RAMAIAH
Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

(AUTONOMOUS INSTITUTE AFFILIATED TO VTU)

M. S. R. I. T. POST, BANGALORE – 560054

2021-2022

Mini Project Report

"Automated Way of Vehicle Theft Detection in Parking Facilities"

for the subject

Computer Vision (ISE555)

In Fifth Semester

Submitted by:

Jeevika K	**1MS20IS055**
Maitri P Tadas	**1MS20IS067**
Owais Iqbal	**1MS20IS081**
Prapti Bopana	**1MS20IS088**

Submitted to:

Dr. Megha P Arakeri

Associate Professor,Dept. of ISE

TABLE OF CONTENTS

Abstract

Security of parked cars against theft is a long existing concern. The car owners or parking lot operators are worried about having the vehicles stolen from parking lots, so they use CCTV cameras in parking lots to detect theft. We propose a system in which several video cameras are installed in different parking lots. The proposed computational model will capture and analyze one frame per second. There will be several videos running simultaneously from different parking areas which will be monitored by humans, so to avoid human error, the monitor will give a notification if there is any movement detected after the video footage is processed. We present an automated way of detecting vehicle theft as it happens using moving object detection. The detected edges of the output should give a clear image of the moving object from the video and finds the edge change ratio. The security personnel or the parking lot operator gets notified about the movement. Canny is one of the modern Edge detection techniques and choosing this one over other methods is because of the double thresholding and its better performance.

Introduction

Security and Surveillance of any locality is a prime concern of an individual or a group of people to detect intrusion or any other unwanted phenomena. The use of CCTV is getting popular because of the availability and increasing advancement of image processing from video. To detect a crime or intrusion, real time image processing can be an optimal solution. In many environments, the retrieval of the video or the purpose of using a video surveillance might be different. Depending on the nature of the need for the video surveillance, the image processing method varies. Object tracking in video surveillance systems are widely used by security agencies to have real time monitoring and to detect potential security threats by instant.

There are many methods of detecting images but the focus on edge detection is explained here. The edges of a particular object are detected, and it will be extracted from the background. If the image contains some noise the image must go through a filter, most likely a Gaussian filter. In the proposed system, the extraction of the edge of the object is done by Canny Edge Detection which can be used for object classification and detection of a complex video. The detection of an object's actual shape is not easy because issues like light variation, shadow and noise affect the quality of the video as well as the image. An individual image is nothing but a frame. In this proposed system, each frame is compared to a particular background image and subtracted from it to detect whether there is any change in the edges.

By comparing the Canny image with these images, we can see significant improvement in detail of the principal edges and at the same time more rejection of irrelevant features in our proposed Canny result. All the edges form closed loops which gives 'spaghetti' effect is it's serious drawback and thus the method gives irrelevant features. These irrelevant features are caused by zero crossing which can be recovered by Canny method through double thresholding. The quality of the lines regarding continuity, thinness and straightness is superior in the Canny image.

Most of the apartments nowadays have large parking areas and the security of the parked cars have become a prior concern. We propose a system in which several video cameras are installed in different parking lots. The proposed computational

model will capture and analyze one frame per second. The detected edges of the output should give a clear image of the moving object from the video. As there will be several videos running simultaneously from different parking areas which will be monitored by humans, so to avoid human error the monitor will give a notification if there is any movement detected after the video footage is processed.

Methodology

1. Capturing Video

The input data is a real time video signal that is captured by a closed circuit camera. The system allows an automated way to detect a movement of a vehicle which is already parked by the driver and for a certain period of time the car is not supposed to move from the parking lot. Parking lots must have real time video surveillance and uninterrupted video signal to access an efficient and systematic approach for continuous observation. This video signal will go through various computational modules so that the image acquisition can be done. The first step is to extract the images from the video for further analysis.

2. Extracting Images from the Video

A video is nothing but a series of images that are often referred to as frames. All you need to do is loop over all the frames in a video sequence, and then process one frame at a time. The proposed computational model will capture and analyze 1 frame per second. At run time, the video automatically extract the images which are segmented to frames analogous to the surveillance area having the static camera. Each frame is then classified to resize the scaling of the image to be processed.

3. Conversion to Grayscale

The frames extracted from the real time video corresponding to the parking lot are in RGB form. Retaining the color information proved difficult on edge detection and also affects the processing time. Though the Canny method gives similar results for both RGB and Grayscale, to avoid the complexity, it is preferred to use Gray-scale images to achieve better results for edge detection. To determine the edges each frame should be converted

into the Gray-scale form first. Images that carry only intensity information and where each pixel is a single sample is called a Grayscale image.

4) Edge Detection

Edge detection is an image processing technique for finding the boundaries of objects within images, to track the foreground and background edge features. An edge is the arrangement of the curved lines arranged in a typical manner where the brightness changes sharply. It works by detecting discontinuities in brightness. This step is to be executed so that the edges of the vehicles are detected and extracted from the background. The detection of an object's actual shape is not easy because issues like light variation, shadow and noise affect the quality of the video as well as the image. Canny Edge Detector presents better performance against these obstacles as it goes through double thresholding. Canny method has also proven more effective for object classification and detection of a complex video. The edges of the objects whether it is moving or in static state, is to be detected through the Canny operator. The Canny Edge Detection methodology can itself be subdivided into following steps-

a) Noise Reduction

Edge detection results are highly sensitive to image noise, so noise removal is a prerequisite for efficient edge detection. One way to get rid of the noise on the image, is by applying Gaussian blur to smooth it. To do so, image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…). The image should go through a Gaussian filter to reduce the noise which is produced by the false edges.

b) Determination of Intensity Gradient

Edges correspond to a change of pixels' intensity. To detect it, the

easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y). An intensity gradient of Canny grayscale image is found where a sharp change is detected in the intensity. The Canny algorithm finds the edges by determining the gradients of an image. In the result we see that some of the edges are thick and others are thin.

c) Suppression of Spurious Results

Spurious response corresponds to the noise that is created from false edges. Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges.The principle is that the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.The edge strength of the current pixel is checked in both positive and negative direction and the value will be preserved if it is larger than another mask having same direction, other values will be suppressed.

d) Determination of Potential Edges

The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant.High threshold is used to identify the strong pixels (intensity higher than the high threshold).Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold).All pixels having intensity between both thresholds are flagged as weak.

e) Determination of Actual Edges

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one .

5. Comparison of Two Consecutive Frames

The algorithm to do so has three parts which includes the selection of an optimum reference, then performing the arithmetic subtraction operation and finally selecting an appropriate threshold. An individual image is a frame. A reference frame is nothing but a frame which is temporarily selected as the background image for the adjacent frames with dynamic pattern. Each frame is compared to a particular background image and subtracted from it to detect whether there is any change in the edges. The module will then compute the edge change ratio between two consecutive frames. From the difference between current frame and previous frame, a moving object can be detected.

First background subtraction method is applied for detecting the moving object and then Gaussian mixture model(GMM) features are used for tracking the objects. The subtraction value is compared with threshold which is empirically fixed. The algorithm has experimentally shown to be quite accurate and effective in detecting a single moving object even under bad lighting conditions or shadow regions. If frame contains unwanted information(noise) which may lead to inaccurate result for tracking, the median filter is used to remove the noise

6. Decision

Finally the model takes its decision by using all the values of edge change ratio and detects if the object has any movement. Moving object detection can be perceived by differencing the consecutive frames. We aim to detect movement of a vehicle which is already parked by the driver for a set period of time and the car is not supposed to move from the parking lot within that time. In this span of time CCTV camera is capturing the real time video in a parking lot. Notification of a movement of vehicle is provided to the security personnel to discern an immediate action against possible theft detection.

Implementation

get_first_frame.py

```
import cv2 as cv


# videoPath = "carPark.mp4"

videoPath = "park3.webm"


def getFirstFrame(videofile):

    vidcap = cv.VideoCapture(videofile)

    success, image = vidcap.read()

    image = cv.resize(image, (int(image.shape[1]*1.5), int(image.shape[0]*1.5)),
interpolation=cv.INTER_CUBIC)

    if success:

        cv.imwrite("first_frame.jpg", image)  # save frame as JPEG file


getFirstFrame(videoPath)
```

ParkingSpacePick.py

```
import cv2
import pickle


width, height = 42, 92
x = 32
y = 309



try:
    with open('CarPos', 'rb') as f:
        posList = pickle.load(f)

except:
    posList = []



def mouseClick(events, x, y, flags, params):
    if events == cv2.EVENT_LBUTTONDOWN:
        posList.append((x, y))

    if events == cv2.EVENT_RBUTTONDOWN:
```

```
        for i, pos in enumerate(posList):
            x1, y1 = pos
            if x1 < x < x1 + width and y1 < y < y1 + height:
                posList.pop(i)

    with open('CarPos', 'wb') as f:
        pickle.dump(posList, f)



while True:
    img = cv2.imread('first_frame.jpg')
    x = 32
    for pos in posList:
        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), (255, 0, 255), 2)
    # while x < img.shape[1]:
    #     cv2.rectangle(img, (x, y), (x + width, y + height), (255, 0, 255), 2)
    #     x+= int(width*1.0476191)

    cv2.imshow("Image", img)
    cv2.setMouseCallback("Image", mouseClick)
    if cv2.waitKey(1) & 0xFF==ord('d'):
```

```
        break
```

parkSpace2.py

```
import cv2
import pickle
import cvzone
import numpy as np
import pyzbar.pyzbar as pyzbar

cap = cv2.VideoCapture('park3.webm')

cam  = cv2.VideoCapture(0)

with open('CarPos', 'rb') as f:
    posList = pickle.load(f)

posList.sort(key = lambda item: item[1])

x = False

width, height = 42, 92
```

```
def detectActivity(imgPro, prevFrame):
    global move_hist
    frame_diff_all = cv2.absdiff(imgPro, prevFrame)
    imgMedian = cv2.medianBlur(frame_diff_all, 5)
    kernel = np.ones((3, 3), np.uint8)
    frame_diff_all = cv2.dilate(imgMedian, kernel, iterations=1)

    id = 0

    for pos in posList:
        id+=1
        x, y = pos

        frame_diff = frame_diff_all[y:y + height, x:x + width]

        count = cv2.countNonZero(frame_diff)

        if count < 250:
            color = (0, 255, 0)
            thickness = 5
```

```
        else:
            color = (0, 0, 255)
            thickness = 2
            move_hist.add(str(id))
            if str(id) not in allowed:
                cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)

        if len(move_hist.difference(allowed)) > 0:

            move_hist = move_hist.difference(allowed)
            cvzone.putTextRect(img, f'Possible Theft detected at space ID {",
".join(move_hist)}', (25, 50), scale=2,
                               thickness=2, colorR=(0,0,200))

        # cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)

        cvzone.putTextRect(img, " id: "+ str(id), (x, y + height - 3), scale=0.75,
                           thickness=1, offset=0, colorR=(0,0,0))



img = cv2.imread('first_frame.jpg')
```

```
prevGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
prevBlur = cv2.GaussianBlur(prevGray, (3, 3), 1)
prevCanny = cv2.Canny(prevBlur, 100, 100)

move_hist = set()
allowed = set()

reference = prevCanny.copy()
frame_count = 0
last_frame_count = -50
while True:
    frame_count += 1
    if cap.get(cv2.CAP_PROP_POS_FRAMES) ==
cap.get(cv2.CAP_PROP_FRAME_COUNT):
        print("--------------------start--------------------")
        move_hist = set()
        prevCanny = reference.copy()
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        frame_count = 0
        last_frame_count = -50

    next, cam_img = cam.read()
```

```
    cam_img = cv2.resize(cam_img, (int(cam_img.shape[1] * 0.5), int(cam_img.shape[0]
* 0.5)), interpolation=cv2.INTER_CUBIC)



    if (frame_count - last_frame_count) > 50:
        decodedObjects = pyzbar.decode(cam_img)
        for obj in decodedObjects:
            last_frame_count = frame_count
            identity = str(obj.data).strip("b'")
            if identity not in allowed:
                allowed.add(identity)
            else:
                allowed.remove(identity)
    else:
        cvzone.putTextRect(cam_img, f'Successfully scanned for id: {identity}', (25, 50),
scale=1,
                           thickness=1, colorR=(0,200,0))


    cv2.imshow("image", cam_img)


    success, img = cap.read()
    img = cv2.resize(img, (int(img.shape[1]*1.5), int(img.shape[0]*1.5)),
interpolation=cv2.INTER_CUBIC)
```

```
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
    imgCanny = cv2.Canny(imgGray, 100, 100)

    detectActivity(imgCanny, prevCanny)
3
    prevCanny = imgCanny.copy()
    cv2.imshow("Image", img)

    if cv2.waitKey(5) & 0xFF==ord('d'):
        x=True
        break
```

Results


Image
Possible Theft detected at space ID 10

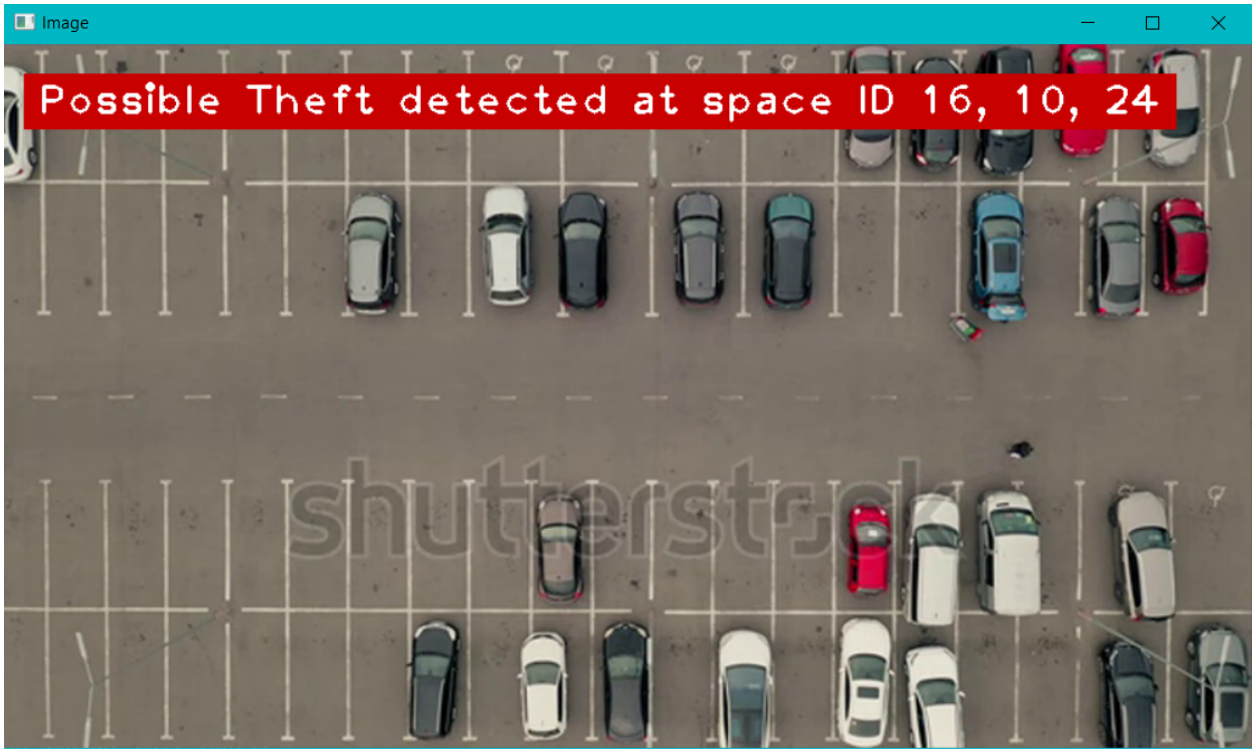
Image
Possible Theft detected at space ID 16, 10, 24

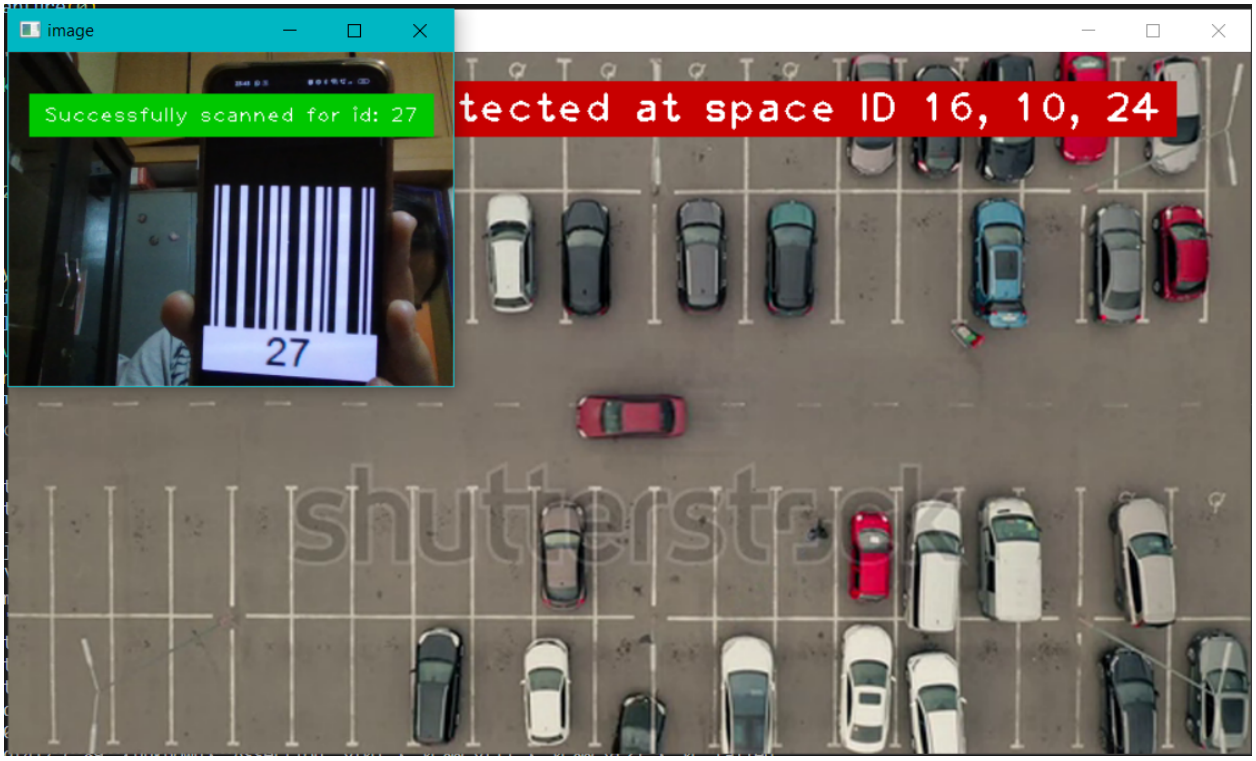
image
Successfully scanned for id: 27
27
tected at space ID 16, 10, 24

Project Schedule

Activity	**Duration**
Search for the problem statement	4/10/2022 - 19/11/2022
Research	20/11/2022 - 25/11/2022
Implementation	26/11/2022 - 12/12/2022
Testing	12/12/2022 - 13/12/2022
Evaluation	22/12/2022

References

[1]"An Automated Way of Vehicle Theft Detection in Parking Facilities by Identifying Moving Vehicles in CCTV Video Stream" - Sayma Shammi, Sayeed Islam, Hafiz Abdur Rahman, Hasan U. Zaman

[2] Moving Object Detection Based on Background Subtraction & Frame Differencing Technique - Md. Zakir Hussain , Mrs. Ayesha Naaz , Mohd Nayeem Uddin

[3] https://opencv.org/

[4] S. Banerjee; P. Choudekar; M. K. Muju, "Real time car parking system using image processing," Electronics Computer Technology (ICECT), April 2011, pp. 99 - 103.

[5]A. Pazhampilly Sreedevi, B. Sarath S Nair, "Image processing based real time vehicle theft detection and prevention system," Process Automation, Control and Computing (PACC), July 2011, pp. 1 - 6.

[6] S. Ojha; S. Sakhare, "Image processing techniques for object tracking in video surveillance- a survey," Pervasive Computing (ICPC), January 2015, pp. 1 - 6.

[7] WREN C R, AZARBAYEJANIA, ARRELLA.Pfinder, "Real2time tracking of the human body," IEEE Transactions on PATTERN ANAL.USA, vol. 19, pp. 780-785, July 1997.

[8] F. F. Chamasemani; L. S. Affendey; F. Khalid; N. Mustapha, "Object detection and representation method for surveillance video indexing," Smart Instrumentation, Measurement and Applications (ICSIMA), November 2015, pp. 1 - 5.

[9] F. F. Chamasemani; L. S. Affendey; F. Khalid; N. Mustapha, "Object detection and representation method for surveillance video indexing," Smart Instrumentation, Measurement and Applications (ICSIMA), November 2015, pp. 1 - 5.

[10] M. Zhao; H. Liu; Y. Wan, "An improved Canny Edge Detection Algorithm," IEEE International Conference on Progress in Informatics and Computing, 2015, pp. 234 – 237.