Mobile Application for Object Detection and Audio Feedback to Aid Visually Impaired Navigation

Beulah T - 23/PCSA/104

Guide - Ms. Blessy Boaz

Base paper

Title

Real-Time Object Detection and Audio Feedback for the Visually Impaired (IEEE)

Project Implementation

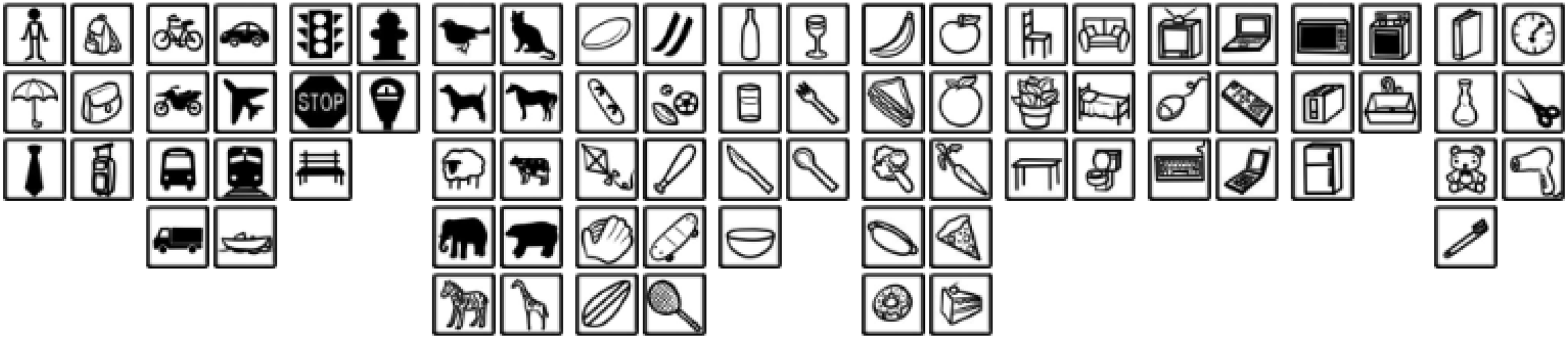
Title

Mobile Application for Object Detection and Audio Feedback to Aid Visually Impaired Navigation

Dataset used

- Microsoft COCO (Common Objects in Context) dataset.
- The dataset contains over 330,000 images and more than 2.5 million labeled object instances across 80 categories.

COCO Dataset



Aspect	Base paper	Project Implementation
YOLO Version	YOLOv3	YOLOv8n
Speech Feedback	gTTS (requires internet)	pyttsx3 (offline)
Grid Size	SxS	Automatically handled by YOLOv8
Bounding Box Count	B bounding boxes per grid cell	Flexible and automatic in YOLOv8
Audio Generation	Prepares an audio file (MP3)	Direct speech synthesis
Performance	Real-time with GPU	Faster and more accurate with YOLOv8n

Implementation

[1]:
```python
import cv2
import pyttsx3
import time
import numpy as np
import matplotlib.pyplot as plt
from ultralytics import YOLO
import random
```

[3]:
```python
# Initialize YOLO model for version 8
model_yolo = YOLO('yolov8n.pt')

# Initialize pyttsx3 for speech feedback
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Speech speed
```

[5]:
```python
# Function to generate random color
def generate_random_color():
    return tuple(np.random.randint(0, 255, 3).tolist())  # Random RGB color
```

[7]:

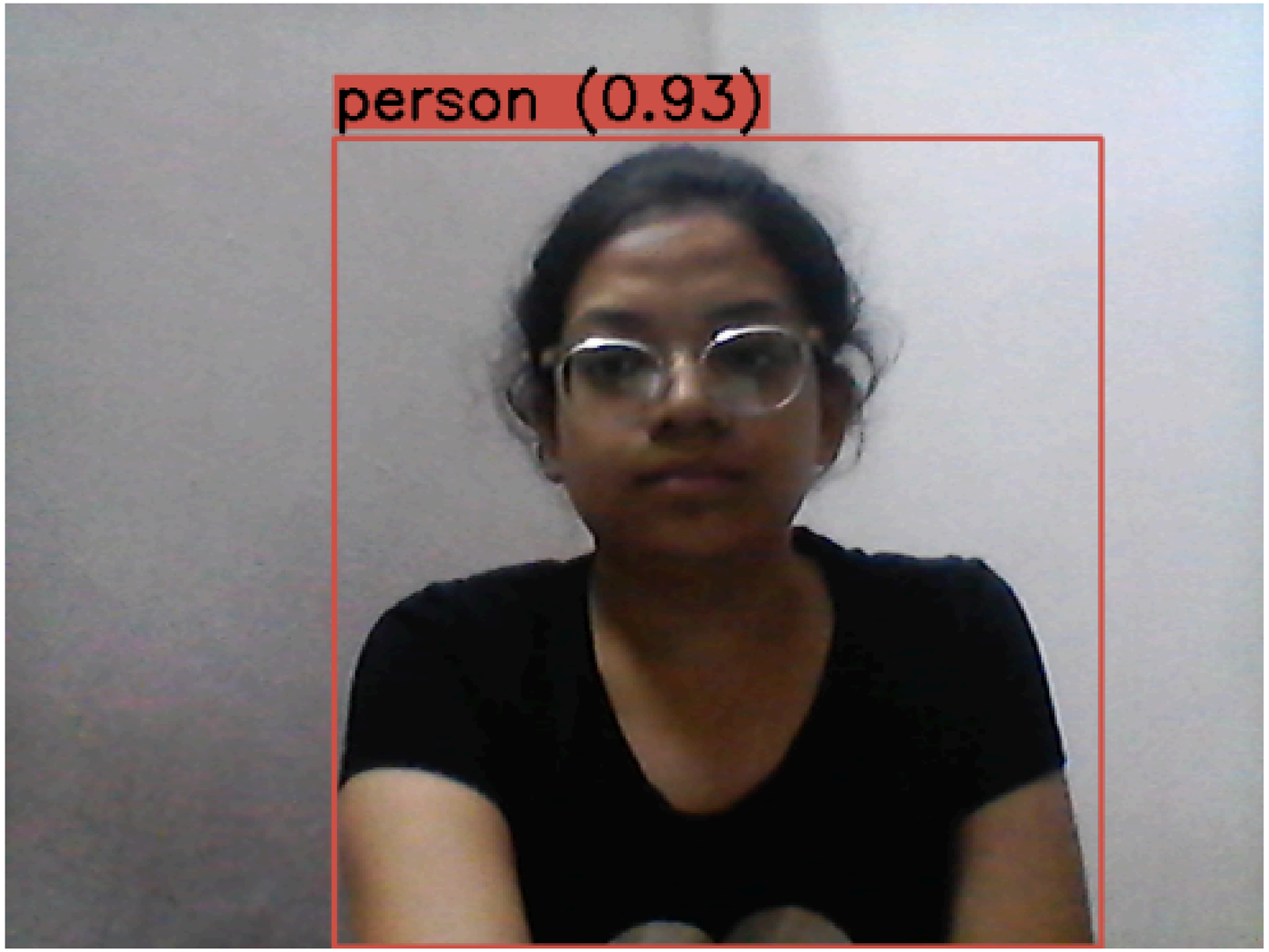
```python
# Function to detect objects and display results using YOLOv9
def detect_objects_and_display():
    cap = cv2.VideoCapture(0)  # Start webcam capture
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Failed to grab frame.")
            break

        results = model_yolo(frame)
        detections = results[0].boxes.data.cpu().numpy()
        labels = [model_yolo.names[int(label)] for label in detections[:, 5].astype(int)]
        confs = detections[:, 4]
        boxes = detections[:, :4]

        relative_positions = []  # List to store detected objects' relative positions
        colors = []  # List to store colors for bounding boxes

        for label, (xmin, ymin, xmax, ymax), confidence in zip(labels, boxes, confs):
            color = generate_random_color()  # Generate a random color for each object
            colors.append(color)  # Store the color for the current bounding box
            cv2.rectangle(frame, (int(xmin), int(ymin)), (int(xmax), int(ymax)), color, 2)

            # Prepare text and background for labels
            text = f"{label} ({confidence:.2f})"
            text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 1, 2)[0]
            text_x = int(xmin)
            text_y = int(ymin) - 10
            cv2.rectangle(frame, (text_x, text_y - text_size[1]), (text_x + text_size[0], text_y + 5), color, -1)
```

```
            # Calculate the relative position of the object
            frame_center_x = frame.shape[1] // 2
            if (xmin + xmax) // 2 < frame_center_x - 0.1 * frame.shape[1]:
                position = "left"
            elif (xmin + xmax) // 2 > frame_center_x + 0.1 * frame.shape[1]:
                position = "right"
            else:
                position = "center"

            relative_positions.append(f"{label} at {position}")

        # Provide audio feedback for detected objects and their positions
        if relative_positions:
            positions_str = ", ".join(relative_positions)
            engine.say("Detected: " + positions_str)
            engine.runAndWait()

        # Convert frame to RGB for matplotlib
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Display the frame using matplotlib
        plt.imshow(frame_rgb)
        plt.axis('off')  # Hide axes
        plt.show()

        time.sleep(0.1)  # Short delay before next frame

cap.release()
```

```
# Call the function to start detection and display
print("Using YOLOv8 for object detection.")
detect_objects_and_display()
```

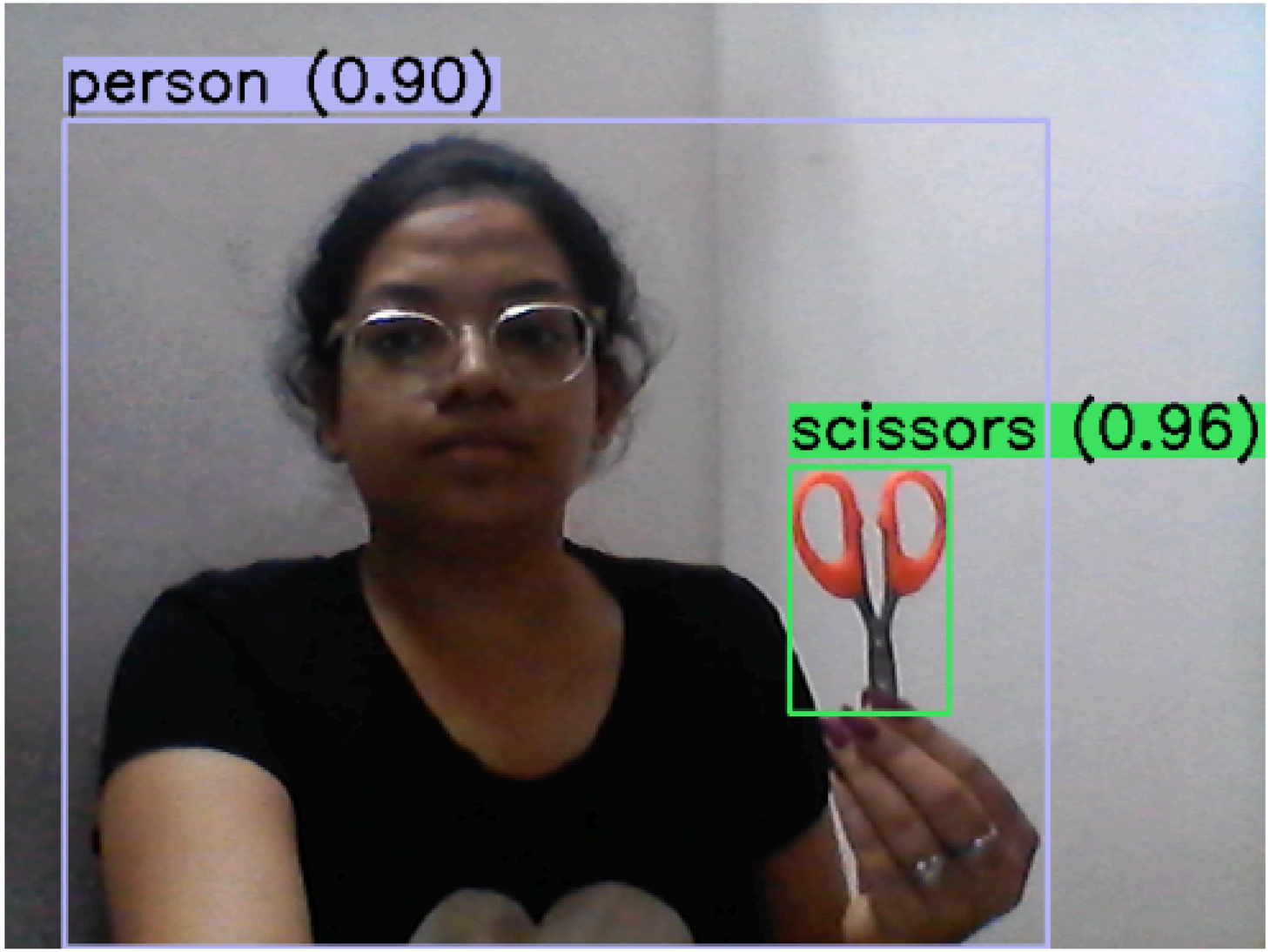
Output

```
0: 480x640 1 person, 703.2ms
Speed: 10.0ms preprocess, 703.2ms inference, 4.0ms postprocess per image at shape (1, 3, 480, 640)
```



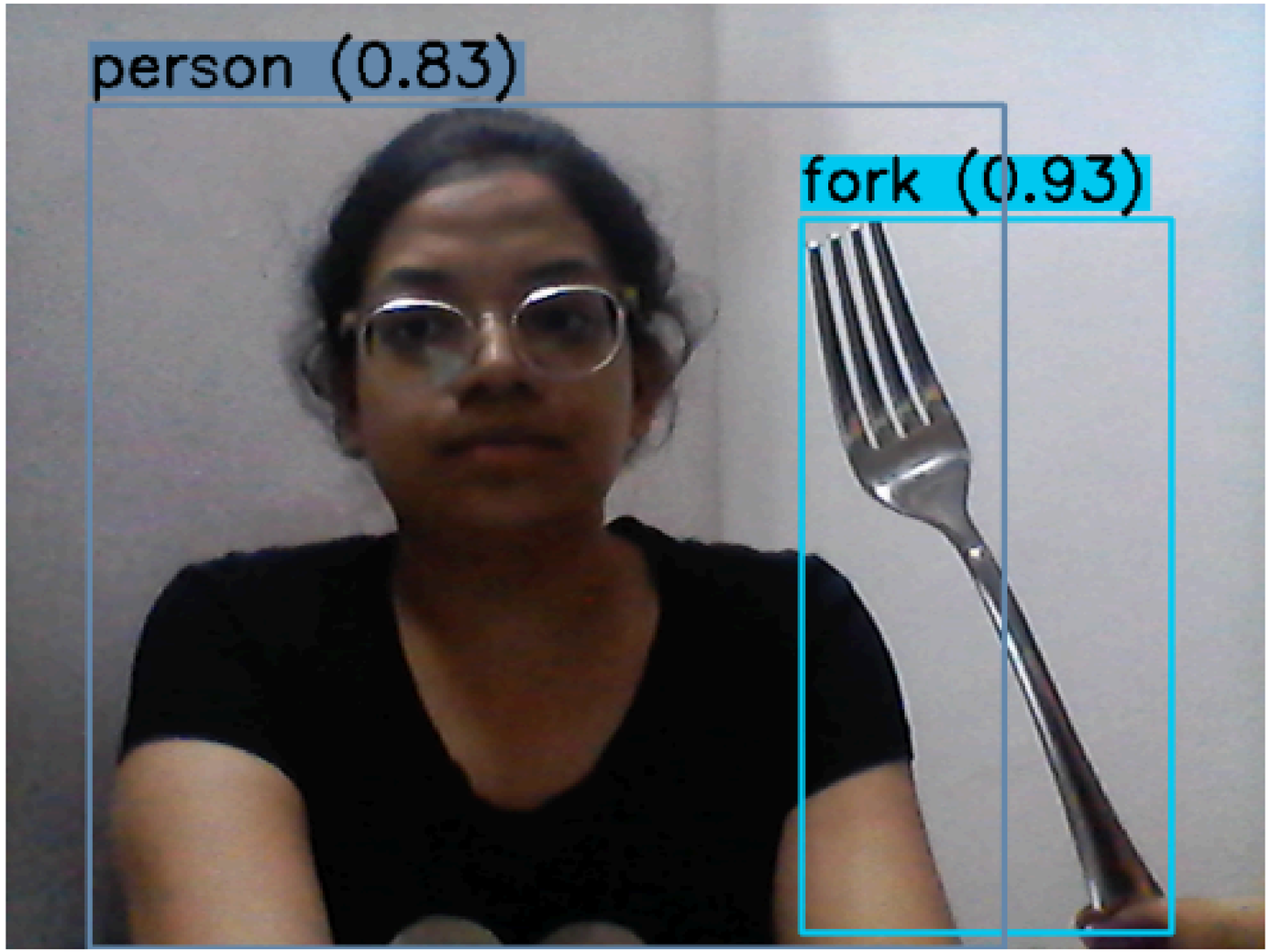
Output

```
0: 480x640 1 person, 1 scissors, 593.2ms
Speed: 13.0ms preprocess, 593.2ms inference, 4.0ms postprocess per image at shape (1, 3, 480, 640)
```



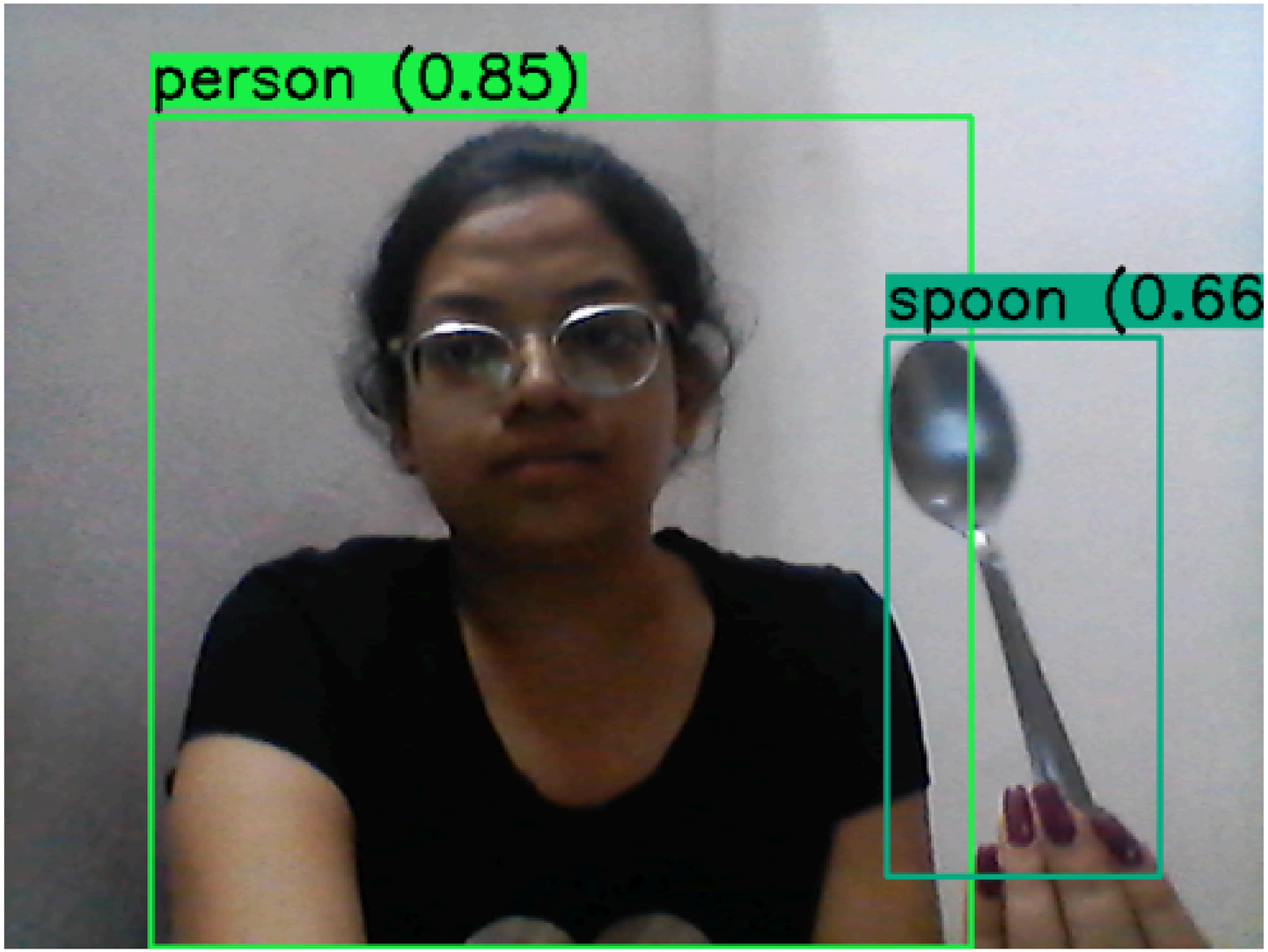
Output

```
0: 480x640 1 person, 1 fork, 535.1ms
Speed: 11.0ms preprocess, 535.1ms inference, 4.0ms postprocess per image at shape (1, 3, 480, 640)
```



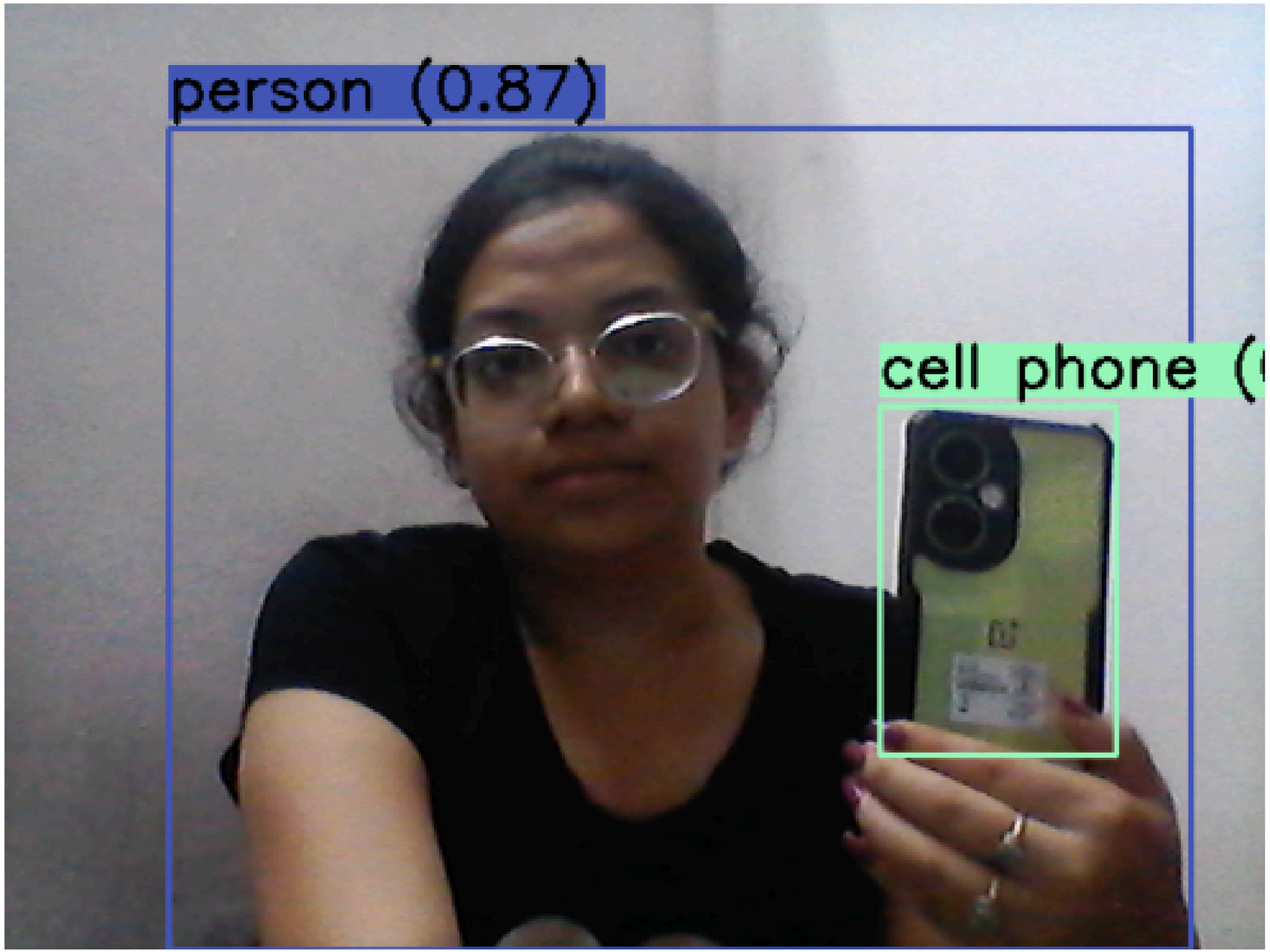
Output

```
0: 480x640 1 person, 1 spoon, 953.2ms
Speed: 8.0ms preprocess, 953.2ms inference, 4.0ms postprocess per image at shape (1, 3, 480, 640)
```



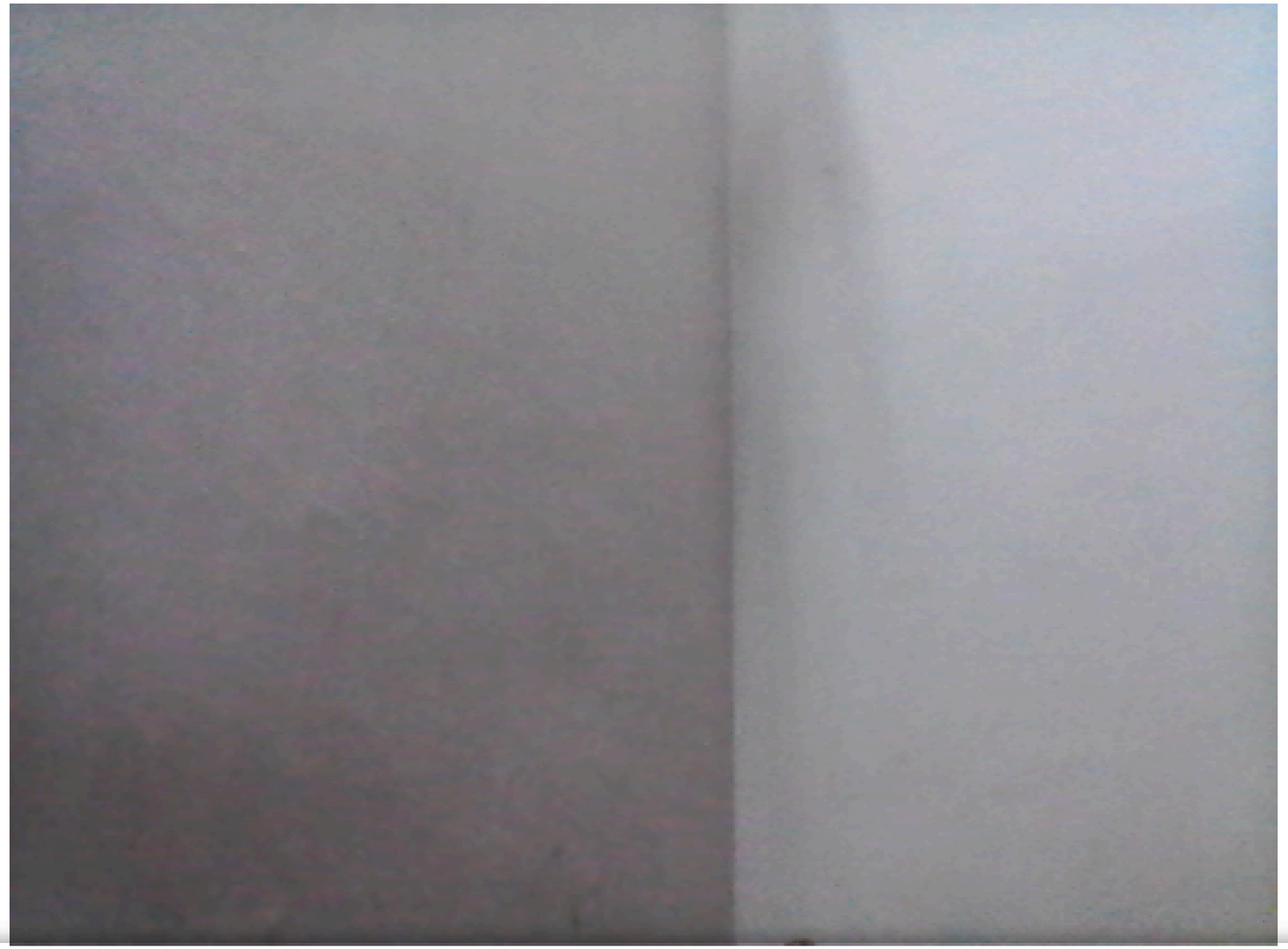
Output

```
0: 480x640 1 person, 1 cell phone, 654.2ms
Speed: 9.0ms preprocess, 654.2ms inference, 5.0ms postprocess per image at shape (1, 3, 480, 640)
```



Output

```
0: 480x640 (no detections), 530.1ms
Speed: 10.0ms preprocess, 530.1ms inference, 2.0ms postprocess per image at shape (1, 3, 480, 640)
```



Thank You