COMP 3065 – Computer Vision

Coursework

Li Kai WU – 20027091

2022/4/23

(Word count: 1305)

Content

List of Figure

List of Figures

1. Introduction

(Due to the size of video is too big, I cannot submit through module, therefore I decide to send a link in here: https://www.youtube.com/watch?v=jjlBnrzSGjc)

(https://nottinghamedu1-my.sharepoint.com/:f:/g/personal/scylw1_nottingham_edu_cn/EsZiQNn8IvlFtQGOB43wBcEB7ah5vlmha_cq3Amk407uoQ?e=7L9O9z

and here is the link for my own video in OneDrive, need to change the video into video name)

The vehicle detection system (VDS) will be implemented in this project. This project is based on python by using Computer Vision. The statistics provided by the system can have many applications. For example, if you set a price for advertising billboards on highways, the more cars there are, the higher the price. In addition, the government can use this statistic to know how many vehicles enter a city each day.

2. Main Objectives

The main aim of this project is making a vehicle detection system, when object passing by, the system will be able to track the object direction. When object is going up and down, VDS will store the data of vehicle and display on the screen. In this system, when play the video, there will be four lines. In the middle of two lines are estimated the object. When object crossing these two lines, the system will track it and count it as different direction of vehicle (Figure 1).

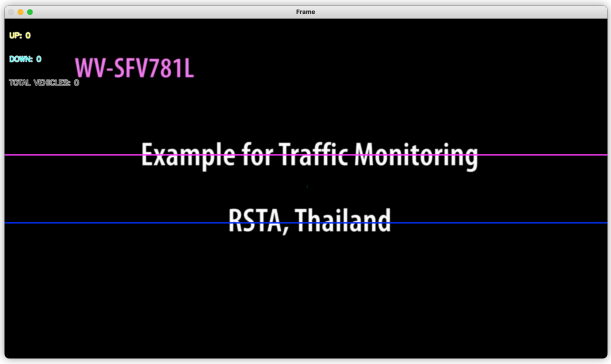

Figure 1. The information when user run the system with example video.

The most of library are using is from Computer Vision, there are several functions are used from cv2.

3. Specification

In this project, the system needs to recognize the vehicle ID and coordinate. Therefore, the class called Car need to be created in the file.

Therefore, the functionality has been used will be explain in this section. First, the VideoCapture is the class for video capturing from video files, image sequences or cameras.

```
video_stored = cv2.VideoCapture("example2.mp4")
```

Figure 2. The way to capturing video

After video setting, there are several parameters need to set up. The line of the estimate, and the value of the screen. Therefore, I use Python's OpenCV library's background subtraction method and some morphing to improve accuracy (Figure 3).

If the video has been stored, the while function will run and detected each object unless the video close or turn off (The video will close when user press SPACE (Figure 4)).

```
# Background Subtractor
BackGround_Sub = cv2.createBackgroundSubtractorMOG2(detectShadows=True)
```

```
# Background subtract to increase the accuracy of the detection
BackGround_Sub_apply = BackGround_Sub.apply(frame)
BackGround_Sub_apply2 = BackGround_Sub.apply(frame)
```

Figure 3. Background subtractor method and morphing

```
# When user click "Space", then the video will turn off
if cv2.waitKey(1) & 0xff == ord(' '):
    break
```

Figure 4. The SPACE keyboarded can stop the video

Therefore, the main part of VDS is in the while function. We need to binarized the value for find the contour, so the cv2.threshold will be used for binarization. This function applies a fixed level threshold to a multichannel array. This function is usually used to obtain two-stage (binary) images from grayscale images or to remove noise.

To find the contours we need to use the cv2.findContours. This function retrieves a contour from a binary image using the @Cite Suzuki85 algorithm. Contour is an effective tool for shape analysis and target detection and recognition.

The main program for tracking vehicles comes after these methods. The FOR function is using for looping the detection functionality. Each contour will be defined.

The cv2.contourArea can get the coordinates of the center of mass. We can use the functions in OpenCV: cv2.contourArea (InputArray Contour). This function calculates the contour area. Similar to the moment, the area is calculated using Green's formula

When area is larger than screen area, then it will start tracking the object (Figure 5). Set x, y, width, and height with cv2.boundingRect(contour), this method is calculating the upper right bounding rectangle of the point set or non-zero pixels of the grayscale image. This function computes and returns a specified set of points or the minimum upper-right border of a non-zero pixel of a grayscale image.

```
for contour in contours:
    area = cv2.contourArea(contour)

    if area > areaTH:
        # Tracking
        m = cv2.moments(contour)
        cx = int(m['m10'] / m['m00'])
        cy = int(m['m01'] / m['m00'])
        x, y, width, height = cv2.boundingRect(contour)
```

Figure 5. The condition to track the vehicle

Therefore, when the contours are inside of the range (limitation of line), the loop for car tracking will be called (Figure 6). The i will be updated make sure the system is not tracking for same ID twice. Therefore, if vehicle existed, the going_UP and going_DOWN will estimate the vehicle direction. When vehicle is going up, then it will print the ID for each vehicle and display the crossing time. It can be used for reality monitoring.

After direction tracked, there is a getState will be used for making sure system go track other vehicle if the vehicle is record by system either going down or going up.

And for each vehicle's ID, I used cv2.putText to print the ID for each vehicle on the scene.

```
if cy in range(up_limit, down_limit):
    for i in cars:
        if abs(x - i.getX()) <= width and abs(y - i.getY()) <= height:
            new = False
            i.updating(cx, cy)

            if i.going_UP(up_line):
                contour_up += 1
                print("ID:", i.getId(), 'the object is going up at', time.strftime("%c"))
            elif i.going_DOWN(down_line):
                contour_down += 1
                print("ID:", i.getId(), 'the object is going down at', time.strftime("%c"))
            break
        if i.getState() == '1':
            if i.getDirection() == 'down' and i.getY() > down_limit:
                i.setDone()
            elif i.getDirection() == 'up' and i.getY() < up_limit:
                i.setDone()
        if i.timedOut():
            index = cars.index(i)
            cars.pop(index)
            del i
```

Figure 6. Vehicle tracking loop

```
for i in cars:
    cv2.putText(frame, str(i.getId()), (i.getX(), i.getY()), cv2.FONT_HERSHEY_SIMPLEX, 0.3, i.getRGB(), 1,
                cv2.LINE_AA)
```

Figure 7. Print ID on scene

Finally, the following functionality can display the number when vehicle going up or down. There will be explain in three parts (Figure 8).

We need to give a parameter for the string value by using contour. For example, when contour is counting by going up, then the GoingUP will display the number of vehicles going up. Therefore, need to set up the range for estimate the direction of vehicle by using cv2.polylines. Finally, use cv2.putText to display the number on the scene. As Figure shown, there are two putText method below. Which is for the white text background style for making sure the text is able to read. Otherwise, the text might be easy to see during the video playing.

```
# Display the Texts on the Screen
GoingUp = 'UP: ' + str(contour_up)
GoingDown = 'DOWN: ' + str(contour_down)
cars_Object = 'TOTAL VEHICLES: ' + str(total_car)

# The Line in video for making sure the range of counting system
frame1 = cv2.polylines(frame, [lines_L1], False, line_down_color, thickness=2)

# The style of Text
cv2.putText(frame1, GoingDown, (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 1, cv2.LINE_AA)
# The bottom of the Text (To make sure user is able to see the value)
cv2.putText(frame1, GoingDown, (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

frame2 = cv2.polylines(frame, [lines_L2], False, line_up_color, thickness=2)

cv2.putText(frame2, GoingUp, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1, cv2.LINE_AA)
cv2.putText(frame2, GoingUp, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

cv2.putText(frame2, cars_Object, (10, 140), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(frame1, cars_Object, (10, 140), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
```

Figure 8. Display vehicle number

4. Result Obtain

For the result of VDS will be present in this section. The example of video is all get it from YouTube, user can easily to find other example to test this system.

When run this system, the number for UP, DOWN, or TOTAL VEHICLE need to be zero (Figure 9). Therefore, when vehicles show on the scene, the VDS will track them (Figure 10). When vehicle crossing from TOP to BELOW, the UP will add one. And if vehicle crossing from BELOW to TOP, the DOWN will add one. The TOTAL VEHICLES will increase no matter the vehicle is crossing up or down. The ID will also follow the vehicle.

Figure 9. Initial value when start the system

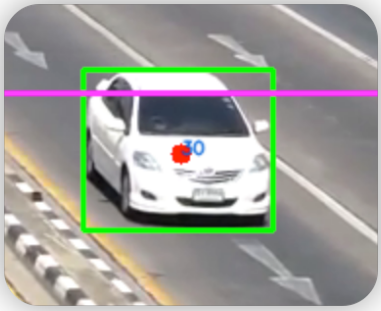

Figure 10. The contour when vehicle comes out

The below figure is the final result when the system run. As figure shown, the UP and DOWN are 25 and 47. And the TOTAL VEHICLES is 72.

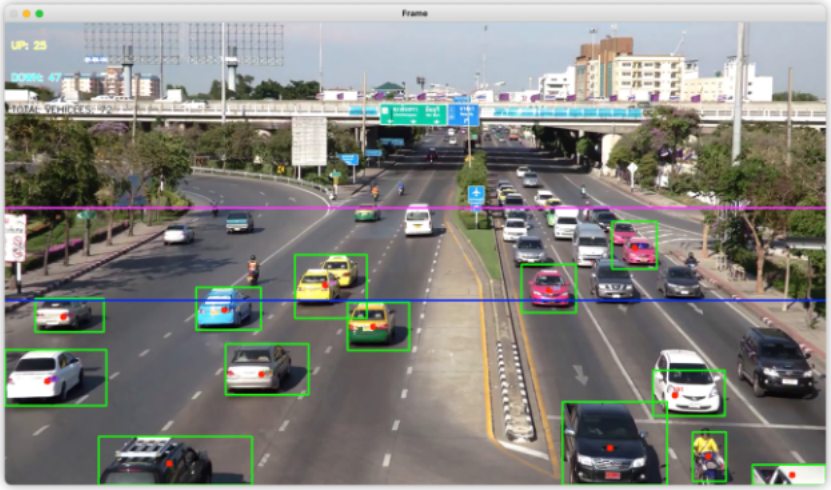

Figure 11. The final result of VDS

5. Advantage & Disadvantage

The project records the comings and goings of vehicles, and many cities can use it to confirm the day's traffic flow. This can be really useful if this system develops completely. For example, For example, in cities, cars come and go, and the number of cars is large, electronic monitoring device are numerous, every moment will produce a large amount of data, the electronic monitoring equipment can keep these urban traffic data, every car models, license plate number, location, and the passing of time will not escape from these monitoring devices. Analysis of these data can get some valuable things, such as where traffic accidents are frequent, need to take further safety measures, where violations are frequent, need to further analyze what causes, etc.

Back to this project, the vehicle tracking is good. Whenever the vehicle appears in the specific range, the VDS can easily to track them and detected their direction (Figure 12).

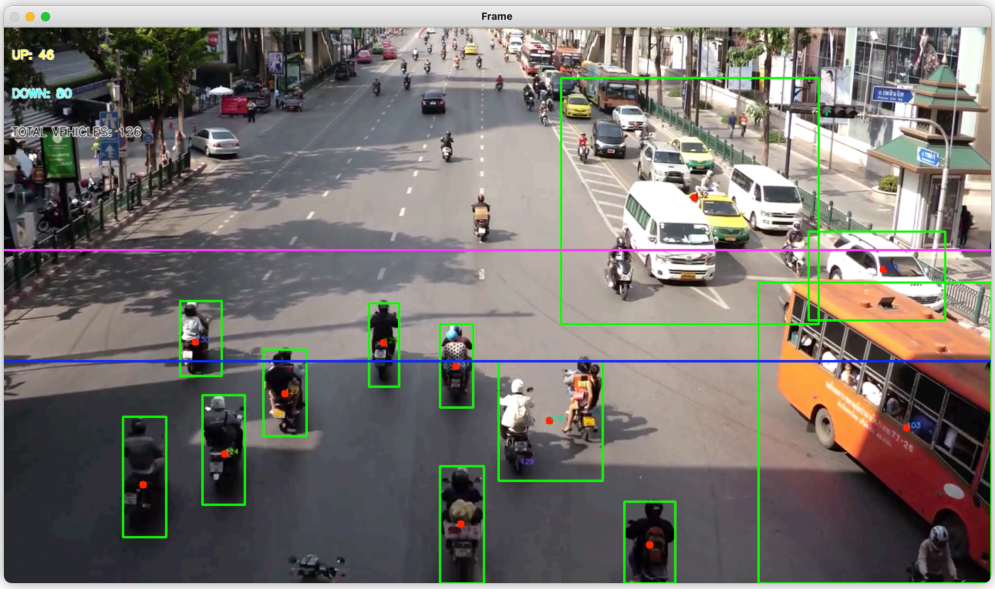

Figure 12. Vehicle tracking when vehicle appear in scene.

Moreover, as long as the vehicle crossing the line (UP or DOWN range), the number will increase for sure.

However, this system is not perfect. Sometime when vehicles are waiting for traffic line, it might accidently increase the number for UP or DOWN. It needs to increase the range's accuracy for going UP or DOWN. And if the vehicles are near to each other, the contour might not be easily to figure out the object, and if the vehicle is too large, the contour might be separate by two or three (Figure 13 & 14).

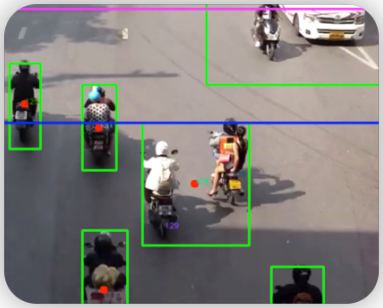

Figure 13. The problem when objects are too close.

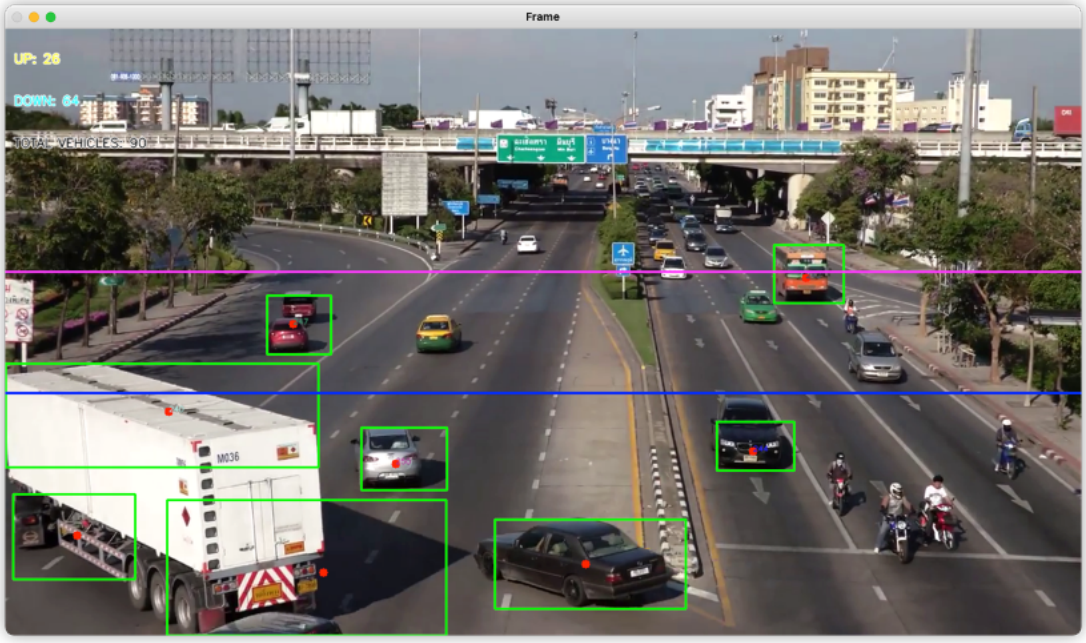

Figure 14. The problem when object is too large.