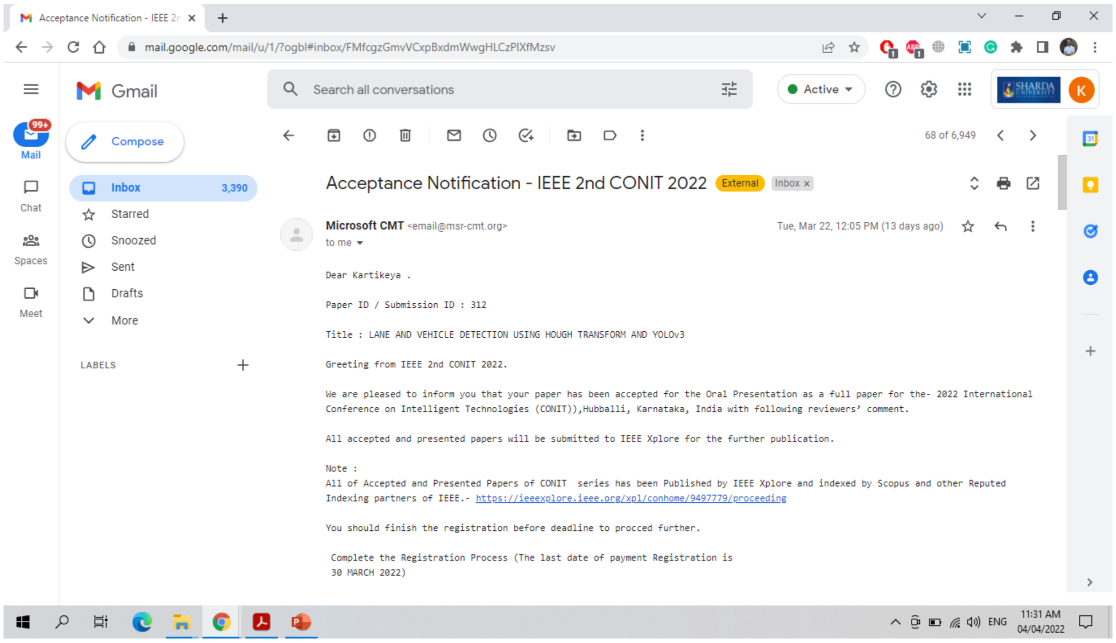
Outcome Proof for 7th & 8th Semester

List of Paper Publication

1. Review of "LANE AND VEHICLE DETECTION USING HOUGH TRANSFORM AND YOLOv3" in 2022 The International Conference for Intelligent Technologies, Karnataka, India. IEEECONIT2022, Submission (312) has been edited. On 5th April,2022 **(Accepted).**
   Link:- http://inconf.in/
   https://cmt3.research.microsoft.com/IEEECONIT2022



Acceptance Notification - IEEE 2nd CONIT 2022 External Inbox

Microsoft CMT <email@msr-cmt.org> Tue, Mar 22, 12:05 PM (13 days ago)
to me

Dear Kartikeya .

Paper ID / Submission ID : 312

Title : LANE AND VEHICLE DETECTION USING HOUGH TRANSFORM AND YOLOv3

Greeting from IEEE 2nd CONIT 2022.

We are pleased to inform you that your paper has been accepted for the Oral Presentation as a full paper for the- 2022 International Conference on Intelligent Technologies (CONIT)),Hubballi, Karnataka, India with following reviewers' comment.

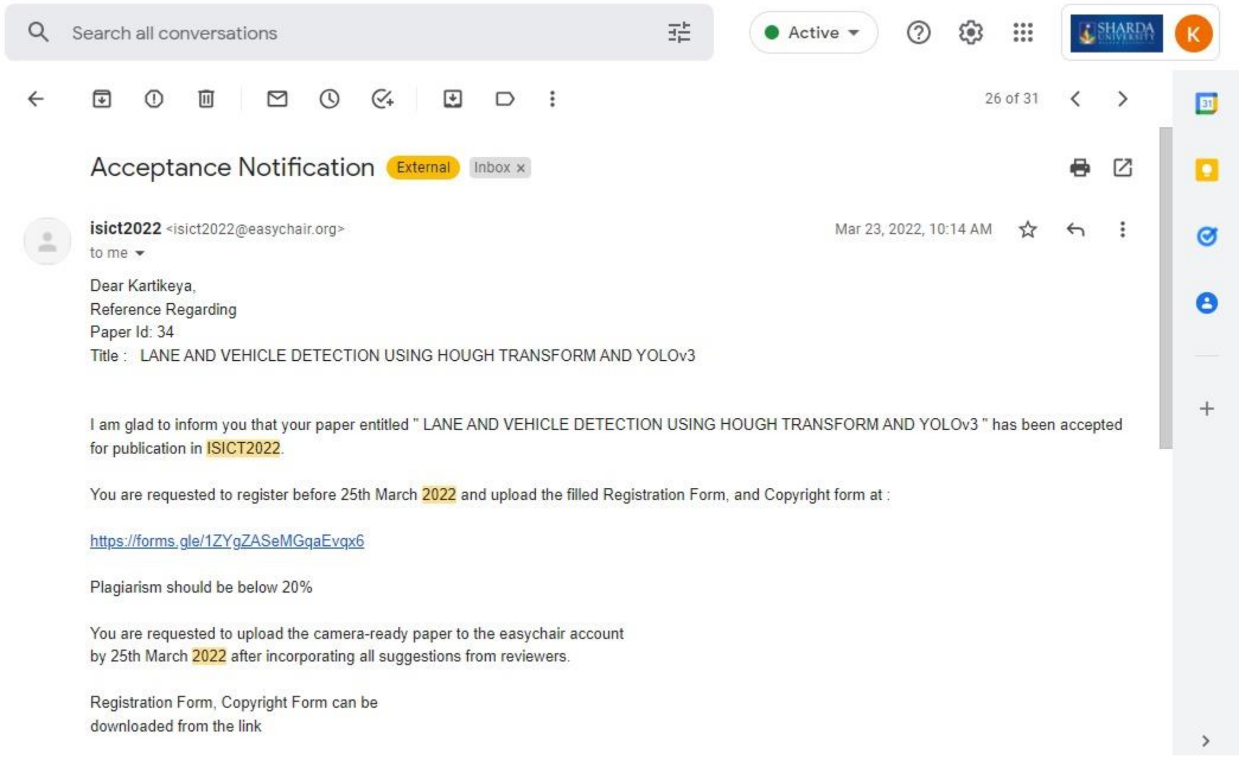
All accepted and presented papers will be submitted to IEEE Xplore for the further publication.

Note :
All of Accepted and Presented Papers of CONIT series has been Published by IEEE Xplore and indexed by Scopus and other Reputed Indexing partners of IEEE.- https://ieeexplore.ieee.org/xpl/conhome/9497779/proceeding

You should finish the registration before deadline to procced further.

Complete the Registration Process (The last date of payment Registration is
30 MARCH 2022)

2. Review of “LANE AND VEHICLE DETECTION USING HOUGH TRANSFORM AND YOLOv3” in 2022 International Symposium on Information & Communication Technology (isict2022), Submission (312) has been edited. On 23th March,2022 **(Accepted).**



Acceptance Notification External Inbox ×

isict2022 <isict2022@easychair.org> Mar 23, 2022, 10:14 AM
to me

Dear Kartikeya,
Reference Regarding
Paper Id: 34
Title : LANE AND VEHICLE DETECTION USING HOUGH TRANSFORM AND YOLOv3

I am glad to inform you that your paper entitled " LANE AND VEHICLE DETECTION USING HOUGH TRANSFORM AND YOLOv3 " has been accepted for publication in ISICT2022.

You are requested to register before 25th March 2022 and upload the filled Registration Form, and Copyright form at :

https://forms.gle/1ZYgZASeMGqaEvqx6

Plagiarism should be below 20%

You are requested to upload the camera-ready paper to the easychair account
by 25th March 2022 after incorporating all suggestions from reviewers.

Registration Form, Copyright Form can be
downloaded from the link

Implementation of Project

1. The implementation on Lane Detection Using Hough Transform

The "Draw Lines()" vector illustration method is used to link the intersection points for every carriageway (left or right) to form a single straight edge that follows the real traffic inside the photos.

The horizontal lines are then generated using the Hough transform, as seen in Fig. 1, and thus are connected mostly by the "Draw Lines()" method to look such as the counterparts in Figs. 2 and 3.

The method does this by doing the proper procedures:

1) Orientation (Left or Right): Every Hough horizontal line is classified as belonging towards the left or right sections. This would be done depending on just the online sector's inclination. Unless the gradient is high somewhere between 0.4 and 1.0, the sector comes to that same left linear category; if this is low and between -0.4 and -1.0, the sector falls to that same right line group.

2) The durations and detects (pedestrian crossings only with x-axis) of all categorized Left as well as Right splices are computed and saved, together with respective gradients.

3) For every class (Left and Right), a line fitting approach is performed, with the gradient of every vertical line indicating whether the vertical line is excellent or poor (sound).

4) The Left and Right arcs may still be produced; although to decrease distortion, the data from the image sequences are used to create an Nth order screen (which has already been tested).

5) The FIR filter formula used in this version seems to be as follows:

$$Yk = a0 * Xk + \ a1 * Xk - 1 + \ a2 * Xk - 2 + \ a3 * Xk - 3 + \cdots .. an * Xk - n \qquad (8)$$

Table 1 shows the resulting FIR filter settings and factors utilized in this design.

Parameters	Value
Order	7
A_0	0.075
A_1	0.125
A_2	0.175
A_3	0.250
A_4	0.175
A_5	0.125
A_6	0.075

6) By using data of "the feature map" determined in step 5 of the process (V), the gradients and detects of the consequent endpoints are designed to characterize the left line in blue as well as the right line in red.

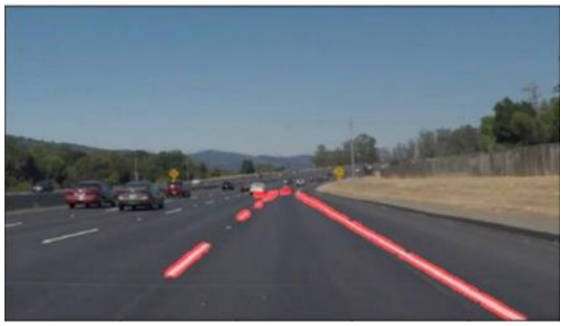


Fig. 1. The image with Hough line segments

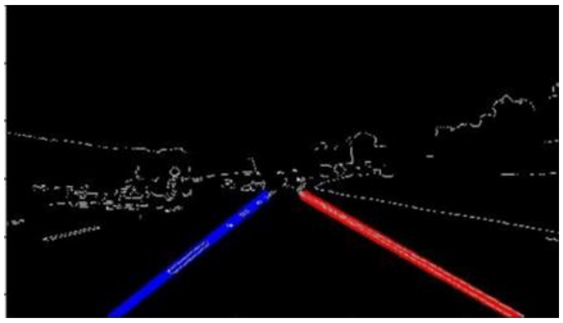


Fig. 2. The image after drawing lane lines (blue and red) by extrapolating the Hough lines segments.

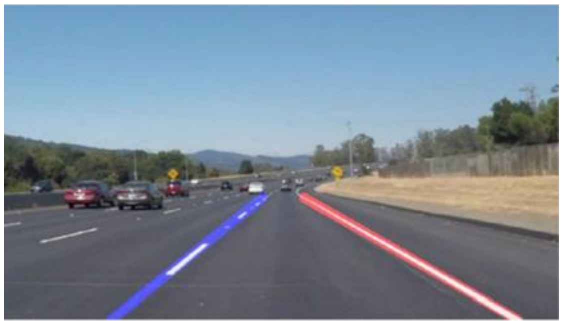


Fig. 3. The image after drawing lane lines (blue and red) by extrapolating the Hough lines segments.

2. The implementation on Vehicle detection using YOLOv3

Merge of the 3 techniques

- The below image shows a combination of three methodologies to produce the last detection result.
- To split the picture Grid cells are used. In every grid cell, B bounding boxes are projected, along with their reliability or best output score. The class probability of the cell is estimated to identify each item class.
- At least three various types of objects are identified, including a truck, a dog, and a bicycle. To construct all of the projections at the same time A single convolutional neural network is used.

When IOU is utilized, the object's bounding box boxes are just like the object's actual boxes. This phenomenon eliminates any unnecessary bounding boxes that do not correspond to the attributes of the items (like height and width). The end detection will consist of separate bounding boxes that are tailored to the objects.

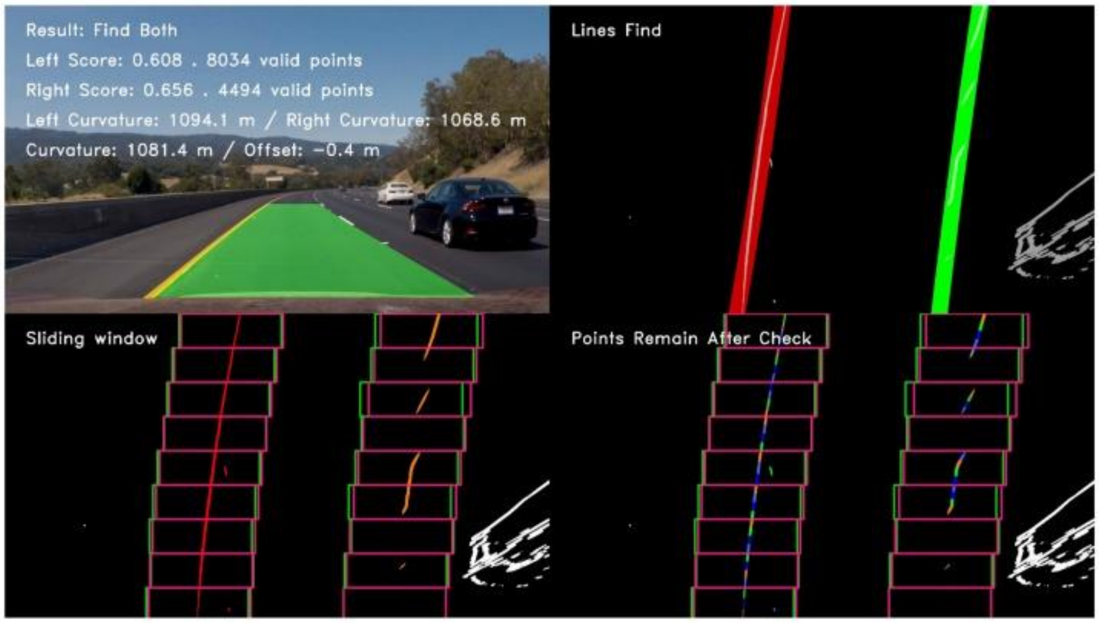
3. Code: -

```
 import matplotlib.pylab as plt

import cv2

import numpy as np


car_cascade = cv2.CascadeClassifier('carx.xml')
```

```
def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    # channel_count = img.shape[2]
    match_mask_color = 255
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image



def draw_the_lines(img, lines):
    img = np.copy(img)
    blank_image = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

    for line in lines:
        for x1, y1, x2, y2 in line:
            cv2.line(blank_image, (x1, y1), (x2, y2), (0, 220, 0), thickness=4)

    img = cv2.addWeighted(img, 0.8, blank_image, 1, 0.0)
    return img

def process(image):
    #image = cv2.imread("road.jpg")
    #image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    #print(image.shape)
    height = image.shape[0]
    width = image.shape[1]

    region_of_interest_vertices = [# to be adjusted as per the image or road, i.e. till road's extreme vertices
        (0, height),
        (0.5*width , 0.5*height),
        (width, height)
    ]
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    canny_image = cv2.Canny(gray_image, 100, 120)
    cropped_image = region_of_interest(canny_image,
                                    np.array([region_of_interest_vertices], np.int32))
    lines = cv2.HoughLinesP(cropped_image,
                        rho=2,
                        theta=np.pi/180,
                        threshold=50,
                        lines=np.array([]),
                        minLineLength=40,
                        maxLineGap=1)
    image_with_lines = draw_the_lines(image, lines)
    return image_with_lines

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

while(True):

    ret, frame = cap.read()
    frame = process(frame)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(gray, 1.1, 3)
    for (x, y, w, h) in cars:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.imshow('frame',frame)

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

cap.release()
```

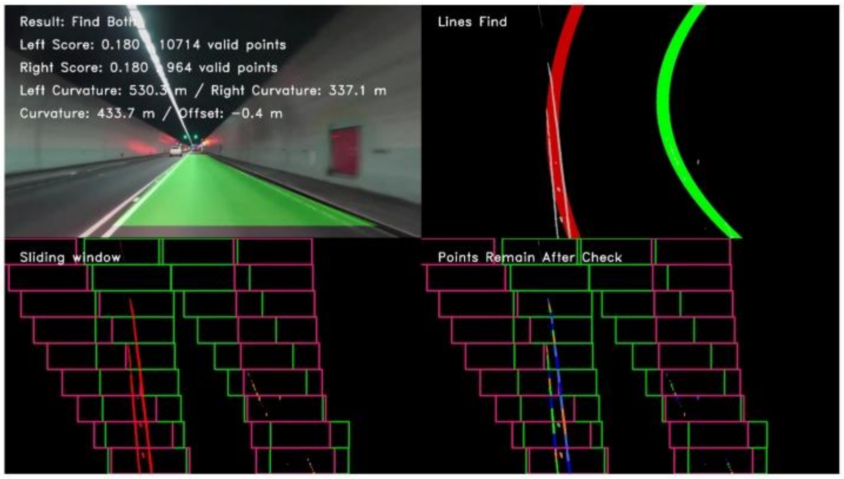
```
cv2.destroyAllWindows()



#plt.imshow(canny_image)

#plt.imshow(cropped_image)

#plt.imshow(image)

#plt.imshow(image_with_lines)

#plt.show()
```

4. Result Output



```
Result: Find Both
Left Score: 0.608 . 8034 valid points
Right Score: 0.656 . 4494 valid points
Left Curvature: 1094.1 m / Right Curvature: 1068.6 m
Curvature: 1081.4 m / Offset: -0.4 m
```

Fig 4. The test result of a typical detected frame in the highway driving video.



```
Result: Find Both
Left Score: 0.180 . 10714 valid points
Right Score: 0.180 . 964 valid points
Left Curvature: 530.3 m / Right Curvature: 337.1 m
Curvature: 433.7 m / Offset: -0.4 m
```

Fig 5. The test result of a typical detected frame in the tunnel road driving video.

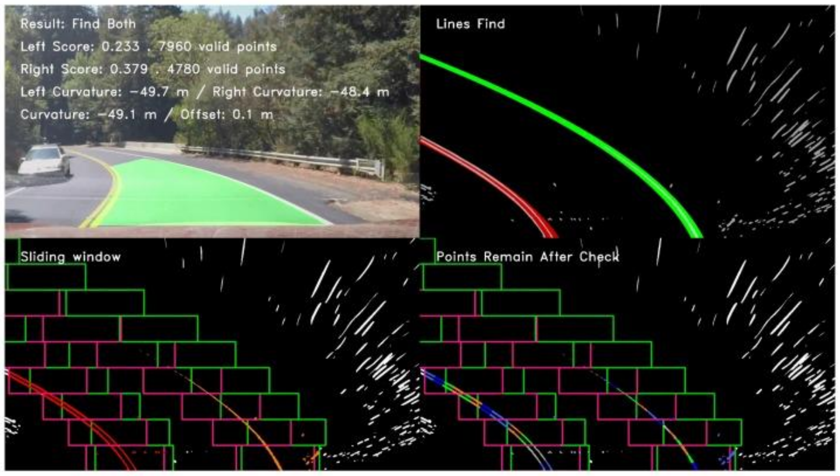

Result: Find Both
Left Score: 0.233 , 7960 valid points
Right Score: 0.379 , 4780 valid points
Left Curvature: −49.7 m / Right Curvature: −48.4 m
Curvature: −49.1 m / Offset: 0.1 m
Lines Find
Sliding window
Points Remain After Check

Fig 6. The test result of a typical detected frame in the mountain road driving video

5. Conclusion

The assessment of travel demand factors is indeed a critical component of roadway construction and management, and also signalized intersections measures for current facilities. Various methods, including those of static location monitoring and "sensor automobile information" devices, could be used to gather traffic information. Furthermore, Wardrop's moving observer method (MOM) may be employed as a traffic information acquisition approach. Although as usually recognized, artificial intelligence (AI) with deep learning (DL) are widely employed in a wide range of real scenarios, notably automobile classification and tracking. In order to address the issue that standard border tracking algorithms need not take recognition rate and productivity is affected into account, this work proposes a protection system adaptive routing yolov3 approach. Below are the primary enhancements:

1. Based on the features of traffic line pictures' uneven transverse and longitudinal dispersion frequency, it really is recommended to partition the pictures across s * 2S squares to increase height recognition intensity.

2. The sensor size has been limited to four sensor scales: 13 * 13, 26 * 26, 52 * 52104 * 104, which is more suited for detecting tiny objects also including lane dividers.

3. The basic yolov3 device's fully connected layers are reduced from 53 to 49 tiers, simplifying the system and improving system efficiency.

4. Parameters including such closest cluster location and transfer functions have been modified, making them more meaningful and suited for traffic line recognition environments. The real test results reveal that perhaps the revised method has high detection accuracy while identifying level roads, however, the identification is easily influenced whenever the roadways have considerable gradients. As a result, the next report will concentrate on fixing the situation of lane line recognition in huge gradient sceneries.

An image recognition computation, exceptional precision as well as authentic affected productivity are critical for such security plus direct control of driverless cars. Several research on sensor driverless cars have indeed been done, but the results have indeed been unimpressive due to an exchange involving precision and operating efficiency. As a result, this work provides an image processing and analysis method for driverless cars which delivers the optimal exchange among precision and reliability. The suggested technique minimizes errors, boosts TP, and considerably decreases FP whilst keeping authentic potential using Gaussian simulation, gradient descent rebuilding, including the use of clustering ambiguity. . The suggested Gaussian YOLOv3 approach enhances the mAP by 3.09 but also 3.5 for such KITTI as well as BDD samples, correspondingly, as related to the background. Moreover, since the suggested technique seems to have a better precision than earlier research with comparable fps, it is indeed a good exchange among precision and affected productivity. As a consequence, the suggested method has the potential to considerably enhance the camera-based target recognition technology for driverless cars, and this is projected to contribute heavily to the widespread usage of self driving technologies. This research offers M-YOLO, a deep learning models approach that relies on YOLO v3. M – YOLO's feature recovery backbone network is based on the MobileNet v2 infrastructure. The level unique version offered by the MobileNet system could significantly minimize the number of variables as well as the complexity of the simulation. The detecting component continues to rely on YOLO v3's multi-scale different recommendation engine. Simultaneously, the K-means technique is utilized in this study to re-cluster the information in order to produce the anchor boxes that are appropriate for this report. Furthermore, M-YOLO employs the EIoU gradient descent to constantly refine the system. According to the statistics, M-average YOLO's detection performance may approach 94.96%, including low-light conditions, the fps could exceed 10 .