Multi Object Detection Using Machine Learning for Visually Impaired People

A project report submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE and ENGINEERING

By

K.V.K VARMA	**16L31A05E4**
B. ASHISH	**16L31A05E7**
S.V.S RAMESH	**16L31A05H5**
C.H.B.V PRAVEEN	**16L31A05C7**

Under the guidance of
Mr. B. Vamsi
Assistant Professor

DEPARTMENT OF **COMPUTER SCIENCE AND ENGINEERING**

VIGNAN'S INSTITUTE OF INFORMATION TECHNOLOGY

(Autonomous) Affiliated to JNTU Kakinada & Approved by AICTE, New Delhi

Re-Accredited by NBA & NAAC (CGPA of 3.41/ 4.00)

ISO 9001:2008, ISO 14001:2004, OHSAS 18001:2007 Certified Institution

VISAKHAPATNAM – 530 039

April 2019

VIGNAN'S INSTITUTE OF INFORMATION TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled as "**Multi Object Detection Using Machine Learning for Visually Impaired People**" is the bonafide record of project work carried out under my supervision by

1. **K.V.K Varma, 16L31A05E4**
2. **B. Ashish, 16L31A05E7**
3. **S.V.S Ramesh, 16L31A05H5**
4. **C.H.B.V Praveen, 16L31A05C7**

during the academic year 2016-2020, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University, Kakinada. The results embodied in this project report have not been **s**ubmitted to any other University or Institute for the award of any Degree or Diploma.

Head of the Department
Dr.Challa Narsimham
Professor

Signature of the Project Guide
B. Vamsi
Assistant Professor

Signature of JNTUK External

DECLARATION

We hereby declare that the project report entitled "**Multi Object Detection Using Machine Learning for Visually Impaired People**" has been written by us and has not been submitted either in part or whole for the award of any degree, diploma or any other similar title to this or any other university .

1. K.V.K Varma	16L31A05E4
2. B. Ashish	16L31A05E7
3. S.V.S Ramesh	16L31A05H5
4. C.H.B.V Praveen	16L31A05C7

Date:

Place:

ACKNOWLEDGEMENT

We express our gratitude to our beloved professor **Dr. B. Arundhathi**, for providing necessary departmental facilities in spite of her busy schedule and guiding us in every possible way.

We are indeed very grateful to **Dr. Challa Narsimham**, Professor and Head of the Department, Computer Science and Engineering Department, Vignan's Institute of Information Technology, Visakhapatnam, for his ever willingness to share his valuable knowledge and constantly inspire us through suggestions.

We express our deep gratitude to our project coordinator**. Mrs. K. Neelima**, Associate Professor, CSE Department, Vignan's Institute of Information Technology, Visakhapatnam, for rendering us guidance and valuable advices always. She has been a perennial source of inspiration and motivation of this project.

We express our deep gratitude to our project guide **Mr. B. Vamsi**, Assistant Professor, CSE Department, Vignan's Institute of Information technology, Visakhapatnam, for rendering us guidance and valuable advice always. He has been a perennial source of inspiration and motivation of this project.

We sincerely thank all the Staff Members of the Department for giving us their support in all the stages of the project work and completion of this project. In all humility and reverence, we express our profound sense of gratitude to all the elders and Professors who have willingly spared time, experience and knowledge to guide us in our project.

1. K.V.K Varma	16L31A05E4
2. B. Ashish	16L31A05E7
3. S.V.S Ramesh	16L31A05H5
4. C.H.B.V Praveen	16L31A05C7

ABSTRACT

This project presents a new technique for assisting visually-impaired people in recognizing objects in environment. The existing methodologies which aims to solve the problem of object recognition in a traditional way with a single-label strategy. By using machine learning and vision technologies, our work is to provide an efficient solutions for assisting visually-impaired people with a multi-label strategy. In the proposed work we are solving the existing system problem by using classification / clustering techniques which are used to reduce the recognize time of multi objects in less time with best possible complexity. The model used to assist the visually impaired people can independently recognize objects which are near to them.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

CHAPTER 1

INTRODUCTION

1.1 About the Project

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc. Each of these "eternal" problems is solved by trial and error by the efforts of numerous groups of the engineers and scientists.

As modern technical solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems.

Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is refers to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term “object recognition”, they often mean “object detection”. It may be challenging for beginners to distinguish between different related computer vision tasks. So, we can distinguish between these three computer vision tasks with

this example:

Image Classification: This is done by Predict the type or class of an object in an image.

Input: An image which consists of a single object, such as a photograph.

Output: A class label (e.g. one or more integers that are mapped to class labels).

Object Localization: This is done through, Locate the presence of objects in an image and

indicate their location with a bounding box.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height).

Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

One of the further extension to this breakdown of computer vision tasks is object segmentation, also called "object instance segmentation" or "semantic segmentation," where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box. From this breakdown, we can understand that object recognition refers to a suite of challenging computer vision tasks.

For example, image classification is simply straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition.

Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy. We need to understand terms such as object detection, object localization, loss function for object detection and localization, and finally explore an object detection algorithm known as "You only look once" (YOLO).

Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition.

1.2 Motivation

In everyday life, we have come across many people who are visually impaired. They are not able to see the things in front of them. So, we decided to provide them a solution by building a

system which can help them by detecting the multiple objects before them and tells them the particular object which is in front of them.

1.3Purpose

The main purpose of this project is to help the visually impaired people to recognize the objects, which present in front of them both in indoor and outdoor. So, to do this we are implementing this project, which can detect the objects from an image or a video which is captured and given to the application. Since these people are physically challenging, we are providing this application which could possibly detect the different classes of objects and produce voice message as an output. So, that they can hear the name of that particular object as a voice output.

1.4 Organization of Documentation

This documentation is divided into chapter wise. There are nine chapters in this document:

Chapter 1 : Contains introduction and motivation about the project and it is also contains the main objectives of the project.

Chapter 2: Literature survey of the project. It contains proposed system and applications. Further will be discussed.

Chapter 3: Contains Feasibility Study about the project.

Chapter 4: Contains design and modules of the project. Further will be discussed.

Chapter 5: Contains implementation of the project and form of the project.

Chapter 6: Contains system testing which will be discussed further.

Chapter 7: Contains Results.

Chapter 8: Contains conclusions.

Chapter 9: Contains references made about the project.

CHAPTER 2

CHAPTER 2

LITERATURE SURVEY

2.1 Reference Papers

Yunchao Wei, Wei Xia, Min Lin has proposed their work on "A flexible CNN Framework for Multi-Label Image Classification"

Their work states that, convolution Neural Network (CNN) has demonstrated promising performance in single-label image classification tasks, however, how CNN best cops with multi-lable images still remains an open problem, mainly due to the complex underlying object layouts and insufficient multi-label training images. [1]

Rim Trabelsi, Issam Jabri and Farid Melganihas proposed their work on "Indoor object Recognition in RGBD images with Complex-Valued Neural Networks for Visually-Impaired People" in the year 2018

In this work they proposed a new multi-model technique for assisting visually-impaired people in recognizing objects in public indoor environment. Unlike common methods which aims to solve the problem of multi-class object recognition in a traditional single-label strategy, a comprehensive approach is developed here allowing samples to take more than one label at a time using a new complex-valued representation. [2]

Liang Zhang and Michael Towsey has proposed their work in "Using multi-label classification for acoustic pattern and assisting bird species surveys, Applied Acoustics" in the year 2016. This paper proposes a binary relevance based multi-label classification approach to recognize simultaneous acoustic patterns in one-minute audio clips. [4]

X. Yang and Y. Tian has proposed their work in "Robust door detection in unfamiliar environments by combining edge corner features" in 2010. In this paper they proposed a new hybrid network based on YOLO and ResNet (Yolo-resnet) for multi-object detection. Our method integrates ResNet into the feature extraction of the YOLOv3 framework and the detection results demonstrated that our hybrid network is efficient for detecting multi-objects from an image of complex natural scenes. [14]

Zhong-Qui Zhao and Peng Zheng has proposed their work on "Object Detection with Deep Learning" in the year 2018.Their work states that

Traditional object detection methods are built on handcrafted features and shallow

trainable architectures. Their performance easily stagnates by constructing complex ensembles which combine multiple low-level image features with high-level context from object detectors and scene classifiers. With the rapid development in deep learning, more powerful tools, which are able to learn semantic, high-level, deeper features, are introduced to address the problems existing in traditional architectures. These models behave differently in network architecture, training strategy and optimization function, etc. In this paper, we provide a review on deep learning based object detection frameworks. Our review begins with a brief introduction on the history of deep learning and its representative tool, namely Convolutional Neural Network (CNN). [15]

Agarwal S, Awan A and Roth D has proposed their work on "Learning to detect objects in images via a sparse part-based representation" in the year 2004.Their work states that the problem of detecting objects in still, gray-scale images. Our primary focus is the development of a learning based approach to the problem that makes use of a sparse, part-based representation. [11]

2.2 Existing System

Complex-Valued Neural Networks: ML-CVNN

CVNNs are250 an extension of the classic RVNNs which are able to deal with information belonging to the complex coordinate space with complex-valued parameters and variables. State-of-the-art proposals have shown efficiency with single-based classification techniques, yet it remains a niche area with applica-255 tion to multi-label classification. In this section, we are going to present the new multi-label complex-valued neural networks approach, ML-CVNN, to be the first-ever multi-label classification technique that used complex-valued configuration.

2.3 Proposed Method

RetinaNet

To train the network model in a more effective manner, we herein adopt the same strategy as that used for DSSD (the performance of the residual network is better than that of the VGG network). The goal is to improve accuracy. However, the first implemented for the modification was the replacement of the VGG network which is used in the original SSD with ResNet. We will also add a series of convolution feature layers at the end of the underlying network. These feature layers will gradually be reduced in size that allowed

prediction of the detection results on multiple scales. When the input size is given as 300 and 320, although the ResNet–101 layer is deeper than the VGG–16 layer, it is experimentally known that it replaces the SSD's underlying convolution network with a residual network, and it does not improve its accuracy but rather decreases it.

2.4 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). How a system implements functional requirements is detailed in the system design. In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. Each use case illustrates behavioural scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

2.4.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The user interface to guide the operating person in providing input.

2.4.2 Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct Direction to the management for getting correct information from the

computerized system. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

2.4.3 Output Design

An output is one, which meets the requirements of the blind people and presents the information clearly through voice. In this system results of processing are communicated to the visually impaired people. In output design it is determined how the information is to be displaced for immediate need. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help people object detection.

Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

1. Select methods for presenting information.
2. Create document, report, or other formats that contain information produced by the system.

2.5 Non Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. In general, functional requirements define what a system is supposed to do whereas non-functional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements," and "non-behavioural requirements". Qualities, that is, non-functional requirements, can be divided into two main categories:

1. Execution qualities, such as security and usability, which are observable at run time.
2. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

2.6 Software Environment

2.6.1 Introduction to python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

What can Python do?

Python can be used on a server to create web applications.
Python can be used alongside software to create workflows.
Python can connect to database systems. It can also read and modify files.
Python can be used to handle big data and perform complex mathematics.
Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
Python has a simple syntax similar to the English language.
Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
]In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

Python was designed for readability, and has some similarities to the English language with influence from mathematics.

Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on Linux open the command line or on Mac open the Terminal and type:

```
python --version
```

2.6.2 Introduction to tkinter

Python GUI – tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

There are two main methods used which the user needs to remember while creating the Python application with GUI.

- **Tk(screenName=None, baseName=None, className='Tk', useTk=1):** To create a main window, tkinter offers a method

'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

```
m=tkinter.Tk() where m is the name of the main window object
```

- **mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

```
m.mainloop()
```

tkinter also offers access to the geometric configuration of the widgets which can organizethe widgets in the parent windows. There are mainly three geometry manager classes class.

pack() method **:** It organizes the widgets in blocks before placing in the parent widget.

grid() method **:** It organizes the widgets in grid (table-like structure) before placing in the parent widget.

place() method **:** It organizes the widgets by placing them on specific positions directed by the programmer.

2.6.3 Widgets in tkinter

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter.

Button: To add a button in your application, this widget is used

Canvas: It is used to draw pictures and other complex layout like graphics, text and widgets.

CheckButton: To select any number of options by displaying a number of options to a user as toggle buttons.

Entry: It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.

Frame: It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.

Label: It refers to the display box where you can put any text or image which can be updated any time as per the code.

Listbox: It offers a list to the user from which the user can accept any number of options.

MenuButton: It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality.

Menu: It is used to create all kinds of menus used by the application.

Message: It refers to the multi-line and non-editable text. It works same as that of Label.

RadioButton: It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.

Scale: It is used to provide a graphical slider that allows to select any value from that scale. **Scrollbar**: It refers to the slide controller which will be used to implement listed widgets. **Text:** To edit a multi-line text and format the way it has to be displayed.

TopLevel: This widget is directly controlled by the window manager. It don't need any parent window to work on.

2.6.4 Advantages of tkinter

Brevity: Python programs using Tkinter can be very brief, partly because of the power of Python, but also due to Tk. In particular, reasonable default values are defined for many options used in creating a widget, and packing it (i.e., placing and displaying).

Cross platform: Tk provides widgets on Windows, Macs, and most UNIX implementations with very little platform-specific dependence. Some newer GUI frameworks are achieving a degree of platform independence, but it will be some time before they match Tk's in this respect.

Maturity: First released in 1990, the core is well developed and stable.

Extensibility: Many extensions of Tk exist, and more are being frequently distributed on the Web. Any extension is immediately accessible from Tkinter, if not through an extension to Tkinter, than at least through Tkinter's access to the Tcl language.

2.6.5 Advantages of python

Python is a high level, interpreted and general purpose dynamic programming language that focuses on code readability. It has fewer steps when compared to Java and C. It was founded in 1991 by developer Guido Van Rossum. It is used in many organizations as it

supports multiple programming paradigms. It also performs automatic memory management.

Advantages:

1) Presence of third-party modules
2) Extensive support libraries(NumPy for numerical calculations, Pandas for data analytics etc)
3) Open source and community development
4) Easy to learn
5) User-friendly data structures
6) High-level language
7) Dynamically typed language(No need to mention data type based on value assigned, it takes data type)
8) Object-oriented language
9) Portable and Interactive
10) Portable across Operating systems

2.6.6 What tkinter Can't Do

Sometimes hard to debug in that Tkinter Widget at their core aren't python objects; Tkinter provides a wrapper around the actual tk widget which sometimes mean you get weird error messages. There is very little printing support (you can generate postscript docs from the canvas, but there's no built-in support in the text widget for printing).

Some people claim Tk is ugly. Tkinter isn't "pythonic" as wxPython. Non-native look-and-feel.

Simplistic model is easy to learn, but becomes cumbersome with complex interfaces.

To be truly usable, requires downloading extra toolkits

Probably a dead-end technology, as toolkits such as wxPython gain mindshare.

2.7 Hardware and Software Specification

- The following are the tables representing the hardware requirements and software requirements involved in this project:

- **Software Specifications**

Component	Configuration
Operating System	Windows XP or Later Version
Database	System storage
Language	Python

Table 1 Software Interfaces

- **Hardware Specifications**

Component	Recommended Configuration
Processor	Pentium-IV 3.5GHz
RAM	4 GB(min)
Hard Disk	40GB
Input Device	Standard Windows Keyboard, Mouse, mobile phone with camera
Output Device	Monitor, speakers

Table 2 Hardware Interfaces

CHAPTER 3

CHAPTER 3

FEASIBILITY STUDY

3.1 Background

The aim of Object Detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In some other situations, the pose information is more detailed and contains the parameters of a linear or non- linear transformation. For example for face detection in a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. In the case of a fixed rigid object in an image, only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability

Getting to use modern object detection methods in applications and systems, as well as building new applications based on these methods is not a straight forwardtask. Early implementations of object detection involved the use of classical algorithms, like the ones supported in OpenCV, the popular computer vision library. However, these classical algorithms could not achieve enough performance to work under different conditions.

The breakthrough and rapid adoption of deep learning in 2012 brought into existence modern and highly accurate object detection algorithms and methods such as R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet and fast yet highly accurate ones like SSD and YOLO. Using these methods and algorithms, based on deep learning which is also based on machine learning require lots of mathematical and deep learning frameworks understanding

3.2 Pre-Requirements

To perform object detection using ImageAI, you need to install all the required packages.

- Download Python 3

- Download Object Detection model file
- Install ImageAI and dependencies.
- Install Tensorflow
- Install OpenCV
- Install Keras

Download python

Download and Install Python 3 version from official Python Language website

https://www.python.org/downloads/

Python can be easy to pick up whether you're a first time programmer or you're experienced with other languages. Python is the language that is stable, flexible, and provides various tools to developers. All these properties of Python make it the first choice for Machine learning. From development to implementation and maintenance, Python is helping developers to be productive and confident about the software they are developing. There are many benefits of using Python in Machine learning. These benefits are increasing the popularity of Python.

Install ImageAI

ImageAI can be installed from the official website ***http://www.imageai.org/***

We can also get ImageAI using the PIP(Python Package Installer) as **pip3 install imageai**

ImageAI is an easy to use Computer Vision Python library that empowers developers to easily integrate state-of-the-art Artificial Intelligence features into their new and existing applications and systems

Dependencies with ImageAI should also be installed namely H5py, Numpy, Matplotlib, SciPy, Pillow. All these libraries can be installed using PIP command in the terminal.

Install Tensorflow

Tensorflow 1.15 can also be installed using Python PIP with **pip3 install tensorflow==1.15.0**

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Install OpenCV

OpenCV can be installed using python PIP with the command **pip3 install opencv-python**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Install Keras

Keras can be installed using Python PIP with the command **pip3 install keras**

tf.keras is TensorFlow's high-level API for building and training deep learning models.

3.3 Algorithm RetinaNet

RetinaNet for object detection is a single, unified network composed of a backbone network and two task-specific subnetworks. In RetinaNet, an one-stage detector, by using focal loss, lower loss is contributed by “easy” negative samples so that the loss is focusing on “hard” samples, which improves the prediction accuracy. With ResNet+FPN (Feature Pyramid Network) as backbone for feature extraction, plus two task-specific subnetworks for classification and bounding box regression, forming the RetinaNet.
RetinaNet has been formed by making two improvements over existing single stage object

detection models (like YOLO and SSD)

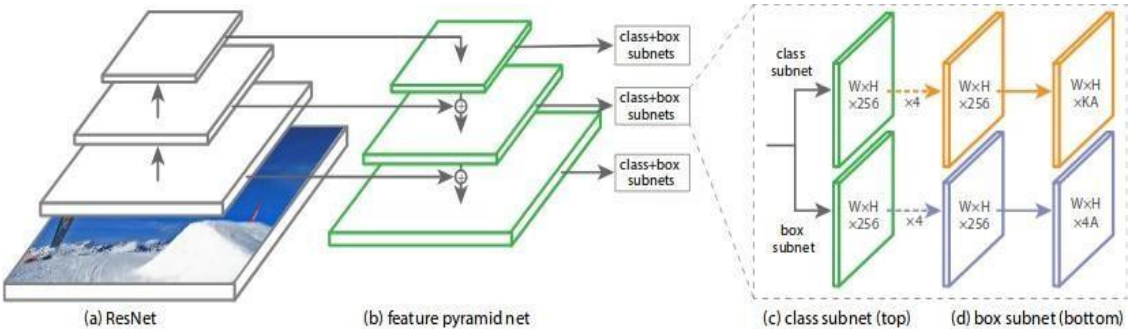
3.3.1 Feature Pyramid Networks (FPN) for Object Detection

3.3.2 Focal Loss for Dense Object Detection

Feature Pyramid Network (FPN) is used on top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution input image. FPN is multiscale, semantically strong at all scales, and fast to compute.Pyramid networks have been used

conventionally to identify objects at different scales. A Feature Pyramid Network (FPN) makes use of the inherent multi- scale pyramidal hierarchy of deep CNNs to create feature pyramids.

Fig1 : A Feature Pyramid Network (FPN)



The one-stage RetinaNet network architecture uses a Feature Pyramid Network (FPN) backbone on top of a feedforward ResNet architecture (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work tofocus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN while running at faster speeds.

Focal Loss is an improvement on cross-entropy loss that helps to reduce the relative loss for well-classified examples and putting more focus on hard, misclassified examples.

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

The focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

CHAPTER 4

CHAPTER 4

SYSTEM DESIGN

4.1 Input Design

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screen, an user interface is provided. In which we can insert an image and can run the program. To insert an image there will be a browse files will be appeared from that we can select an image for object detection. In order to avoid format errors, our project will be run on different formats of an image such as jpeg, png etc., In addition to this, videos can also be processed of formats mp4, mkv. The UI contains buttons which are helpful for user to navigate to next phase.

For blind people, a dedicated mobile phone with camera on which this application runs. He/she just need to capture the photo. This way we can ease the use of application in terms of inputs. The application automatically recognize the object in the given image or a video.

4.2 Output Design

The output is displayed in the user interface part with the count of distinct object along with object names and their respective probabilities. And also a voice message will be played through the default music player which will be very helpful and usable method for visually impaired people.

The output is in the form of both image and audio form. The output image have the objects detected in it with the bounding boxes around the detected object so the user is able to see and there will be an audio played from player.

4.3 UML Design

UML stands for Unified Modeling Language. UML is a standardized general purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. UML is comprised of two major components, are Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non- software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

4.3.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

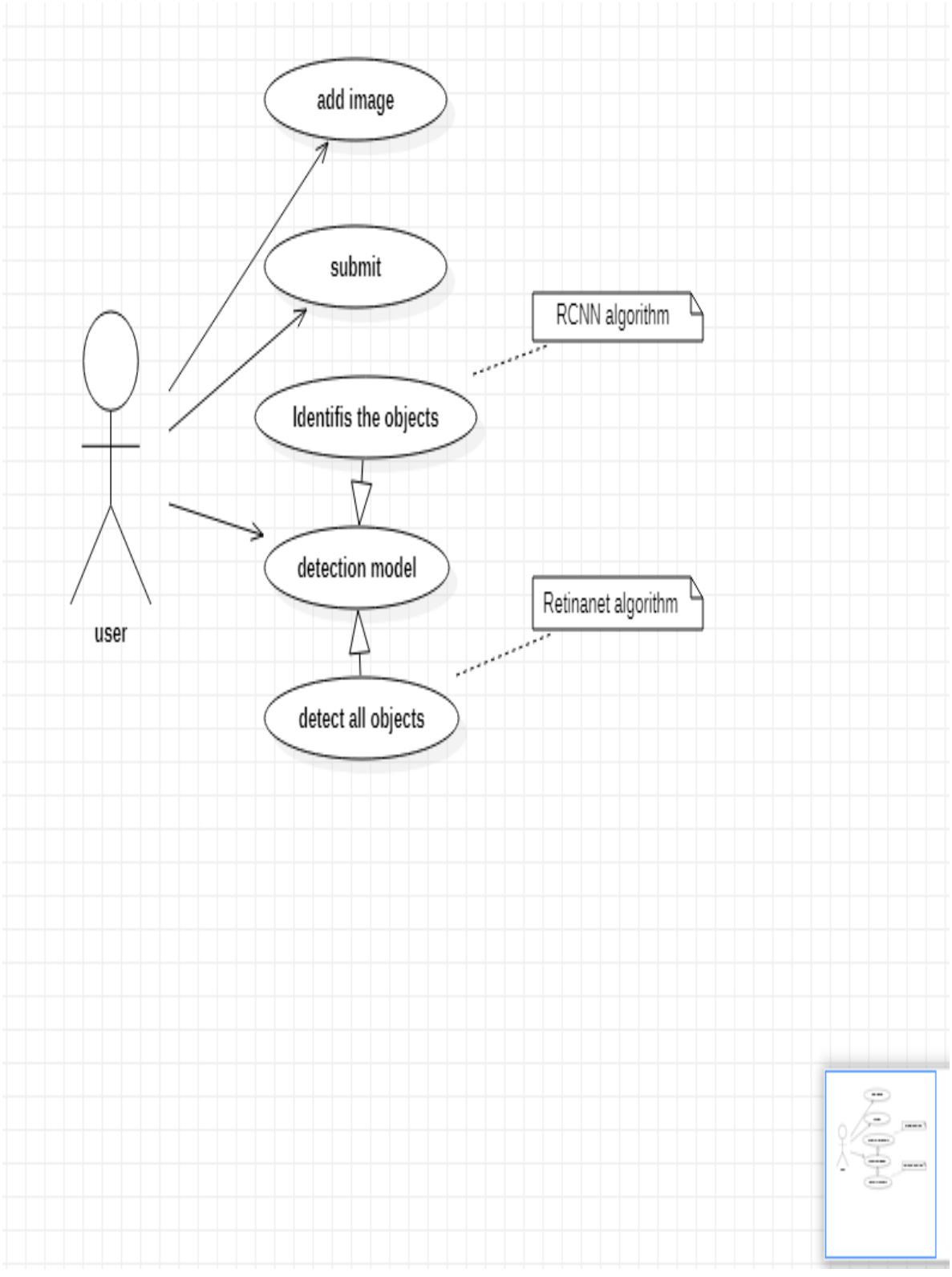


Fig 1 use case diagram

4.3.2 Class Diagram

A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

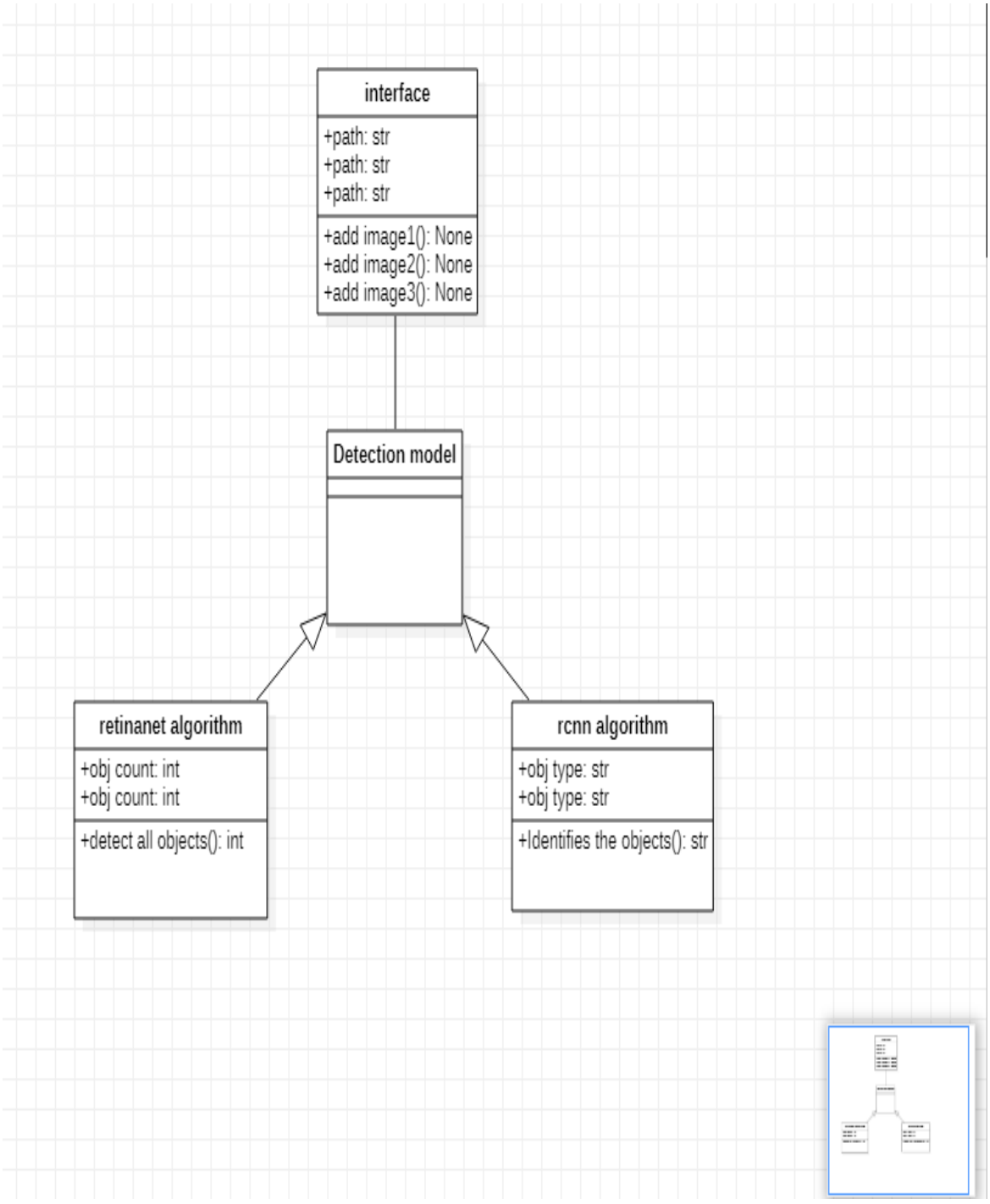


Fig 2 class diagram

4.3.3 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

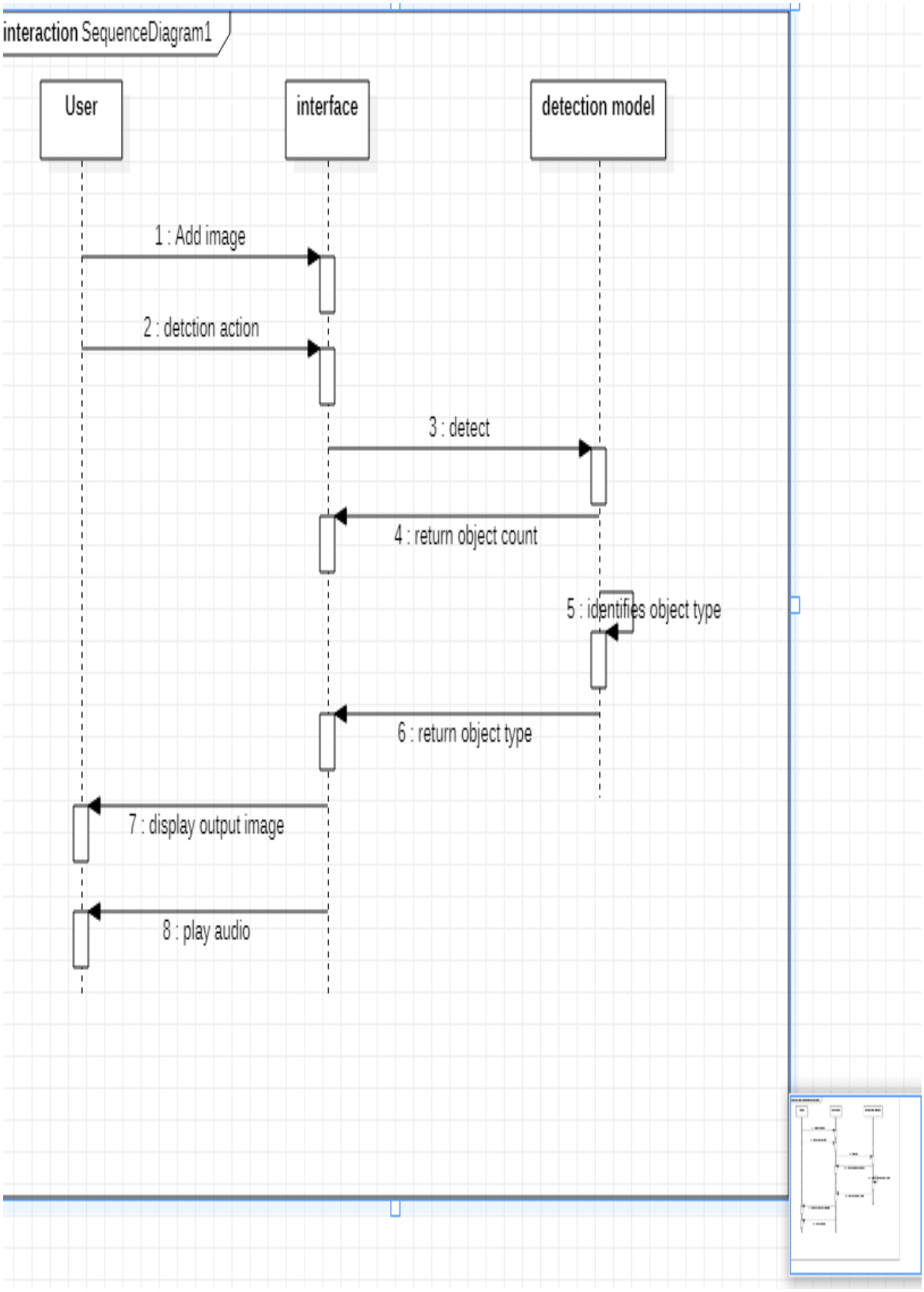


Fig 3 Sequence Diagram

4.3.4 Collaboration diagram

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time.

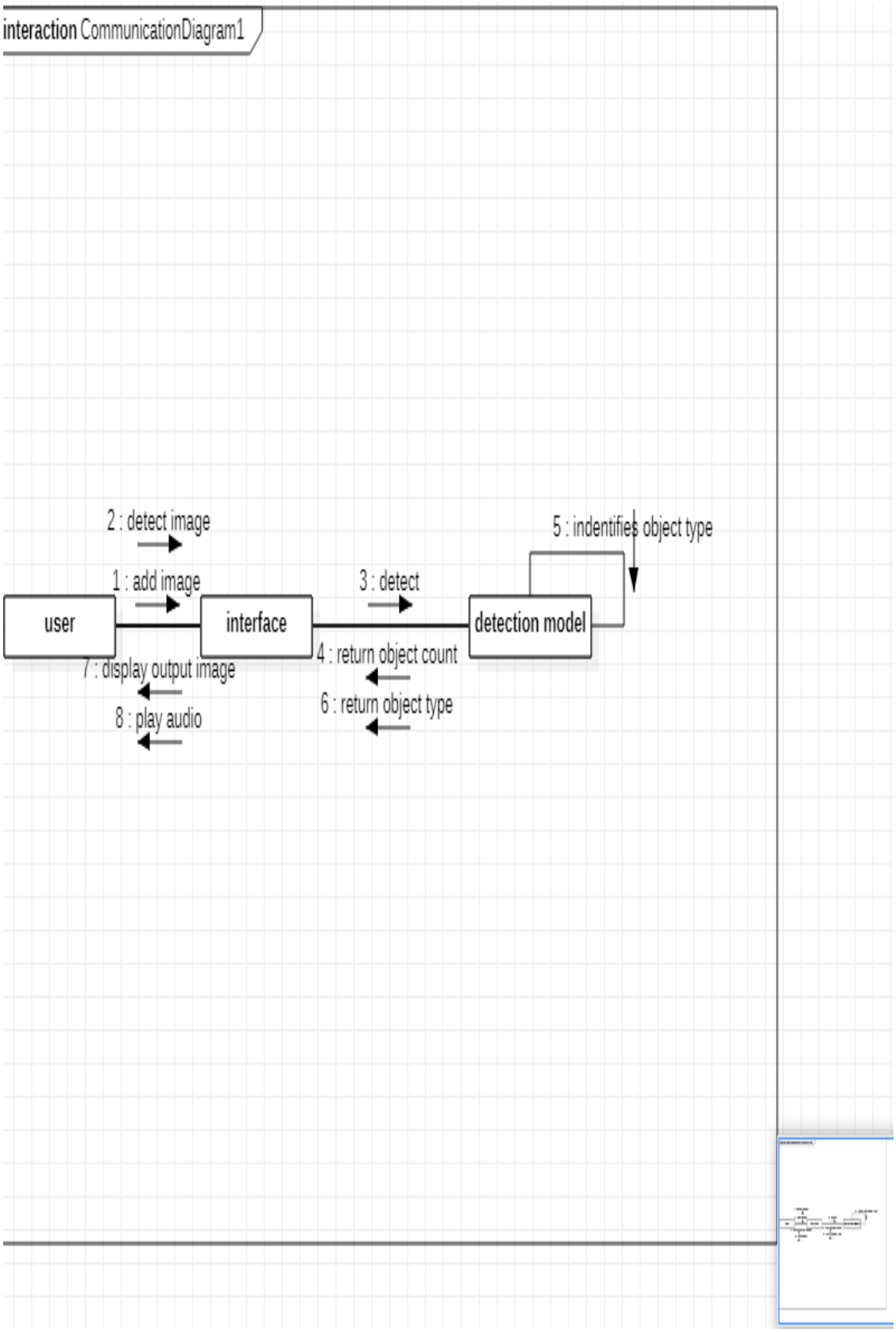


Fig 4 collaboration diagram

4.3.5 Deployment diagram

A deployment diagram in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

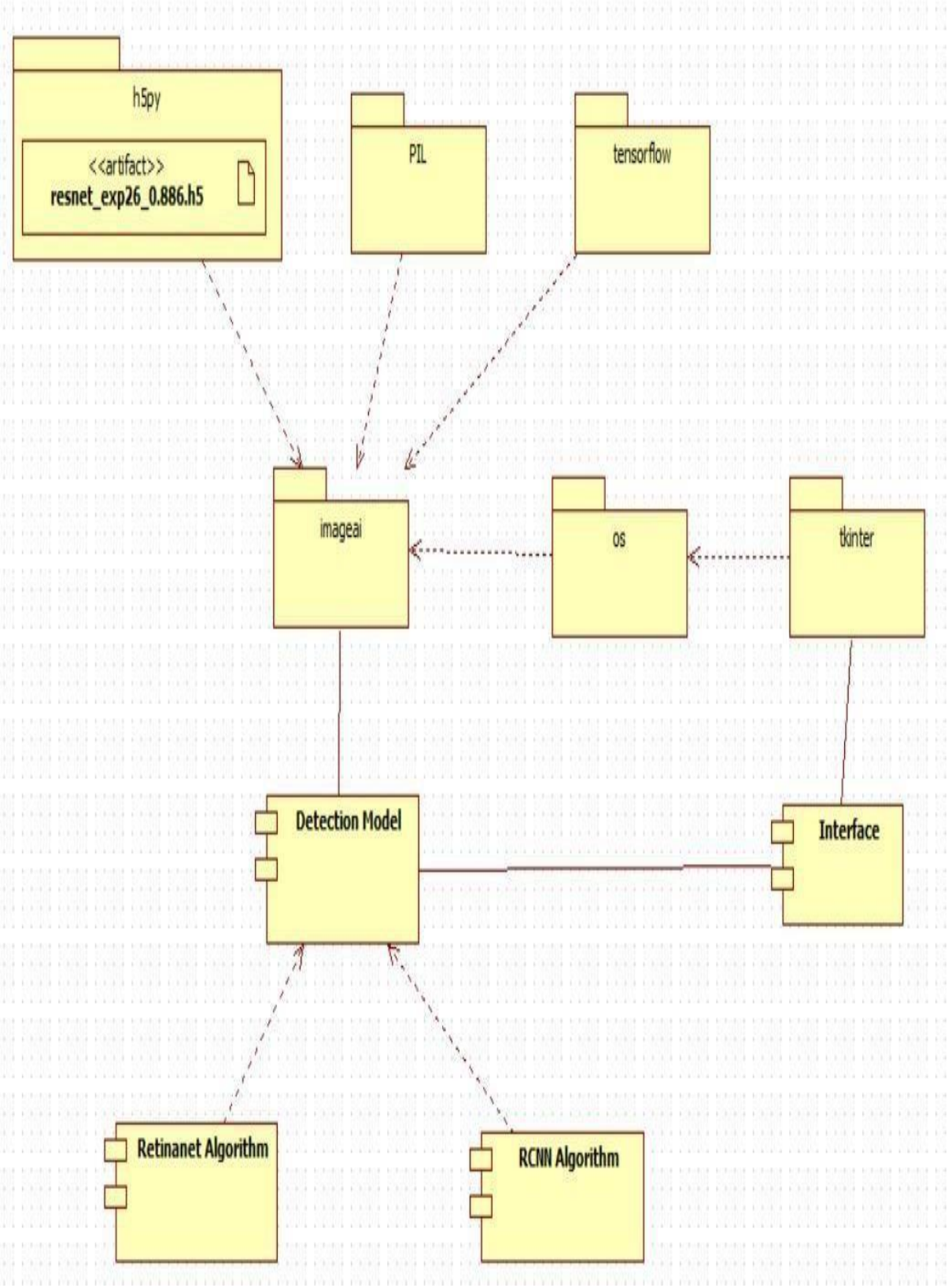


Fig 5 Deployment diagram

CHAPTER 5

CHAPTER 5

IMPLEMENTATION

5.1 SampleCodes

5.1.1 Main_code.py

```
#object_detection dependencies

from imageai.Detection import ObjectDetection


#video_detection dependecies

from imageai.Detection import VideoObjectDetection
import cv2

#user interface dependencies
from tkinter import *
from tkinter import ttk
from tkinter import filedialog
from PIL import Image, ImageTk

#voice_output dependencies
from gtts import gTTS
import os

#opening a file dialog to browse the image
file=[]

def fileDialog():

   global filename
   #filepath
   filename = filedialog.askopenfilename(initialdir = "/", title = "Select A File", filetype =
   (("jpeg files","*.jpg"),("all files","*.*")) )
   #displaying path
   label = Label(box1, text = "")
   label.grid(column = 1, row = 2)
   label.configure(text = filename)
#checking filename contains a video or an image

      #checking an image or not
      if ((".jpg" in filename) or (".JPG" in filename) or (".png" in filename) or (".PNG" in
filename)):
#loading an image
img = Image.open(filename)
img=img.resize((665,490),Image.ANTIALIAS)
```

```
    photo = ImageTk.PhotoImage(img)

#labelling the image
        label2 = Label(box2,image=photo)
        label2.image = photo
        label2.place(x=1,y=1)

        file.insert(0,filename)
      print(filename)

#checking a video or not
      elif ((".mp4" in filename) or(".MP4" in filename) or (".mkv" in filename) or (".MKV" in
    filename) or (".mvi" in filename) or (".MVI" in filename)):

        print(filename)
        camera = cv2.VideoCapture(0)
        os.system(filename)

#detecting the objects in an image
def objdetec():

      print("running : "+filename)


      if ((".jpg" in filename) or (".JPG" in filename) or (".png" in filename) or (".PNG" in
    filename)):

        #object detection begins from here
        detector = ObjectDetection()

        #model,input and output paths
        global model_path
        model_path = "E:/project_jarvis/face_recognition/object_detection/models/model.h5"

        input_path = filename
        output_path = "E:/project_jarvis/face_recognition/object_detection/op/opimage.jpg"

        #loading the trianed model
        detector.setModelTypeAsRetinaNet()
        detector.setModelPath(model_path)
        detector.loadModel()

        #detecting the objects in given image
        detection = detector.detectObjectsFromImage(input_image=input_path,
    output_image_path=output_path)

      l=[]
      s=set()
      for eachObject in detection:
```

```
        print(eachObject["name"] , " : " , eachObject["percentage_probability"] )
        if(eachObject["percentage_probability"] >15):
            l.append(eachObject["name"])
            s.add(eachObject["name"])
  #print(l,s)

  #voice output
        string ="hey !"
        if(len(s) == 1 and len(l) >1):
           string = string + "There are "+str(len(l))+" "+l[0]+"'s"+" before you ."
        elif(len(s) == 0):
            string = string + "Sorry there are no objects detected in the Image ."
        elif(len(s)==1 and len(l)==1):
           string = string + "There is a "+l[0]+" before you ."
        else:
           string = string + "There are "
           for i in s:
              string = string +str(l.count(i))+" "+str(i)+"'s "
           string = string +" before you."

        print(string)

        language = 'en'

        myobj = gTTS(text=string, lang=language, slow=False)

        myobj.save("welcome1.mp3")

elif ((".mp4" in filename) or(".MP4" in filename) or (".mkv" in filename) or (".MKV" in
filename)):

        #video_object_detection
        #def video_obj_detec():

        model_path = "E:/project_jarvis/face_recognition/object_detection/models/model.h5"
        output_path = "E:/project_jarvis/face_recognition/object_detection/op/video"
        input_path = filename


     camera = cv2.VideoCapture(0)


def forFrame(frame_number,output_array, output_count):
            print("for frame", frame_number)
            for dict in output_array:
                  for j in dict:
                        res = {key: dict[key] for key in dict.keys()
                    & {'name', 'percentage_probability'}}
                        print(str(res))
```

Multi Object Detection Using Machine Learning for Visually Impaired People

```
            print("output for each object:", output_array)
            print("output count for unique objects:", output_count)
            print("--end of frame--")

        detector = VideoObjectDetection()
        detector.setModelPath(model_path)
        detector.loadModel()

        video_path =
detector.detectObjectsFromVideo(input_file_path=input_path,output_file_path=output_path,
frames_per_second=2, per_frame_function=forFrame, minimum_percentage_probability=30,
log_progress=True)
        os.system(output_path)

        print(video_path)

#playing the audio for detected objects
def playAudio():

   os.system("welcome1.mp3")

#displaying detected image
def detectedImage():


#detctedImagePath="E:/project_jarvis/face_recognition/object_detection/op/opimage.jpg"

      img =
Image.open("E:/project_jarvis/face_recognition/object_detection/op/opimage.jpg")
      img=img.resize((665,540),Image.ANTIALIAS)
      photo = ImageTk.PhotoImage(img)

      label2 = Label(box3,image=photo)
      label2.image = photo
      label2.place(x=2,y=55)

#root instance to clss_tkinter
root = Tk()
root.title("object detection")
root.geometry("2200x1000")

#creating frame_box for opening a file-dialog
box1 = Frame(root, borderwidth=2, relief="solid",bg="light blue")
box1.place(x=5, y=5,width=675, height=100)

#displaying input image
box2 = Frame(root, borderwidth=2, relief="solid",bg="light blue")
box2.place(x=5, y=110,width=675, height=500)
```

```
#frame-button for running the detection_program
box4=Frame(root, borderwidth=2, relief="solid",bg="light blue")
box4.place(x=5, y=615,width=675, height=85)

#frame-button for displaying output detected image
box3=Frame(root, borderwidth=2, relief="solid",bg="light blue")
box3.place(x=685, y=5,width=680, height=605)

#frame-button for voice output
box5=Frame(root, borderwidth=2, relief="solid",bg="light blue")
box5.place(x=685, y=615,width=680, height=85)

#button-browse
btn = Button(box1, text = "Browse folder to open an
image",bd=5,bg="#234D42",fg="white",command=fileDialog)
btn.place(x=240,y=20)


#button-run
btn = Button(box4, text = "Run the
programme",bg="#234D42",fg="white",command=objdetec,bd=5)
btn.place(x=250,y=20)

#button-display-output-image
btn = Button(box3, text = "show the detcted
img",bd=5,bg="#234D42",fg="white",command=detectedImage)
btn.place(x=250,y=20)

#button-audio-output
btn = Button(box5, text = "play
audio",bg="#234D42",fg="white",command=playAudio,bd=5)
btn.place(x=280,y=20)


root.mainloop()
```

CHAPTER 6

CHAPTER 6

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 Itis the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

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.

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.

Functional test

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 is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It testsa 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.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed. Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.4 Testing Methodologies

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.

- User Acceptance Testing.
- Output Testing.
- Validation Testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design. The following are the types of Integration Testing:

1) Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2) Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.

- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

6.5 Validation Checking

Validation checks are performed on the following fields.

Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used

in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

User Training

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

6.6 Maintenance

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

Testing Strategy

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testingstrategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

CHAPTER 7

CHAPTER 7

RESULTS

7.1 Home Screen

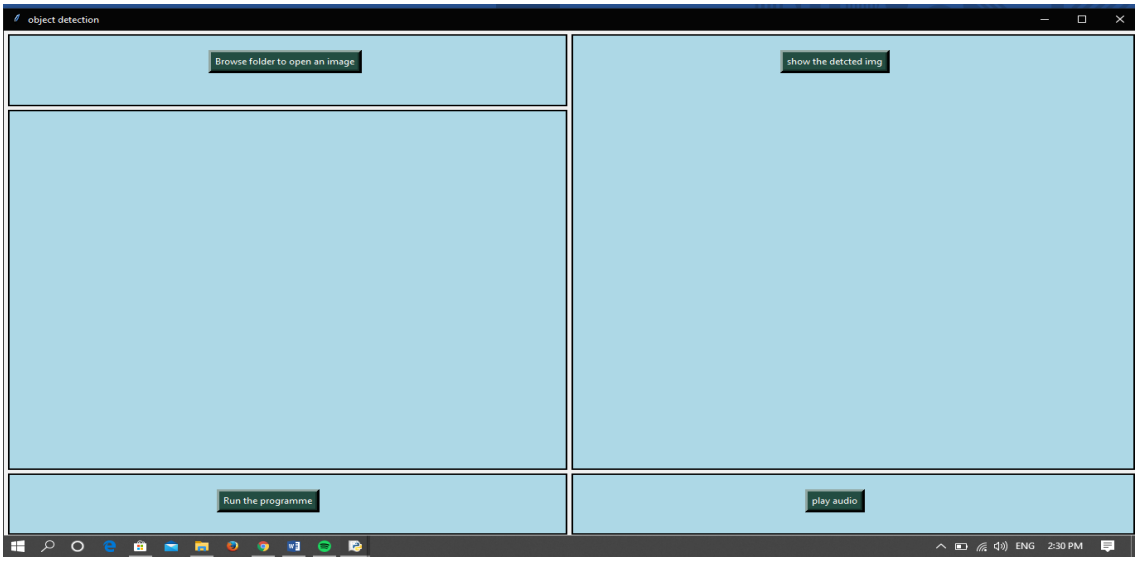


Fig 1: Screenshot of Home page

7.2 Opening a file dialog

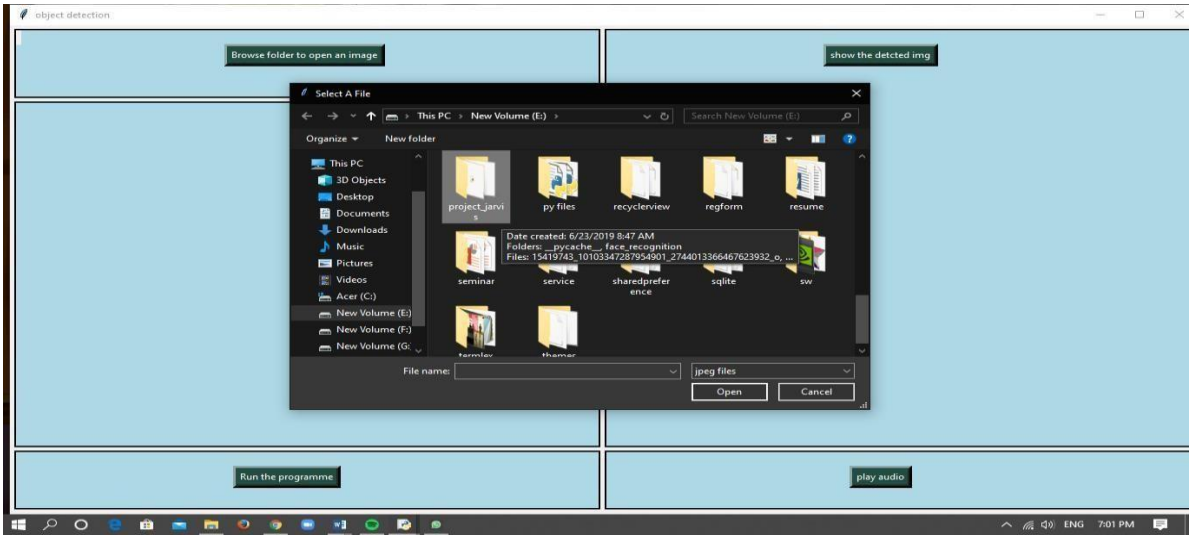


Fig 2: opening a file dialog to insert an image

7.3 Displaying Input image

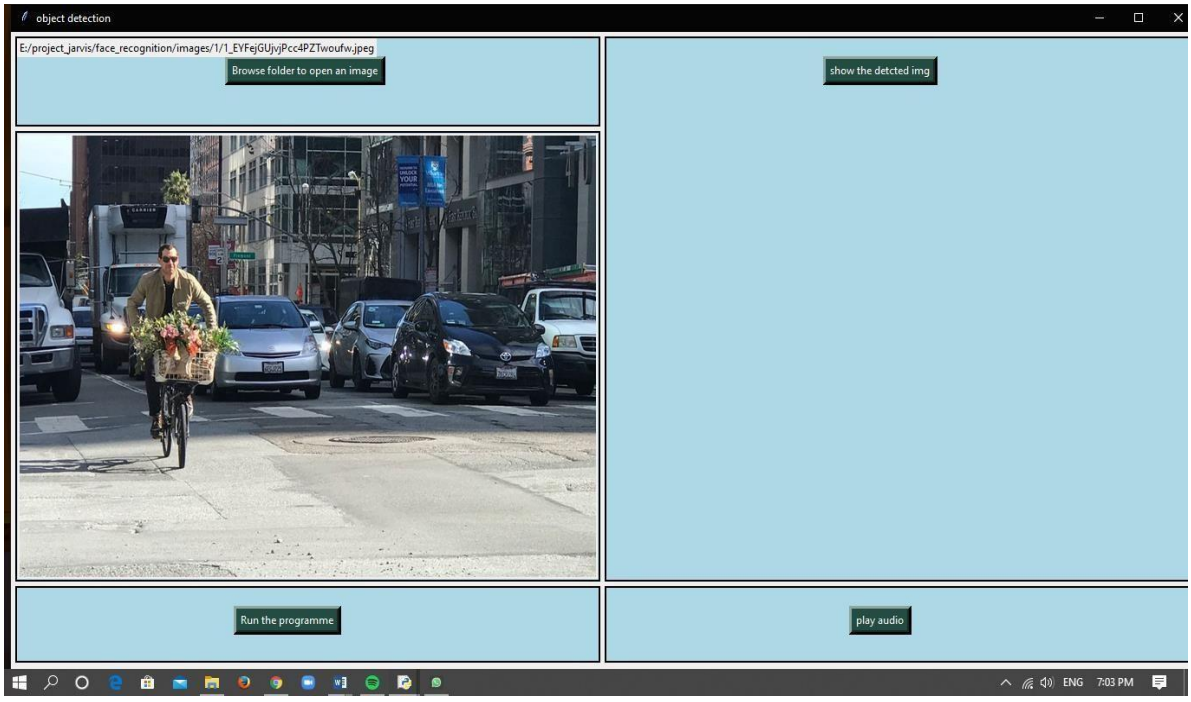


Fig 3: input image

7.4 Running program

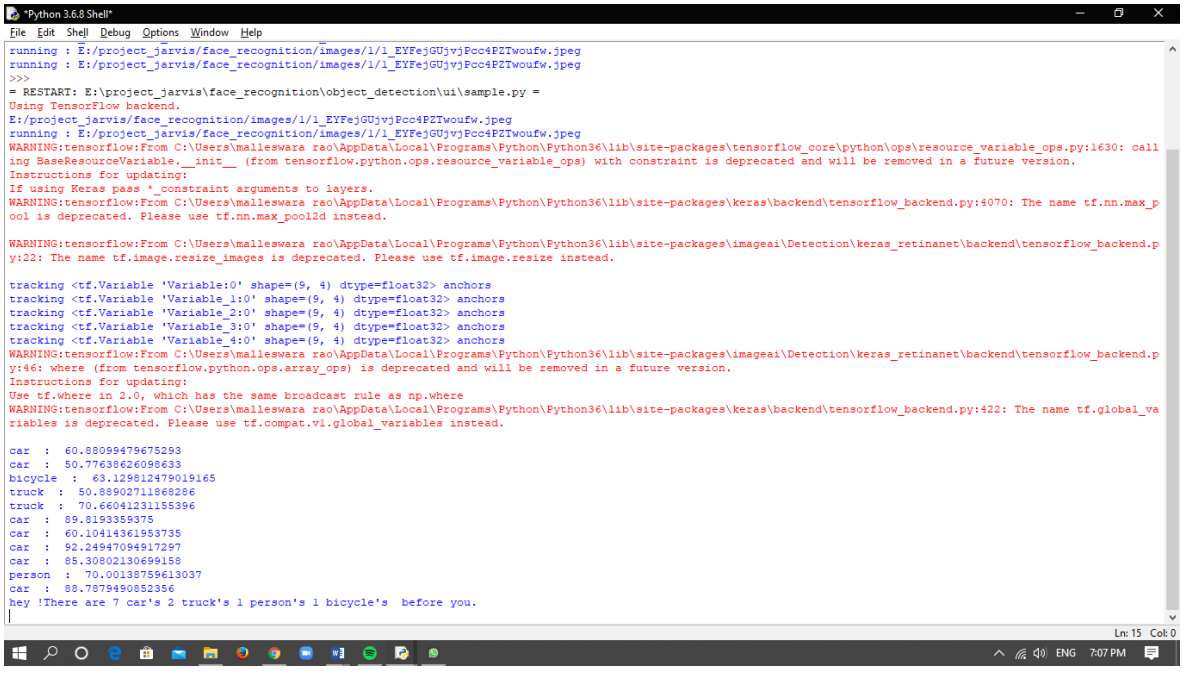


Fig 4: running the program

7.5 Displaying output image

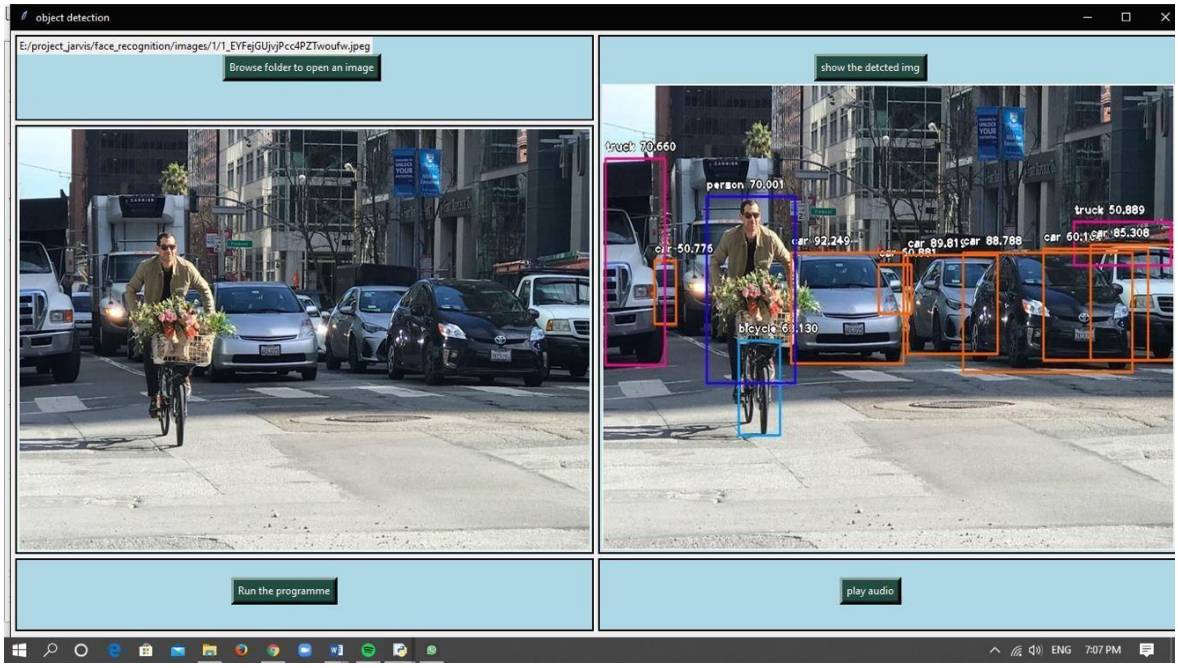


Fig 5: output image

7.6 Playing audio output

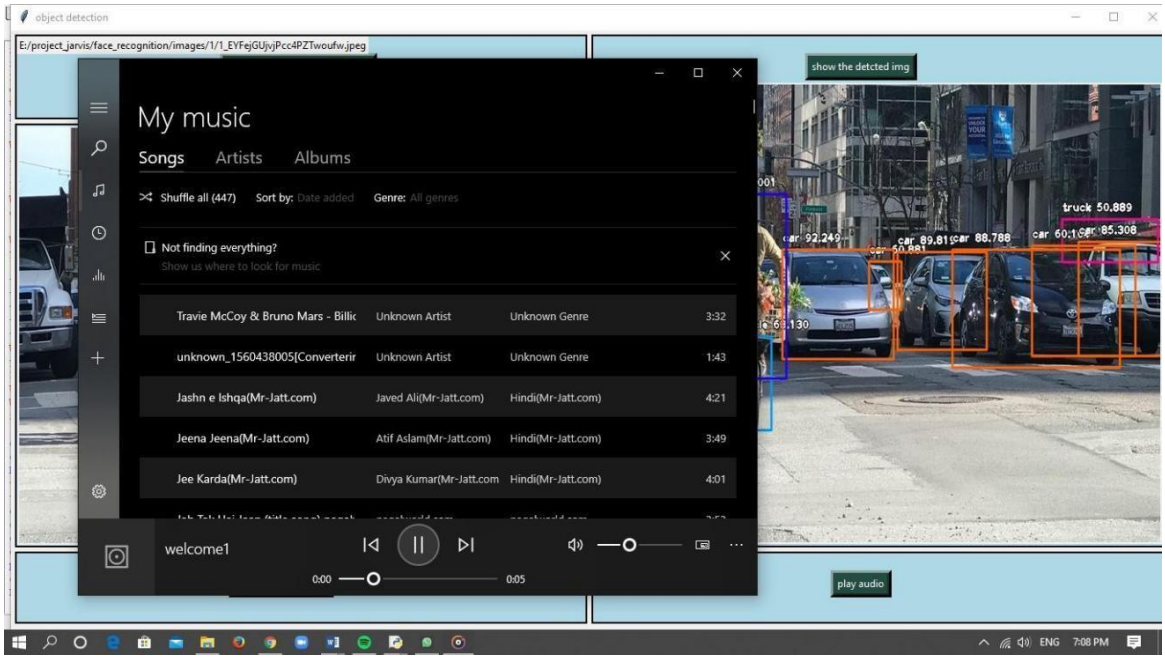


Fig 6: Audio output

CHAPTER 8

CHAPTER 8

CONCLUSION

Object Detection is an important task, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation and scene understanding, object tracking.

By using this thesis and based on experimental results we are able to detect objects more precisely and identify the multiple objects in an image individually with exact location of an object in the picture in X, Y axis along with their accuracy percentage of detection and also able to convert the list of objects in the image into speech thereby helping the visually impaired, solving their vision problem. This project also provides experimental results on different methods for object detection and identification and compares each method for their efficiencies.

CHAPTER 9

CHAPTER 9

REFERENCES

[1] Wei Y ,Xia W, Lin M, Huang J, Ni B, Dong J, Zhao Y, Yan S, HCP:A Flexible CNN framework for Multi-Label image classification, IEEE transactions on pattern analysis and machine intelligence 38 (9) (2016) 1901– 1907.

[2] Rim Trabelsi, Issam Jabri and Farid Melgani. Indoor Object recognition in RGBD images with complex valued neural networks for visually-impaired people, Neurocomputing (2018).

[3] Kiruthiga N, Divya E, HariPriya R, Haripriya V., "Real Time Object Recognition and classification using Deep Learning", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), March-April (2019).

[4] Liang Zhang and Michael Towsey. Using multi-label classification for acoustic pattern detection and assisting bird species surveys, Applied Acoustics (2016).

[5] H. Pirsiavash and D. Ramanam. Detecting activities of daily living in First person Camera Views, in: Computer Vision and Pattern Recognition605 (CVPR), 2012 IEEE, 2012, pp. 2847–2854.

[6] Mohamed Lamine Mekhalfi and Farid Melgani. A Compressive Sensing Approach to Describe Indoor Scenes for Blind People IEEE (July 2015).

[7] Y. J. Lee, J. Ghosh, K. Grauman, discovering important people and objects for egocentric vedio summarization, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp, 1346-1353.

[8] X. Yang and Y. Tian. Robust door detection in unfamiliar environments by combining edge and corner features, in: IEEE Conference on Computer Vision and Pattern Recognition (2010), pp, 57-64.

[9] F. M. Hasanuzzaman, X.Yang, Y.Tian,.Robust and effective component based banknote recognition for the blind,IEEE Transactions on systems, Man, and Cybernetics 42(6) (2012) 1021-1030.

[10] S. Hou, S. Zhou, L. Chen, Y. Feng. Multi label learning with label relevance in

advertising vedio, Neurocomputing 171 (2016) 1901-1907.

[11] Agarwal S, Awan A and Roth D (2004). Learning to detect objects in images via a sparse part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. 26, 1475-1490.

[12] Cadena C, Dick A, Reid I (2015).” A fast, modular scene understanding system using context-aware object detection”, in Robotics and Automation (ICRA), 2015 IEEE International Conference on (Seattle, Wa).

[13] Erhan D, Szegedy C, Toshev A, and Anguelov, D.(2014). “Scalable object detection using Deep neural networks”, in Computer vision and Pattern Recognition Frontiers in robotics and AI.

[14] Zhenyu Lu, Jia Lu, Quanbo Ge, Tianming zhan. “Multi-object Detection method based on Yolo and ResNet Hybrid Networks, in: IEEE $th International conference 2019.

[15] Zhong-Qui Zhao, Peng Zheng.”Object Detection with Deep Learning: A review. In: IEEE Transactions on NeuralNetworks and Learning system (2019).