In [1]:

```
import cv2
import pyttsx3
import numpy as np
```

In [2]:

```
# Initialize text-to-speech engine
engine = pyttsx3.init()
```

In [3]:

```
# Load pre-trained object detection model
net = cv2.dnn.readNetFromDarknet('yolov4-tiny.cfg', 'yolov4-tiny.weights')
```

In [4]:

```
classes = []
with open('classes.txt', 'r') as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
```

In [5]:

```
print(classes)
```

```
['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'b
oat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bir
d', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe',
'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboar
d', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfbo
ard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'b
owl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'p
izza', 'donut', 'cake', 'chair', 'sofa', 'pottedplant', 'bed', 'diningtable', 'toi
let', 'tvmonitor', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwa
ve', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissor
s', 'teddy bear', 'hair drier', 'toothbrush']
```

In [6]:

```
print(len(classes))
```

```
80
```

In [8]:

```
# Initialize video capture device
cap = cv2.VideoCapture(0)

# Constants for distance estimation
KNOWN_WIDTH = 0.2  # Width of the object we know the distance to (in meters)
FOCAL_LENGTH = 500  # Focal length of the camera (in pixels)

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Detect objects in the frame
    height, width, channels = frame.shape
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Draw bounding boxes around detected objects and estimate distances
    class_ids = []
    confidences = []
    boxes = []
    distances = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                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 / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

                # Estimate distance to the object
                distance = (KNOWN_WIDTH * FOCAL_LENGTH) / w
                distances.append(distance)
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    font = cv2.FONT_HERSHEY_PLAIN
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = f"{classes[class_ids[i]]}: {distances[i]:.2f} meters"
            color = colors[class_ids[i]]
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            cv2.putText(frame, label, (x, y + 30), font, 2, color, 2)

            # Speak out detected object label and distance
            engine.say(label)
            engine.runAndWait()

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

    # Press 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord('e'):
        break

# Release video capture device and destroy windows
cap.release()
cv2.destroyAllWindows()
```

In [ ]: