import cv2

import matplotlib

import matplotlib.pyplot as plt

import numpy as np

from google.colab.patches import cv2\_imshow

img = cv2.imread('road.jpg',cv2.IMREAD\_COLOR)

# Display original image

cv2\_imshow(img)

# Convert to graycsale

img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

edges = cv2.Canny(img\_gray, 50,200) # Canny Edge Detection

cv2\_imshow(edges)

#Detect points that form a line

lines=cv2.HoughLinesP(edges,1,np.pi/100,55,minLineLength=10,maxLineGap=250)

#Draw lines on the image

for line in lines:

  x1,y1,x2,y2=line[0]

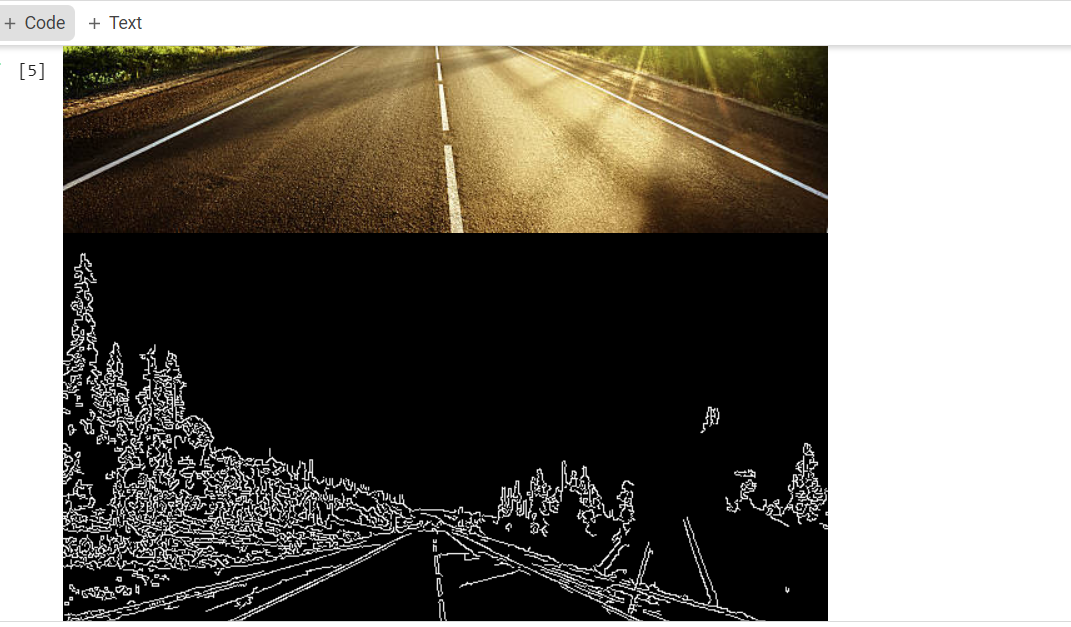
  cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)

#Result Image

cv2\_imshow(img)

Text

Description automatically generated



A picture containing arrow

Description automatically generated

#Hue circle

img = cv2.imread('eyes.jfif', cv2.IMREAD\_COLOR)

#Convert to grayscale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

#Blur using 3 \* 3 kernel

gray\_blurred = cv2.blur(gray, (3,3))

#Apply Hough transform on the blurred image

detected\_circles = cv2.HoughCircles(gray\_blurred,

                    cv2.HOUGH\_GRADIENT, 1, 20, param1 = 50,

                    param2 = 30, minRadius = 1, maxRadius = 40)

#Draw the circles that are detected.

if detected\_circles is not None:

  #Convert the circle parameters a, b and r to integers.

  detected\_circles = np.uint16(np.around(detected\_circles))

  for pt in detected\_circles[0, :]:

    a, b, r = pt[0], pt[1], pt[2]

    #Draw the circumference of the circle.

    cv2.circle(img, (a,b), r, (0, 255, 0), 2)

    #Draw a small circle (of radius 1) to show the center.

    cv2.circle(img, (a,b), 1, (0,0, 255), 3)

    cv2\_imshow(img)

    cv2.waitKey(0)

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

import cv2

import numpy as np

img = cv2.imread('chess.jpg')

#original image

cv2\_imshow(img)

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray=np.float32(gray)

dst=cv2.cornerHarris(gray,2,3,0.04)

#result is dilated for marking the corners, not important

dst=cv2.dilate(dst,None)

#Threshold for an optimal value, it may vary depending on the image

img[dst>0.01\*dst.max()]=[0,0,255]

#dilated image

cv2\_imshow(dst)

#corner detection output

cv2\_imshow(img)

Application

Description automatically generated with low confidence

A picture containing graphical user interface

Description automatically generated

import cv2

import numpy as np

img = cv2.imread('road.jpg')

#original image

cv2\_imshow(img)

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray=np.float32(gray)

dst=cv2.cornerHarris(gray,2,3,0.04)

#result is dilated for marking the corners, not important

dst=cv2.dilate(dst,None)

#Threshold for an optimal value, it may vary depending on the image

img[dst>0.01\*dst.max()]=[0,0,255]

#dilated image

cv2\_imshow(dst)

#corner detection output

cv2\_imshow(img)

Graphical user interface, website

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

nemo=cv2.imread('nemo.jfif')

plt.imshow(nemo)

plt.show()

nemo\_rgb=cv2.cvtColor(nemo,cv2.COLOR\_BGR2RGB)

plt.imshow(nemo\_rgb)

plt.show()

hsv\_nemo=cv2.cvtColor(nemo\_rgb,cv2.COLOR\_RGB2HSV)

plt.imshow(hsv\_nemo)

plt.show()

Graphical user interface

Description automatically generated

Diagram

Description automatically generated with low confidence

#color segmentation

import cv2

import numpy as np

import matplotlib.pyplot as plt

from google.colab.patches import cv2\_imshow

from matplotlib.colors import hsv\_to\_rgb

%matplotlib inline

#for orange color segmentation

light\_orange=(1,190,200)

dark\_orange=(18,255,255)

lo\_square=np.full((10,10,3),light\_orange,dtype=np.uint8)/255.0

do\_square=np.full((10,10,3),dark\_orange,dtype=np.uint8)/255.0

plt.subplot(1,2,1)

plt.imshow(hsv\_to\_rgb(do\_square))

plt.subplot(1,2,2)

plt.imshow(hsv\_to\_rgb(lo\_square))

plt.show()

Chart

Description automatically generated

mask=cv2.inRange(hsv\_nemo,light\_orange,dark\_orange)

result=cv2.bitwise\_and(nemo,nemo,mask=mask)

plt.subplot(1,2,1)

Graphical user interface, application

Description automatically generated

mask = cv2.inRange(hsv\_nemo,light\_orange,dark\_orange)

result = cv2.bitwise\_and(nemo,nemo,mask=mask)

plt.subplot(1,2,1)

plt.imshow(mask,cmap="gray")

plt.subplot(1,2,2)

plt.imshow(result)

plt.show()

Graphical user interface, application

Description automatically generated