High Level Design (HLD)

SDD (Social Distancing Detection)

Revision Number: 1.0

Last Date of Revision: 07/03/2022

Document Version Control

Date Issued	Version	Description	Author
01-02-2022	1	Initial HLD-v1.0	Meenakshi Rao
05-03-2022	2	Updated KPI-v1.1	Meenakshi Rao

Contents

Abstract

Social distancing detection is very much needed in this pandemic time. This project uses deep learning to evaluate the distance between people so that helps to monitor healthy distance. The detection tool was developed to alert people to maintain a safe distance with each other by evaluating a video feed. The video frame from the camera was used as input, and the open-source object detection pre-trained model.

Main technologies used here are OpenCV, Deep Learning, and Computer Vision. The main objective here is, first human detecting, secondly measure distance between them, if any violation displays that.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact as a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	Description
SDD	Social Distancing Detection
HLD	High-Level Design
Database	Collection of the information monitored by this system
IDE	Integrated Development Environment
YOLO	You Only Look Once

2 General Description

2.1 Product Perspective

This is the base or starting point for measuring healthy distance between people. It can be improved further to bring it as a product.

2.2 Problem Statement

Social distancing detection using deep learning to evaluate the distance between people to mitigate the impact of this coronavirus pandemic. The detection tool was developed to alert people to maintain a safe distance with each other by evaluating a video feed. The video frame from the camera was used as input, and the open-source object detection pre-trained model.

2.3 PROPOSED SOLUTION

The solution proposed here is SDD Social Distancing Detection system is a deep learning, computer vision-based model helps to detect people and alert people if they don't follow minimum distance.

2.4 FURTHER IMPROVEMENTS

- Distances are measured in terms of pixels. Distances need to be mapped to measurable units like meters, feet etc. As a beginning of improvement, "utilize a proper camera calibration.
- People detection process is done by OpenCV's YOLO which leads to slow process due to post-processing required by model. Alternate to improve "frame throughput" could be SSD Single Shot Detector on GPU.

2.5 Technical Requirements

This document addresses the requirements for person detection by processing frames in a video stream, calculate the "pairwise distances" among all detected people. After computing these distances verify whether any pair of people coming under distance less than predefined N pixels apart.

- YOLO object detector model is used here for person detection. *yolo-coco* folder contains all necessary files required for people detection. These files include CNN architecture definition, pre-trained weights and class names. This YOLO model is compatible with OpenCV's DNN model.
- Along with OpenCV, we use *numpy* and *argparse* libraries.
- *imutils* library has *resize* () function which is used here to reduce dimensions of input video, otherwise large video difficult to process.

2.6 Data Requirements

- Video from camera is input to this model.
- YOLO object detector files are required to be downloaded. It consists of CNN architecture definition, pre-trained weights and class names.
- Defined some variables like non-maxima suppression threshold NMS_THRESH, minimum object detection confidence or probability MIN_CONF and minimum safe distance in pixels that two people can be far from each other MIN_DISTANCE.

2.7 Tools Used

Python Programming Language and frameworks such as NumPy, imutils and OS

- *PyCharm* is used a IDE
- *GitHub* is used for making all works available to the public
- *OpenCV* is used for people detection
- *Yolo* module gives necessary files to load CNN model for people detection.
- *Colab* notebook for speed up the process.

2.7.1 Hardware Requirements

Laptop with 12GB RAM, 64-bit processor, Window-10 Operating System.

2.8 Constraints

System should be user-friendly, and end users should not be required to know the methodology of project modules.

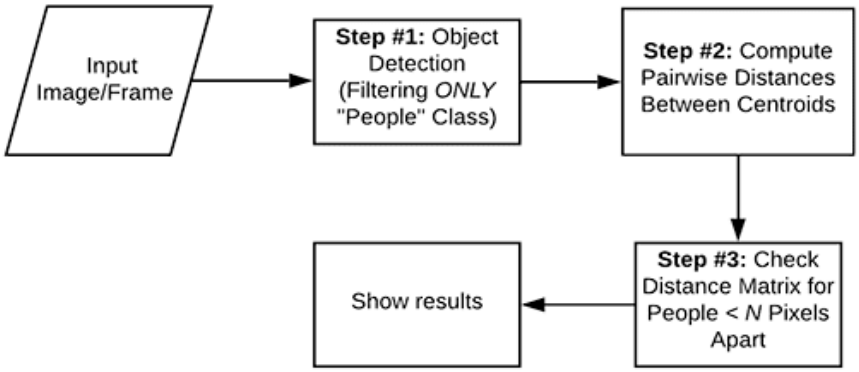
2.9 Assumptions

As a input mp4 video has taken and assuming model is compatible with any other videos of other type of sources like webcam with no or minimum modifications.

3 Design Details

3.1 Process Flow

Proposed Methodology is given below:



3.2 Event Log

The system should log every event so that the user will know what process is running internally. Developer can choose among file or database-based logging. Logging helps to debug issues if any. Logging is mandatory in every system design.

3.3 Error Handling

Logical errors, unexpected errors need to handled wisely so that it will end up with getting stuck while execution. Should errors be encountered, as exception will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

3.4 Performance

Model should produce expected results. There should not be ambiguity in people detection or measuring pair-wise distance. There is always option for future enhancements.

3.5 Reusability

The code written and the components used should have the ability to be reused with no problems.

3.6 Application Compatibility

Different modules are used in this project with specific task to do. Proper transfer of information should happen among these modules. Final model should able to run on different platform as well.

3.7 Resource Utilization

When any task is performed it will likely use all the processing power available until that function is finished.

3.8 Deployment

Final application can be deployed also in one of the platform like Microsoft Azure, Google Cloud or Amazon.

4 Dashboards

As per the project requirement, Dashboards may be necessary to display and indicated certain KPIs and relevant indicators for the problems that if not addresses in time could cause bad impact.

4.1 KPIs (Key Performance Indicators)

- Person detection
- Displaying healthy and unhealthy distances among people
- Saving output for future reference
- Successful logging
- Modular coding
- Exception’s handling

5 Conclusion

The designed model can measure healthy distance between people and display it. As a future enhancement measurement can be in feet or meters. And another improvement can be making it compatible to work with different types of video inputs