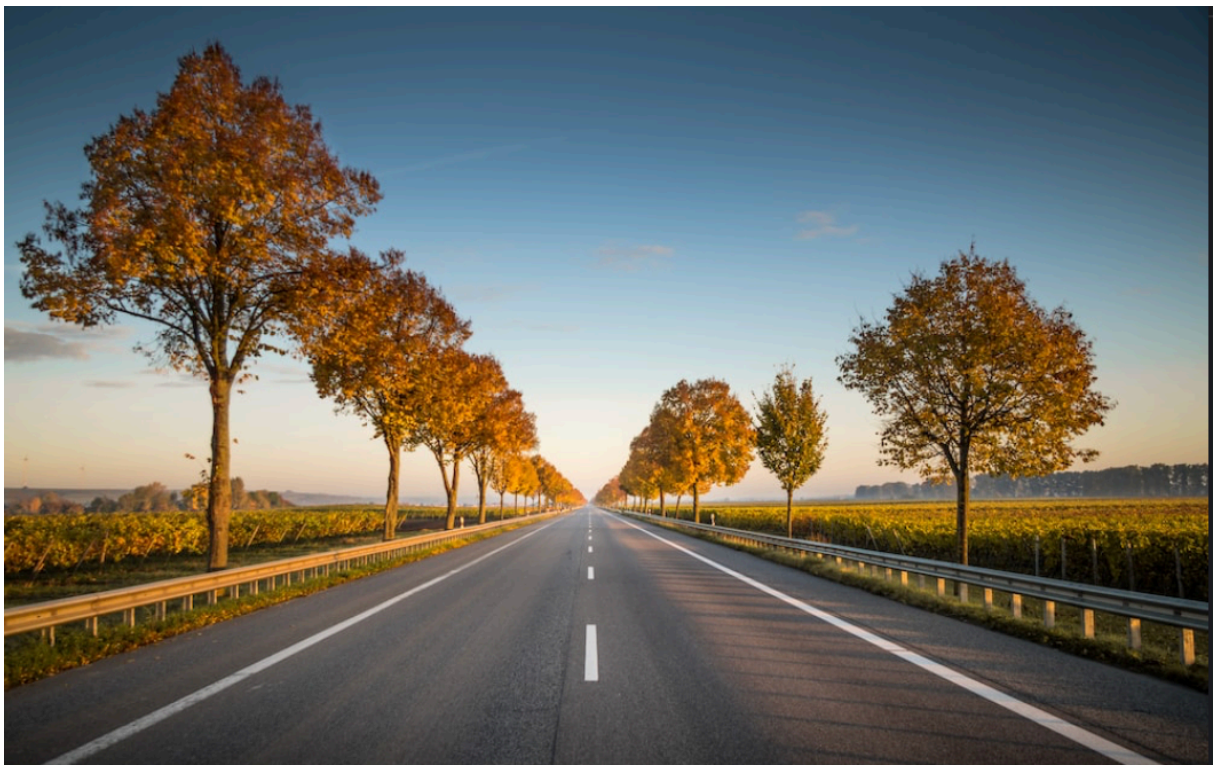
ROAD LANE DETECTION

(CORIZO PROJECT)

LIBRARIES USED:

OpenCV - pip install opencv-contrib-python
Numpy - pip install numpy

Files Used: Road.png



IN THIS FILE I WILL DETECT ROAD LINE

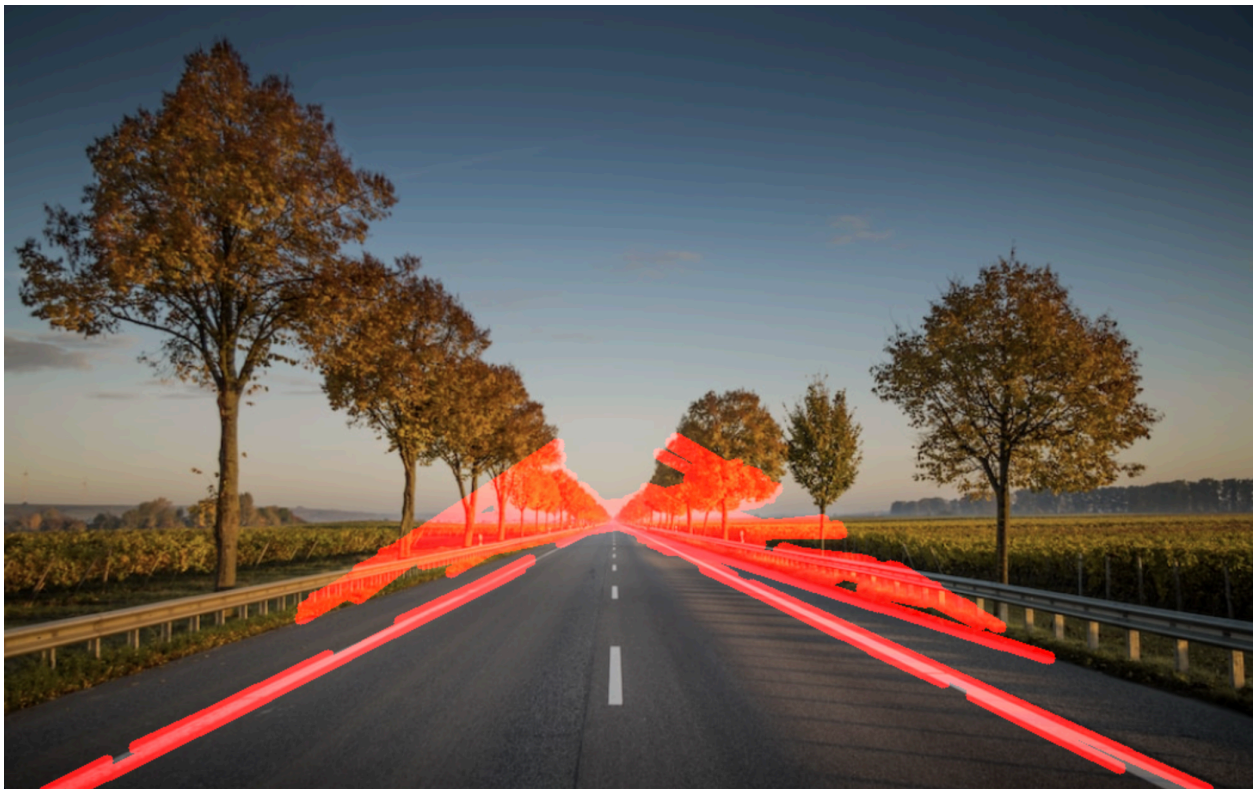
(Without video capturing)

Code-1(without live video capturing)

INPUT:

```
import cv2
import numpy as np
img = cv2.imread('road.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blur, 50, 150)
mask = np.zeros_like(edges)
height, width = mask.shape
polygon = np.array([[
    (0, height),
    (width, height),
    (width // 2, height // 2)
]])
cv2.fillPoly(mask, polygon, 255)
masked_edges = cv2.bitwise_and(edges, mask)

lines = cv2.HoughLinesP(masked_edges, rho=6, theta=np.pi/60, threshold=160,
lines=np.array([]), minLineLength=40, maxLineGap=25)

line_img = np.zeros_like(img)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(line_img, (x1, y1), (x2, y2), (0, 0, 255), 10)

result = cv2.addWeighted(img, 0.8, line_img, 1.0, 0.0)

cv2.imshow('Result', result)
cv2.waitKey(0)
```

OUTPUT:



Screenshot of the code:

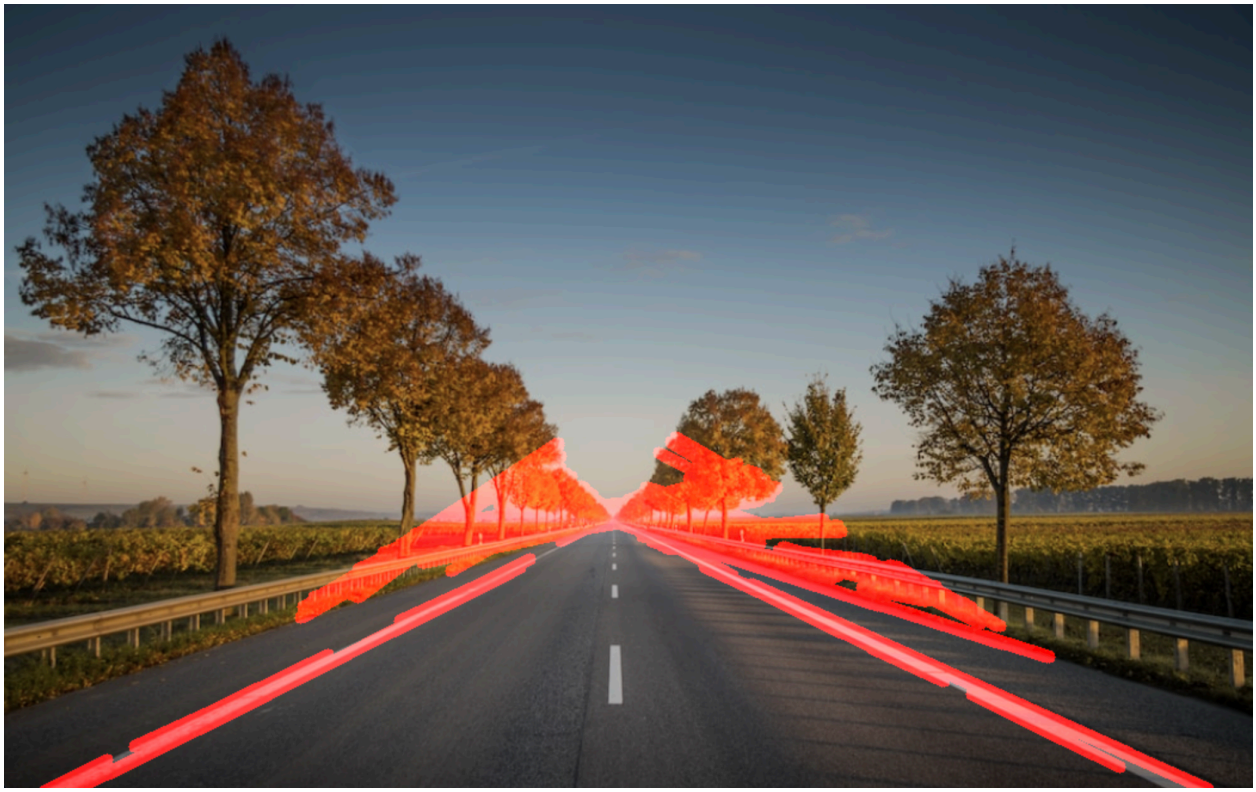
```python
import cv2
import numpy as np
img = cv2.imread('road.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blur, 50, 150)
mask = np.zeros_like(edges)
height, width = mask.shape
polygon = np.array([[
    (0, height),
    (width, height),
    (width // 2, height // 2)
]])
cv2.fillPoly(mask, polygon, 255)
masked_edges = cv2.bitwise_and(edges, mask)

lines = cv2.HoughLinesP(masked_edges, rho=6, theta=np.pi/60, threshold=160, lines=np.array([]), minLineLength=40, maxLineGap=25)

line_img = np.zeros_like(img)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(line_img, (x1, y1), (x2, y2), (0, 0, 255), 10)

result = cv2.addWeighted(img, 0.8, line_img, 1.0, 0.0)

cv2.imshow('Result', result)
cv2.waitKey(0)
```

Code-2(with live video capturing)

INPUT:

```
import cv2
import numpy as np

cap=cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

ret,img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blur, 50, 150)
mask = np.zeros_like(edges)
height, width = mask.shape
polygon = np.array([[
    (0, height),
    (width, height),
    (width // 2, height // 2)
]])
cv2.fillPoly(mask, polygon, 255)
masked_edges = cv2.bitwise_and(edges, mask)

lines = cv2.HoughLinesP(masked_edges, rho=6, theta=np.pi/60, threshold=160,
lines=np.array([]), minLineLength=40, maxLineGap=25)

line_img = np.zeros_like(img)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(line_img, (x1, y1), (x2, y2), (0, 0, 255), 10)

result = cv2.addWeighted(img, 0.8, line_img, 1.0, 0.0)

cv2.imshow('Result', result)
cv2.waitKey(1)

```

OUTPUT:



Screenshot of the code:

```python
import cv2
import numpy as np

cap=cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

ret,img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blur, 50, 150)
mask = np.zeros_like(edges)
height, width = mask.shape
polygon = np.array([[
    (0, height),
    (width, height),
    (width // 2, height // 2)
]])
cv2.fillPoly(mask, polygon, 255)
masked_edges = cv2.bitwise_and(edges, mask)

lines = cv2.HoughLinesP(masked_edges, rho=6, theta=np.pi/60, threshold=160, lines=np.array([]), minLineLength=40, maxLineGap=25)

line_img = np.zeros_like(img)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(line_img, (x1, y1), (x2, y2), (0, 0, 255), 10)

result = cv2.addWeighted(img, 0.8, line_img, 1.0, 0.0)

cv2.imshow('Result', result)
cv2.waitKey(1)
```