**根据OPENCV官网示例加工整理**

https://docs.opencv.org/4.x/d3/df2/tutorial\_py\_basic\_ops.html

目录

[任务1：打开图片和存储图片 1](#_Toc93776554)

[任务2：捕捉视频并显示，存储 8](#_Toc93776555)

[任务3：绘图 16](#_Toc93776556)

[任务4：鼠标绘图 17](#_Toc93776557)

[任务5：调色板 20](#_Toc93776558)

[任务6：图片及基础操作 22](#_Toc93776559)

[任务7：图片操作之加减合成 32](#_Toc93776560)

[任务8：性能计算 34](#_Toc93776561)

[任务9：图像空间 35](#_Toc93776562)

开始：软件安装

百度搜索python下载相关软件进行安装

C:\Users\Administrator>pip list

Package Version

--------------- --------

cycler 0.11.0

kiwisolver 1.3.2

matplotlib 3.4.3

numpy 1.21.4

opencv-python 4.5.4.58

Pillow 8.4.0

pip 21.3.1

pyparsing 3.0.5

pyserial 3.5

python-dateutil 2.8.2

setuptools 57.4.0

six 1.16.0

建立文件夹用于存储示例代码

F:\python test

## 任务1：打开图片和存储图片

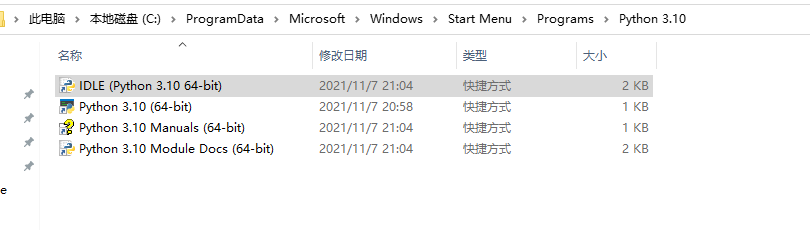
使用python打开图片p1.png,并进行显示

将文件保存为p2.png

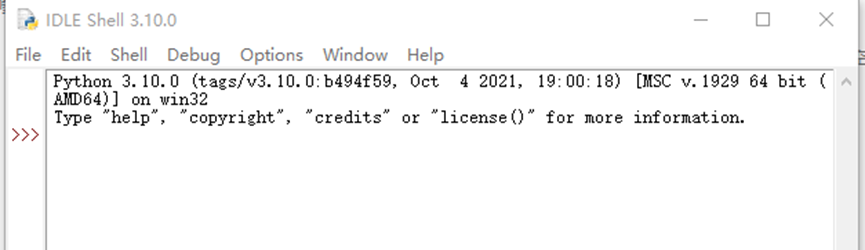


将图片文件p1.png存放在目录中：F:\python test 中

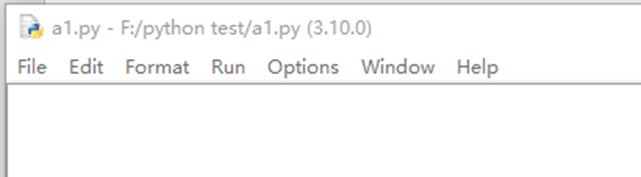
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Python 3.10

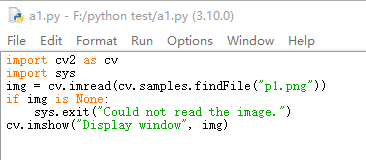


打开IDLE软件：



新建文件a1.py





F5(快捷键)运行该程序

代码如下：

文件名：a1.py

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2 as cv

import sys

img = cv.imread(cv.samples.findFile("p1.png"))

if img is None:

sys.exit("Could not read the image.")

cv.imshow("Display window", img)

import cv2 as cv 引入库

import sys 引入库

img = cv.imread(cv.samples.findFile("p1.png")) 读取图片文件

if img is None:

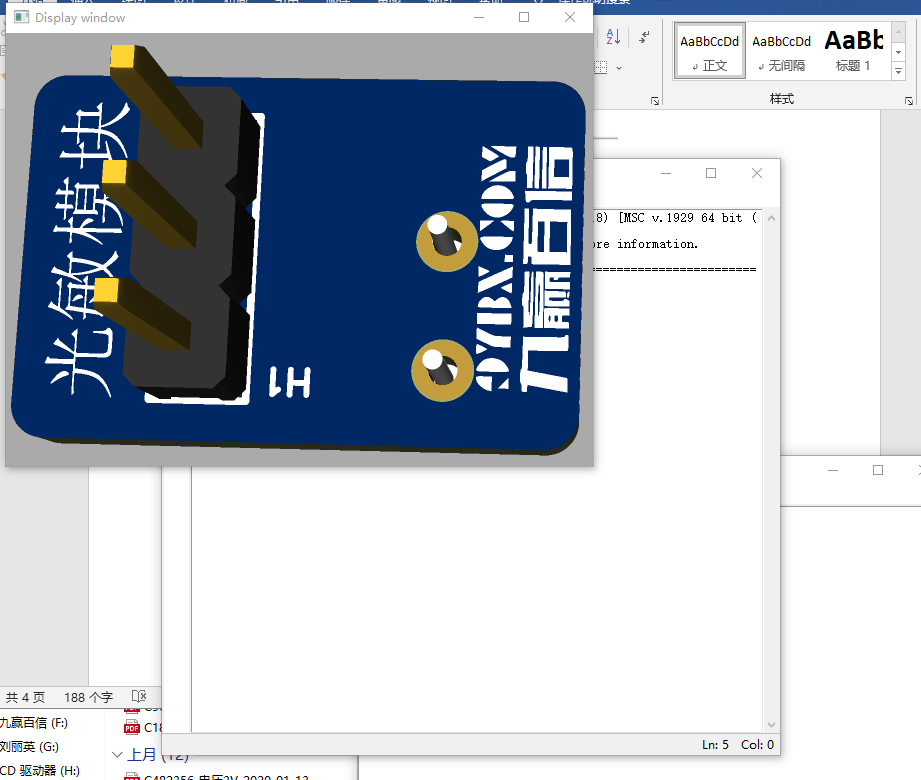
sys.exit("Could not read the image.")

cv.imshow("Display window", img) 显示文件



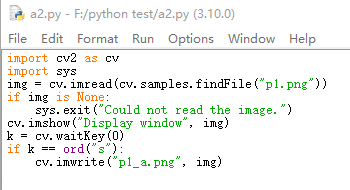
以上是代码的二维码！

运行结果如下：



|  |  |
| --- | --- |
| cv2.imread（） |  |
| img = cv2.imread('1.jpg', cv2.IMREAD\_UNCHANGED) | #包含alpha通道 |
| img = cv2.imread('1.jpg', cv2.IMREAD\_COLOR) | 彩色图像 |
| img = cv2.imread('1.jpg', 1) |
|  |  |
| img = cv2.imread('1.jpg', cv2.COLOR\_BGR2GRAY) | 颜色转换空间 |
|  |  |
| img = cv2.imread('1.jpg', cv2.IMREAD\_GRAYSCALE) | 灰度图像 |
| img = cv2.imread('1.jpg', 0) |
|  |  |
| img.shape[0]的值为图片的高度1262 |  |
| img.shape[1]的值为图片的宽度12688 |  |
| img.shape[3]的值为图片的通道3 |  |

文件名：a2.py



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2 as cv

import sys

img = cv.imread(cv.samples.findFile("p1.png"))

if img is None:

sys.exit("Could not read the image.")

cv.imshow("Display window", img)

k = cv.waitKey(0)

if k == ord("s"):

cv.imwrite("p1\_a.png", img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

k = cv.waitKey(0) 等待按键输入

if k == ord("s"): 如果输入的是s

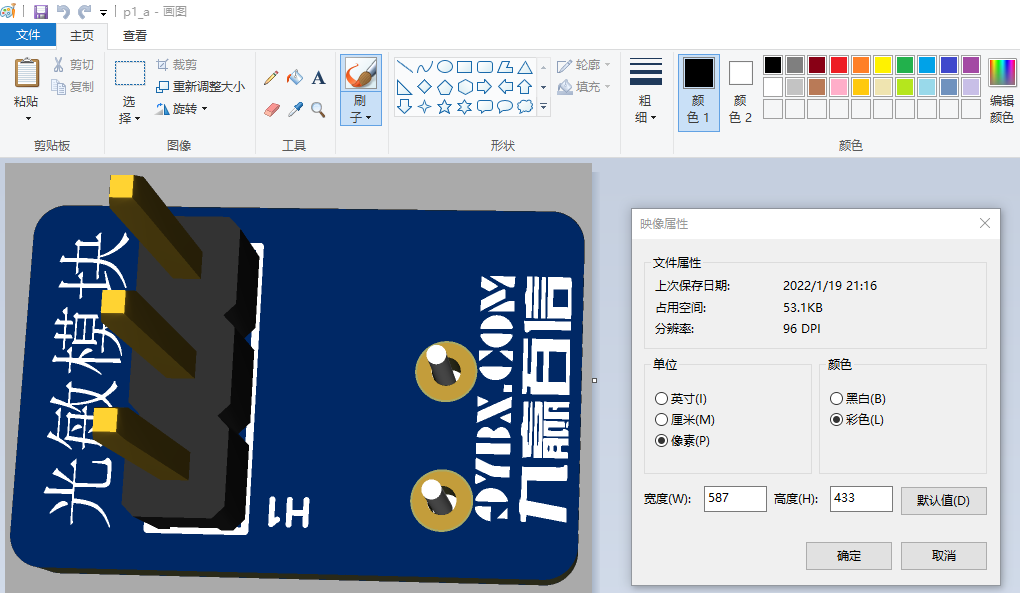
cv.imwrite("p1\_a.png", img) 保存文件

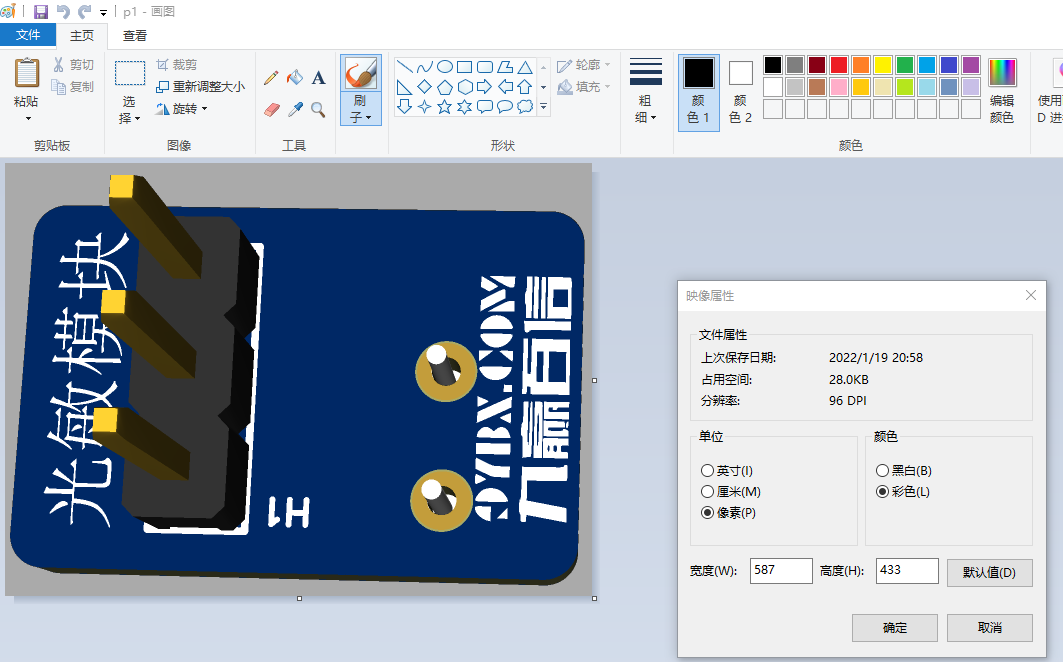


代码的二维码



复制的文件更大了。28KB🡪54KB

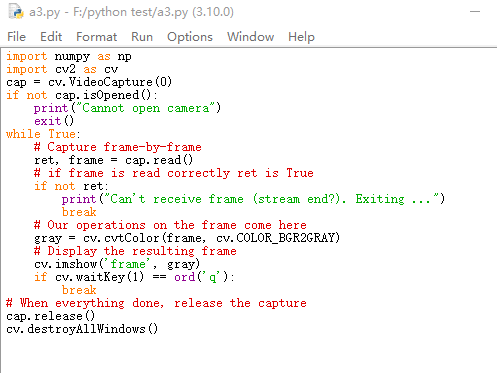




Png文件大小，和参数有关！

## 任务2：捕捉视频并显示，存储

前提是需要连接摄像头



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

cap = cv.VideoCapture(0)

if not cap.isOpened():

print("Cannot open camera")

exit()

while True:

# Capture frame-by-frame

ret, frame = cap.read()

# if frame is read correctly ret is True

if not ret:

print("Can't receive frame (stream end?). Exiting ...")

break

# Our operations on the frame come here

gray = cv.cvtColor(frame, cv.COLOR\_BGR2GRAY)

# Display the resulting frame

cv.imshow('frame', gray)

if cv.waitKey(1) == ord('q'):

break

# When everything done, release the capture

cap.release()

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



文件名：a3.py

cv::cvtColor()支持多种颜色空间之间的转换，其支持的转换类型和转换码如下：

1、RGB和BGR（opencv默认的彩色图像的颜色空间是BGR）颜色空间的转换

cv::COLOR\_BGR2RGB

cv::COLOR\_RGB2BGR

cv::COLOR\_RGBA2BGRA

cv::COLOR\_BGRA2RGBA

2、向RGB和BGR图像中增添alpha通道

cv::COLOR\_RGB2RGBA

cv::COLOR\_BGR2BGRA

3、从RGB和BGR图像中去除alpha通道

cv::COLOR\_RGBA2RGB

cv::COLOR\_BGRA2BGR

4、从RBG和BGR颜色空间转换到灰度空间

cv::COLOR\_RGB2GRAY

cv::COLOR\_BGR2GRAY

cv::COLOR\_RGBA2GRAY

cv::COLOR\_BGRA2GRAY

5、从灰度空间转换到RGB和BGR颜色空间

cv::COLOR\_GRAY2RGB

cv::COLOR\_GRAY2BGR

cv::COLOR\_GRAY2RGBA

cv::COLOR\_GRAY2BGRA

6、RGB和BGR颜色空间与BGR565颜色空间之间的转换

cv::COLOR\_RGB2BGR565

cv::COLOR\_BGR2BGR565

cv::COLOR\_BGR5652RGB

cv::COLOR\_BGR5652BGR

cv::COLOR\_RGBA2BGR565

cv::COLOR\_BGRA2BGR565

cv::COLOR\_BGR5652RGBA

cv::COLOR\_BGR5652BGRA

7、灰度空间域BGR565之间的转换

cv::COLOR\_GRAY2BGR555

cv::COLOR\_BGR5552GRAY

8、RGB和BGR颜色空间与CIE XYZ之间的转换

cv::COLOR\_RGB2XYZ

cv::COLOR\_BGR2XYZ

cv::COLOR\_XYZ2RGB

cv::COLOR\_XYZ2BGR

9、RGB和BGR颜色空间与uma色度（YCrCb空间）之间的转换

cv::COLOR\_RGB2YCrCb

cv::COLOR\_BGR2YCrCb

cv::COLOR\_YCrCb2RGB

cv::COLOR\_YCrCb2BGR

10、RGB和BGR颜色空间与HSV颜色空间之间的相互转换

cv::COLOR\_RGB2HSV

cv::COLOR\_BGR2HSV

cv::COLOR\_HSV2RGB

cv::COLOR\_HSV2BGR

11、RGB和BGR颜色空间与HLS颜色空间之间的相互转换

cv::COLOR\_RGB2HLS

cv::COLOR\_BGR2HLS

cv::COLOR\_HLS2RGB

cv::COLOR\_HLS2BGR

12、RGB和BGR颜色空间与CIE Lab颜色空间之间的相互转换

cv::COLOR\_RGB2Lab

cv::COLOR\_BGR2Lab

cv::COLOR\_Lab2RGB

cv::COLOR\_Lab2BGR

13、RGB和BGR颜色空间与CIE Luv颜色空间之间的相互转换

cv::COLOR\_RGB2Luv

cv::COLOR\_BGR2Luv

cv::COLOR\_Luv2RGB

cv::COLOR\_Luv2BGR

14、Bayer格式（raw data）向RGB或BGR颜色空间的转换

cv::COLOR\_BayerBG2RGB

cv::COLOR\_BayerGB2RGB

cv::COLOR\_BayerRG2RGB

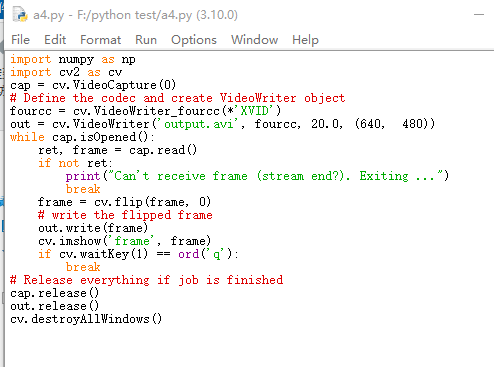
cv::COLOR\_BayerGR2RGB

cv::COLOR\_BayerBG2BGR

cv::COLOR\_BayerGB2BGR

cv::COLOR\_BayerRG2BGR

cv::COLOR\_BayerGR2BGR



保存视频为output.avi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

cap = cv.VideoCapture(0)

# Define the codec and create VideoWriter object

fourcc = cv.VideoWriter\_fourcc(\*'XVID')

out = cv.VideoWriter('output.avi', fourcc, 20.0, (640, 480))

while cap.isOpened():

ret, frame = cap.read()

if not ret:

print("Can't receive frame (stream end?). Exiting ...")

break

frame = cv.flip(frame, 0)

# write the flipped frame

out.write(frame)

cv.imshow('frame', frame)

if cv.waitKey(1) == ord('q'):

break

# Release everything if job is finished

cap.release()

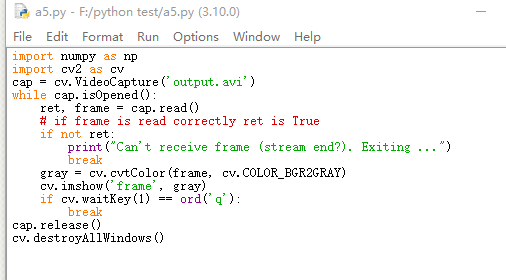
out.release()

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



文件a5.py 读取并显示刚才存储output.avi



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

cap = cv.VideoCapture('output.avi')

while cap.isOpened():

ret, frame = cap.read()

# if frame is read correctly ret is True

if not ret:

print("Can't receive frame (stream end?). Exiting ...")

break

gray = cv.cvtColor(frame, cv.COLOR\_BGR2GRAY)

cv.imshow('frame', gray)

if cv.waitKey(1) == ord('q'):

break

cap.release()

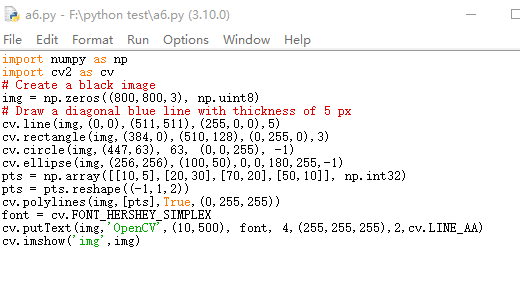
cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 任务3：绘图



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

# Create a black image

img = np.zeros((800,800,3), np.uint8)

# Draw a diagonal blue line with thickness of 5 px

cv.line(img,(0,0),(511,511),(255,0,0),5)

cv.rectangle(img,(384,0),(510,128),(0,255,0),3)

cv.circle(img,(447,63), 63, (0,0,255), -1)

cv.ellipse(img,(256,256),(100,50),0,0,180,255,-1)

pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)

pts = pts.reshape((-1,1,2))

cv.polylines(img,[pts],True,(0,255,255))

font = cv.FONT\_HERSHEY\_SIMPLEX

cv.putText(img,'OpenCV',(10,500), font, 4,(255,255,255),2,cv.LINE\_AA)

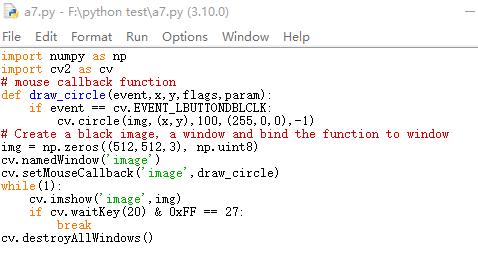
cv.imshow('img',img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



<https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html>

## 任务4：鼠标绘图



双击鼠标，绘图

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

# mouse callback function

def draw\_circle(event,x,y,flags,param):

if event == cv.EVENT\_LBUTTONDBLCLK:

cv.circle(img,(x,y),100,(255,0,0),-1)

# Create a black image, a window and bind the function to window

img = np.zeros((512,512,3), np.uint8)

cv.namedWindow('image')

cv.setMouseCallback('image',draw\_circle)

while(1):

cv.imshow('image',img)

if cv.waitKey(20) & 0xFF == 27:

break

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if event == cv.EVENT\_LBUTTONDBLCLK: 双击鼠标事件

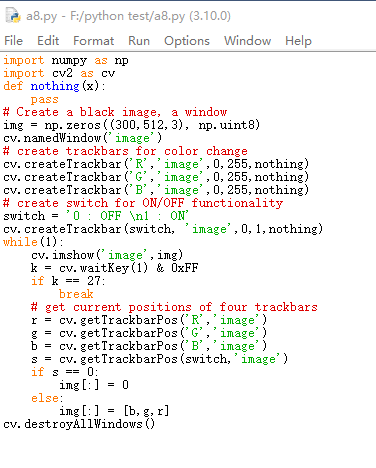
cv.circle(img,(x,y),100,(255,0,0),-1) 绘图

,(255,0,0) 颜色：（蓝色，绿色，红色）



<https://docs.opencv.org/4.x/db/d5b/tutorial_py_mouse_handling.html>

## 任务5：调色板



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

def nothing(x):

pass

# Create a black image, a window

img = np.zeros((300,512,3), np.uint8)

cv.namedWindow('image')

# create trackbars for color change

cv.createTrackbar('R','image',0,255,nothing)

cv.createTrackbar('G','image',0,255,nothing)

cv.createTrackbar('B','image',0,255,nothing)

# create switch for ON/OFF functionality

switch = '0 : OFF \n1 : ON'

cv.createTrackbar(switch, 'image',0,1,nothing)

while(1):

cv.imshow('image',img)

k = cv.waitKey(1) & 0xFF

if k == 27:

break

# get current positions of four trackbars

r = cv.getTrackbarPos('R','image')

g = cv.getTrackbarPos('G','image')

b = cv.getTrackbarPos('B','image')

s = cv.getTrackbarPos(switch,'image')

if s == 0:

img[:] = 0

else:

img[:] = [b,g,r]

cv.destroyAllWindows()

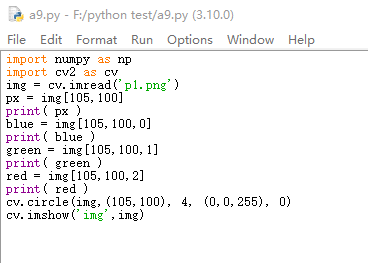
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



<https://docs.opencv.org/4.x/d9/dc8/tutorial_py_trackbar.html>

## 任务6：图片及基础操作

读取图片的一个点



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img = cv.imread('p1.png')

px = img[105,100]

print( px )

blue = img[105,100,0]

print( blue )

green = img[105,100,1]

print( green )

red = img[105,100,2]

print( red )

cv.circle(img,(105,100), 4, (0,0,255), 0)

cv.imshow('img',img)

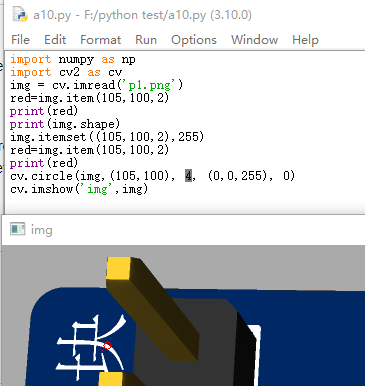
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

px = img[105,100] 读取指定点的颜色值

cv.circle(img,(105,100), 4, (0,0,255), 0) 在指定点绘制圆，指示选择点的位置



访问像素点，并更改像素点的颜色值



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img = cv.imread('p1.png')

red=img.item(105,100,2)

print(red)

print(img.shape)

img.itemset((105,100,2),255)

red=img.item(105,100,2)

print(red)

cv.circle(img,(105,100), 4, (0,0,255), 0)

cv.imshow('img',img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

红色圆圈的左下角的红色点为更改后的显示。注意，这个程序并没有保存图像，对原图像没有更改。

一个简单的扩展应用，比如我们可以把一个颜色的值进行颜色替换。

<https://docs.opencv.org/4.x/d3/df2/tutorial_py_basic_ops.html>



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img = cv.imread('p1.png')

blue=img.item(105,100,0)

green=img.item(105,100,1)

red=img.item(105,100,2)

print(blue)

print(green)

print(red)

#101 40 0

a=1

b=1

while a<150:

a=a+1

b=1

while b<150:

b=b+1

blue=img.item(a,b,0)

green=img.item(a,b,1)

red=img.item(a,b,2)

if blue==101 and green==40 and red==0:

img.itemset((a,b,2),255)

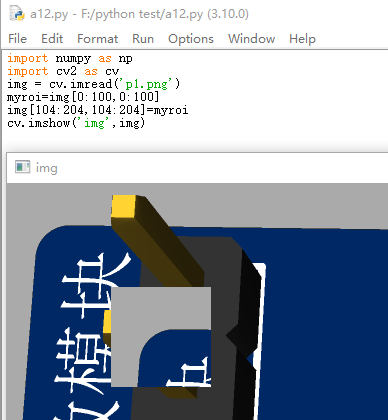
cv.imshow('img',img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



设置 Region of Interest (ROI)

将一个区域图像设置为ROI,然后在另外一个地方重现，或者修改后重现。



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img = cv.imread('p1.png')

myroi=img[0:100,0:100]

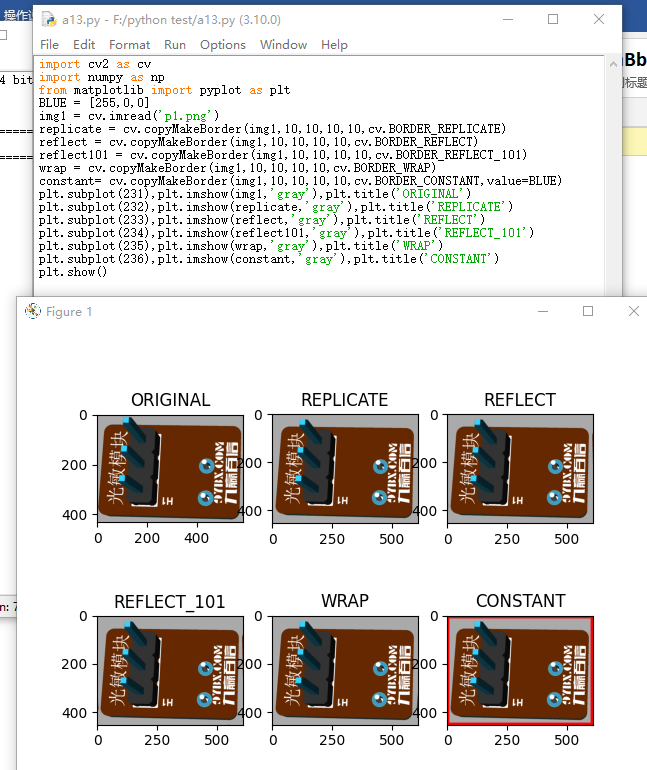
img[104:204,104:204]=myroi

cv.imshow('img',img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



绘制边框



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2 as cv

import numpy as np

from matplotlib import pyplot as plt

BLUE = [255,0,0]

img1 = cv.imread('p1.png')

replicate = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER\_REPLICATE)

reflect = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER\_REFLECT)

reflect101 = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER\_REFLECT\_101)

wrap = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER\_WRAP)

constant= cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER\_CONSTANT,value=BLUE)

plt.subplot(231),plt.imshow(img1,'gray'),plt.title('ORIGINAL')

plt.subplot(232),plt.imshow(replicate,'gray'),plt.title('REPLICATE')

plt.subplot(233),plt.imshow(reflect,'gray'),plt.title('REFLECT')

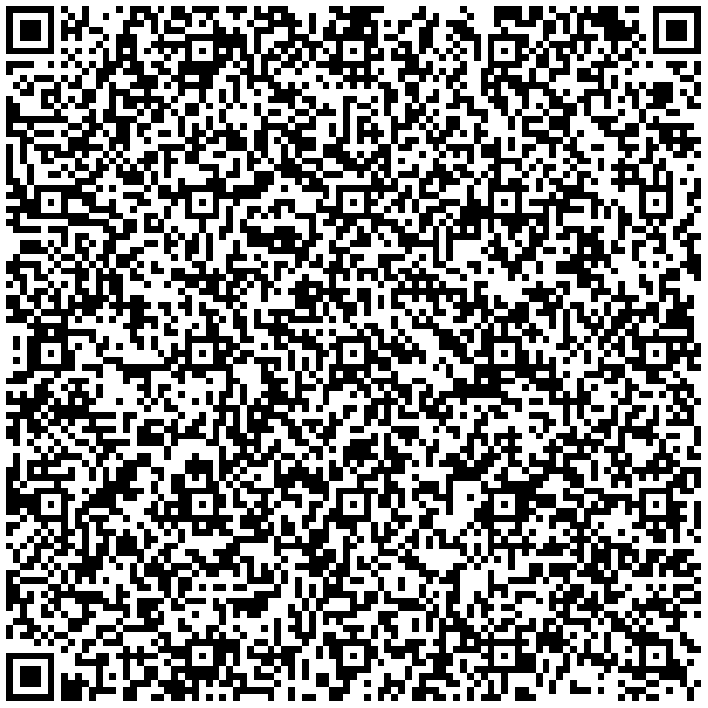
plt.subplot(234),plt.imshow(reflect101,'gray'),plt.title('REFLECT\_101')

plt.subplot(235),plt.imshow(wrap,'gray'),plt.title('WRAP')

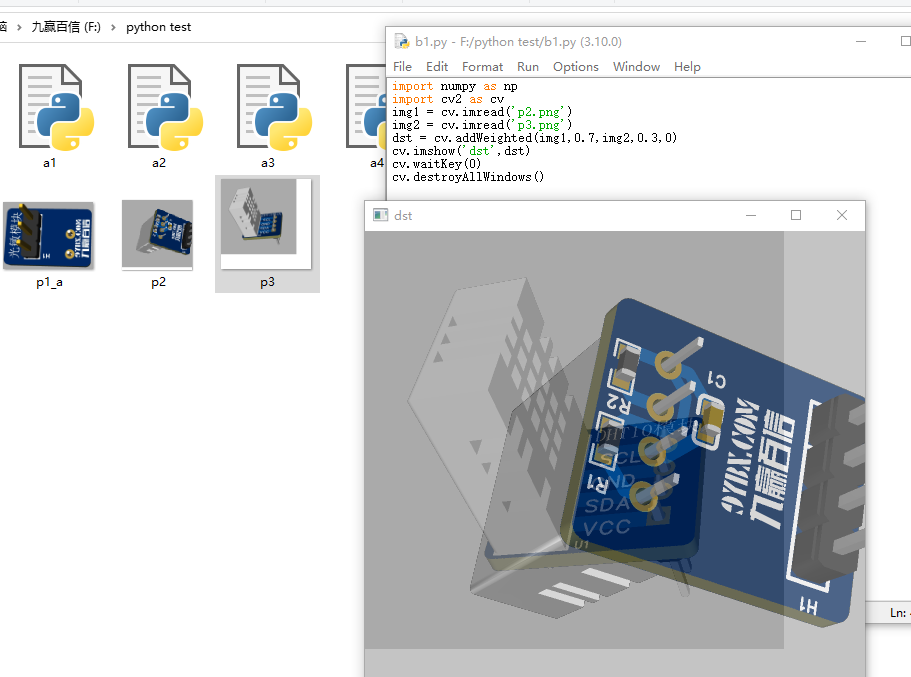
plt.subplot(236),plt.imshow(constant,'gray'),plt.title('CONSTANT')

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



## 任务7：图片操作之加减合成



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img1 = cv.imread('p2.png')

img2 = cv.imread('p3.png')

dst = cv.addWeighted(img1,0.7,img2,0.3,0)

cv.imshow('dst',dst)

cv.waitKey(0)

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



更多参考代码：

# Load two images

img1 = cv.imread('messi5.jpg')

img2 = cv.imread('opencv-logo-white.png')

# I want to put logo on top-left corner, So I create a ROI

rows,cols,channels = img2.shape

roi = img1[0:rows, 0:cols]

# Now create a mask of logo and create its inverse mask also

img2gray = cv.cvtColor(img2,cv.COLOR\_BGR2GRAY)

ret, mask = cv.threshold(img2gray, 10, 255, cv.THRESH\_BINARY)

mask\_inv = cv.bitwise\_not(mask)

# Now black-out the area of logo in ROI

img1\_bg = cv.bitwise\_and(roi,roi,mask = mask\_inv)

# Take only region of logo from logo image.

img2\_fg = cv.bitwise\_and(img2,img2,mask = mask)

# Put logo in ROI and modify the main image

dst = cv.add(img1\_bg,img2\_fg)

img1[0:rows, 0:cols ] = dst

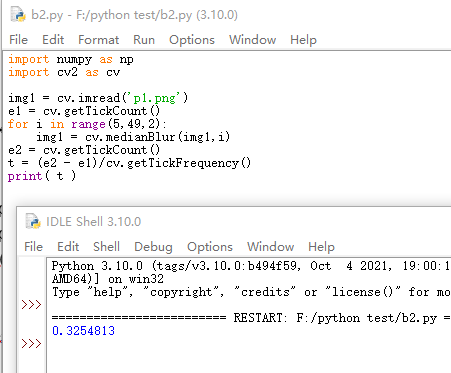
cv.imshow('res',img1)

cv.waitKey(0)

cv.destroyAllWindows()

<https://docs.opencv.org/4.x/d0/d86/tutorial_py_image_arithmetics.html>

## 任务8：性能计算



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img1 = cv.imread('p1.png')

e1 = cv.getTickCount()

for i in range(5,49,2):

img1 = cv.medianBlur(img1,i)

e2 = cv.getTickCount()

t = (e2 - e1)/cv.getTickFrequency()

print( t )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



## 任务9：图像空间

import cv2 as cv

flags = [i for i in dir(cv) if i.startswith('COLOR\_')]

print( flags )

['COLOR\_BAYER\_BG2BGR', 'COLOR\_BAYER\_BG2BGRA', 'COLOR\_BAYER\_BG2BGR\_EA', 'COLOR\_BAYER\_BG2BGR\_VNG', 'COLOR\_BAYER\_BG2GRAY', 'COLOR\_BAYER\_BG2RGB', 'COLOR\_BAYER\_BG2RGBA', 'COLOR\_BAYER\_BG2RGB\_EA', 'COLOR\_BAYER\_BG2RGB\_VNG', 'COLOR\_BAYER\_GB2BGR', 'COLOR\_BAYER\_GB2BGRA', 'COLOR\_BAYER\_GB2BGR\_EA', 'COLOR\_BAYER\_GB2BGR\_VNG', 'COLOR\_BAYER\_GB2GRAY', 'COLOR\_BAYER\_GB2RGB', 'COLOR\_BAYER\_GB2RGBA', 'COLOR\_BAYER\_GB2RGB\_EA', 'COLOR\_BAYER\_GB2RGB\_VNG', 'COLOR\_BAYER\_GR2BGR', 'COLOR\_BAYER\_GR2BGRA', 'COLOR\_BAYER\_GR2BGR\_EA', 'COLOR\_BAYER\_GR2BGR\_VNG', 'COLOR\_BAYER\_GR2GRAY', 'COLOR\_BAYER\_GR2RGB', 'COLOR\_BAYER\_GR2RGBA', 'COLOR\_BAYER\_GR2RGB\_EA', 'COLOR\_BAYER\_GR2RGB\_VNG', 'COLOR\_BAYER\_RG2BGR', 'COLOR\_BAYER\_RG2BGRA', 'COLOR\_BAYER\_RG2BGR\_EA', 'COLOR\_BAYER\_RG2BGR\_VNG', 'COLOR\_BAYER\_RG2GRAY', 'COLOR\_BAYER\_RG2RGB', 'COLOR\_BAYER\_RG2RGBA', 'COLOR\_BAYER\_RG2RGB\_EA', 'COLOR\_BAYER\_RG2RGB\_VNG', 'COLOR\_BGR2BGR555', 'COLOR\_BGR2BGR565', 'COLOR\_BGR2BGRA', 'COLOR\_BGR2GRAY', 'COLOR\_BGR2HLS', 'COLOR\_BGR2HLS\_FULL', 'COLOR\_BGR2HSV', 'COLOR\_BGR2HSV\_FULL', 'COLOR\_BGR2LAB', 'COLOR\_BGR2LUV', 'COLOR\_BGR2Lab', 'COLOR\_BGR2Luv', 'COLOR\_BGR2RGB', 'COLOR\_BGR2RGBA', 'COLOR\_BGR2XYZ', 'COLOR\_BGR2YCR\_CB', 'COLOR\_BGR2YCrCb', 'COLOR\_BGR2YUV', 'COLOR\_BGR2YUV\_I420', 'COLOR\_BGR2YUV\_IYUV', 'COLOR\_BGR2YUV\_YV12', 'COLOR\_BGR5552BGR', 'COLOR\_BGR5552BGRA', 'COLOR\_BGR5552GRAY', 'COLOR\_BGR5552RGB', 'COLOR\_BGR5552RGBA', 'COLOR\_BGR5652BGR', 'COLOR\_BGR5652BGRA', 'COLOR\_BGR5652GRAY', 'COLOR\_BGR5652RGB', 'COLOR\_BGR5652RGBA', 'COLOR\_BGRA2BGR', 'COLOR\_BGRA2BGR555', 'COLOR\_BGRA2BGR565', 'COLOR\_BGRA2GRAY', 'COLOR\_BGRA2RGB', 'COLOR\_BGRA2RGBA', 'COLOR\_BGRA2YUV\_I420', 'COLOR\_BGRA2YUV\_IYUV', 'COLOR\_BGRA2YUV\_YV12', 'COLOR\_BayerBG2BGR', 'COLOR\_BayerBG2BGRA', 'COLOR\_BayerBG2BGR\_EA', 'COLOR\_BayerBG2BGR\_VNG', 'COLOR\_BayerBG2GRAY', 'COLOR\_BayerBG2RGB', 'COLOR\_BayerBG2RGBA', 'COLOR\_BayerBG2RGB\_EA', 'COLOR\_BayerBG2RGB\_VNG', 'COLOR\_BayerGB2BGR', 'COLOR\_BayerGB2BGRA', 'COLOR\_BayerGB2BGR\_EA', 'COLOR\_BayerGB2BGR\_VNG', 'COLOR\_BayerGB2GRAY', 'COLOR\_BayerGB2RGB', 'COLOR\_BayerGB2RGBA', 'COLOR\_BayerGB2RGB\_EA', 'COLOR\_BayerGB2RGB\_VNG', 'COLOR\_BayerGR2BGR', 'COLOR\_BayerGR2BGRA', 'COLOR\_BayerGR2BGR\_EA', 'COLOR\_BayerGR2BGR\_VNG', 'COLOR\_BayerGR2GRAY', 'COLOR\_BayerGR2RGB', 'COLOR\_BayerGR2RGBA', 'COLOR\_BayerGR2RGB\_EA', 'COLOR\_BayerGR2RGB\_VNG', 'COLOR\_BayerRG2BGR', 'COLOR\_BayerRG2BGRA', 'COLOR\_BayerRG2BGR\_EA', 'COLOR\_BayerRG2BGR\_VNG', 'COLOR\_BayerRG2GRAY', 'COLOR\_BayerRG2RGB', 'COLOR\_BayerRG2RGBA', 'COLOR\_BayerRG2RGB\_EA', 'COLOR\_BayerRG2RGB\_VNG', 'COLOR\_COLORCVT\_MAX', 'COLOR\_GRAY2BGR', 'COLOR\_GRAY2BGR555', 'COLOR\_GRAY2BGR565', 'COLOR\_GRAY2BGRA', 'COLOR\_GRAY2RGB', 'COLOR\_GRAY2RGBA', 'COLOR\_HLS2BGR', 'COLOR\_HLS2BGR\_FULL', 'COLOR\_HLS2RGB', 'COLOR\_HLS2RGB\_FULL', 'COLOR\_HSV2BGR', 'COLOR\_HSV2BGR\_FULL', 'COLOR\_HSV2RGB', 'COLOR\_HSV2RGB\_FULL', 'COLOR\_LAB2BGR', 'COLOR\_LAB2LBGR', 'COLOR\_LAB2LRGB', 'COLOR\_LAB2RGB', 'COLOR\_LBGR2LAB', 'COLOR\_LBGR2LUV', 'COLOR\_LBGR2Lab', 'COLOR\_LBGR2Luv', 'COLOR\_LRGB2LAB', 'COLOR\_LRGB2LUV', 'COLOR\_LRGB2Lab', 'COLOR\_LRGB2Luv', 'COLOR\_LUV2BGR', 'COLOR\_LUV2LBGR', 'COLOR\_LUV2LRGB', 'COLOR\_LUV2RGB', 'COLOR\_Lab2BGR', 'COLOR\_Lab2LBGR', 'COLOR\_Lab2LRGB', 'COLOR\_Lab2RGB', 'COLOR\_Luv2BGR', 'COLOR\_Luv2LBGR', 'COLOR\_Luv2LRGB', 'COLOR\_Luv2RGB', 'COLOR\_M\_RGBA2RGBA', 'COLOR\_RGB2BGR', 'COLOR\_RGB2BGR555', 'COLOR\_RGB2BGR565', 'COLOR\_RGB2BGRA', 'COLOR\_RGB2GRAY', 'COLOR\_RGB2HLS', 'COLOR\_RGB2HLS\_FULL', 'COLOR\_RGB2HSV', 'COLOR\_RGB2HSV\_FULL', 'COLOR\_RGB2LAB', 'COLOR\_RGB2LUV', 'COLOR\_RGB2Lab', 'COLOR\_RGB2Luv', 'COLOR\_RGB2RGBA', 'COLOR\_RGB2XYZ', 'COLOR\_RGB2YCR\_CB', 'COLOR\_RGB2YCrCb', 'COLOR\_RGB2YUV', 'COLOR\_RGB2YUV\_I420', 'COLOR\_RGB2YUV\_IYUV', 'COLOR\_RGB2YUV\_YV12', 'COLOR\_RGBA2BGR', 'COLOR\_RGBA2BGR555', 'COLOR\_RGBA2BGR565', 'COLOR\_RGBA2BGRA', 'COLOR\_RGBA2GRAY', 'COLOR\_RGBA2M\_RGBA', 'COLOR\_RGBA2RGB', 'COLOR\_RGBA2YUV\_I420', 'COLOR\_RGBA2YUV\_IYUV', 'COLOR\_RGBA2YUV\_YV12', 'COLOR\_RGBA2mRGBA', 'COLOR\_XYZ2BGR', 'COLOR\_XYZ2RGB', 'COLOR\_YCR\_CB2BGR', 'COLOR\_YCR\_CB2RGB', 'COLOR\_YCrCb2BGR', 'COLOR\_YCrCb2RGB', 'COLOR\_YUV2BGR', 'COLOR\_YUV2BGRA\_I420', 'COLOR\_YUV2BGRA\_IYUV', 'COLOR\_YUV2BGRA\_NV12', 'COLOR\_YUV2BGRA\_NV21', 'COLOR\_YUV2BGRA\_UYNV', 'COLOR\_YUV2BGRA\_UYVY', 'COLOR\_YUV2BGRA\_Y422', 'COLOR\_YUV2BGRA\_YUNV', 'COLOR\_YUV2BGRA\_YUY2', 'COLOR\_YUV2BGRA\_YUYV', 'COLOR\_YUV2BGRA\_YV12', 'COLOR\_YUV2BGRA\_YVYU', 'COLOR\_YUV2BGR\_I420', 'COLOR\_YUV2BGR\_IYUV', 'COLOR\_YUV2BGR\_NV12', 'COLOR\_YUV2BGR\_NV21', 'COLOR\_YUV2BGR\_UYNV', 'COLOR\_YUV2BGR\_UYVY', 'COLOR\_YUV2BGR\_Y422', 'COLOR\_YUV2BGR\_YUNV', 'COLOR\_YUV2BGR\_YUY2', 'COLOR\_YUV2BGR\_YUYV', 'COLOR\_YUV2BGR\_YV12', 'COLOR\_YUV2BGR\_YVYU', 'COLOR\_YUV2GRAY\_420', 'COLOR\_YUV2GRAY\_I420', 'COLOR\_YUV2GRAY\_IYUV', 'COLOR\_YUV2GRAY\_NV12', 'COLOR\_YUV2GRAY\_NV21', 'COLOR\_YUV2GRAY\_UYNV', 'COLOR\_YUV2GRAY\_UYVY', 'COLOR\_YUV2GRAY\_Y422', 'COLOR\_YUV2GRAY\_YUNV', 'COLOR\_YUV2GRAY\_YUY2', 'COLOR\_YUV2GRAY\_YUYV', 'COLOR\_YUV2GRAY\_YV12', 'COLOR\_YUV2GRAY\_YVYU', 'COLOR\_YUV2RGB', 'COLOR\_YUV2RGBA\_I420', 'COLOR\_YUV2RGBA\_IYUV', 'COLOR\_YUV2RGBA\_NV12', 'COLOR\_YUV2RGBA\_NV21', 'COLOR\_YUV2RGBA\_UYNV', 'COLOR\_YUV2RGBA\_UYVY', 'COLOR\_YUV2RGBA\_Y422', 'COLOR\_YUV2RGBA\_YUNV', 'COLOR\_YUV2RGBA\_YUY2', 'COLOR\_YUV2RGBA\_YUYV', 'COLOR\_YUV2RGBA\_YV12', 'COLOR\_YUV2RGBA\_YVYU', 'COLOR\_YUV2RGB\_I420', 'COLOR\_YUV2RGB\_IYUV', 'COLOR\_YUV2RGB\_NV12', 'COLOR\_YUV2RGB\_NV21', 'COLOR\_YUV2RGB\_UYNV', 'COLOR\_YUV2RGB\_UYVY', 'COLOR\_YUV2RGB\_Y422', 'COLOR\_YUV2RGB\_YUNV', 'COLOR\_YUV2RGB\_YUY2', 'COLOR\_YUV2RGB\_YUYV', 'COLOR\_YUV2RGB\_YV12', 'COLOR\_YUV2RGB\_YVYU', 'COLOR\_YUV420P2BGR', 'COLOR\_YUV420P2BGRA', 'COLOR\_YUV420P2GRAY', 'COLOR\_YUV420P2RGB', 'COLOR\_YUV420P2RGBA', 'COLOR\_YUV420SP2BGR', 'COLOR\_YUV420SP2BGRA', 'COLOR\_YUV420SP2GRAY', 'COLOR\_YUV420SP2RGB', 'COLOR\_YUV420SP2RGBA', 'COLOR\_YUV420p2BGR', 'COLOR\_YUV420p2BGRA', 'COLOR\_YUV420p2GRAY', 'COLOR\_YUV420p2RGB', 'COLOR\_YUV420p2RGBA', 'COLOR\_YUV420sp2BGR', 'COLOR\_YUV420sp2BGRA', 'COLOR\_YUV420sp2GRAY', 'COLOR\_YUV420sp2RGB', 'COLOR\_YUV420sp2RGBA', 'COLOR\_mRGBA2RGBA']

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2 as cv

import numpy as np

cap = cv.VideoCapture(0)

while(1):

# Take each frame

\_, frame = cap.read()

# Convert BGR to HSV

hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)

# define range of blue color in HSV

lower\_blue = np.array([110,50,50])

upper\_blue = np.array([130,255,255])

# Threshold the HSV image to get only blue colors

mask = cv.inRange(hsv, lower\_blue, upper\_blue)

# Bitwise-AND mask and original image

res = cv.bitwise\_and(frame,frame, mask= mask)

cv.imshow('frame',frame)

cv.imshow('mask',mask)

cv.imshow('res',res)

k = cv.waitKey(5) & 0xFF

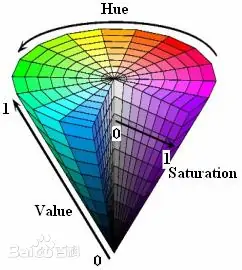
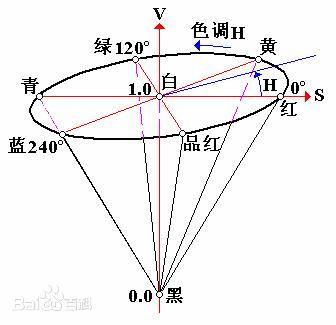
if k == 27:

break

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HSV(Hue, Saturation, Value)是根据颜色的直观特性由A. R. Smith在1978年创建的一种颜色空间, 也称六角锥体模型(Hexcone Model)。



这个模型中颜色的参数分别是：**色调（H），饱和度（S），明度（V）**

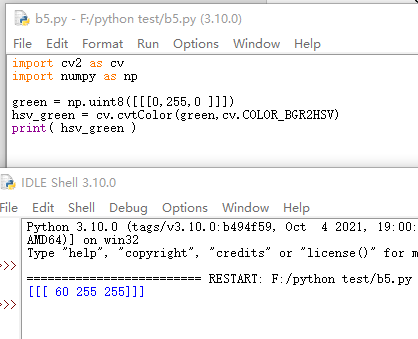
|  |  |
| --- | --- |
| **色调H** | 用角度度量，取值范围为0°～360°，从红色开始按逆时针方向计算，红色为0°，绿色为120°,蓝色为240°。它们的补色是：黄色为60°，青色为180°,紫色为300°； |
| **饱和度S** | 饱和度S表示颜色接近光谱色的程度。一种颜色，可以看成是某种光谱色与白色混合的结果。其中光谱色所占的比例愈大，颜色接近光谱色的程度就愈高，颜色的饱和度也就愈高。饱和度高，颜色则深而艳。光谱色的白光成分为0，饱和度达到最高。通常取值范围为0%～100%，值越大，颜色越饱和。 |
| **明度V** | 明度表示颜色明亮的程度，对于光源色，明度值与发光体的光亮度有关；对于物体色，此值和物体的透射比或反射比有关。通常取值范围为0%（黑）到100%（白）。  RGB和CMY颜色模型都是面向硬件的，而HSV（Hue Saturation Value）颜色模型是面向用户的。 |

HSV模型的三维表示从RGB立方体演化而来。设想从RGB沿立方体对角线的白色顶点向黑色顶点观察，就可以看到立方体的六边形外形。六边形边界表示色彩，水平轴表示纯度，明度沿垂直轴测量。

更多HSV知识：

<https://zhuanlan.zhihu.com/p/67930839>

BGR->HSV



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2 as cv

import numpy as np

green = np.uint8([[[0,255,0 ]]])

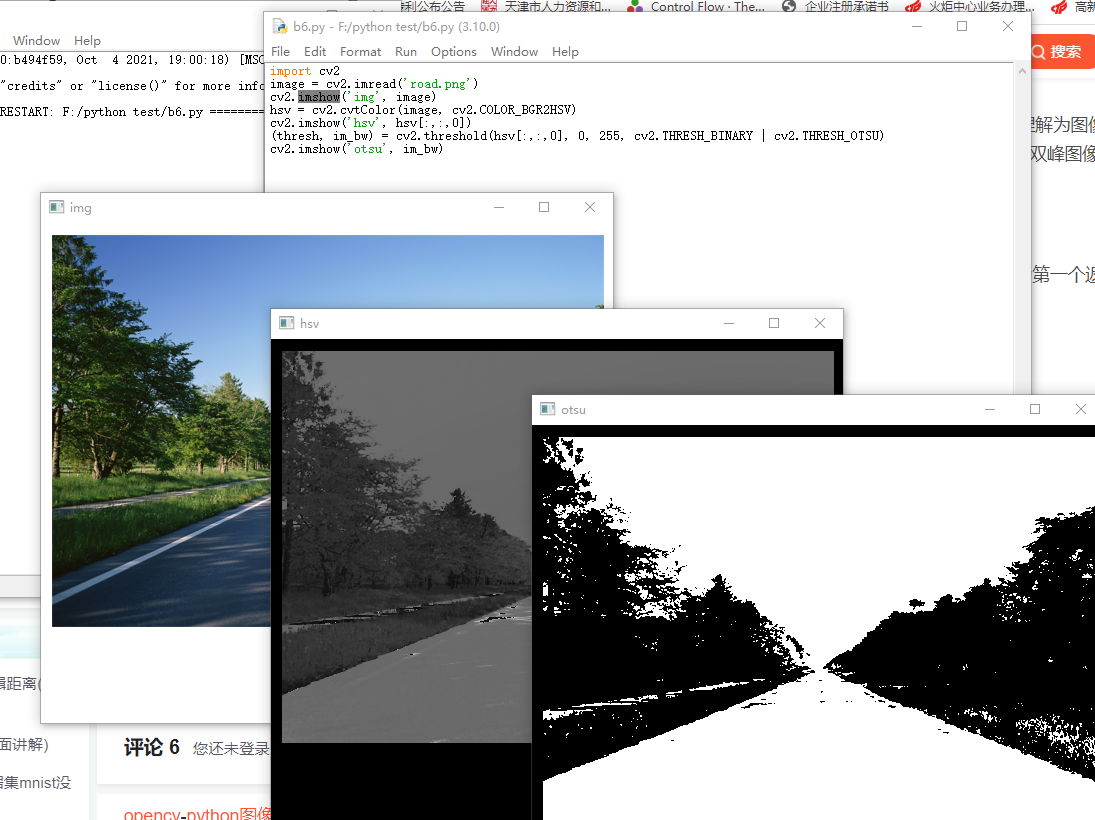
hsv\_green = cv.cvtColor(green,cv.COLOR\_BGR2HSV)

print( hsv\_green )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



图像的绘图和阈值的示例



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2

image = cv2.imread('road.png')

cv2.imshow('img', image)

hsv = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

cv2.imshow('hsv', hsv[:,:,0])

(thresh, im\_bw) = cv2.threshold(hsv[:,:,0], 0, 255, cv2.THRESH\_BINARY | cv2.THRESH\_OTSU)

cv2.imshow('otsu', im\_bw)

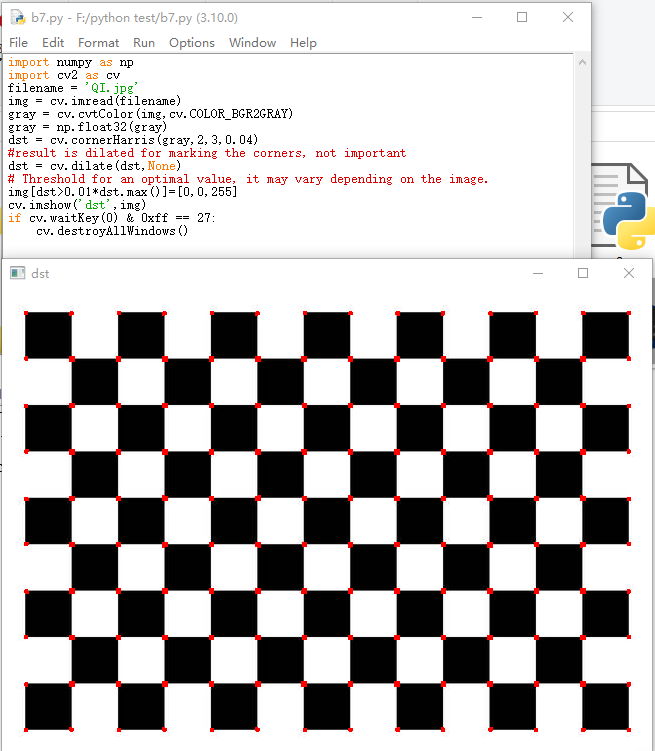
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



任务10：角点检测

Harris Corner Detection

<https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html>



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

filename = 'QI.jpg'

img = cv.imread(filename)

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

gray = np.float32(gray)

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

#result is dilated for marking the corners, not important

dst = cv.dilate(dst,None)

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

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

cv.imshow('dst',img)

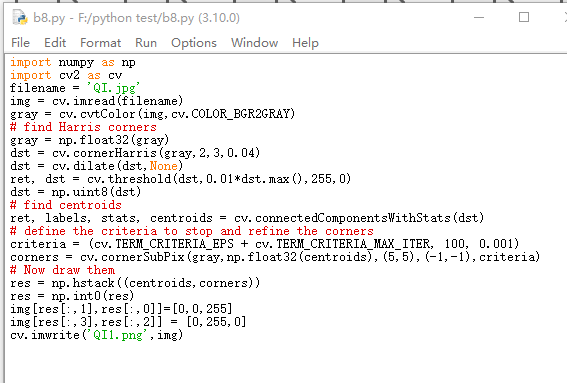
if cv.waitKey(0) & 0xff == 27:

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



亚像素角点检测



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

filename = 'QI.jpg'

img = cv.imread(filename)

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

# find Harris corners

gray = np.float32(gray)

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

dst = cv.dilate(dst,None)

ret, dst = cv.threshold(dst,0.01\*dst.max(),255,0)

dst = np.uint8(dst)

# find centroids

ret, labels, stats, centroids = cv.connectedComponentsWithStats(dst)

# define the criteria to stop and refine the corners

criteria = (cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER, 100, 0.001)

corners = cv.cornerSubPix(gray,np.float32(centroids),(5,5),(-1,-1),criteria)

# Now draw them

res = np.hstack((centroids,corners))

res = np.int0(res)

img[res[:,1],res[:,0]]=[0,0,255]

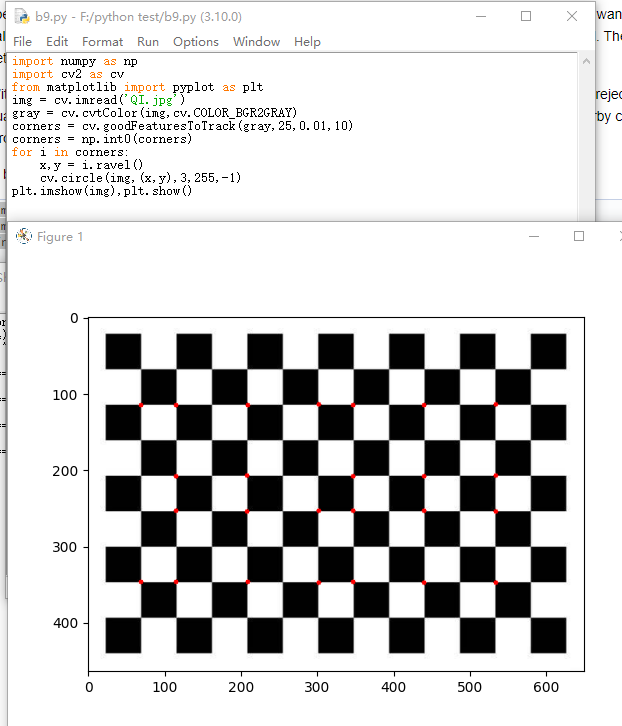
img[res[:,3],res[:,2]] = [0,255,0]

cv.imwrite('QI1.png',img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Shi-Tomasi角点检测



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

from matplotlib import pyplot as plt

img = cv.imread('QI.jpg')

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

corners = cv.goodFeaturesToTrack(gray,25,0.01,10)

corners = np.int0(corners)

for i in corners:

x,y = i.ravel()

cv.circle(img,(x,y),3,255,-1)

plt.imshow(img),plt.show()

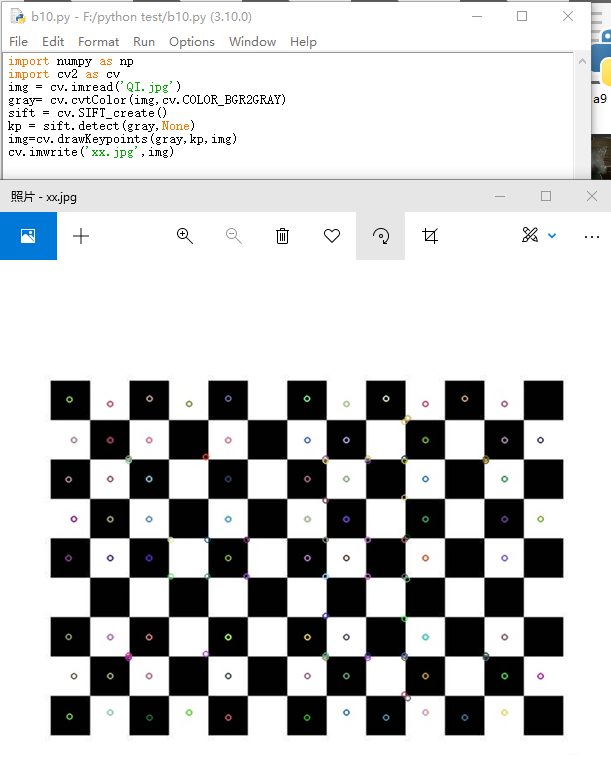
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



SIFT (Scale-Invariant Feature Transform)

<https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html>

SIFT是由UBC（university of British Column）的教授David Lowe 于1999年提出， 并在2004年得以完善的一种检测图像关键点（key points , 或者称为图像的interest points(兴趣点) )， 并对关键点提取其局部尺度不变特征的描绘子， 采用这个描绘子进行用于对两幅相关的图像进行匹配（matching）。



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img = cv.imread('QI.jpg')

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

sift = cv.SIFT\_create()

kp = sift.detect(gray,None)

img=cv.drawKeypoints(gray,kp,img)

cv.imwrite('xx.jpg',img)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



img=cv.drawKeypoints(gray,kp,img,flags=cv.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

cv.imwrite('sift\_keypoints.jpg',img)

sift = cv.SIFT\_create()

kp, des = sift.detectAndCompute(gray,None)

SURF (Speeded-Up Robust Features)

Surf算法是对Sift算法的一种改进，主要是在算法的执行效率上，比Sift算法来讲运行更快！

<https://docs.opencv.org/4.x/df/dd2/tutorial_py_surf_intro.html>

FAST Algorithm for Corner Detection

更多的

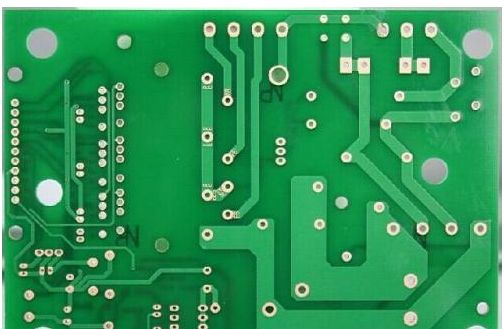
<https://docs.opencv.org/4.x/db/d27/tutorial_py_table_of_contents_feature2d.html>

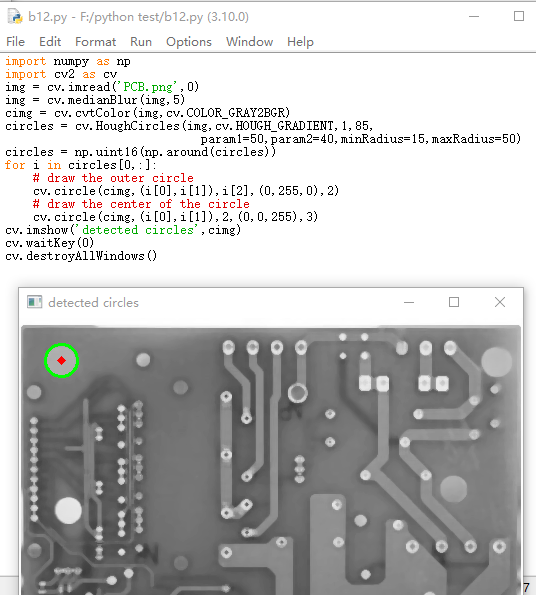
## 任务10. 圆的识别和检测

<https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html>



在一个图片中找出圆，可以用于机器视觉的辅助定位。





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

img = cv.imread('PCB.png',0)

img = cv.medianBlur(img,5)

cimg = cv.cvtColor(img,cv.COLOR\_GRAY2BGR)

circles = cv.HoughCircles(img,cv.HOUGH\_GRADIENT,1,85,

param1=50,param2=40,minRadius=15,maxRadius=50)

circles = np.uint16(np.around(circles))

for i in circles[0,:]:

# draw the outer circle

cv.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)

# draw the center of the circle

cv.circle(cimg,(i[0],i[1]),2,(0,0,255),3)

cv.imshow('detected circles',cimg)

cv.waitKey(0)

cv.destroyAllWindows()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



cvHoughCircles( //cvHoughCircles函数需要估计每一个像素梯度的方向，

//因此会在内部自动调用cvSobel,而二值边缘图像的处理是比较难的

img1,storage,CV\_HOUGH\_GRADIENT,

1, //累加器图像的分辨率,增大则分辨率变小

18, //很重要的一个参数，告诉两个圆之间的距离的最小距离，如果已知一副图像，可以先行计

//算出符合自己需要的两个圆之间的最小距离。

100, //canny算法的阈值上限，下限为一半（即100以上为边缘点，50以下抛弃，中间视是否相连而//定）

25, //决定成圆的多寡 ，一个圆上的像素超过这个阈值，则成圆，否则丢弃

32,//最小圆半径，这个可以通过图片确定你需要的圆的区间范围

45 //最大圆半径

);

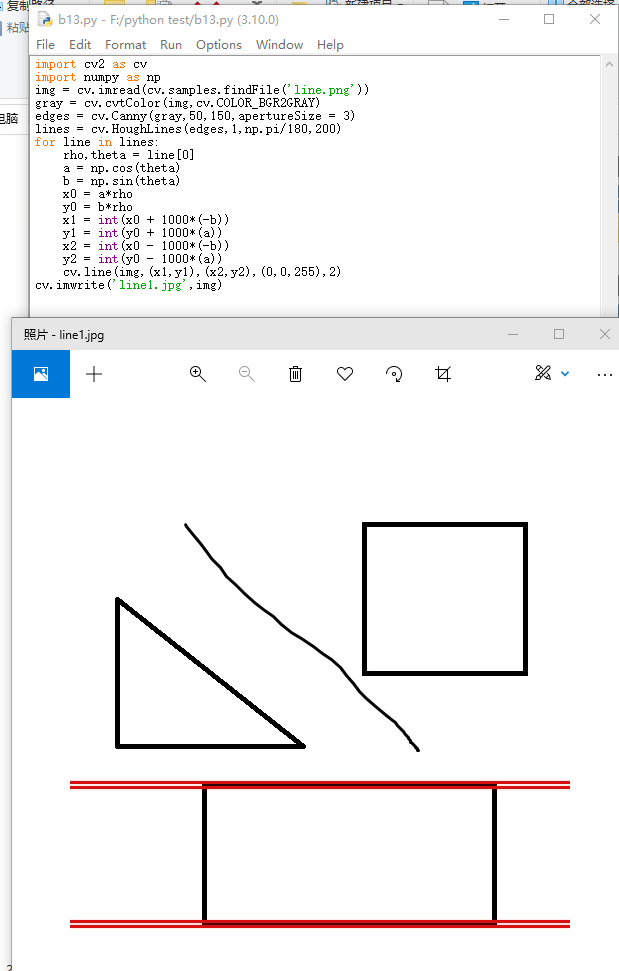
|  |  |
| --- | --- |
| circles = cv.HoughCircles(img,cv.HOUGH\_GRADIENT,1,85,  param1=50,param2=40,minRadius=15,maxRadius=50) | |
| img |  |
| cv.HOUGH\_GRADIENT |  |
| 1 | 累加器图像的分辨率,增大则分辨率变小 |
| 85 | 很重要的一个参数，告诉两个圆之间的距离的最小距离 |
| param1=50 | canny算法的阈值上限，下限为一半（即100以上为边缘点，50以下抛弃，中间视是否相连而//定 |
| param2=40 | 决定成圆的多寡 ，一个圆上的像素超过这个阈值，则成圆，否则丢弃 |
| minRadius=15 | 最小圆半径 |
| maxRadius=50 | 最大圆半径 |

## 任务11.直线的识别和检测

直线的识别和检测可以作为车道线识别的基础和元件

<https://docs.opencv.org/4.x/d6/d10/tutorial_py_houghlines.html>





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import cv2 as cv

import numpy as np

img = cv.imread(cv.samples.findFile('line.png'))

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

edges = cv.Canny(gray,50,150,apertureSize = 3)

lines = cv.HoughLines(edges,1,np.pi/180,200)

for line in lines:

rho,theta = line[0]

a = np.cos(theta)

b = np.sin(theta)

x0 = a\*rho

y0 = b\*rho

x1 = int(x0 + 1000\*(-b))

y1 = int(y0 + 1000\*(a))

x2 = int(x0 - 1000\*(-b))

y2 = int(y0 - 1000\*(a))

cv.line(img,(x1,y1),(x2,y2),(0,0,255),2)

cv.imwrite('line1.jpg',img)

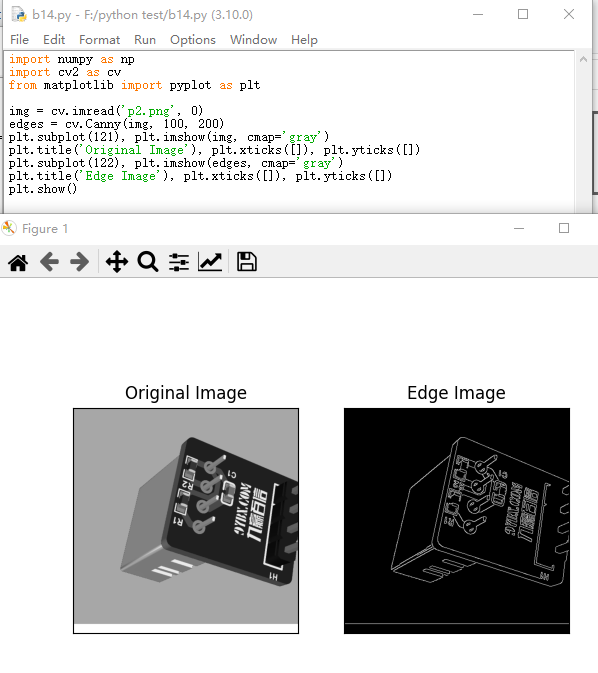
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



|  |  |
| --- | --- |
| HoughLines(InputArray image, OutputArray lines, double rho, double theta, int threshold, double srn=0, double stn=0, double min\_theta=0, double max\_theta=CV\_PI ) | |
| cv.HoughLines(edges,1,np.pi/180,200) | |
|  | image参数表示边缘检测的输出图像，该图像为单通道8位二进制图像 |
|  | rho参数表示参数极径 r 以像素值为单位的分辨率，这里一般使用1像素 |
|  | theta参数表示参数极角の以弧度为单位的分辨率，这里使用1度 |
|  | threshold参数表示检测一条直线所需最少的曲线交点 |
|  | lines参数表示储存着检测到的直线的参数对 (r,の)的容器 |
|  | srn参数、stn参数默认都为0。如果srn = 0且stn = 0，则使用经典的Hough变换 |
|  | min\_theta参数表示对于标准和多尺度Hough变换，检查线条的最小角度 |
|  | max\_theta参数表示对于标准和多尺度Hough变换，检查线条的最大角度 |
|  |  |
| HoughLinesP(image, rho, theta, threshold[, lines[, minLineLength[, maxLineGap]]]) | |
|  | image参数表示边缘检测的输出图像，该图像为单通道8位二进制图像。 |
|  | rho参数表示参数极径 r 以像素值为单位的分辨率，这里一般使用 1 像素 |
|  | theta参数表示参数极角の以弧度为单位的分辨率，这里使用 1度 |
|  | threshold参数表示检测一条直线所需最少的曲线交点。 |
|  | lines参数表示储存着检测到的直线的参数的容器，也就是线段两个端点的坐标 |
|  | minLineLength参数表示能组成一条直线的最少点的数量，点数量不足的直线将被抛弃 |
|  | maxLineGap参数表示能被认为在一条直线上的亮点的最大距离。 |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Canny边缘检测 | |
| edges=cv.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]]) | |
| edges=cv.Canny(dx, dy, threshold1, threshold2[, edges[, L2gradient]]) | |
| dx | 16-bit 输入图像的x导数 (CV\_16SC1 or CV\_16SC3). |
| dy | 16-bit 输入图像的y导数 (和dx有相同类型) |
| edges | 输出的边缘地图; 单通道8-bit 图像, 与输入图像有相同大小 |
| threshold1 | 第一个阈值，低阈值 |
| threshold2 | 第二个阈值，高阈值 |
| L2gradient | 如果为真，则使用更精确的L2范数进行计算（即两个方向的倒数的平方和再开方），否则使用L1范数（直接将两个方向导数的绝对值相加） |

边缘检测



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import cv2 as cv

from matplotlib import pyplot as plt

img = cv.imread('p2.png', 0)

edges = cv.Canny(img, 100, 200)

plt.subplot(121), plt.imshow(img, cmap='gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122), plt.imshow(edges, cmap='gray')

plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

