SUMMER INTERNSHIP REPORT

on

Abandoned Object Detection

JIIT in-house

INTERNSHIP INCHARGE
Dr. Hemant Kumar

PERIOD
5 May 2020 to 15 June 2020

SUBMITTED BY
Kapil Sharma (9917102148)

Department of Electronics and Communication Engineering
Jaypee Institute of Information Technology,
NOIDA(U.P),AUGUST 2020

CERTIFICATE

This is to certify that the summer internship report entitled, "**Abandoned Object Detection**" submitted by **"Kapil Sharma"** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Electronics and Communication Engineering of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature of Supervisor(s)

Name:Dr. Hemant Kumar
ECE Department
JIIT Sec-128
Noida-201304

DECLARATION

I **Kapil Sharma**, hereby declare that this summer internship report entitled **"Abandoned Object Detection"** submitted to **Jaypee Institute of Information Technology, Noida** for the partial fulfilment of the degree of bachelor of technology is the bonafide record of the work done by me in the due course of my summer internship and the contents and facts prepared and presented by me without any bias and are authentic to the best of my knowledge. I also declare that it has not previously formed the basis of an award for me any degree/ diploma, fellowship or other similar titles of any institute/ society.

Place : Noida

Name : Kapil Sharma
Enrollment No. : 9917102148

ACKNOWLEDGEMENT

The time I spent in doing the In-House training project from 5 May 2020 to 15 June 2020 was a memorable one for me as it was rich in experience sharing and helped me to discover my potential. I have had so many experiences and opportunities that I believe will forever shape and influence my professional skills while fostering personal growth and development.

With a sage sense of gratitude, I acknowledge various people who directly or indirectly contributed to the development of this work and who influenced my thinking, behavior and acts during training. I express my deep sense of gratitude to **Training Incharge Dr. Hemant Kumar** for his technical support and constant supervision which contributed immensely to my development. I also thank him for his guidance which was a remarkable force that enabled me to complete the training program.

I would like to thank my parents and my friends whose love and guidance are with me in whatever I pursue. Nobody has been more important to me than the members of my family and friends.

I also express my sincere gratitude to **Dr. Satyendra Kumar, T&P Coordinator**, Jaypee Institute of Information and Technology, for his stimulating guidance, continuous encouragement and evaluating me on this training.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 General Background

Computer vision is an interdisciplinary field which has been gaining a huge amount of attraction in recent years. The goal of computer vision is to teach a computer to extract information from image as close as possible to humans who can naturally understand everything by seeing the same image. In order to achieve this goal, one of the integral parts of computer vision is object detection and recognition that deals with localizing a particular object region or contour from an image or video and then classifying it. Object detection and recognition have been applied to different problem domains over so many years, whether the objects were handwritten characters , house numbers , traffic signs , objects from the VOC dataset, objects from the 1000-category ImageNet dataset or Caltech-101 dataset. In recent days, object detection is being used for so many applications. There are some state-of-the-arts which work for different types of object detection such as flower detection , fruit detection , food segmentation and detection ,cats and dogs detection etc.The main goal of all these detection algorithms is to obtain higher efficiency and cover different complex use cases by overcoming different limitations.
During the last few years, abandoned object detection has emerged as a hot topic in the video-surveillance community. As a consequence, many systems has been proposed for automatic monitoring of public and private places, while addressing several challenges affecting detection performance. Due to the complexity of these systems, researchers often address independently the different analysis stages such as foreground segmentation, stationary object detection, and abandonment validation. Detection of abandoned objects is a crucial task to ensure public security. Due to the increasing number of surveillance devices and the ensuing data flood, it has become impossible to manually process all the feeds from a surveillance camera. Automatic detection of abandoned objects is one step towards providing better measures for public safety.[1]

1.2 Problem Statement

Presently cameras are being used only for storing recordings with imprinted time and date.
The existing device is not proactive and is not efficient to alarm the security personnel who can immediately check the incident.
A system is being presented which is smart enough to detect an abandoned object and mark it on the screen.

1.3 Work Done

1. We are processing the live feed of the CCTV camera with image processing .
2. If a person is releasing off some piece of luggage the camera will catch the activity .
3. This frames are been detected and been image processed by Edge detection. The processing is done by the OpenCV.
4. If the bag is untouched for a some period of time the analyser decides and further gives an alarm to the authority.[2]

The basic steps that are discussed above which are used to perform the abandoned objects detection is shown in Figure. 1.1.

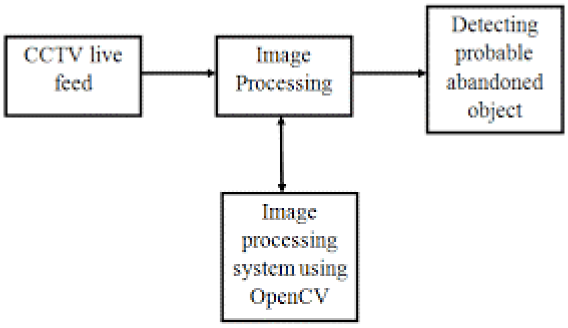
CCTV live feed
Image Processing
Detecting probable abandoned object
Image processing system using OpenCV

Figure 1.1: Shows how the abandoned object gets detected

1.4 Canny Edge Detection

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stages.

1.4.1 Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5×5 Gaussian filter.

1.4.2 Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient(G) = \sqrt{G_x^2 + G_y^2} \tag{1.1}$$

$$Angle(\theta) = \tan^{-1}(\frac{G_y}{G_x}) \tag{1.2}$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

1.4.3 Non-maximum Suppression

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient.

1.4.4 Hysteresis Thresholding

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.

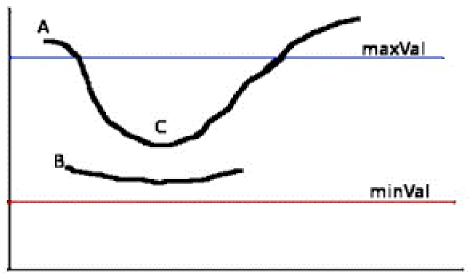

Figure 1.2: Showing Threshold Hystresis

As shown in figure 1.2,the edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.
As shown in Figure 1.3,the resultant image shows the edges got detected in a frame. Here the threshold values that has been set minVal=50 , maxVal=100.

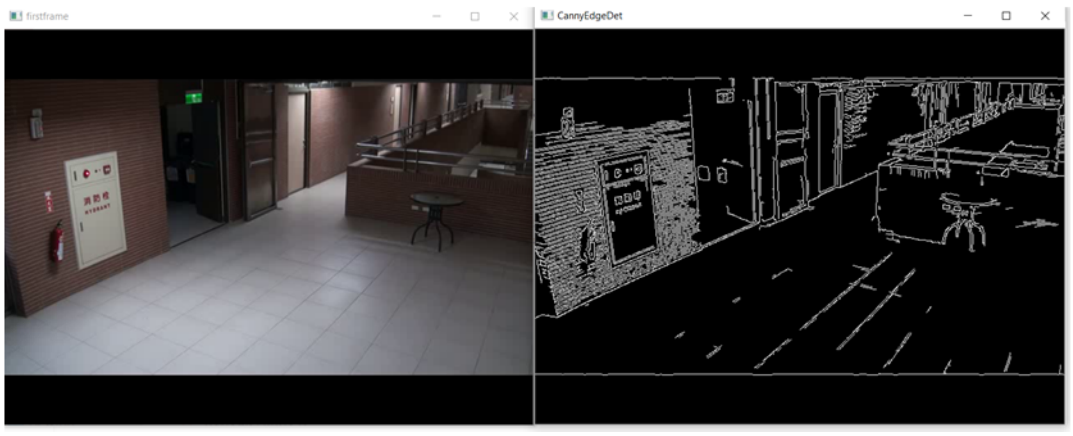

Figure 1.3: The result of canny edge detected Frame

1.5 Image Processing

Image processing is an area that keeps on growing, with new applications being developed with every passing minute. It is captivating area which has evolved with applications in various areas like entertainment industry, space program etc. The most concerning prospect of this data transition is its capability to transmit and acquire complex information that oversteps normal written text. Visual information sent in the way of images, has evolved as one of the major mode of communication in the 21st century. Description of image processing in detail is described as below.

It is a process of converting an image into the digital image to obtain the enhancement in image or to select some effective information from it. It uses any type of signal processing in which image is input and output of processing may result into an image or features of an image. It includes vast applications in various trades such as business, medical, industries etc.
Following are the main steps including in image processing:

1. Acquiring an image using scanner or by digital cameras.
2. Manipulation and analysis of the image is performed, which includes data compression and enhancement of image.
3. Obtained outcome of image processing can be an image or the details of the characteristics of image based upon the analysis of an input image.

Chapter 2

Basics of the Works

2.1 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.[3]
Background subtraction is a major preprocessing steps in many vision based applications. For example, consider the cases like visitor counter where a static camera takes the number of visitors entering or leaving the room, or a traffic camera extracting information about the vehicles etc. In all these cases, first you need to extract the person or vehicles alone. Technically, you need to extract the moving foreground from static background.
As shown in Figure 2.1 the background of the frame gets subtracted from the image and hence the foreground objects (people standing) are extracted while the background remains black.

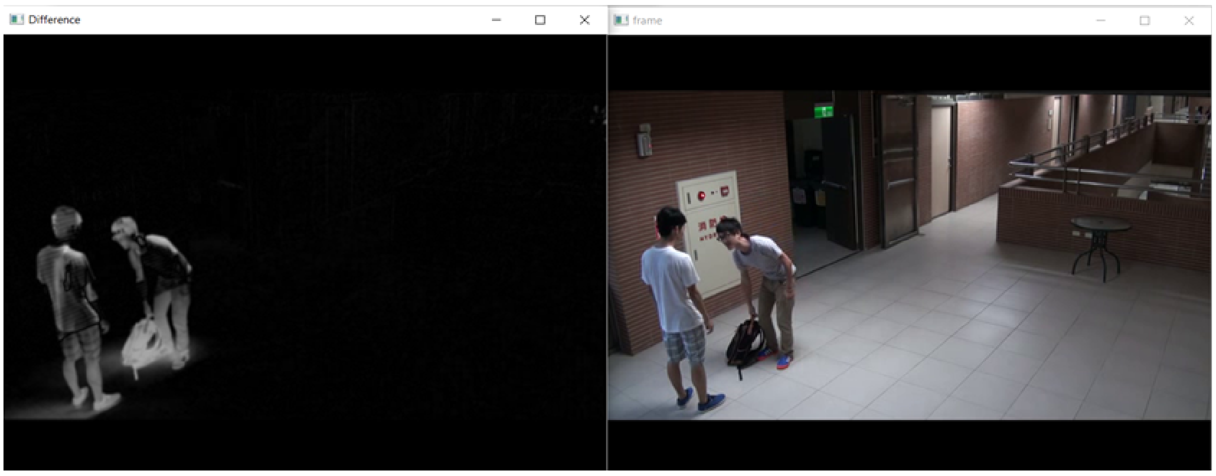

Figure 2.1: The result of background subtracted image

2.2 Morphological Transformation

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play.
• Erosion:
The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).
erosion = cv2.erode(img,kernel,iterations = 1)
•Dilation:
It is just opposite of erosion. Here, a pixel element is '1' if atleast one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.
dilation = cv2.dilate(img,kernel,iterations = 1)
• Opening:
Opening is just another name of erosion followed by dilation. It is useful in removing noise. Here we use the function, cv.morphologyEx().
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
• Closing:
Closing is reverse of Opening, Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

2.3 Finding Contours

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity or colour. Contours come handy in shape analysis, finding the size of the object of interest, and object detection. In OpenCV, finding contours is like finding white object from black background. So, object to be found should be white and background should be black.
OpenCV has findContour() function that helps in extracting the contours from the image. It works best on binary images. There are three arguments in cv.findContours() function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy. Contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.
We know contours are the boundaries of a shape with same intensity. It stores the (x,y) coordinates of the boundary of a shape. But the number of coordinates to be stored is specified by the contour approximation method.
• CV2.CHAIN_APPROX_NONE stores absolutely all the contour points. That is, any 2 subsequent points (x1,y1) and (x2,y2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, max(abs(x1-x2),abs(y2-y1))==1.

• CV2.CHAIN_APPROX_SIMPLE compresses horizontal, vertical, and diagonal segments and leaves only their end points, thereby saving memory. For example, an up-right rectangular contour is encoded with 4 points.

2.4 Moments

Image moments help you to calculate some features like center of mass of the object, area of the object etc.
The function cv.moments() gives a dictionary of all moment values calculated.
From this moments, you can extract useful data like area, centroid etc. Centroid is given by the relations, Cx=M10/M00 and Cy=M01/M00. This can be done as follows:
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
Here we draw the contours on a frame using findContour(). We applied to find contours on canny edge detected image. As shown in Figure 2.2 random contours are being detected in a frame.

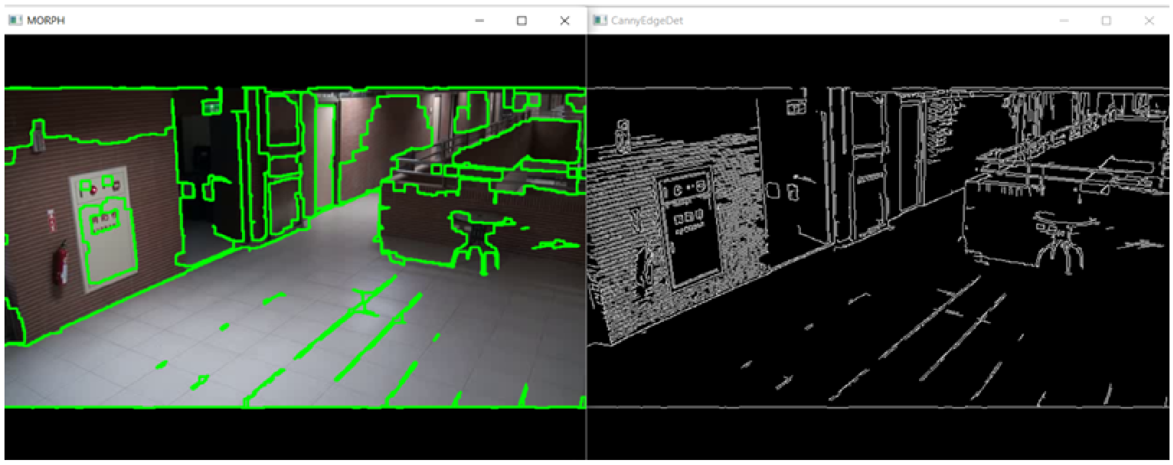

Figure 2.2: Random Contours detected in a Frame

Chapter 3

System Requirements and Software Requirements

3.1 Functional Requirements

1. Processor Required: Intel Core i5 and above
2. RAM Required: 4 Gb and above

3.2 Software Requirements

1. Operating System like Windows (64- bit), Mac, Linux
2. Python 3.5 and above

3.2.1 Operating System

An operating system (OS) is a system software that manages computer hardware, software resources, and provides common services for computer programs. Every computer must have at least one OS to run other programs. An application like Chrome, MS Word, Games, etc. needs some environment in which it will run and perform its task. The OS helps to communicate with the computer without knowing how to speak the computer's language. It is not possible for any user to use any computer or mobile device without having operating system.

3.2.2 Python

Python is an example of a high level language such as C++, PHP, Pascal, C#, and Java. It is a powerful programming language which is easy to learn which has a simple but effective approach to object-oriented programming. It was created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code reliability with its notable use of significant whitespace.
• Python is an interpreted, high level, general purpose programming language.
• It is a dynamic yet modern object -oriented programming language which can be used to do a lot of things and is easy to learn.
• It is a high level language that means this language is closer to humans than computer.
• It is a very flexible language and is also known as a general purpose programming language.

• Python is object -oriented means it regards everything as an object.
• Python being an interpreted language does not need to be complied. For example java programming language.
• Python has been used in a lot of places like in creating games for statistical data and visualization, speech and face recognition.
• Python's name is derived from the British comedy group Monty Python.

3.3 Python Packages Used

1. **OpenCV**
   OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development. OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV is used for all sorts of image and video analysis, like facial recognition and detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more.

2. **Numpy**
   Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal.

3. **Collections**
   This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, dict, list, set, and tuple.
   • Counter objects
   A counter tool is provided to support convenient and rapid tallies. A counter is a dict subclass for counting hashable objects. It is an unordered collection where elements are stored as dictionary keys and their counts are stored as dictionary values. Counts are allowed to be any integer value including zero or negative counts. The Counter class is similar to bags or multisets in other languages.
   • Defaultdicts
   Returns a new dictionary-like object. defaultdict is a subclass of the built-in dict class. It overrides one method and adds one writable instance variable. The remaining functionality is the same as for the dict class.
   The first argument provides the initial value for the default_factory attribute; it defaults to None. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

Chapter 4

Project Implementation And Description

An abandoned object is an object left by its owner and not re-attended within a predefined time. Most existing abandoned object detection algorithms use foreground information generated from background models. The background subtraction technique is one of the effective ways to extract foreground information from images.[4]

In this project, I applied a method to detect abandoned object from surveillance video. In first step, foreground objects are extracted using background subtraction. In second step, static objects are detected by using contour features of foreground objects of consecutive frames. In third step, detected static objects are classified into human and non-human objects by using edge based object recognition method which is capable to generate the score for full or partial visible object. Nonhuman static object is analyzed to detect abandoned object. Experimental results show that proposed system is efficient and effective for real-time video surveillance.

4.1 Results

Scenario 1 - In Figure 4.1 we got the refrence frame from the CCTV feed. It only has permanent objects and is used in background subtraction.

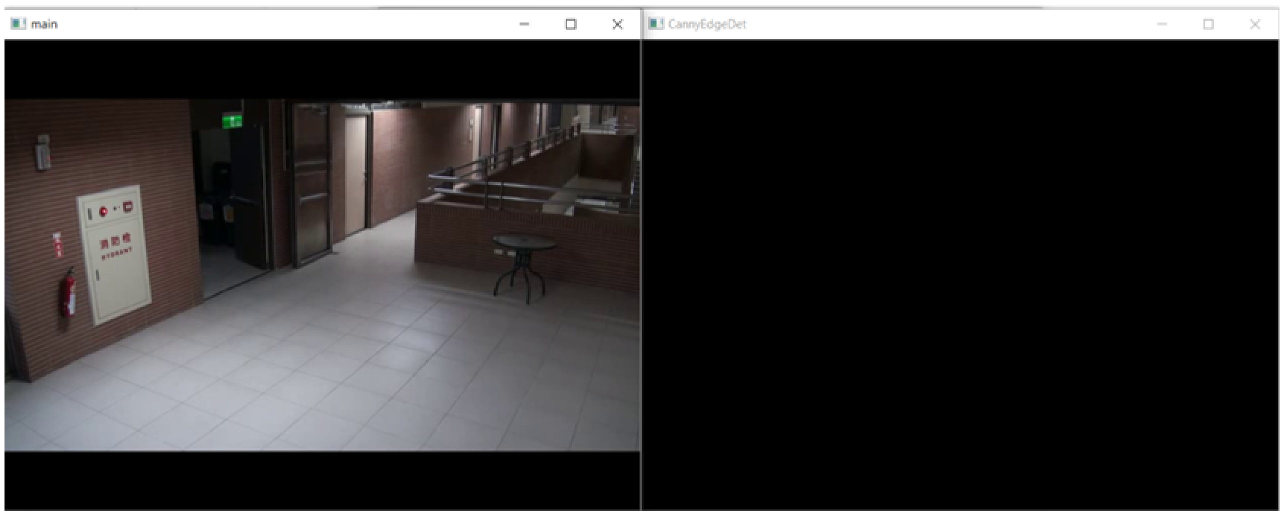

Figure 4.1: Reference frame with only the permanent objects

Scenario 2 - Applied Canny edge detection to foreground Objects, detected using background subtraction. After the edges being detected we applied the closing morphological transformation to remove the background noise from the foreground object. And hence the Resultant detected image is shown in Figure 4.2 .

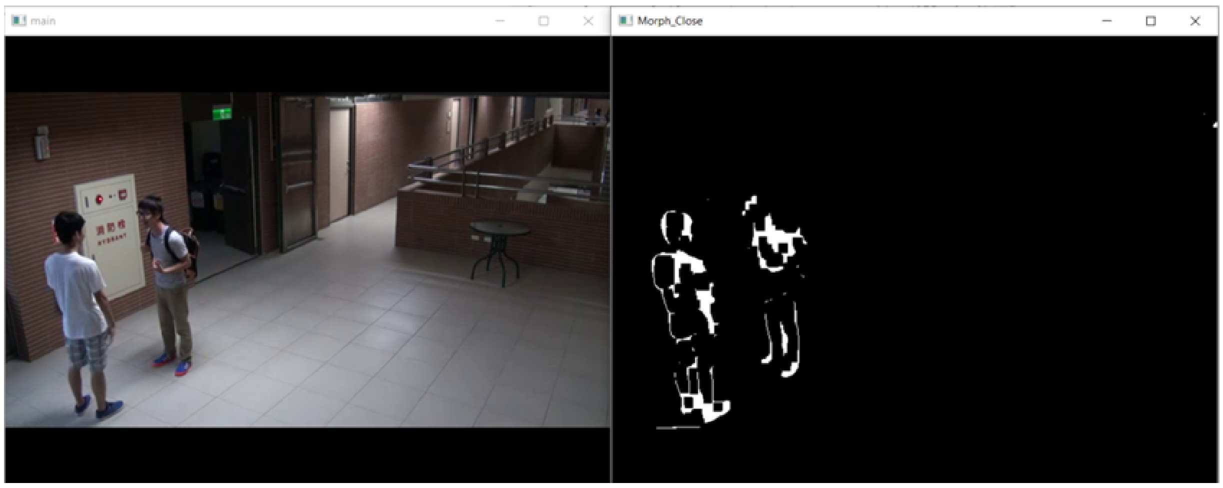

Figure 4.2: Foreground objects are being detected

Scenario 3 - After we found the edged of foreground objects we draw contours around the objects. This helps us to locate the objects that has been stationary for specific time.
As shown in Figure 4.3 we got the stationary object detected as a threat object along with its area.

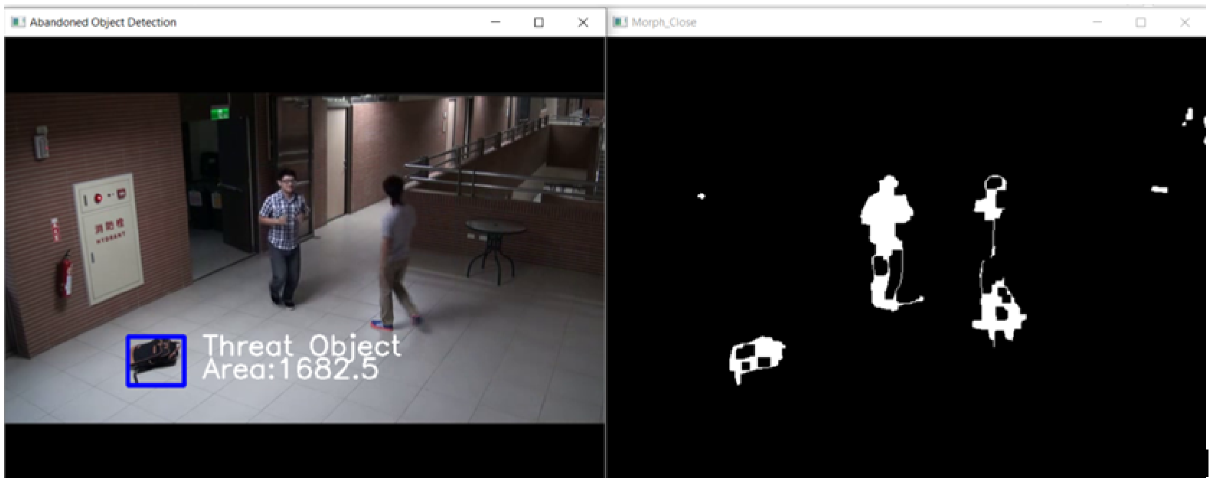

Figure 4.3: Object has not been moved and hence a message is displayed

Chapter 5

Discussion , Conclusion And Recommendations For Further Work

5.1 Discussions

As crime rate is increasing day by day, video surveillance has become a need. It helps security monitor the day to day events .Incidents of bomb blasts by terrorists at busy public places are among the prime concerns to security agencies across the globe.
All of these places are well equipped with CCTV's and are not able to make the best use of them. We hear about bomb attacks which are held all over the country, which is a huge amount of loss in terms of both property and life.
It becomes a crucial task to have safety all over public places to prevent massive destruction caused by the bomb placed in Public places .
To stop this incidences ,I was motivated to help us detect an unwanted placed object and providing an alarm to the Security.The framework of Abondoned Object Detection is shown in figure 5.1.

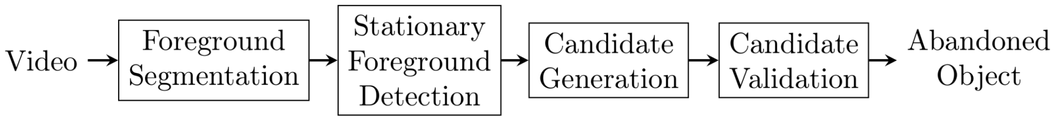

Figure 5.1: Framework for abandoned Object Detection

The project is made to enhance the security of the public in open spaces. By applying the above framework to the CCTV feed we can detect the abandoned object that has not been moved for specific time.

5.2 Conclusion

• This system introduces a framework to discover the abandoned objects in the public areas such as railway stations, shopping malls. This system detects and classifies object into different classes and also detects abandoned objects.
• This system becomes very helpful to the guards monitoring the public places as they get alerted before any dangerous circumstances occur.

Bibliography

[1] P. S. P. Divya C. Patil, "Abandoned object detection with region of interest", *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 6, no. 6, pp. 2231–2307, January 2017.

[2] A. Bhondave, "Suspicious object detection using back-tracing technique", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 1, pp. 406–408, 2016.

[3] Y. S. Pritee Gupta and M. Gupt., "Moving object detection using frame difference, background subtraction and sobs for video surveillance application", pp. 151–156, 2014.

[4] K. M. S. A. N. Ajit C Joshi Keerthan R, "A survey on abandoned object detection", *International Journal of Emerging Technologies and Innovative Research (www.jetir.org)*, vol. 4, no. 5, pp. 263–265, May 2017.

Appendix A

Code

```
import numpy as np
import cv2
from collections import Counter, defaultdict

    def empty(a):
pass

    #———————————————
#size the window first
#———————————————
cv2.namedWindow('CannyEdgeDet',cv2.WINDOW_NORMAL)
cv2.namedWindow('Abandoned Object Detection',cv2.WINDOW_NORMAL)
cv2.namedWindow('Morph_CLOSE',cv2.WINDOW_NORMAL)
cv2.namedWindow('Parameters',cv2.WINDOW_NORMAL)
cv2.createTrackbar('Threshold1','Parameters',10,500,empty)
cv2.createTrackbar('Threshold2','Parameters',200,500,empty)

    # location of video
file_path =r'/Users/KAPILSHARMA/Desktop/video1.avi'

    cap = cv2.VideoCapture(file_path)
_,firstframe = cap.read()
firstframe_gray = cv2.cvtColor(firstframe, cv2.COLOR_BGR2GRAY)
firstframe_blur = cv2.GaussianBlur(firstframe_gray,(21,21),0)

    consecutiveframe=20

    track_temp=[ ]
track_master=[ ]
track_temp2=[ ]

    top_contour_dict = defaultdict(int)
obj_detected_dict = defaultdict(int)

    frameno = 0
while (cap.isOpened()):
ret, frame = cap.read()
cv2.imshow('main',frame)
```

```
    if ret==0:
break

    frameno = frameno + 1

    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
frame_blur = cv2.GaussianBlur(frame_gray,(21,21),0)

    frame_diff = cv2.absdiff(firstframe, frame)

    #Canny Edge Detection
threshold1 = cv2.getTrackbarPos(”Threshold1”, ”Parameters”)
threshold2 = cv2.getTrackbarPos(”Threshold2”, ”Parameters”)
edged = cv2.Canny(frame_diff,threshold1,threshold2) #any gradient between 10 and
200 are considered edges
cv2.imshow(’CannyEdgeDet’,edged)
kernel2 = np.ones((5,5),np.uint8) #higher the kernel, eg (10,10), more will be eroded
or dilated
thresh2 = cv2.morphologyEx(edged,cv2.MORPH_CLOSE, kernel2,iterations=2)
cv2.imshow(’Morph_Close’, thresh2)

    #Create a copy of the thresh to find contours
(cnts,_) = cv2.findContours(thresh2.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPL

    mycnts =[ ] # every new frame, set to empty list.
# loop over the contours for c in cnts:

    area=cv2.contourArea(c)
# Calculate Centroid using cv2.moments
M = cv2.moments(c)
if M[’m00’] == 0:
pass
else:
cx = int(M[’m10’]/M[’m00’])
cy = int(M[’m01’]/M[’m00’])

    #——————————————————————————-
# Set contour criteria
#——————————————————————————-
if cv2.contourArea(c) ¡ 200 or cv2.contourArea(c)¿20000:
pass
else:
mycnts.append(c)

    # compute the bounding box for the contour, draw it on the frame,
(x, y, w, h) = cv2.boundingRect(c)
#cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
#cv2.putText(frame,’C %s,%s,%.0f’%(cx,cy,cx+cy), (cx,cy),cv2.FONT_HERSHEY_SIMPLEX,
1, (0,0,0),2)
```

```
    #Store the cx+cy, a single value into a list ; max length of 10000
#Once hit 10000, tranfer top 20 points to dictionary ; empty list
sumcxcy=cx+cy

    #track_list.append(cx+cy)
track_temp.append([cx+cy,frameno])

    track_master.append([cx+cy,frameno])
countuniqueframe = set(j for i, j in track_master) # get a set of unique frameno. then
len(countuniqueframe)

    #——————————————————————————-
# Store history of frames ; no. of frames stored set by 'consecutiveframe' ;
# if no. of no. of unique frames ¿ consecutiveframes, then 'pop or remove' the earliest
frame ; defined by
# minframeno. Objective is to count the same values occurs in all the frames under
this list. if yes,
# it is likely that it is a stationary object and not a passing object (walking person)
# And the value is stored separately in top_contour_dict , and counted each time.
This dict is the master
# dict to store the list of suspecious object. Ideally, it should be a short list. if there
is a long list
# there will be many false detection. To keep the list short, increase the 'consecu-
tiveframe'.
# Keeping the number of frames by remove the minframeno.; but since hard to re-
move, rather formed a new list without
# the minframeno.
#——————————————————————————-
if len(countuniqueframe)¿consecutiveframe or False:
minframeno=min(j for i, j in track_master)
for i, j in track_master:
if j != minframeno: # get a new list. omit the those with the minframeno
track_temp2.append([i,j])

    track_master=list(track_temp2) # transfer to the master list
track_temp2=[ ]

    #count each of the sumcxcy
#if the same sumcxcy occurs in all the frames, store in master contour dictionary,
add 1

    countcxcy = Counter(i for i, j in track_master)
#print countcxcy
#example countcxcy : Counter(504: 46, 537: 30, 530: 1, 523: 1, 488: 1)
#if j which is the count occurs in all the frame, store the sumcxcy in dictionary, add
1
for i,j in countcxcy.items():
if j¿=consecutiveframe:
top_contour_dict[i] += 1
```

```
    if sumcxcy in top_contour_dict:
if top_contour_dict[sumcxcy]¿100:
cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 3)
cv2.putText(frame,'%s'%('Threat Object'), (x+w+20,y+20),cv2.FONT_HERSHEY_SIMPLEX,
1, (255,255,255),2)
cv2.putText(frame,'Area:'+str(area), (x+w+20,y+45),cv2.FONT_HERSHEY_SIMPLEX,
1, (255,255,255),2)
print ('Detected : ', sumcxcy,frameno, obj_detected_dict)

    # Stored those objects that are detected, and store the last frame that it hap-
pened.
# Need to find a way to clean the top_contour_dict, else contour will be detected after
the
# object is removed because the value is still in the dict.
# Method is to record the last frame that the object is detected with the Current
Frame (frameno)
# if Current Frame - Last Frame detected ¿ some big number say 100 x 3, then it
means that
# object may have been removed because it has not been detected for 100x3 frames.

    obj_detected_dict[sumcxcy]=frameno

    for i, j in list(obj_detected_dict.items()):
if frameno - obj_detected_dict[i]¿200:
print ('PopBefore',i, obj_detected_dict[i],frameno,obj_detected_dict)
#print ('PopBefore : top_contour :',top_contour_dict)
obj_detected_dict.pop(i)

    # Set the count for eg 448 to zero. because it has not be 'activated' for 200 frames.
Likely, to have been removed.
top_contour_dict[i]=0
print ('PopAfter',i, obj_detected_dict[i],frameno,obj_detected_dict)
#print ('PopAfter : top_contour :',top_contour_dict)

    cv2.imshow('Abandoned Object Detection',frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
break

    cap.release()
cv2.destroyAllWindows()
```