Self driving cars using OpenCV

Abirami Ravishankar

Objective

To understand the use OpenCV in self driving cars to understand the nature of the obstacle present in front of the vehicle based on the time it takes to move away from the car and the nature of the path to be taken after encountering.

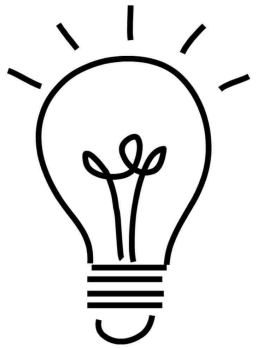
Progress in the field

- Hybrid Navigation
- Homogenization and decoupling
- Vehicle communication systems
- Reprogrammable
- Digital traces

Challenges faced?

The human factor

Plan of action



- Depth analysis of the obstacle(s) present in front.
- Calculating the total time the object spends in front of the camera.
- Detecting the nature of the obstacle in front of it (moving/stationary).
- Movement of the car based on this data received by the camera and works real time and mostly using data fed into it over a period of time.

What can OpenCV do?

With autonomous driving in ADAS (automated driver assist system) in

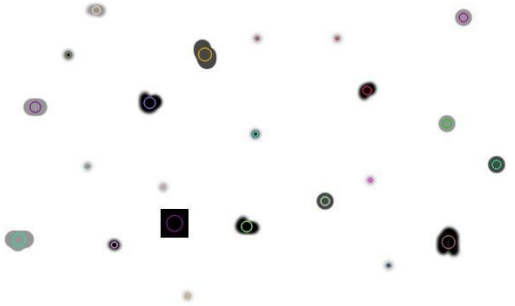
- Traffic light detection
- Traffic signs detection
- Pedestrian detection
- Lane detection

Other applications of OpenCV

- Object identification
- Face detection

Traffic Light Detection

- Sliding Windows
- Color Thresholding
- Spot Light Detection
- BLOB Analysis and Morphological Filters



The dark connected regions are blobs.

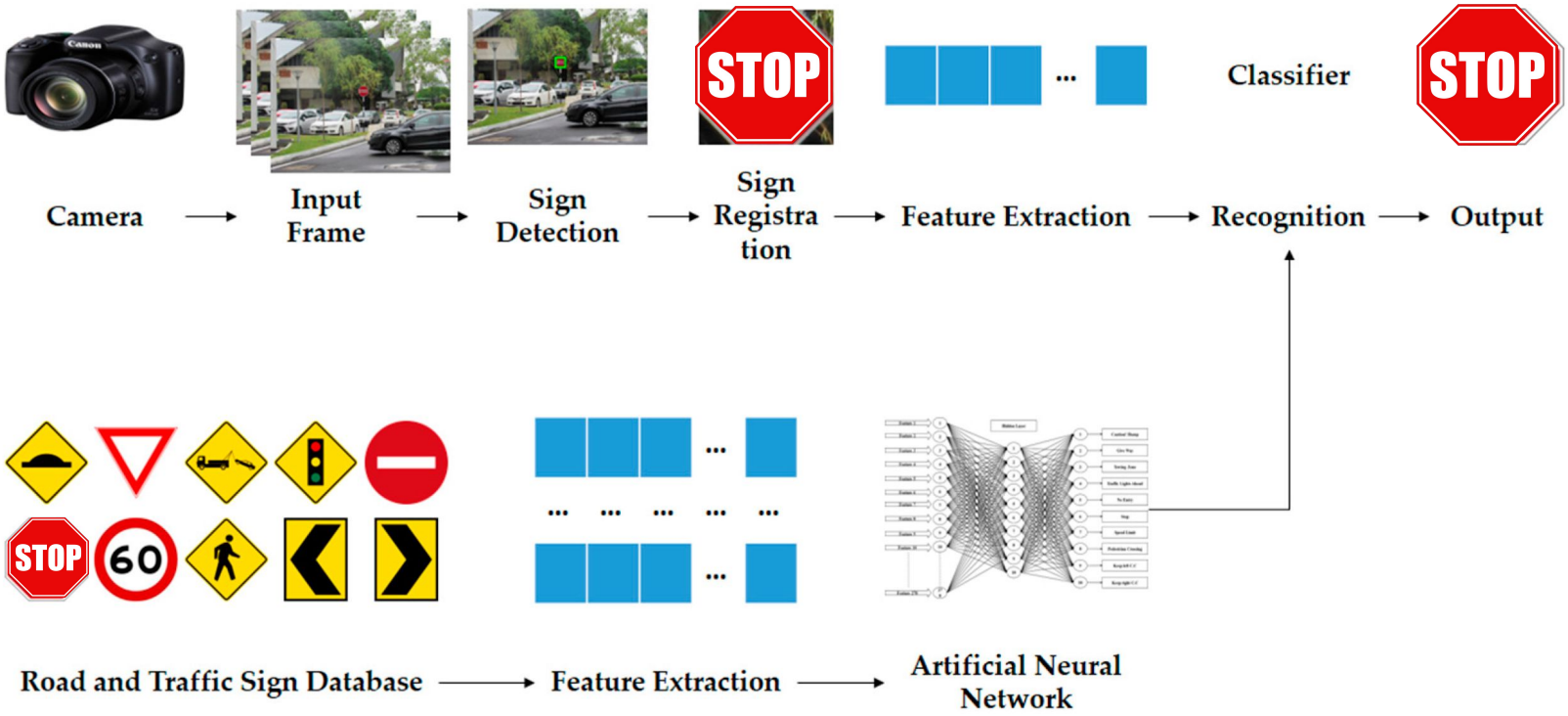
Dilation and flood fill on the red image

Left the original image. Center the dilated object.
Right the eroded object.

Code

```
1   threshold = 100
2
3   gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
4   kernel = np.ones((9, 9), np.uint8)
5   tophat = cv2.morphologyEx(gray, cv2.MORPH_TOPHAT, kernel)
6   ret, thresh = cv2.threshold(tophat, threshold, 255, cv2.THRESH_BINARY)
7
8   dist_transform = cv2.distanceTransform(thresh, cv2.DIST_L2, 5)
9   ret, markers = cv2.connectedComponents(np.uint8(dist_transform))
10  watershed = cv2.watershed(im, markers)
```

Traffic Sign Detection

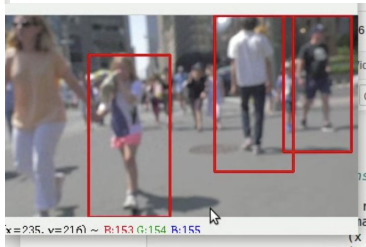


Pedestrian detection

Code

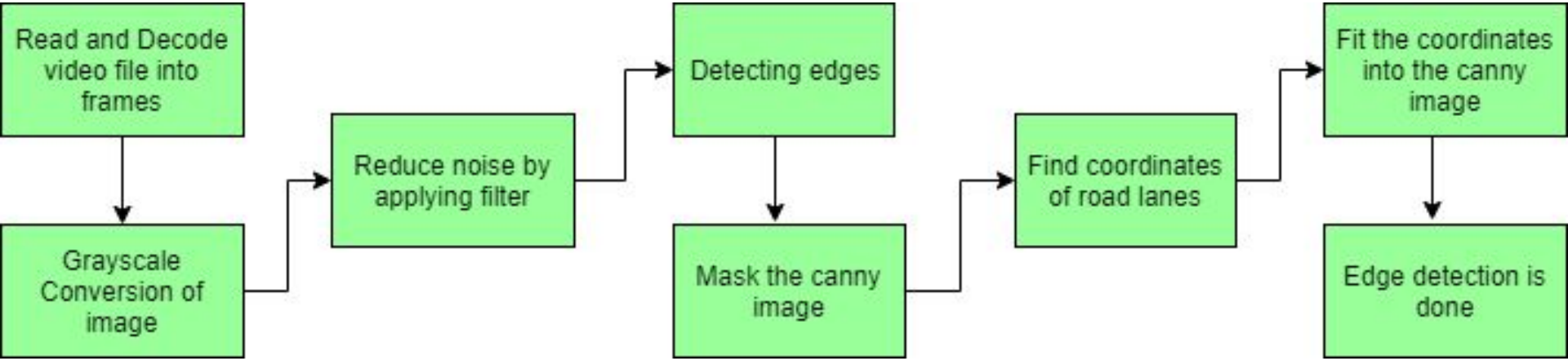
```
# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

cap = cv2.VideoCapture('vid.mp4')

while cap.isOpened():
    # Reading the video stream
    ret, image = cap.read()
    if ret:
        image = imutils.resize(image,
                               width=min(400, image.shape[1]))

        # Detecting all the regions
        # in the Image that has a
        # pedestrians inside it
        (regions, _) = hog.detectMultiScale(image,
                                            winStride=(4, 4),
                                            padding=(4, 4),
                                            scale=1.05)

        # Drawing the regions in the
        # Image
        for (x, y, w, h) in regions:
            cv2.rectangle(image, (x, y),
                          (x + w, y + h),
                          (0, 0, 255), 2)

        # Showing the output Image
        cv2.imshow("Image", image)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        break

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

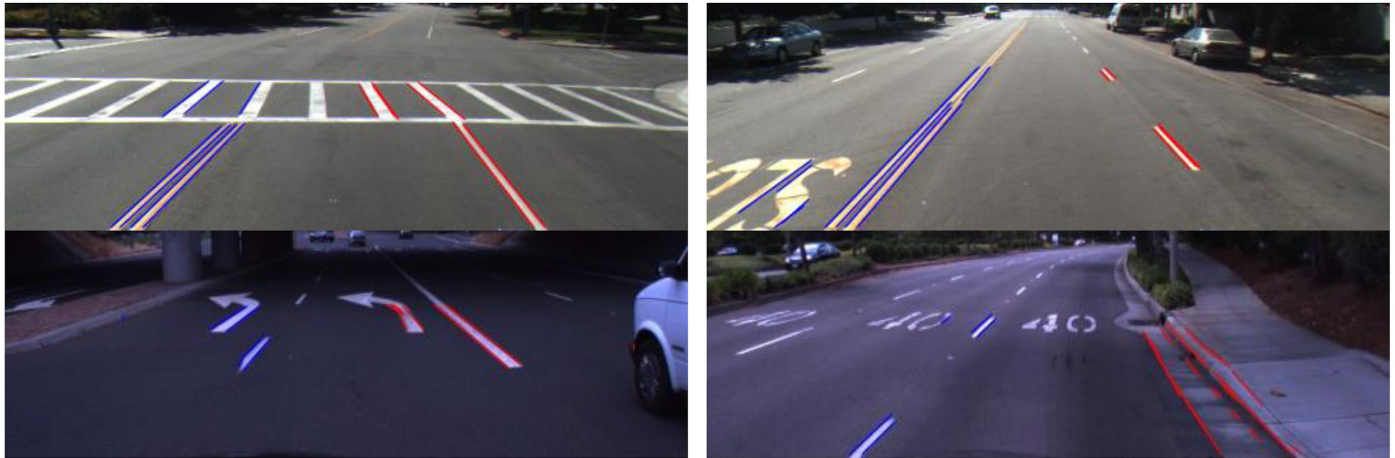
Output



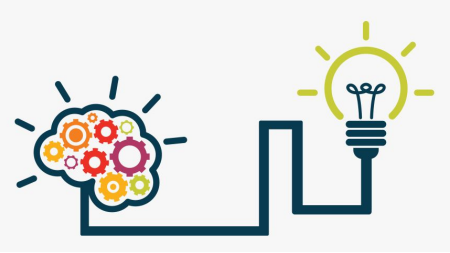
In [ ]:

Lane Detection





Conclusion



- Pre training it with obstacles would make it more efficient than solely relying on real time statistics.

- Training it based on the time an obstacle spends in front of the vehicle and nature of the obstacle

- Thus, using OpenCV and training it with obstacle prior is a more accurate approach to self driving cars.

Thank you!

Questions?