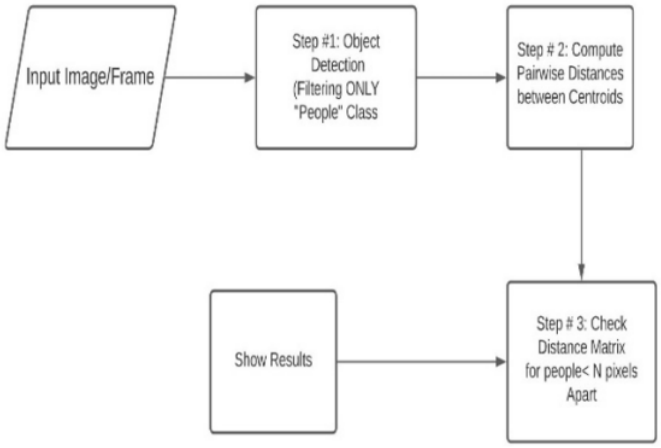
SOCIAL DISTANCE DETECTION USING DEEP LEARNING

.

INTRODUCTION

The COVID-19 pandemic has underscored the importance of social distancing to prevent virus spread. Ensuring compliance in public spaces can be challenging. This project addresses this issue by developing a system that detects social distancing violations in video feeds using YOLOv3 object detection and OpenCV for computer vision.



OBJECTIVES

- Develop a real-time system to detect people in video frames.
- Calculate the distance between detected individuals.
- Identify and highlight violations of social distancing rules.
- Display the total count of violations on each video frame.

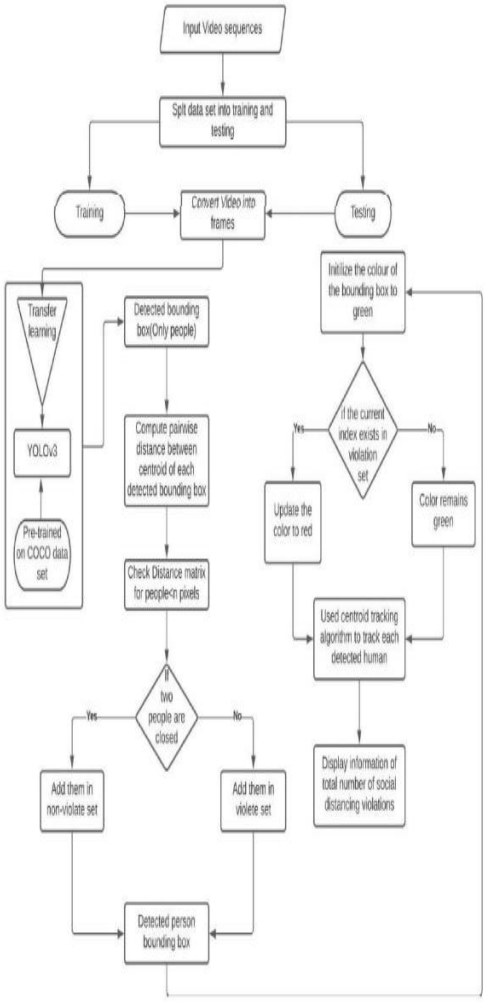
METHEDOLOGY

We have used YOLO V3 (Version-3) model which is pre-trained with COCO dataset. We have implemented this project with DNN functionalities like blogFromImage, NMS (Non maxima suppression), dark net. We use blobFromImage functionality for pre-processing of the image, NMS for suppressing the weak signals or weak detections and dark net as a framework of deep neural networks. COCO dataset contains 80 objects but we use only person object out of those objects in the project because we calculate social distancing between persons. If the detected object is person then only we proceed for next steps.

SYSTEM ARCHITECTURE

We set minimum distance as 50 pixels with minimum thresh hold as 0.3 and minimum configurations as 0.3 in this project. Libraries used for the project includes OpenCv, NumPy, argparse. We have used OpenCv for image processing functionalities. It is an open-source library, some functionalities of imutils used in project are show (for displaying images) and resize. The scipy (scientific python) package is used for importing distance metrics i.e., Euclidean distance, NumPy (numerical python) is used for working with arrays and argparse is used for passing input video as a command line argument.

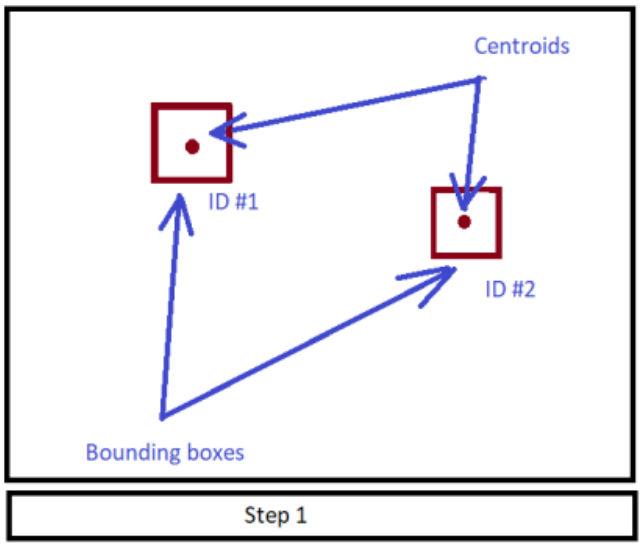
.



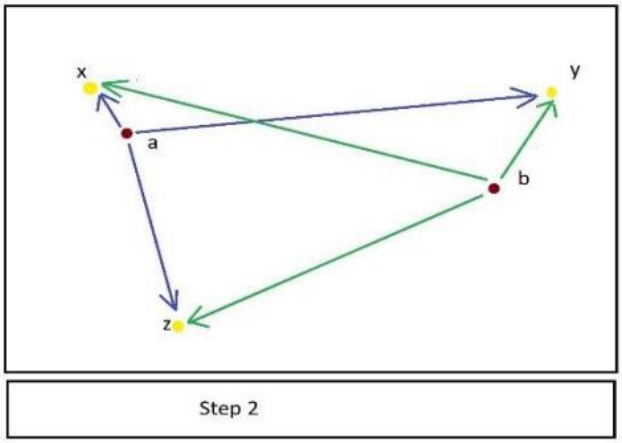
IMPLEMENTATION

Module 1: Object detection using YOLO V3 algorithm Here object refers to the person from the COCO dataset which is used with YOLO weights for training. From the objects of COCO dataset we consider only person object. If the object is person then only the object will be detected other objects will not be detected.

Module 2: Object Tracking using OpenCv Step 1: Accepting the bounding boxes and computing the corresponding centroids which means the center of bounding boxes.

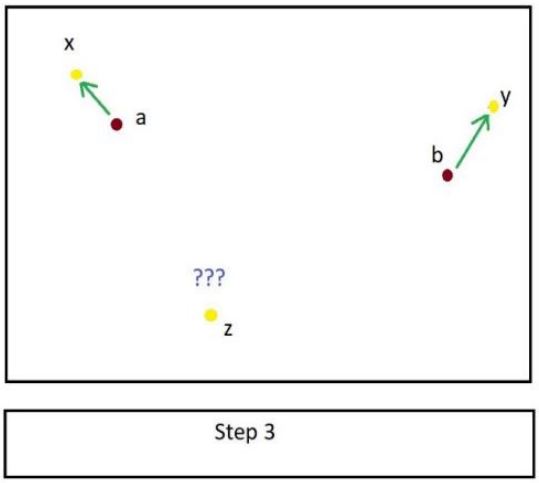


Step 2: Computing Euclidean distance between new centroids (yellow) represented by x, y, z and old centroids (brown) represented by a, b. • Centroid tracking works on an assumption that the pair of centroids with minimum Euclidean distance or the closest pair is must be the same person. So, unique ID will be generated to that pair i.e., person. • In the above image there are two existing centroids and three new centroids from the previous frame which describes a new person has been detected in this frame.



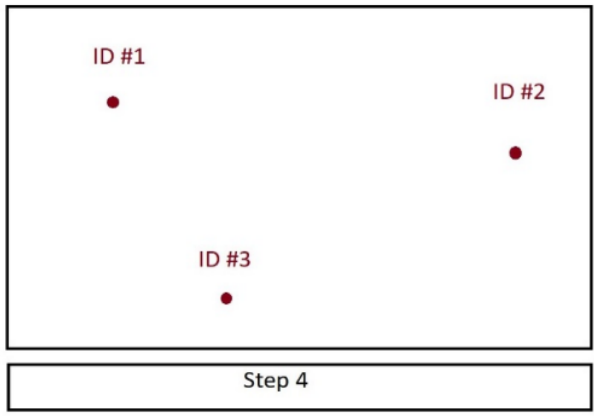
Step 3: Association of ID's as we know Euclidean distances

.



Step 3

Step 4: New ID's will be registered by storing co-ordinates of bounding boxes of new object i.e., person.



Step 4

Step 5: The object which leaves our frame area then we will just deregister the object.

Module 3: Distance computation using Euclidean distance as metric Computation of distance between the pairs of detected persons using Euclidean distance as metric.

Module 4: Adding violations and displaying number of violations
The persons who are less than M pixels (M represents minimum distance set by us in the project) distance apart from each other will be considered as violation pair.

They will be highlighted with red color bounding boxes and the number of violations at that particular frame will be displayed. The persons who are maintaining more than M pixels of distance will be considered as non-violation pair and they will be highlighted with green color bounding boxes.

RESULTS

We used argparse by which we gave our input video as argument for displaying the output. In output, Social distancing violations will be displayed for that particular frame and violated persons will be highlighted and non-violated persons will also be highlighted.

Future Enhancements

1. Enhanced Detection: Integrate more advanced models for better accuracy.
2. Multi-Camera Support: Extend the system to handle multiple camera feeds.
3. Alert System: Implement a real-time alert system for immediate notification of violations.

.

Conclusion

This project successfully demonstrates a system that detects social distancing violations using YOLOv3 and OpenCV. By leveraging computer vision techniques, it provides a scalable and efficient solution for monitoring social distancing in real-time.

References

- YOLOv3: https://pjreddie.com/darknet/yolo/
- OpenCV: https://opencv.org/
- Numpy: https://numpy.org/
-COCO dataset available: https://www.kaggle.com/awsaf49/coco-2017-dataset