Low Level Design

Social Distancing Detection System

Written By	Sai Kumar Reddy k
Documentation Version	0.1
Last Revised Date	11-03-2022

Document Control

Change Record:

Version	Date	Author	Comments
0.1	11-March-2022	Sai Kumar	

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

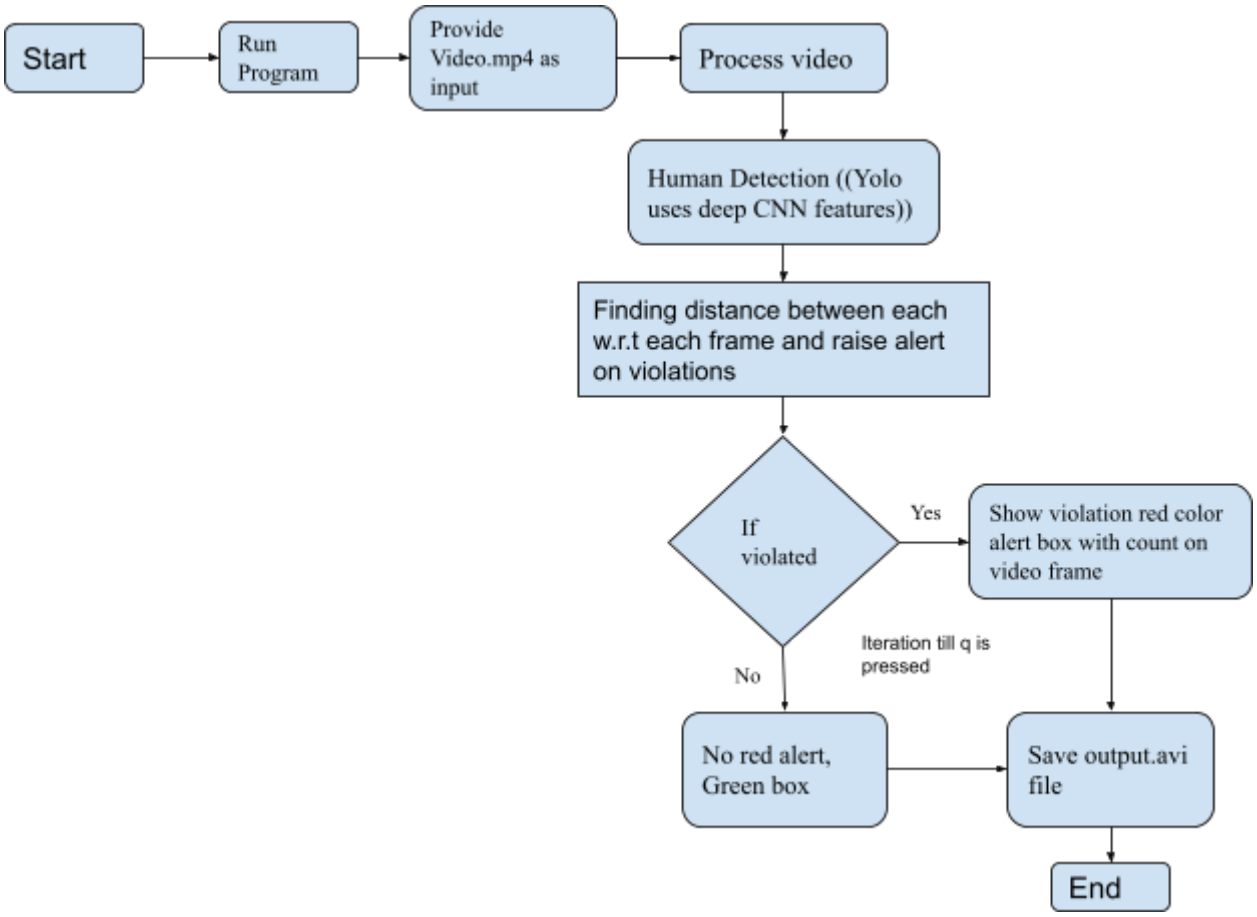
1.1. What is a Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for the Social Distancing Detection System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmers can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2. Architecture



3. Architecture Description

3.1. Yolov3

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. YOLO uses features learned by a deep convolutional neural network to detect an object.

3.2. Open CV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

3.3. Euclidean Distance

Euclidean Distance represents the shortest distance between two points. Most machine learning algorithms including K-Means use this distance metric to measure the similarity between observations.

3.4. Object detection:

- We will be using YOLOv3, trained on the COCO dataset for object detection.
- In general, single-stage detectors like YOLO tend to be less accurate than two-stage detectors (R-CNN) but are significantly faster.
- YOLO treats object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities.
- It is used to return the person's prediction probability, bounding box coordinates for the detection, and the centroid of the person.

3.5. Distance calculation:

- NMS (Non-maxima suppression) is also used to reduce overlapping bounding boxes to only a single bounding box, thus representing the true detection of the object. Having overlapping boxes is not exactly practical and ideal, especially if we need to count the number of objects in an image.
- Euclidean distance is then computed between all pairs of the returned centroids. Simply, a

centroid is the center of a bounding box.

- Based on these pairwise distances, we check to see if any two people are less than/close to 'N' pixels apart.

3.6. Features:

The following are examples of the added features. Note: You can easily on/off them in the config. options (mylib/config.py):

1. Real-Time alert:

- If selected, we send an email alert in real-time. Use case: If the total number of violations (say 10 or 30) exceeded in a store/building, we simply alert the staff.
- You can set the max. violations limit in config (Threshold = 15).
- This is pretty useful considering the COVID-19 scenario.

Note: To set up the sender email, please refer to the instructions inside 'mylib/mailer.py'. Setup receiver email in the config.

2. Threading:

- Multi-Threading is implemented in 'mylib/thread.py'. If you ever see a lag/delay in your real-time stream, consider using it.
- Threading removes OpenCV's internal buffer (which basically stores the new frames yet to be processed until your system processes the old frames) and thus reduces the lag/increases fps.
- If your system is not capable of simultaneously processing and outputting the result, you might see a delay in the stream. This is where threading comes into action.
- It is most suitable for solid performance on complex real-time applications. To use threading:

set Thread = True in the config.

3. People counter:

- If enabled, we simply count the total number of people: set People_Counter = True in the config.

4. Desired violations limits:

- You can also set your desired minimum and maximum violations limits. For example, MAX_DISTANCE = 80 implies the maximum distance 2 people can be closer together is

80 pixels. If they fell under 80, we treat it as an 'abnormal' violation (yellow).

- Similarly MIN_DISTANCE = 50 implies the minimum distance between 2 people. If they fell under 50 px (which is closer than 80), we treat it as a more 'serious' violation (red).
- Anything above 80 px is considered as a safe distance and thus, 'no' violation (green).

4. Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application is accessible to the user	1. Application should be installed with requirements.txt file	Application should be accessible to the user
Verify whether the Application loads completely for the user when the system is accessed	1. 1. Application should be installed with requirements.txt file 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether the User is able to run the application	1. Application is accessible	The User should be able to run the application
Verify whether user is able to successfully run the application in GPU runtime	1. Application is accessible 2. User is able to run the application	User should be able to successfully run the application with GPU runtime
Verify whether user is able to mount drive in google collab	1. Application is accessible 2. User is able to run the application	User should be able to mount drive in google callable and able to access the files from drive
Verify whether user is able to input video file	1. Application is accessible 2. User is able to run the application 3. User is able to input data	User should be able to provide input a video file for social distance detection

Verify whether the paths to respective files configured properly or not	1. Application is accessible 2. User is able to run the application 3. User is able to input data 4. All files are configured	User should be provided with respective config files and paths
Verify whether user is able to run the system or not	1. Application is accessible 2. User is able to run the application 3. User is able to input data 4. All files are configured	User should be able to run the system without predefined errors
Verify whether user is able to see the processing of imputed video file	1. Application is accessible 2. User is able to run the application 3. User is able to input data 4. All files are configured	User should be able to see process happening on providing video input file
Verify whether user is able to observe the boxes, alerts for violations per each frame of input video file	1. Application is accessible 2. User is able to run the application 3. User is able to input data 4. All files are configured 5. User able to observe the process and steps involved in the system	User should be able to observes the steps and process involved like the boxes, alerts for violations per each frame of input video file
Verify whether User is able to quit the process on entering q	1. Application is accessible 2. User is able to run the application 3. User is able to input data 4. All files are configured 5. User able to observe the process and steps involved in the system 6. User is able to quit	User should be able quit the process
Verify whether user is able to download the output.avi file	1. Application is accessible 2. User is able to run the application	User should be able to access the output file

	3. User is able to input data 4. All files are configured 5. User able to observe the process and steps involved in the system 6. User is able to download output file	