University of Nottingham
UK | CHINA | MALAYSIA

COMP3065: Computer Vision Coursework Report

Yiwen Xin (20321037)

May 6, 2024

words count: 1204

School of Computer Science
University of Nottingham Ningbo China

Contents

1 Introduction

In this coursework, I utilized OpenCV to perform real-time object detection using a pre-trained YOLOv3 model. Additionally, I implemented a user interface using Tkinter, allowing users to dynamically adjust parameters such as confidence threshold, non-maximum suppression (NMS) threshold, and display settings. While meeting the requirements of the assignment tasks, this provided users with a user-friendly UI interface.

2 Overview of the System

The code comprises several components:

- OpenCV (cv2): Used for all image processing tasks, including reading video frames, processing them through a neural network, and rendering output.
- YOLOv3: An efficient and powerful object detection model that detects objects in real-time by predicting bounding boxes and associated class probabilities.
- Tkinter: Provides a graphical user interface for real-time parameter adjustments by the user.

3 Video Processing Workflow

3.1 Overview

- Frame Acquisition: The video is captured frame by frame from a video file.
- Pre-processing: Each frame is pre-processed to form a blob using OpenCV's 'cv2.dnn.blobFromImage', which adjusts the frame's dimensions and color settings to be compatible with the neural network.
- Object Detection: The pre-processed frame is passed through YOLOv3 to detect objects. Each object detected is marked by a bounding box.

- Post-processing: Applies Non-Maximum Suppression to refine the bounding boxes by eliminating overlaps and ensuring that each detected object is counted once.
- Rendering: The final frame, with annotated bounding boxes and object counts, is displayed on the screen and saved to an output file.

3.2 Implementation

It starts by initializing various global variables like cap for video capture, video output for writing annotated video frames, pause to control processing pauses, show confidences to toggle confidence display, bounding box color for box colors, object count to track detected objects, and font scale for text overlay scaling.

Then, it enters a loop where it continuously reads frames from the video source (cap). For each frame, it preprocesses the image and passes it through the neural network (net) to obtain object detection predictions. These predictions are filtered based on confidence thresholds and then processed to remove redundant bounding boxes using non-maximum suppression (NMS). Bounding boxes are drawn around detected objects on the frame, and optionally, text overlays showing class labels and confidence scores are added.

The count of detected objects is updated and displayed on the frame. The annotated frame is written to the output video stream (video output) and displayed in a window titled 'frame'.

During this process, the code listens for user input key events. Pressing the 'Esc' key terminates the processing loop, while pressing the 'Space' key toggles the processing pause. Once the loop ends, it releases the video capture and output resources, and closes all OpenCV windows.

4 User interface

4.1 Overview

- Entry Widgets: Allow the user to set the confidence and NMS thresholds.

- Checkbutton: Toggles the visibility of confidence scores on the detection boxes.
- Color Chooser: Enables the selection of a color for the bounding boxes.
- Buttons: Include functions to update parameters and control the application flow, like starting or stopping the video processing.

The picture below shows the User interface

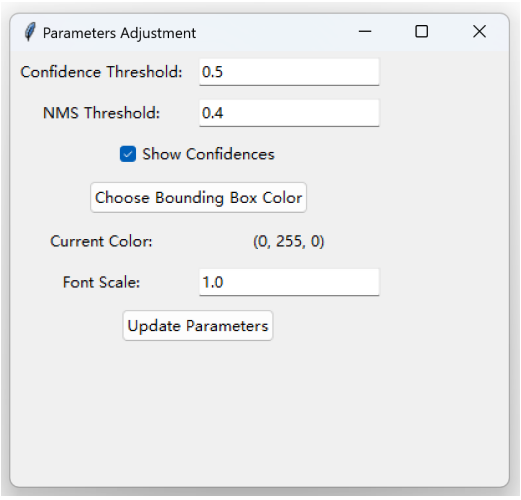

Figure 1: User interface

4.2 Implementation

The user interface (UI) functions offer an interactive platform for adjusting parameters crucial for video processing. Primarily, the update parameters() function enables dynamic updates based on user inputs. This function incorporates parameters such as confidence and non-maximum suppression (NMS) thresholds, the option to display confidences, bounding box color selection, and font scale adjustment. Additionally, the choose color() function facilitates the selection of bounding box colors via a color chooser dialog, ensuring a visually appealing representation. Moreover, to enhance user experience, a confirmation prompt is triggered via the confirm close() function

when attempting to exit the application. By encapsulating these functionalities within a graphical user interface (GUI) powered by Tkinter, users can conveniently fine-tune parameters and visually monitor video processing in real-time.

5 Additional feature

5.1 Utilization of Confidence Variable

- The confidence scores (confidences) are calculated for each detected object during inference using a pre-trained YOLOv3 object detection model.
- These scores are compared against a predefined confidence threshold (confidence threshold) to filter out low-confidence detections. Only detections with confidence scores higher than this threshold are considered significant.
- The confidence scores of the detected objects are stored in the confidences list.
- Later in the code, when drawing bounding boxes around detected objects, the confidence scores are optionally displayed alongside the object labels if show confidences is set to True.

The following is the checkbox for selecting whether to display confidences.

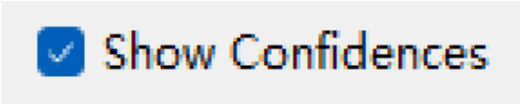

Figure 2: checkbox

5.2 Selection of Bounding Box Color

- The color for drawing bounding boxes around detected objects is chosen using the bounding box color var.
- Initially, a default bounding box color (0, 255, 0) (green) is set.

- Users have the option to choose a custom color using the choose color() function, which opens a color chooser dialog.
- Upon selecting a color from the dialog, its RGB values are converted and stored in bounding box color var.
- The selected color is then used dynamically for displaying the bounding boxes around detected objects.

The following figure shows the colorbox in UI interface

Figure 3: colorbox

5.3 Parameter Update

The function "update parameters" updates the parameters used in the video processing loop based on user input from the GUI.

- The function "update parameters" updates the parameters used in the video processing loop based on user input from the GUI.
- Non-maximum suppression threshold (nms threshold): Controls the threshold for suppressing overlapping bounding boxes.
- Show confidences (show confidences): A boolean indicating whether to display confidence scores along with object labels.

- Bounding box color (bounding box color): Allows users to choose the color of bounding boxes drawn around detected objects.
- Font scale (font scale): Determines the scale factor for font size used in displaying object labels and confidence scores.

5.4 Multi-object recognition

Multi-object recognition is implemented through a video processing loop. The code begins by initializing necessary variables and enters a loop to process each frame of the input video. Within this loop, the frame is preprocessed and passed through a pre-trained deep neural network model to detect objects. Detected objects are filtered based on confidence scores, and non-maximum suppression is applied to remove overlapping bounding boxes. Bounding boxes are drawn around the detected objects, optionally accompanied by class labels and confidence scores. The count of detected objects is displayed on each frame. Key events are handled to pause or resume video processing, and the process continues until the video ends or the user interrupts it. Finally, resources are released, and OpenCV windows are closed. This process enables real-time or offline multi-object recognition with bounding box visualization in videos.

6 Use process

After adjusting the parameters, when the user presses the "Update Parameters" button, the video processing begins. The program accepts a video in MP4 format and outputs a video in AVI format. Users can press the ESC key to terminate the video processing prematurely. Alternatively, they can wait until the video processing is complete, at which point the program will exit automatically.

Below is a sample image from the video processing process

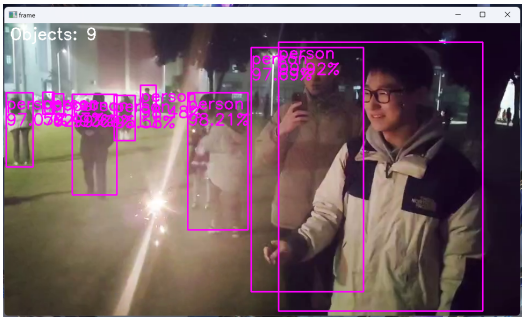

Figure 4: sample image

7 Conclusion

The code demonstrates a powerful application combining real-time object detection with an interactive user interface. It showcases how computer vision can be made accessible and adjustable in real-time, providing valuable insights and tools across various industries.

This system, by leveraging advanced technologies like OpenCV and YOLOv3 within a user-friendly environment created with Tkinter, presents a versatile platform suitable for numerous real-world applications. Future enhancements can further extend its capabilities, making it an indispensable tool in areas requiring dynamic and robust object detection.