Table of Contents

CHAPTER 1: Introduction

From the learning gained during the Deep Learning Fundamentals training, I’ve made a Realtime Human Detection and Counter using MobileNet for video surveillances.

Monitoring the number of store visitors online has become very important during these difficult times of the pandemic. It is necessary to control the number of visitors to the premises to meet the requirements of social distance. This process can be automated by installing real-time people counting sensors that help track the number of people in stores and display automated notifications to incoming shoppers on any screen with an Internet connection, such as a TV, tablet, or smartphone. In this paper we propose a real time crowd counting method using OpenCV. With reference to the previous methods such as edge recognition, morphological filter, SVM order which are not real time applications, our method trains the system using video streaming and provides result in real time. OpenCV for people counting, image processing and deep learning object detector are used. This method leverages both object detection and tracking to improve the accuracy of the people counter.

Single image crowd counting method evaluates the number of people in the crowded image. Conventional methods are challenging due to severe obstruction and complex backgrounds. Crowd count detection has various applications such as public safety, scheduling trains, traffic control etc. In our proposed method we make use of OpenCV is a programming language that can be used to perform standard computer vision and image processing tasks. Related work done in this field included crowd detection using detection and regression methods, RGBD counting and multitask strategies. These methods lacked accuracy. The main objective is to develop a simple real- time system to count the number of people by using OpenCV. This is mainly to ensure safety of people under all circumstances. It is also possible to use a specialized LED screen, which can be installed both indoors and outdoors. On the screen, it is possible to display the current data of the room occupancy level, as well as specialized notifications in green and red, according to the settings of the allowed visitor quantity in the premises. The screen can be mounted on a stand or a wall.

CHAPTER 2: Deep Learning Concepts & Frameworks

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It’s achieving results that were not possible before.

2.1 Transfer Learning

Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours.

2.2 MobileNet

MobileNet is a type of convolutional neural network designed for mobile and embedded vision applications. They are based on a streamlined architecture that uses depthwise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices.

2.3 Computer Vision

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. It is most widely used field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs. Different types of computer vision include image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching.

Computer Vision itself is a big domain and is divided into various subdomains like scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual servoing, 3D scene modeling, and image restoration.

Application of Computer Vision

It has various different application that too in various fields. Some of them are listed below:

- Object Detection
- Screen Reader
- Intruder Detection
- Code and Character Reader
- Robotics
- Motion Analysis
- Image Restoration

There are many left to list as it is very wide topic and here in this project, we have used one of the application i.e. Object Detection.

Detection and Enumeration in Computer Vision

Detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.

For Detection process in computer vision, there are various methods and each one have different level of accuracy according to their advancement level, like is some methods that is invented in very early stage, they give more cases of false detection as compared to the advanced methods that had been discovered after that.

CHAPTER 3: System architecture

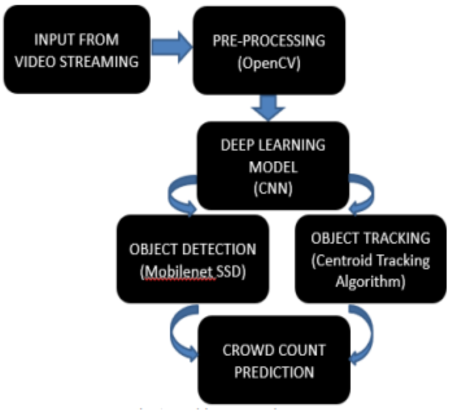

INPUT FROM VIDEO STREAMING
PRE-PROCESSING (OpenCV)
DEEP LEARNING MODEL (CNN)
OBJECT DETECTION (Mobilenet SSD)
OBJECT TRACKING (Centroid Tracking Algorithm)
CROWD COUNT PREDICTION

Figure.0.1Architecture Diagram

A webcam is used for video input. The throughput rate of estimated frames per second (FPS) is calculated. The frames are then swapped to RGB and resized. Image processing and standard computer vision functions are performed using OpenCV. OpenCV is also used for opening and writing of video files, deep neural network inference and showing the output frames on screen. In the proposed system we use Object Detection and Object Tracking in two phases for more accuracy. Mobile-Net Single Shot Detector is used for Object Detection.

We run it only once after every N frames since it is expensive. Mobile-Net contains all the pre-trained deep learning model files. An object tracker is used for every object detected to keep a track of the objects. We use a combination of Correlations filters and centroid tracking algorithm for the tracking purpose. Firstly, we use the coordinates of bounding boxes to determine the center, which is also called centroid. Then the Euclidean distance between existing and new centroids is calculated and given object Ids. Objects that have entered the field are registered and those which have left are deregistered. Dlib is used for the implementation of object tracking. OpenCV and Python is used to implement and predict the crowd count.

3.1 Methodology

3.1.1 ***Video Streaming*****:** For object detection, we work with a webcam and calculate the Frames Per Second (FPS) throughput rate. When working on this problem the first two constraints to think of is capability with FPS and Accuracy.

3.1.2 ***Pre-Processing*****:** Frames are pre-processed by resizing and switching to rgb. OpenCV is a library for performing common computer vision and image processing tasks. Deep neural network inference, opening and writing video files, and showing output frames to our screen will all be done with OpenCV.

3.1.3 ***Object Detection***: It is a computer technology that deals with identifying instances of semantic objects of a certain class in images and videos and constructing bounding boxes around those objects. It is related to computer vision and image processing.

3.1.4 ***Object Tracking***: We use the centroid tracking algorithm for this. The center is calculated using bounding boxes. The distance between new and existing centroids is then determined using Euclidean geometry. It also unregisters objects that have been removed from the field.

3.2 Algorithm

3.2.1 ***Deep Learning Algorithm***: Deep learning is often regarded as a branch of artificial intelligence. It's an area that relies on analysing computer formulas to learn and improve. Although AI makes use of simpler assumptions, Deep learning makes use of phoney neural networks that are meant to simulate how people consider and understand. Up until now, neural organizations were constrained by figuring power, and thus intricacy was restricted. Larger, more refined neural networks have been made possible by advances in Big Data processing, allowing computers to notice, understand, and respond to complex situations faster than humans. Image order, language interpretation, and discourse acknowledgment have all been aided by deep learning.

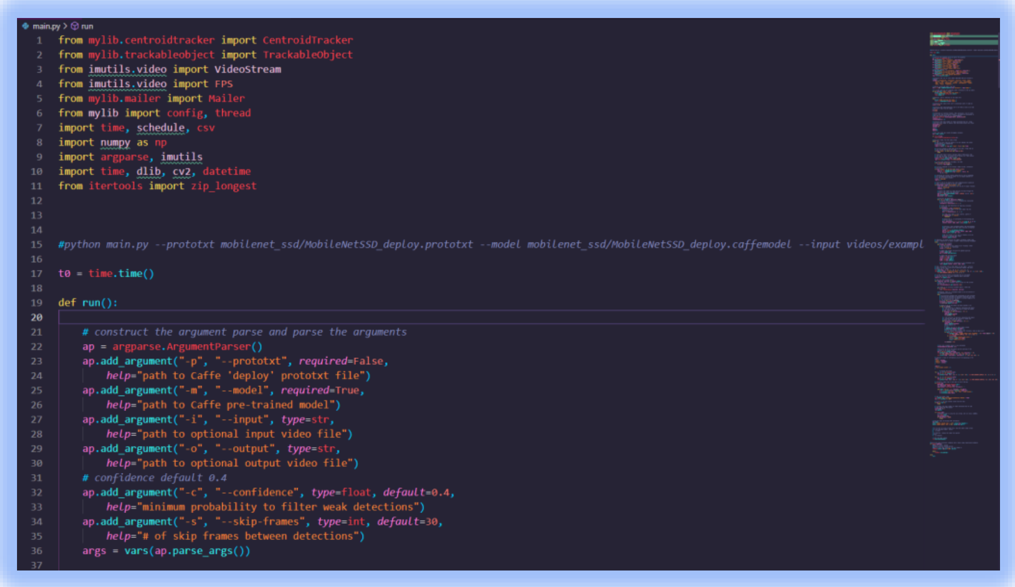
3.2.2 ***Centroid-based Tracking Algorithm***: Centroid-based tracking is a tracking algorithm that is simple to understand but extremely effective. Since it is based on the Euclidean distance between one current object centroids and the second new object centroids between subsequent frames in a film, this object tracking algorithm is known as centroid tracking. The centroid tracking algorithm uses (x, y) coordinates for every detected object in each frame, assuming that some sets of the bounding box are transferred. Bounding boxes must be calculated for each frame of the film, or, to put it another way, for each object identified by the camcorder. Following the assignment of bounding boxes within the frame with their (x, y) coordinates, the centroid of each bounding box is determined, and each bounding box is given a unique ID. The centroid of an object is computed in each subsequent frame using the bounding box definition that we discussed earlier. However, giving a new unique ID for each detection of the thing which hinder the objective of object tracking, so we'll see if we can compare the centroid of the new object to that of an existing object to resolve this, and to do so, we'll use the distance formula to measure the Euclidean distance between the two objects.

3.2.3 ***OpenCV***: OpenCV (Open Source Computer Vision Library) is a programming library for computer vision and artificial intelligence. The aim of OpenCV was to give PC vision applications a logical framework and to speed up the use of machine learning in business processes. Since OpenCV is a BSD-approved project, it makes it easier for organizations to use and change the code. The library contains 2500 enhanced figures and includes a detailed game plan for both masterpieces as well as cutting-edge PC vision and AI computations. These figurines are frequently used to recognize and see faces, perceive objects, depict human activities in chronicles, monitor camera improvements, track moving articles, separate 3D models of things, generate 3D point fogs from sound framework cameras, enter pictures to give a significant standard image of a whole scene, find tantamount pictures from a picture database, and so forth, destroy red eyes from photos taken with stripe, monitor eye progressions, see the view and find markers to overlay it with augmented reality, and so on. OpenCV has a consumer community of over 47 thousand people, with a download count of over 18 million.

3.2.4 ***Single Shot Detector Algorithm***: Single Shot Detector (SSD) Algorithm is used for object detection. Mobilenet SSD contains all the pretrained deep learning model files. SSD comprises of two parts: Extract feature maps and Apply convolution filter to detect objects. The Single Shot Detector (SSD) is intended to be independent of the base network, allowing it to work on top of any base network, including VGG, YOLO, and Mobile-Net. Mobile-Net was integrated into the SSD framework to address the challenges of running high-resource and power- consuming neural networks on low-end devices in real time.

CHAPTER 4: Screenshots

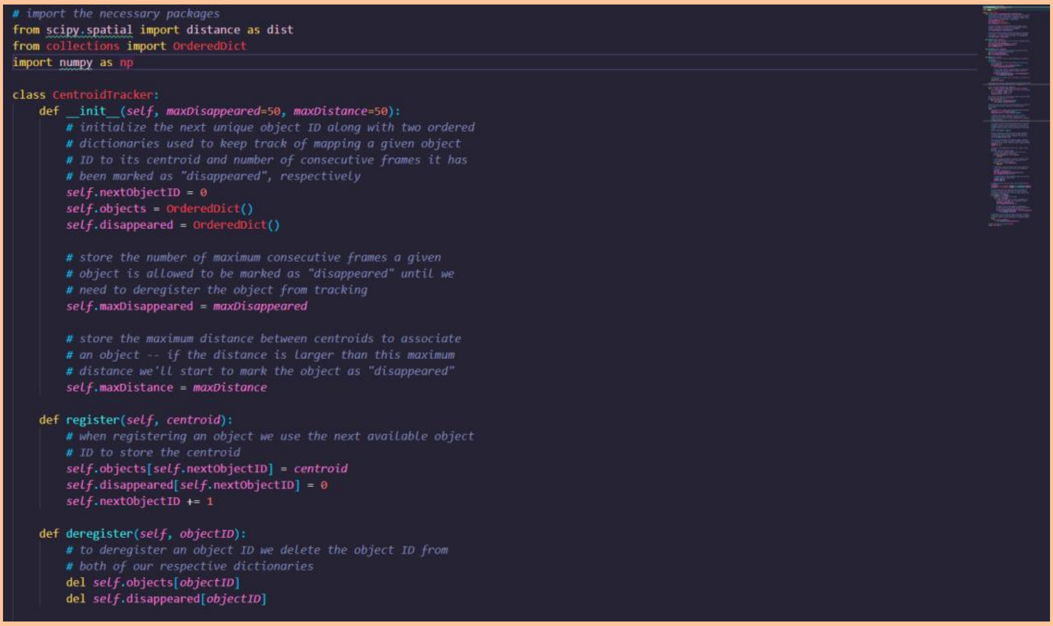
main.py



```python
from mylib.centroidtracker import CentroidTracker
from mylib.trackableobject import TrackableObject
from imutils.video import VideoStream
from imutils.video import FPS
from mylib.mailer import Mailer
from mylib import config, thread
import time, schedule, csv
import numpy as np
import argparse, imutils
import time, dlib, cv2, datetime
from itertools import zip_longest



#python main.py --prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel --input videos/exampl

t0 = time.time()

def run():

    # construct the argument parse and parse the arguments
    ap = argparse.ArgumentParser()
    ap.add_argument("-p", "--prototxt", required=False,
        help="path to Caffe 'deploy' prototxt file")
    ap.add_argument("-m", "--model", required=True,
        help="path to Caffe pre-trained model")
    ap.add_argument("-i", "--input", type=str,
        help="path to optional input video file")
    ap.add_argument("-o", "--output", type=str,
        help="path to optional output video file")
    # confidence default 0.4
    ap.add_argument("-c", "--confidence", type=float, default=0.4,
        help="minimum probability to filter weak detections")
    ap.add_argument("-s", "--skip-frames", type=int, default=30,
        help="# of skip frames between detections")
    args = vars(ap.parse_args())

```

Figure 4 1

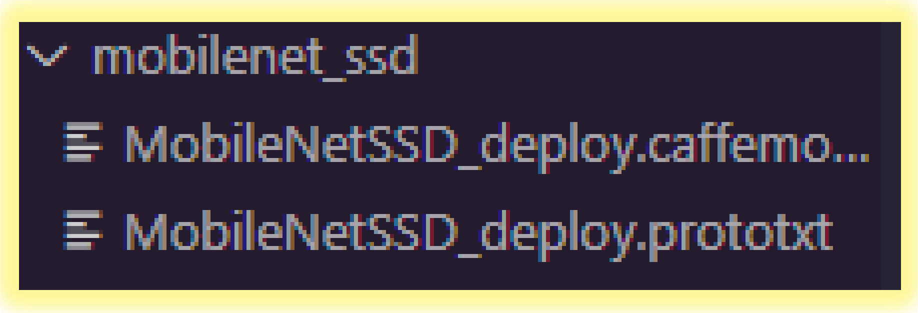
Centroidtracker.py



```python
# import the necessary packages
from scipy.spatial import distance as dist
from collections import OrderedDict
import numpy as np

class CentroidTracker:
    def __init__(self, maxDisappeared=50, maxDistance=50):
        # initialize the next unique object ID along with two ordered
        # dictionaries used to keep track of mapping a given object
        # ID to its centroid and number of consecutive frames it has
        # been marked as "disappeared", respectively
        self.nextObjectID = 0
        self.objects = OrderedDict()
        self.disappeared = OrderedDict()

        # store the number of maximum consecutive frames a given
        # object is allowed to be marked as "disappeared" until we
        # need to deregister the object from tracking
        self.maxDisappeared = maxDisappeared

        # store the maximum distance between centroids to associate
        # an object -- if the distance is larger than this maximum
        # distance we'll start to mark the object as "disappeared"
        self.maxDistance = maxDistance

    def register(self, centroid):
        # when registering an object we use the next available object
        # ID to store the centroid
        self.objects[self.nextObjectID] = centroid
        self.disappeared[self.nextObjectID] = 0
        self.nextObjectID += 1

    def deregister(self, objectID):
        # to deregister an object ID we delete the object ID from
        # both of our respective dictionaries
        del self.objects[objectID]
        del self.disappeared[objectID]
```

Figure 4 2

Mobilenet Model:



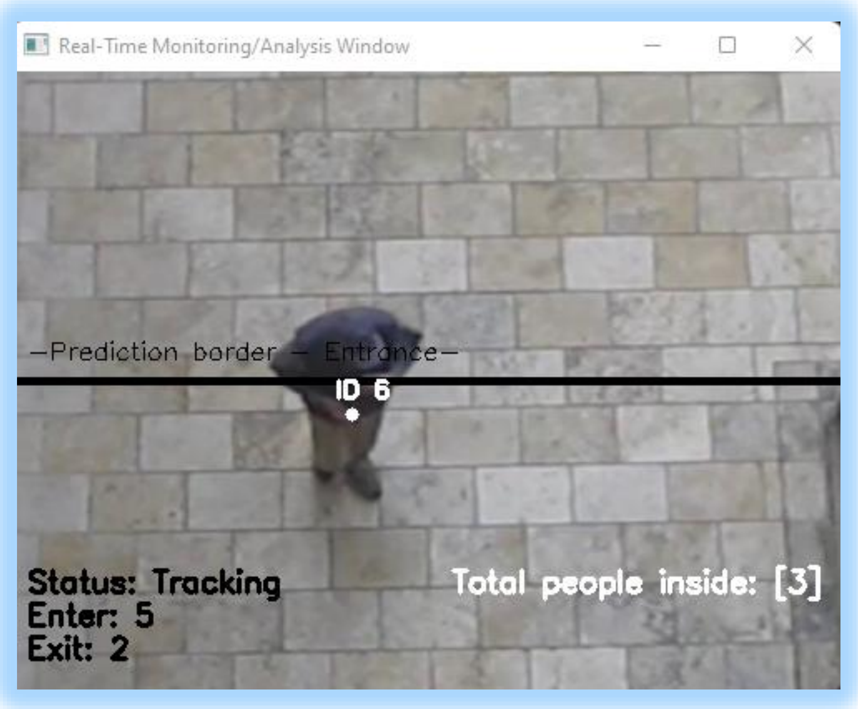
mobilenet_ssd

MobileNetSSD_deploy.caffemo...

MobileNetSSD_deploy.prototxt

Figure 4 3

Output:



Real-Time Monitoring/Analysis Window

—Prediction border — Entrance—

ID 6

Status: Tracking
Enter: 5
Exit: 2

Total people inside: [3]

Figure 4 4

CHAPTER 5: Project Summary and Conclusion

To conclude we present a surveillance system that consists of the following components visual tracking, crowd segmentation, and a counting/event recognition module. Our experimental results document that there is a significant benefit of making extensive use of the site geometry to constrain the people detection problem and to extract relevant scene information. The system is capable of keep track of people coming in and out of a place and track this over time. This model-based approach also allows to make effective use of spatial context which enables the system to detect certain events automatically. This would help authorities to prevent COVID-19 social gathering violations during the pandemic.