Our project’s code

import cv2 import numpy as np from gtts import gTTS import os import time

## Load YOLO

net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg") layer_names = net.getLayerNames() unconnected_out_layers = net.getUnconnectedOutLayers() output_layers = [layer_names[i - 1] for i in unconnected_out_layers]

## Load COCO names

with open("coco.names", "r") as f: classes = [line.strip() for line in f.readlines()]

## Initialize camera

cap = cv2.VideoCapture(0)

confidence_threshold = 0.3 distance_threshold = 1.0 frame_counter = 0 detection_interval = 30 # Adjust this value as needed

def calculate_distance(bbox_width, frame_width): known_width = 0.5 focal_length = (frame_width * distance_threshold) / known_width distance = (known_width * focal_length) / bbox_width return distance

def play_audio(message): tts = gTTS(text=message, lang='en') audio_file = "Object.mp3" tts.save(audio_file) os.system(f"start {audio_file}") while True: ret, frame = cap.read() if not ret: print("Failed to grab frame") break height, width, channels = frame.shape center_region_start = int(width / 3) center_region_end = int(2 * width / 3)

```
frame_counter += 1
if frame_counter % detection_interval == 0:
  blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
  net.setInput(blob)
  outs = net.forward(output_layers)
  detected_object = None

  for out in outs:
    for detection in out:
      scores = detection[5:]
      if len(scores) == 0:
        continue
      class_id = np.argmax(scores)
      confidence = scores[class_id]
      if confidence > confidence_threshold:
        detected_class = classes[class_id]
        print(f"Detected: {detected_class}, Confidence: {confidence}")

        center_x = int(detection[0] * width)
        center_y = int(detection[1] * height)
        w = int(detection[2] * width)
        h = int(detection[3] * height)

        x = int(center_x - w / 3)
        y = int(center_y - h / 3)

        if center_region_start <= center_x <= center_region_end:
          distance = calculate_distance(w, width)
          if distance <= distance_threshold:
            detected_object = detected_class
            # Draw bounding box and label for detected object
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, f"{detected_object} {round(confidence, 2)}",
                  (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            break
    if detected_object:
      break

  # If an object is detected within the distance threshold, output a message and convert to
speech
  if detected_object:
    message = f"{detected_object} detected"
   # print(message)
```

```
        play_audio(message)
        time.sleep(5)  # Wait for 5 seconds (adjust as needed)
   # time.sleep(5)

# Display the resulting frame
cv2.imshow('Object Detection', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break


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