

OBJECT DETECTION FOR BLIND PEOPLE WITH SPEECH AS OUTPUT

J Component - Report

SWE1010 – IMAGE AND VIEDO ANALYTICS

Fall Semester 2024 - 25

# PROFESSOR INCHARGE:

SARANYARAJ D

**BY:**

**G.S. SAIPRIYA -21MIA1155**

**G. SRINIVASA REDDY -21MIA1028**

**B. SANJAY -21MIA1073**



**CONTENTS**

* **ABSTRACT**
* **OBJECTIVE**
* **MOTIVATION**
* **METHODOLOGY**
* **DATA SET**
* **MODULE DESCRIPTION**
* **ALGORITHM**
* **TRAINING DATA**
* **SYSTEM BLOCK DIAGRAM**
* **ARCHITECTURE**
* **ADVANTAGES AND DISADVANTAGES**
* **HARDWARE AND SOFTWARE COMPONENTS**
* **EXPERIMENT RESULTS**



# ABSTRACT:

This project tries to transform the visual world into the audio world with the potential to inform blind people objects as well as their spatial locations. Objects detected from the scene are represented by their names and converted to speech. Their spatial locations are encoded into the 2-channel audio with the help of sound simulation**.**

# PROPOSED IDEA:

* Help the blind “see” better using Image-to-Text and Text-to-Voice, without any complex hardware.
* Features are implemented using the model You Only Look Once (YOLO) algorithm that runs through a variation of an extremely complex Convolutional Neural Network architecture.
* This project tries to transform the visual world into the audio world with the potential to inform blind people objects as well as their spatial locations. Objects detected from the scene are represented by their names and converted to speech. Their spatial locations are encoded into the 2-channel audio with the help of sound simulation.

# OBJECTIVE:

Here, our goal is to begin to use artificial neural networks, in the form of artificial neurons, to teach our program what object looks like. In this case, we will use objects, images. We train the data which is further used to classify and identify the image by speech.

# MOTIVATION:

Many people suffer from temporary or permanent disabilities.



# CURRENT SCENARIO



Here are some scenarios related to our project. As we evolved from one generation to generation our needs and facilities are also evolved with us. So we want to come up with new technology for blind people as technology is improving day by day.

Similarly our project helps in these evolutions by providing comfort to the disabled who are facing obstacles due to their disability.

# METHODOLOGY:

* YOLO (You Only Look Once) is a network for object detection that consists of determining the location on the image where certain objects are present.
* Classifies object along with class probabilities for real time applications.
* It uses single CNN network for both localization and classification of objects using bounding box.
* The YOLO design enables end-to-end training and real time speeds while maintaining high average precision.
* YOLO Algorithm for Object Detection: Explain how YOLO operates and its suitability for real-time applications due to its fast processing capabilities.



* Image Processing with OpenCV: Demonstrate the use of OpenCV for capturing and processing visual information.
  + Object Detection Framework: Describe the system setup, which includes:
  + Hardware: Raspberry Pi 3B+, camera module, and optional sensors to enhance spatial awareness.
  + Software: Leveraging YOLO for detection, OpenCV for image acquisition, and Google Text-to-Speech (GTTS) for audio output

# DATASET - COCO DATASET, YOLO (80 CLASSES):

****

COCO is a large-scale

object detection, segmentation, and

captioning dataset.

COCO has several features:

* + Object segmentation
  + Recognition in context
  + Super pixel stuff segmentation
  + 330K images (>200K labeled)
  + 1.5 million object instances
  + 80 object categories
  + 91 stuff categories
  + 5 captions per image
  + 250,000 people with key points

Our algorithm also has to decide whether to speak out a detected object and at what time. Obviously it’s undesirable to keep speaking out the same object to the user even if the detection result is correct. It’s also undesirable if two object names are spoken overlapping or very closely that the user won’t be able to distinguish.



# MODULE DESCRIPTION:

1. **IMAGE CAPTURE:**

The first step in the working of the Blind-Sight application is Image Capturing. Image Capturing is the process of obtaining images from a video which have to be converted into frames. Live stream is captured with a camera.

# FEATURE EXTRACTION:

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.

# OBJECT CLASSIFICATION:

Object Classification is a classification problem which tends to classify different objects which could flowers, faces, fruits or any object we could imagine. The object classification process includes the following:

* + Finding Relative Position
  + Class Identification with Confidence
  + Text Description

# SPEECH SYNTHESIS:

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. It is the process of generating spoken language by machine on the basis of written input. Speech synthesis involves the process of text analysis and letter to sound conversion.



# ALGORITHM – YOLO:

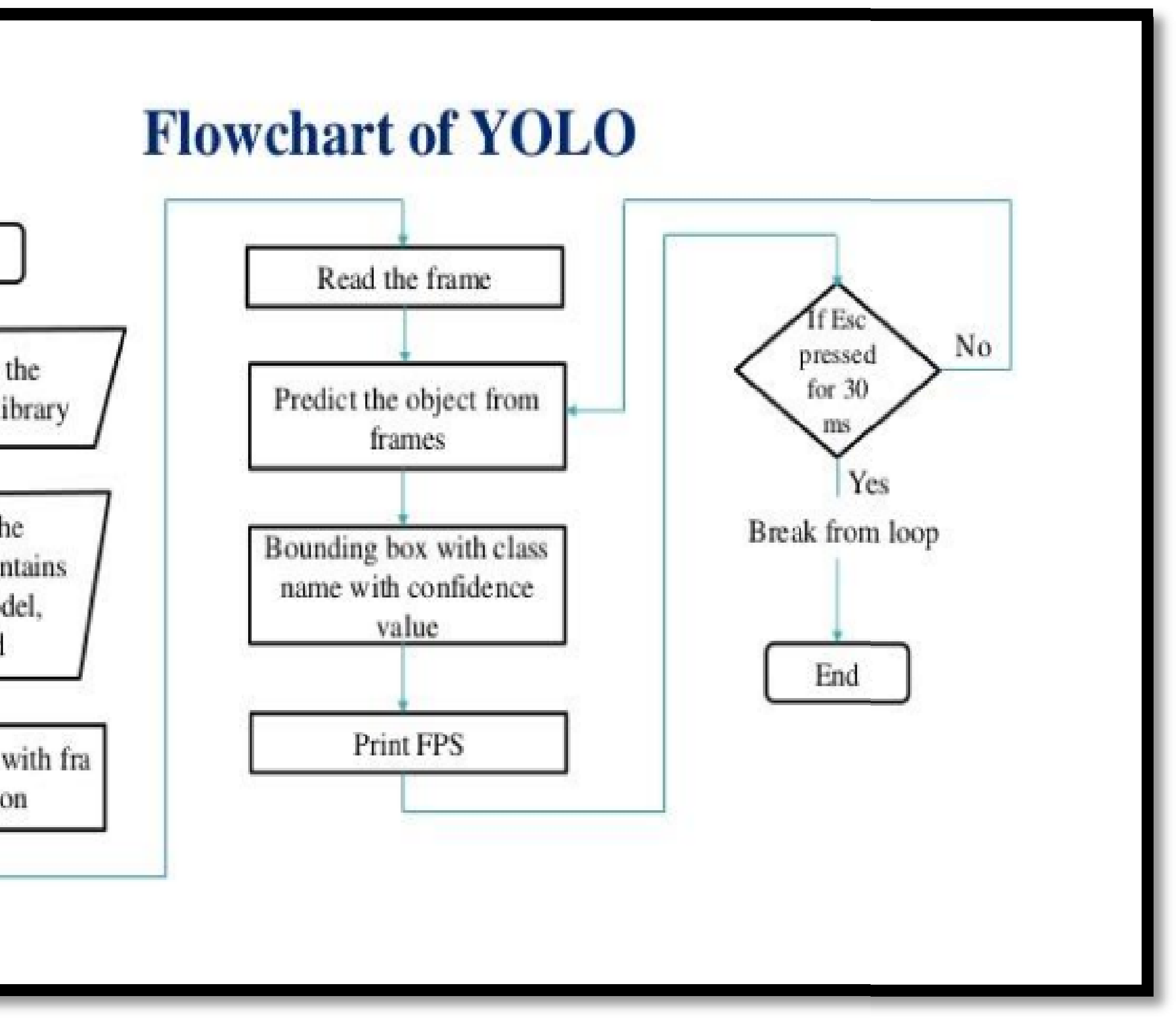
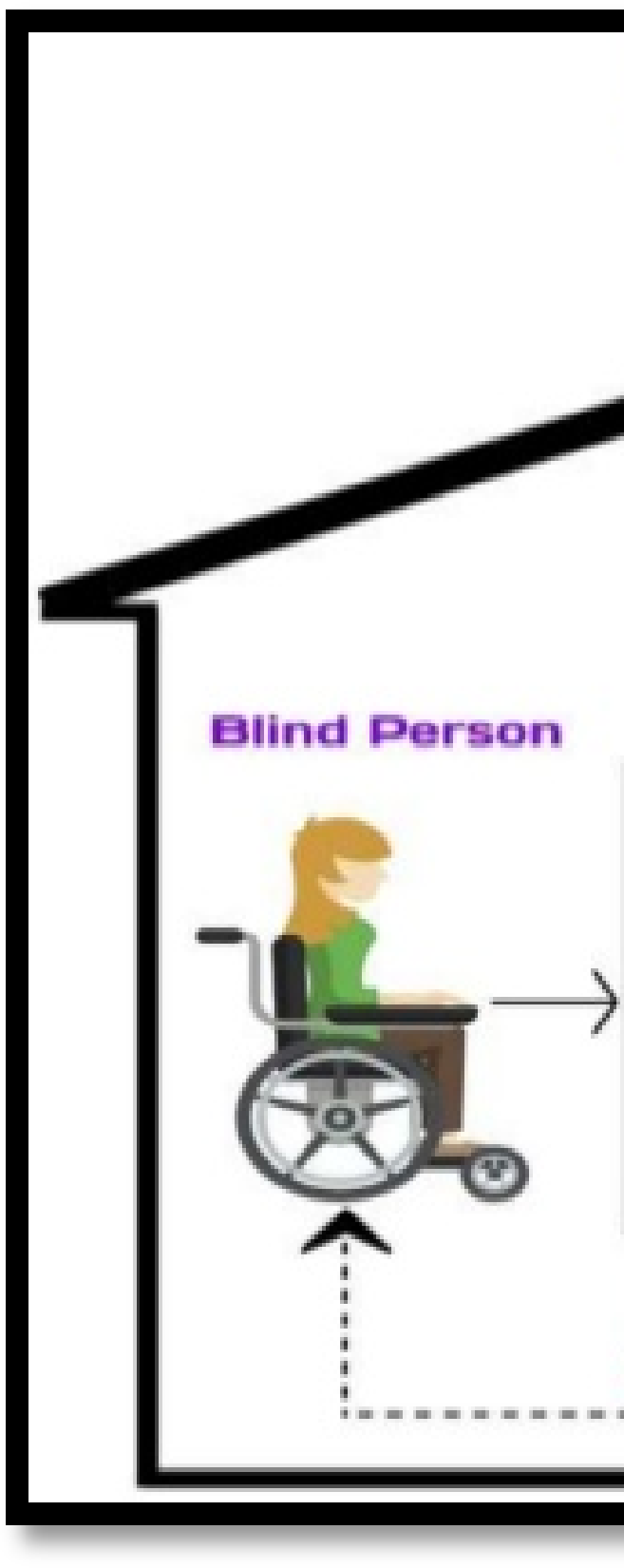
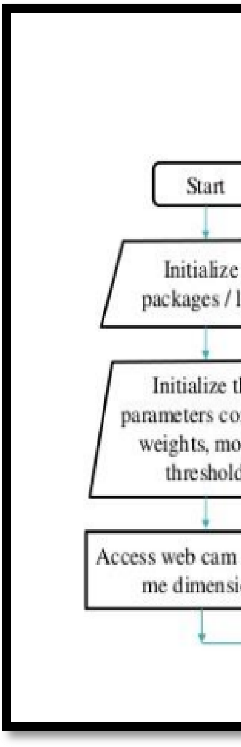
* (You Only Look Once) is a network for object detection that consists of determining the location on the image where certain objects are present.
* Classifies object along with class probabilities for real time applications.
* It uses single CNN network for both localization and classification of objects using bounding box.
* The YOLO design enables end-to-end training and real time speeds while maintaining high average precision.

# TRAINING DATA:

* **Training Data:** The model is trained with the Common Objects In Context (COCO) dataset.
* Object classes in coco.names file are indexed.
* C represents the class index of the object we are trying to label.
* The training has already been done on COCO.
* At a high level, the COCO format defines exactly how your annotations (bounding boxes, object classes, etc) and image metadata (like height, width, image sources, etc) are stored on disk.

# PREDECTION/DETECTION TIME:

If we are feeding 1280 x 720 frames from our camera into YOLO at Prediction time. YOLO will automatically resize it to 416 x 234 and fit it into a popular standard-sized 416 x 416 network by padding the excess with 0s. YOLO divides each image into S x S cells each with a size of 32 x 32 (reduction factor=32). This creates 416/32 = 13 x 13 cells.



**SYSTEM BLOCK DIAGRAM:**

**Input Data:** We will be using our webcam to feed images at 30 frames-per-

second to this trained model and we can set it to only process every other frame to speed things up.

**API:** The class prediction of the objects detected in every frame will be a string

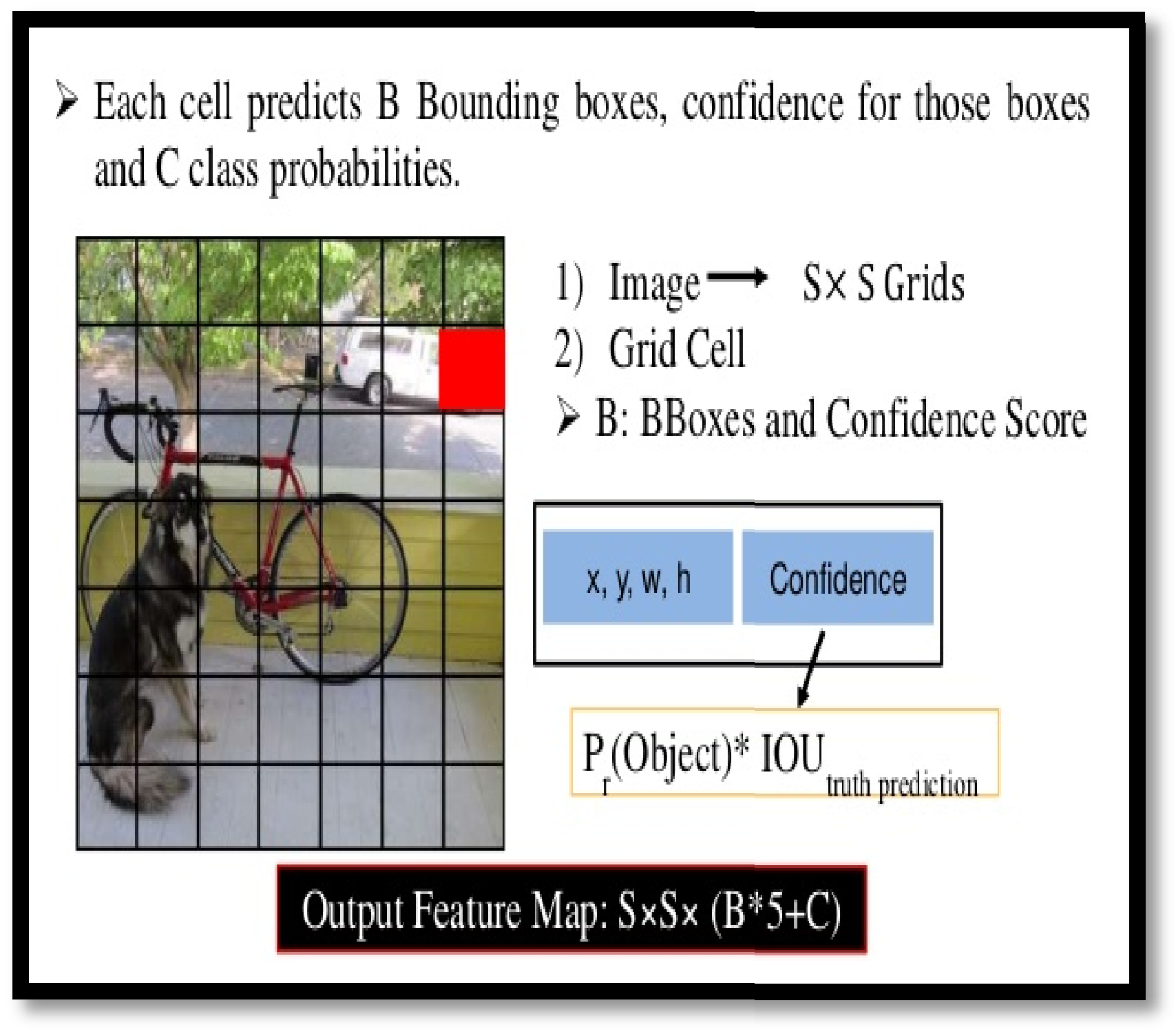
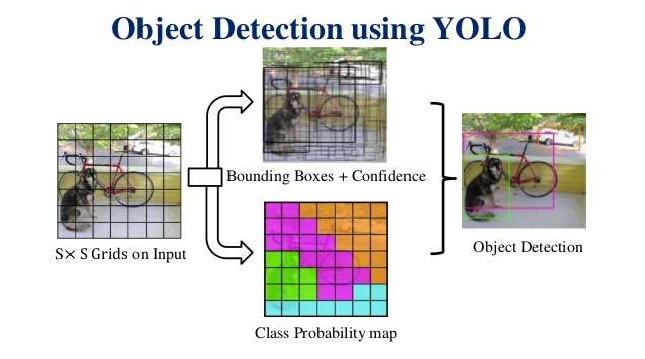
e.g. “teddy bear”. We will also obtain the coordinates of the objects in the image

and append the position

“top”/“mid”/“bottom” & “left”/“ce

ter”/“right” to the

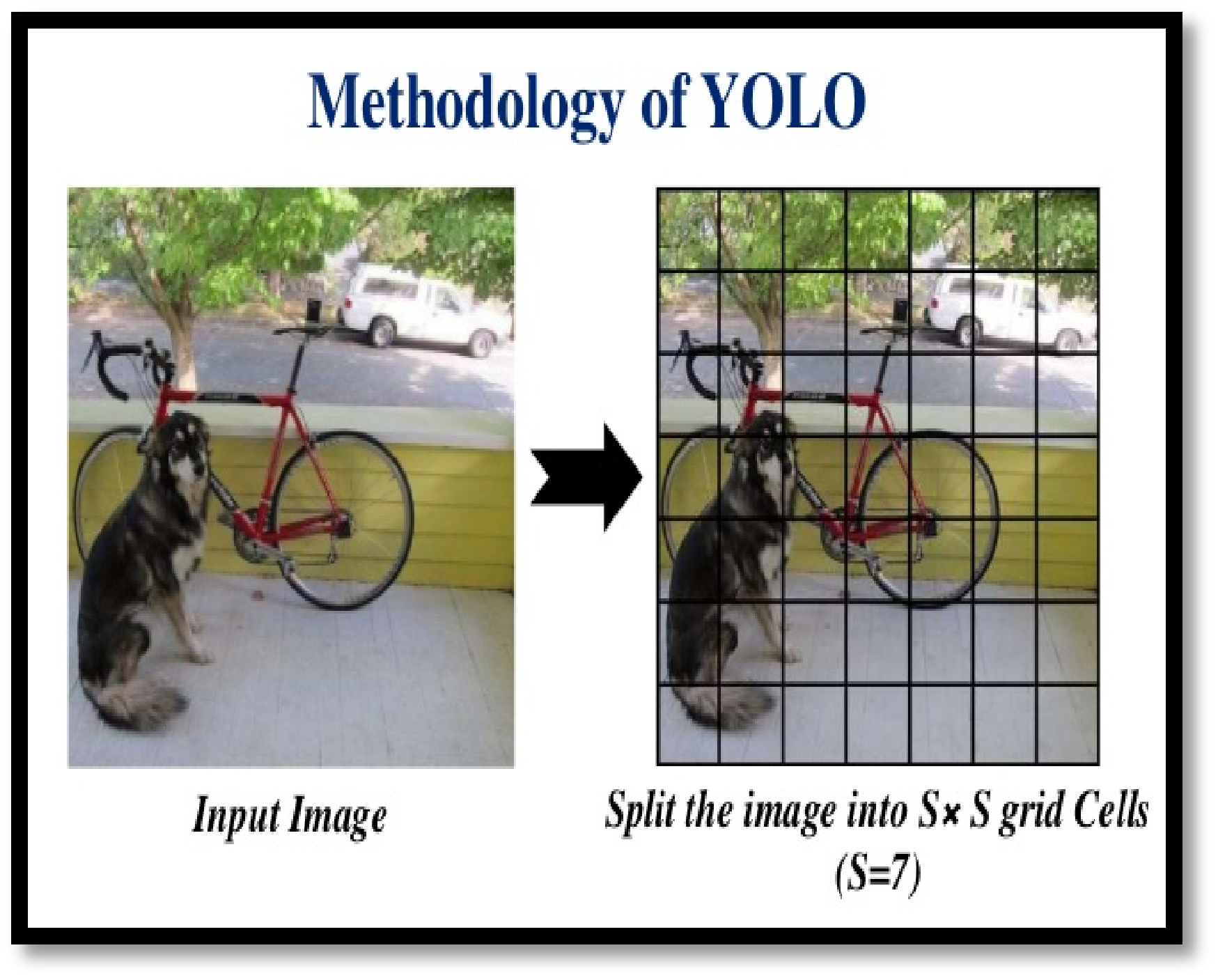
class prediction “cat”. We can then send the text description to the Google Text-to- Speech API using the gTTS package.



**Output:** We will also obtain the coordinates of the bounding box of every object detected in our frames, overlay the boxes on the objects detected and return the

stream of frames as a video playback. We will also schedule to get a voice

feedback on the 1st frame of each second (instead of 30 fps) e.g. “bottom left teddy bear” — meaning a cat was detected on the bottom-left of my camera view.





**Future Scope:**

Explore potential improvements, such as the integration of multiple languages, enhanced object detection abilities, or more compact hardware designs. Mention the possibility of employing advanced models or expanding the system's vocabulary for recognizing a broader range of objects.

# COCO DATASET:

Person Bottle Cell phone Pizza Donut Cake Chair

Potted plant Bed Toaster

Refrigerator Book

Clock Vase Scissors Teddy bear Hair drier Toothbrush Bicycle

TV monitor Laptop Mouse Remote Keyboard Microwave Oven



# CODE:

import numpy as np import time

import cv2 import os import imutils

import subprocess from gtts import gTTS

from pydub import AudioSegment

AudioSegment.converter = r"C:\Users\dell\OneDrive\Desktop\object-detection-with-voice- feedback\ffmpeg.exe"

AudioSegment.ffprobe = r"C:\Users\dell\OneDrive\Desktop\object-detection-with-voice- feedback\ffprobe.exe"

# load the COCO class labels our YOLO model was trained on LABELS = open("coco.names").read().strip().split("\n")

with open("coco.names", "r") as f:

classes = [line.strip() for line in f.readlines()]

colors = np.random.uniform(0, 255, size=(len(classes), 3))

# load our YOLO object detector trained on COCO dataset (80 classes) print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet("yolov3.cfg", "yolov3.weights") font = cv2.FONT\_HERSHEY\_PLAIN

# determine only the \*output\* layer names that we need from YOLO ln = net.getLayerNames()

ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# initialize

cap = cv2.VideoCapture(0)

frame\_count = 0 start = time.time() first = True frames = [] flag=1

while True:

frame\_count += 1

# Capture frame-by-frame ret, frame = cap.read() cv2.imshow("aj", frame) frames.append(frame)

if cv2.waitKey(25) & 0xFF == ord('q'): break

if ret:

key = cv2.waitKey(1)

if frame\_count % 60 == 0: end = time.time()

# grab the frame dimensions and convert it to a blob (H, W) = frame.shape[:2]

# construct a blob from the input image and then perform a forward

# pass of the YOLO object detector, giving us our bounding boxes and # associated probabilities

blob = cv2.dnn.blobFromImage(frame, 1/ 255.0, (416, 416), swapRB=True, crop=False)

net.setInput(blob) layerOutputs = net.forward(ln)

# initialize our lists of detected bounding boxes, confidences, and # class IDs, respectively

boxes = [] confidences = [] classIDs = [] centers = []

# loop over each of the layer outputs for output in layerOutputs:

# loop over each of the detections for detection in output:

# extract the class ID and confidence (i.e., probability) of # the current object detection

scores = detection[5:] classID = np.argmax(scores) confidence = scores[classID]



to the actually bounding

box.astype("int")

# filter out weak predictions by ensuring the detected # probability is greater than the minimum probability if confidence > 0.5:

# scale the bounding box coordinates back relative # size of the image, keeping in mind that YOLO

# returns the center (x, y)-coordinates of the

# box followed by the boxes' width and height box = detection[0:4] \* np.array([W, H, W, H]) (centerX, centerY, width, height) =

and

confidences,

# use the center (x, y)-coordinates to derive the top

# and left corner of the bounding box x = int(centerX - (width / 2))

y = int(centerY - (height / 2))

# update our list of bounding box coordinates, # and class IDs

boxes.append([x, y, int(width), int(height)]) confidences.append(float(confidence)) classIDs.append(classID) centers.append((centerX, centerY))

# apply non-maxima suppression to suppress weak, overlapping bounding # boxes

idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3)

for i in range(len(boxes)): if i in idxs:

x, y, w, h = boxes[i]

label = str(classes[classIDs[i]]) confidence = confidences[i] color = colors[classIDs[i]]

cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2) cv2.putText(frame, label + " " + str(round(confidence, 2)),

(x, y + 30), font, 3, color, 3)



texts = ["The environment has following objects"] # ensure at least one detection exists

if len(idxs) > 0:

# loop over the indexes we are keeping for i in idxs.flatten():

# find positions

centerX, centerY = centers[i][0], centers[i][1]

if centerX <= W/3:

W\_pos = "left " elif centerX <= (W/3 \* 2):

W\_pos = "center "

else:

W\_pos = "right "



if centerY <= H/3:

H\_pos = "top " elif centerY <= (H/3 \* 2):

H\_pos = "mid "

else:

H\_pos = "bottom "

texts.append(H\_pos + W\_pos + LABELS[classIDs[i]]) flag=0

print(texts)

if (flag==0):

description = ', '.join(texts)

tts = gTTS(description, lang='en') tts.save(r"C:\Users\dell\OneDrive\Desktop\object-detection-with-

voice-feedback\tts.mp3")

tts =

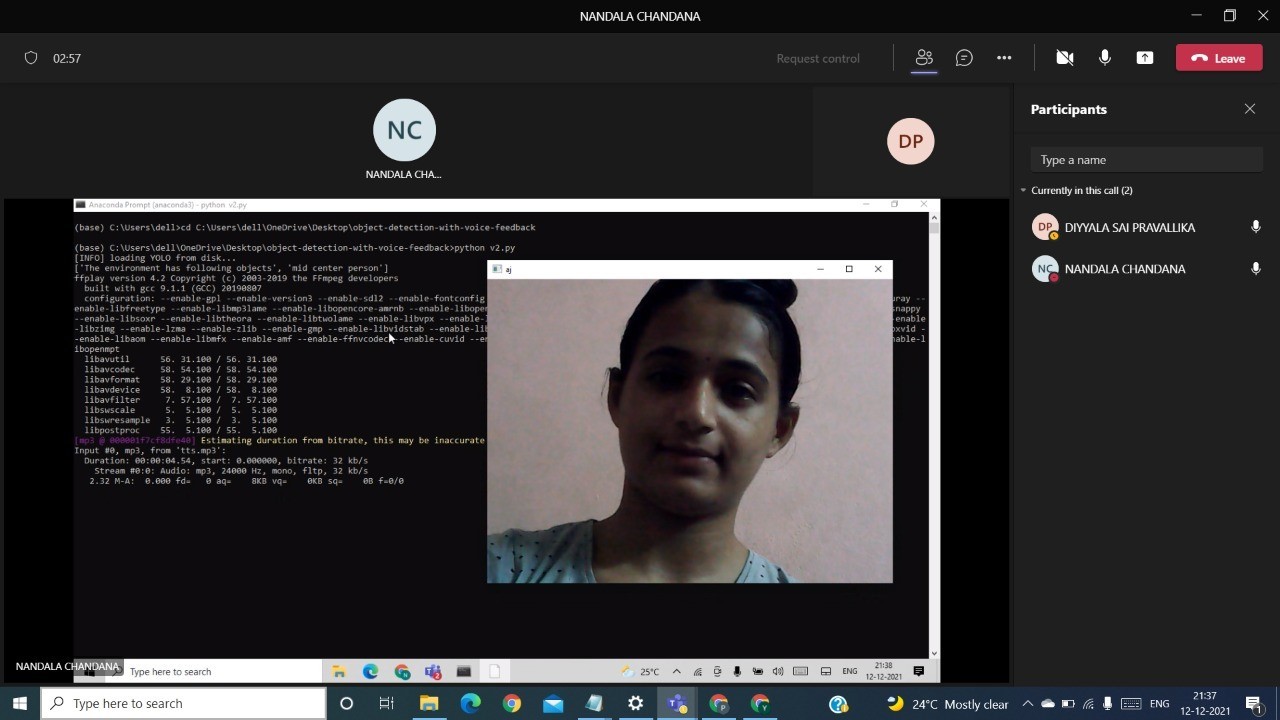
AudioSegment.from\_mp3(r"C:\Users\dell\OneDrive\Desktop\object-detection-with-voice- feedback\tts.mp3")

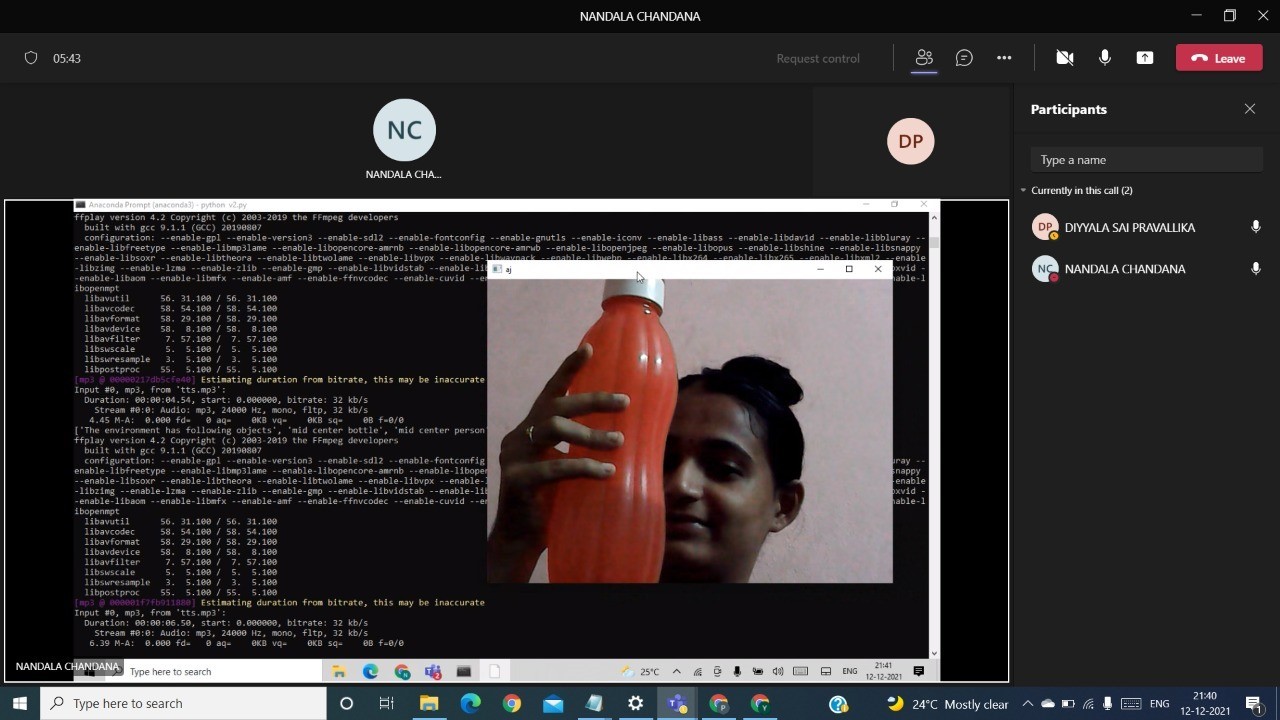
subprocess.call(["ffplay", "-nodisp", "-autoexit", "tts.mp3"])

cap.release() cv2.destroyAllWindows() os.remove("tts.mp3")

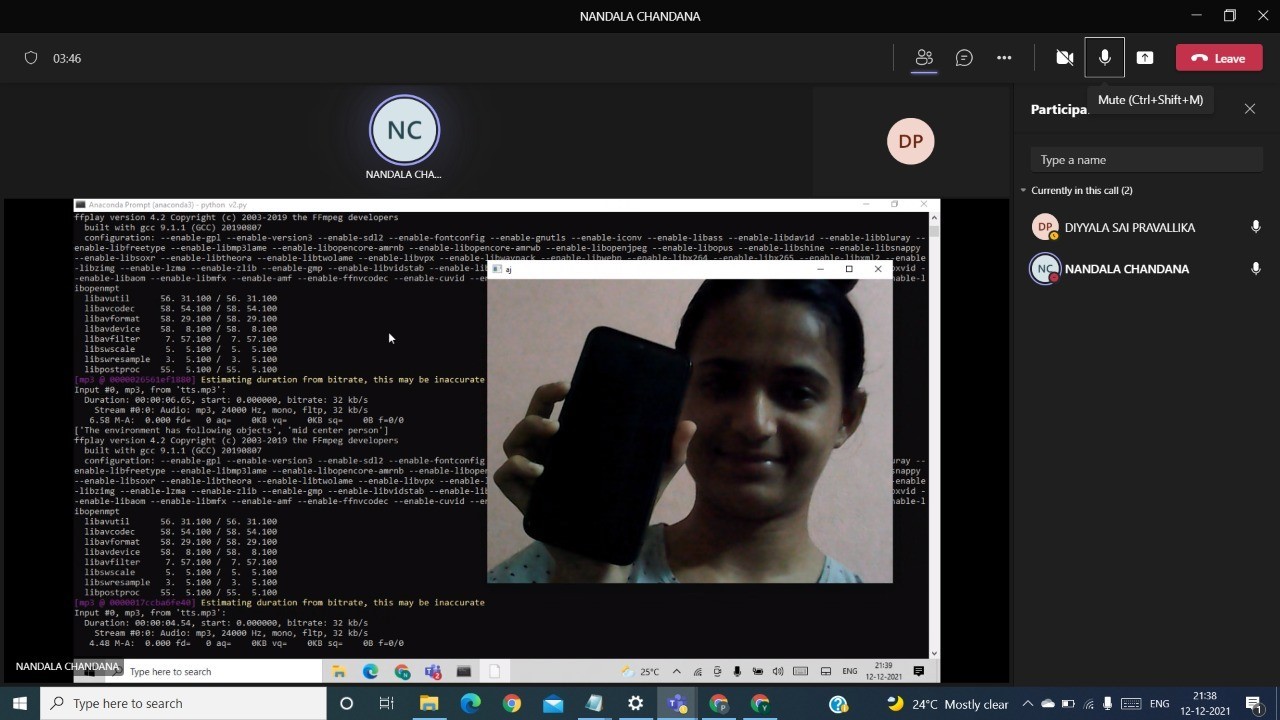


# OUTPUTS:











Discussion

* **Advantages:** Discuss the benefits of using YOLO and voice feedback for real-time object detection, such as enhanced safety for visually impaired users and the ability to navigate independently.
* Limitations and Drawbacks:
* **Limited Detection Range and Accuracy:** The YOLO algorithm performs well in controlled lighting and close-range scenarios but may struggle in low-light conditions or when objects are far from the camera, which limits the system's effectiveness in outdoor or complex environments.
* **Processing Power Constraints:** Since the system is built on a Raspberry Pi, a lightweight processor, it may experience delays in detection, particularly if multiple objects appear simultaneously or if higher-resolution detection is required. This can affect real-time feedback.
* **False Positives and Detection Errors:** The model can sometimes produce false positives, identifying objects incorrectly, which can lead to confusion for the visually impaired user. Additionally, detecting small or overlapping objects can be challenging.
* **Audio Overload in Complex Environments:** In busy settings with multiple objects, the system may provide numerous audio notifications simultaneously, overwhelming the user. Currently, it lacks filtering or prioritization for essential objects (e.g., obstacles vs. non-essential items).
* **Dependence on Stable Internet for GTTS**: The Google Text-to-Speech (GTTS) feature, while effective, requires an internet connection, which can limit the system's usability in areas with poor or no connectivity. Offline text-to-speech alternatives may not match the quality and language support of GTTS.
* **Limited Object Recognition Capabilities:** The system's training dataset (e.g., COCO) only includes a limited number of common objects. Therefore, it may not recognize specific or uncommon objects, which could reduce utility in unfamiliar environments.
* **Future Enhancements:** To address these drawbacks, future improvements could include integrating a more powerful processing unit, optimizing the YOLO model for resource-constrained devices, employing offline text-to-speech alternatives, adding filtering mechanisms to reduce audio overload, and training the model on larger, more diverse datasets.



**Conclusion:**

* Empowering visually impaired individuals:
  + Enhances independence and safety by helping users navigate their environment.
* Practical solution:
  + Integrates YOLO algorithm for object detection with real-time voice feedback to identify obstacles and nearby objects.
* Affordable and scalable:
  + - Uses Raspberry Pi and open-source software, making the system accessible and scalable for wider adoption.
* Current limitations:
  + Issues with detection range, processing power, and reliance on internet connectivity for voice output.
* Future development possibilities:
  + Incorporate offline text-to-speech solutions.
  + Optimize processing capabilities for improved efficiency.
  + Expand object recognition range.
  + Implement adaptive features like prioritizing critical objects in audio feedback.