CGC ASSIGNMENT – 6

Name: RAHUL VARMA

Roll No: S20200010212

combine motion and haar cascade:

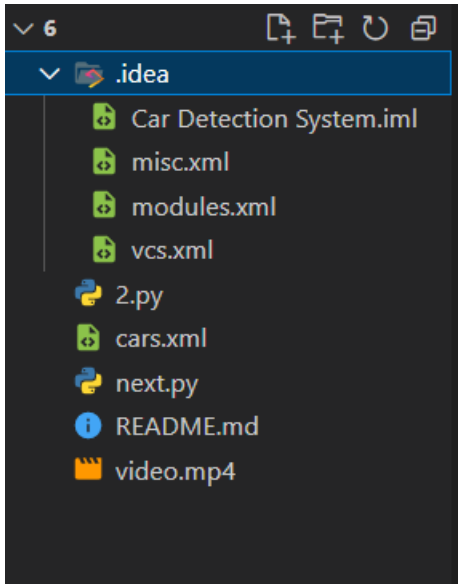
In this assignment we are using the last assignment haar cascade code and combing the haar cascade code with the motion code which we did in the 3rd and 4th assignments.

or case (disjunction).

Motion	Haar	Output
0	1	1
1	0	1
1	1	1
0	0	0

1, 0 means detected and not detected.

Folder Structure:



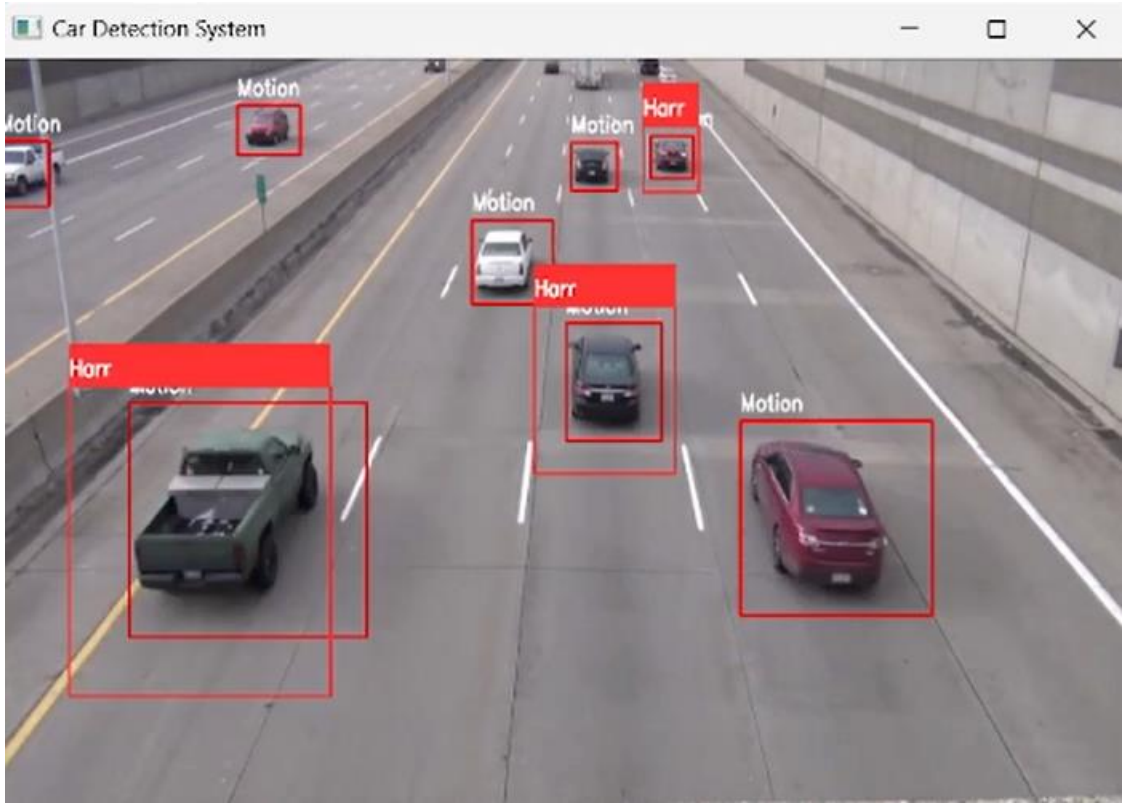
We Use cars.xml file for detecting the object as the car.

Output 1:

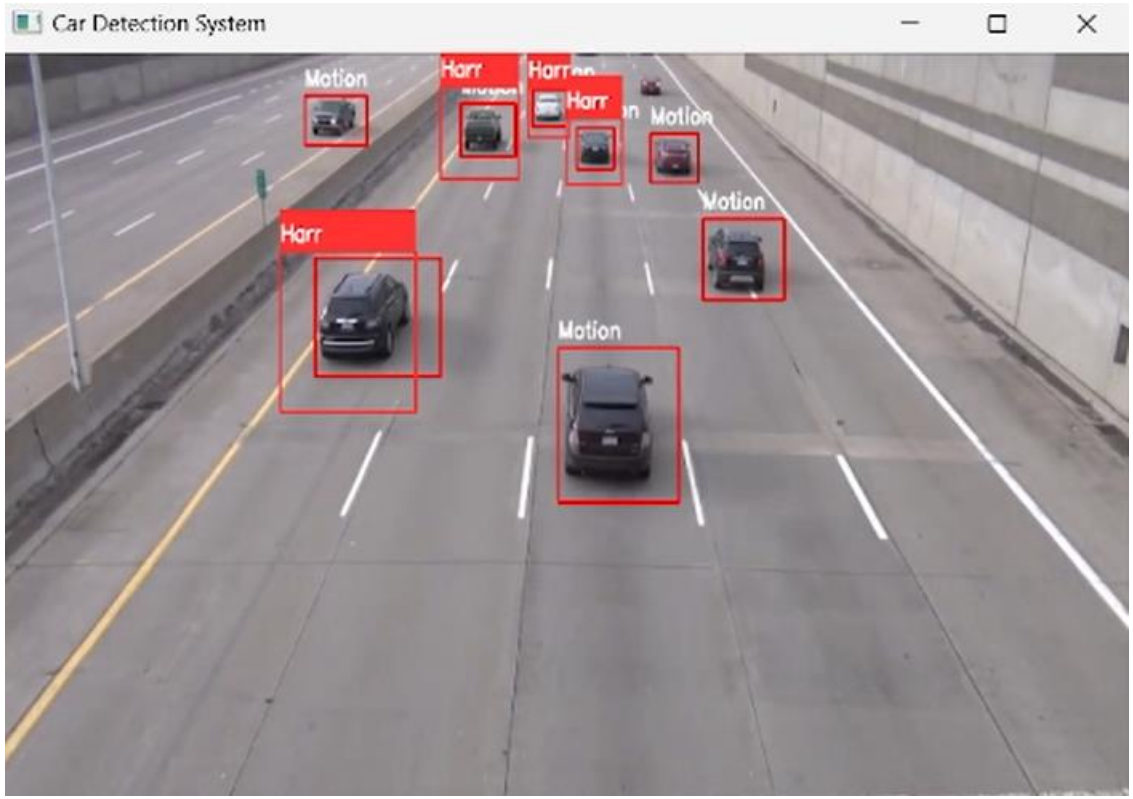
Here White car is due to only motion detection right corner red car also.

Whereas Black and green car are due to both Harr and Motion.

According to the above table if one is active so we can display that car.

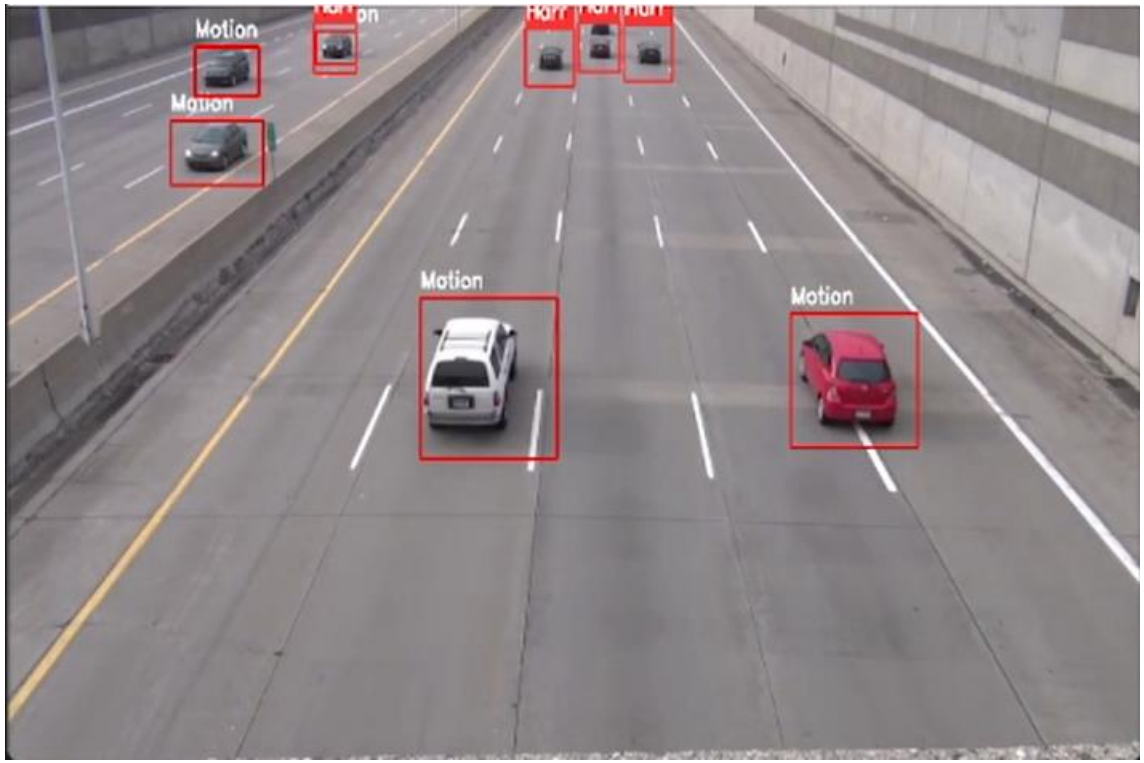


Output 2:



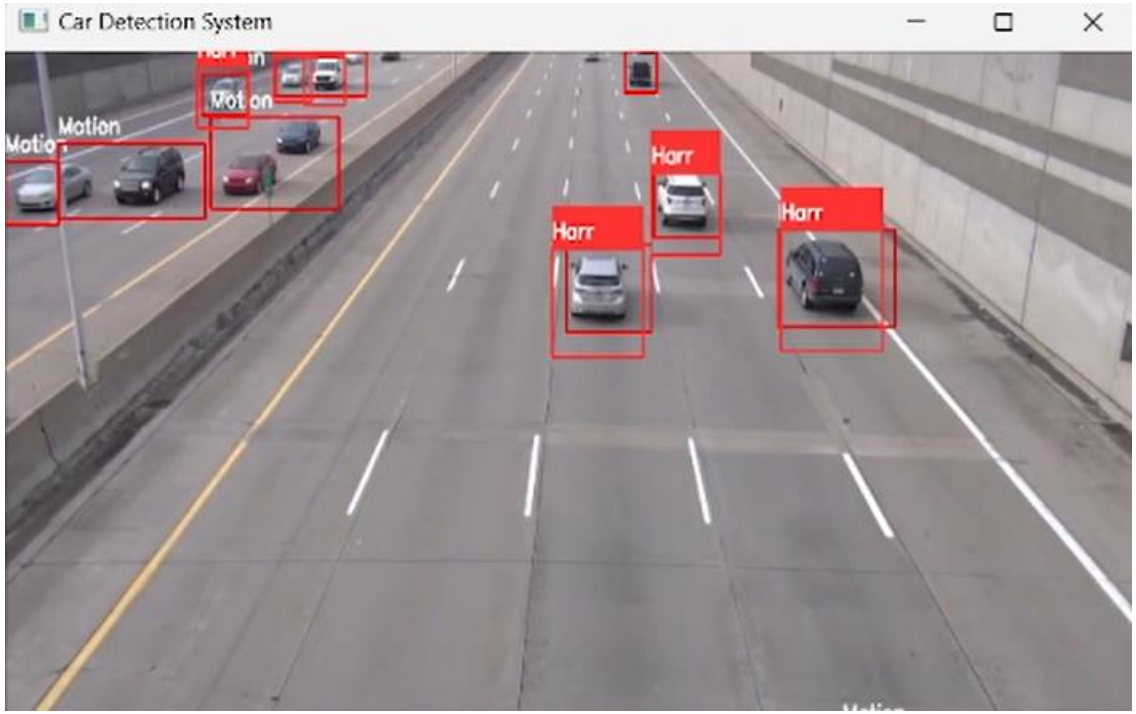
Some are detected due to only motion and some are detected using both motion and harr.

Output 3:



These all due to only motion.

Output 4:



Right side due to Harr and Left Side due to Motion.

Code:

```python
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')
car_cascade = cv2.CascadeClassifier('cars.xml')

while True:
    ret, frames1 = cap.read()
    ret, frames2 = cap.read()
    gray = cv2.cvtColor(frames1, cv2.COLOR_BGR2GRAY)

    # For Motion:
    cars = car_cascade.detectMultiScale(gray, 1.1, 9)
    diff = cv2.absdiff(frames1, frames2)

    # For Motion:
    diff_gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(diff_gray, (5, 5), 0)

    # For Car:
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)

    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # For Motion:
    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)
        if cv2.contourArea(contour) < 1000:
            continue
        cv2.rectangle(frames1, (x, y), (x + w, y + h), (0, 0, 255), 2)
        cv2.putText(frames1, 'Motion', (x, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

    # For Car:
    for (x, y, w, h) in cars:
        plate = frames1[y:y + h, x:x + w]
        cv2.rectangle(frames1, (x, y), (x + w, y + h), (51, 51, 255), 2)
        cv2.rectangle(frames1, (x, y - 40), (x + w, y), (51, 51, 255), -2)
        cv2.putText(frames1, 'Harr', (x, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
        cv2.imshow('car', plate)

    frames1 = cv2.resize(frames1, (600, 400))
    cv2.putText(frames1, "Name: {}".format('RAHUL VARMA'), (10, 570), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 3)
    cv2.putText(frames1, "Roll No: {}".format('S20200010212'), (10, 600), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 3)

    cv2.imshow('Car Detection System', frames1)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

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

```python
import cv2
import numpy as np

cap = cv2.VideoCapture('video.mp4')
car_cascade = cv2.CascadeClassifier('cars.xml')

while True:
    ret, frames1 = cap.read()
    ret, frames2 = cap.read()
    gray = cv2.cvtColor(frames1, cv2.COLOR_BGR2GRAY)

    # For Motion:
    cars = car_cascade.detectMultiScale(gray, 1.1, 9)
    diff = cv2.absdiff(frames1, frames2)

    # For Motion:
    diff_gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(diff_gray, (5, 5), 0)

    # For Car:
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)

    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # For Motion:
    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)
        if cv2.contourArea(contour) < 1000:
            continue
        cv2.rectangle(frames1, (x, y), (x + w, y + h), (0, 0, 255), 2)
        cv2.putText(frames1, 'Motion', (x, y - 10),
```

```
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

    # For Car:
    for (x, y, w, h) in cars:
        plate = frames1[y:y + h, x:x + w]
        cv2.rectangle(frames1, (x, y), (x + w, y + h), (51, 51, 255), 2)
        cv2.rectangle(frames1, (x, y - 40), (x + w, y), (51, 51, 255), -2)
        cv2.putText(frames1, 'Harr', (x, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
        cv2.imshow('car', plate)

    frames1 = cv2.resize(frames1, (600, 400))
    cv2.putText(frames1, "Name: {}".format('RAHUL VARMA'), (10, 570), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255,
255), 3)
    cv2.putText(frames1, "Roll No: {}".format('S20200010212'), (10, 600), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255,
255), 3)

    cv2.imshow('Car Detection System', frames1)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

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

Detailed Explanation:

- The code uses the OpenCV library for computer vision and image processing. It starts by reading in a video file using cv2.VideoCapture(). Then it initializes a cv2.CascadeClassifier object with an XML file containing the trained Haar cascade for car detection.

- Inside the while loop, it reads the frames from the video file using cap.read() and applies some preprocessing. For motion detection, it computes the absolute difference between two consecutive frames using cv2.absdiff(), converts the result to grayscale using cv2.cvtColor(), applies a Gaussian blur using cv2.GaussianBlur(), and applies thresholding using cv2.threshold(). Then it dilates the resulting binary image using cv2.dilate(), and finds the contours using cv2.findContours().

- For car detection, it uses the Haar cascade to detect cars in the frames using car_cascade.detectMultiScale(). Then it extracts the region of interest for each car using array slicing, and draws a rectangle around it using cv2.rectangle(). It also adds a label using cv2.putText().

- Finally, it resizes the frames, adds some text information, and displays them using cv2.imshow(). If the user presses the 'q' key, the loop exits.

- Overall, this code combines two different techniques for detecting objects in a video: motion detection and Haar cascades. By combining these techniques, it is possible to achieve better detection performance and reduce false positives.