
# 图像处理入门

# 从 Python 到 Opencv

[西南科技大学 IPCS 9 班]曾均睿 著

# 目录

# Python 编程

这里主要针对有一定基础的同学，在 Python 编程中，掌握基础语法和数据类型是非常重要的。它们是构建程序的基石，是提供解决问题和开发应用的工具。在这里，我将简单介绍一些常用的语法和数据类型。

## 一、环境搭建

详细请参考此篇纯净 Python 环境的安装以及配置 PyCharm 编辑器。

## 二、基础语法

### （1）If ... Else 语句

#### else 示例

```
a = 200
b = 33
if b > a:
 print("b is greater than a")
elif a == b:
 print("a and b are equal")
else:
 print("a is greater than b")
```

#### 简写 If ... Else

```
a = 2
b = 330
print("A") if a > b else print("B")
```

### （2）While 循环

#### break 语句

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

如果使用 break 语句，即使 while 条件为真，我们也可以停止循环。

### continue 语句

```
1. i = 0
2. while i < 6:
3.   i += 1
4.   if i == 3:
5.     continue
6.   print(i)
```

如果使用 continue 语句，我们可以停止当前的迭代，并继续下一个。

## （3）For 循环

### range() 函数

```
1. for x in range(6):
2.     print(x)
```

### 嵌套循环

```
1. adj = ["red", "big", "tasty"]
2. fruits = ["apple", "banana", "cherry"]
3. 
4. for x in adj:
5.     for y in fruits:
6.         print(x, y)
```

### pass 语句

```
1. for x in [0, 1, 2]:
2.   pass
```

### break 语句

```
1. fruits = ["apple", "banana", "cherry"]
2. for x in fruits:
3.     print(x)
4.     if x == "banana":
5.         break
```

continue 语句

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

# 三、集合数据类型

Python 编程语言中有四种集合数据类型：

- **列表（List）** 是一种有序和可更改的集合。允许重复的成员。
- **元组（Tuple）** 是一种有序且不可更改的集合。允许重复的成员。
- **集合（Set）** 是一个无序和无索引的集合。没有重复的成员。
- **词典（Dictionary）**是一个无序，可变和有索引的集合。无重复的成员。

这里我们重点来了解列表。

**列表**

列表是一个有序且可更改的集合。在 Python 中，列表用方括号编写。

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
print(len(thislist))
```

**索引**

```
thislist = ["apple", "banana", "cherry", "kiwi", "melon", "mango"]
print(thislist[-1])
print(thislist[1])
print(thislist[2:5])
print(thislist[:4])
print(thislist[2:])
print(thislist[-4:-1])
for x in thislist:
  print(x)
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
thislist.append("orange")
print(thislist)
thislist.remove("banana")
thislist.pop()
```

元组、集合、词典可通过文章末尾的资源共享进行学习

# 四、函数

**调用函数**

如需调用函数，请使用函数名称后跟括号：

```
def my_function():
    print("Hello from a function")

my_function()
```

**参数**

信息可以作为参数传递给函数。

参数在函数名后的括号内指定。您可以根据需要添加任意数量的参数，只需用逗号分隔即可。

下面的例子有一个带参数（fname）的函数。当调用此函数时，我们传递一个名字，在函数内部使用它来打印全名：

```
def my_function(fname):
    print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

**返回值**

如需使函数返回值，请使用 `return` 语句：

```
def my_function(x):
    return 5 * x


print(my_function(3))
print(my_function(5))
```

**pass 语句**

函数定义不能为空，但是如果您出于某种原因写了无内容的函数定义，请使用 pass 语句来避免错误。

```
def myfunction():
    pass
```

**lambda 函数**

语法

lambda *arguments* : *expression*

```
def myfunc(n):
    return lambda a: a * n
```

# 五、类和对象

Python 是一种面向对象的编程语言。

Python 中的几乎所有东西都是对象，拥有属性和方法。

类（Class）类似对象构造函数，或者是用于创建对象的"蓝图"。

**创建类**

如需创建类，使用 class 关键字。

使用名为 x 的属性，创建一个名为 MyClass 的类：

```
class MyClass:
    x = 5
```

**创建对象**

创建一个名为 p1 的对象，并打印 x 的值：

```
p1 = MyClass()
print(p1.x)
```

**__init__() 函数**

上面的例子是最简单形式的类和对象，在实际应用程序中并不真正有用。要理解类的含义，我们必须先了解内置的 __init__() 函数。所有类都有一个名为 __init__() 的函数，它始终在启动类时执行。使用 __init__() 函数将值赋给对象属性，或者在创建对象时需要执行的其他操作：

创建名为 Person 的类，使用 __init__() 函数为 name 和 age 赋值：

```
1. class Person:
2.     def __init__(self, name, age):
3.         self.name = name
4.         self.age = age
5.
6. p1 = Person("John", 36)
7. print(p1.name)
8. print(p1.age)
```

**对象方法**

对象也可以包含方法。对象中的方法是属于该对象的函数。让我们在 Person 类中创建方法：

插入一个打印问候语的函数，并在 p1 对象上执行它：

```
1. class Person:
2.     def __init__(self, name, age):
3.         self.name = name
4.         self.age = age
5.
6.     def myfunc(self):
7.         print("Hello my name is " + self.name)
8.
9.
10. p1 = Person("John", 36)
11. p1.myfunc()
```

**self 参数**

self 参数是对类的当前实例的引用，用于访问属于该类的变量。它不必被命名为 self，您可以随意调用它，但它必须是类中任意函数的首个参数。（这里建议就使用 self）

# Opencv 图像库

OpenCV（Open Source Computer Vision）是一个开源的计算机视觉库，它提供了丰富的图像处理和计算机视觉算法。OpenCV 最初是由 Intel 开发，并于 2000 年首次发布。现在，OpenCV 由一个庞大的社区维护和支持，广泛应用于学术研究、工业应用和个人项目中。

**pip 安装**

pip install opencv-python -i https://pypi.tuna.tsinghua.edu.cn/simple

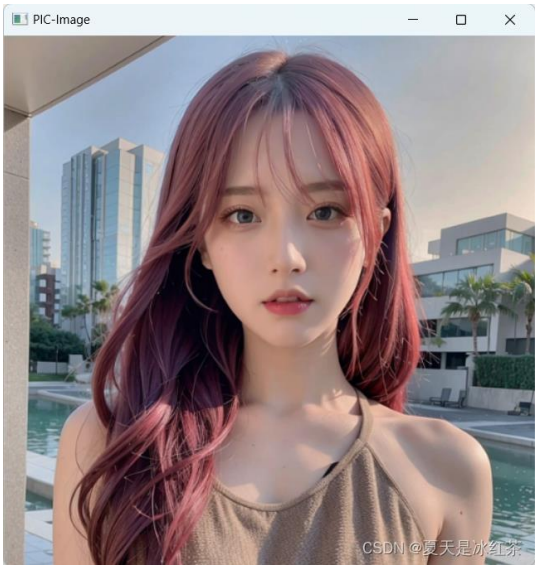
pip install opencv-contrib-python -i https://pypi.tuna.tsinghua.edu.cn/simple

## 一、如何读取图像和视频并调用摄像头

**读取图像**

```
1. import cv2
2.
3. pic_path="source/AI.png"
4. img = cv2.imread(pic_path)
5. cv2.imshow("PIC-Image",img)
6. cv2.waitKey(0)
```

成功读取了图像。




**读取视频**

```
import cv2
frame_Width = 640
frame_Height = 480
cap = cv2.VideoCapture("E:/pycharmlujin/Opencv/Opencv_learning/Resources/test_video.mp4")
while True:
    success, img = cap.read()
    img = cv2.resize(img, (frame_Width, frame_Height))
    cv2.imshow("Result", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

运行的效果图如下：


**调用摄像头**

```
import cv2
frameWidth = 640
frameHeight = 480
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)    #设置宽度
cap.set(4, frameHeight)   #设置高度
cap.set(10,150)   #设置亮度
while True:
    success, img = cap.read()
    cv2.imshow("Result", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

这次打开我们的网络摄像头，我们看到了**李航老师的《统计学习方法》：**

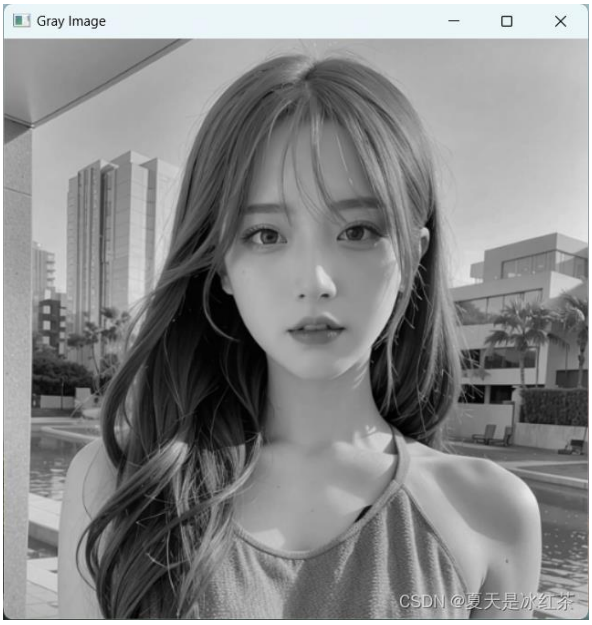

# 二、Opencv 的五个基本功能

**main 文件**

```
import cv2
import numpy as np

img = cv2.imread("source/AI.png")
kernel = np.ones((5,5),np.uint8)

imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)
imgCanny = cv2.Canny(img,150,200)
imgDilation = cv2.dilate(imgCanny,kernel,iterations=1)
imgEroded = cv2.erode(imgDilation,kernel,iterations=1)

cv2.imshow("Gray Image",imgGray)
cv2.imshow("Blur Image",imgBlur)
cv2.imshow("Canny Image",imgCanny)
cv2.imshow("Dilation Image",imgDilation)
cv2.imshow("Eroded Image",imgEroded)
cv2.waitKey(0)
```

**Grayscale Image**

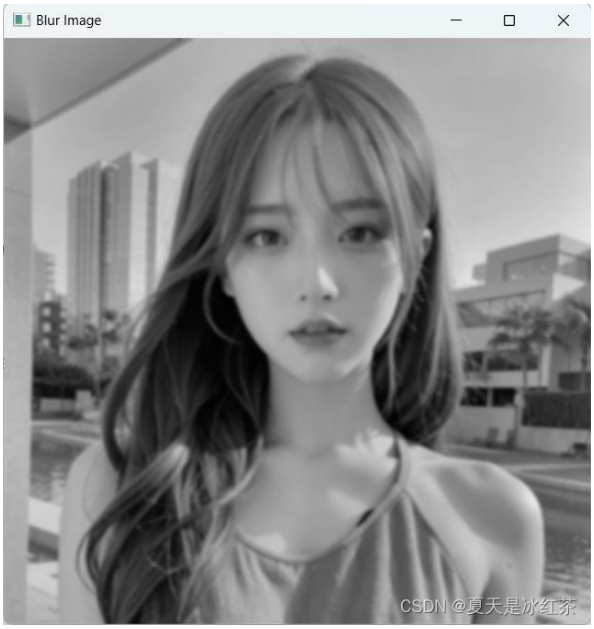
```
1.  imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```




将原来的 RGB 格式的图像转换为灰度空间，像素只有明暗程度，丢失了颜色信息。

**Gaussian Blur**

```
1.  imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)
```




我们添加上高斯模糊，可以明显的发现它与灰度图像的区别，较为模糊。

**Canny Edge detection**

```
1.  imgCanny = cv2.Canny(img,150,200)
```


Canny 检测在边缘检测当中，比起其他的检测效果要好些。

Image Dilation

```
1.  imgDilation = cv2.dilate(imgCanny,kernel,iterations=1)
```


这是图像处理的膨胀，在对 Canny 检测后的图像修改下，它的边缘线条变粗。

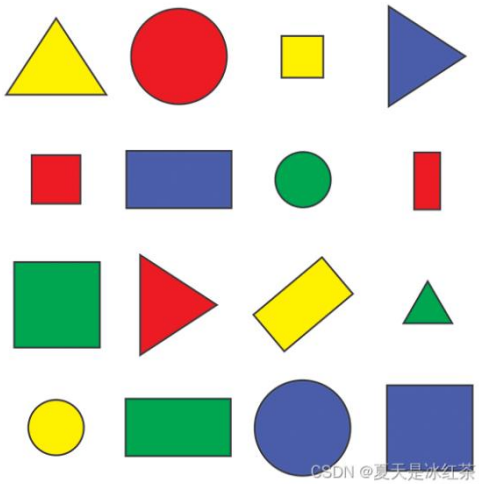
**Eroded Image**

```
1.  imgEroded = cv2.erode(imgDilation,kernel,iterations=1)
```


又在膨胀后的图片下进行图像侵蚀。

# 三、如何裁剪和调整图像大小

这是我所用的素材图：



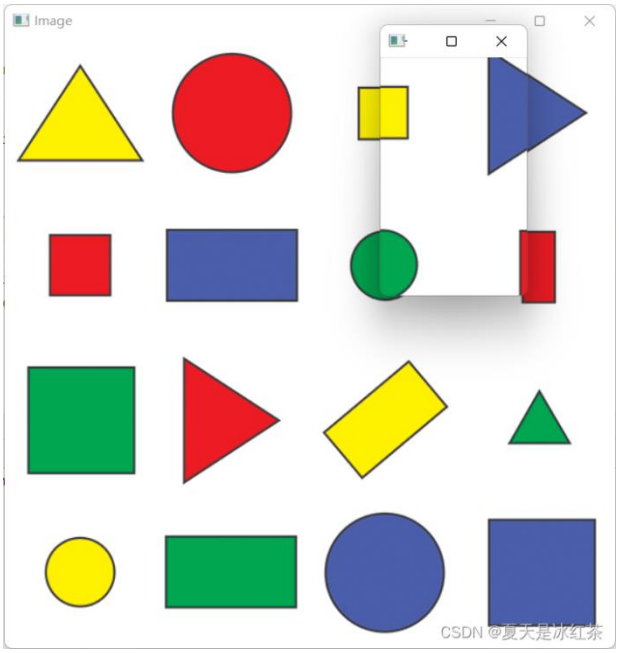

也可以寻找一个合适的图片，只需要更改下，图片的路径即可。

**main 文件**

```
1.  import cv2
```

```
import numpy as np

img = cv2.imread("Resources/shapes.png")
print(img.shape)

imgResize = cv2.resize(img,(1000,500))
print(imgResize.shape)

imgCropped = img[20:245,357:495]

cv2.imshow("Image",img)
# cv2.imshow("Image Resize",imgResize)
cv2.imshow("Image Cropped",imgCropped)

cv2.waitKey(0)
```

运行的效果如下，这里主要是想让大家了解怎么去裁剪图像。




# 四、如何绘制形状和文本

### （1）画线

绘制一条线，我们需要向 cv2.line()函数当中，传入线条的始末坐标。在此之前，还需要用 numpy 创建一个黑色的背景。

```
import numpy as np
import cv2

img = np.zeros((450,450,3),np.uint8)
cv2.line(img,(0,0),(450,450),(255,0,0),3)

cv2.imshow('imgline',img)
cv2.waitKey(0)
```

**（2）画矩形**

我们需要向 **cv2.rectangle()**函数中提供左上角和右下角。

```
cv2.rectangle(img,(126,0),(400,126),(0,255,0),3)
```

**（3）画圆圈**

绘制一个圆，需要提供其中心坐标和半径。我们将在上面绘制的矩形内绘制一个圆。

```
cv2.circle(img,(263,63), 63, (0,0,255), -1)
```

**（4）画椭圆**

绘制椭圆，一个参数是其中心坐标(x, y)，另一个参数是长轴，短轴的长度。**angle** 是椭圆沿逆时针旋转的角度。**startAngle** 和 **endAngle** 表示从主轴沿逆时针方向测量的椭圆弧的开始和结束

```
cv2.ellipse(img,(250,250),(120,50),0,0,180,255,-1)
```

**（5）画多边形**

首先，需要的是顶点的坐标，将这些点组成形状为 **Rows*1*2** 的数组，其中 **Rows** 是顶点数，并且其类型应为 **int32**。在这里，我们绘制了一个带有四个顶点的黄色小多边形。

```
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
pts = pts.reshape((-1,1,2))
cv2.polylines(img,[pts],True,(0,255,255))
```

如果第三个参数为 **False**，将会得到一条连接所有点的折线，而不是闭合形状。**cv2.polylines()**可用于绘制多条线。只需创建要绘制的所有线条的列表，然后将其传递给函数即可。所有线条将单独绘制。

**（6）向图像添加文本**

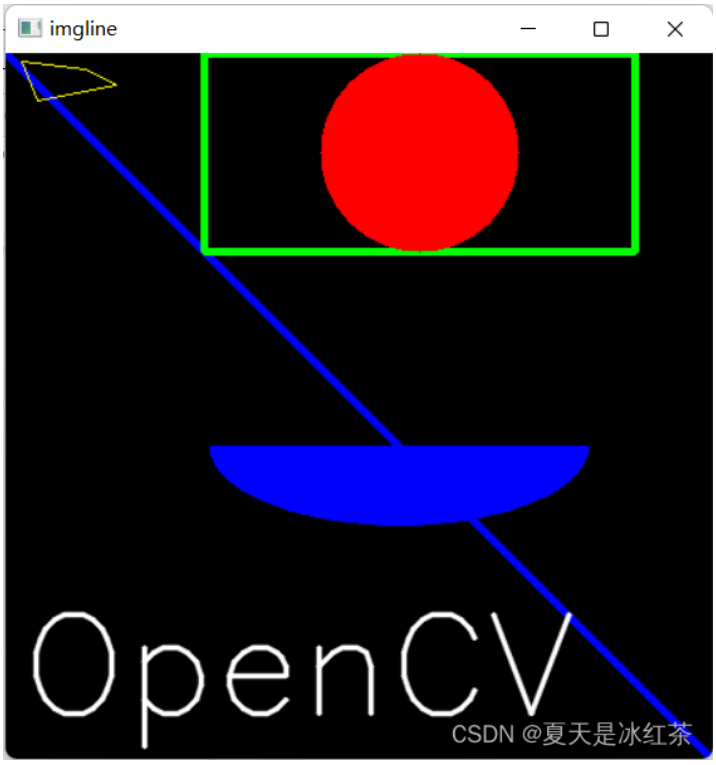
这其中的参数分别是文本，左下角坐标，字体的类型，字体大小，颜色，厚度，线条的类型。

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV',(10,420), font, 3,(255,255,255),2,cv2.LINE_AA)
```

main 文件

```
import numpy as np
import cv2
##############创建背景##################
img = np.zeros((450,450,3),np.uint8)
##############对角线条##################
cv2.line(img,(0,0),(450,450),(255,0,0),3)
##############画矩形框##################
cv2.rectangle(img,(126,0),(400,126),(0,255,0),3)
##############画实心圆##################
cv2.circle(img,(263,63), 63, (0,0,255), -1)
##############画椭圆形##################
cv2.ellipse(img,(250,250),(120,50),0,0,180,255,-1)
##############画多边形##################
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
pts = pts.reshape((-1,1,2))
cv2.polylines(img,[pts],True,(0,255,255))
##############放置文本##################
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV',(10,420), font, 3,(255,255,255),2,cv2.LINE_AA)

cv2.imshow('imgline',img)
cv2.waitKey(0)
```

实际效果图：




# 五、透视变换修正

在一堆的扑克牌当中，要如何使其修正呢？请看这里。

此处，我们的目标是要将黑桃 K 取出，使其修正。它的最后效果是这样的。


效果非常的好，这里需要用到透视变换的知识。此处要重点介绍一下 Opencv 的两个函数

- **cv2.getPerspectiveTransform（src, dst）从四对对应点计算透视变，src 源图像中四边形顶点的坐标，dst 目标图像中相应四边形顶点的坐标。**
- **warpPerspective(src, M, dsize)，对图像应用透视变换。**

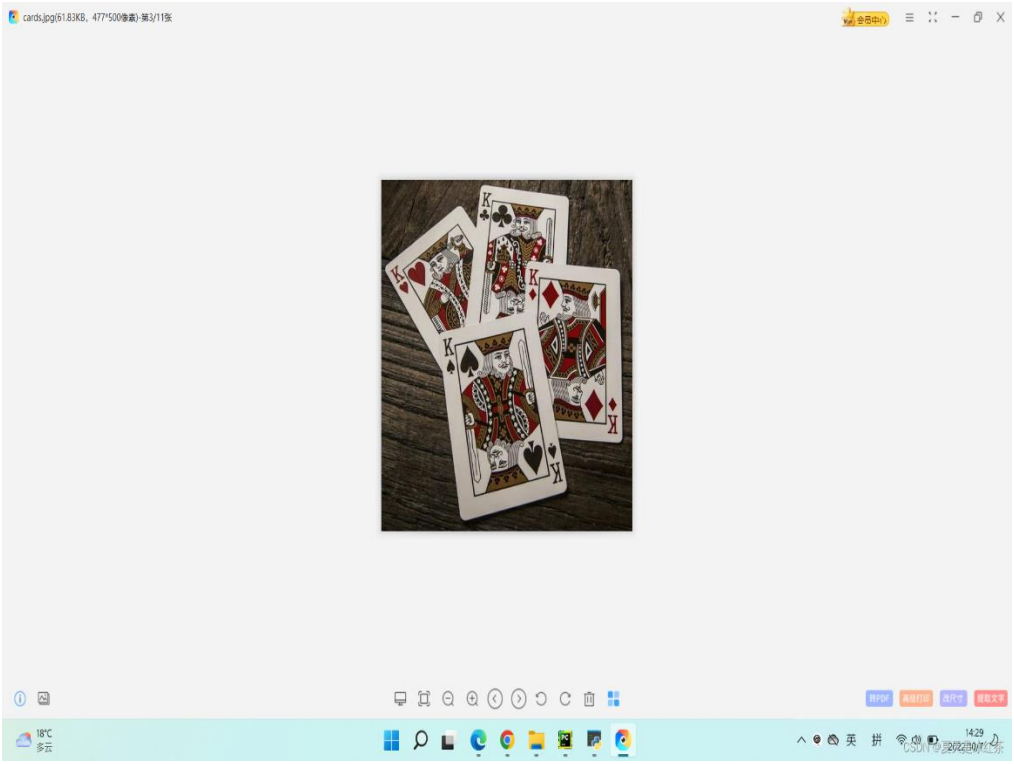
main 文件

```
import cv2
import numpy as np

img = cv2.imread("Resources/cards.jpg")

width,height = 250,350
pts1 = np.float32([[111,219],[287,188],[154,482],[352,440]])
#大概数值可通过 PS 的标尺得到
pts2 = np.float32([[0,0],[width,0],[0,height],[width,height]])
matrix = cv2.getPerspectiveTransform(pts1,pts2)
#透视变换函数，src：源图像中待测矩形的四点坐标，sdt：目标图像中矩形的四点坐标
imgOutput = cv2.warpPerspective(img,matrix,(width,height))
#参数：输入图像，变换矩阵，目标图像 shape

cv2.imshow("Image",img)
cv2.imshow("Output",imgOutput)
cv2.waitKey(0)
```

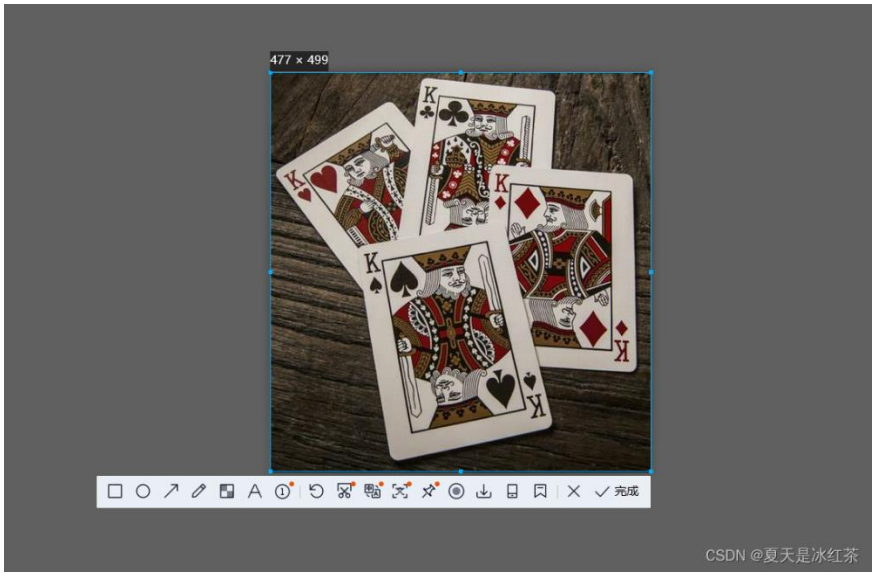
大家一定比较好奇，这个 pts1 是怎么来的，这里来介绍一下具体的方法。

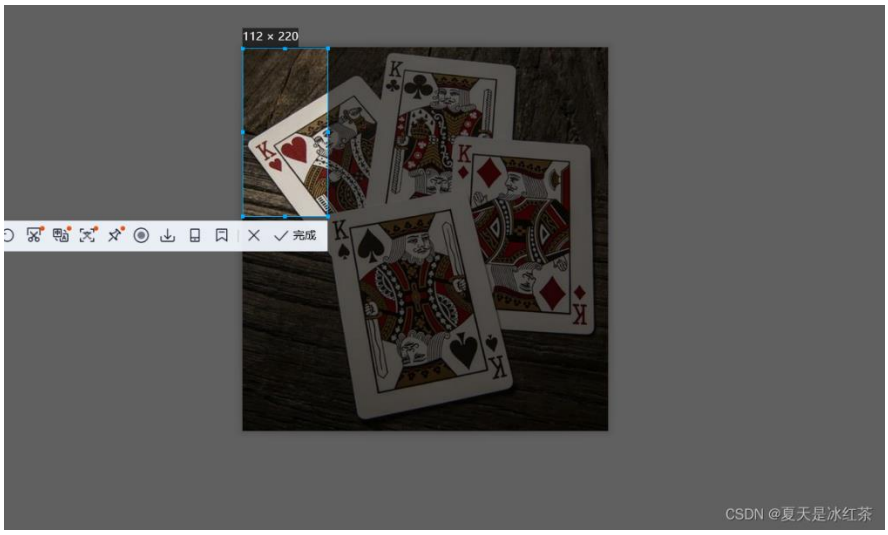
**第一种方法**

采用 QQ 截图，打开我们的图片，并且 QQ 在线。



这样，保持原始大小，不要有任何的放大和缩小。点击 Ctrl+Alt+A，在左上角有数值。这是我用 windows 系统的截图。






请看，数值上是 112*220，填入我们所构成的矩阵。这便是第一个点，按照顺序就可创建这个矩阵。

**第二种方法**

我们使用 PS，直接打开图片，不用置入对象。使用标尺，如下的左边和上边。


从相应的标尺的地方可以拉出一条线，如下，只是我们对应好位置后，需要记下相应的数值。



**第三种方法**

采用轮廓角点提取的方法，此方法在这里不详解，只是提供一个思路。

简单来说就是通过边缘检测算法（推荐使用 Canny 边缘检测）获得图像的边缘。接下来，使用 cv2.findContours()函数找到边缘的轮廓。

对于角点提取，这里使用了 cv2.goodFeaturesToTrack()函数。可以调整 maxCorners 参数来控制检测到的角点数量，minDistance 参数用于指定角点的质量，minDistance 参数指定两个角点之间的最小距离。

最后，通过 cv2.drawContours()函数绘制轮廓，并使用 cv2.circle()函数绘制检测到的角点。

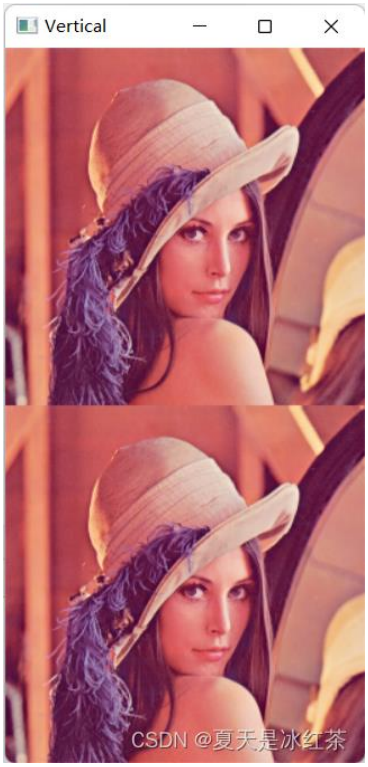
# 六、堆叠多个图像显示

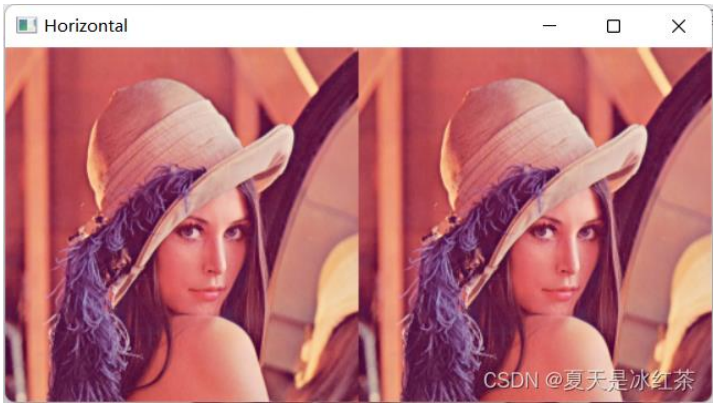
这次我们要实现的效果是让多张图片在一个窗口中显现，使用 matplotlib 也可以实现，但今天我们自己创一个函数来实现这个功能，有的时候，我们在做实时检测时可以用到它，可以与原图进行对比。

先来开个小菜，比如，如果只是实现水平、垂直的合并，非常简单。我们只需要用到 np.hstack 和 np.vstack 来实现。

**水平、垂直显示**

```
1. import cv2
2. import numpy as np
3. img = cv2.imread('Resources/lena.png')
4. img=cv2.resize(img,(0,0),None,0.5,0.5)
5. 
6. imgHor = np.hstack((img,img))
7. imgVer = np.vstack((img,img))
8. # 水平，垂直
9. cv2.imshow("Horizontal",imgHor)
10. cv2.imshow("Vertical",imgVer)
11. cv2.waitKey(0)
```



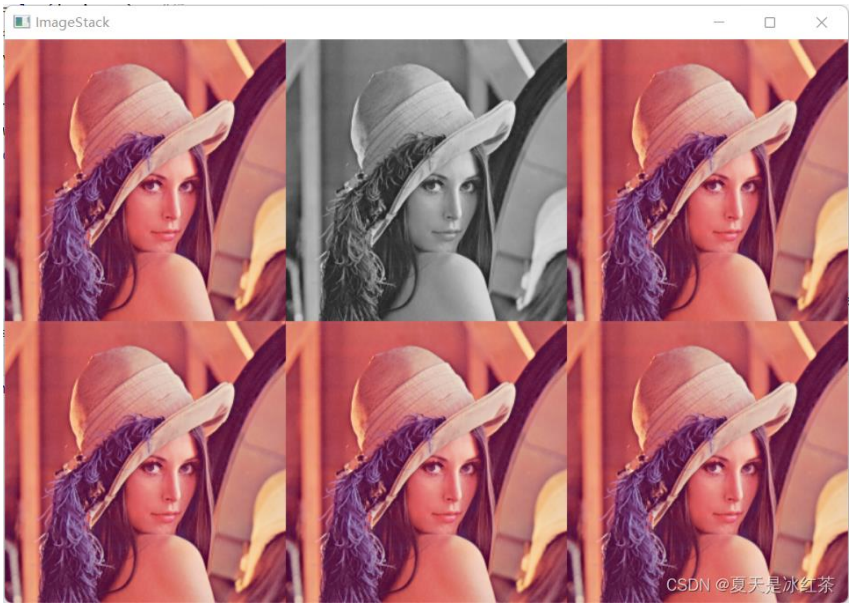



如果只有两个，无可厚非，但我们有时候，要面对四个、六个、八个就捉襟见肘了。我们需要封装一个函数，让他通过传入图像构成的列表，使其能够显示我们的窗口。

```
1.  import cv2
2.  import numpy as np
3.
4.  def stackImages(scale,imgArray):
5.      rows = len(imgArray)   #行
6.      cols = len(imgArray[0])    #列
7.      rowsAvailable = isinstance(imgArray[0], list)
8.      width = imgArray[0][0].shape[1]
9.      height = imgArray[0][0].shape[0]
10.     if rowsAvailable:
11.         for x in range ( 0, rows):
12.             for y in range(0, cols):
13.                 if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
14.                     imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None
, scale, scale)
15.                 else:
16.                     imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0]
[0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)
17.                 if len(imgArray[x][y].shape) == 2:
18.                     imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR_
GRAY2BGR)
19.         imageBlank = np.zeros((height, width, 3), np.uint8)
20.         hor = [imageBlank]*rows
21.         hor_con = [imageBlank]*rows
22.         for x in range(0, rows):
23.             hor[x] = np.hstack(imgArray[x])
24.         ver = np.vstack(hor)
25.     else:
```

```
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)
            if len(imgArray[x].shape) == 2:
                imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)

        hor= np.hstack(imgArray)
        ver = hor
    return ver
img = cv2.imread('Resources/lena.png')
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

imgStack = stackImages(0.5,([img,imgGray,img],[img,img,img]))
# imgHor = np.hstack((img,img))
# imgVer = np.vstack((img,img))
#水平，垂直
# cv2.imshow("Horizontal",imgHor)
# cv2.imshow("Vertical",imgVer)
cv2.imshow("ImageStack",imgStack)
cv2.waitKey(0)
```

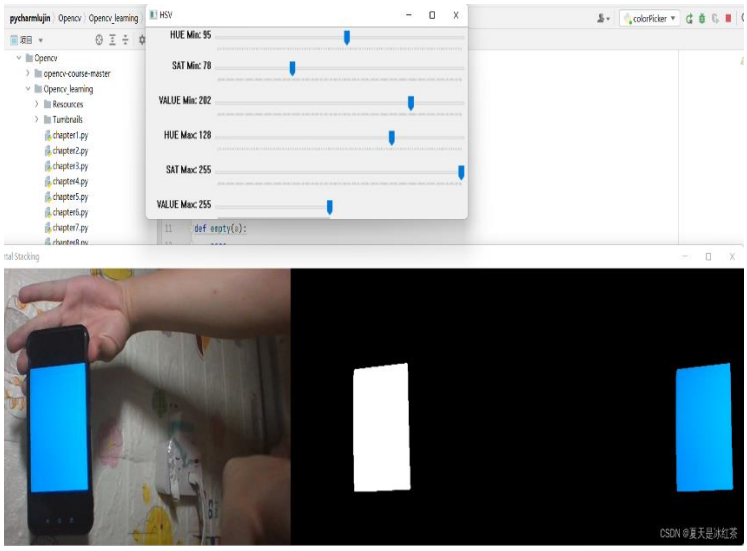
在这里我们创建了一个 stackImages(scale,imgArray)，他用来接收图像的规模大小，以及传入需要合并图像的列表。其中的内容我觉得还是很清晰的，如果你不能明白这里也没有关系，因为你可以将它作为一个函数调用。



# 七、颜色检测和对象提取

### 颜色检测

首先，我们需要实现一个颜色检测的脚本，它的效果如下：




在这个窗口所显示的是实时摄像、白板、显示蒙版，放在一起只是为了对比，可以看到根据我们对上面轨迹栏的调整，获得了 6 个值，让第三个窗口显示出蓝屏。

### 蒙版

蒙版（Mask）在图像处理中是一种表示图像区域的二维数组，其中的元素可以是 0 或 1，通常用于指定感兴趣的区域或特定操作的区域。蒙版可以看作是对图像进行区域选择或遮罩的一种机制。0 表示背景或无效区域，1 表示前景或有效区域。

### 脚本

```
1. import cv2
2. import numpy as np
3.
4. frameWidth = 320
5. frameHeight = 240
6. cap = cv2.VideoCapture(0)
7. cap.set(3, frameWidth)
```

```
cap.set(4, frameHeight)
cap.set(10,150)
def empty(a):
    pass

cv2.namedWindow("HSV")
cv2.resizeWindow("HSV",640,250)
cv2.createTrackbar("HUE Min","HSV",0,179,empty)
cv2.createTrackbar("SAT Min","HSV",0,255,empty)
cv2.createTrackbar("VALUE Min","HSV",0,255,empty)
cv2.createTrackbar("HUE Max","HSV",179,179,empty)
cv2.createTrackbar("SAT Max","HSV",255,255,empty)
cv2.createTrackbar("VALUE Max","HSV",255,255,empty)

while True:
    ret, img = cap.read()
    if ret == False:
        break
    imgHsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

    h_min = cv2.getTrackbarPos("HUE Min","HSV")
    h_max = cv2.getTrackbarPos("HUE Max", "HSV")
    s_min = cv2.getTrackbarPos("SAT Min", "HSV")
    s_max = cv2.getTrackbarPos("SAT Max", "HSV")
    v_min = cv2.getTrackbarPos("VALUE Min", "HSV")
    v_max = cv2.getTrackbarPos("VALUE Max", "HSV")
    print(h_min)

    lower = np.array([h_min,s_min,v_min])
    upper = np.array([h_max,s_max,v_max])
    mask = cv2.inRange(imgHsv,lower,upper)
    result = cv2.bitwise_and(img,img, mask = mask)
    mask = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
    hStack = np.hstack([img,mask,result])
    #cv2.imshow('Original', img)
    #cv2.imshow('HSV Color Space', imgHsv)
    #cv2.imshow('Mask', mask)
    #cv2.imshow('Result', result)
    cv2.imshow('Horizontal Stacking', hStack)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

在这里 cv2.inRange()函数来检查数组元素是否位于其他两个数组的元素之间，cv2.bitwise_and 是位运算，将相同的位置进行与的叠加，就是我们所说的蒙版。

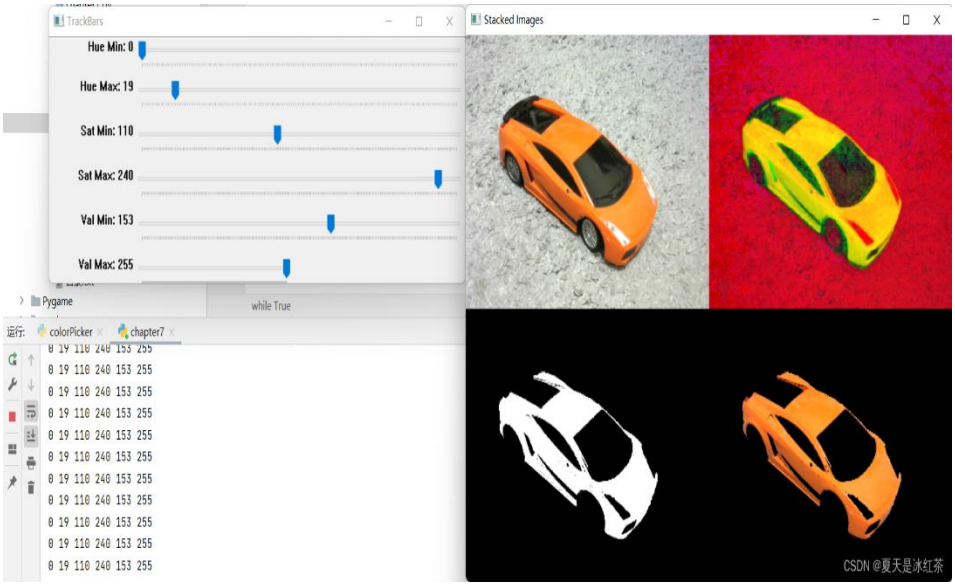
**物体提取**

那么，我们现在将会通过上面的颜色检测，将我们的目标检测出来。

```
1.  import cv2
2.  import numpy as np
3.
4.  def empty(a):
5.      pass
6.
7.  def stackImages(scale,imgArray):
8.      rows = len(imgArray)
9.      cols = len(imgArray[0])
10.     rowsAvailable = isinstance(imgArray[0], list)
11.     width = imgArray[0][0].shape[1]
12.     height = imgArray[0][0].shape[0]
13.     if rowsAvailable:
14.         for x in range ( 0, rows):
15.             for y in range(0, cols):
16.                 if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
17.                     imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None
, scale, scale)
18.                 else:
19.                     imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0]
[0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)
20.                 if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtCo
lor( imgArray[x][y], cv2.COLOR_GRAY2BGR)
21.         imageBlank = np.zeros((height, width, 3), np.uint8)
22.         hor = [imageBlank]*rows
23.         hor_con = [imageBlank]*rows
24.         for x in range(0, rows):
25.             hor[x] = np.hstack(imgArray[x])
26.         ver = np.vstack(hor)
27.     else:
28.         for x in range(0, rows):
29.             if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
30.                 imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, s
cale)
31.             else:
```

```
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)
            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)
        hor= np.hstack(imgArray)
        ver = hor
    return ver
path = 'source/lambo.png'
cv2.namedWindow("TrackBars")  #创建轨迹栏
cv2.resizeWindow("TrackBars",640,240)  #窗口大小调整
cv2.createTrackbar("Hue Min","TrackBars",0,179,empty)
cv2.createTrackbar("Hue Max","TrackBars",19,179,empty)
cv2.createTrackbar("Sat Min","TrackBars",110,255,empty)
cv2.createTrackbar("Sat Max","TrackBars",240,255,empty)
cv2.createTrackbar("Val Min","TrackBars",153,255,empty)
cv2.createTrackbar("Val Max","TrackBars",255,255,empty)
#经过测试得到的掩码 0 19 110 240 153 255

while True:
    img = cv2.imread(path)
    #图像转化为 HSV 格式，H:色调 S:饱和度 V:明度
    imgHSV = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
    #获取轨迹栏位
    h_min = cv2.getTrackbarPos("Hue Min","TrackBars")
    h_max = cv2.getTrackbarPos("Hue Max", "TrackBars")
    s_min = cv2.getTrackbarPos("Sat Min", "TrackBars")
    s_max = cv2.getTrackbarPos("Sat Max", "TrackBars")
    v_min = cv2.getTrackbarPos("Val Min", "TrackBars")
    v_max = cv2.getTrackbarPos("Val Max", "TrackBars")
    print(h_min,h_max,s_min,s_max,v_min,v_max)
    #创建一个蒙版，提取需要的颜色为白色，不需要的颜色为白色
    lower = np.array([h_min,s_min,v_min])
    upper = np.array([h_max,s_max,v_max])
    mask = cv2.inRange(imgHSV,lower,upper)
    imgResult = cv2.bitwise_and(img,img,mask=mask)
    #使用的图像，新图，应用蒙版，and 操作，颜色叠加得到橙色
    # cv2.imshow("Original",img)
    # cv2.imshow("HSV",imgHSV)
    # cv2.imshow("Mask", mask)
    # cv2.imshow("Result", imgResult)
    imgStack = stackImages(0.6,([img,imgHSV],[mask,imgResult]))
    #定义比例尺
    cv2.imshow("Stacked Images", imgStack)
    cv2.waitKey(1)
```

还记得上一节吗？我们也是用到了这个 stackImages，我们将用它来展示我们的效果。

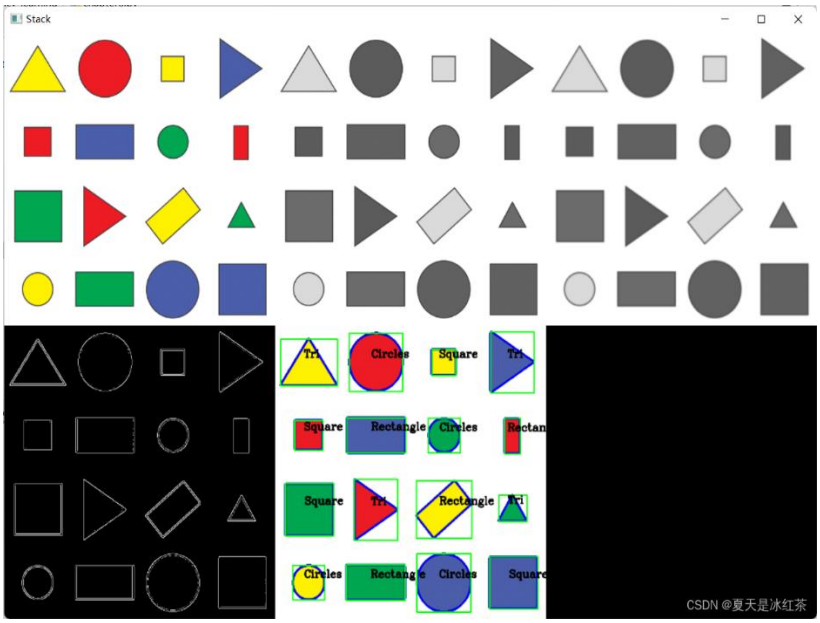



先经过上面的颜色检测后，我们就可以完成对某个图像的提取。

在这里面，蒙版起到了很重要的作用，如果大家学过 PS、PR 的话，对于这个是什么意思，理解起来并不难。

# 八、轮廓和形状检测

**效果展示**



在这里，我们对于原图像中的轮廓和形状进行了检测。分别含有三角形、圆形、正方形、矩形等。且根据图像的大小添加了矩形框。

main 文件

```
import cv2
import numpy as np

def stackImages(scale,imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range ( 0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)
                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        hor_con = [imageBlank]*rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
        ver = np.vstack(hor)
    else:
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)
            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)
        hor= np.hstack(imgArray)
        ver = hor
    return ver
```

```
34.
35. #获取轮廓函数
36. def getContours(img):
37.     contours,hierarchy = cv2.findContours(img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
38.     #contours 轮廓，hierarchy 次等级，cv2.findContours 查找轮廓函数，第二个参数为检索方法，cv2.RETR_EXTERNAL 检索极端的外部轮廓
39.     for cnt in contours:
40.         area = cv2.contourArea(cnt)  #轮廓区域
41.         print(area)
42.         if area>500:
43.             cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 3)  #绘制轮廓函数
44.             #参数：原始图像，轮廓，轮廓索引=-1，即绘制所有的轮廓
45.             peri = cv2.arcLength(cnt,True)
46.             #曲线长度，找到轮廓的弧长
47.             #print(peri)
48.             #逼近角点
49.             approx = cv2.approxPolyDP(cnt,0.02*peri,True)
50.             # 它是原始轮廓周长的百分比，用于控制逼近的程度。较小的值会产生更接近原始轮廓形状的多边形，而较大的值会产生更简化的多边形。
51.             print(len(approx))
52.             #多变型拟合后点的个数
53.             objCor = len(approx)
54.             #边界框边界矩形
55.             x, y, w, h = cv2.boundingRect(approx)
56.
57.             if objCor ==3: objectType ="Tri"
58.             elif objCor == 4:
59.                 aspRatio = w/float(h)
60.                 #纵横比判断正方形还是长方形
61.                 if aspRatio >0.98 and aspRatio <1.03:
62.                     objectType= "Square"
63.                 else:
64.                     objectType="Rectangle"
65.             elif objCor>4: objectType= "Circles"
66.             else:objectType="None"
67.             #绘制外框
68.             cv2.rectangle(imgContour,(x,y),(x+w,y+h),(0,255,0),2)
69.
70.             cv2.putText(imgContour,objectType,
71.                         (x+(w//2)-10,y+(h//2)-10),cv2.FONT_HERSHEY_COMPLEX,0.7,
72.                         (0,0,0),2)
```

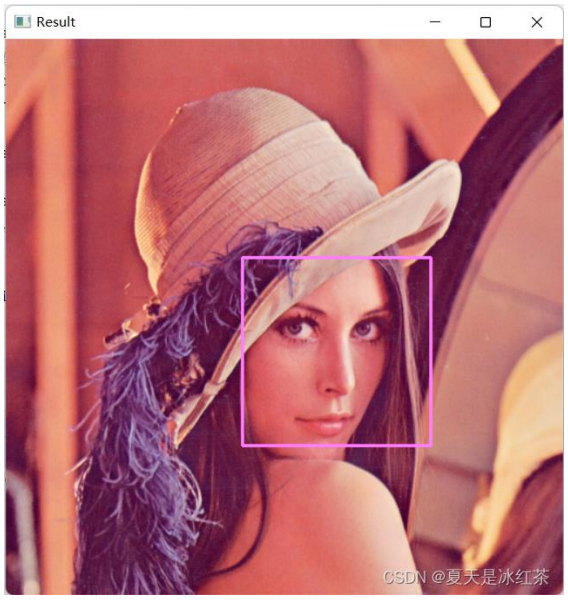
```
            #0.7–大小，2–厚度


path = 'source/shapes.png'
img = cv2.imread(path)
imgContour = img.copy()  #原始图像副本

imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray,(7,7),1)
#参数，灰度图像，内核，sigma=1,越高越模糊
imgCanny = cv2.Canny(imgBlur,50,50)
#边缘检测
getContours(imgCanny)

imgBlank = np.zeros_like(img)  #黑色图像
imgStack = stackImages(0.6,([img,imgGray,imgBlur],
                            [imgCanny,imgContour,imgBlank]))

cv2.imshow("Stack", imgStack)

cv2.waitKey(0)
```

就像我之前所说的，我们只需要调用 stackImages，接下来需要写一个 getContours()的函数，以便于我们寻找它的轮廓，然后根据我们的一拟合点，判断它们的形状是三角形、正方形等，其中的一些函数可能第一次接触，我就尽可能的将注释添上。在这之后，我们就可以根据原图像的一系列转换——灰度转换、高斯模糊、Canny 检测。

# 九、人脸识别与车牌检测（了解）

**人脸识别**

我们先来看看它的效果：




成功检测到人脸，如果你看过我前面的项目，你会非常容易地将它修改成实时地检测人脸。这里我们用到了"haarcascade_frontalface_default.xml"，这是一个入门的教程，暂时不用了解它是怎么来的，它就是一个默认的人脸检测器。

```
1. import cv2
2.
3. faceCascade= cv2.CascadeClassifier("source/haarcascade_frontalface_default.xml")
4.
5. img = cv2.imread('Resources/lena.png')
6. imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
7.
8. faces = faceCascade.detectMultiScale(imgGray,1.1,4)
9. for (x,y,w,h) in faces:
10.     cv2.rectangle(img,(x,y),(x+w,y+h),(255,125,255),2)
11.
12. cv2.imshow("Result", img)
13. cv2.waitKey(0)
```

CascadeClassifier，是 Opencv 中做人脸检测的时候的一个级联分类器，detectMultiScale 用来检测到的对象将作为列表返回个矩形，再简单画框就完成了。

车牌检测


```
import cv2

#############################################
frameWidth = 640
frameHeight = 480
nPlateCascade = cv2.CascadeClassifier("source/haarcascade_russian_plate_number.xml")
minArea = 200
color = (255,0,255)
#############################################
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10,150)
count = 0

while True:
    success, img = cap.read()
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    numberPlates = nPlateCascade.detectMultiScale(imgGray, 1.1, 10)
    for (x, y, w, h) in numberPlates:
        area = w*h
        if area >minArea:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255), 2)
            cv2.putText(img,"Number Plate",(x,y-5),
                        cv2.FONT_HERSHEY_COMPLEX_SMALL,1,color,2)
            imgRoi = img[y:y+h,x:x+w]
```

```
            cv2.imshow("ROI", imgRoi)

    cv2.imshow("Result", img)

    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

    if cv2.waitKey(1) & 0xFF == ord('s'):
        cv2.imwrite("Resources/Scanned/NoPlate_"+str(count)+".jpg",imgRoi)
        cv2.rectangle(img,(0,200),(640,300),(0,255,0),cv2.FILLED)
        cv2.putText(img,"Scan Saved",(150,265),cv2.FONT_HERSHEY_DUPLEX,
                    2,(0,0,255),2)
        cv2.imshow("Result",img)
        cv2.waitKey(500)
        count += 1
```

车牌检测与上面的人脸识别相同。只是在最后添加了保存的功能。

# 总结

通过本教程，应该对 Python 编程和 Opencv 的基础知识有了一定的了解。这些基础知识将为您后续的学习和开发提供坚实的基础。在学习过程中，不仅要掌握语法规则，还要注重实践和不断的练习，这样才能更好地运用 Python 来解决问题和开发应用。

本次的教程就结束了，今后的学习中，如果遇到不会的问题，可以再去查找，在遇到问题时学会新的知识。

# 附件

Opencv 官方文档中文：

OpenCV 中文官方文档

Python 自学网站：

Python 教程 (w3schools.cn)

Python3 教程 | 菜鸟教程 (runoob.com)

Python 简介 - 廖雪峰的官方网站 (liaoxuefeng.com)

GitHub:

Opencv-project-training/Opencv_beginner/RADEME.md at main · Auorui/Opencv-project-training · GitHub