`

Sanjay Ghodawat University Kolhapur

Established as a State Private University under Govt. of Maharashtra Act No. XL dated 3rd May 2017

Empowering Lives Globally !

Mini Project Report

On

“Object Detection Using OpenCV”

Under the guidance of

Mrs. Mrunal Deshpande

School of Computer Science & Engineering

Academic year: 2022 - 23

CERTIFICATE

This is to certify that the mini project report entitled

“Object Detection Using OpenCV”

submitted by

Amol More (36)

Aniket Patil (60)

Akash Shintre (61)

Abhishek Landage (65)

In the partial fulfilment for the Semester VI of T. Y. B. Tech of Computer Science & Engineering is a record of work carried out by the students mentioned above under the guidance and supervision of Mrs. Mrunal Deshpande during the academic year 2022-23.

Place: SGU, Atigre

Date:

Mrs. Mrunal Deshpande	**Dr. B. Suresh Kumar**
Mini Project Guide	**Head-SOCSE**

ACKNOWLEDGEMENT

We express our sincere thanks to **Mrs. Mrunal Deshpande** and **Dr. B. Suresh Kumar, Head-SOCSE** whose supervision, inspiration and valuable guidance , helped us a lot to complete our mini project work.

Their guidance proved to be the most valuable to overcome all the hurdles in the fulfilment of this mini project work.

Last but not the least, this acknowledgement would be incomplete without rendering our sincere gratitude to all those who have helped us in the completion of mini project work.

Sincerely,

1. Amol More (36)
2. Aniket Patil (60)
3. Akash Shintre (61)
4. Abhishek Landage (65)

INDEX

Sr. No.	Chapter	Page No.
1	Introduction	1
2	Literature Review	2
3	Objectives	3
4	Methodology	4-6
	4.1 General Method	4
	4.2 Block Diagram	5
	4.3 Block Diagram Description	5-6
5	System Requirements	7
6	Results	8-11
7	Conclusion	12
8	References	13

1. Introduction

The Object Detection Web App is a mini project that aims to leverage the power of the YOLO (You Only Look Once) algorithm for real-time object detection in a web-based environment. Object detection is a fundamental task in computer vision that involves identifying and localizing objects within an image or video.

The YOLO algorithm stands out among other object detection algorithms for its impressive speed and accuracy. It operates by dividing the input image into a grid and predicting bounding boxes and class probabilities directly from the grid cells. This unique approach allows YOLO to achieve real-time object detection capabilities, making it highly suitable for various applications, including surveillance, autonomous vehicles, and augmented reality.

The Object Detection Web App provides a user-friendly interface that enables users to upload images or videos for object detection. Additionally, users can utilize their webcam for real-time object detection. The web application is built using Streamlit, a popular Python library that simplifies the development of interactive web applications.

By combining the YOLO algorithm's capabilities with the ease of use and interactivity of Streamlit, the Object Detection Web App offers a seamless and accessible experience for users to detect and visualize objects of interest. The app provides visual feedback by highlighting the detected objects with bounding boxes and labels, enhancing the overall user experience.

In the following sections of this mini project report, we will delve into the project's objectives, methodology, results, and conclusion. We will explore the implementation details, the features of Streamlit utilized in the web application, and the implications of the project's success in bridging the gap between object detection algorithms and web-based user interfaces.

2. Literature Review

Object detection is a well-established field in computer vision, and numerous algorithms and techniques have been developed to tackle this task. The mini project on Object Detection Web App utilizes the YOLO (You Only Look Once) algorithm, which has gained significant attention and popularity in recent years. In this literature review, we will explore the relevant studies and advancements in object detection algorithms, as well as the role of web-based interfaces in enhancing user interaction with computer vision systems.

- **OpenCV:** OpenCV is used for image and video processing, including reading and decoding images/videos, performing object detection with YOLO, and drawing bounding boxes.

- **YOLO (You Only Look Once):** YOLO is an object detection algorithm that enables real-time detection by predicting bounding boxes and class probabilities directly from the input image or video frames.

- **Streamlit:** Streamlit simplifies web application development by providing an intuitive interface for creating interactive web interfaces, allowing users to upload images/videos, select options, and display the object detection results seamlessly.

- **NumPy:** NumPy is used for efficient numerical operations, such as handling image data, calculating detection confidence scores, and manipulating arrays of bounding box coordinates in the Object Detection Web App.

3. Objective

The main objectives of this mini project are as follows:

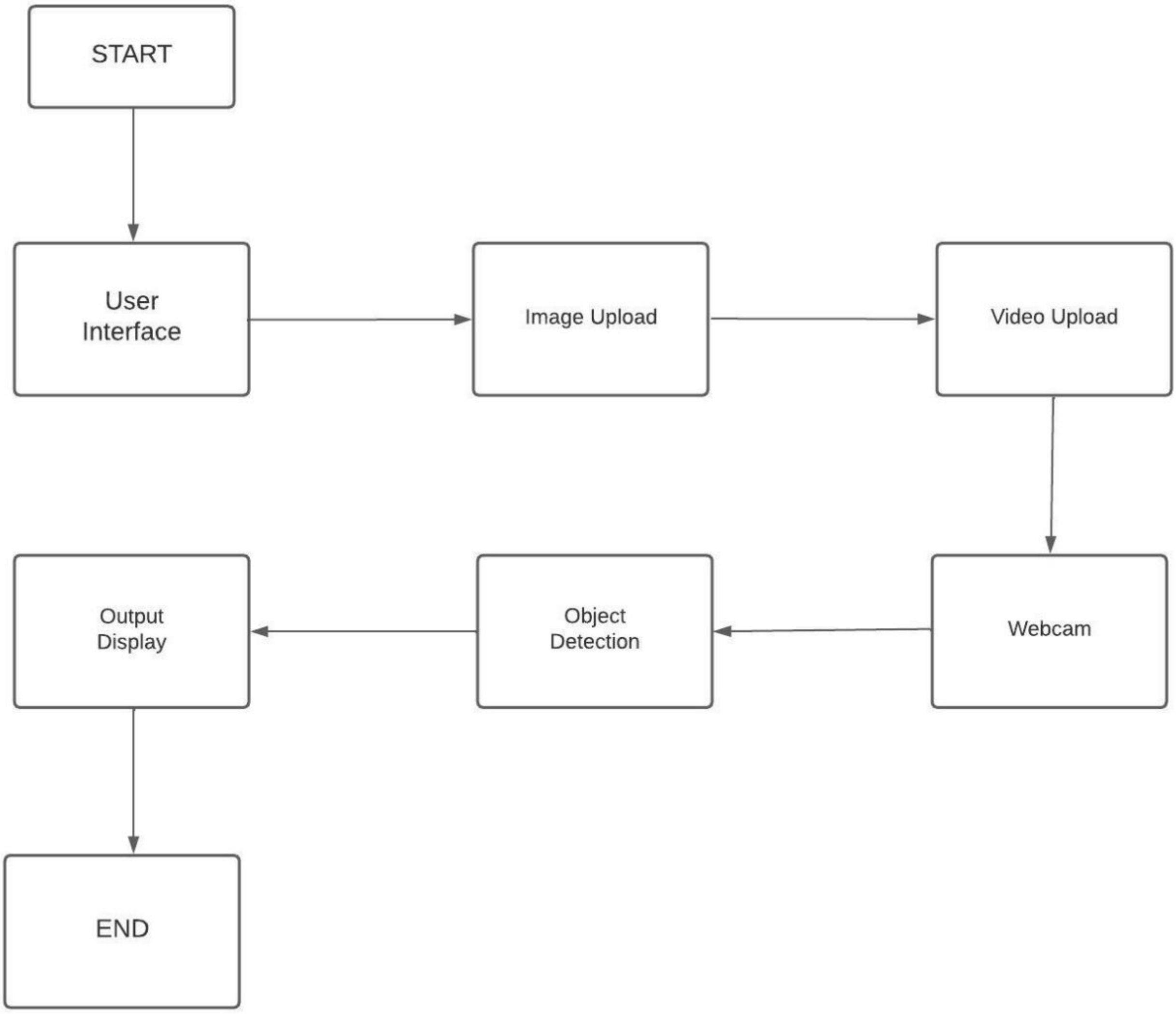
- Implement object detection using the YOLO algorithm.
- Develop a user-friendly web interface for uploading images and videos.
- Enable real-time object detection using the webcam.
- Provide visual feedback to the user by highlighting detected objects with bounding boxes and labels.

4. Methodology

The project was implemented using the following steps:

- **Loading the Pre-trained YOLO Model:** The YOLO model, consisting of weights and configuration files, was loaded using the **cv2.dnn.readNet** function.
- **Defining Object Detection Function:** The **detect_objects** function was created to perform object detection on an input image. This function utilized the YOLO model to detect objects and drew bounding boxes and labels on the image.
- **Creating the Web Application:** The Streamlit library was used to create the web application. The main function **main** was defined to handle the web application logic.
- **Image Upload Option:** Users could choose the "Image" option and upload an image file. The uploaded image was read, and object detection was performed using the **detect_objects** function. The resulting image with bounding boxes and labels was displayed in the web application using Streamlit's **st.image** function.
- **Video Upload Option:** Users could select the "Video" option and upload a video file. The uploaded video was processed frame by frame. Object detection was performed on each frame, and the resulting video with bounding boxes and labels was displayed in the web application using Streamlit's **st.video** function.
- **Webcam Option:** Users could choose the "Webcam" option to perform real-time object detection using their webcam. Frames were captured from the webcam, and object detection was performed on each frame. The resulting image with bounding boxes and labels was displayed in real-time using Streamlit's **st.image** function.

Block Diagram/Algorithm with description:



- **Start**: The flowchart begins with the start symbol.
- **User Interface**: The flowchart branches into different paths based on user input. The user can choose options like uploading an image, video, or using the webcam for object detection.
- **Image Upload**: If the user selects the image option, the flowchart proceeds to handle image uploading and processing.
- **Video Upload**: If the user selects the video option, the flowchart proceeds to handle video uploading and processing.

- **Webcam**: If the user chooses the webcam option, the flowchart proceeds to handle webcam access and real-time object detection.
- **Object Detection**: After the image, video, or webcam processing, the flowchart continues to the object detection step using the YOLO algorithm.
- **Output Display**: The flowchart includes a step to display the processed output, including bounding boxes and labels, to the user.
- **End**: The flowchart concludes with the end symbol.

5.Requirements

Software Requirements:-

- Python
- OpenCV
- NumPy
- Streamlit
- YOLO Weights and Config Files
- COCO Names File

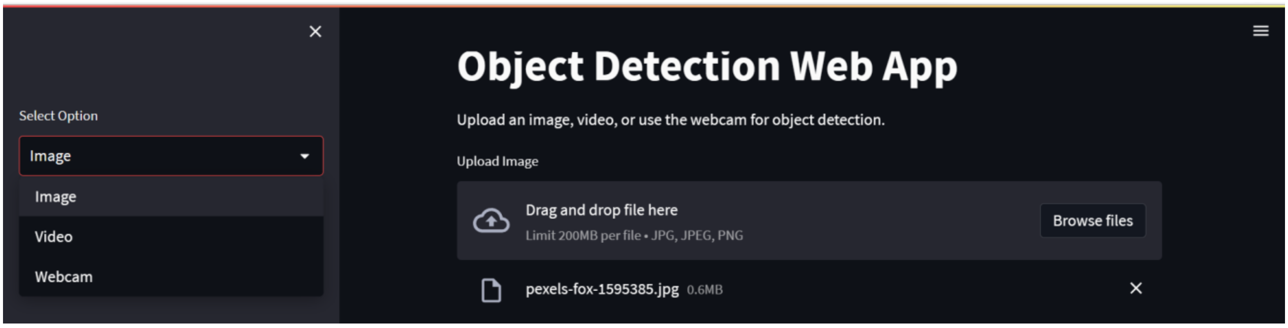
Hardware Requirements:-

- Processor intel i3/i5/i7
- Webcam
- RAM minimum 4GB
- Hard Disk minimum 15GB
- Internet connection

6.Results

Code:

```
import cv2
import numpy as np
import streamlit as st

net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')
classes = []
with open('coco.names', 'r') as f:
    classes = f.read().splitlines()

def detect_objects(img):
    height, width, _ = img.shape
    blob = cv2.dnn.blobFromImage(img, 1/255, (416, 416), (0, 0, 0), swapRB=True, crop=False)
    net.setInput(blob)
    output_layers_names = net.getUnconnectedOutLayersNames()
    layerOutputs = net.forward(output_layers_names)

    boxes = []
    confidences = []
    class_ids = []

    for output in layerOutputs:
        for detection in output:
            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)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    for i in np.array(indexes).flatten():
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence = str(round(confidences[i], 2))
        color = (255, 255, 255)
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label + " " + confidence, (x, y + 20), cv2.FONT_HERSHEY_PLAIN, 2,
color, 2)

    return img

def main():
    st.title("Object Detection Web App")
    st.write("Upload an image, video, or use the webcam for object detection.")

    option = st.sidebar.selectbox("Select Option", ["Image", "Video", "Webcam"])
    if option == "Image":
        uploaded_image = st.file_uploader("Upload Image", type=["jpg", "jpeg", "png"])
        if uploaded_image is not None:
            image = np.array(bytearray(uploaded_image.read()), dtype=np.uint8)
            image = cv2.imdecode(image, 1)
            st.image(image, channels="BGR", caption="Uploaded Image")
            st.write("---")
            st.write("Detecting objects...")
            result_image = detect_objects(image)
            st.image(result_image, channels="BGR", caption="Objects Detected Image")

    elif option == "Video":
        uploaded_video = st.file_uploader("Upload Video", type=["mp4"])
        if uploaded_video is not None:
            video = np.array(bytearray(uploaded_video.read()), dtype=np.uint8)
            video = cv2.imdecode(video, 1)
            st.video(uploaded_video, format="video/mp4")
            st.write("---")
            st.write("Detecting objects...")

            # Save video to a temporary file
            with st.spinner("Processing..."):
                temp_file = st.empty()
                temp_filename = "temp.mp4"
                with open(temp_filename, "wb") as file:
                    file.write(uploaded_video.read())

                # Perform object detection on the video
                cap = cv2.VideoCapture(temp_filename)
                video_result = cv2.VideoWriter('output.mp4', cv2.VideoWriter_fourcc(*'avc1'), 30,
(int(cap.get(3)), int(cap.get(4))))
                while cap.isOpened():
                    ret, frame = cap.read()
                    if not ret:
                        break
                    result_frame = detect_objects(frame)
                    video_result.write(result_frame)
                cap.release()
                video_result.release()
            temp_file.info("Video processing completed!")
            st.video("output.mp4", format="video/mp4")

    elif option == "Webcam":
        cap = cv2.VideoCapture(0)
        stframe = st.empty()

        while True:
            _, img = cap.read()
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Perform object detection on the webcam frame
            detected_img = detect_objects(img)

            # Display the webcam frame with detected objects
            stframe.image(detected_img, caption="Webcam", channels="RGB",
use_column_width=True)
            if cv2.waitKey(1) == 27:
                break
        cap.release()
        cv2.destroyAllWindows()
if __name__ == '__main__':
    main()
```


Object Detection Web App

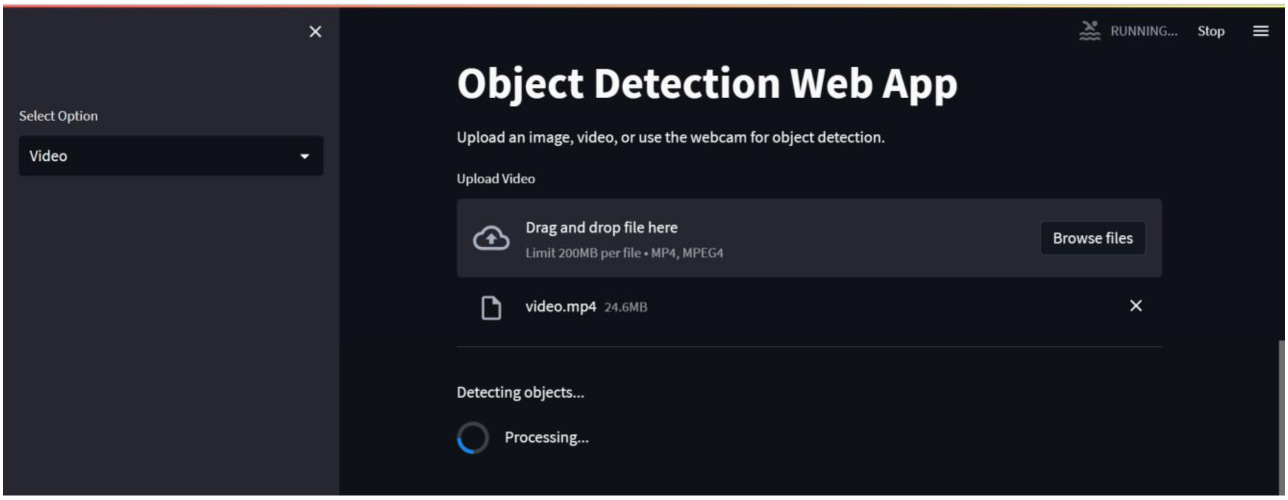
Upload an image, video, or use the webcam for object detection.

Upload Image

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

pexels-fox-1595385.jpg 0.6MB


Object Detection Web App

Upload an image, video, or use the webcam for object detection.

Upload Video

Drag and drop file here

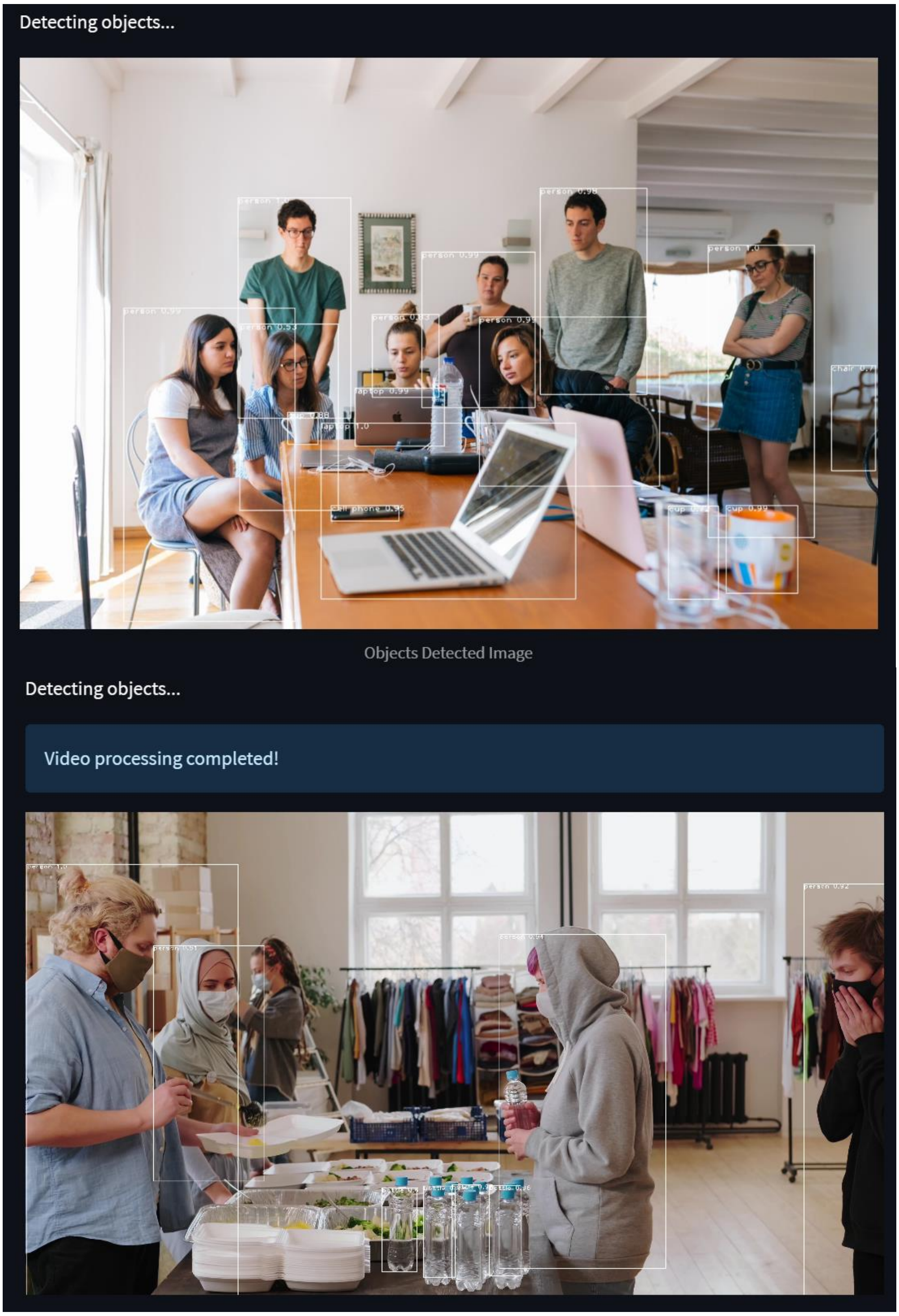
Limit 200MB per file • MP4, MPEG4

video.mp4 24.6MB

Detecting objects...

Processing...

Output:



Detecting objects...

Objects Detected Image

Detecting objects...

Video processing completed!

7. Conclusion

The Object Detection Web App successfully achieved the objectives of implementing object detection using the YOLO algorithm and providing a user-friendly web interface. The web application, built with Streamlit, allowed users to upload images or videos for object detection and also provided real-time object detection using the webcam. The detected objects were visually highlighted with bounding boxes and labels using Streamlit's interactive visualization capabilities.

The mini project demonstrated the effectiveness of the YOLO algorithm for object detection tasks and showcased the power of computer vision in real-world applications. The web interface, developed with Streamlit, provided an intuitive and accessible way for users to interact with the object detection system, thanks to Streamlit's simplicity and ease of use.

In conclusion, the Object Detection Web App mini project provided a practical implementation of object detection using the YOLO algorithm and demonstrated the capabilities of computer vision in a web-based environment. The project leveraged Streamlit's features to create an interactive and engaging user interface, making it easy for users to upload images or videos and perform object detection. The project can serve as a foundation for further enhancements and applications in the field of computer vision and object detection.

8. References

- YOLO : You Only Look Once - Real Time Object Detection - GeeksforGeeks
- YOLO Algorithm for Object Detection Explained [+Examples] (v7labs.com)
- https://www.udemy.com/course/streamlit-bootcamp/