VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI-590018

DIGITAL IMAGE PROCESSING LABORATORY WITH MINI PROJECT (18AIL67)

REPORT ON

"BLIND ASSISTANCE"

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

Bachelor of Engineering

In

Artificial Intelligence & Machine Learning

By

DISHA GUPTA

1KS20AI011

RAKSHITA KULKARNI

1KS20AI033

Under the guidance of

Prof. Roopa K Murthy	**Dr. Amulyashree S**
Asst. Prof, Dept. of AIML	Asst. Prof, Dept. of AIML

Department of Artificial Intelligence & Machine Learning

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

Department of Artificial Intelligence & Machine Learning

KSIT

K S INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to certify that Mini Project work entitled **"BLIND ASSISTANCE"** is carried out by **DISHA GUPTA** bearing USN **1KS20AI011** and **RAKSHITA KULKARNI** bearing USN **1KS20AI033** bonafide student of **K.S. Institute of Technology** in the partial fulfilment for the award of the **Bachelor of Engineering in Artificial Intelligence & Machine Learning** of the **Visvesvaraya Technological University, Belagavi**, during the year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini-project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said degree for the Sixth semester.

[illegible]

Prof. Anu Mathews
Asst. Professor & In-charge HOD,
Dept. of AIML, KSIT

[illegible]

Dr. Dilip Kumar K
Principal/Director, KSIT

[illegible] 11/7/2023

Prof. Roopa K Murthy
Asst. Prof, Dept. of AIML

[illegible] 11/7/23

Dr. Amulyashree S
Asst. Prof, Dept. of AIML

Name of the Examiners Signature with date

1. Sahana Sharma M [illegible] 13/7/23
2. Dr. Madhu B.R [illegible] 13/7/23

ACKNOWLEDGEMENT

I take this opportunity to thank everyone involved in the successful implementation of this mini project. I would like to thank the college for providing me an opportunity to work on the mini project.

I take this opportunity to express my sincere gratitude to my college **K.S. Institute of Technology**, Bengaluru for providing the environment to work on this mini project.

I would like to express my gratitude to our **MANAGEMENT,** K.S. Institute of Technology, Bengaluru, for providing a very good infrastructure and all the support provided for carrying out this mini project work in college.

I would like to express my gratitude to **Dr. K.V.A Balaji, CEO**, K.S. Institute of Technology, Bengaluru, for his valuable guidance.

I would like to express my gratitude to **Dr. Dilip Kumar K**, **Principal/Director**, K.S. Institute of Technology, Bengaluru, for his continuous support.

I like to extend my gratitude to **Prof. Anu Mathews**, **Asst. Professor and In-charge HOD**, Department ofArtificial Intelligence & Machine Learning, for providing very good facilities and all the supportin successfully carrying out this Mini Project.

I also like to thank my Mini Project Coordinators, **Dr. Amulyashree S, Asst. Professor, Prof. Roopa K Murthy, Asst. Professor,** Department of Artificial Intelligence & Machine Learning, for their help and support in successfully carrying out the Mini Project work.

I am also thankful to the teaching and non-teaching staff of the Artificial Intelligence & Machine Learning Department, KSIT for the help provided in completing this Mini Project.

DISHA GUPTA

1KS20AI011

RAKSHITA KULKARNI

1KS20AI033

ABSTRACT

This project presents a vision enhancer module designed specifically for blind individuals. The system utilizes a mobile application that sends real-time frames to a laptop-based wireless networked system. The core feature of the system is real-time object detection using the SSD_MOBILENET algorithm and TensorFlow-APIs. Additionally, the system incorporates approximate distance calculation and voice-based wireless feedback generation. This functionality allows the blind user to receive wireless voice-based feedback regarding the distance of detected objects, indicating whether they are too close or at a safer distance. The system aims to simplify, improve efficiency, and provide reliable assistance for blind individuals by offering wireless feedback for object detection and obstacle voidance.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 What is digital image processing?

Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems. The generation and development of digitalimage processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry and medical science has increased.

Many of the techniques of digital image processing, or digital picture processing as it often was called, were developed in the 1960s, at Bell Laboratories, the Jet Propulsion Laboratory, Massachusetts Institute of Technology, University of Maryland, and a few other research facilities, with application to satellite imagery, wire-photo standards conversion, medical imaging, videophone, character recognition, and photograph enhancement. The purpose of early image processing was to improve the quality of the image. It was aimed for human beings to improve the visual effect of people. In image processing, the input is a low-quality image, and the output is an image with improved quality. Common image processing include image enhancement, restoration, encoding, and compression. The first successful application was the American Jet Propulsion Laboratory (JPL). They used image processing techniques such as geometric correction, gradation transformation, noise removal, etc. on the thousands of lunar photos sent back by the Space Detector Ranger 7 in 1964, taking into account the position of the Sun and the environment of the Moon. The impact of the successful mapping of the Moon's surface map by the computer has been a success. Later, more complex image processing was performed on the nearly 100,000 photos sent back by the spacecraft, so that the topographic map, color map and panoramic mosaic of the Moon were obtained, which achieved extraordinary results and laid a solid foundation for human landing on the Moon.

1.2 Problem Statement

The problem addressed by the Blind Assistance Project is the lack of effective assistance and independence for individuals with visual impairments in interpreting and navigating their surroundings. Visual impairments pose significant challenges, limiting the mobility, safety, and overall quality of life for blind or visually impaired individuals.

1.3 Need to solve the problem

The need to solve the problem stated in the problem statement arises from the significant challenges faced by blind individuals in interpreting and navigating their surroundings.

- Independence and Mobility: Blind individuals often rely on external assistance or aids such as white cane to navigate their surroundings. However, these methods have limitations in providing real-time information about the immediate environment.
- Safety and Avoidance of Obstacles: Blind individuals face safety risks due to their inability to perceive obstacles or hazards in their path. The blind assistance model can help by detecting and alerting users to obstacles, allowing them to avoid potential collisions or accidents.
- Spatial Awareness and Environmental Understanding: Understanding the layout and objects in the surrounding environment is crucial for blind individuals to have better understanding of their surroundings and enhancing their spatial awareness.

1.4 What you propose to do in the work

- Development of a **Digital Image Processing Model**: Develop a model that utilizes digital image processing techniques to capture and analyze real-time images of the surrounding environment. This model should be capable of detecting and recognizing objects.
- **Real-time Object Detection**: Implement advanced object detection algorithms, such as SSD_MOBILENET, using TensorFlow APIs or similar frameworks, to accurately detect and localize objects within the captured images. This will provide blind individuals with essential information about their surroundings.

- **Approximate Distance Calculation:** Incorporate algorithms and techniques for estimating thedistance between the blind user and detected objects. This functionality will allow the model to provide approximate distance feedback to the user, enabling them assess the proximity of objects and make informed decisions regarding their movements.
- **Voice-Based Wireless Feedback:** Integrate a voice-based feedback method into the project, which provides wireless communication between the image processing system and the blind user. The model should convert the detected object information and approximate distance calculations into voice-based feedback, allowing the user to receive real-time guidance and alerts.

CHAPTER 2

PREVIOUS STUDIES

Sno.	Paper Title	Outcomes	Shortcoming
1	Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey.	This aims to inform both the research community and visually impaired individuals about the progress in assistive technology and navigation.	Electronic travel aids navigation guidance is not giving the required. accuracy.
2	Digital image processing.	By presenting novel results in image restoration, it contribute to the existing body of research in the field.	It does not include image segmentation, feature extraction, or pattern recognition
3	Real time text detection and recognition on hand held objects to assist blind people.	The system addresses the challenge of detecting and recognizing text patterns on various objects.	The system primarily focuses on text recognition.
4	Smart Eye for Visually Impaired-An aid to help the blind people.	It is a potential solution to address the challenges faced by visually impaired people.	Cost or scalability aspects are not feasible.
5	Smart Machine Learning System for Blind Assistance.	Development of a flexible and user-friendly guiding mechanism.	It does not include the model training for all sets of data.
6	Visual Assistance for Blind Using Image Processing.	This will help the visually impaired person to manage day-to-day activities and to navigate through surroundings. Raspberry Pi is used to implement artificial vision using python	Unclear how well the system how well the system performs in real-world scenarios for detecting objects.

Sno.	IEEE PAPERS	Outcomes	Shortcoming
7	PARTHA: A Visually Impaired Assistance System.	The proposed system is reliable, affordable, practical and feasible.	Lack of information on user feedback or usability testing.
8	Blind user wearable audio assistance for indoor navigation based on visual markers and ultrasonic obstacle detection.	It resulted in high rates of successful recognition of visual markers and perception of ultrasonic obstacles.	Not clear on the user interface or the accessibility features implemented to ensure ease of use of blind individuals.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

Processor: AMD Ryzen 5Memory: 4GB or more

3.2 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

Programming Language: Python, Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code reliability with the use of significant indentation via the off-side rule.

IDE: Jupyter Notebook, The Python Notebook is now known as the Jupyter Notebook. It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media.

Operating System:

1) **Windows:**

   Windows operating system, specifically Window 10, includes a range of accessibility features. It offers the Narrator screen reader, Magnifier, and various keyboard and mouse options foraccessibility.

2) **Linux:**

   Linux distributors, such as Ubuntu, can be customized to incorporate accessibility features. It provides a flexible and open-source development environment with various programming toolsand libraries.

3) **iOS:**

   iOS, the operating system for Apple devices, offers a comprehensive set of accessibility features. It provides a well-integrated development environment.

Web Brower:

4) **Google Chrome:**

It is a widely used web browser that provides a rich set of developer tools and extensions. It supports HTML5, JavaScript, which are commonly used technologies for implementing image processing algorithms within a web development.

5) **Microsoft Edge:**

Edge is the default web browser on Windows 10 and offers a fast and secure browsing experience. It supports modern web technologies, including HTML5, JavaScript making it suitable for implementing digital image processing tasks in web applications.

3.3 TECHNOLOGY

3.3.1 SSD DETECTION MODEL

- Single Shot Detection (SSD) refers to a specific object detection algorithm that is designed for real-time object detection.
- It is a framework for object detection that combines high accuracy with real-time performance. It is based on the concept of dividing the input images into a grid of cells and predictive multiple bounding boxes and class probabilities within each cell.

3.3.1.1 SSD ARCHITECTURE

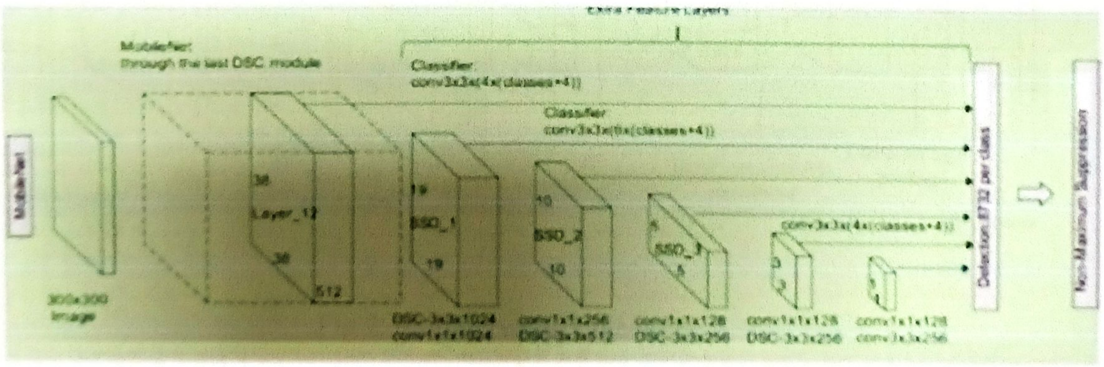

Fig 1. Architecture of SSD

- SSD has two components: an SSD head and a backbone model.
- The backbone model is a trained image classification network as a feature extractor. Like Res Net, this is typically a network trained on ImageNet from which the final fully connected classification layer has been removed.
- The SSD head is nothing but one or more convolutional layers added to the backbone. The outputs are explained as the bounding boxes and classes of objects in the spatial location of the final layer's activations. As a result, we have a deep neural network that can extract the meaning of the input image while maintaining its spatial structure at a lower resolution.
- For an input image, the backbone results in 256 7x7 feature maps in ResNet34. SSD classifies the image using a grid and grid cell responsible for detecting objects in the picture region. Detecting objects means predicting the class and location of an object within that region.

3.3.2 MOBILE NET

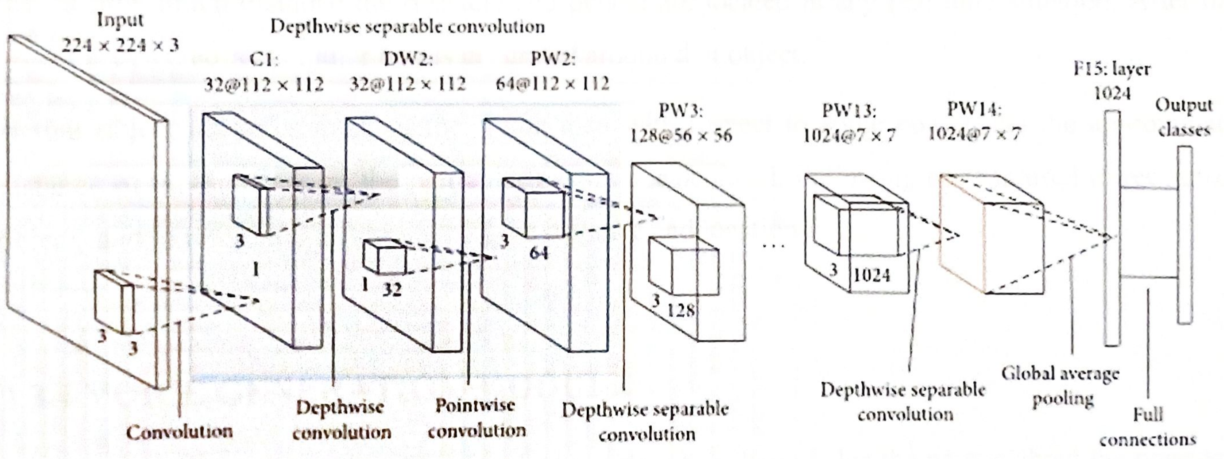

Fig 2. Mobile Net Architecture

This model is based on the ideology of THE Mobile Net model based on depth wise separable convolutions and it forms a factorized Convolutions. This converts a basic standard convolutions into a depth wise convolutions. This 1 × 1 convolutions are also called as pointwise convolutions. For Mobile Nets to work, these depth wise convolution applies a general single filter based concept to each of the input channels. These pointwise convolutions applies a 1 × 1 convolutions to merge with the outputs of the depth wise convolutions. As a standard convolution both filters combines the inputs into a new set of outputs in one single step. The depth wise identifiable convolutions splits this into two layers — a separate layer for the filtering purpose and the other separate layer for the combining purpose. This factorization methodology has the effect of drastically reducing the computation and that of the model size.

3.3.3 DEPTH ESTIMATION

Depth estimation or extraction feature is nothing but the techniques and algorithms which aims to obtain a representation of the spatial structure of a scene. In simpler words, it is used to calculate the distance between two objects. Our prototype is used to assist the blind people which aims to issue warning to the blind people about the hurdles coming on their way. In order to do this, we need to find that at how much distance the obstacle and person are located in any real time situation. After the object is detected rectangular box is generated around that object.

If that object occupies most of the frame then with respect to some constraints the approximate distance of the object from the particular person is calculated. Following code is used to recognize objects and to return the information of the distance and location.

3.3.4 VOICE GENERATION MODULE

After the detection of an object, it is utmost important to acknowledge the person about the presence of that object on his/her way. For the voice generation module PYTTSX3 plays an important role. Pyttsx3 is a conversion library in Python which converts text into speech. This library works well with both Python 2 and 3. To get reference to a pyttsx. Engine instance, a factory function called as

This algorithm works as whenever an object is being detected, approximate distance is being calculated. with the help of cv2 library and cv2.putText() function, the texts are getting displayed on to the screen. To identify the hidden text in an image, we use Python-tesseract for character recognition. OCR detects the text content on images and encodes it in the form which is easily understood by the computer. This text detection is done by scanning and analysis of the image. Thus, the text embedded in images are recognized and "read" using Python-tesseract. Further these texts are pointed to a pyttsx.Engine instance, a factory function called as pyttsx.init() is invoked by an application. During construction, a pyttsx.driver. DriverProxy object is initialized by engine which is responsible for loading a speech engine driver from the pyttsx drivers module. After construction, an object created by an engine is used by the application to register and unregister event callbacks; produce and stop speech; get and set speech engine properties; and start and stop event loops.

CHAPTER 4

METHODOLOGY

4.1 Flow Chart of the work

The following figure illustrates the flow chart of the model

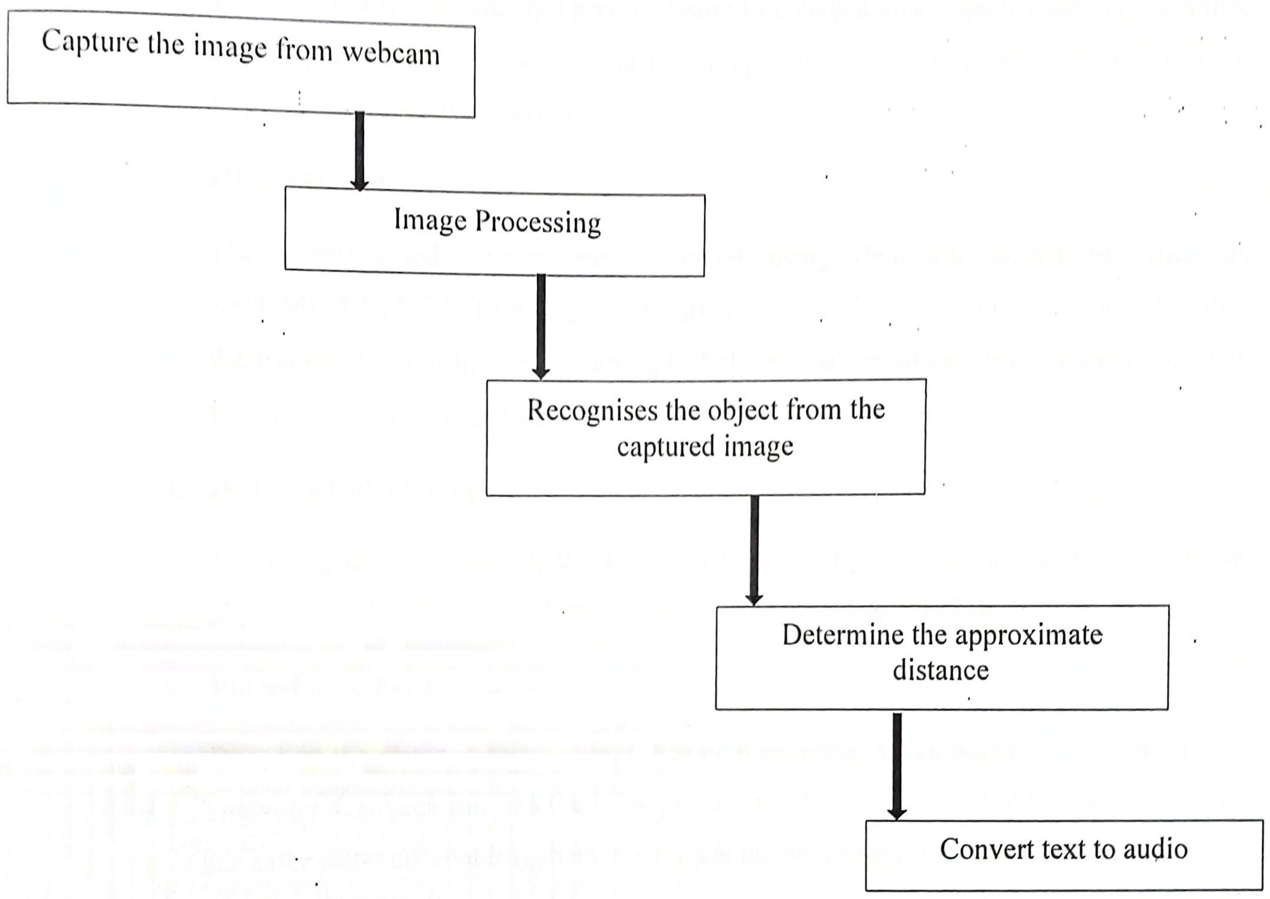
Capture the image from webcam
Image Processing
Recognises the object from the captured image
Determine the approximate distance
Convert text to audio

Fig 3: Flow Chart of the model proposed

4.2 Explanation of Methodology:

The following section explains the methodology of the work-

1. **Data Acquisition:**

   The system utilizes a camera to capture real-time frames of the surrounding environment.

2. **Preprocessing:**

   The captured frames undergo preprocessing to enhance image quality and reduce noise. This step may involve techniques such as image filtering, resizing, or color normalization to prepare for object detection.

3. **Object Detection:**

   The preprocessed frames are processed using detection algorithms, such as SSD_MOBILENET. These algorithms are trained to recognize and locate objectswithin the frames. They utilize deep learning techniques and neural networks to identifyobjects based on their visual features.

4. **Distance Calculation:**

   Once objects are detected, the system determines the approximate distance of these objects and also shows the Warning signal if the object is too close.

5. **Voice-Based Feedback:**

   Based on the distance calculation, the system generates voice-based feedback. The voice feedback informs the blind person whether the detected object is too close or at a safer distance, enabling them to navigate the environment more effectively.

6. **Real-Time Processing:**

   The entire methodology is designed to work in real-time, continuously capturing frames, performing object detection and distance calculation, and generating feedback. This ensures that the blind assistance system an provide timely and relevant informationto the user as they navigate their surroundings.

CHAPTER 5

RESULTS

5.1 Outcomes from the methodology

The following section explains the outcomes-

1. **Object Detection:**

   The system can accurately detect and identify objects in the environment, enabling the blind person to be aware of their presence. This outcome helps in recognizing various objects such as person, bottles, watches, or any other individuals

2. **Distance Estimation:**

   By calculating the approximate distance of detected objects, the system provides information about their proximity to the blind person. This outcome allows the user to understand whether an object is too close or at a safer distance, helping them make informed decisions on their movements.

3. **Voice-Based Feedback:**

   The system generates voice-based feedback to provide real-time information. This outcome allows the user to receive auditory guidance and make appropriate adjustments in their navigation.

4. **Real-Time Assistance:**

   By performing all these tasks in real-time, the methodology ensures that the blind person receives immediate and up-to-date information about their surroundings. This outcome allows for timely response and decision making, enhancing the overall safety and independence of the user.

CHAPTER 6

IMPLEMENTATION CODE

CODE:

```
import numpy as np
import os
import six.moves.urllib as urllib
import urllib.request as allib
import sys
import tarfile
import tensorflow as tf
import zipfile
import time
import pytesseract   #It allows to extract text from images
#import engineio

import torch
from torch.autograd import Variable as V
import models as models
from torchvision import transforms as trn
from torch.nn import functional as F


import pyttsx3           #CONVERTS PYTHON TEXT TO SPEECH
#from .engine import Engine
engine =pyttsx3.init()

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

arch = 'resnet18'

model_file = 'whole_%s_places365_python36.pth.tar' % arch
if not os.access(model_file, os.W_OK):
    weight_url = 'http://places2.csail.mit.edu/models_places365/' + model_file
    os.system('wget ' + weight_url)


pytesseract.pytesseract.tesseract_cmd   =   r'C:\Users\Sandeep   Gupta\Desktop\DIP
PROJECT\tesseract-main\tesseract-main'

# from object_detection.utils import label_map_util
#/object_detection/'   m2
import sys
sys.path.append('object_detection')

from utils import label_map_util


#from object_detection.utils import label_map_util

from utils import visualization_utils as vis_util
MODEL_NAME = 'ssd_inception_v2_coco_2017_11_17'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
```

```
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90


if not os.path.exists(MODEL_NAME + '/frozen_inference_graph.pb'):
print ('Downloading the model')
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)          #CONVERTING TO tar INFERENCES FILE
for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
    tar_file.extract(file, os.getcwd())
print('Download complete')
else:
print ('Model already exists')

#for detection using tensorflow graph modules
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.compat.v1.GraphDef()      #with GraphDef() func we are making
it compatible with the tensorflow
  #with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
  with tf.compat.v2.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')



label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories      =      label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

url='http://10.67.208.240:8080//shot.jpg'

#using the concept of computer vision
import cv2
cap = cv2.VideoCapture(0)    #for video capturing
#if using laptop camera put '0' in parenthesis, if using secondary source for the
camera put '1' in parenthesis


#for detection establishing a session, i.e, graph detection session
with detection_graph.as_default():
  with tf.compat.v1.Session(graph=detection_graph) as sess:
   ret = True  #ret(return type) is true, read the image and it will store under
image_np and it will wait for 20ms
   while (ret):
      ret,image_np = cap.read()

      if cv2.waitKey(20) & 0xFF == ord('b'):

          cv2.imwrite('opencv'+'.jpg', image_np)
  model_file = 'whole_%s_places365_python36.pth.tar' % arch
          if not os.access(model_file, os.W_OK):
              weight_url    =    'http://places2.csail.mit.edu/models_places365/'    +
model_file
              os.system('wget ' + weight_url)

          useGPU = 1
          if useGPU == 1:
              model = torch.load(model_file)
          else:
              model   =   torch.load(model_file,  map_location=lambda  storage,  loc:
storage) # model trained in GPU could be deployed in CPU machine like this!


          model.eval()        #evaluating the model,   the model is used to make
predictions on unseen data without updating its parameters.

          centre_crop = trn.Compose([
                trn.Resize((256,256)), # Resizes the input image to a target size
of 256x256 pixels.
                trn.CenterCrop(224),     #Performs a center crop operation on the
resized image, extracting a square region of size 224x224 pixels from the center of
the image.
                trn.ToTensor(),         # Converts the image from a PIL Image object
to a PyTorch tensor. This transformation also converts the image pixel values to the
range [0, 1] by dividing them by 255.,Converting the image to a tensor allows it to
be directly used as input to the deep learning model.
                trn.Normalize([0.485,   0.456,   0.406],   [0.229,   0.224,   0.225])
#normalizes the tensor image by subtracting the mean and dividing by the standard
deviation. The provided mean and standard deviation values [0.485, 0.456, 0.406] and
[0.229, 0.224, 0.225] respectively,
          ])


      #object detection part
          file_name = 'categories_places365.txt'
          if not os.access(file_name, os.W_OK):
              synset_url                                                           =
'https://raw.githubusercontent.com/csailvision/places365/master/categories_places3
65.txt'
              os.system('wget ' + synset_url)
          classes = list()
          with open(file_name) as class_file:
              for line in class_file:
                  classes.append(line.strip().split(' ')[0][3:])        #splitting
into precision of 0 and 3
          classes = tuple(classes)


          #It will open the image and check if it fits on which are model is trained,
  if that is among the trained version of ssd model then it will print the first
  part otherwise it will go forward
        # prepares the input image for classification by opening it, applying pre-
processing transformations, and creating a tensor representation of the image to be
used as input to the model.
          img_name = 'opencv.jpg'
          if not os.access(img_name, os.W_OK):
              img_url = 'http://places.csail.mit.edu/demo/' + img_name
              os.system('wget ' + img_url)

          img = Image.open(img_name)
```

```
input_img = V(centre_crop(img).unsqueeze(0), volatile=True)

        logit = model.forward(input_img)
        h_x = F.softmax(logit, 1).data.squeeze()
        probs, idx = h_x.sort(0, True)
      #after checking it will print the possible scenes
        print('POSSIBLE SCENES ARE: ' + img_name)
        engine.say("Possible Scene may be")          #python text-to-speech is
initialised in engine in starting of the code(engine.say() will predict it)
        engine.say(img_name)

      #check for classes & will print the versions as per that
        for i in range(0, 5):
            engine.say(classes[idx[i]])
            print('{}'.format(classes[idx[i]]))


      # Expand dimensions since the model expects images to have shape: [1, None,
None, 3]
      image_np_expanded = np.expand_dims(image_np, axis=0)
      image_tensor     =     detection_graph.get_tensor_by_name('image_tensor:0')
#image_tensor is the TensorFlow tensor object representing the input image
      # Each box represents a part of the image where a particular object was
detected.
      boxes    =    detection_graph.get_tensor_by_name('detection_boxes:0')    #The
coordinates of the bounding boxes typically represent the position and size of the
detected objects within an image.
      # Each score represent how level of confidence for each of the objects.
      # Score is shown on the result image, together with the class label.
      scores = detection_graph.get_tensor_by_name('detection_scores:0')
      classes = detection_graph.get_tensor_by_name('detection_classes:0')
      num_detections  =  detection_graph.get_tensor_by_name('num_detections:0')  #
This tensor represents the number of valid detections in the image
      # Actual detection.
      (boxes, scores, classes, num_detections) = sess.run(
          [boxes, scores, classes, num_detections],
          feed_dict={image_tensor: image_np_expanded})



      # Visualization of the results of a detection.
      if cv2.waitKey(2) & 0xFF == ord('a'):
          vis_util.vislize_boxes_and_labels_on_image_array(
          image_np,
          np.squeeze(boxes),
          np.squeeze(classes).astype(np.int32),
          np.squeeze(scores),
          category_index,
          use_normalized_coordinates=True,
          line_thickness=8)
      else:
          vis_util.visualize_boxes_and_labels_on_image_array(
              image_np,
              np.squeeze(boxes),
              np.squeeze(classes).astype(np.int32),
              np.squeeze(scores),
              category_index,
              use_normalized_coordinates=True,
              line_thickness=8)
```

```
    if cv2.waitKey(2) & 0xFF == ord('r'):
     text=pytesseract.image_to_string(image_np)

     engine.say(text)
     engine.runAndWait()


  for i,b in enumerate(boxes[0]):


    #                    car                     bus                    truck
    if classes[0][i] == 3 or classes[0][i] == 6 or classes[0][i] == 8:
      if scores[0][i] >= 0.5:          #if precision score is greater than 50%,
it will go for calculating the distance
        mid_x = (boxes[0][i][1]+boxes[0][i][3])/2      #calculating the average
x and average y values
        mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
        apx_distance = round(((1 - (boxes[0][i][3] - boxes[0][i][1]))**4),1)
#rounding it with the precision of 1 and on scale of 1 calculating the approx
distance
        cv2.putText(image_np,                     '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,255), 2)

        if apx_distance <=0.5:  #if the approx distance is less than 0.5, it
means the object is too close to the person
          if mid_x > 0.3 and mid_x < 0.7:
            cv2.putText(image_np,          'WARNING!!!',              (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
            print("Warning -Vehicles Approaching")
            engine.say("Warning -Vehicles Approaching")  #convert to speech
            engine.runAndWait()

    if classes[0][i] ==44:       #bottle
        if scores[0][i] >= 0.5:
            mid_x = (boxes[0][i][1]+boxes[0][i][3])/2
            mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
            apx_distance      =     round(((1      -      (boxes[0][i][3]      -
boxes[0][i][1]))**4),1)
            cv2.putText(image_np,                     '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,255), 2)
            print(apx_distance)
            engine.say(apx_distance)
            engine.say("units")
            engine.say("BOTTLE IS AT A SAFER DISTANCE")


            if apx_distance <=0.5:
                if mid_x > 0.3 and mid_x < 0.7:
                    cv2.putText(image_np,        'WARNING!!!',        (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
                    print("Warning -BOTTLE very close to the frame")
                    engine.say("Warning -BOTTLE very close to the frame")
                    engine.runAndWait()
    if classes[0][i] ==1:        #person
        if scores[0][i] >= 0.5:
            mid_x = (boxes[0][i][1]+boxes[0][i][3])/2
            mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
            apx_distance      =     round(((1      -      (boxes[0][i][3]      -
boxes[0][i][1]))**4),1)
            cv2.putText(image_np,                     '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,255), 2)
```

```
print(apx_distance)
engine.say(apx_distance)
                    engine.say("units")
                    engine.say("Person is AT A SAFER DISTANCE")


                if apx_distance <=0.5:
                    if mid_x > 0.3 and mid_x < 0.7:
                        cv2.putText(image_np,       'WARNING!!!',       (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
                        print("Warning -Person very close to the frame")
                        engine.say("Warning -Person very close to the frame")
                        engine.runAndWait()




#       plt.figure(figsize=IMAGE_SIZE)
#       plt.imshow(image_np)
      #cv2.imshow('IPWebcam',image_np)
      cv2.imshow('image',cv2.resize(image_np,(1024,768)))
      if cv2.waitKey(2) & 0xFF == ord('t'):       #if we press 't' the open-cv
module or  the camera will get closed and it will wait for 2 secs
          cv2.destroyAllWindows()
          cap.release()
          break
```

SNAPSHOTS

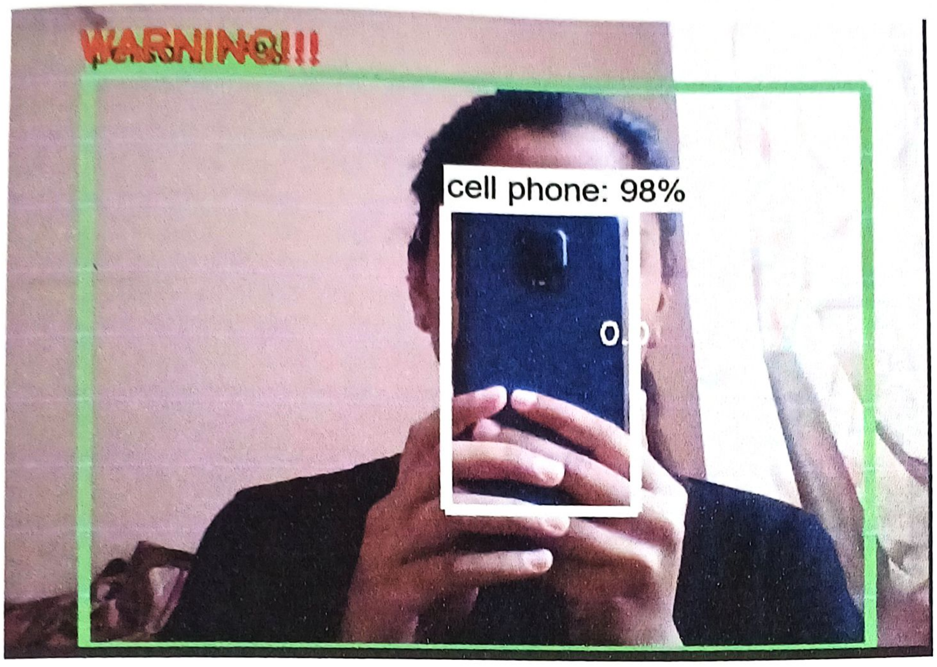

Fig 4. The snapshot of identified person and cell phone from real-time as seen in the camera

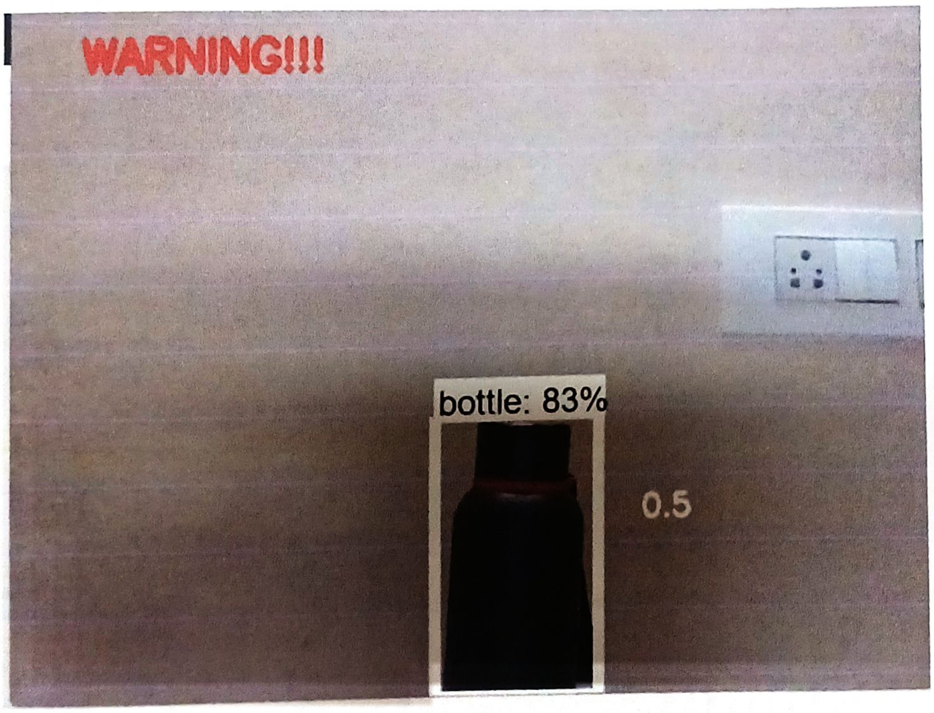

Fig 5. The snapshot of identified bottle from real-time as seen in the camera

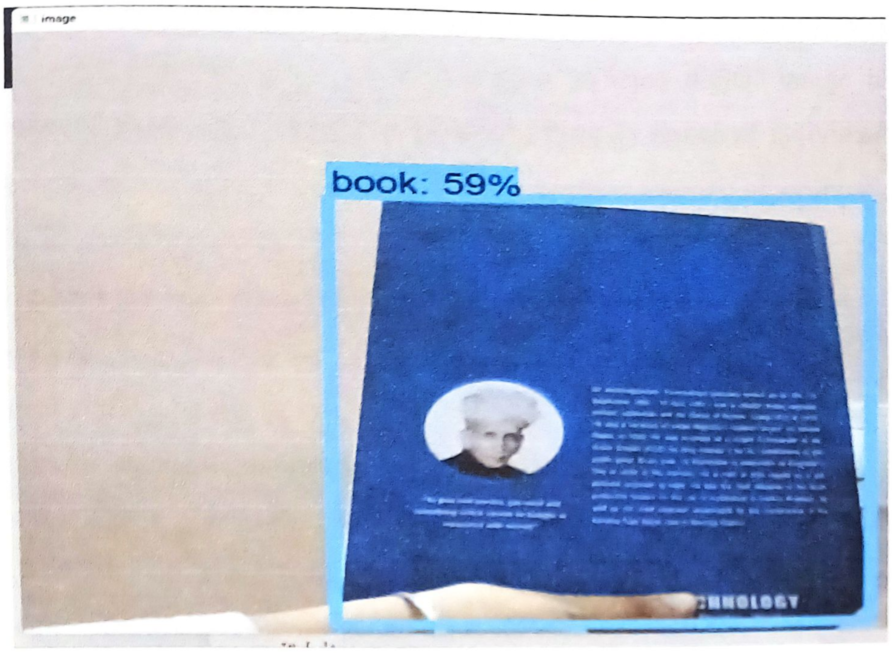

Fig 6 The snapshot of identified book from real-time as seen in the camera

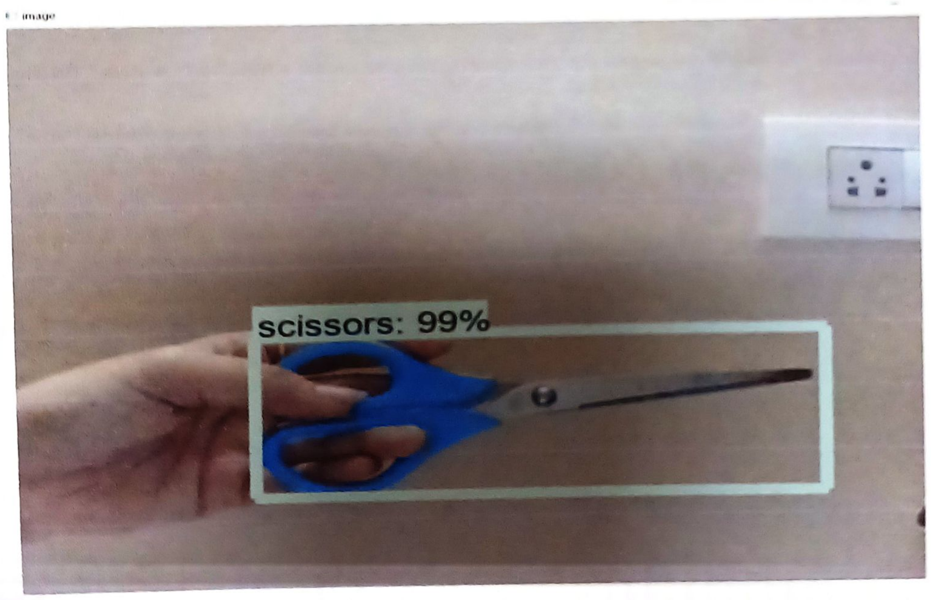

Fig 7. The snapshot of identified scissors from real-time as seen in the camera

CONCLUSION

In conclusion, the blind assistance project utilizing digital image processing holds significant potential to improve the quality of life for visually impaired individuals. By harnessing advanced technologies, such as computer vision and machine learning algorithms, the project aims to enhance the accessibility and independence of blind individuals in navigating their surroundings.

Through the utilization of cameras or other image-capturing devices, the project can capture and analyze visual information in real-time. This information can then be processed and transformed into meaningful auditory or tactile feedback, allowing blind individuals to perceive and understand their environment more effectively. This feedback can include object recognition, scene description, obstacle detection, text-to-speech conversion, and more.

The digital image processing techniques employed in the project provide accurate and reliable results, improving over time with the advancements in machine learning algorithms and the availability of large-scale datasets. These techniques enable the system to recognize objects, detect hazards, and provide contextual information to blind individuals, empowering them to make informed decisions and navigate their surroundings safely.

The digital image processing techniques employed in the project provide accurate and reliable results, improving over time with the advancements in machine learning algorithms and the availability of large-scale datasets. These techniques enable the system to recognize objects, detect hazards, and provide contextual information to blind individuals, empowering them to make informed decisions and navigate their surroundings safely.

FUTURE SCOPE AND ENHANCEMENT

1) Improved Object Detection Algorithms:

As computer vision technologies advance, more advanced object detection algorithms may become available. Future enhancements can involve integrating state-of-the-art algorithms that offer improved accuracy, speed, and robustness in object detection. This would further enhance the system`s ability to identify and classify objects in real time.

2) Incorporation of Deep Learning Models:

Deep learning models, such as convolutional neural networks (CNNs), have shown significant promise in various computer vision tasks. Integrating deep learning models specifically designed for blind assistance can enhance the system's ability to understand and interpret complex visual scenes, leading to more accurate object detection and scene understanding.

3) Integration with Smart Devices and Wearables:

The system can be further enhanced by integrating with smart devices and wearables, such as smart glasses or haptic feedback devices. This would provide a more seamless and immersive user experience, allowing blind individuals to receive feedback and navigate their surroundings hand-free.

REFERENCES

1. B. R. Hunt, "Digital image processing," in Proceedings of the IEEE, vol. 63, no. 4, pp. 693-708, April 1975, doi: 10.1109/PROC.1975.9801.
2. D. Dakopoulos and N. G. Bourbakis, "Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 40, no. 1, pp. 25-35, Jan. 2010, doi: 10.1109/TSMCC.2009.2021255
3. .D. P. Khairnar, R. B. Karad, A. Kapse, G. Kale and P. Jadhav, "PARTHA: A Visually Impaired Assistance System," 2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA), Mumbai, India, 2020, pp. 32-37, doi: 10.1109/CSCITA47329.2020.9137791.
4. S. Deshpande and R. Shriram, "Real time text detection and recognition on hand held objects to assist blind people," 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), Pune, India, 2016, pp. 1020-1024, doi: 10.1109/ICACDOT.2016.7877741.
5. B. Deepthi Jain, S. M. Thakur and K. V. Suresh, "Visual Assistance for Blind Using Image Processing," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2018, pp.
6. S. Durgadevi, K. Thirupurasundari, C. Komathi and S. M. Balaji, "Smart Machine Learning System for Blind Assistance," 2020 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2020, pp. 1-4, doi: 10.1109/ICPECTS49113.2020.9337031.W. C. S. S. Simões and V. F. de Lucena, "Blind user wearable audio assistance for indoor navigation based on visual markers and ultrasonic obstacle detection," 2016 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2016, pp. 60-63, doi:
7. Joe Louis Paul, S. Sasirekha, S. Mohanavalli, C. Jayashree, P. Moohana Priya and K. Monika, "Smart Eye for Visually Impaired-An aid to help the blind people," 2019 Computational Intelligence in Data Science (ICCIDS), Chennai, India, 2019, pp. 1-5, doi: 10.1109/ICCIDS.2019.8862066

8. W. C. S. S. Simões and V. F. de Lucena, "Blind user wearable audio assistance for indoor navigation based on visual markers and ultrasonic obstacle detection," 2016 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2016, pp. 60-63, doi: 10.1109/ICCE.2016.7430522.