MULTI-SURVEILLANCE CAMERA FOR HOSTAGE DETECTION

INNOVATIVE PRODUCT DEVELOPMENT REPORT

Submitted by

A. Vaishnavi	**G. Sahruthi**	**Ch. Harshini**
20RH1A6604	**20RH1A6625**	**21RH5A6602**

Under the Esteemed Guidance of

Mrs. P. Gayatri

Assistant Professor

in partial fulfillment of the Academic Requirements for the Degree of

BACHELOR OF TECHNOLOGY

CSE-AI & ML

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

May- 2023

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NBA & NAAC with 'A' Grade, UGC, Govt. of India

NIRF Indian Ranking–2018, Accepted by MHRD, Govt. of India
Permanently Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution
AAAA+ Rated by Digital Learning Magazine, AAA+ Rated by Careers 360 Magazine,
6th Rank CSR, Platinum Rated by AICTE-CII Survey, Top 100 Rank band by ARIIA, MHRD, Govt. of India National Ranking-Top 100 Rank band by Outlook, National Ranking-Top 100 Rank band by Times Ne ws Magazine
Maisammaguda, Dhullapally, Secunderabad, Kompally-500100

DEPARTMENT OF CSE - AI &ML

CERTIFICATE

This is to certify that the Innovative Product Development work entitled **Multi surveillance camera for hostage detection** is carried out by **Anampally Vaishnavi(20RH1A6604), Garipelly Sahruthi(20RH1A6625), Challa Harshini(21RH5A6602)** in partial fulfillment for the award of degree of **BACHELOR OF TECHNOLOGY** in CSE - AI & ML, Jawaharlal Nehru Technological University, Hyderabad during the academic year 2022-2023.

Supervisor's Signature	**Head of the Department**
Mrs. P. Gayatri	**Dr. G. Kalpana**
Asst. Professor	**Professor**

External Examiner

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution –UGC,Govt. of India)

(Permanently Affiliated to JNTU, Hyderabad, Approved by AICTE - ISO 9001:2015 Certified)

Accredited by NBA & NAAC with 'A' Grade

NIRF India Ranking 2018, Accepted by MHRD, Govt. of India

Maisammaguda, Dhulapally (Post Via Hakimpet), Secunderabad – 500100

Department of CSE –AI & ML

DECLARATION

We hereby declare that the Innovative Product Development entitled **Multi Surveillance camera for hostage detection** submitted to Malla Reddy Engineering College For Women affiliated to Jawaharlal Nehru Technological University, Hyderabad (JNTUH) for the award of the Degree of Bachelor of Technology in CSE-AI & ML is a result of original research work done by us. It is further declared that the Innovative Product Development report or any part thereof has not been previously submitted to any University or Institute for the award of Degree.

A. Vaishnavi (20RH1A6604)

G. Sahruthi (20RH1A6625)

Ch. Harshini (21RH5A6602)

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women** and **Department of CSE-AI&ML** which gave us the opportunity to have expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honorable Minister of Telangana State **Sri**.**Ch. Malla Reddy Garu,** founder chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project success.

We wish to convey gratitude to our **Principal Dr. Y. Madhavee Latha,** for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

We would like to thank **Prof. A. Radha Rani**, Director of Computer Science and Engineering & Information Technology for encouraging us to take up a project on this subject and motivating us towards the Project Work.

We express our sincere gratitude to **Dr.G.Kalpana , Head of the Department** of CSE–AI&ML for inspiring us to take up a project on this subject and successfully guiding us towards its completion.

We would like to thank our guide **Mrs. P. Gayatri, Asst. Professor of CSE-AI&ML** Dept and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

With Regards and Gratitude

A. Vaishnavi (20RH1A6604)

G. Sahruthi (20RH1A6625)

Ch. Harshini (21RH5A6602)

ABSTRACT

A group of terrorist have captured the building with hostages inside. We need to provide an Unmanned solution which could provide the essential information for the troops outside the building. Analyzing the situation and making a report on it will makes more time. So the hostage detection using surveillance camera or smart eye cameras will help out with this problem. Here it will give a report of people count, object identification and also background subtraction. We use technology like Deep Learning and python modules. We can use this at the time of hijacks, terrorists attacks, disaster prone areas, Underwater Rescuing, WildLife Monitoring, Warzones, military purposes, Robbery places and etc.

INDEX

CHAPTER 1

INTRODUCTION

PROJECT DEFINITION

Deep learning object detection is a fast and effective way to predict an object’s location in an image, which can be helpful in many situations. Here it makes background subtraction, count the number of people at a particular location and identifies the object with using the python modules like opencv, pillow etc.

PROJECT OVERVIEW

A group of terrorists have captured the building with hostages inside. We need to provide an Unmanned solution which could provide the essential information for the troops outside the building. Analyzing the situation and making a report on it will makes more time. So the hostage detection using surveillance camera or smart eye cameras will help out with this problem. Here it will give a report of people count, object identification and also background subtraction.

SOFTWARE REQUIREMENTS:

- Operating system: Windows 9,10.
- Coding Language: Python.
- Front-End : Python.
- Setup tools and pip : 3.6 and above.

HARDWARE REQUIREMENTS

- Camera

CHAPTER 2

LITERATURE SURVEY

Existing System:

There is no such existing system like this but there are some kind of existing systems where the background subtractor, Counting number of people and object identification individually. The existed systems in Object identification the accuracy is low to identify it properly. The Existed systems needs database to control and manage. It needs internet connection for sure.

Proposed System:

The proposed system consists of all the three solutions of background subtractor, counting people and object identification. The object identification has greater accuracy than the existing systems as here data augmentation is done. To use this system internet is not required. This proposed system has high accuracy as compared to existing systems.

Advantages:

- It is easy to use.
- High Accuracy.
- It can be worked at low network areas also.
- Can be handled by anyone.
- No use of Internet Connection.

Disadvantages:

- The information will not be recorded in databases.

CHAPTER 3

METHODOLOGY

Deep Learning:

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. A formal definition of deep learning is-neurons.

Deep Learning is a subset of Machine Learning that is based on artificial neural networks (ANNs) with multiple layers, also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain, and they are designed to learn from large amounts of data in an unsupervised or semi-supervised manner.

Deep Learning models are able to automatically learn features from the data, which makes them well-suited for tasks such as image recognition, speech recognition, and natural language processing. The most widely used architectures in deep learning are feedforward neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

Deep learning has revolutionized the world of computer vision—the ability for machines to "see" and interpret the world around them. Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible* layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

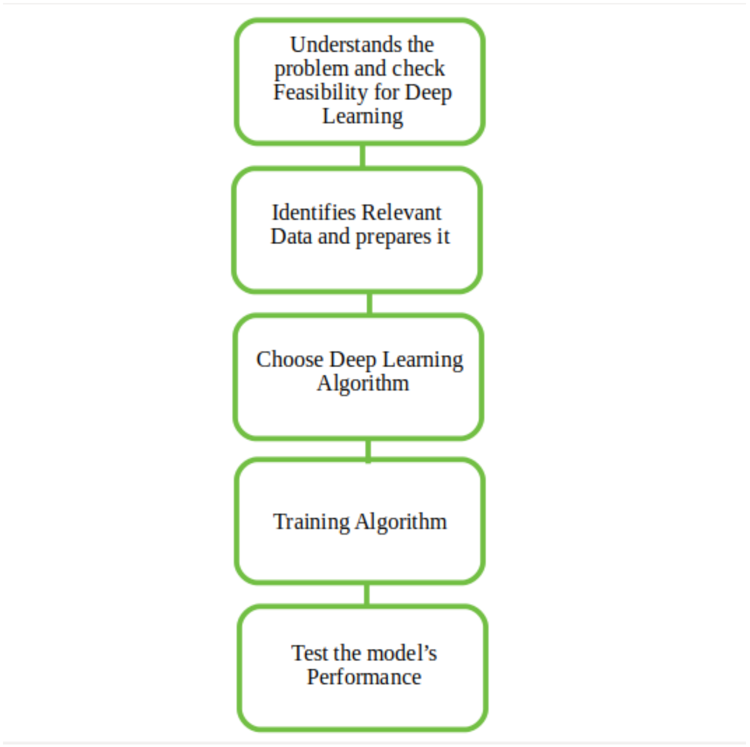

Understands the problem and check Feasibility for Deep Learning
Identifies Relevant Data and prepares it
Choose Deep Learning Algorithm
Training Algorithm
Test the model's Performance

Fig-Flowchart to understand Deep Learning

Image Processing Technology

Deep learning algorithms are well-suited for image processing tasks because they can learn to detect patterns in data that are too complicated for humans to discern. This means that deep learning algorithms can be used to automatically classify images, identify faces or objects in images, and even generate new images.

There are many benefits to using deep learning algorithms for image processing, including:

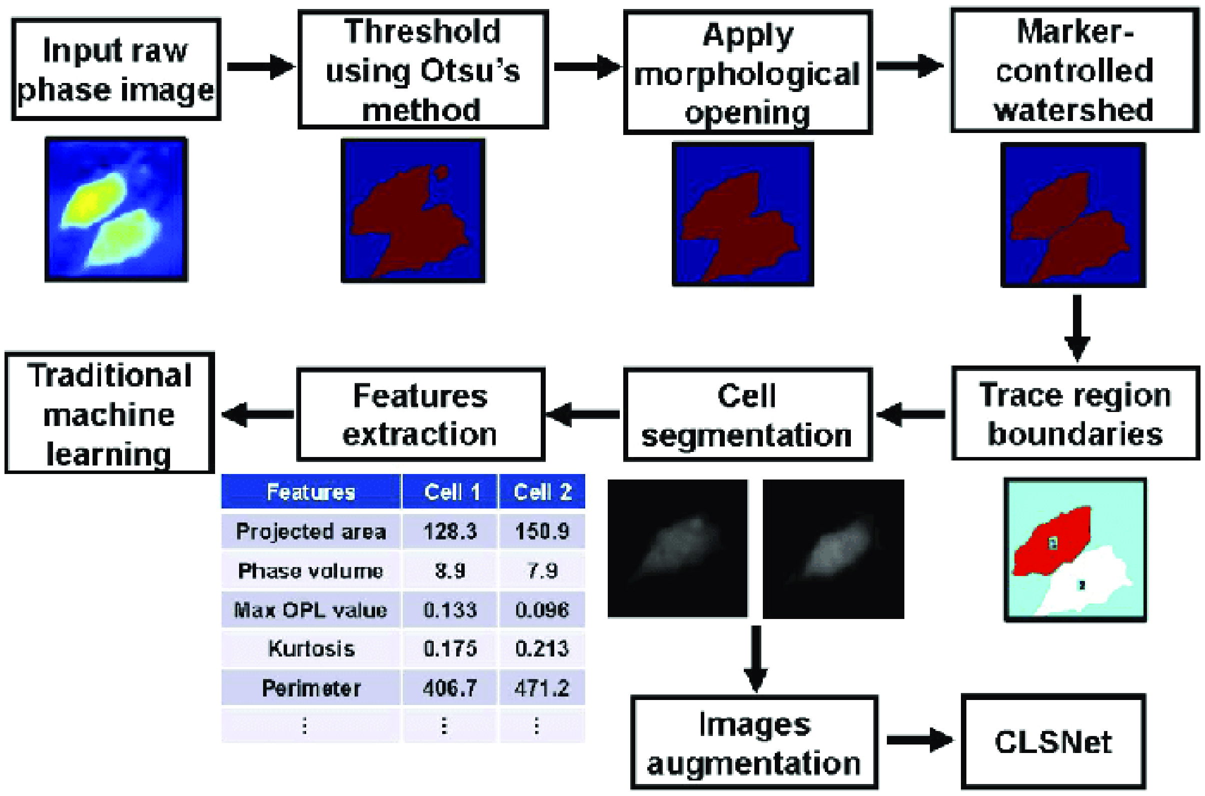
– improved accuracy: deep learning algorithms can learn to detect patterns that are too difficult for humans to discern, which leads to improved classification accuracy;

– increased speed: deep learning algorithms can process large amounts of data very quickly, which can lead to faster image classification;

– increased flexibility: deep learning algorithms can be trained to perform different image processing tasks, such as generating new images or identifying faces in pictures; and

– reduced costs: deep learning algorithms can automate image processing tasks that would otherwise need to be performed by human workers.

Deep learning algorithms can be very effective for image processing tasks. However, it is important to remember that there is no one-size-fits-all solution. The best algorithm for a particular task will depend on the specific data and the desired results. In general, however, deep learning algorithms are a powerful tool that can be used to achieve impressive results.


Input raw phase image
Threshold using Otsu's method
Apply morphological opening
Marker-controlled watershed
Trace region boundaries
Cell segmentation
Features extraction
Traditional machine learning
Images augmentation
CLSNet

Features	Cell 1	Cell 2
Projected area	128.3	150.9
Phase volume	8.9	7.9
Max OPL value	0.133	0.096
Kurtosis	0.175	0.213
Perimeter	406.7	471.2
⋮	⋮	⋮

Fig-Flowchart of image processing, machine-learning and deep-learning steps.

CHAPTER 4

SYSTEM ARCHITECTURE

Background Subtraction

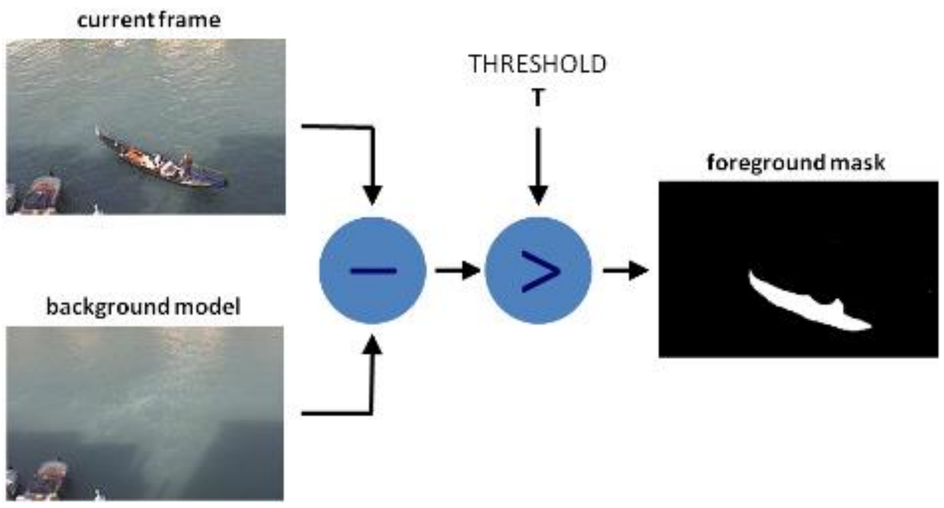


Fig: Background Subtraction

- Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.
- As the name suggests, BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.
- Background modeling consists of two main steps:
  - Background Initialization.
  - Background Update.

  In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

  Steps followed
- Read data from videos or image sequences by using cv::VideoCapture ;
- Create and update the background model by using cv::BackgroundSubtractor class;
- Get and show the foreground mask by using cv::imshow ;

  In OpenCV we have 3 algorithms to do this operation –

  BackgroundSubtractorMOG – It is a Gaussian Mixture-based Background/Foreground

Segmentation Algorithm.

BackgroundSubtractorMOG2 – It is also a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. It provides better adaptability to varying scenes due illumination changes etc.

BackgroundSubtractorGMG – This algorithm combines statistical background image estimation and per-pixel Bayesian segmentation.

Detect humans in an image(OpenCv Counter)

To detect humans in an image and draw bounding boxes around them, you can use the steps given below –

- Import the required library. In all the following examples, the required Python library
- is **OpenCV**. Make sure you have already installed it.

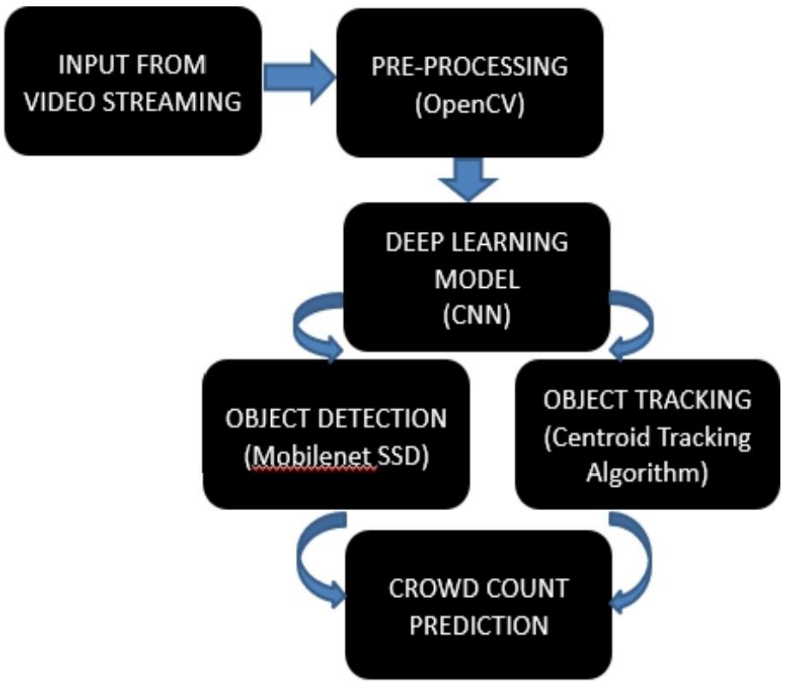

INPUT FROM VIDEO STREAMING
PRE-PROCESSING (OpenCV)
DEEP LEARNING MODEL (CNN)
OBJECT DETECTION (Mobilenet SSD)
OBJECT TRACKING (Centroid Tracking Algorithm)
CROWD COUNT PREDICTION

Fig – Flowchart for counting number of people

- Read the input image using **cv2.imread()** in a grayscale. Specify the full image path.
- Initialize a **HOG** descriptor object **hog = cv2.HOGDescriptor()** and set the SVM detector as **hog.setSVMDetector()** as default people detector.
- Detect humans in the input image using **hog.detectMultiScale()**. It returns the coordinates of detected humans in **(x,y,w,h)** format.

- Loop over all detected humans in the image and draw the bounding rectangles around the detected humans in the original image using **cv2.rectangle().**
- Display the image with the drawn bounding rectangles around the humans.

Object Identification

We will first open the image given above and create the environment of the picture to show it in the output.

Example

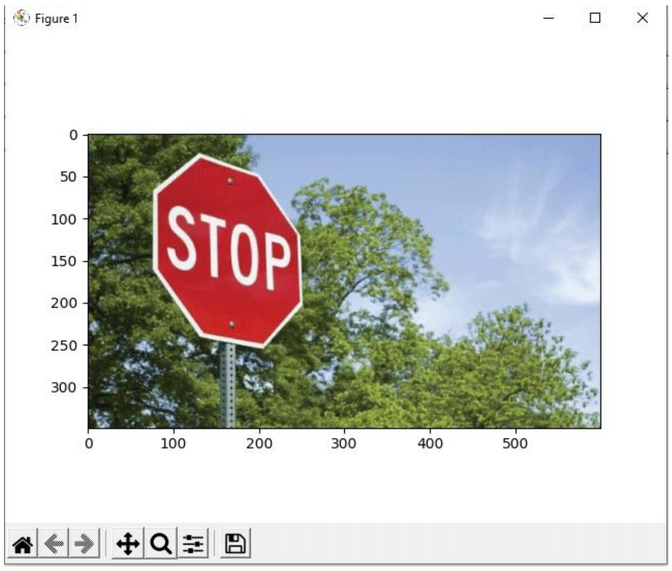
Opening the image using OpenCV and matplotlib library in a Python program:

```
# Import OpenCV module
import cv2
# Import pyplot from matplotlib as pltd
from matplotlib import pyplot as pltd
# Opening the image from files
imaging = cv2.imread("opencv-od.png")
# Altering properties of image with cv2
img_gray = cv2.cvtColor(imaging, cv2.COLOR_BGR2GRAY)
imaging_rgb = cv2.cvtColor(imaging, cv2.COLOR_BGR2RGB)
# Plotting image with subplot() from plt
pltd.subplot(1, 1, 1)
# Displaying image in the output
pltd.imshow(imaging_rgb)
pltd.show()
```

First, we have imported the OpenCV (as cv2) and matplotlib (as plt) libraries into the program to use their functions in the code. After that, we have opened the image file using the imread() function of cv2.

Then, we have defined the properties for the image we opened in the program using the cv2 functions. Then, we subplot the image using the subplot() function of plt and giving parameters in it. In last, we have used the imshow() and show() function of the plt module to show the image in the output.

As we can see in the output, the image is displayed as a result of the program, and its borders have been sub-plotted.



Recognition or object detection in the image

Now, we will use the detectMultiScale() in the program to detect the object present in the image. Following is the syntax for using detectMultiScale() function in the code:

found = xml_data.detectMultiScale(img_gray, minSize = (30, 30))

- We will use a condition statement with this function in the program to check if any object from the image is detected or not and highlight the detected part.

Example:

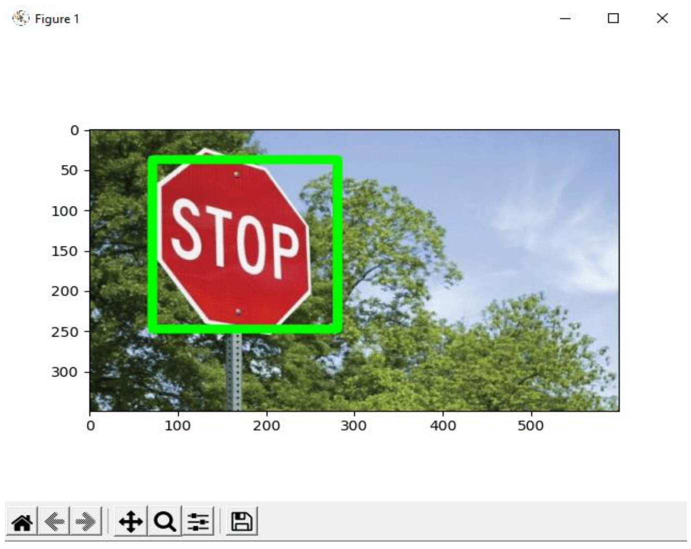
```
# Import OpenCV module
import cv2
# Import pyplot from matplotlib as plt
from matplotlib import pyplot as pltd
# Opening the image from files
imaging = cv2.imread("opencv-od.png")
# Altering properties of image with cv2
imaging_gray = cv2.cvtColor(imaging, cv2.COLOR_BGR2GRAY)
imaging_rgb = cv2.cvtColor(imaging, cv2.COLOR_BGR2RGB)
# Importing Haar cascade classifier xml data
xml_data = cv2.CascadeClassifier('XML-data.xml')
# Detecting object in the image with Haar cascade classifier
```

```
detecting = xml_data.detectMultiScale(imaging_gray, minSize = (30, 30))
# Amount of object detected
amountDetecting = len(detecting)
# Using if condition to highlight the object detected
if amountDetecting != 0:
for (a, b, width, height) in detecting:
cv2.rectangle(imaging_rgb, (a, b), # Highlighting detected object with rectangle
(a + height, b + width),  (0, 275, 0), 9)
# Plotting image with subplot() from plt
pltd.subplot(1, 1, 1)
# Displaying image in the output
pltd.imshow(imaging_rgb)
pltd.show()
```

After opening the image in the program, we have imported the cascade classifier XML file into the program. Then, we used the **detectMultiScale()** function with the imported cascade file to detect the object present in the image or not.

We used if condition in the program to check that object is detected or not, and if the object is detected, we have highlighted the detected object part using for loop with cv2 functions.

After highlighting the detected object part in the image, we have displayed the processed image using the plt **show()** and **imshow()** function.



CHAPTER 5

IMPLEMENTATION

The libraries used for implementation are:

1.OpenCV

2.Django

3.Cmake

4.Asgiref

5.Pillow

6.Imutils

7.numpy

8.tkinter

9.Time

10.Logging

11.Datetime

i)OpenCV: OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Opencv-programs and project.

ii)Django: Django is a Python-based web framework that allows you to quickly create efficient web applications. It is also called batteries included framework because Django provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use and that too for rapid development.

iii)CMake: CMake is a cross-platform Makefile generator! Simply put, CMake automatically generates the Makefiles for your project. It can do much more than that too (e.g., build MS

Visual Studio solutions), but in this discussion I focus on the auto-generation of Makefiles for C/C++ projects.

iv)Asgiref: ASGI is a standard for Python asynchronous web apps and servers to communicate with each other, and positioned as an asynchronous successor to WSGI.

v)Tkinter: Tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

vi)Numpy: NumPy is a Python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.

vii)Time: Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module we need to import it.

viii)Argparse: The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages. The module will also issue errors when users give the program invalid arguments.

ix)Logging: This module defines functions and classes which implement a flexible event logging system for applications and libraries.The key benefit of having the logging API provided by a standard library module is that all Python modules can participate in logging, so your application log can include your own messages integrated with messages from third-party modules.

x)DateTime: The datetime module supplies classes for manipulating dates and times.While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

SOURCE CODE

Frontend.py

```
from tkinter import *
from PIL import Image, ImageTk
import background_subtraction
import count_people
import identify_obj
class Window(Frame):
	def __init__(self, master = None):
		Frame.__init__(self, master)
		self.master = master
		self.init_window()
	def init_window(self):
		self.master.title("DRDO Multi Surveillance Drone")
		self.pack(fill = BOTH, expand = YES)
		runButton = Button(self, text="Background substraction", command = 
self.back_sub, bg = "#ff3300", activebackground = "#00ff99")
		runButton.place(height = 50, width = 200, x = 50, y = 540)
		runButton1 = Button(self, text="Identify Objects", command = self.iden_obj, 
bg = "#ff3300", activebackground = "#00ff99")
		runButton1.place(height = 50, width = 200, x = 50, y = 600)
		quitButton1 = Button(self, text="Count People", command = self.count, bg = 
"#ff3300")
		quitButton1.place(height = 50, width = 200,x = 350, y = 540)
		quitButton = Button(self, text="Exit", command = self.client_exit, bg = 
"#ff3300")
		quitButton.place(height = 50, width = 200,x = 350, y = 600)
		self.showImg()
		menu = Menu(self.master)
		self.master.config(menu = menu)
		file = Menu(menu)
		file.add_command(label = 'Background substraction', command = 
self.back_sub)
		file.add_command(label = 'Identify Objects', command = self.iden_obj)
		file.add_command(label = 'Exit', command = self.client_exit)
		menu.add_cascade(label = 'File', menu = file)
		edit = Menu(menu)
		edit.add_command(label = 'About', command = self.about_app)
		edit.add_command(label = 'Team', command = self.showTxt)
		menu.add_cascade(label = 'Edit', menu = edit)
	def client_exit(self):
        print("Exit")
        exit()
    def back_sub(self):
        background_subtraction.main()
    def iden_obj(self):
        identify_obj.main()

    def count(self):
        count_people.main()
    def showImg(self):
        load = Image.open('../images/pic1.png')
        render = ImageTk.PhotoImage(load)

        img = Label(self, image = render)
        img.image = render
        img.place(x = 50, y=20, height = 500, width = 500)

    def showTxt(self):
        text = Label(self, text = 'Apurv | Aakash | Aishwary | Pooja | Shrikant')
        text.pack()

    def about_app(self):
        text = Label(self, text = 'This application is developed by IIIT NR Students as
a part of Project of DRDO DRUSE')
        text.pack()
root = Tk()
root.geometry("600x700")
app = Window(root)
root.mainloop()
```

background_subtraction.py

```
import numpy as np
import cv2
```

```
def main():
   cap = cv2.VideoCapture(2)
   fgbg = cv2.createBackgroundSubtractorMOG2()
   while(1):
      ret, frame = cap.read()
      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2BGRA)
      fgmask = fgbg.apply(gray)
      cv2.imshow('fgmask',gray)
      cv2.imshow('frame',fgmask)
      key = cv2.waitKey(1) & 0xFF
      if key == ord("q"):
         break
   cap.release()
   cv2.destroyAllWindows()
```

countpeople.py

```
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2
def main():
      CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
            "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
            "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
            "sofa", "train", "tvmonitor"]
      COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
      print("[INFO] loading model...")
      net = cv2.dnn.readNetFromCaffe('MobileNetSSD_deploy.prototxt.txt',
'MobileNetSSD_deploy.caffemodel')
      print("[INFO] starting video stream...")
```

```
vs = VideoStream(src=2).start()
time.sleep(2.0)
fps = FPS().start()
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)
    labels = []
    net.setInput(blob)
    detections = net.forward()
    det_conf = detections[0, 0, :, 2]
    top_indices = [i for i, conf in enumerate(det_conf) if conf >=    0.2]
    top_conf = det_conf[top_indices]
    det_indx = detections[0, 0, :, 1]
    top_label_indices = det_indx[top_indices].tolist()
    c = 0
    for i in top_label_indices:
        if(CLASSES[int(i)]=='person'):
            c = c+1
            print('person')
            for i in np.arange(0, detections.shape[2]):
                if confidence > 0.5:
                    (startX, startY, endX, endY) = box.astype("int")



                    label = "{}: {:.2f}%".format(CLASSES[idx],
                        confidence * 100)


                    cv2.rectangle(frame, (startX, startY), (endX, endY),
                            COLORS[idx], 2)
                    y = startY - 15 if startY - 15 > 15 else startY + 15
```

```
                        cv2.putText(frame, label, (startX, y),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx], 2)

        label2= "{}: {:.2f}".format('No. of Person Detected (Code By Us)',c)

        cv2.putText(frame,label2 , (10, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,255), 2
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break
        fps.update()
    fps.stop()
    print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
    cv2.destroyAllWindows()
    vs.stop()
```

face.py

```
import cv2
import sys
import logging as log
import datetime as dt
from time import sleep
def main():
  cascPath = "haarcascade_frontalface_default.xml"
  faceCascade = cv2.CascadeClassifier(cascPath)
  #log.basicConfig(filename='webcam.log',level=log.INFO)

  video_capture = cv2.VideoCapture(2)
  anterior = 0
  i=0
  while True:
        if not video_capture.isOpened():
            print('Unable to load camera.')
            sleep(5)
            pass
        ret, frame = video_capture.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(
            gray,
            scaleFactor=1.1,
            minNeighbors=5,
            minSize=(30, 30)
        )
        # Draw a rectangle around the faces
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        if anterior != len(faces):
            anterior = len(faces)
            #log.info("faces: "+str(len(faces))+" at "+str(dt.datetime.now()))
        label2= "{}: {:.2f}%".format('No. of Person', anterior)
        cv2.putText(frame, label2, (10, 40),cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,255),
2)
        # show the output frame
        # Display the resulting frame
        #cv2.putText(frame, "Number of faces detected: " + str(anterior),
cv2.FONT_HERSHEY_TRIPLEX, 0.5,  (0,0,0), 1)
        cv2.imshow('Video', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        # Display the resulting frame
        cv2.imshow('Video', frame)
    print(anterior)
    video_capture.release()
    cv2.destroyAllWindows()
```

identify_object.py

```
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

def main():
    # initialize the list of class labels MobileNet SSD was trained to
    # detect, then generate a set of bounding box colors for each class
    CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]
    COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
    # load our serialized model from disk
    print("[INFO] loading model...")
    net = cv2.dnn.readNetFromCaffe('../models/MobileNetSSD_deploy.prototxt.txt',
'../models/MobileNetSSD_deploy.caffemodel')
    # initialize the video stream, allow the cammera sensor to warmup,
    # and initialize the FPS counter
    print("[INFO] starting video stream...")
    vs = VideoStream(src=1).start()
    time.sleep(2.0)
    fps = FPS().start()
    # loop over the frames from the video stream
    while True:
        # grab the frame from the threaded video stream and resize it
        # to have a maximum width of 400 pixels
        frame = vs.read()
        frame = imutils.resize(frame, width=800)
```

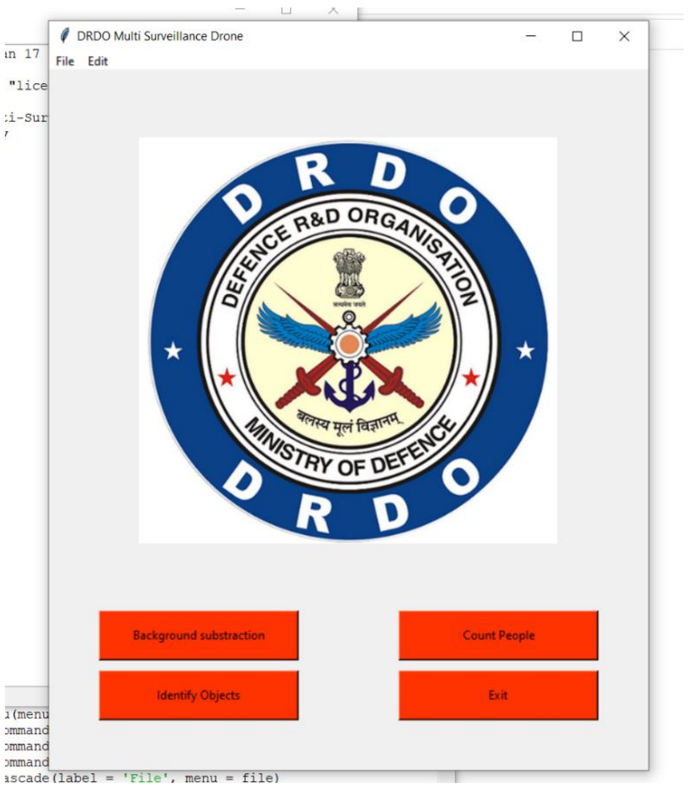
```
# grab the frame dimensions and convert it to a blob
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)
# pass the blob through the network and obtain the detections and
# predictions
net.setInput(blob)
detections = net.forward()

# 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]
        people = 0
        # filter out weak detections by ensuring the `confidence` is
        # greater than the minimum confidence
        if confidence > 0.2:
                # extract the index of the class label from the
                # `detections`, then compute the (x, y)-coordinates of
                # the bounding box for the object
                idx = int(detections[0, 0, i, 1])
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                # draw the prediction on the frame
                label = "{}: {:.2f}%".format(CLASSES[idx],
                        confidence * 100)
                if (CLASSES[idx]=='person'):
                        people = people +1
                cv2.rectangle(frame, (startX, startY), (endX, endY),
                        COLORS[idx], 2)
                y = startY - 15 if startY - 15 > 15 else startY + 15
```

```
            cv2.putText(frame, label, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

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

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

    # update the FPS counter
    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()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

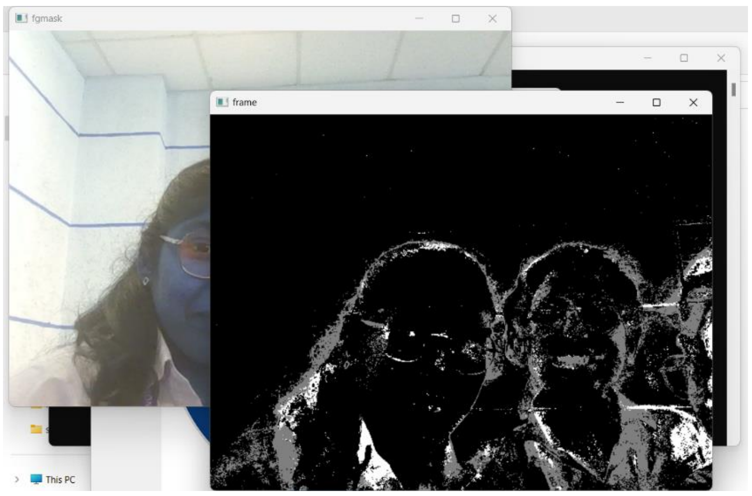
CHAPTER 6

OUTPUT RESULTS

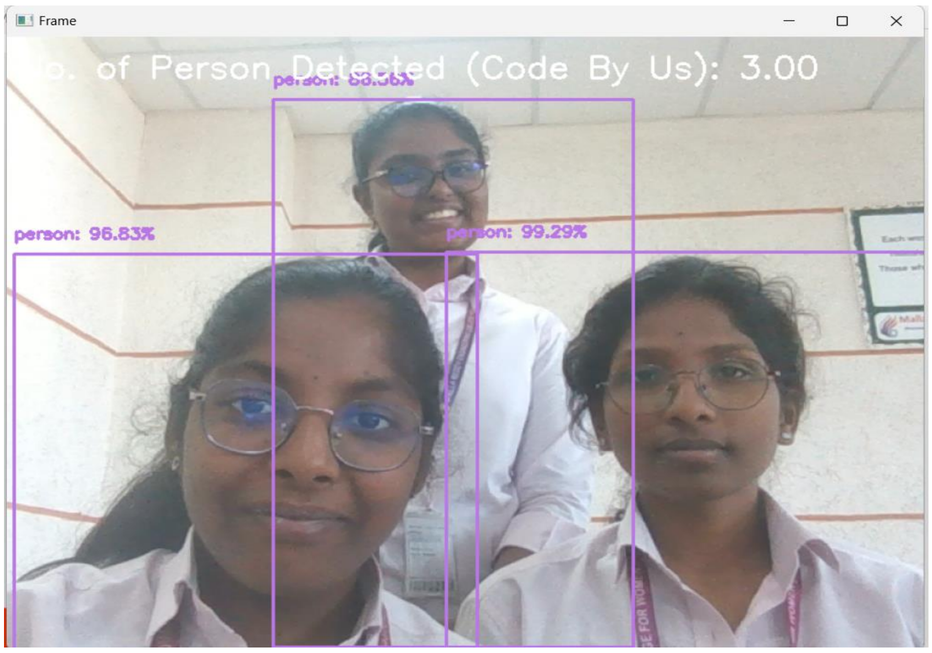
Interface:



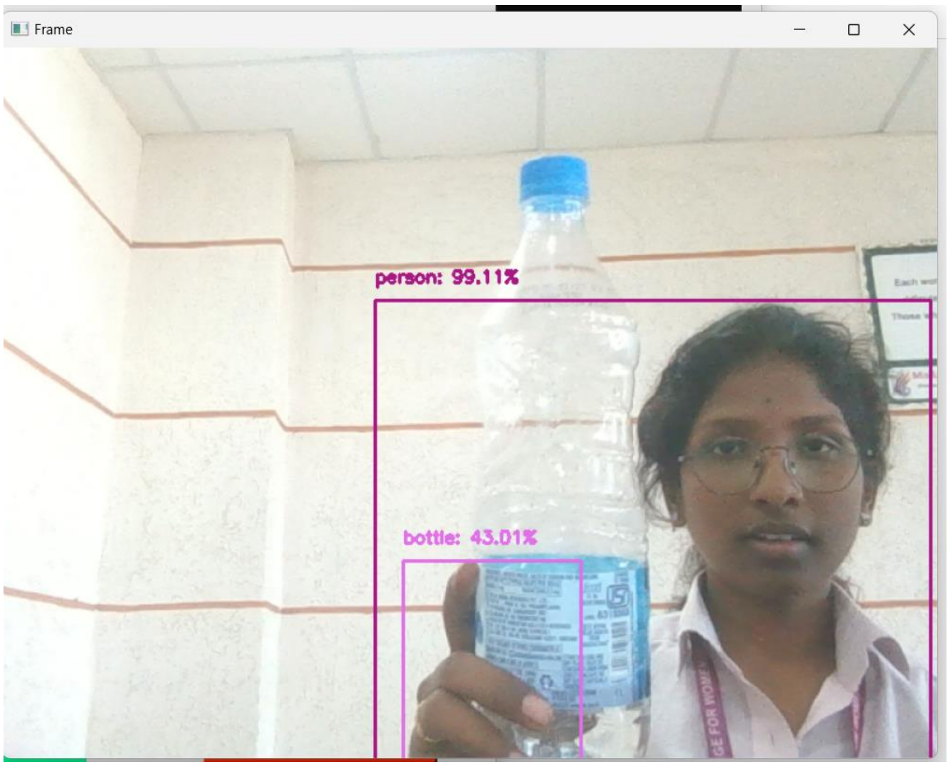
Background subtraction:



Count people:



Identify objects:



CHAPTER 7
CONCLUSION AND FUTURE SCOPE

From the project we can conclude that this system gives a unmanned solution to rescue the people in critical situations. This software can be used as CCTV, smart eye or the camera inserted in a drone. This software can be used in real time applications like rescue operations, hijacks, terrorists' attacks, wildlife monitoring, military purposes, warzones and etc.,

Future Scope:

It has a great future scope because of its wide variety of applications that can be used everywhere like Visual Surveillance of Human Activities, Visual Observation of Animals and Insects Behaviors, Visual Observation of Natural Environments, Visual Analysis of Human Activities, Visual Hull Computation, Human-Machine Interaction (HMI), Vision-based Hand Gesture Recognition, Content-based Video Coding and many more like AI-Powered Video Analytics, Foot Tracking, Animal Detection, Airport Facial Recognition, Autonomous Driving Video Surveillance.

REFERENCES

1)A. KUMAR SINGH, D. SINGH and M. GOYAL, "People Counting System Using Python," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1750-1754, doi: 10.1109/ICCMC51019.2021.9418290.

2) . Liu, J. Liu and M. Zhang, "A detection and tracking based method for real-time people counting", *2013 Chinese Automation Congress*, 2013.

3) Tsong-Yi Chen, Chao-Ho Chen, Da-Jinn Wang and Yi-Li Kuo, "A People Counting System Based on Face-Detection", *2010 Fourth International Conference on Genetic and Evolutionary Computing*, 2010.

4) A. Sharma, J. Pathak, M. Prakash and J. N. Singh, "Object Detection using OpenCV and Python," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 501-505, doi: 10.1109/ICAC3N53548.2021.9725638.

5) P Viola and M Jones, "Rapid object detection using a boosted cascade of simple features", *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, December 8-14, 2001.

6) Thorne, Brian & Grasset, Raphael. (2010). Python for Prototyping Computer Vision Applications.

7) M. Piccardi, "Background subtraction techniques: a review," 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, Netherlands, 2004, pp. 3099-3104 vol.4, doi: 10.1109/ICSMC.2004.1400815.

8) C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking," Proc. IEEE CVPR 1999, pp. 246-252, June 1999.