People Counting and Tracking System

A UG PROJECT PHASE-1 REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

GOWROJU LAXMI PRIYA	**18UK1A0518**
CHALLAKONDA SRICHANDANA	**18UK1A0567**
JUNNUTHULA AKSHAYA	**18UK1A0523**
MANDA MANASA	**18UK1A0531**

Under the guidance of

Ms. ARSHIYA FARHEEN

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

(Affiliated to JNTUH, HYDERABAD)

BOLLIKUNTA, WARANGAL (T.S) – 506005

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

BOLLIKUNTA, WARANGAL – 506005

2018-2022

CERTIFICATE OF COMPLETION

UG PROJECT PHASE-1

This is to certify that the UG Project Phase-1 entitled **“People Counting and Tracking System”** is being submitted by **GOWROJU LAXMI PRIYA (18UK1A0518), CHALLAKONDA SRICHANDANA (18UK1A0567), JUNNNUTHULA AKSHAYA (18UK1A0523), MANDA MANASA (18UK1A0531)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** to **Jawaharlal Nehru Technological University Hyderabad** during the academic year 2021- 2022, is a record of work carried out by them under the guidance and supervision.

Project Guide
MS. ARSHIYA FARHEEN
(Assistant Professor)

Head of the Department
Dr R NAVEEN KUMAR
(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved DR **P PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **DR R NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **Ms. ARSHIYA FARHEEN**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

GOWROJU LAXMI PRIYA	**18UK1A0518**
CHALLAKONDA SRICHANDANA	**18UK1A0567**
JUNNUTHULA AKSHAYA	**18UK1A0523**
MANDA MANASA	**18UK1A0531**

ABSTRACT

Real-time people flow estimation can be very useful to gain insights for many commercial and non- commercial applications. Counting people on streets or at entrances of places is indeed beneficial for security, tracking, and marketing purposes. People counters can be used to monitor occupancy of entire buildings, individual rooms or anything some of the application where you can implement people counters are

- Retail stores and supermarkets
- Higher education
- Corporate workplaces
- Restaurants, hospitality and leisure facilities

Object tracking techniques use methods like deep sort, centroid tracker, csrt, kcf, and camshift which track the detected object by comparing the similarity of detected objects with each other in each processed frame. If the object has the same similarity metric throughout the frame then it will track the same object throughout the sequence of frames and retain the same object ID for that object. This constant object ID for a particular object makes it easier for us to do the counting operations.

We can use any one of the above-mentioned methods for tracking. Usually, we use the deep sort method, which gives very good output compared to any other tracker, and also it gives better frames per second (FPS) compared to the centroid tracker, and the rest, but the major drawback of the deep sort method is that we need to fine-tune it for our custom object tracking application. Here we will use the centroid tracker in our project. Once we have a proper hold on the detected object throughout the frames, then we can perform the counting operation, object IN-OUT operation on that object. The Aim of this project is to develop a people counter device to count the number of pedestrians walking through a door or corridor through a video or camera.

Keywords- OpenCV, People Counting, Computer Vision, Background Maintenance, Segmentation.

TABLE OF CONTENTS

LIST OF CHAPTERS **PAGE NO**

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 OVERVIEW:

People counting have a wide range of applications in the context of pervasive systems. These applications range from efficient allocation of resources in smart buildings to handling emergency situations. There exist several vision-based algorithms for people counting. Each algorithm performs differently in terms of efficiency, flexibility and accuracy for different indoor scenarios. Hence, evaluating these algorithms with respect to different application scenarios, environment conditions and camera orientations will provide a better choice for actual deployment.

1.2 PURPOSE:

A people counting system provides tools to save money, gain valuable analytics, improve the visitor experience and optimize operations. Of all people counting methods, video-based people counting is the most accurate at over 98%. People counting systems protect employees and customers by limiting the chances of standing close to someone. With People Counting System you can track customer behaviour and use the data to improve customer service.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

Counting people on streets or at entrances of places is indeed beneficial for security, tracking, and marketing purposes. People counters can be used to monitor occupancy of entire buildings, individual rooms or anything. A people counting system provides tools to save money, gain valuable analytics, improve the visitor experience and optimize operations.

2.2. PROPOSED SOLUTION

The aim of this project is to develop a people counter device to count the number of pedestrians walking through a door or corridor through a video or camera. Most of the time, this system is used at the entrance of a building so that the total number of visitors can be recorded. Live stream of visitors flow is streamed on to a web application.

3. THEORTICAL ANALYSIS

3.1. TECHNICAL ARCHITECTURE

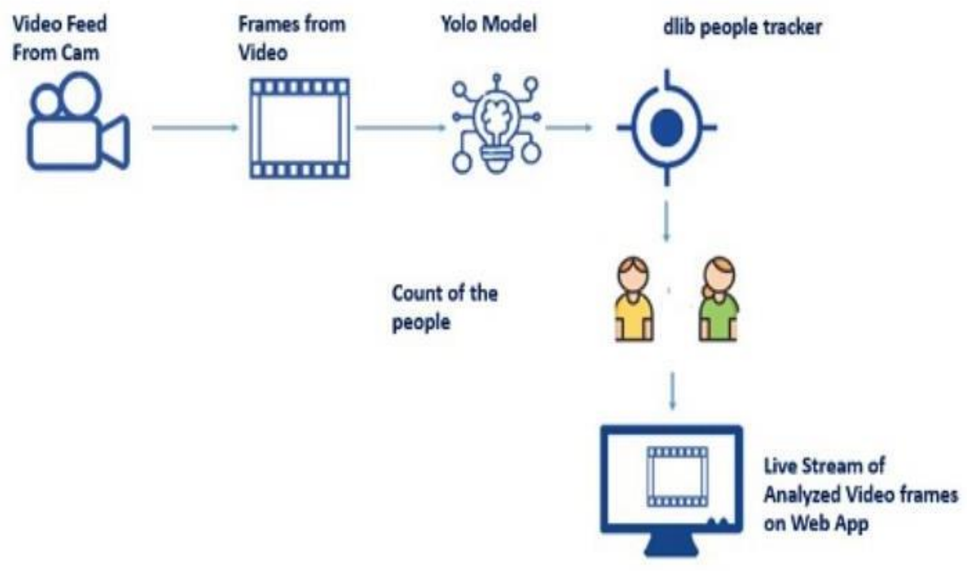


Figure 1: Architecture

3.2.HARDWARE/SOFTWARE DESIGNING

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the projectteam and user throughout the software development process. We use Centroid Tracking Algorithm here. Multiple processes are involved in the centroid tracking algorithm, we will go with each and every step in this activity.

User Requirements

1. Good Practice

For many projects, the total set of user requirements can be ambitious, making it difficult or even impossible to deliver a solution that meets all the requirements, in a way, that is robust, cost-effective, and maintainable and can be rolled out quickly to a large user base. It is important to match the user requirements specification against the available technology and solutions that can be implemented in a timely, robust and practical way. This may result in an agreement that some of the requirements say 20%, will not be delivered.

Such a compromise will make sure the remaining 80% can be delivered quickly. This compromise is important for global projects with a large user base. On such projects, the speed and ease ofimplementation is an important consideration in the overall solution.

To be successful at requirements gathering and to give your project an increased likelihood of success, follow these rules:

1. Don’t assume you know what the customer wants, ask!
2. Involve the users from the start.
3. Define and agree on the scope of the project.
4. Ensure requirements are specific, realistic and measurable.
5. Get clarity if there is any doubt.
6. Create a clear, concise and thorough requirements document and share it with the customer.
7. Confirm your understanding of the requirements with the customer by playing them back.
8. Avoid talking technology or solutions until the requirements are fully understood.
9. Get the requirements agreed with the stakeholders before the project starts.

2. COMMON MISTALKES

Basing a solution on complex or new technology and then discovering that it cannot easily be rolledout to the ‘real world.’

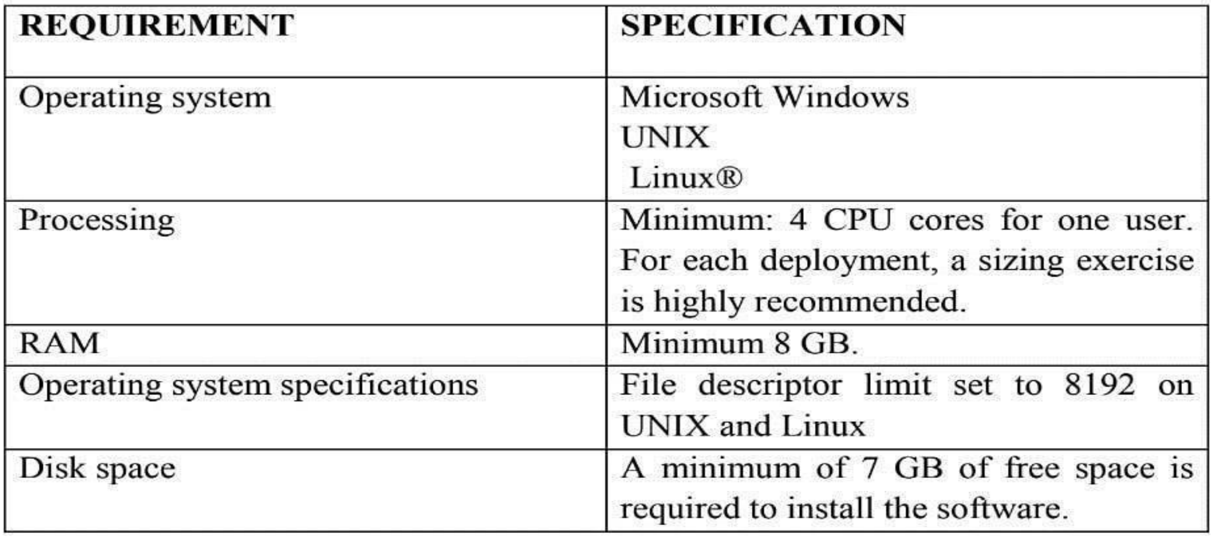
- Not prioritizing the User Requirements into ‘must have,’ ‘should have,’ ‘couldhave’ and ‘would have,’ known as the Moscow principle.
- Not enough consultation with real users and practitioners.
- Solving the ‘problem’ before you know what it is.

Lacking a clear understanding and making assumptions instead of asking for clarification.

SOFTWARE REQUIREMENTS

- Spyder IDE
- Python (pandas, NumPy)
- HTML
- Flask
- OpenCV
- Requests
- Windows OS
- MobideSSD detection model
- Object Tracking
- Object Detection

HARDWARE REQUIREMENTS



REQUIREMENT	SPECIFICATION
Operating system	Microsoft Windows UNIX Linux®
Processing	Minimum: 4 CPU cores for one user. For each deployment, a sizing exercise is highly recommended.
RAM	Minimum 8 GB.
Operating system specifications	File descriptor limit set to 8192 on UNIX and Linux
Disk space	A minimum of 7 GB of free space is required to install the software.

Table 1: Hardware Requirement

4: DESIGN

4.1. INTRODUCTION

The owner's knowledge of how many and when people are inside the building, company or shopping mall, help him to optimize the scheduling of labour and monitor the promotional events effectiveness. Security measure can benefit from people counting; this help to determine the numberof guards can be assigned.

There are several people counting technologies as:

1. Ultrasonic Sensor: there should be cluster of tree-node sensors; each node mounts an ultrasonic area. Wide area needs multiple clusters, coordinating node of cluster reads form nodes by RF link. Distributed algorithm of nodes decides counting process for detected people. Such system needs clock synchronization at millisecond level to exchange data simultaneously. Protocol of clock synchronization is imposing a disadvantage of this technology.
2. Infrared sensor array: a system requires IR sensors matrix which form the detectors. The matrix provides sensor signals. The algorithms of pattern recognition are detecting people moving across sensor area. Such system requires power processing and synchronization for detecting people.

3. Infrared Motion Sensors: three PIR sensors are required for each passage monitored. A coordinator connects to the sensors by a wireless RF link. The coordinator receives the events of detected motion from the sender. The coordinator concludes people count from correlating phase, number and time difference of signal peaks. PIR sensors are alternative to previous technology, however, the effort and cost of installing multiple nodes of sensor for each Surveillance area is a cost-side disadvantage.
4. Sensor Fusion: The system consists of PIR sensors, CO2 and camera. It depends on a Hidden Markovian Model that relays on a Filter of Extended Kalman to derive building occupancy.

This technology combines readings of current sensor and historical data to estimate the true stateof the system, adjusting for stochastic processes and sensor noise.

5. Video counter: it based on ceiling mounted camera; people are identified by background subtraction of image. The objects (blobs) are identified and their size estimated and compared to pixels dimension of people which established previously. This analysis leads to people count.

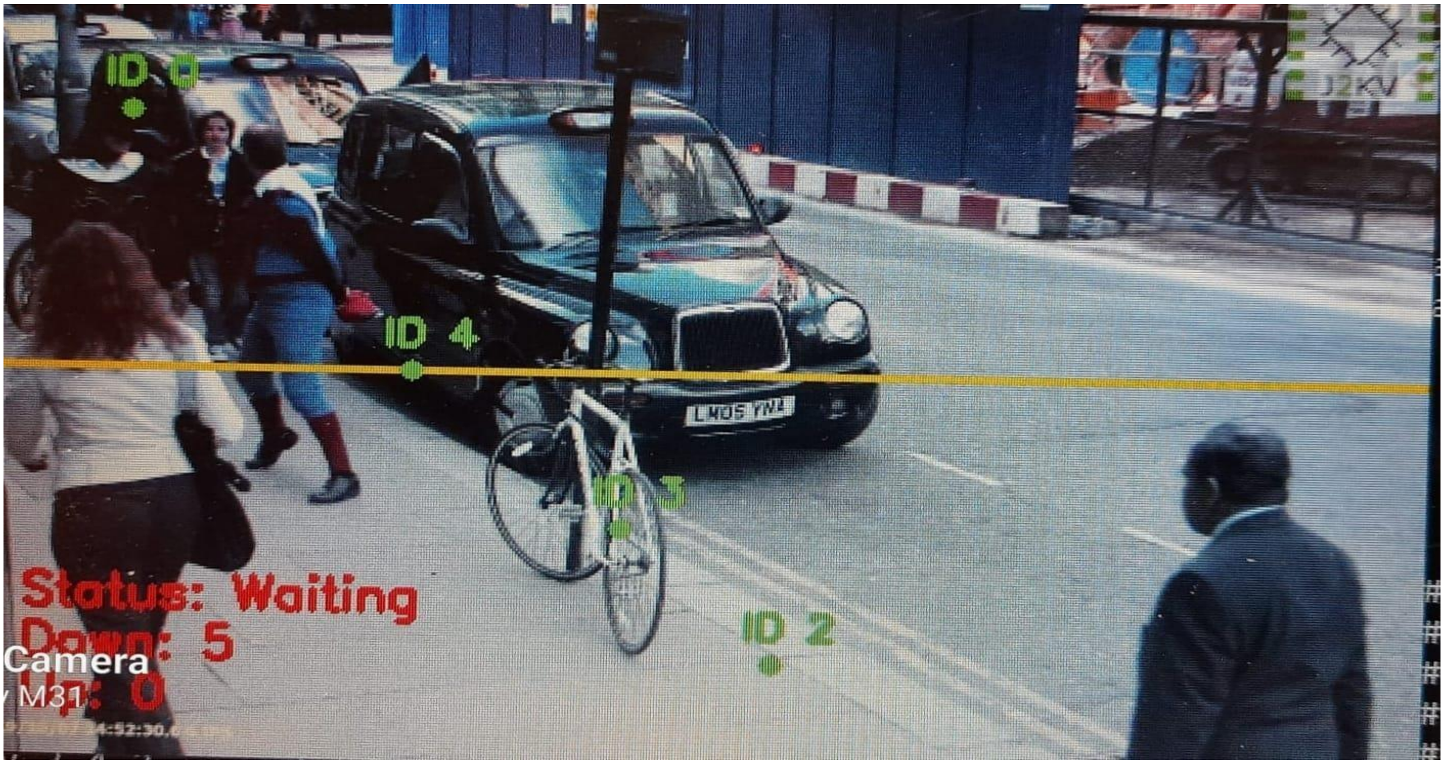


Figure 2: people counting

4.2. PROCEDURE

This project has two phases, phase 1 is to implement detection of person and the second phase is to implement tracking of a person

Phase1: Person Detection

1. Get the camera / Video Feed
2. Detect people in the frame using the caffe pre-trained model
3. Localize the pedestrians in the frame
4. use the detected person bounding box coordinates to track them

Phase 2: Tracking Phase:

1. create an object tracker for each detected object to track the object as it moves around the frame
2. continue tracking until the N-th frame is reached and then re-run our object detector The entire process repeats

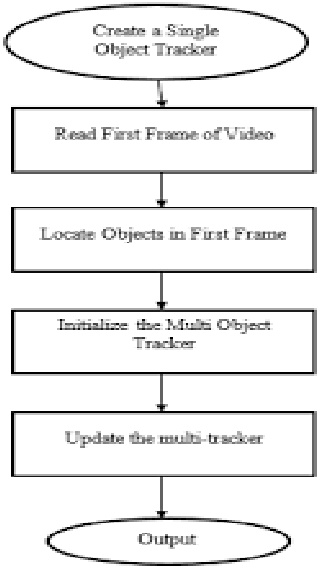

Create a Single Object Tracker
Read First Frame of Video
Locate Objects in First Frame
Initialize the Multi Object Tracker
Update the multi-tracker
Output

Figure 3: Flow chart

4.3. ALGORITHMS AND TECHNIQUES

Centroid Tracking Algorithm

The centroid tracker has the following steps:

- Accepts the bouding box coordinates and computes the centroid.
- The algorithm accepts the bounding box coordinates that are xmin, ymin, xmax, and ymax and the gives (x_center, y_center) coordinates for each of the detected objects in each frame.
- The Centroid is calculated is given below:

$X\_cen = ((xmin + xmax) // 2$

$Y\_cen = ((ymin + ymax) // 2)$

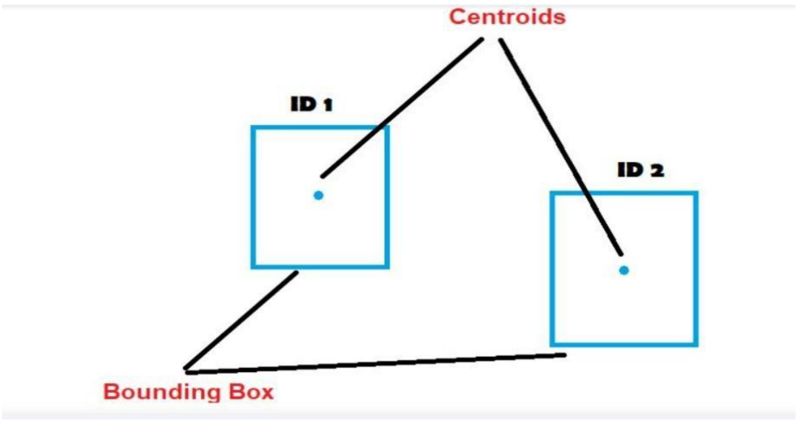


Figure 4: Centroid Calculation

where xmin, ymin, xmax, and ymax are the bounding box coordinates for object detection model (here YOLO v3).

- Then, it calculates the euclidian distance between the new detected bounding box and the existing object.

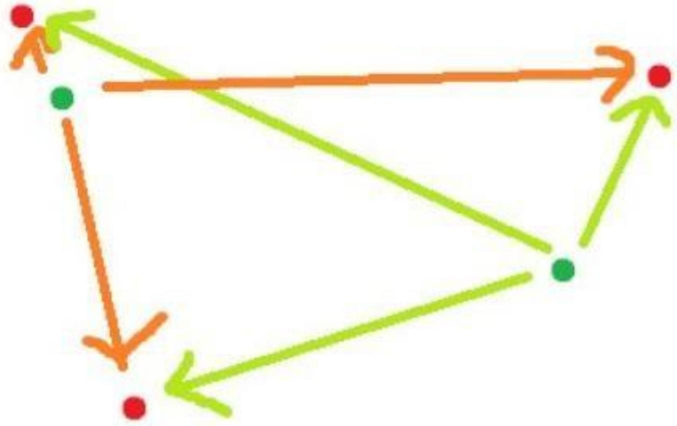


Figure 5: Euclidian Distance

- Update the centroid for the existing object. After calculating the euclidian distance between the detected bounding box and the existing bounding box, it will update the position of the centroid in the frame, there by tracking the object.

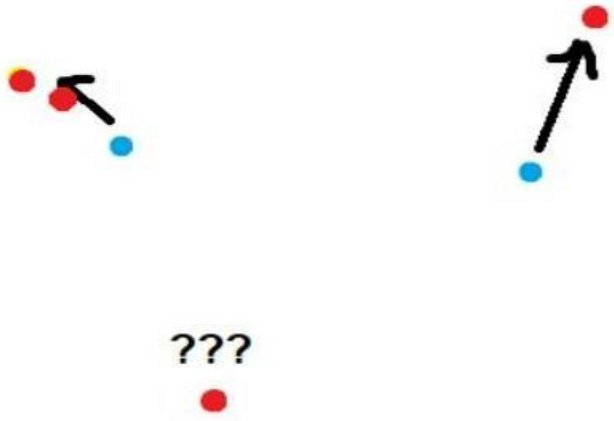


Figure 6: Updating Centroid

- Registering new objects. When a new object enters or the same object is been detected, the centroid tracker will register the new object with a unique ID, so that it becomes helpful for different applications.

Figure 7: Registering New Objects

- De-registering the previous objects. Once the object is not in the frame, the algorithm will de-register the object ID, stating that the object is not available or left the frame. This is the basic operation of centroid tracking. Here for the object detection model, we will use YOLO v3 model, for detecting a class person in the frame.

YOLO v3 model

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

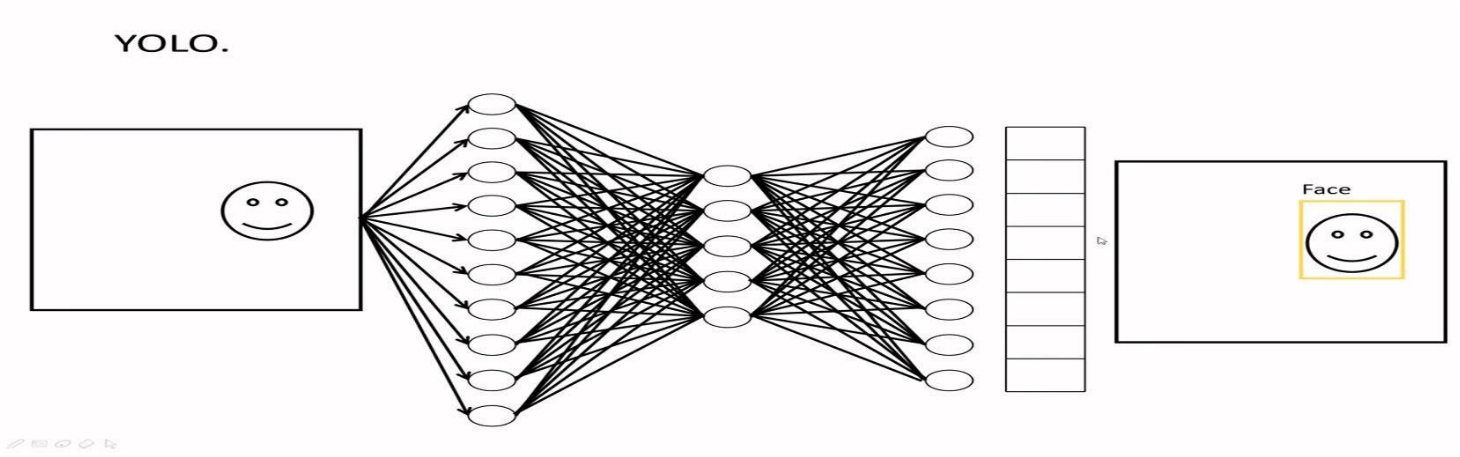


Figure 8: YOLO figure

5. CONCLUSION

In UG Project Phase-1, we have worked on problem statement, literature survey and also done the experimental analyses which are required for the project to move forward. We have discussed about the OpenCv and explained the algorithms to be used in the project. We also discussed about the flowcharts and use case diagrams which are used in the project. Based on the experimental analysis we have designed the model for the project. Entire designing part is involved in UG Project Phase-1

6. FUTURE SCOPE

UG Project Phase-2 is the extension of UG Project Phase-1. UG Project Phase-2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase-1. All the implementation is done and conclusions will be retrieved in the phase. We will also work on the applications, advantages, and disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase-2.

People Counting and Tracking System

A UG PROJECT PHASE-2 REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

GOWROJU LAXMI PRIYA	**18UK1A0518**
CHALLAKONDA SRICHANDANA	**18UK1A0567**
JUNNUTHULA AKSHAYA	**18UK1A0523**
MANDA MANASA	**18UK1A0531**

Under the guidance of

Ms. ARSHIYA FARHEEN

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

(Affiliated to JNTUH, HYDERABAD)

BOLLIKUNTA, WARANGAL (T.S) – 506005

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

BOLLIKUNTA, WARANGAL – 506005

2018-2022

CERTIFICATE OF COMPLETION

UG PROJECT PHASE-2

This is to certify that the UG Project Phase-2 entitled **“**People Counting and Tracking System**”** is being submitted by **GOWROJU LAXMI PRIYA (18UK1A0518), CHALLAKONDA SRICHANDANA (18UK1A0567), JUNNNUTHULA AKSHAYA (18UK1A0523), MANDA MANASA(18UK1A0531)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** to **Jawaharlal Nehru Technological University Hyderabad** during the academic year 2021- 2022, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
MS. ARSHIYA FARHEEN	**Dr R NAVEEN KUMAR**
(Assistant Professor)	(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved DR **P PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **DR R NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-2.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-2 and for their support in completing the UG Project Phase-2

We express heartfelt thanks to the guide, **Ms. ARSHIYA FARHEEN**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion this UG Project Phase-2.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

GOWROJU LAXMI PRIYA	**18UK1A0518**
CHALLAKONDA SRICHANDANA	**18UK1A0567**
JUNNUTHULA AKSHAYA	**18UK1A0523**
MANDA MANASA	**18UK1A0531**

TABLE OF CONTENTS

LIST OF CHAPTERS **PAGE NO**

1. INTRODUCTION

People counting have a wide range of applications in the context of pervasive systems. These applications range from efficient allocation of resources in smart buildings to handling emergency situations. There exist several vision-based algorithms for people counting. Each algorithm performs differently in terms of efficiency, flexibility and accuracy for different indoor scenarios. Hence, evaluating these algorithms with respect to different application scenarios, environment conditions and camera orientations will provide a better choice for actual deployment.

Object tracking techniques use methods like deep sort, centroid tracker, csrt, kcf, and camshift which track the detected object by comparing the similarity of detected objects with each other in each processed frame. If the object has the same similarity metric throughout the frame then it will track the same object throughout the sequence of frames and retain the same object ID for that object. This constant object ID for a particular object makes it easier for us to do the counting operations.

We can use any one of the above-mentioned methods for tracking. Usually, we use the deep sort method, which gives very good output compared to any other tracker, and also it gives better frames per second (FPS) compared to the centroid tracker, and the rest, but the major drawback of the deep sort method is that we need tofine-tune it for our custom object tracking application. Here we will use the centroid tracker in our project. Once we have a proper hold on the detected object throughout the frames, then we can perform the counting operation, object IN-OUT operation on that object. The Aim of this project is to develop a people counter device to count the number of pedestrians walking through a door or corridor through a video or camera

UG Project Phase-2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase-1. All the implementation is done and conclusions are retrieved in this phase. We will also work on the applications, advantages, and disadvantages of the project in this phase. Future scopeof the project will be also discussed in the UG Project Phase-2.

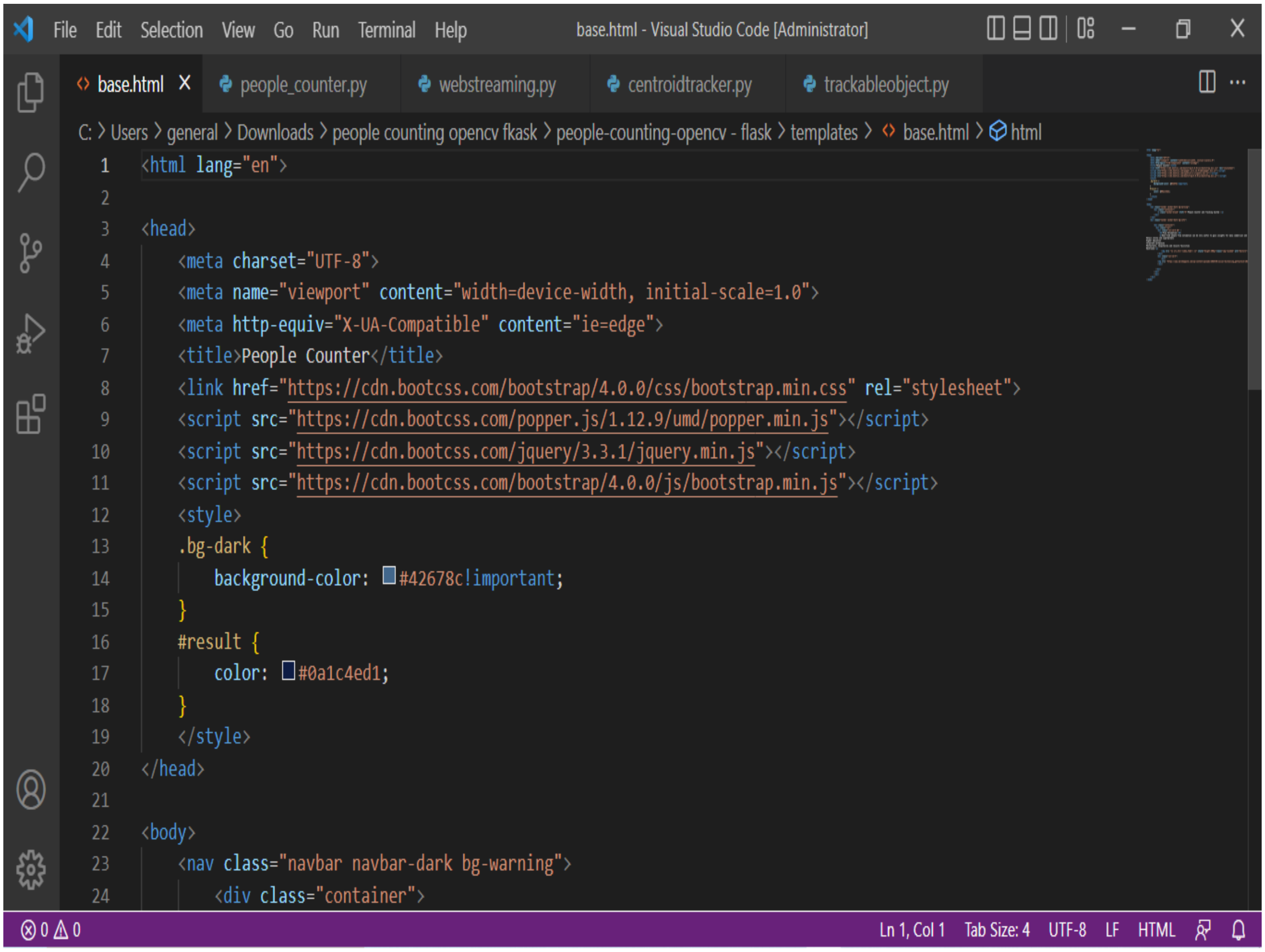
2. CODE SNIPPETS

2.1 MODEL CODE

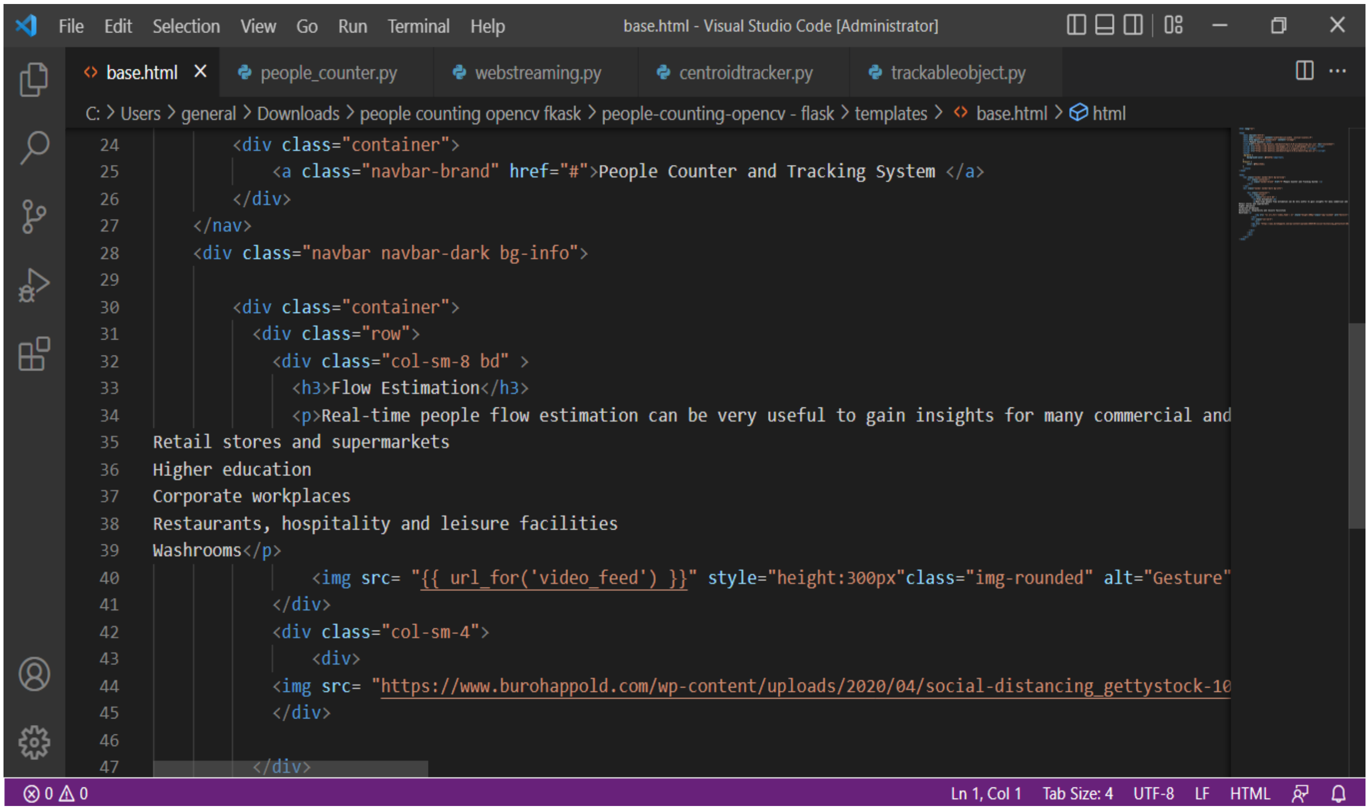
https://drive.google.com/drive/folders/1UBD9xgRTfXcdjJUWEU1KiCl0yAVQXeAu?usp=sharing

2.2 HTML CODE AND PYTHON CODE

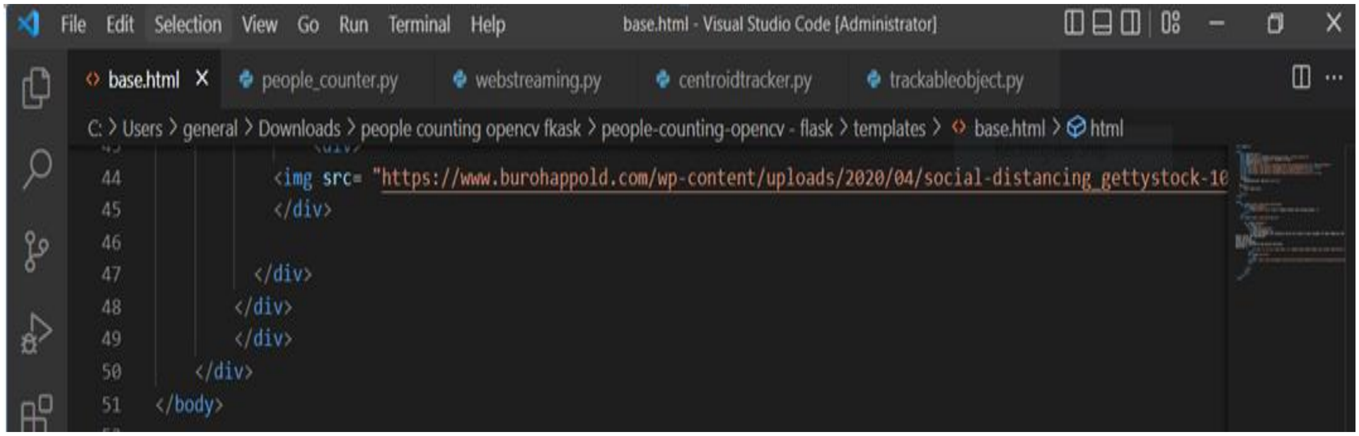
1. base.html


File Edit Selection View Go Run Terminal Help
base.html - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > templates > base.html > html
Ln 1, Col 1
Tab Size: 4
UTF-8
LF
HTML

```html
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>People Counter</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <style>
    .bg-dark {
        background-color: #42678c!important;
    }
    #result {
        color: #0a1c4ed1;
    }
    </style>
</head>

<body>
    <nav class="navbar navbar-dark bg-warning">
        <div class="container">
```

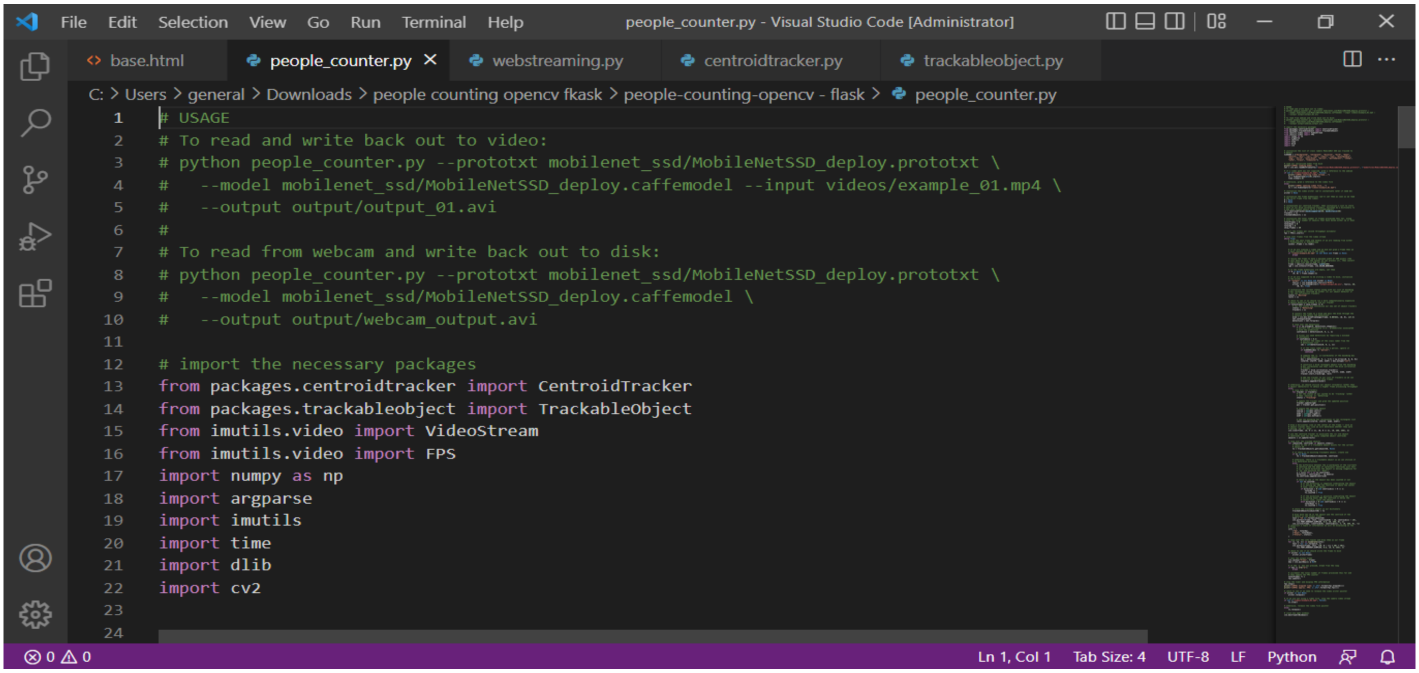

File Edit Selection View Go Run Terminal Help
base.html - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > templates > base.html > html
⊗ 0 ⚠ 0
Ln 1, Col 1
Tab Size: 4
UTF-8
LF
HTML

```
        <div class="container">
            <a class="navbar-brand" href="#">People Counter and Tracking System </a>
        </div>
    </nav>
    <div class="navbar navbar-dark bg-info">

        <div class="container">
          <div class="row">
            <div class="col-sm-8 bd" >
              <h3>Flow Estimation</h3>
              <p>Real-time people flow estimation can be very useful to gain insights for many commercial and
Retail stores and supermarkets
Higher education
Corporate workplaces
Restaurants, hospitality and leisure facilities
Washrooms</p>
                <img src= "{{ url_for('video_feed') }}" style="height:300px"class="img-rounded" alt="Gesture"
            </div>
            <div class="col-sm-4">
                <div>
            <img src= "https://www.burohappold.com/wp-content/uploads/2020/04/social-distancing_gettystock-10
            </div>

          </div>
```

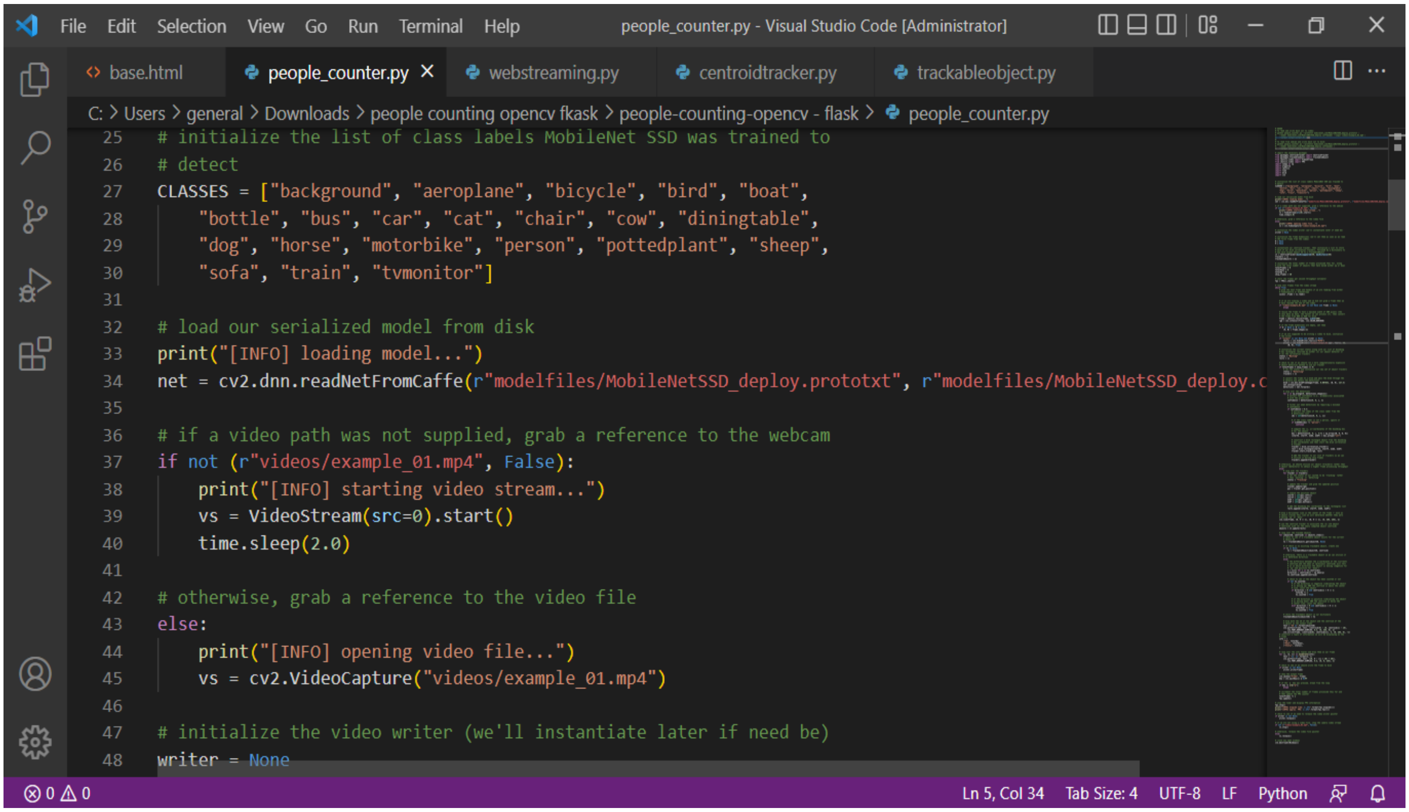

File Edit Selection View Go Run Terminal Help
base.html - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > templates > base.html > html

```
            <img src= "https://www.burohappold.com/wp-content/uploads/2020/04/social-distancing_gettystock-10
            </div>

          </div>
        </div>
        </div>
    </div>
</body>
```

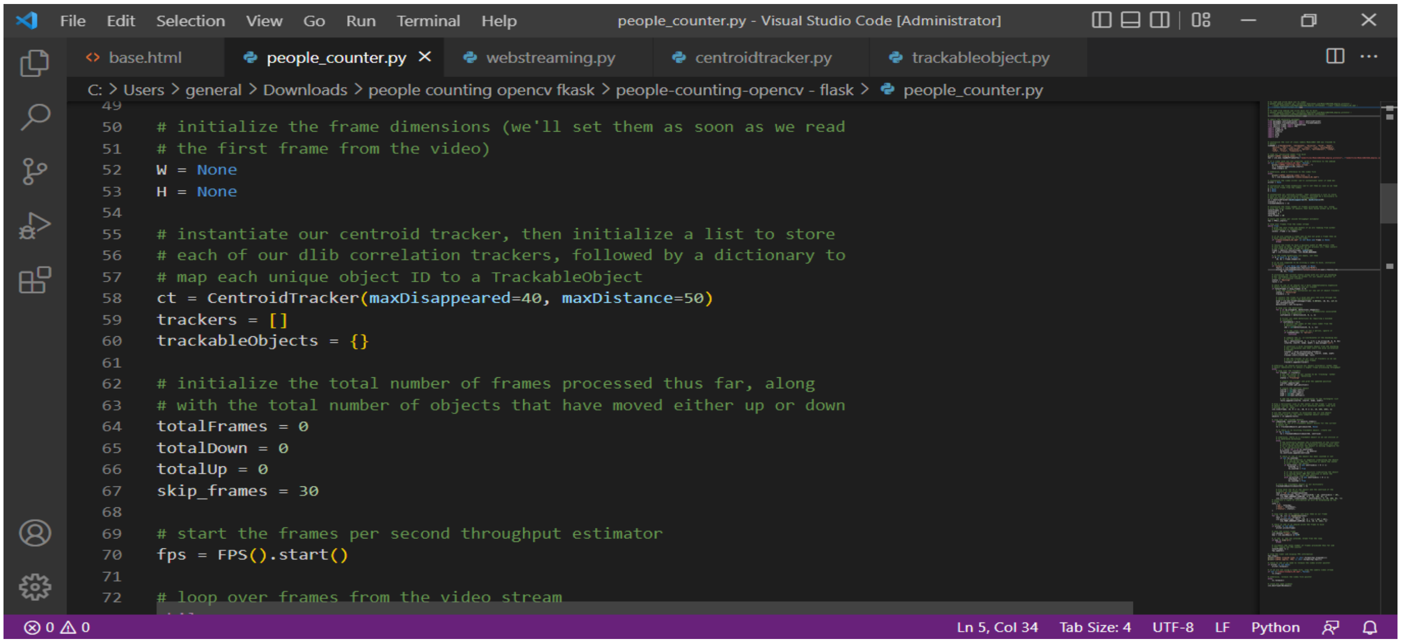
2. people_counter.py


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
0 0
Ln 1, Col 1 Tab Size: 4 UTF-8 LF Python

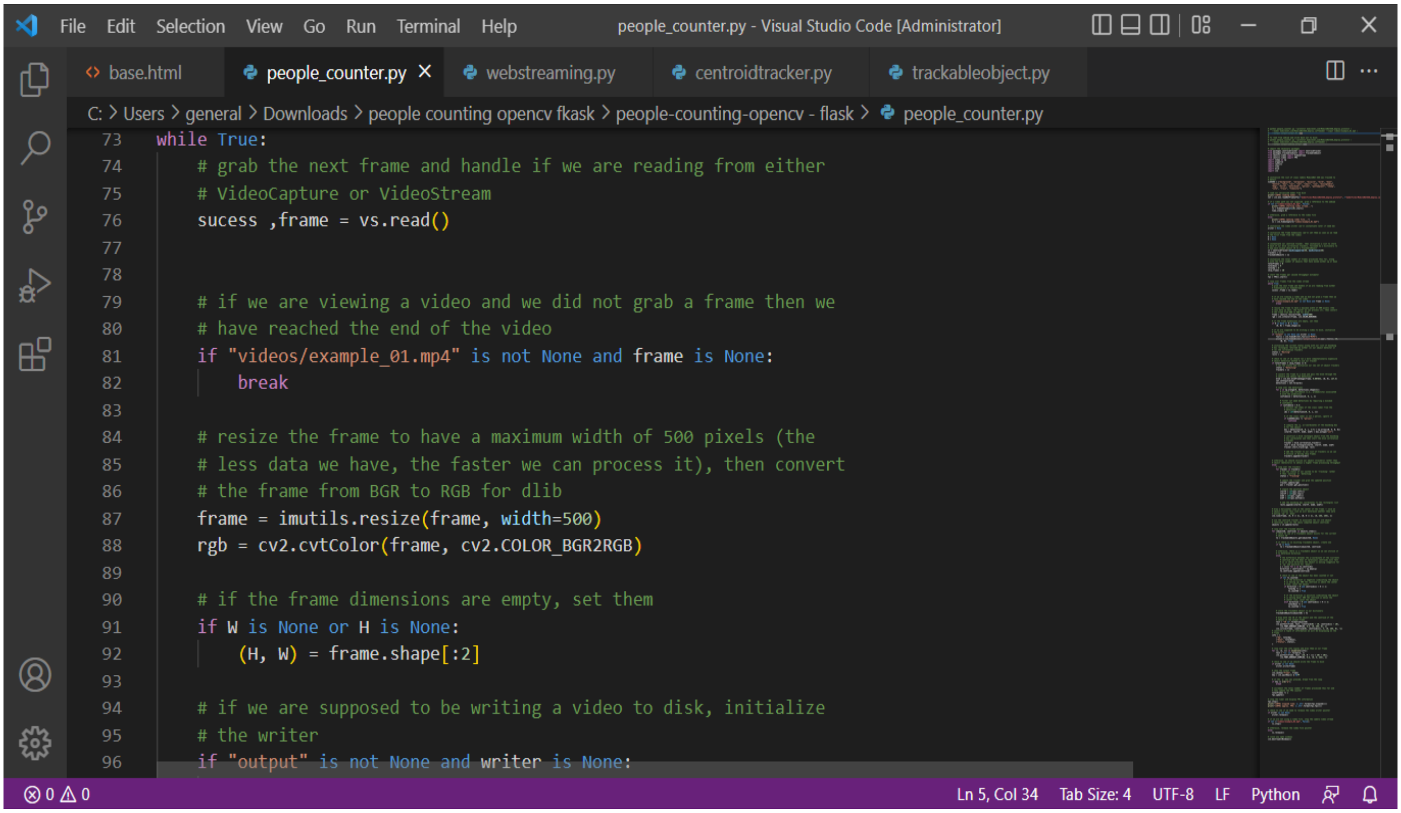
```python
# USAGE
# To read and write back out to video:
# python people_counter.py --prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt \
#   --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel --input videos/example_01.mp4 \
#   --output output/output_01.avi
#
# To read from webcam and write back out to disk:
# python people_counter.py --prototxt mobilenet_ssd/MobileNetSSD_deploy.prototxt \
#   --model mobilenet_ssd/MobileNetSSD_deploy.caffemodel \
#   --output output/webcam_output.avi

# import the necessary packages
from packages.centroidtracker import CentroidTracker
from packages.trackableobject import TrackableObject
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2

```


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
0 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

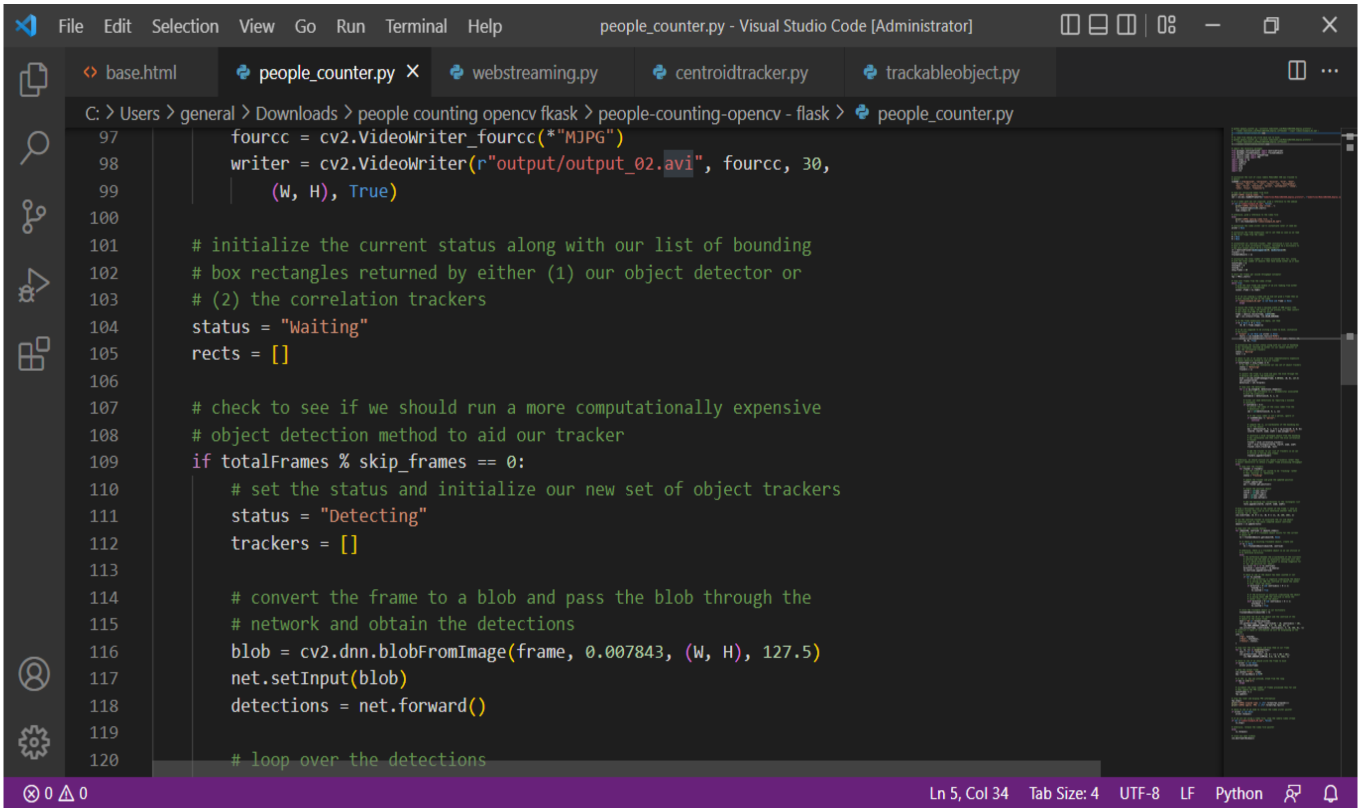
```python
# initialize the list of class labels MobileNet SSD was trained to
# detect
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(r"modelfiles/MobileNetSSD_deploy.prototxt", r"modelfiles/MobileNetSSD_deploy.c

# if a video path was not supplied, grab a reference to the webcam
if not (r"videos/example_01.mp4", False):
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(2.0)

# otherwise, grab a reference to the video file
else:
    print("[INFO] opening video file...")
    vs = cv2.VideoCapture("videos/example_01.mp4")

# initialize the video writer (we'll instantiate later if need be)
writer = None
```


File
Edit
Selection
View
Go
Run
Terminal
Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34
Tab Size: 4
UTF-8
LF
Python

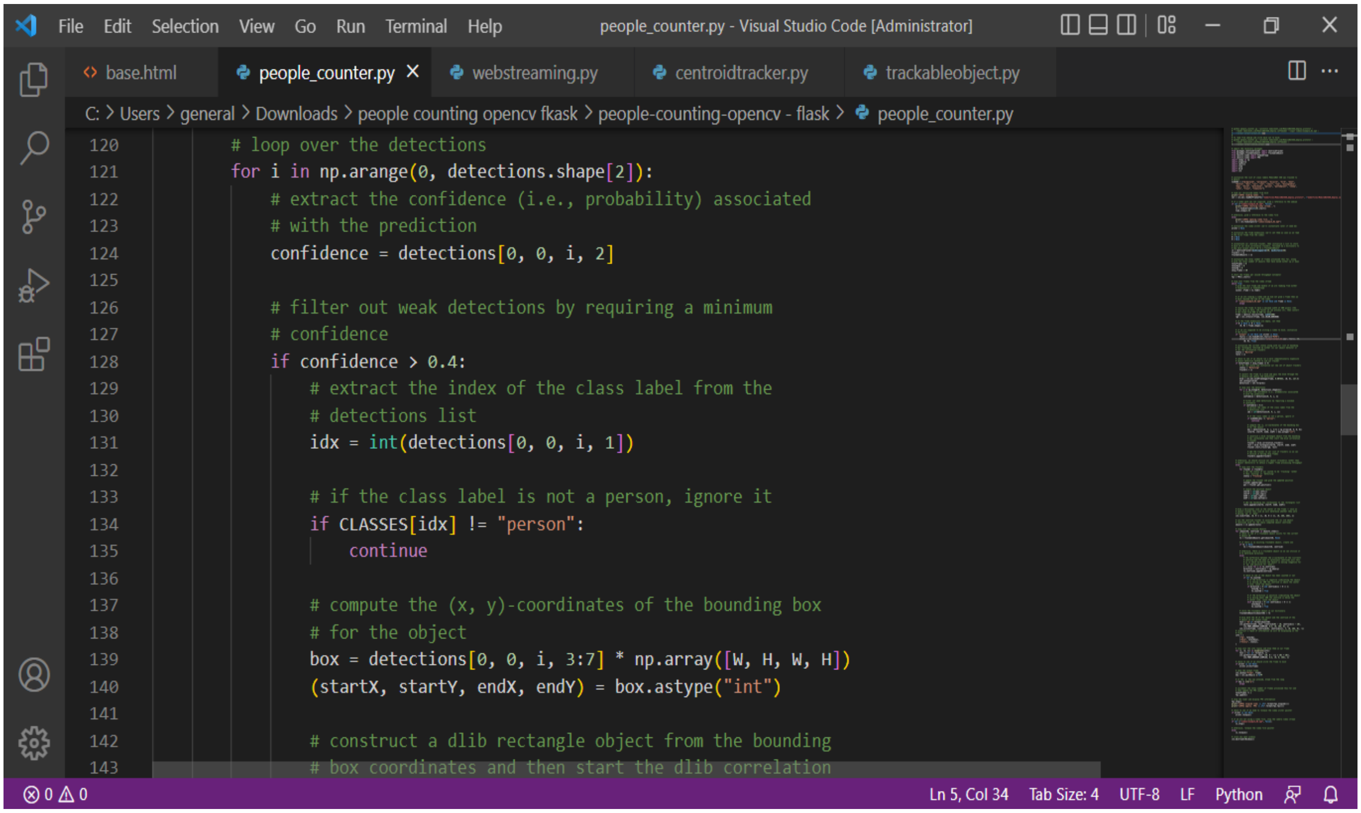
```
# initialize the frame dimensions (we'll set them as soon as we read
# the first frame from the video)
W = None
H = None

# instantiate our centroid tracker, then initialize a list to store
# each of our dlib correlation trackers, followed by a dictionary to
# map each unique object ID to a TrackableObject
ct = CentroidTracker(maxDisappeared=40, maxDistance=50)
trackers = []
trackableObjects = {}

# initialize the total number of frames processed thus far, along
# with the total number of objects that have moved either up or down
totalFrames = 0
totalDown = 0
totalUp = 0
skip_frames = 30

# start the frames per second throughput estimator
fps = FPS().start()

# loop over frames from the video stream
```


File
Edit
Selection
View
Go
Run
Terminal
Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34
Tab Size: 4
UTF-8
LF
Python

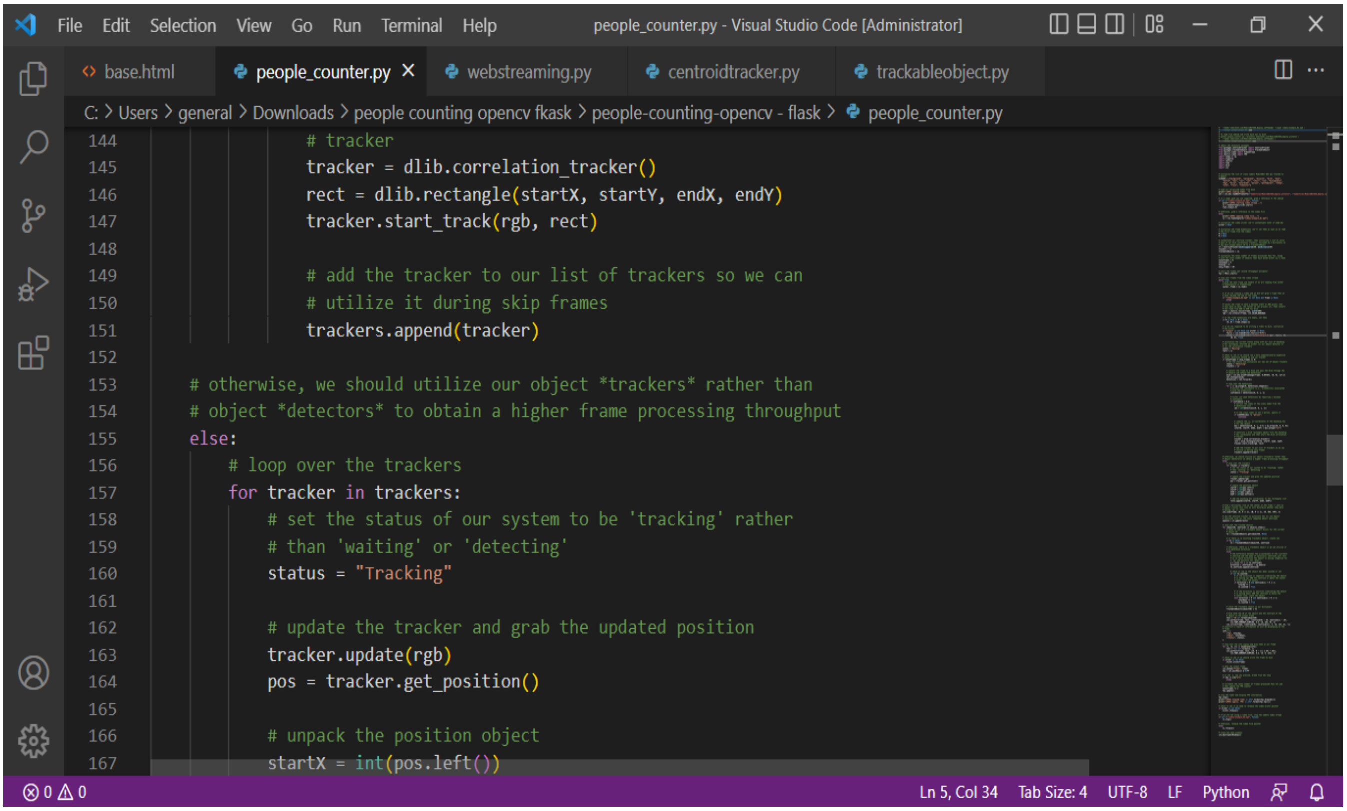
```
while True:
    # grab the next frame and handle if we are reading from either
    # VideoCapture or VideoStream
    sucess ,frame = vs.read()


    # if we are viewing a video and we did not grab a frame then we
    # have reached the end of the video
    if "videos/example_01.mp4" is not None and frame is None:
        break

    # resize the frame to have a maximum width of 500 pixels (the
    # less data we have, the faster we can process it), then convert
    # the frame from BGR to RGB for dlib
    frame = imutils.resize(frame, width=500)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # if the frame dimensions are empty, set them
    if W is None or H is None:
        (H, W) = frame.shape[:2]

    # if we are supposed to be writing a video to disk, initialize
    # the writer
    if "output" is not None and writer is None:
```


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

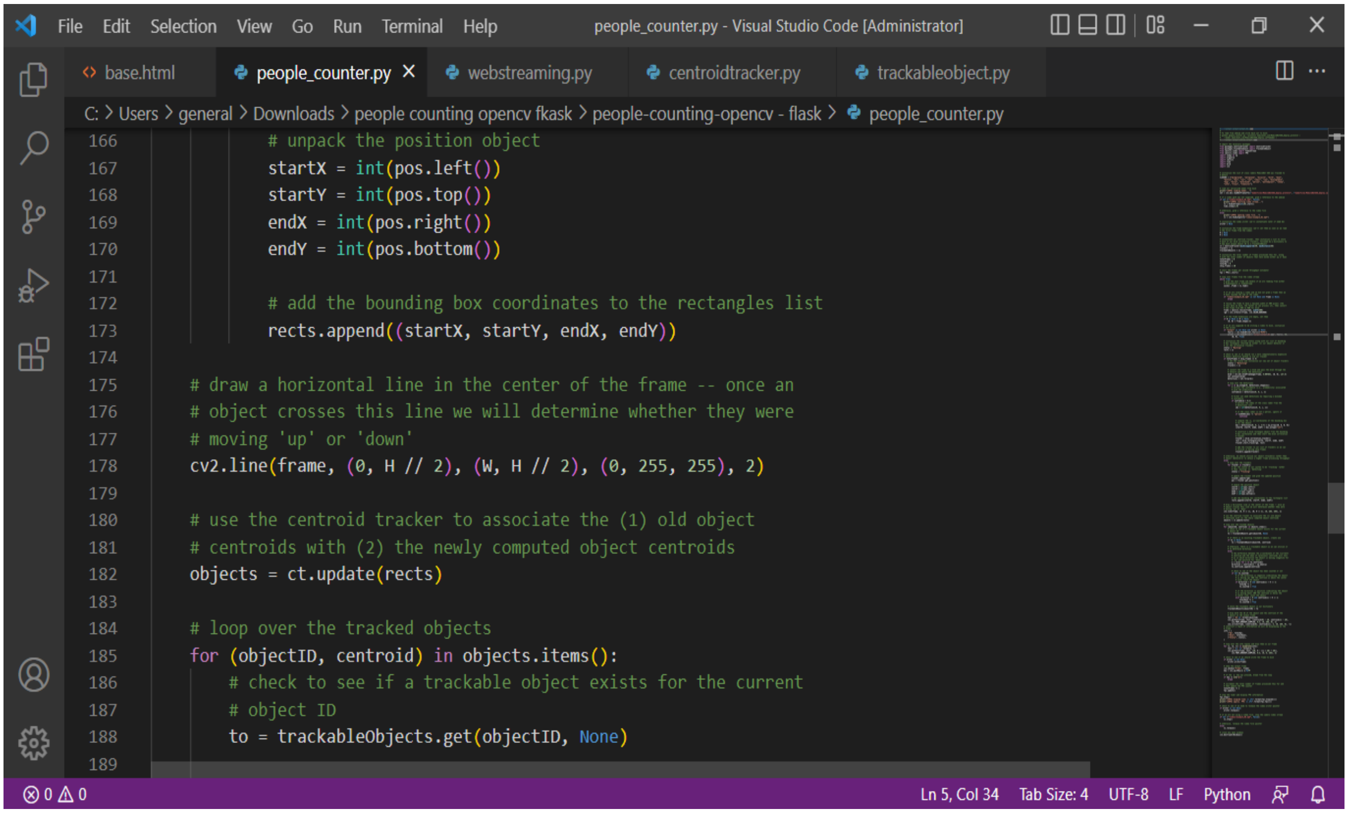
```python
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(r"output/output_02.avi", fourcc, 30,
            (W, H), True)

    # initialize the current status along with our list of bounding
    # box rectangles returned by either (1) our object detector or
    # (2) the correlation trackers
    status = "Waiting"
    rects = []

    # check to see if we should run a more computationally expensive
    # object detection method to aid our tracker
    if totalFrames % skip_frames == 0:
        # set the status and initialize our new set of object trackers
        status = "Detecting"
        trackers = []

        # convert the frame to a blob and pass the blob through the
        # network and obtain the detections
        blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
        net.setInput(blob)
        detections = net.forward()

        # loop over the detections
```


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

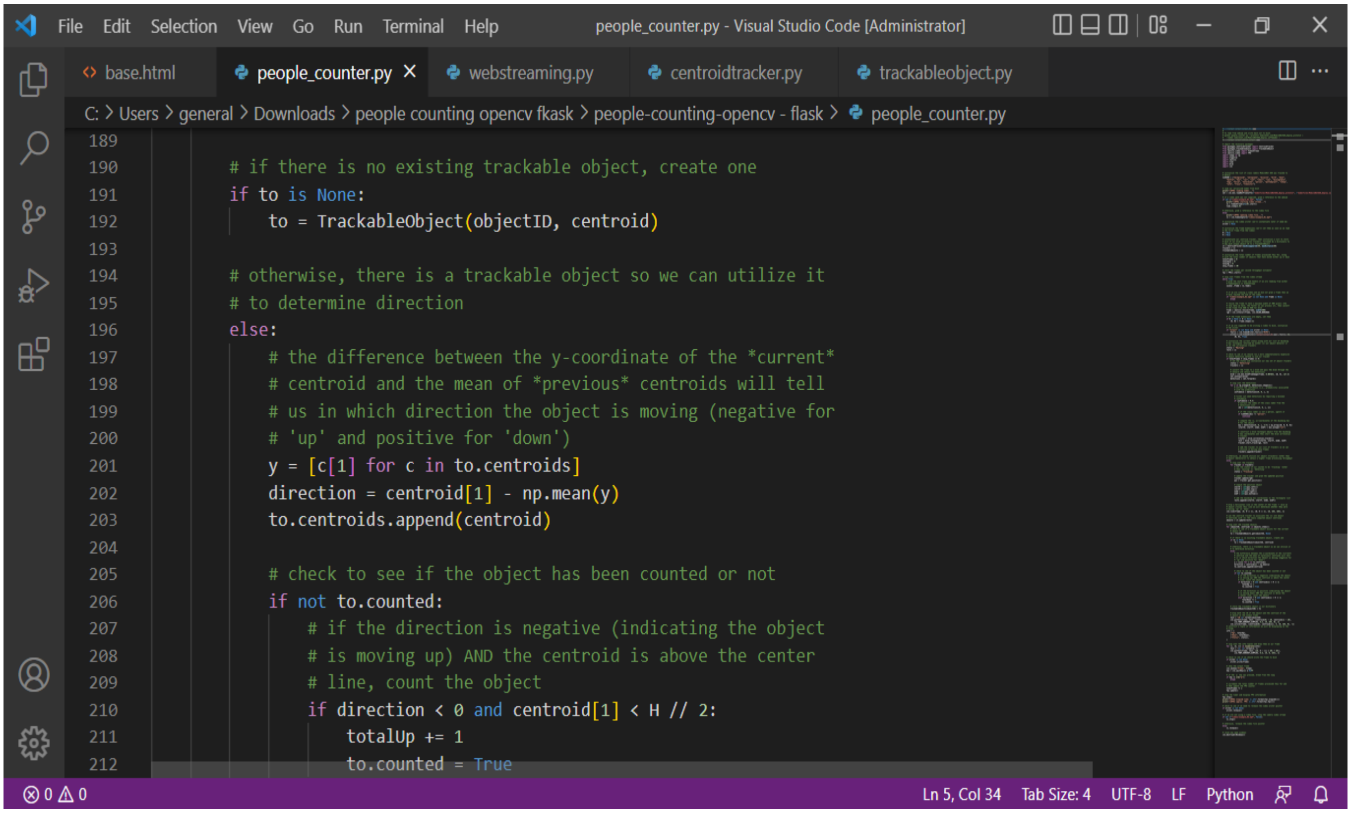
```python
        # loop over the detections
        for i in np.arange(0, detections.shape[2]):
            # extract the confidence (i.e., probability) associated
            # with the prediction
            confidence = detections[0, 0, i, 2]

            # filter out weak detections by requiring a minimum
            # confidence
            if confidence > 0.4:
                # extract the index of the class label from the
                # detections list
                idx = int(detections[0, 0, i, 1])

                # if the class label is not a person, ignore it
                if CLASSES[idx] != "person":
                    continue

                # compute the (x, y)-coordinates of the bounding box
                # for the object
                box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
                (startX, startY, endX, endY) = box.astype("int")

                # construct a dlib rectangle object from the bounding
                # box coordinates and then start the dlib correlation
```


File
Edit
Selection
View
Go
Run
Terminal
Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34
Tab Size: 4
UTF-8
LF
Python

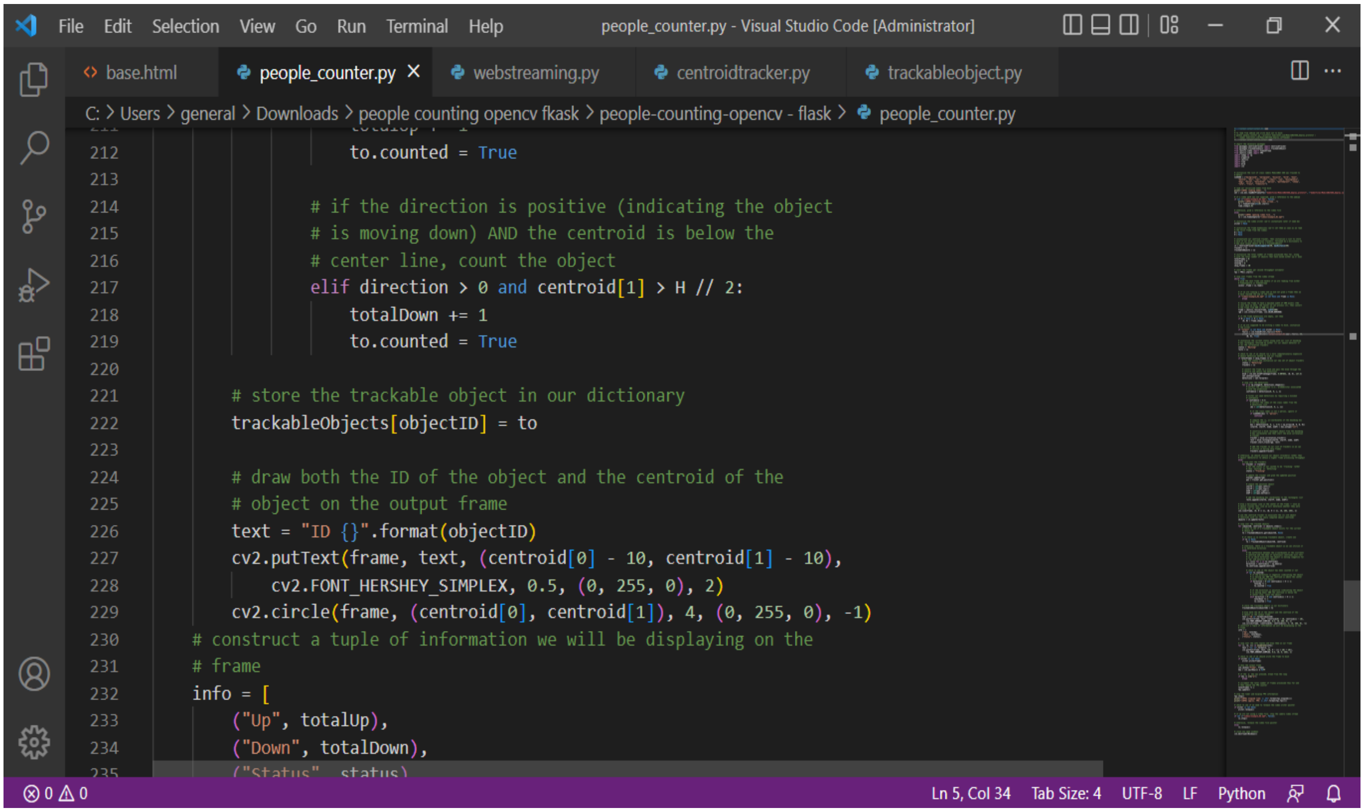
```python
                    # tracker
                    tracker = dlib.correlation_tracker()
                    rect = dlib.rectangle(startX, startY, endX, endY)
                    tracker.start_track(rgb, rect)

                    # add the tracker to our list of trackers so we can
                    # utilize it during skip frames
                    trackers.append(tracker)

        # otherwise, we should utilize our object *trackers* rather than
        # object *detectors* to obtain a higher frame processing throughput
        else:
            # loop over the trackers
            for tracker in trackers:
                # set the status of our system to be 'tracking' rather
                # than 'waiting' or 'detecting'
                status = "Tracking"

                # update the tracker and grab the updated position
                tracker.update(rgb)
                pos = tracker.get_position()

                # unpack the position object
                startX = int(pos.left())
```


File
Edit
Selection
View
Go
Run
Terminal
Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34
Tab Size: 4
UTF-8
LF
Python

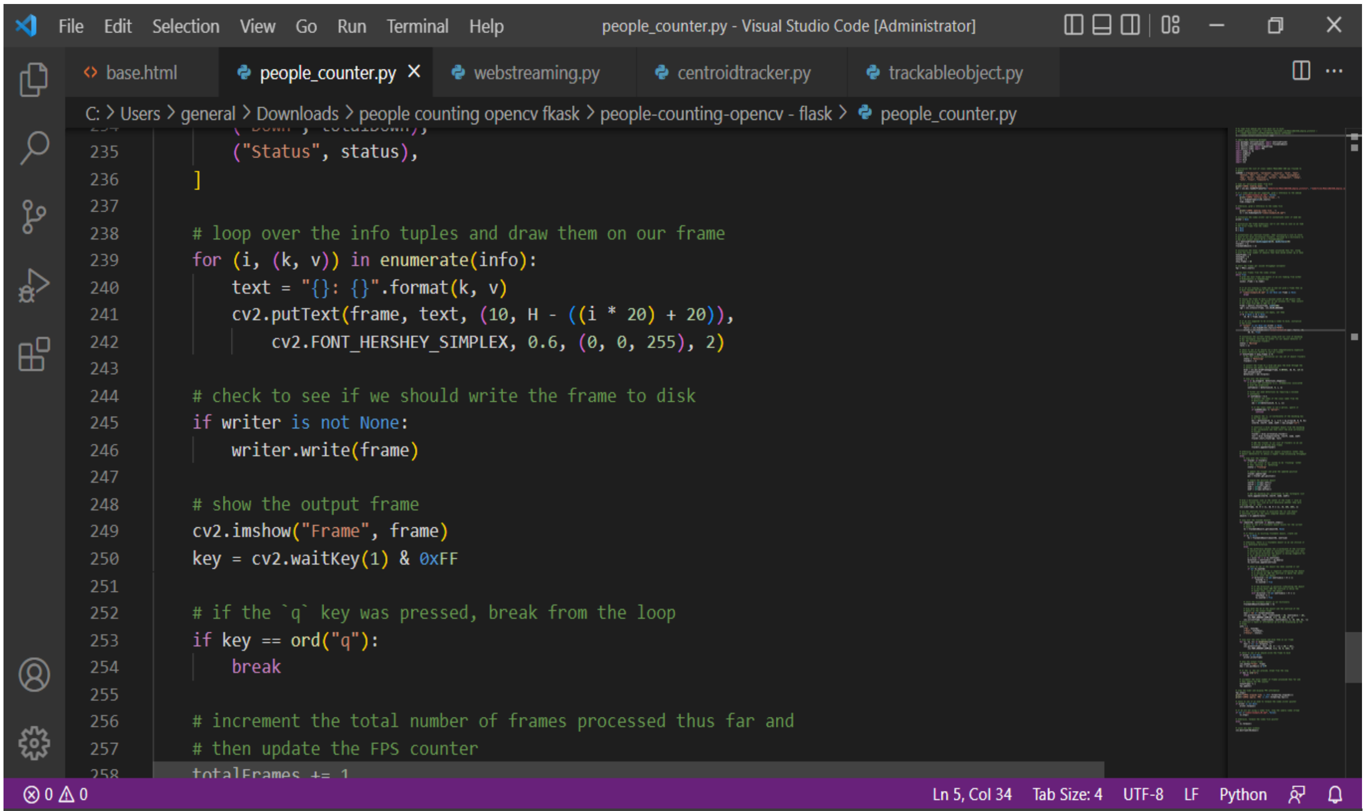
```python
                # unpack the position object
                startX = int(pos.left())
                startY = int(pos.top())
                endX = int(pos.right())
                endY = int(pos.bottom())

                # add the bounding box coordinates to the rectangles list
                rects.append((startX, startY, endX, endY))

        # draw a horizontal line in the center of the frame -- once an
        # object crosses this line we will determine whether they were
        # moving 'up' or 'down'
        cv2.line(frame, (0, H // 2), (W, H // 2), (0, 255, 255), 2)

        # use the centroid tracker to associate the (1) old object
        # centroids with (2) the newly computed object centroids
        objects = ct.update(rects)

        # loop over the tracked objects
        for (objectID, centroid) in objects.items():
            # check to see if a trackable object exists for the current
            # object ID
            to = trackableObjects.get(objectID, None)

```


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

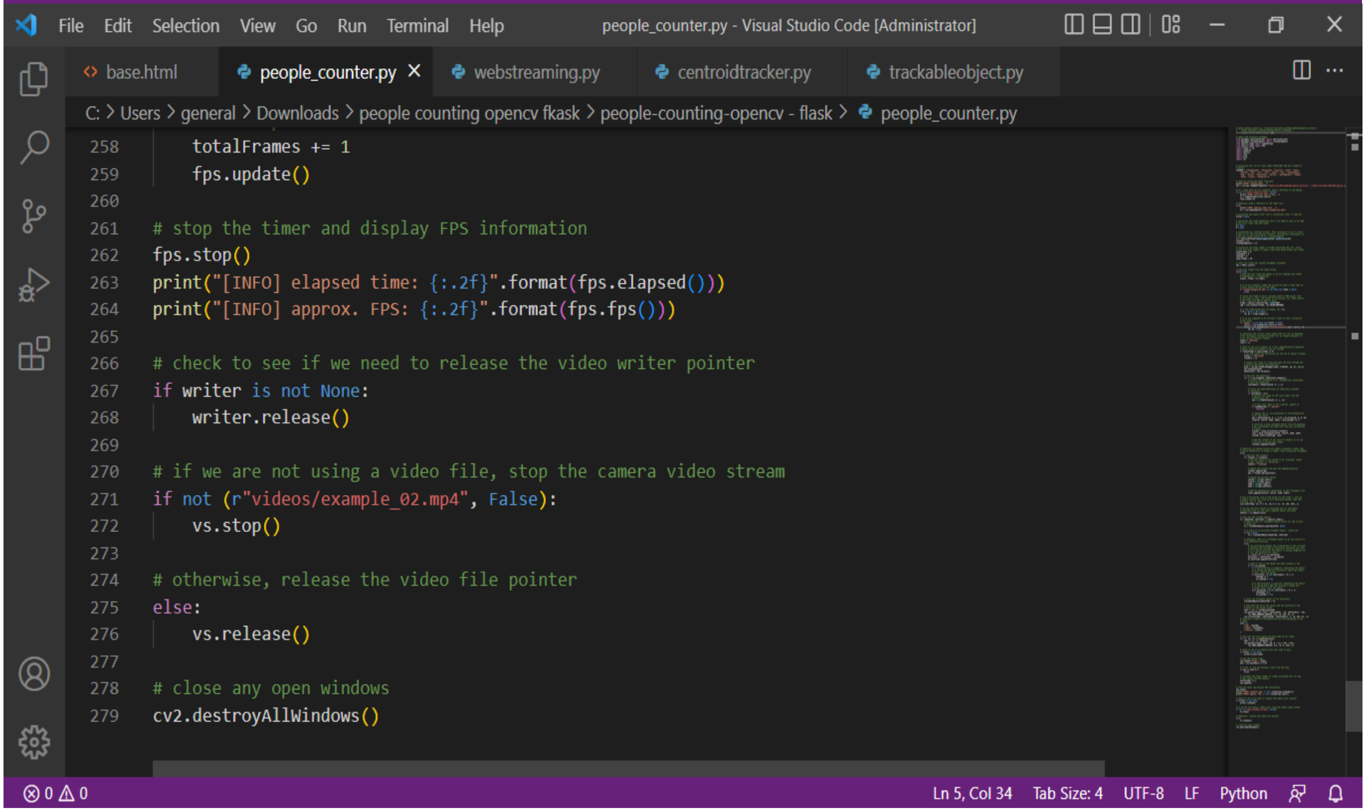
```

            # if there is no existing trackable object, create one
            if to is None:
                to = TrackableObject(objectID, centroid)

            # otherwise, there is a trackable object so we can utilize it
            # to determine direction
            else:
                # the difference between the y-coordinate of the *current*
                # centroid and the mean of *previous* centroids will tell
                # us in which direction the object is moving (negative for
                # 'up' and positive for 'down')
                y = [c[1] for c in to.centroids]
                direction = centroid[1] - np.mean(y)
                to.centroids.append(centroid)

                # check to see if the object has been counted or not
                if not to.counted:
                    # if the direction is negative (indicating the object
                    # is moving up) AND the centroid is above the center
                    # line, count the object
                    if direction < 0 and centroid[1] < H // 2:
                        totalUp += 1
                        to.counted = True
```


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

```
                        to.counted = True

                    # if the direction is positive (indicating the object
                    # is moving down) AND the centroid is below the
                    # center line, count the object
                    elif direction > 0 and centroid[1] > H // 2:
                        totalDown += 1
                        to.counted = True

            # store the trackable object in our dictionary
            trackableObjects[objectID] = to

            # draw both the ID of the object and the centroid of the
            # object on the output frame
            text = "ID {}".format(objectID)
            cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)
        # construct a tuple of information we will be displaying on the
        # frame
        info = [
            ("Up", totalUp),
            ("Down", totalDown),
            ("Status", status)
```


File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

```
        ("Down", totalDown),
        ("Status", status),
    ]

    # loop over the info tuples and draw them on our frame
    for (i, (k, v)) in enumerate(info):
        text = "{}: {}".format(k, v)
        cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

    # check to see if we should write the frame to disk
    if writer is not None:
        writer.write(frame)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

    # increment the total number of frames processed thus far and
    # then update the FPS counter
    totalFrames += 1
```

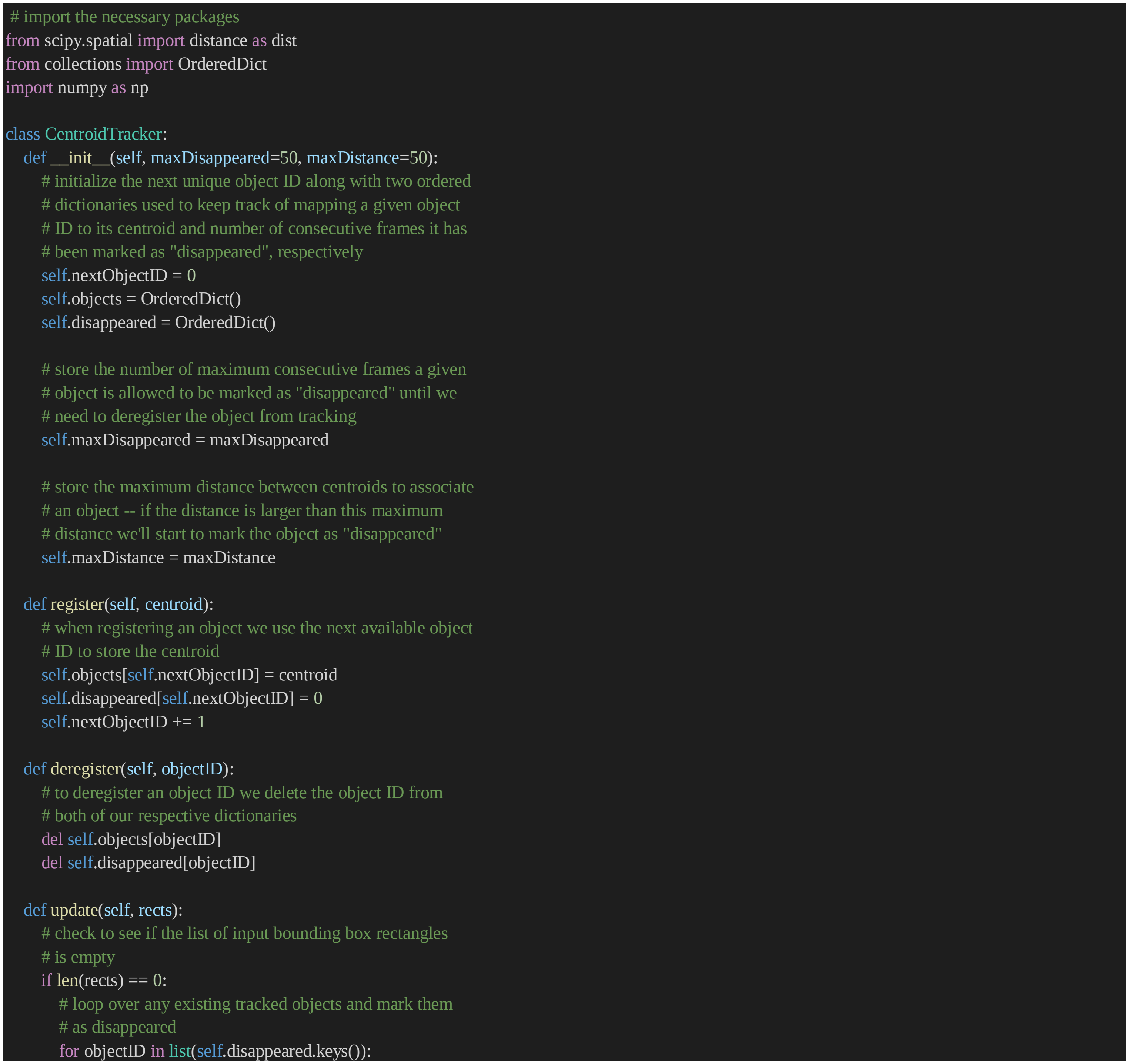

File Edit Selection View Go Run Terminal Help
people_counter.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > people_counter.py
⊗ 0 ⚠ 0
Ln 5, Col 34 Tab Size: 4 UTF-8 LF Python

```
    totalFrames += 1
    fps.update()

# stop the timer and display FPS information
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# check to see if we need to release the video writer pointer
if writer is not None:
    writer.release()

# if we are not using a video file, stop the camera video stream
if not (r"videos/example_02.mp4", False):
    vs.stop()

# otherwise, release the video file pointer
else:
    vs.release()

# close any open windows
cv2.destroyAllWindows()
```

3. webstreaming.py

https://drive.google.com/file/d/12wqUaZiVKFA_37-cfbehB13N092EhfTy/view?usp=sharing

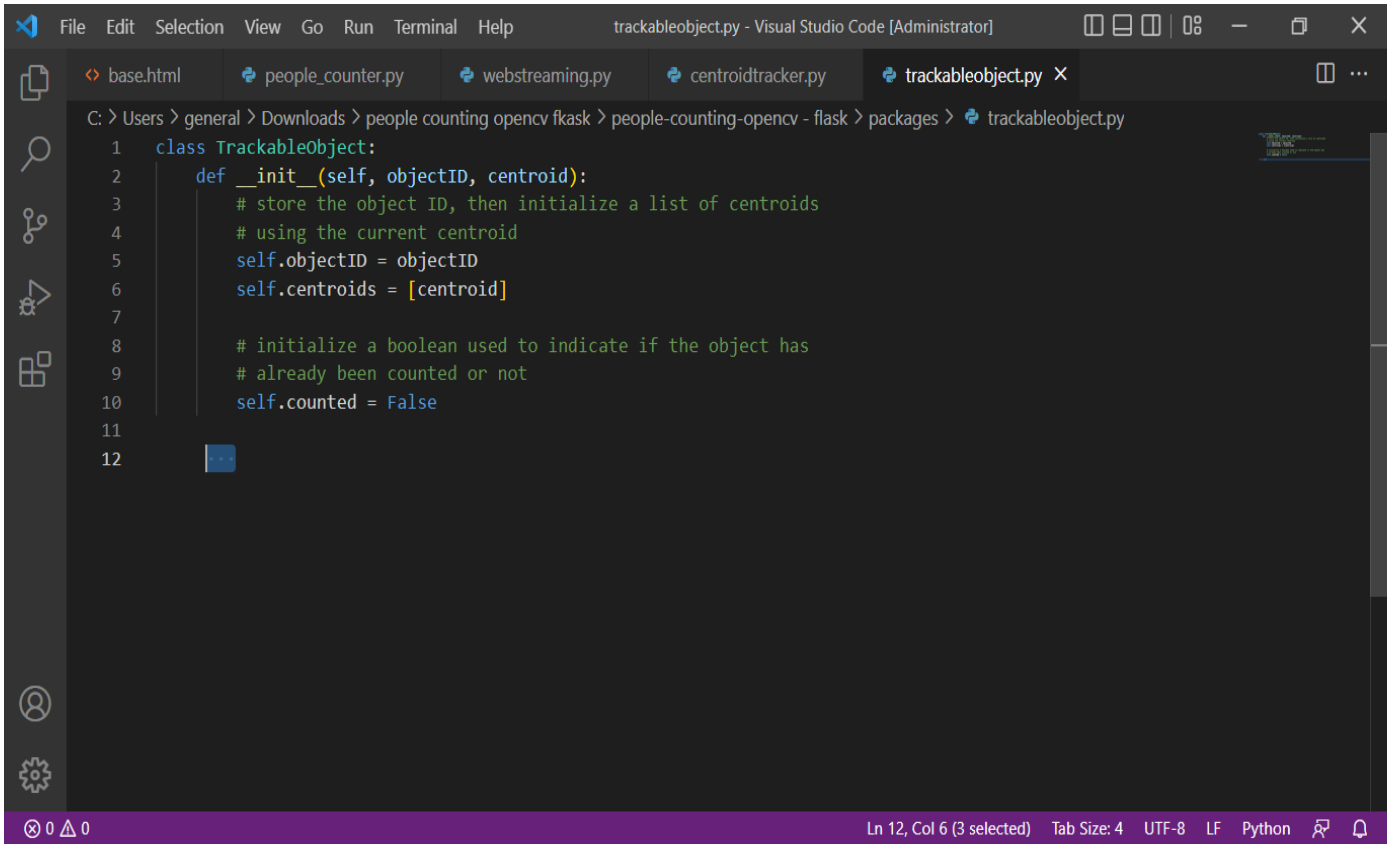
4. centroidtracker.py



```
 # 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]

    def update(self, rects):
        # check to see if the list of input bounding box rectangles
        # is empty
        if len(rects) == 0:
            # loop over any existing tracked objects and mark them
            # as disappeared
            for objectID in list(self.disappeared.keys()):
                self.disappeared[objectID] += 1

                # if we have reached a maximum number of consecutive
                # frames where a given object has been marked as
                # missing, deregister it
                if self.disappeared[objectID] > self.maxDisappeared:
                    self.deregister(objectID)

            # return early as there are no centroids or tracking info
            # to update
            return self.objects

        # initialize an array of input centroids for the current frame
        inputCentroids = np.zeros((len(rects), 2), dtype="int")

        # loop over the bounding box rectangles
        for (i, (startX, startY, endX, endY)) in enumerate(rects):
            # use the bounding box coordinates to derive the centroid
            cX = int((startX + endX) / 2.0)
            cY = int((startY + endY) / 2.0)
            inputCentroids[i] = (cX, cY)

        # if we are currently not tracking any objects take the input
        # centroids and register each of them
        if len(self.objects) == 0:
            for i in range(0, len(inputCentroids)):
                self.register(inputCentroids[i])

        # otherwise, are are currently tracking objects so we need to
        # try to match the input centroids to existing object
        # centroids
        else:
            # grab the set of object IDs and corresponding centroids
            objectIDs = list(self.objects.keys())
            objectCentroids = list(self.objects.values())

            # compute the distance between each pair of object
            # centroids and input centroids, respectively -- our
            # goal will be to match an input centroid to an existing
            # object centroid
            D = dist.cdist(np.array(objectCentroids), inputCentroids)

            # in order to perform this matching we must (1) find the
            # smallest value in each row and then (2) sort the row
            # indexes based on their minimum values so that the row
            # with the smallest value as at the *front* of the index
            # list
            rows = D.min(axis=1).argsort()

            # next, we perform a similar process on the columns by
            # finding the smallest value in each column and then
            # sorting using the previously computed row index list
            cols = D.argmin(axis=1)[rows]
            # in order to determine if we need to update, register,
            # or deregister an object we need to keep track of which
            # of the rows and column indexes we have already examined
            usedRows = set()
            usedCols = set()

            # loop over the combination of the (row, column) index
            # tuples
            for (row, col) in zip(rows, cols):
                # if we have already examined either the row or
                # column value before, ignore it
                if row in usedRows or col in usedCols:
                    continue
                # if the distance between centroids is greater than
                # the maximum distance, do not associate the two
                # centroids to the same object
                if D[row, col] > self.maxDistance:
                    continue
                # otherwise, grab the object ID for the current row,
                # set its new centroid, and reset the disappeared
                # counter
                objectID = objectIDs[row]
                self.objects[objectID] = inputCentroids[col]
                self.disappeared[objectID] = 0

                # indicate that we have examined each of the row and
                # column indexes, respectively
                usedRows.add(row)
                usedCols.add(col)

            # compute both the row and column index we have NOT yet
            # examined
            unusedRows = set(range(0, D.shape[0])).difference(usedRows)
            unusedCols = set(range(0, D.shape[1])).difference(usedCols)

            # in the event that the number of object centroids is
            # equal or greater than the number of input centroids
            # we need to check and see if some of these objects have
            # potentially disappeared
            if D.shape[0] >= D.shape[1]:
                # loop over the unused row indexes
                for row in unusedRows:
                    # grab the object ID for the corresponding row
                    # index and increment the disappeared counter
                    objectID = objectIDs[row]
                    self.disappeared[objectID] += 1

                    # check to see if the number of consecutive
                    # frames the object has been marked "disappeared"
                    # for warrants deregistering the object
                    if self.disappeared[objectID] > self.maxDisappeared:
                        self.deregister(objectID)
```

```
    # otherwise, if the number of input centroids is greater
    # than the number of existing object centroids we need to
    # register each new input centroid as a trackable object
    else:
        for col in unusedCols:
            self.register(inputCentroids[col])

# return the set of trackable objects
return self.objects
```

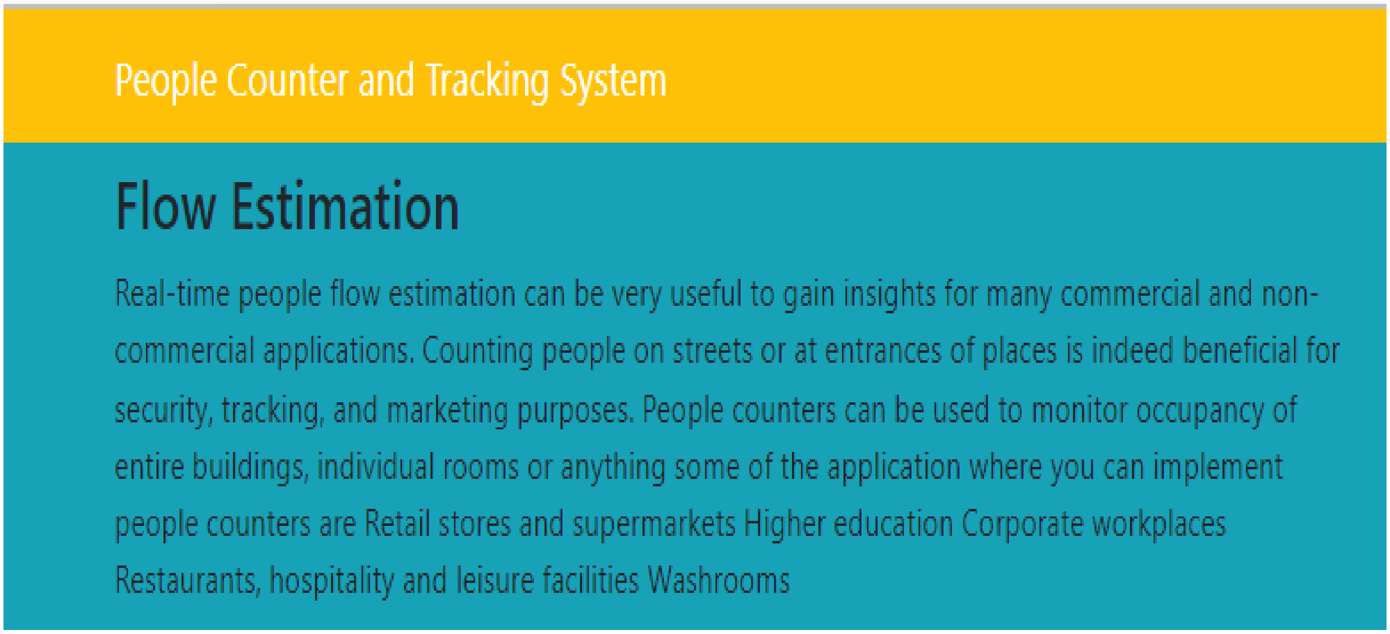
5. trackableobject.py



File
Edit
Selection
View
Go
Run
Terminal
Help
trackableobject.py - Visual Studio Code [Administrator]
base.html
people_counter.py
webstreaming.py
centroidtracker.py
trackableobject.py
C: > Users > general > Downloads > people counting opencv fkask > people-counting-opencv - flask > packages > trackableobject.py
⊗ 0 ⚠ 0
Ln 12, Col 6 (3 selected)
Tab Size: 4
UTF-8
LF
Python

```
class TrackableObject:
    def __init__(self, objectID, centroid):
        # store the object ID, then initialize a list of centroids
        # using the current centroid
        self.objectID = objectID
        self.centroids = [centroid]

        # initialize a boolean used to indicate if the object has
        # already been counted or not
        self.counted = False

```

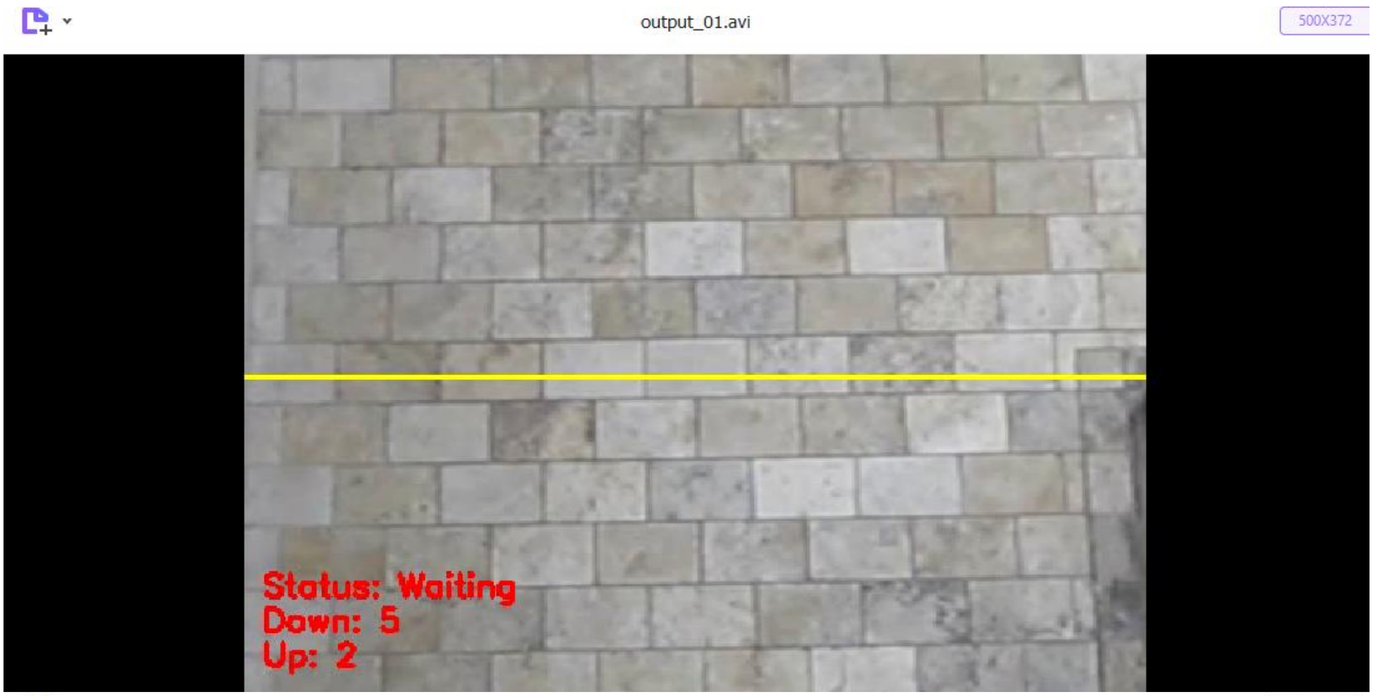
3. CONCLUSION



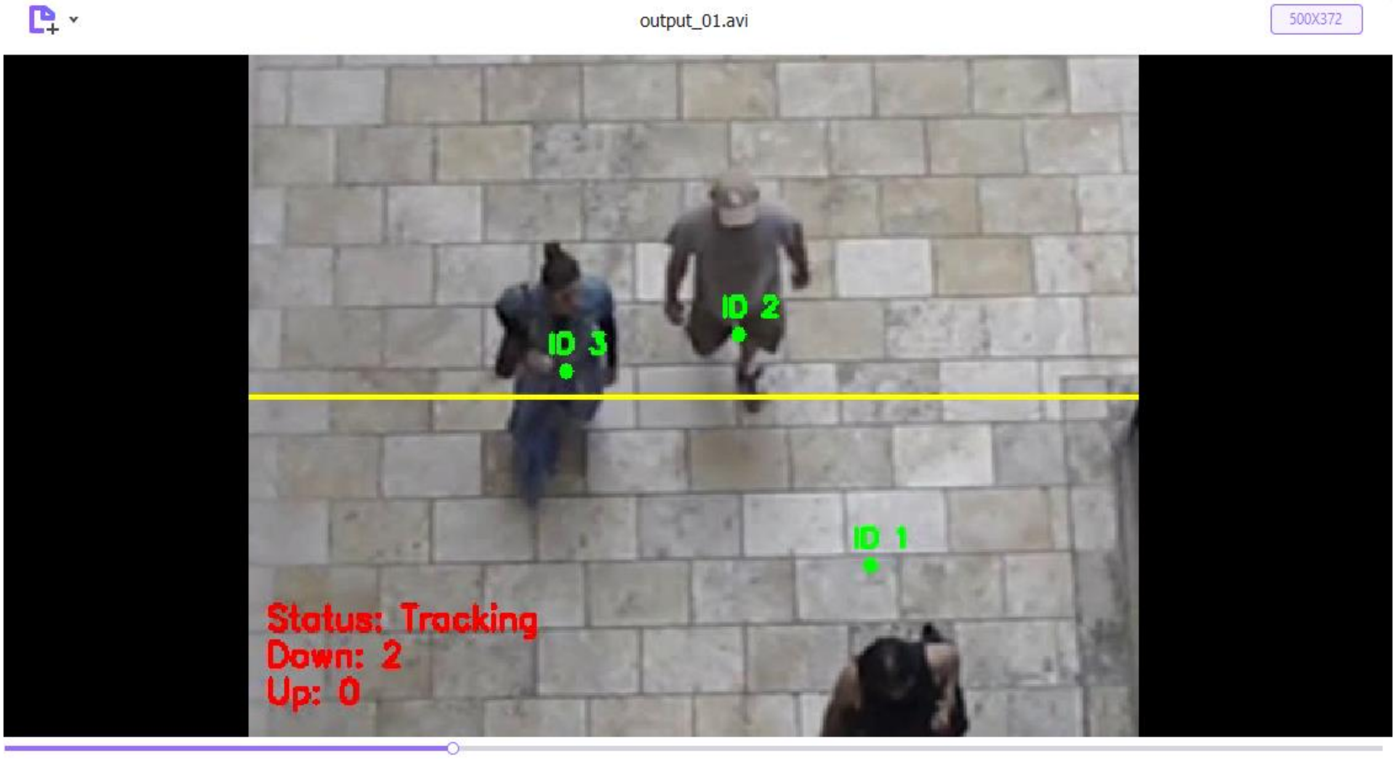
People Counter and Tracking System

Flow Estimation

Real-time people flow estimation can be very useful to gain insights for many commercial and non-commercial applications. Counting people on streets or at entrances of places is indeed beneficial for security, tracking, and marketing purposes. People counters can be used to monitor occupancy of entire buildings, individual rooms or anything some of the application where you can implement people counters are Retail stores and supermarkets Higher education Corporate workplaces Restaurants, hospitality and leisure facilities Washrooms



Status: Waiting
Down: 5
Up: 2



Status: Tracking
Down: 2
Up: 0

4. APPLICATION

1. Used at Retail stores
2. Used at Supermarkets
3. Used at Corporate workplaces
4. Used at Restaurants, hospitality and leisure facilities

5. ADVANTAGES

1. It gives you real time data on usage of space
2. It improves Customer safety
3. It provides insight into customer behavior at any business
4. Retail people counters provide valuable visitor analytics
5. Increased Safety
6. Streamlined Operations

6. FUTURE SCOPE

- We develop a people counter device to count the number of pedestrians walking through a door or corridor through a video or camera. Most of the time, this system is used at the entrance of a building so that the total number of visitors can be recorded. Live stream of visitor’s flow is streamed on to a web application.
- Overhead closed-circuit television camera, or IP camera, tracks people's movements.
- The camera connects to a people counter which accurately detects and records how many people pass through the counting zone.
- Most of the time, this system is used at the entrance of a building so that the total number of visitors can be recorded. Live stream of visitor’s flow is streamed on to a web application

7.BIBILIOGRAPHY

Thiago Teixeira, Gershon Dublon, Andreas Savvides,"A Survey of Human-Sensing: Methods for Detecting Presence, Count, Location, Track, and Identity," *ENALAB technical report*, 2010.
Google Scholar

Suguna P Subramanian, Jrgen Sommer, Stephen Schmitt and Wolfgang Rosenstiel,"RIL Reliable RFID based Indoor Localization for Pedestrians," *IEEE International Conference on Software, Telecommunications and Computer Networks*, pp. 218-222, 2008.
Google Scholar

Zebin Cai, Zhu Liang Yu, Hao Liu and Ke Zhang,"Counting People in Crowded Scenes by Video Analyzing," *IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1841-1845, 2014.
Google Scholar

Kazuhiko Hashimoto, Katsuya Morinaka, Nobuyuki Yoshiike, Chjihiro Kawaguchi and Satoshi Matsueda,"People Count System Using Multi-Sensing Application," *IEEE International Conference on Solid State Sensors and Actuators*, vol. 2, pp. 1291-1294, 1997.
Google Scholar

Huijing Zhao and Ryosuke Shibasaki,"A Novel System for Tracking Pedestrians Using Multiple Single-Row Laser-Range Scanners," *IEEE Transactions on Systems, Man, And Cybernetics Part A: Systems And Humans*, vol. 35, no. 2, pp. 283-291, 2005.
Google Scholar

Andre Treptow, Grzegorz Cielniak and Tom Duckett,"Real-time People Tracking for Mobile Robots using Thermal Vision," *Robotics and Autonomous Systems, Elsevier*, vol. 54, no.9, pp. 729-739, 2006.
Google Scholar

Ya-Li Hou, and Grantham K. H. Pang,"People Counting and Human Detection in a Challenging Situation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems And Humans*, vol. 41, no. 1, pp. 24-33, 2011.
Google Scholar

Zheng Ma and Antoni B. Chan,"Crossing the Line: Crowd Counting by Integer Programming with Local Features," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2539-2546, 2013.
Google Scholar

Thomas B. Moeslund, Erik Granum, "A Survey of Computer Vision-Based Human Motion Capture," *IEEE Transactions on Computer Vision and Image Understanding*, vol. 81, Issue 3, pp. 231268, 2001.