# Object and Text Detection System Report

May 6, 2025

## 1 Introduction

This report details a real-time object and text detection system implemented using Python, leveraging computer vision and machine learning techniques. The system utilizes a webcam feed to detect objects with the YOLOv8 model and extract text using Tesseract OCR, providing audio feedback via text-to-speech. The system is designed to be cross-platform compatible and optimized for performance with lower resolution and frame-skipping techniques.

## 2 System Overview

The system integrates several key components to achieve real-time detection and feedback:

- **Webcam Input**: Captures video feed at a resolution of 320x240 pixels for efficient processing.
- **YOLOv8 Model**: Detects objects from the COCO dataset, identifying up to 80 classes such as "person," "car," and "book."
- **Tesseract OCR**: Extracts text from grayscale video frames after applying Otsu's thresholding.
- **Text-to-Speech**: Uses pyttsx3 to provide audio feedback on detected objects and text.
- **Debouncing Mechanism**: Reduces redundant audio output by speaking only when detections change.
- **FPS Monitoring**: Tracks system performance by calculating frames per second.

## 3 Implementation Details

### 3.1 Dependencies

The system relies on the following Python libraries:

- `opencv-python` (cv2): For webcam access and image processing.

- `pytesseract`: For optical character recognition.
- `ultralytics`: For YOLOv8 object detection.
- `cvzone`: For drawing bounding boxes and text on frames.
- `pyttsx3`: For text-to-speech functionality.

### 3.2 Cross-Platform Compatibility

The system dynamically sets the Tesseract executable path based on the operating system:

- Windows: `C:\Program Files\Tesseract-OCR\tesseract.exe`
- Other (e.g., Linux, macOS): Assumes `tesseract` is in the system PATH.

### 3.3 Processing Pipeline

The system processes webcam frames as follows:

1. **Frame Acquisition**: Captures and resizes frames to 320x240 pixels.
2. **Frame Skipping**: Processes every third frame to improve performance.
3. **Object Detection**: Applies YOLOv8 to detect objects with a confidence threshold of 0.5, drawing bounding boxes and labels.
4. **Text Detection**: Converts frames to grayscale, applies Otsu's thresholding, and extracts text using Tesseract.
5. **Output Generation**: Combines object and text detections, speaks changes via text-to-speech, and displays the annotated frame.

## 4 Performance Optimizations

To ensure real-time performance, the system incorporates:

- **Low Resolution**: Uses 320x240 pixels to reduce computational load.
- **Frame Skipping**: Processes one out of every three frames.
- **Debouncing**: Limits audio output to changes in detections, preventing repetitive speech.
- **FPS Monitoring**: Provides real-time performance feedback.

# 5 Sample Output

During testing, the system successfully detected a person in the webcam feed. The output log displayed the following detection details:

- **Detection Log Example 1**: At timestamp 0:480x640, 1 person detected in 198.6ms. Speed: 11.3ms preprocess, 198.6ms inference, 1.8ms postprocess per image at shape (1, 3, 480, 640).
- **Detection Log Example 2**: At timestamp 0:480x640, 1 person detected in 127.4ms. Speed: 3.6ms preprocess, 127.4ms inference, 3.6ms postprocess per image at shape (1, 3, 480, 640).
- **Detection Log Example 3**: At timestamp 0:480x640, 1 person detected in 121.8ms. Speed: 4.2ms preprocess, 121.8ms inference, 1.9ms postprocess per image at shape (1, 3, 480, 640).
- **Detection Log Example 4**: At timestamp 0:480x640, 1 person detected in 114.3ms. Speed: 3.9ms preprocess, 114.3ms inference, 1.9ms postprocess per image at shape (1, 3, 480, 640).

The system consistently identified the person with varying inference times, averaging around 120198ms per frame, demonstrating stable performance.

# 6 Challenges and Limitations

- **Lighting Conditions**: Text detection may fail under poor lighting due to reliance on thresholding.
- **Model Accuracy**: YOLOv8 may miss objects with low confidence or in cluttered scenes.
- **System Resources**: Performance depends on hardware, with lower-end devices potentially experiencing lag.
- **Tesseract Dependency**: Requires proper installation and configuration, which may vary across platforms.

# 7 Future Improvements

Potential enhancements include:

- **Adaptive Thresholding**: Implement dynamic thresholding to improve text detection in varying lighting.
- **Model Fine-Tuning**: Train YOLOv8 on custom datasets for specific use cases.
- **Multi-Threading**: Process object and text detection concurrently to boost performance.
- **User Interface**: Develop a GUI to configure settings and display results.

# 8 Conclusion

The object and text detection system effectively combines YOLOv8, Tesseract OCR, and text-to-speech to provide real-time environmental awareness through a webcam feed. Despite challenges with lighting and resource constraints, the system is optimized for performance and cross-platform use, with clear potential for further enhancements.