Project Report - 3

Using Intelligent Transportation Systems to Enhance Pedestrian Safety at Beirut Signalized Intersection.

By

Team - 10	
Supriya Manda	**#R11847461**
Ajay Konkitala	**#R11845972**
Teja Potturi	**#R11842952**
Neha Lagatapati	**#R11856841**
Nikhil Chandra Reddy Desireddy	**#R11808040**
Lakshmi Surya Sesha Sai Varma Manthena	**#R11847455**

Proposed Solution:

As per the research paper, it has been stated that- "Around 25 percent of the injuries accidents in the Greater Beirut Area (GBA) are pedestrian's related crashes of which 45% occur at signalized intersections. The high percentage of pedestrian victims of road accidents is due to the lack of traffic control devices and the absence of knowledge of the pedestrian rights to cross the roadway and most of the times pedestrians have exceptional safety needs and tasks that must be addressed. Recent improvements in intelligent transportation systems (ITS) show great potential in assisting to reduce the number of injuries, fatalities and the costs associated with crashes knowing that the primary beneficiaries are pedestrians. The target of this paper is to select the most effective and appropriate ITS treatments that can really affect mobility, safety and environmental developments in GBA Al-Ghazal signalized intersections using the vehicle to pedestrian (V2P) and the pedestrian to infrastructure (P2I) communications and technologies as traffic signals adjustments, automated pedestrian detection (APD), and countdown pedestrian signal (CPS). In addition, we will check the effect of those treatments on the pedestrian-vehicle conflict". But the outcome of the implemented process has certain flaws which would be further overcomed by further development and will be discussed in this paper. We propose to use a live camera at the intersections which eventually detects the pedestrians, traffic and controls the signals depending upon their movements.

Timeline to accomplish the project:
Implemented and executed the project and delivered the outcome successfully.

Report what has been done so far:

ANALYSIS:

Problem Statement in paper:

- The paper "Using Intelligent Transportation Systems to Enhance Pedestrian Safety at Beirut Signalized Intersection" is the study done by Rania Wehbe, Zaher Massaad and Elie Otayek. It focuses on the high percentage of pedestrian victims of road accidents at the Al-Ghazal signalized intersection in Greater Beirut Area (GBA).
- This particular intersection was chosen due to lack of traffic facilities combined with high pedestrian volume, which favors the accidents of the pedestrians. Due to this, Al-Ghazal signalized intersection has been considered to be a black spot for pedestrians.
- They mention the various reasons for such accidents. Some of the reasons mentioned are bad geometry concept, high traffic volumes, missing crosswalk, yield markings, no traffic signals or poor maintenance of traffic signals.

Existing Solutions:

- Various solution scope along with their implementation cost is mentioned in the paper theoretically. The authors state that NRSC implements the given solution analysis practically. Below are the intelligent transportation systems (ITS) solutions proposed in the paper.
- At an intersection, the signals and crosswalks should be well organized.
- The pedestrians should be alerted on the time remaining to cross the intersection (countdown pedestrian signal).

- Automated Pedestrian Detection: This is the most advanced ITS strategy. Using this method, the presence of pedestrians is detected without their effort. This reduces around 185 to 45 accidents but has a high implementation cost (~$10,000 - $70,000) per crosswalk.

Proposed Solution:

- The Automated Pedestrian Detection technique that is mentioned in the paper is taken as the base point for this project implementation which is cost effective.
- We will be able to detect the presence of a pedestrian without any action from them through video recording at traffic signals.
- If a pedestrian is detected to be waiting at the signal, the signal will change, and the walkway will be opened for the pedestrians to cross safely.
- This procedure is cost effective as the cameras that are already installed to detect signal jumping will be used to implement it.
- The program used in this project will be inserted into the already existing cameras.
- Additionally, we will be able to detect vehicles such as cycles along with the pedestrians in this solution.
- Due to this, the solution can easily be implemented at any traffic signal without high costs.
- This will enhance the safety of pedestrians by reducing the fatalities.

Analyzed the ways to implement the project and checked different existing solutions. For example the proposed solution in the base paper is to use infrared rays, radio waves and pressure mats to detect a person and control the intersection based on that. But, this method is very costly to implement at around $10000 per month. One solution we thought of is to check for pedestrians between red signals to decide whether to allow the pedestrians or not. But, now the solution is to check for the pedestrian traffic all the time and control the traffic if the system detects a person. But later analyzed that there is a problem of traffic stalling using this method. If there is a continuous flow of pedestrian traffic, the system would allow the pedestrian flow continuously which will stall other traffic. So, we have to make sure to allow other traffic to flow even though our priority is pedestrian traffic.

One other thing analyzed is what module we should use for the face detection. The software we chose to develop the project is in python. So, there are multiple modules we can use to detect the face like

- **OpenCV (Open Source Computer Vision Library)**
- Dlib
- Haar Cascade Classifier (built into OpenCV)
- MTCNN (Multi-Task Cascaded Convolutional Networks)
- Face Recognition.

We finally decided to use the module OpenCV. It contains Viola-Jones algorithm and the HOG (Histogram of Oriented Gradients) algorithm and is considered one of the best modules. The main reason for this choice is its ability to train custom models along with the custom pre trained models even if the pre trained models are more than enough. Further, we use **sockets** to establish a client-server connection in order to pass the message to the traffic signal. In addition to this, a time mechanism has been implemented that gives the pedestrians the time to cross without the change of signal(to green) until he is no longer detected. Once the pedestrian is no longer detected, the signal changes and allows the vehicles further.

Models and modules need to be used. (for now and for further detection).

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is used for a wide range of tasks including image and video processing, object detection and tracking, and machine learning. The library is written in C++, but also provides interfaces for Python, Java, and MATLAB, making it accessible to developers using different programming languages.

OpenCV includes a large collection of algorithms that can be used to process images and videos, such as filters, feature detection, and object recognition. It also provides tools for camera calibration, stereo vision, and 3D reconstruction. OpenCV is an open-source project and is maintained by a community of developers who contribute to its development and improvement.

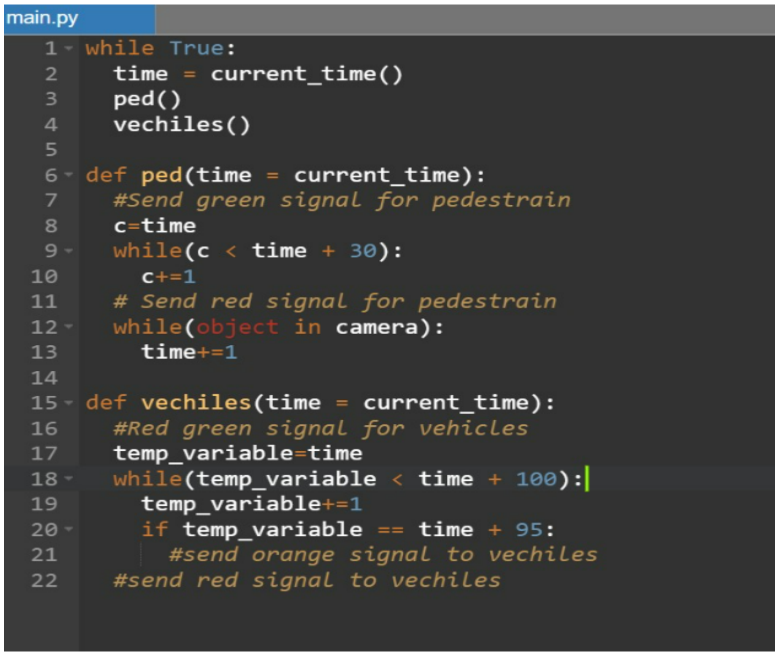
imutils is a Python package that provides a set of convenience functions to help with basic image processing tasks such as resizing, rotating, translating, and displaying images using OpenCV. The package is built on top of OpenCV, and its goal is to simplify common image processing tasks and make them easier to perform in Python.

Some of the common functions provided by imutils include:

- Resizing an image to a specified width or height while maintaining its aspect ratio
- Rotating an image by a specified angle
- Translating an image by a specified number of pixels in the x and y directions
- Converting an image from one color space to another, such as from RGB to grayscale
- Displaying an image in a window on the screen, with support for keyboard events and mouse callbacks
- imutils can be installed using pip, and it is compatible with both Python 2 and Python 3. It can be a useful tool for quickly prototyping image processing pipelines or for performing basic image processing tasks in a Python script or Jupyter notebook.
- Data set download for the model to test and train.

Pseudo code for signals// *hardcode for traffic signal*

Till date, live cameras haven’t been implemented to control the traffic signals. Below is the pseudocode developed to control the signals based on pedestrian detection. It explains that the walk time for pedestrians is 30sec. If the pedestrian hasn’t crossed yet, the main signal doesn’t change to green but stays red. And the pedestrian signal stays Frozen.



```
while True:
  time = current_time()
  ped()
  vechiles()

def ped(time = current_time):
  #Send green signal for pedestrain
  c=time
  while(c < time + 30):
    c+=1
  # Send red signal for pedestrain
  while(object in camera):
    time+=1

def vechiles(time = current_time):
  #Red green signal for vehicles
  temp_variable=time
  while(temp_variable < time + 100):
    temp_variable+=1
    if temp_variable == time + 95:
      #send orange signal to vechiles
  #send red signal to vechiles
```

Sockets:

We use **sockets** to establish a client-server connection in order to pass the message to the traffic signal. In addition to this, a time mechanism has been implemented that gives the pedestrians the time to cross without the change of signal(to green) until he is no longer detected. Once the pedestrian is no longer detected, the signal changes and allows the vehicles further.

To send messages over a network, sockets and the socket API are utilized. They offer a channel for inter-process dialogue (IPC). The network may be physically connected to an external network and have its own links to other networks, or it may be a logical network that is internal to the computer.

Updated Code:

#Client.py

```
#importing modules

import cv2

import imutils

import socket

import time

from cryptography.fernet import Fernet




s = socket.socket(socket.AF_INET,

                socket.SOCK_STREAM)


s.bind(('', 5000))


s.listen(1)

c, addr = s.accept()
```

```
print("CONNECTION FROM:", str(addr))

hog = cv2.HOGDescriptor()

hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# Reading the Image
#image = cv2.imread('img.jpg')
vid = cv2.VideoCapture(0)

center_coordinates = (40,40)

radius = 30

color = (0, 0, 255)

thickness = -1
def ped():
  c.send("green,pedestrain".encode())
  #Red green signal for pedestrain
  time.sleep(10)
  c.send("red,pedestrain".encode())
```

```
# Send red signal for pedestrain

hog = cv2.HOGDescriptor()

hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

car_cascade = cv2.CascadeClassifier('cars.xml')

vid = cv2.VideoCapture(0)

while(True):


  ret, image = vid.read()

  img = cv2.imread("car.png", cv2.IMREAD_COLOR)



  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

  cars = car_cascade.detectMultiScale(gray, 1.1, 1)

  print(len(cars))

  if len(cars)!=0:



      for (x,y,w,h) in cars:

          cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

      result=cv2.imwrite('pic.png', img)
```

```
    key = Fernet.generate_key()


    # string the key in a file

    with open('filekey.key', 'wb') as filekey:

       filekey.write(key)




    with open('filekey.key', 'rb') as filekey:

        key = filekey.read()


    fernet = Fernet(key)




    # opening the original file to encrypt

    with open('pic.png', 'rb') as file:

        original = file.read()

    encrypted = fernet.encrypt(original)

    print(cars)

    with open('pic_encrypted.png', 'wb') as encrypted_file:

        encrypted_file.write(encrypted)

    c.send("car detected:image saved as new_pic.png ".encode())

image = imutils.resize(image,

                    width=min(1000, image.shape[1]))
    (regions, _) = hog.detectMultiScale(image,

                                        winStride=(4, 4),

                                        padding=(4, 4),

                                        scale=1.05)



    #print(regions)

    if len(regions)==0:

        #cv2.release()

        break

    elif len(regions)!=0:

            # msg="red"

              image = cv2.circle(image, center_coordinates, radius, color,
thickness)

            # c.send(msg.encode())

            # time.sleep(10)




    for (x, y, w, h) in regions:

        cv2.rectangle(image, (x, y),(x + w, y + h),(255, 0, 0), 2)



        # Showing the output Image

        #print(regions)

    cv2.imshow("Image", image)
    if cv2.waitKey(1) & 0xFF == ord('q'):

        break


  vid.release()

  cv2.destroyAllWindows()


def vechiles():

  #Red green signal for vehicles


  c.send("green,vehicles".encode())

  time.sleep(9)


  c.send("orange,vehicles".encode())

  #siganal pampali

  time.sleep(2)


  c.send("red,vehicles".encode())

  #red signal pamapli




while True:

  ped()

  vechiles()
```

#Server.py:

```
import socket

from cryptography.fernet import Fernet



s = socket.socket(socket.AF_INET,

                socket.SOCK_STREAM)



s.connect(('127.0.0.1', 5000))



msg = s.recv(1024)



while msg:

    print('Received:' + msg.decode())

    if msg.decode()[0:3]=='car':

        with open('filekey.key', 'rb') as filekey:

                        key = filekey.read()



                    # using the generated key

        fernet = Fernet(key)

        with open('PIC_encrypted.png', 'rb') as enc_file:

            encrypted = enc_file.read()
        # decrypting the file

        decrypted = fernet.decrypt(encrypted)



        # opening the file in write mode and

        # writing the decrypted data

        with open('new_pic_recieved.png', 'wb') as dec_file:

            dec_file.write(decrypted)



    msg = s.recv(1024)



# disconnect the client

#s.close()
```

Image encryption:

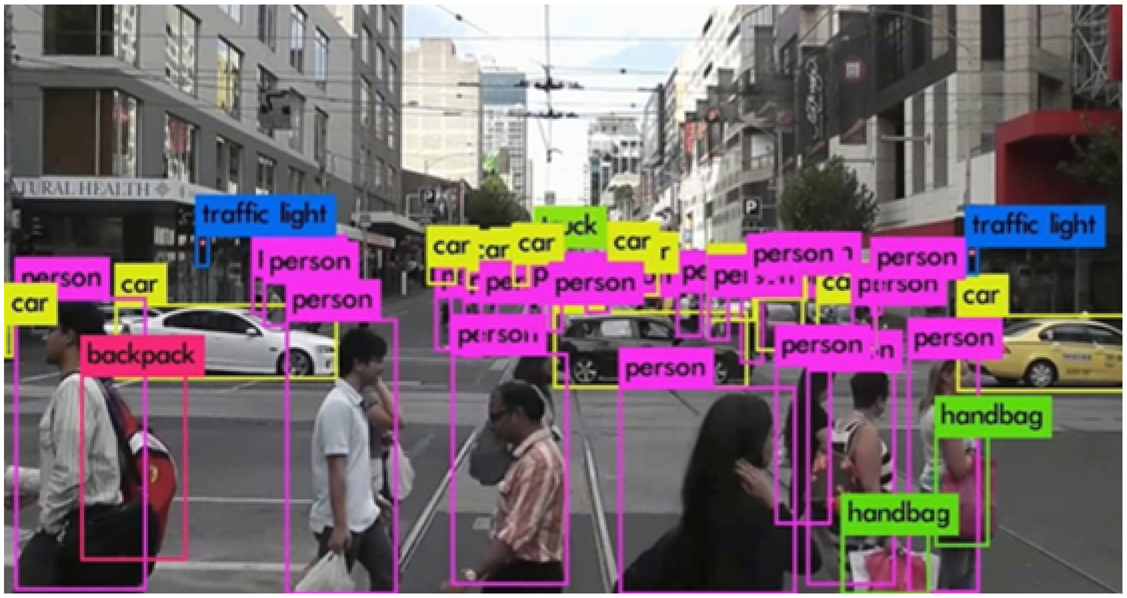
An image is sent from client to server via **RSA Encryption**.

FERNET Module in python generates a secret key for encrypting images and the encrypted bytes are sent to the server whereas the key is stored on a local file. This encrypted image is also stored but hidden. Upon key generation ,decryption is done on the server and is stored on the local file accordingly.

Modules Used:

Users can include OpenCV functions into Python scripts thanks to the OpenCV-Python module. It offers a range of image processing tools, computer vision techniques, and machine learning methods that may be applied to several tasks, including face recognition, object tracking, picture filtering, and video analysis. Several operating systems, including Windows, Linux, Mac OS, iOS, and Android, are all compatible with the library. Moreover, it supports programming languages, including Java, C++, and Python. Using HOG Descriptor _getDefaultPeopleDetector was an experiment. Reading the image, drawing a red circle of a certain thickness, and then displaying the resultant image is what we tried to achieve so far.

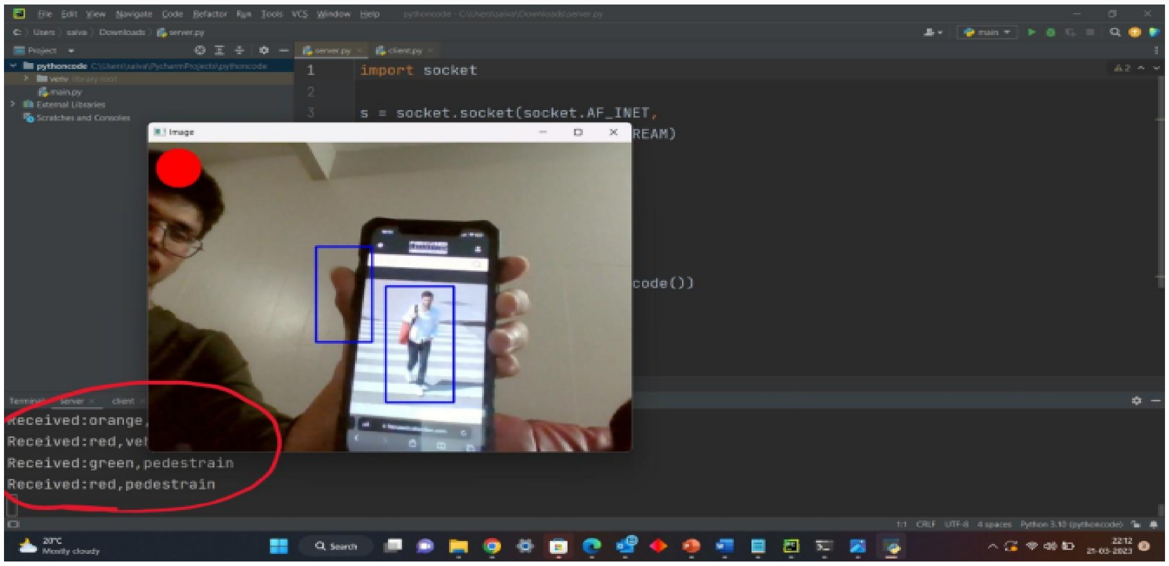
We also have a plan to employ YOLo V3 in the future for things like hit-and-run detection using license plates. This method detects objects in the actual world. It is mostly used for image detection and was trained using enormous datasets that were previously preserved. For instance, if an image has a lot of data, it analyzes the things that are present in the image, such as fans or people, and the findings are displayed with the fan or person highlighted. This is directly tied to our image detection, which we are still working to improve in terms of both its use and its efficacy. This is still in the talks whether to implement it or not.



Implementation:

Image detection is the essential part in our project. We tried to implement it by recognizing pedestrians on crosswalks and then controlling the signals. OpenCV played an Important role in detecting pedestrians. OpenCV-Python library allows users to use the OpenCV functionalities in Python programs. It provides a variety of image processing functions, computer vision algorithms, and machine learning algorithms that can be used for various applications such as face detection, object tracking, image filtering, video analysis, and more. Successful detection of pedestrians is achieved by indicating a square box. Usage of different traffic images is done to see if the code is working. If the pedestrian is detected Signals are controlled so that to avoid accidents vehicle signals continue to be red. To avoid the wear and tear of the camera, reduced usage of the camera is used when not needed. For the future scope I am planning to train the model using CNN(Convolution Neural Model). That is a bit tough but we started using it. Hopefully by the next round we will be able to do something on it. Attaching the images that are detected while testing.

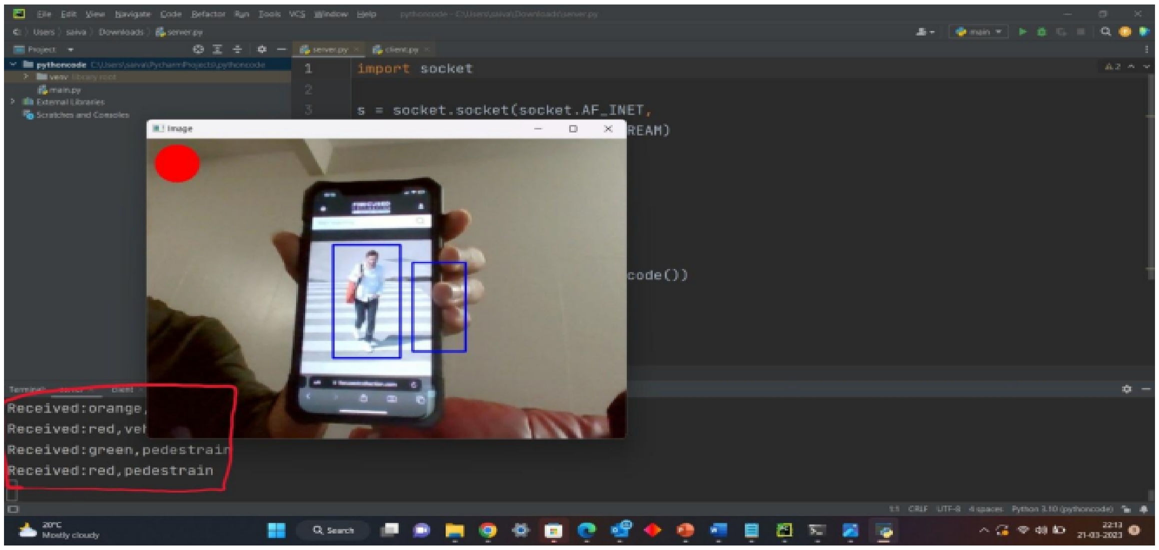
Below are some of the snapshots taken while execution:



```
import socket

s = socket.socket(socket.AF_INET,
REAM)
code())
```

```
Received:orange,
Received:red,veh
Received:green,pedestrain
Received:red,pedestrain
```



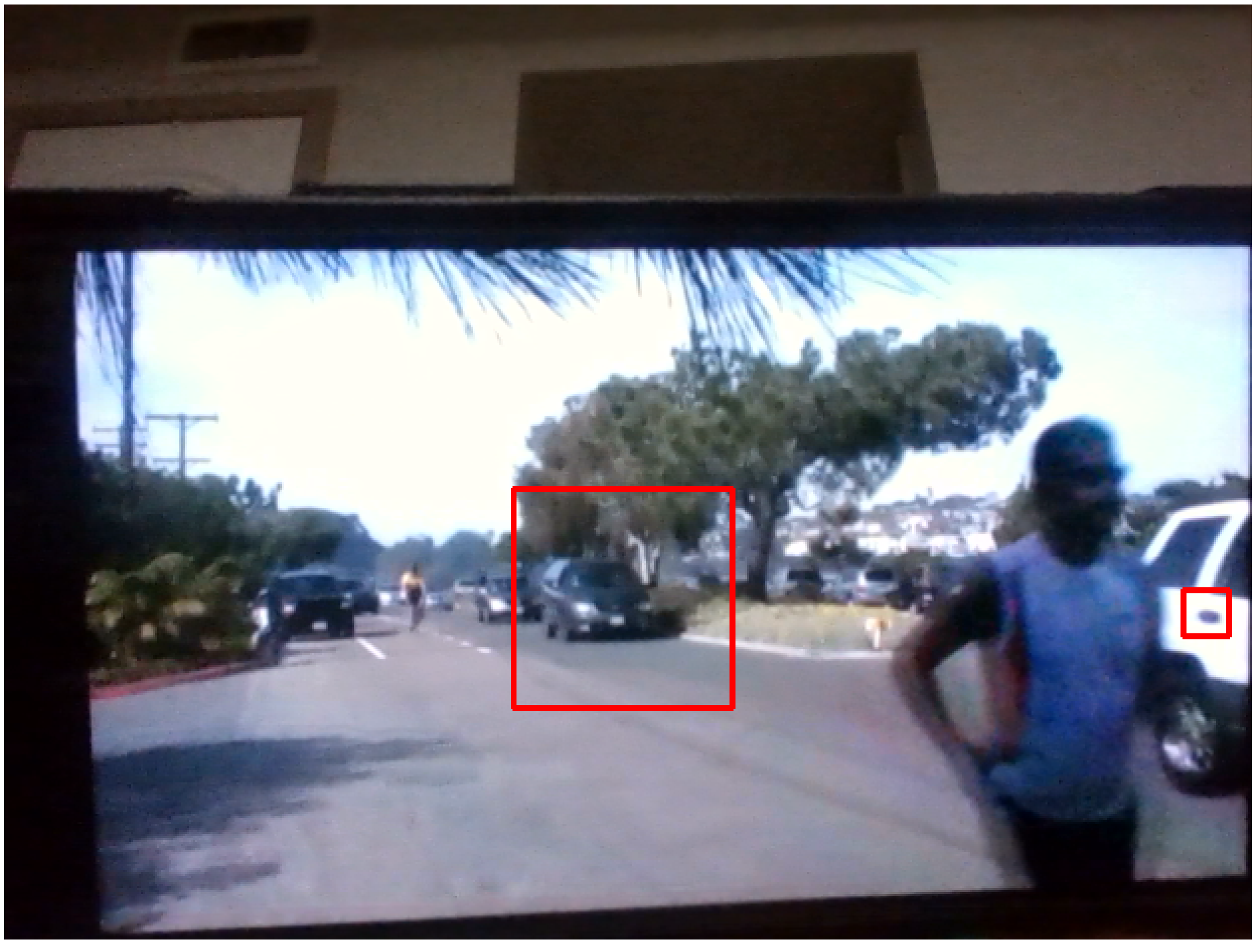
```
import socket

s = socket.socket(socket.AF_INET,
REAM)
code())
```

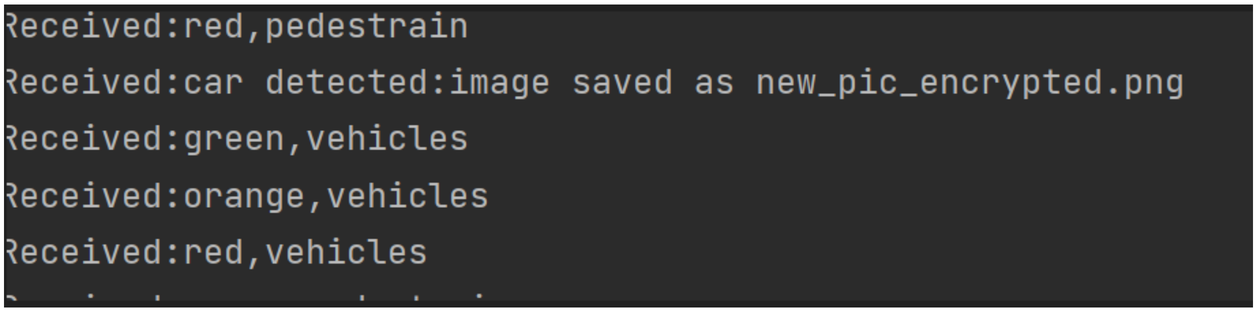
```
Received:orange,
Received:red,veh
Received:green,pedestrain
Received:red,pedestrain
```

Output:

Car Detection:

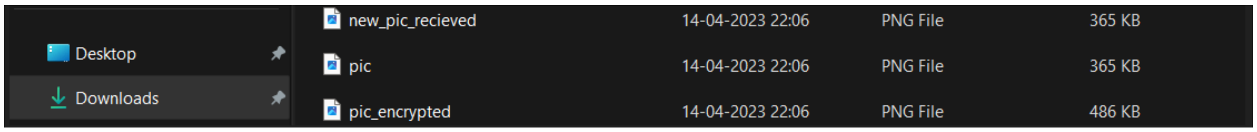


O/P when a vehicle is detected.



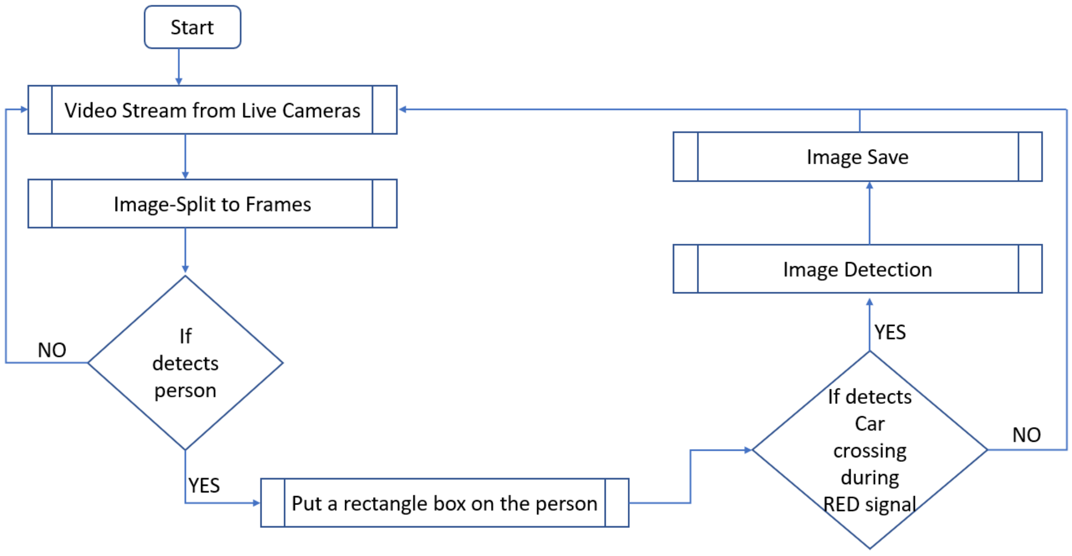
```
Received:red,pedestrain
Received:car detected:image saved as new_pic_encrypted.png
Received:green,vehicles
Received:orange,vehicles
Received:red,vehicles
```

Encrypted image saved on local file:



new_pic_recieved	14-04-2023 22:06	PNG File	365 KB
pic	14-04-2023 22:06	PNG File	365 KB
pic_encrypted	14-04-2023 22:06	PNG File	486 KB

Flow Chart:


Start
Video Stream from Live Cameras
Image-Split to Frames
If detects person
NO
YES
Put a rectangle box on the person
If detects Car crossing during RED signal
YES
NO
Image Detection
Image Save

\

Steps to follow:

Step 1: Read the video stream using webcam
Step 2: Divide it into frames in which each frame will act as an image.
Step 3: Input each frame image to the pretrained opencv model.
Step 4: If a person is detected in each image, go to Step 5. If no person is detected, go to Step 2.
Step 5: The detected person will be shown in a rectangular box highlighting the person.
Step 6: It tries to detect if any car is crossing during the Red signal. If no person is detected, it loops back to the video stream.
Step 7: If the car is detected, it tries to save the image detection done. It then returns to Step 2.

Report the contribution of each person:

Supriya Manda - R11847461 - Designing the process and the pattern for the code.Updated and Implemented client side code.

Ajay Konkitala - R11845972 - Analyzed OPENCV Module and its implementation. I tried to get as much information on how to use the module. Tried using _getDefaultPeopleDetector for detection of People. Also researched YoLo v3 for our future developments., developing the model for crosswalk detection and implementation.

Teja Potturi - R11842952 - Model analysis, implementation of modules and software setup.Updated and Implemented server side code.

Sai Varma Manthena-R11847455- Detecting the suitable module for person detection and solving the bugs in the coding part. collecting the data for the images which contain the pedestrians in the traffic, Hardcoded the signal logic.

Nikhil Chandra Reddy Desireddy - R11808040 - Analyzed about different face detection modules in python. Analyzed possible solutions and their drawbacks and what's the best possible solution to implement., trained different models and concluded on the best one to be used.

Neha Lagatapati - R11856841 - Analyzed and worked on Encryption and Decryption methodologies for enhancement of security for upcoming changes for future implementation.

Future Scope :

- ➢ Currently using the model, we are able to detect the pedestrian waiting to cross the road.
- ➢ Along with this the model is able to detect if a car is attempting to cross the road when the signal shows red light.
- ➢ This can be further developed by adding the features to detect and change the signals based on the vehicle count.
- ➢ The model can be further trained based on the frequency of the vehicles and pedestrians.
- ➢ When these advanced features are implemented in the model, it will help to reduce vehicle congestion on road along with focus on pedestrian safety for road crossing.
- ➢ Once the testing and training of the model with mentioned advanced features in future scope is achieved, the model can be implemented in the real world pedestrians problems.