been prepared for those who seek to learn the basics and various functions of Pandas. It will be specifically useful for people working with data cleansing and analysis. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus. Pandas library uses most of the functionalities of NumPy. It is suggested that you go through our tutorial on NumPy before proceeding with this tutorial. You can access it from Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.

- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Audience

This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.umPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.

- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.tandard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**.

```
pip install numpy
```

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python)

6.2 R-CNN Understanding Region-Based Convolutional Neural Network

To know more about the selective search algorithm, follow this link. These 2000 candidate region proposals are warped into a square and fed into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM to classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values to increase the precision of the bounding box. For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could've been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.

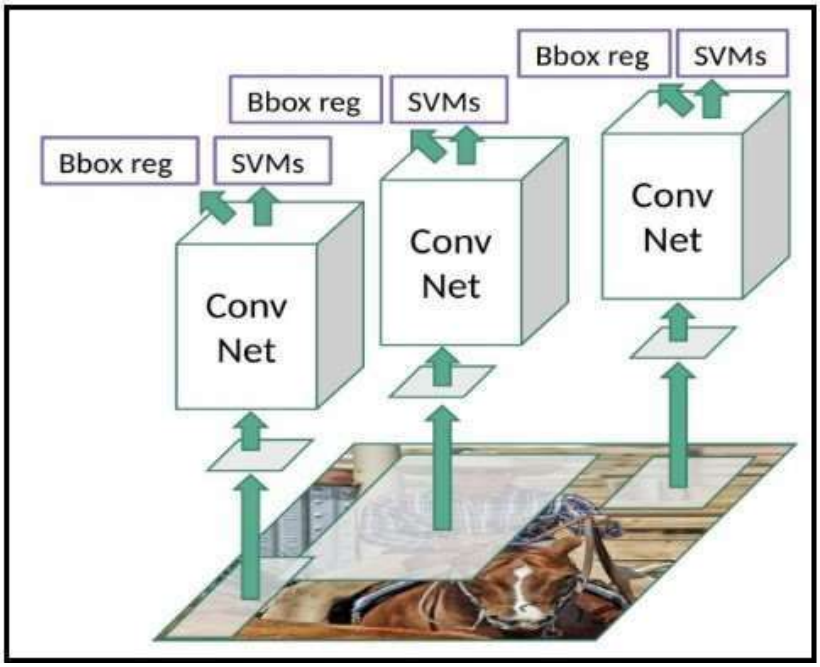


Figure 6.1 R-CNN network structure

6.2.1 Intuition of RCNN

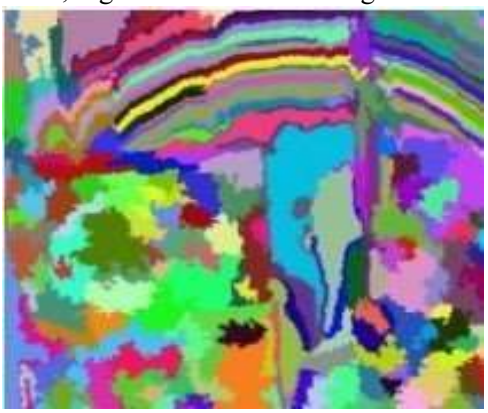
Instead of working on a massive number of regions, the RCNN algorithm proposes a bunch of boxes in the image and checks if any of these boxes contain any object. RCNN uses selective search to extract these boxes from an image (these boxes are called regions).

Let's first understand what selective search is and how it identifies the different regions. There are basically four regions that form an object: varying scales, colors, textures, and enclosure. Selective search identifies these patterns in the image and based on that, proposes various regions. Here is a brief overview of how selective search works:

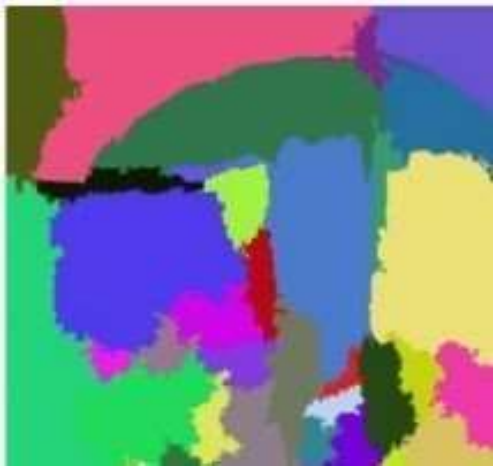
Steps:

- It first takes an image as input:

- Then, it generates initial sub-segmentations so that we have multiple regions from this



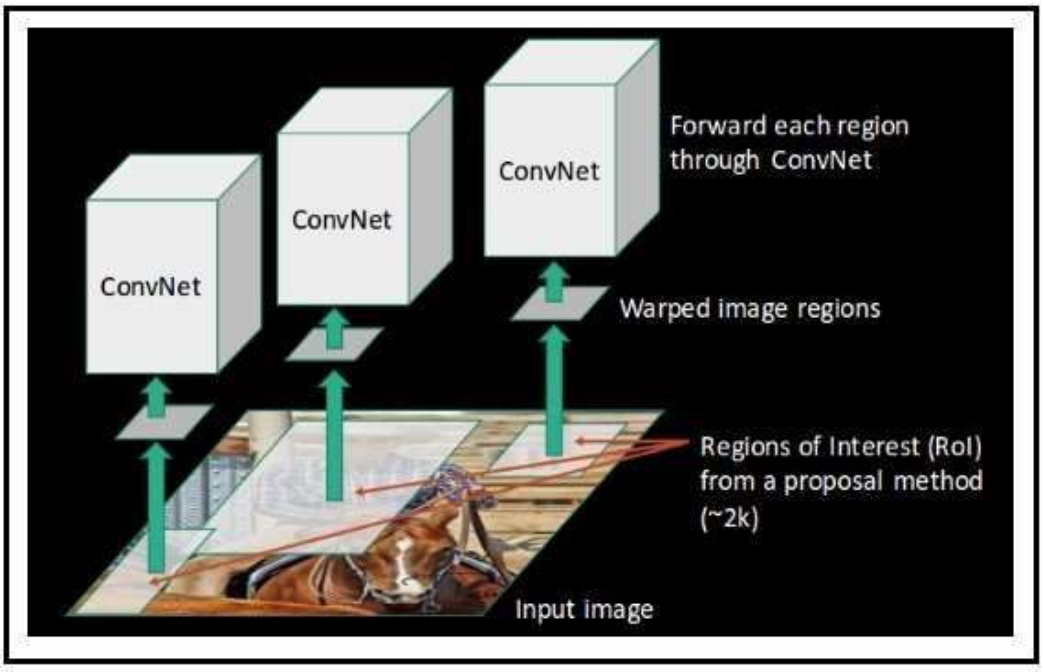
- The technique then combines the similar regions to form a larger region (based oncolor similarity, texture similarity, size similarity, and shape compatibility):



- Finally, these regions then produce the final object locations (Region of Interest).
- Below is a succint summary of the steps followed in RCNN to detect objects:
- We first take a pre-trained convolutional neural network.
- Then, this model is retrained. We train the last layer of the network based on the number of classes that need to be detected.
- The third step is to get the Region of Interest for each image. We then reshape all these regions so that they can match the CNN input size.
- After getting the regions, we train SVM to classify objects and background. For each class, we train one binary SVM.
- Finally, we train a linear regression model to generate tighter bounding boxes for each identified object in the image.
- You might get a better idea of the above steps with a visual example (Images for the example shown below are taken from this paper) . So let's take one!
- First, an image is taken as an input:

- Then, we get the Regions of Interest (ROI) using some proposal method (for example, selective search as seen above):

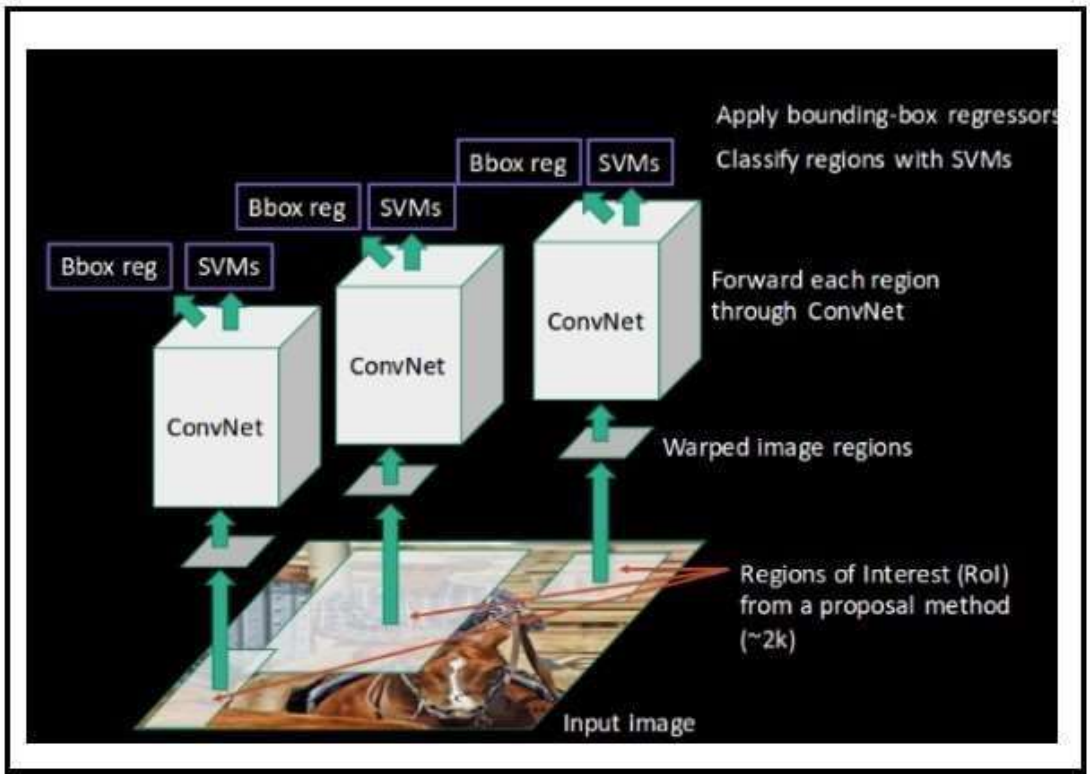
- All these regions are then reshaped as per the input of the CNN, and each region is passed to the ConvNet:



- CNN then extracts features for each region and SVMs are used to divide these regions into different classes:

- Finally, a bounding box regression (Bbox reg) is used to predict the bounding boxes for each identified region:

- And this, in a nutshell, is how an RCNN helps us to detect objects



Problems with R-CNN

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- 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.

Fast R-CNN

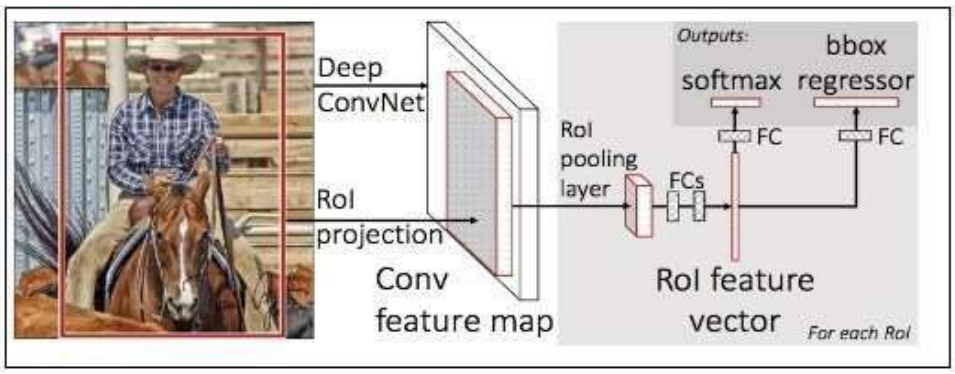


Figure 6.2 Fast R-CNN layer structure

The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer 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.

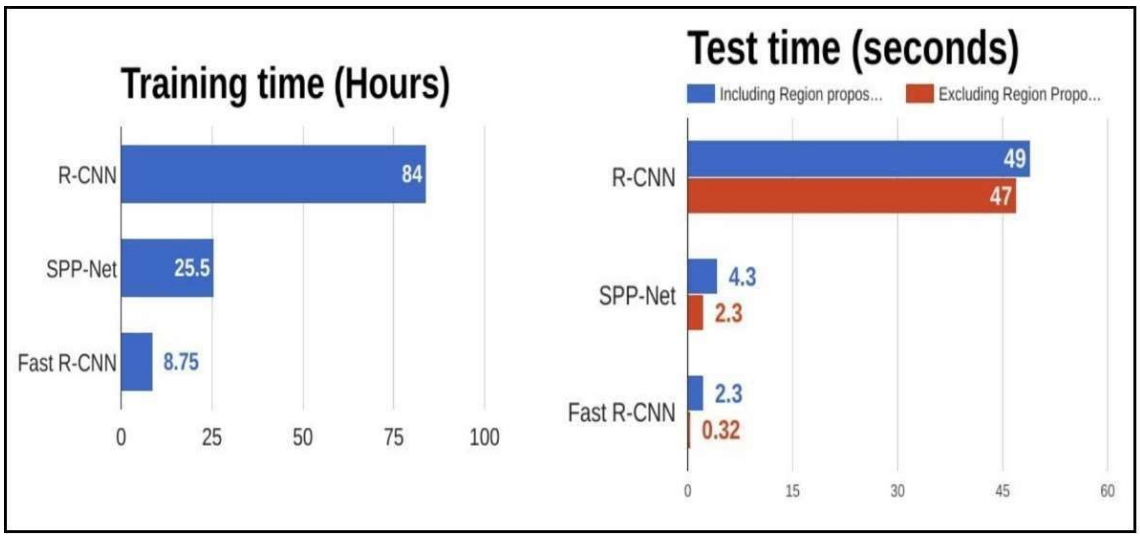


Figure 6.3 Comparison of object detection algorithms

From the above graphs, you can infer that Fast R-CNN is significantly faster in training and testing sessions over R-CNN. When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

Faster R-CNN

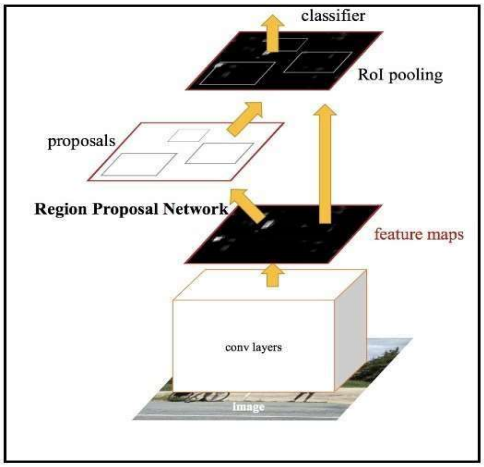


Figure 6.4 Faster R-CNN network structure

Both of the above algorithms(R-CNN & Fast R-CNN) uses selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, Shaoqing Ren et al. came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals.

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

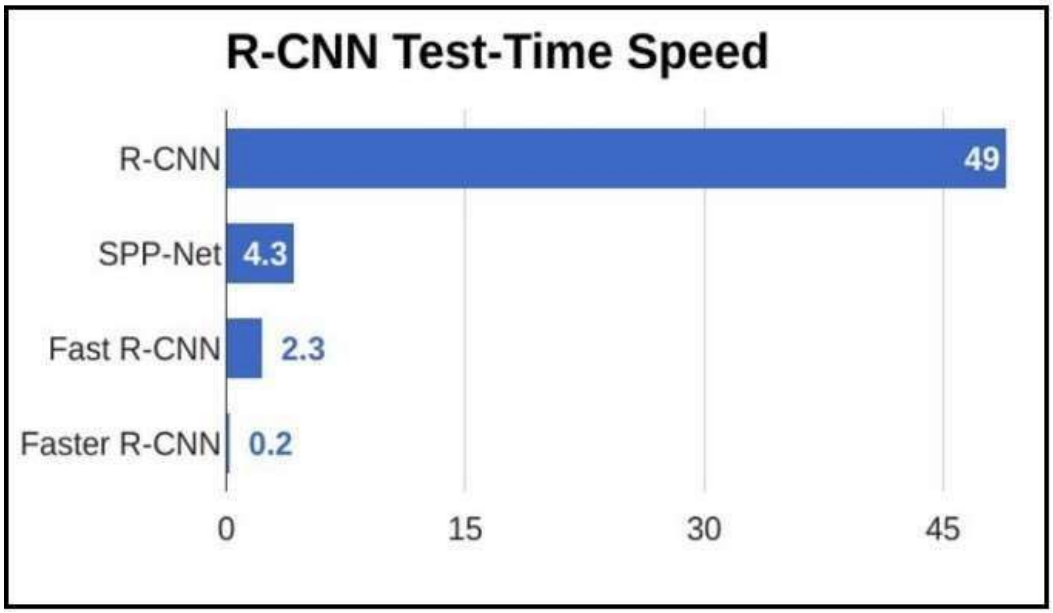


Figure 6.5 Comparison of test-time speed of object detection algorithms

From the above graph, you can see that Faster R-CNN is much faster than it's predecessors. Therefore, it can even be used for real-time object detection.

YOLO(You Only Look Once)

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region based algorithms seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

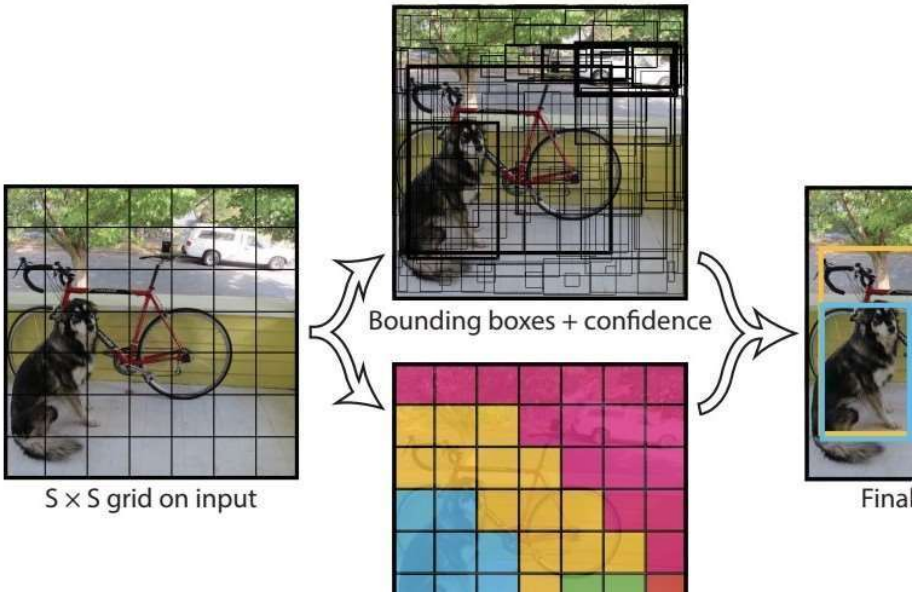


Figure 6.6 YOLO model detection

How YOLO works is that we take an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

YOLO is orders of magnitude faster(45 frames per second) than other object detection algorithms. 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.

CHAPTER -7

SYSTEM IMPLEMENTATION

7.1 Modules

The project is divided into two modules.

- **Object detection**
- **voice feedback**

7.1.1 Object detection

This is a module where the crucial information is collected from data set . Camera helps to detect the objects and with the help of raspberry pi3 object is detected and spoken SD card is also used in it

Required Items

Figure 7.1 Mobile camera

Figure 7.2 SD card

Figure 7.3 Head phones

7.2 Voice feedback

This is a module is the interesting module which the object detected is displayed in the screen and then converts it into the voice to help the visullay impaired person to understand what it is.

Figure 7.4 shows the voice alert

CHAPTER - 8

SYSTEM CONFIGURATION

8.1 Software Requirements:

- Languages used : Python
- Library used :Open CV, Tensorflow
- API used :Google Text to Speech
- Server :IP webcam server
- Remote Desktop: Microsoft RD client App
- For object prediction and classification we have taken YOLOv3 trained weights

8.2 Hardware Requirements:

- Operating system :Raspberry pi3 b+
- Ram :1GB
- Camera : Mobile Camera
- Hard disk :32GB SD card
- Voice reply :Head phones

8.3 Raspberry pi3 B+ Model:

The Raspberry Pi Compute Module (CM1), Compute Module 3 (CM3) and Compute Module 3 Lite (CM3L) are DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (for CM1 and CM3) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these module have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards opening up more options for the designer. The CM1 contains a BCM2835 processor (as used on the original Raspberry Pi and Raspberry Pi B+ models), 512MByte LPDDR2 RAM and 4Gbytes eMMC Flash. The CM3 contains a BCM2837 processor (as used on the Raspberry Pi 3), 1Gbyte LPDDR2 RAM and 4Gbytes eMMC Flash. Finally the CM3L product is the same as CM3 except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device. Note that the BCM2837 processor is an evolution of the BCM2835 processor. The only real differences are that the BCM2837 can address more RAM (up to 1Gbyte) and the ARM CPU complex has been upgraded from a single core ARM11 in BCM2835 to a Quad core Cortex A53 with dedicated 512Kbyte L2 cache in BCM2837. All IO

interfaces and peripherals stay the same and hence the two chips are largely software and hardware compatible. The pinout of CM1 and CM3 are identical. Apart from the CPU upgrade and increase in RAM the other significant hardware differences to be aware of are that CM3 has grown from 30mm to 31mm in height, the VBAT supply can now draw significantly more power under heavy CPU load, and the HDMI HPD N 1V8 (GPIO46 1V8 on CM1) and EMMC EN N 1V8 (GPIO47 1V8 on CM1) are now driven from an IO expander rather than the processor. If a designer of a CM1 product has a suitably specified VBAT, can accomodate the extra 1mm module height increase and has followed the design rules with respect to GPIO46 1V8 and GPIO47 1V8 then a CM3 should work fine in a board designed for a CM1.

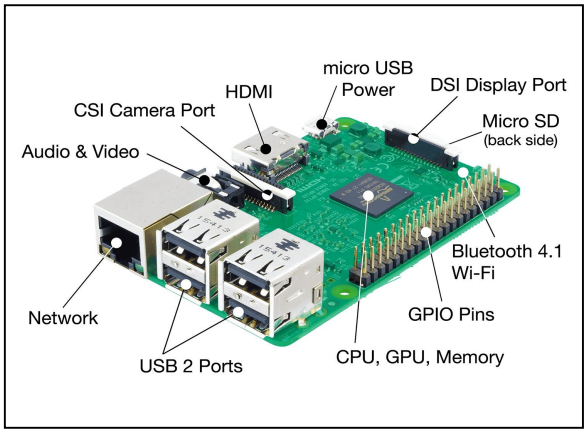
8.3.1 Hardware Features

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 35u hard gold plating

8.3.2 Software Features

- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set
- Mature and stable Linux softwarestack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

Figure 8.1 Raspberry pi3



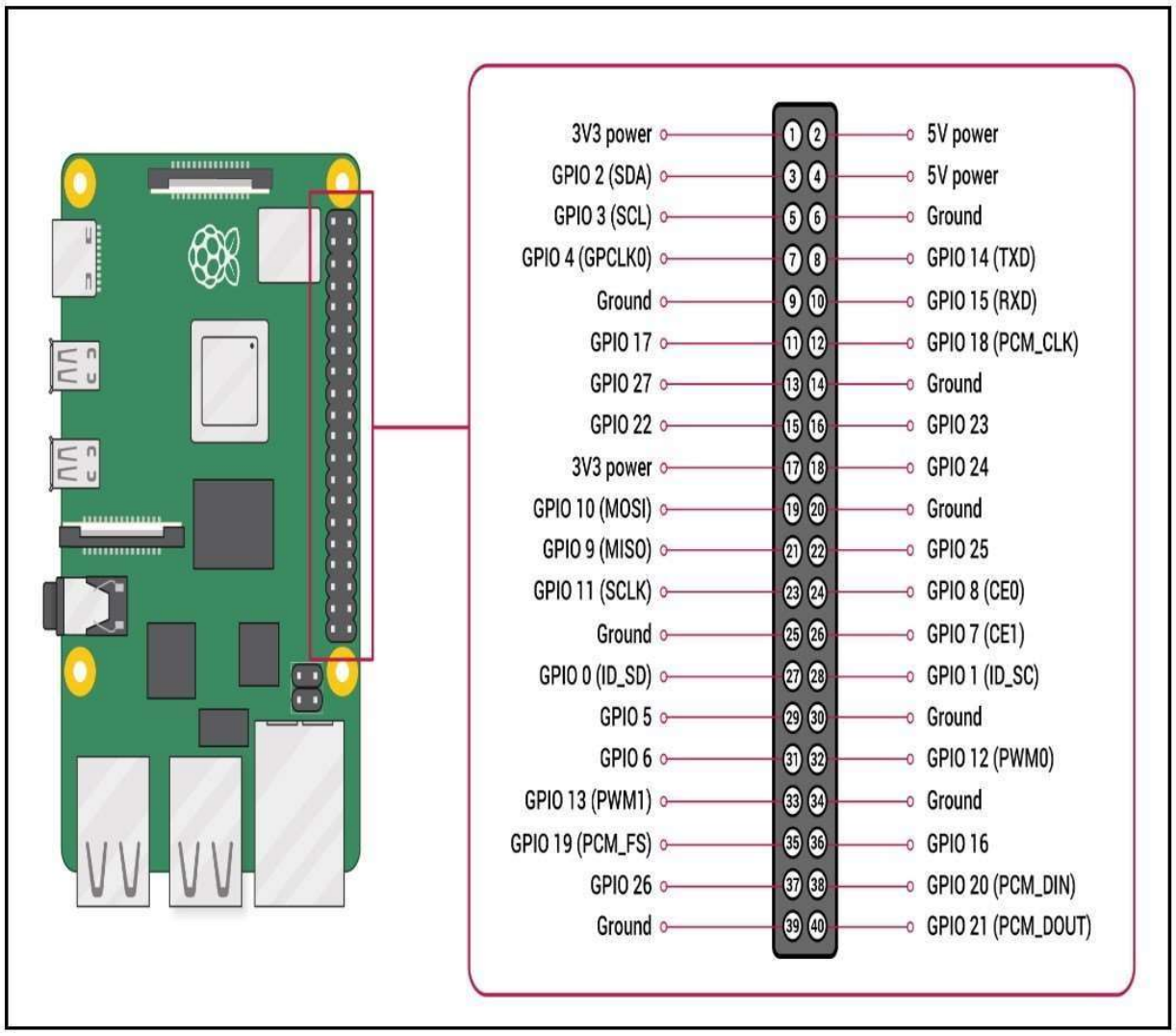


Figure 8.2 Pinout of Raspberry pi3

CHAPTER 9

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the testing. Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are varioustypes of test. Each test type addresses a specific testing requirement.

9.1. Testing Levels

9.1.1. Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.1.2. Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

9.1.3. Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items: Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised. Output : identified classes of application outputs must be exercised. Systems / Procedures : interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests isdetermined.

9.1.4. System Testing:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

9.2. Testing Methods

Testing is of two types. They are:

a) Static Testing

b) Dynamic Testing

9.2.1 Static Testing:

Static Testing is a technique for assessing structural characteristics of code. It examines the structure of code, but code is not executed, it does not involve actual execution. Static Testing is of three types. They are:

a) Inspection

b) Walkthroughs

c) Technical Reviews

9.2.1.1 Inspection:

An inspection is one of the most common sorts of review practices found in software project. The goal of the inspection is to identify defects. Commonly inspected work products include software requirements specifications and test plans.

9.2.1.2 Walkthrough:

Walkthrough in software testing is used to review documents with peers, managers, and fellow team members who are guided by the author of the document to gather feedback and reach a consensus. A walkthrough can be pre-planned or organized based on the needs.

9.2.1.3 Technical Reviews:

A software technical review is a form of peer review in which "a team of qualified personnel examines the suitability of the software product for its intended use and identifies discrepancies from specifications and standards. Technical reviews may also provide recommendations of alternatives and examination of various alternatives.

9.2.2 Dynamic Testing:

Dynamic testing is a term used in software engineering to describe the testing of the dynamic behavior of code. That is, dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run. It involves working with the software, giving input values and checking if the output is as expected by executing specific test cases which can be done manually or with the use of an automated process. Dynamic Testing is of two types. They are:

a) Black Box Testing

b) White Box Testing

9.2.2.1 Black Box Testing:

Black Box Testing is also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. Black Box Testing is of four types. They are:

a) Boundary Value Analysis

b) Equivalence Partitioning

c) Cause Effect Graph based Testing

d) Error Guessing Boundary Value Analysis

Boundary Value Analysis is applicable when the module to be tested is a function of several independent variables. This method becomes important for physical quantities where boundary checking condition is crucial. Equivalence Partitioning: Equivalence partitioning or equivalence class partitioning is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once. This technique tries to define test cases that uncover classes of errors, thereby reducing the total number of test cases that must be developed. Cause Effect Graph Based Testing: Cause effect graphing techniques helps in selecting combination of input conditions in a systematic way such that number of testcases does not become unmanageably large.The following process is used to derive the test cases.

i. **Division of Specification**:

The specification is divided into workable pieces as cause effect graphing becomes complex when used on large specification.

ii. Identification of Causes and Effects:

In this step, we will identify causes and effects in the specifications. A cause is a distinct input condition identified in the problem. Similar effect is the output condition.

iii. Transformation of specific effect into a cause effect graph:

Based on the analysis of the specification it is transformed into a Boolean graph linking the causes and effects. This is cause effect graph.

iv. Conversion into Decision Table:

The cause effect graph obtained is converted into limited entry decision table by verifying state conditions in the graph. Each column in the table indicates test graph. Error Guessing: It is the preferred method used when all the other methods fail. Sometimes, it is used to test some special cases. According to this method, errors or bugs can be guessed which do not fit in any of the earlier defined situation

a) Divide by zero

b) What if all inputs are negative?

c) What if input list is empty?

The tester does not have to use any particular testing technique.

9.2.2.2 White Box Testing:

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. White box testing is of four types. They are:

a) Logic Coverage Criteria

b) Basis Path Testing

c) Data Flow Testing

d) Mutation Testing

Logic Coverage Criteria:

Decision coverage is a stronger logic coverage criterion. Based on this criterion, a fair number of test cases should be written, for each decision to have true and false value at least one time. Put it differently, each branch changes direction must be executed at least once.

Basis Path Testing:

Basis path testing is the oldest structural testing technique. This testing is based on the control structure of the program. Based on the control structure a flow graph is prepared and all possible paths can be covered and tested during testing. Path coverage is useful for detecting more errors but the problem with this technique is that programs that contain loops may have an infinite number of possible paths and it is not practical to test all the paths. So, some selected paths are executed for the maximum coverage of logic.

Data Flow Testing:

Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects.

Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

Mutation Testing:

Mutation Testing is a type of Software testing where we mutate certain statements in the source code and check if the test cases are able to find the errors. he changes in mutant program are kept extremely small, so it does not affect the overall objective of the program. The goal of Mutation Testing is to assess the quality of the test cases which should be robust enough to fail mutant code. This method is also called as Fault-based testing strategy as it involves creating a fault in the program. In this project, we perform the manual testing which checks whether the given inputs get the appropriate outputs or not. This comes under the Black Box testing. We take an input from the user and an output is displayed. We check the given output with the result present in the dataset. We verify whether given output and the result in the dataset is same or not. Thus, testing is performed in the project.

CHAPTER-10

RESULTS

10.1 Object Detection

Fig 10.1 Shows bottle and bowl

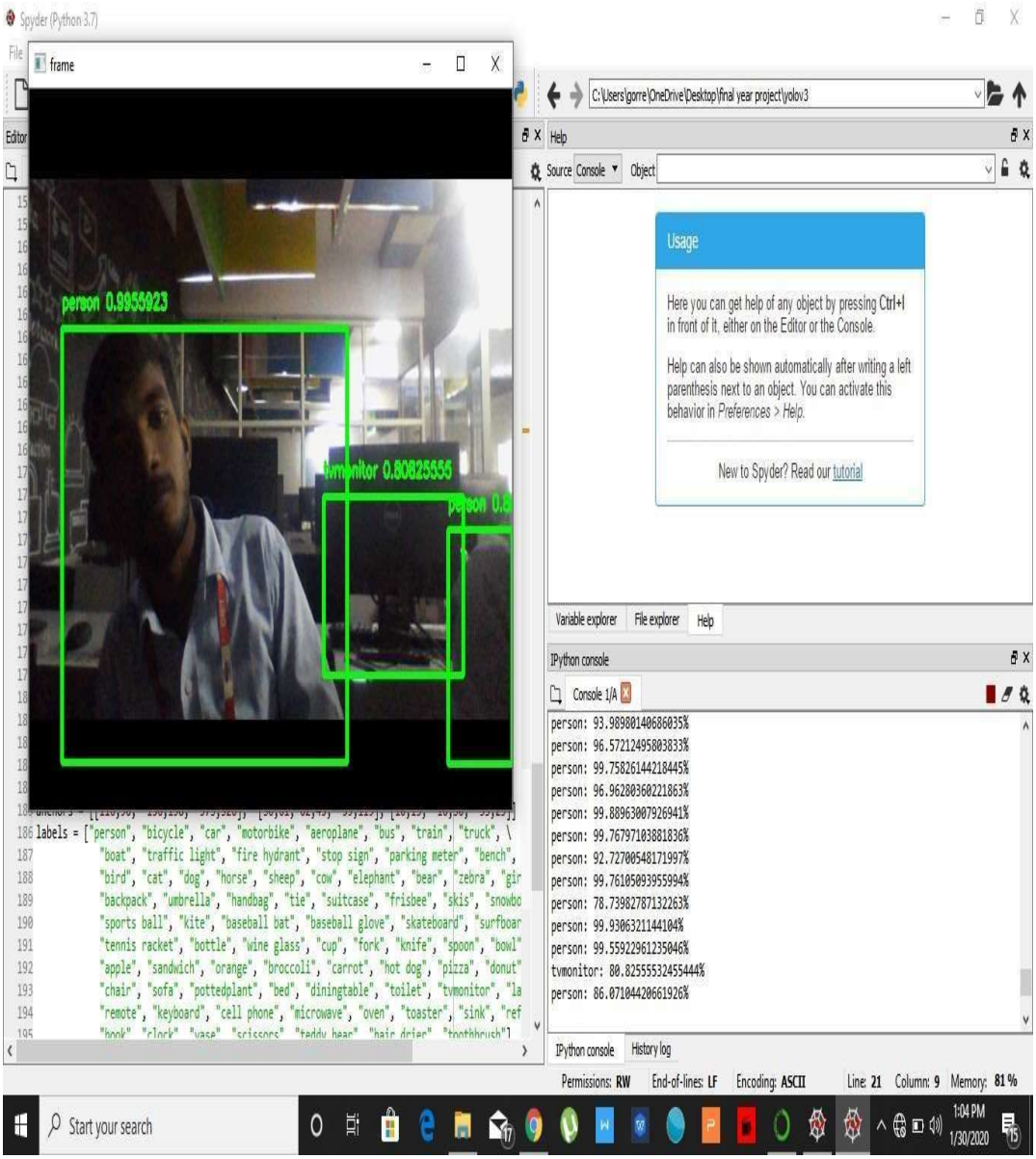


Figure 10.2 shows persons and TV monitor

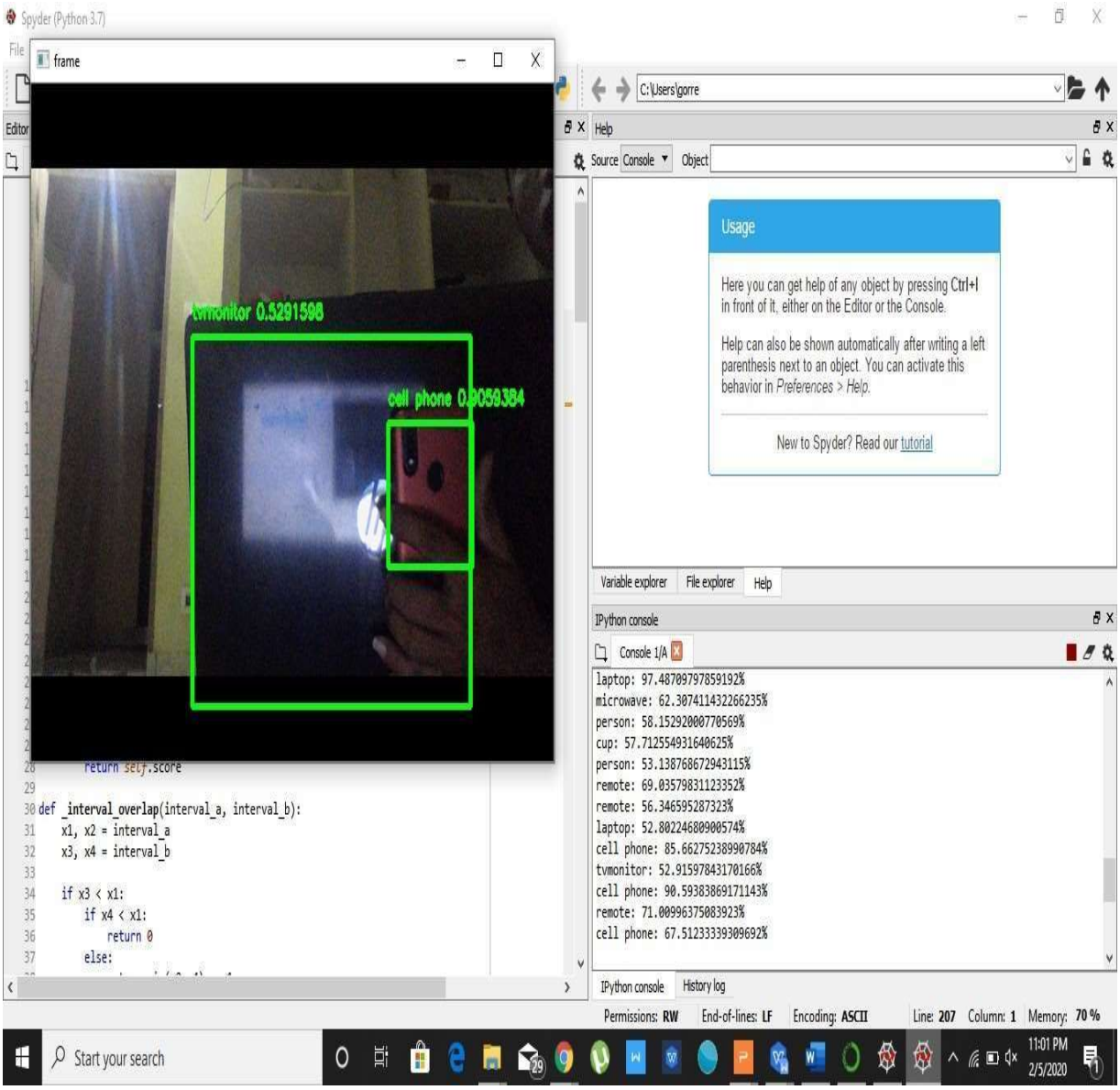


Figure 10.3 shows computer and mobile

CHAPTER 11
CONCLUSION AND FUTURE SCOPE

11.1 Conclusion

Due to its powerful learning ability and advantages in dealing with occlusion, scale transformation and background switches, deep learning based object detection has been a research hotspot in recent years. This paper provides a detailed review on deep learning based object detection frameworks which handle different sub-problems, such as occlusion, clutter and low resolution, with different degrees of modifications on R-CNN. The review starts on generic object detection pipelines which provide base architectures for other related tasks. Then, three other common tasks, namely salient object detection, face detection and pedestrian detection, are also briefly reviewed. Finally, we propose several promising future directions to gain a thorough understanding of the object detection landscape. This review is also meaningful for the developments in neural networks and related learning systems, which provides valuable insights and guidelines for future progress.

11.2 Future Scope

The future scope of the project determines to recognize multiple objects in a view with better accuracy and less detection time. The extension of this system identifies any kind of entity with faster frame rate. The text to speech module has also been developed according to the futuristic pace. Instead of using the pre-trained models, self-trained models can be used. The model can be trained to recognize objects which are frequently encountered by the user. Thus, it can be customized for the specific needs of the user and ensures safer navigation. The addition of a face recognition feature, the application can be trained to store the information about the people closely related to the person, which would help them to differentiate between peers and strangers.

APPENDIX

SOURCE CODE:

**********Object Detection**********

```
import cv2
from tensorflow.keras.models import load_model
import numpy as np

class BoundBox:
    def init (self, xmin, ymin, xmax, ymax, objness = None, classes = None):
        self.xmin = xmin
        self.ymin = ymin
        self.xmax = xmax
        self.ymax = ymax

        self.objness = objness
        self.classes = classes

        self.label = -1
        self.score = -1

    def get_label(self):
        if self.label == -1:
            self.label = np.argmax(self.classes)

        return self.label

    def get_score(self):
        if self.score == -1:
            self.score = self.classes[self.get_label()]
                    return self.score
```

```
def _interval_overlap(interval_a,interval_b):
    x1, x2 = interval_a
    x3, x4 = interval_b

    if x3 < x1:
        if x4 < x1:
            return 0
        else:
            return min(x2,x4) - x1
    else:
        if x2 < x3:
             return 0
        else:
            return min(x2,x4) - x3

def _sigmoid(x):
    return 1. / (1. + np.exp(-x))

def bbox_iou(box1, box2):
    intersect_w = _interval_overlap([box1.xmin, box1.xmax], [box2.xmin, box2.xmax])
    intersect_h = _interval_overlap([box1.ymin, box1.ymax], [box2.ymin, box2.ymax])

    intersect = intersect_w * intersect_h

    w1, h1 = box1.xmax-box1.xmin, box1.ymax-box1.ymin
    w2, h2 = box2.xmax-box2.xmin, box2.ymax-box2.ymin

    union = w1*h1 + w2*h2 - intersect

    return float(intersect) / union

def decode_netout(netout, anchors, obj_thresh, nms_thresh, net_h, net_w):
```

```
grid_h, grid_w = netout.shape[:2]
nb_box = 3
netout = netout.reshape((grid_h, grid_w, nb_box, -1))
nb_class = netout.shape[-1] - 5

boxes = []

netout[..., :2]  = _sigmoid(netout[..., :2])
netout[..., 4:]  = _sigmoid(netout[..., 4:])
netout[..., 5:] = netout[..., 4][..., np.newaxis] * netout[..., 5:]
netout[..., 5:] *= netout[..., 5:] > obj_thresh

for i in range(grid_h*grid_w):
    row = i / grid_w
    col = i % grid_w

    for b in range(nb_box):

        objectness = netout[int(row)][int(col)][b][4]
        objectness = netout[..., :4]

        if(objectness.all() <= obj_thresh): continue

        # first 4 elements are x, y, w, and h
        x, y, w, h = netout[int(row)][int(col)][b][:4]

        x = (col + x) / grid_w # center position, unit: image width
        y = (row + y) / grid_h # center position, unit: image height
        w = anchors[2 * b + 0] * np.exp(w) / net_w # unit: image width
        h = anchors[2 * b + 1] * np.exp(h) / net_h # unit: image height

                classes = netout[int(row)][col][b][5:]
```

```
            box = BoundBox(x-w/2, y-h/2, x+w/2, y+h/2, objectness, classes)
            box = BoundBox(x-w/2, y-h/2, x+w/2, y+h/2, None, classes)

            boxes.append(box)

    return boxes

def preprocess_input(image, net_h, net_w):
    new_h, new_w, _ = image.shape

    if (float(net_w)/new_w) < (float(net_h)/new_h):
        new_h = (new_h * net_w)/new_w
        new_w = net_w
    else:
        new_w = (new_w * net_h)/new_h
        new_h = net_h

      resized = cv2.resize(image[:,:,::-1]/255., (int(new_w), int(new_h)))

    new_image = np.ones((net_h, net_w, 3)) * 0.5
    new_image[int((net_h-new_h)//2):int((net_h+new_h)//2), int((net_w-
    new_w)//2):int((net_w+new_w)//2), :] = resized
    new_image = np.expand_dims(new_image, 0)

    return new_image

def correct_yolo_boxes(boxes, image_h, image_w, net_h, net_w):
    if (float(net_w)/image_w) < (float(net_h)/image_h):
        new_w = net_w
        new_h = (image_h*net_w)/image_w
    else:
        new_h = net_w
```

```
        new_w = (image_w*net_h)/image_h

    for i in range(len(boxes)):
        x_offset, x_scale = (net_w - new_w)/2./net_w, float(new_w)/net_w
        y_offset, y_scale = (net_h - new_h)/2./net_h, float(new_h)/net_h

        boxes[i].xmin = int((boxes[i].xmin - x_offset) / x_scale * image_w)
        boxes[i].xmax = int((boxes[i].xmax - x_offset) / x_scale * image_w)
        boxes[i].ymin = int((boxes[i].ymin - y_offset) / y_scale * image_h)
        boxes[i].ymax = int((boxes[i].ymax - y_offset) / y_scale * image_h)

def do_nms(boxes, nms_thresh):
    if len(boxes) > 0:
        nb_class = len(boxes[0].classes)
    else:
        return

    for c in range(nb_class):
        sorted_indices = np.argsort([-box.classes[c] for box in boxes])

        for i in range(len(sorted_indices)):
            index_i = sorted_indices[i]

            if boxes[index_i].classes[c] == 0: continue

            for j in range(i+1, len(sorted_indices)):
                index_j = sorted_indices[j]

                if bbox_iou(boxes[index_i], boxes[index_j]) >= nms_thresh:
                    boxes[index_j].classes[c] = 0

def draw_boxes(image, boxes, labels, obj_thresh):
```

```
    for box in boxes:
        label_str = ''
        label = -1

        for i in range(len(labels)):
            if box.classes[i] > obj_thresh:
                label_str += labels[i]
                label = i
                print(labels[i] + ': ' + str(box.classes[i]*100) + '%')

        if label >= 0:
            cv2.rectangle(image, (box.xmin,box.ymin), (box.xmax,box.ymax), (0,255,0), 3)
            cv2.putText(image,
                        label_str + ' ' + str(box.get_score()),
                        (box.xmin, box.ymin - 13),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        1e-3 * image.shape[0],
                        (0,255,0), 2)

    return image

net_h, net_w = 416, 416
obj_thresh, nms_thresh = 0.5, 0.45
anchors = [[116,90, 156,198, 373,326], [30,61, 62,45, 59,119], [10,13, 16,30, 33,23]]
labels = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck", \ "boat",
        "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", \ "bird", "cat", "dog",
        "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", \
         "backpack", "umbrella", "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", \
        "sports ball", "kite", "baseball bat", "baseball glove", "skateboard", "surfboard", \ "tennis
        racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl",
         "banana",\"apple", "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza",
         "donut", "cake", \"chair", "sofa", "pottedplant", "bed",
```

```
"diningtable","toilet","tvmonitor", "laptop", "mouse", \"remote", "keyboard", "cell phone",
"microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors",
"teddy bear", "hair drier", "toothbrush"]

        model = load_model('model.h5')

        cap = cv2.VideoCapture(0)

        main_val = 0

        while True:

            main_val += 1

            ret, image = cap.read()
            image_h, image_w, _ = image.shape
            new_image = preprocess_input(image, net_h, net_w)

            yolos = model.predict(new_image)
            boxes = []

            for i in range(len(yolos)):
                boxes += decode_netout(yolos[i][0], anchors[i], obj_thresh, nms_thresh, net_h, net_w)

            correct_yolo_boxes(boxes, image_h, image_w, net_h, net_w)

            do_nms(boxes, nms_thresh)

            draw_boxes(image, boxes, labels, obj_thresh)

            cv2.imshow('frame', image)
                if cv2.waitKey(10) & 0xFF==27:
```

```
            break
        cv2.destroyAllWindows()
        cap.release()
```

**********Trained_Data_for_Object_Detection**********

```
import struct
import numpy as np
from tensorflow.keras.layers import Conv2D, Input, BatchNormalization, LeakyReLU,
ZeroPadding2D
from tensorflow.keras.layers import UpSampling2D, add, concatenate
from tensorflow.keras.models import Model

def _conv_block(inp, convs, skip=True):
    x = inp
    count = 0

    for conv in convs:
        if count == (len(convs) - 2) and skip:
            skip_connection = x
        count += 1

        if conv['stride'] > 1: x = ZeroPadding2D(((1,0),(1,0)))(x)         x = Conv2D(conv['filter'],
                conv['kernel'],
                strides=conv['stride'],
                padding='valid' if conv['stride'] > 1 else 'same',
                name='conv_' + str(conv['layer_idx']),
                use_bias=False if conv['bnorm'] else True)(x)
        if conv['bnorm']: x = BatchNormalization(epsilon=0.001, name='bnorm_'str(conv['layer_idx']))(x)
        if conv['leaky']: x = LeakyReLU(alpha=0.1, name='leaky_' + str(conv['layer_idx']))(x)

    return add([skip_connection, x]) if skip else x
```

```
def make_yolov3_model():
    input_image = Input(shape=(None, None, 3))
    #Layer 0 => 4
    x = _conv_block(input_image, [{'filter': 32, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 0},
    {'filter': 64, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 1},{'filter': 32, 'kernel': 1, 'stride': 1,
    'bnorm': True, 'leaky': True, 'layer_idx': 2},{'filter': 64, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True,
    'layer_idx': 3}])
    #Layer 5 => 8
    x = _conv_block(x, [{'filter': 128, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 5},
        {'filter': 64, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 6},
        {'filter': 128, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 7}])
    # Layer 9 => 11
    x = _conv_block(x, [{'filter': 64, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 9},
        {'filter': 128, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 10}])
    # Layer 12 => 15
    x = _conv_block(x, [{'filter': 256, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 12},
    {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 13},
    {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 14}])
    # Layer 16 => 36
    for i in range(7):
        x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx':
            16+i*3},
        {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 17+i*3}])
    skip_36 = x
    # Layer 37 => 40
    x = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 37},
```

```
    {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 38},{'filter': 512, 'kernel':
    3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 39}])
# Layer 41 => 61
for i in range(7):
        x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx':
                41+i*3},
        {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 42+i*3}])
skip_61 = x
# Layer 62 => 65
x = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 62},
        {'filter': 512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 63},
        {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 64}])
# Layer 66 => 74
for i in range(3):
        x = _conv_block(x, [{'filter': 512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True,'layer_idx':
        66+i*3},
            {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 67+i*3}])
# Layer 75 => 79
x = _conv_block(x, [{'filter': 512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 75},
          {'filter': 1024, 'kernel': 3,  'stride': 1, 'bnorm': True, 'leaky':  True, 'layer_idx': 76},{'filter':  512,
           'kernel': 1,  'stride':  1,  'bnorm': True, 'leaky':  True, 'layer_idx':  77},{'filter': 1024, 'kernel': 3,
           'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 78},{'filter': 512, 'kernel': 1, 'stride': 1,
           'bnorm': True, 'leaky': True, 'layer_idx': 79}], skip=False)
# Layer 80 => 82
yolo_82 = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx':
80},
        {'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'leaky': False, 'layer_idx': 81}], skip=False)
```

```
        # Layer 83 => 86
        x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 84}],
        skip=False)
        x = UpSampling2D(2)(x)
        x = concatenate([x, skip_61])
        # Layer 87 => 91
        x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 87},
                    {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 88},{'filter': 256,
                    'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 89},{'filter': 512, 'kernel': 3,
                   'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 90},{'filter': 256, 'kernel': 1, 'stride': 1,
'bnorm':                    True, 'leaky': True, 'layer_idx': 91}], skip=False)
        # Layer 92 => 94
        yolo_94 = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx':
        92},
        {'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'leaky': False, 'layer_idx': 93}], skip=False)
        # Layer 95 => 98
        x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 96}],
        skip=False)
        x = UpSampling2D(2)(x)
        x = concatenate([x, skip_36])
        # Layer 99 => 106
        yolo_106 = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx':
        99},
        {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 100},{'filter': 128, 'kernel':
 1,      'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 101},{'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm':
         True,  'leaky': True,  'layer_idx': 102}, {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,  'leaky': True,
```

```
'layer_idx': 103},{'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 104},
{'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'leaky': False, 'layer_idx': 105}], skip=False)
model = Model(input_image, [yolo_82, yolo_94,yolo_106])
return model

class WeightReader:
def init (self, weight_file):
with open(weight_file, 'rb') as w_f:
major, = struct.unpack('i', w_f.read(4))
minor, = struct.unpack('i', w_f.read(4))
revision, = struct.unpack('i', w_f.read(4))
if (major*10 + minor) >= 2 and major < 1000 and minor < 1000:
w_f.read(8)
else:
w_f.read(4)
transpose = (major > 1000) or (minor > 1000)
binary = w_f.read()
self.offset = 0
self.all_weights = np.frombuffer(binary, dtype='float32')

def read_bytes(self, size):
self.offset = self.offset + size
return self.all_weights[self.offset-size:self.offset]

def load_weights(self, model):
for i in range(106):
try:
conv_layer = model.get_layer('conv_' + str(i))
print("loading weights of convolution #" + str(i))
if i not in [81, 93, 105]:
norm_layer = model.get_layer('bnorm_' + str(i))
size = np.prod(norm_layer.get_weights()[0].shape)
```

```
beta = self.read_bytes(size) # bias
gamma = self.read_bytes(size) # scale
mean = self.read_bytes(size) # mean
var = self.read_bytes(size) # variance
weights = norm_layer.set_weights([gamma, beta, mean, var])
if len(conv_layer.get_weights()) > 1:
bias =self.read_bytes(np.prod(conv_layer.get_weights()[1].shape))
Kernel=self.read_bytes(np.prod(conv_layer.get_weights()[0].shape))
kernel = kernel.reshape(list(reversed(conv_layer.get_weights()[0].shape)))
kernel = kernel.transpose([2,3,1,0])
conv_layer.set_weights([kernel, bias])
else:
kernel = self.read_bytes(np.prod(conv_layer.get_weights()[0].shape))
kernel = kernel.reshape(list(reversed(conv_layer.get_weights()[0].shape)))
kernel = kernel.transpose([2,3,1,0])
conv_layer.set_weights([kernel])
except ValueError:
print("no convolution #" + str(i))
def reset(self):
self.offset = 0
model = make_yolov3_model()

weight_reader = WeightReader('yolov3.weights')

weight_reader.load_weights(model)

model.save('model.h5')
```

REFERENCES

[1] Tomasz Malisiewicz and Alexei A Efros, “Improving spatial support for objects via multiple segmentations,” 2007.

[2] Thomas Blaschke, “Object based image analysis for remote sensing,” ISPRS journal of photogrammetry and remote sensing, vol. 65, no. 1, pp. 2–16, 2010.

[3] Sreenath Rao Vantaram and Eli Saber, “Survey of contemporary trends in color image segmentation,” Journal of Electronic Imaging, vol. 21, no. 4, pp. 040901–1, 2012.

[4] Liangliang Cao and Li Fei-Fei, “Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes,” in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. IEEE, 2007, pp. 1–8.

[5] Dorit S Hochbaum and Vikas Singh, “An efficient algorithm for co-segmentation,” in Computer Vision, 2009 IEEE 12th International Conference on. IEEE, 2009, pp. 269–276.

[6] Armand Joulin, Francis Bach, and Jean Ponce, “Discriminative clustering for image co-segmentation,” in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 1943–1950.

[7] Dhruva Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen, “icoseg: Interactive co-segmentation with intelligent scribble guidance,” in CVPR, 2010 IEEE Conference on. IEEE, 2010, pp. 3169–3176.

[8] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother, “Cosegmentation revisited: Models and optimization,” in Computer Vision–ECCV 2010, pp. 465– 479. Springer, 2010.

[9] Yuning Chai, Victor Lempitsky, and Andrew Zisserman, “Bicos: A bi-level co-segmentation method for image classification,” 2011.

[10] Yuning Chai, Esa Rahtu, Victor Lempitsky, Luc Van Gool, and Andrew Zisserman, “Tricos: A tri-level class-discriminative co-segmentation method for image classification,” in Computer Vision–ECCV 2012, pp. 794–807. Springer, 2012.