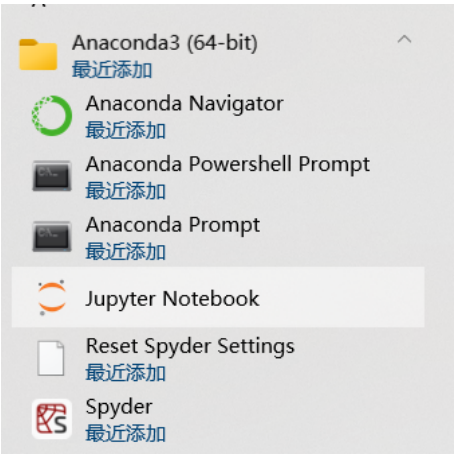
# OpenCV Summer Learning

## #download anaconda

## 显示图片

*1.在windows光标里先打开upyter Notebook*




*2.跳到浏览器里面输入代码*

显示图片：

```
//这四行固定
import cv2
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
//imag是自定义的图片名字，引号内要加图片的路径`
imag=cv2.imread("D:/1.png")
cv2.imshow('1',imag)

//这个是让图片一直显示
cv2.waitKey(0)

//按任意键退出
cv2.destroyAllWindows()

//或者放在一个函数里：
def cv_show(name,img):
        cv2.imshow(name,img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

//最后调用这个函数
imag.imshow
```

# 绘图

## 画直线

```
cv2.line(img,(0,0),(150,150),(255,255,255),15)//参数：图片，开始坐标，结束坐标，颜色（bgr蓝绿红），线条粗细
cv2.imshow('image',imag)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 画矩形

```
cv2.rectangle(img,(15,25),(200,150),(0,0,255),15)//参数：图像，左上角坐标，右下角坐标，颜色和线条粗细
```

## 画圆

```
cv2.circle(img,(100,63), 55, (0,255,0), -1)//参数:图像/帧，圆心，半径，颜色和。 注意我们粗细为-1。 这意味着将填充对象，所以我们会得到一个圆。
```

## 画多边形

```
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)//四个点的坐标，此例为四边形
cv2.polylines(imag, [pts], True, (0,255,255), 3)//参数：绘制的对象，坐标，我们应该连接终止的和起始点，颜色和粗细
```

## 在图片上写字

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(imag,'OpenCV Tuts!',(0,130), font, 1, (200,255,155), 2, cv2.LINE_AA)//参数：绘制的对象，要绘制的文字，文本左下角的坐标，字体类型，文本大小的比例因子（乘以字体类型的默认大小就是显示文字的大小），颜色，粗细，绘制文本的线条类型（这里是抗锯齿线）
```

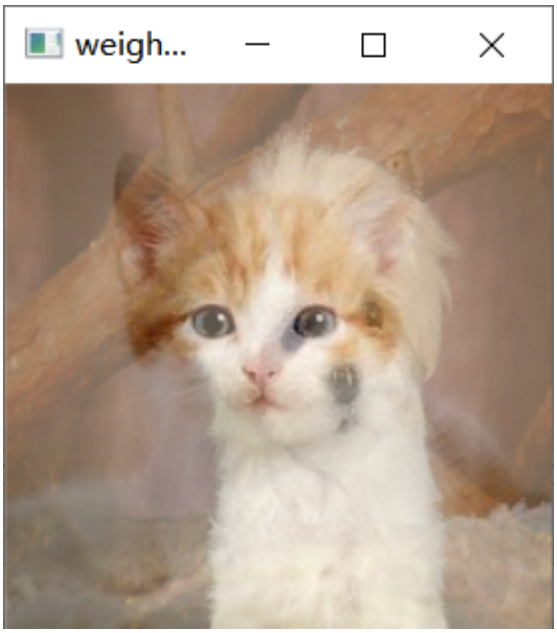
## *完整代码*

```
import numpy as np
import cv2

img = cv2.imread('watch.jpg',cv2.IMREAD_COLOR)
cv2.line(img,(0,0),(200,300),(255,255,255),50)
cv2.rectangle(img,(500,250),(1000,500),(0,0,255),15)
cv2.circle(img,(447,63), 63, (0,255,0), -1)
pts = np.array([[100,50],[200,300],[700,200],[500,100]], np.int32)
```

```
pts = pts.reshape((-1,1,2))
cv2.polylines(img, [pts], True, (0,255,255), 3)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV Tuts!',(10,500), font, 6, (200,255,155), 13, cv2.LINE_AA)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# 两张图片相加合成

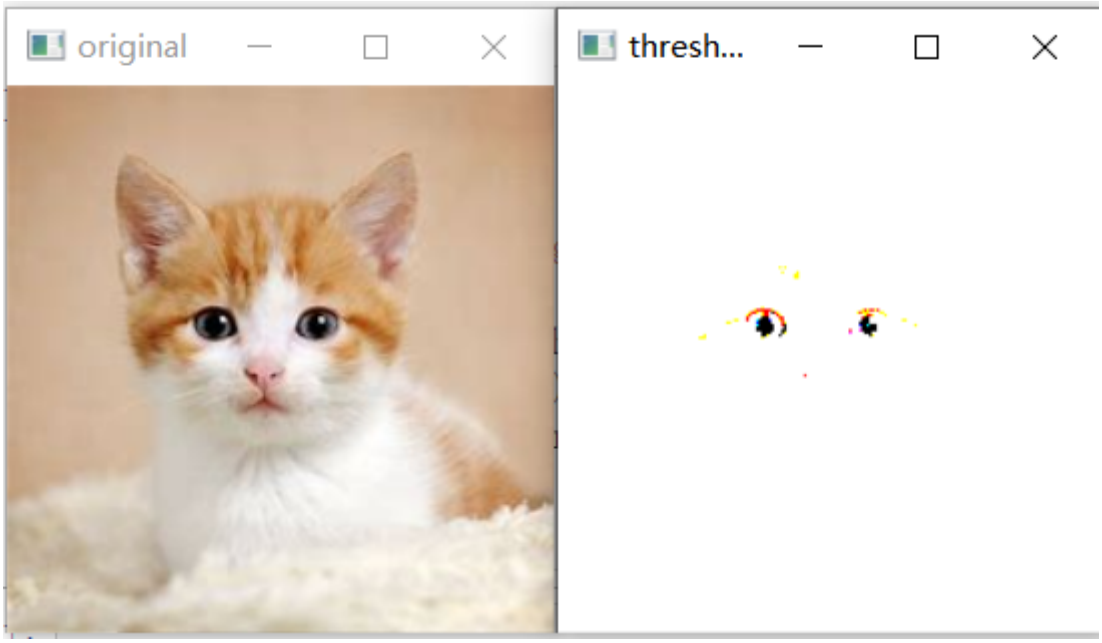
```
import cv2
import numpy as np

img1 = cv2.imread('3D-Matplotlib.png')
img2 = cv2.imread('mainsvmimage.png')

weighted = cv2.addWeighted(img1, 0.6, img2, 0.4, 0)//参数：第一个图像，权重，第二个图像，权重，然后是伽马值，这是一个光的测量值。 我们现在就把它保留为零
cv2.imshow('weighted',weighted)
cv2.waitKey(0)
cv2.destroyAllWindows()
import cv2
import numpy as np
//此时，会发现，这个代码会报错Sizes of input arguments do not match。是因为，这两张图片像素尺寸不匹配，需要自行改变两张图片的像素尺寸使其一致。

## 完整代码

img1 = cv2.imread('D:/1.png')
img2 = cv2.imread('D:/2.png')
target_size=(218,218)//目标尺寸
resized_img1 = cv2.resize(img1.astype(np.uint8), target_size)
target_size1=(218,218)//目标尺寸
resized_img2 = cv2.resize(img2.astype(np.uint8), target_size1)
weighted = cv2.addWeighted(resized_img1, 0.6, resized_img2, 0.4, 0)
cv2.imshow('weighted',weighted)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




cute huh..

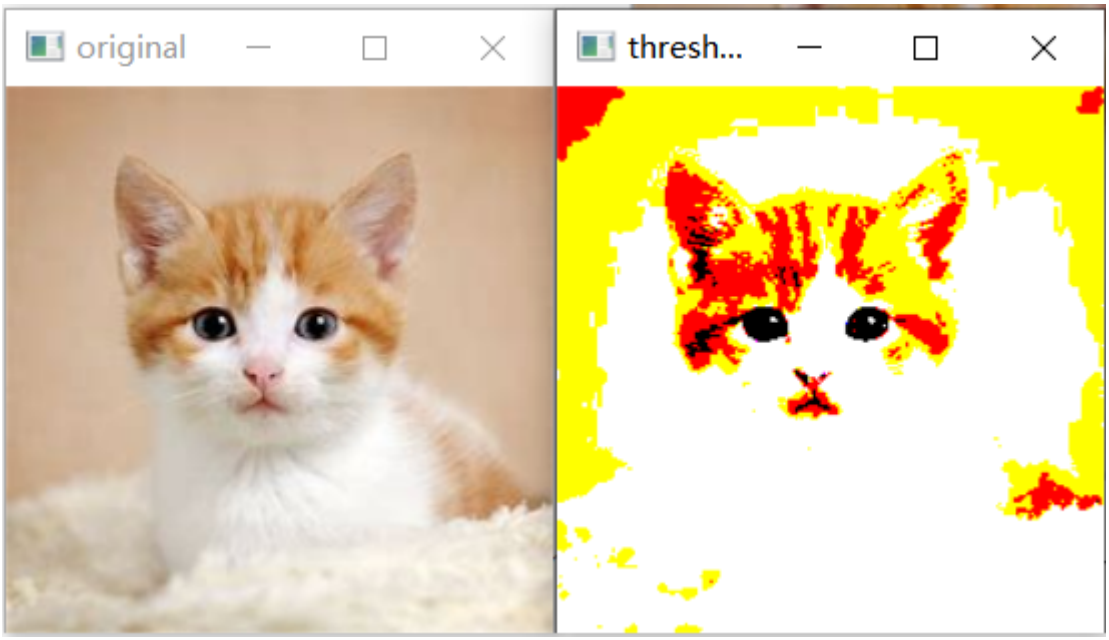
## 阈值

```
import cv2
import numpy as np

img = cv2.imread('D:/1.png')
grayscaled = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
retval, threshold = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)//参数:图像, 阈值，最大值，阈值类型。 通常情况下，10 的阈值会有点差。 我们选择 12，低光照的图片，选择低的数字。 通常 125-150 左右的东西可能效果最好。
cv2.imshow('original',img)
cv2.imshow('threshold',threshold)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



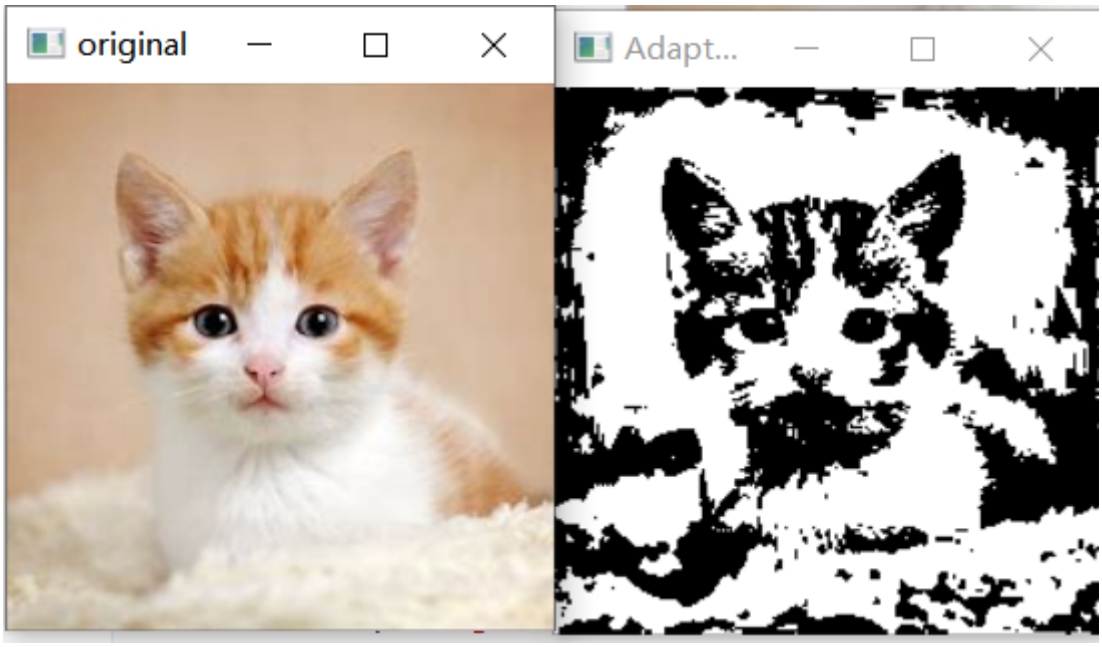

retval, threshold = cv2.threshold(img, 150, 255, cv2.THRESH_BINARY)

把阈值改为150可得



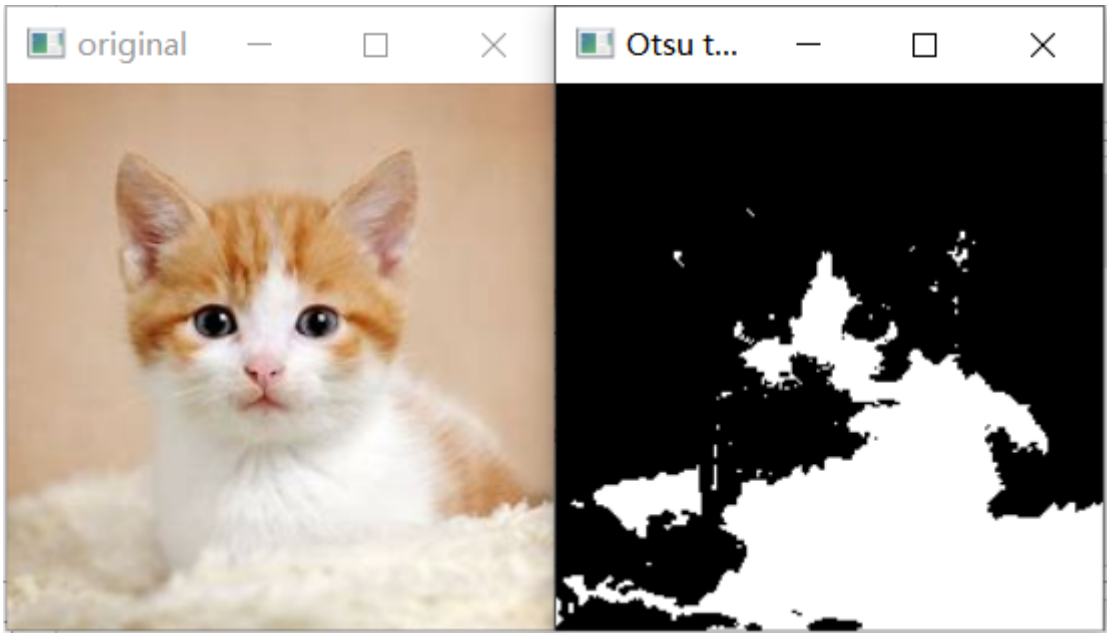

## 自适应阈值

```
import cv2
import numpy as np

img = cv2.imread('D:/1.png')
th = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 115, 1)
cv2.imshow('original',img)
cv2.imshow('Adaptive threshold',th)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




## 大津阈值

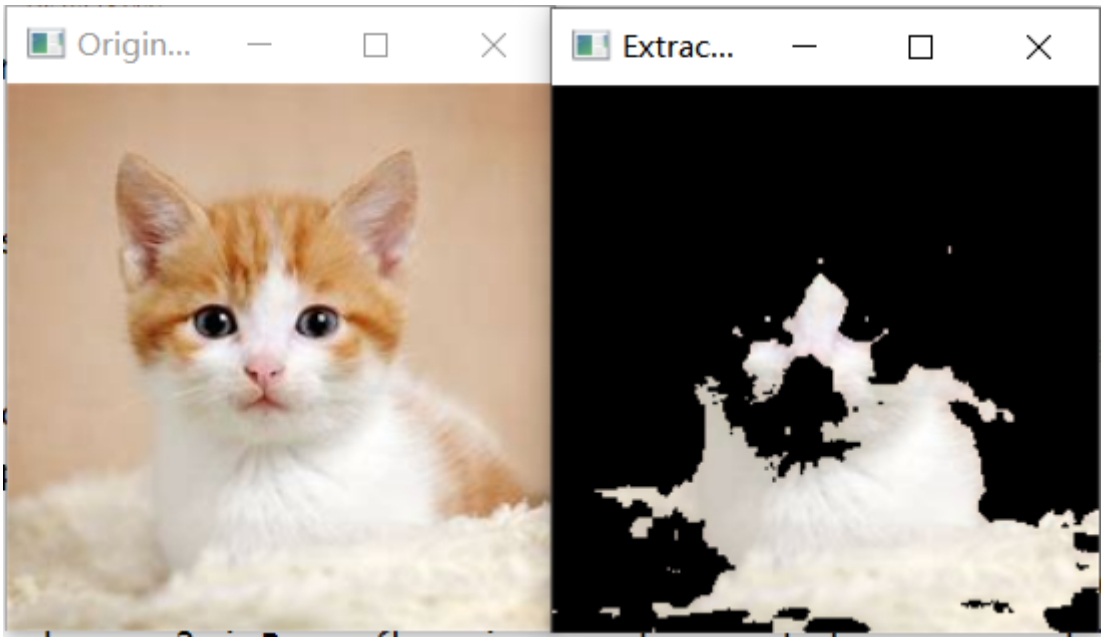
```
import cv2
import numpy as np
img = cv2.imread('D:/1.png')
retval2,threshold2 =
cv2.threshold(grayscaled,125,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow('original',img)
cv2.imshow('Otsu threshold',threshold2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




## 颜色过滤

当BGR 转换为 HSV，想要得到三个数值，这里推荐https://lab.pyzy.net/palette.html

```python
import cv2
import numpy as np

# 读取图像
image = cv2.imread("D:/1.png")

# 将图像从BGR颜色空间转换为HSV颜色空间
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# 定义所需颜色的范围 这里过滤的是白色，除了白色以外的部分显示黑色
lower_white = np.array([0, 0, 200])
upper_white = np.array([180, 30, 255])

# 创建掩膜，将在指定颜色范围内的像素设置为255，其余像素设置为0
mask = cv2.inRange(hsv_image, lower_white, upper_white)

# 应用掩膜到原始图像，提取特定颜色范围的像素
result = cv2.bitwise_and(image, image, mask=mask)

# 显示原始图像和提取的结果图像
cv2.imshow("Original Image", image)
cv2.imshow("Extracted Result", result)
cv2.waitKey(0)
cv2.destroyAllWindows()

```




```python
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while(1):
```

```
    _, frame = cap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_gray = np.array([0, 0, 31])
    upper_gray = np.array([180, 30, 230])

    mask = cv2.inRange(hsv, lower_gray, upper_gray)
    res = cv2.bitwise_and(frame,frame, mask= mask)

    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)

    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

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

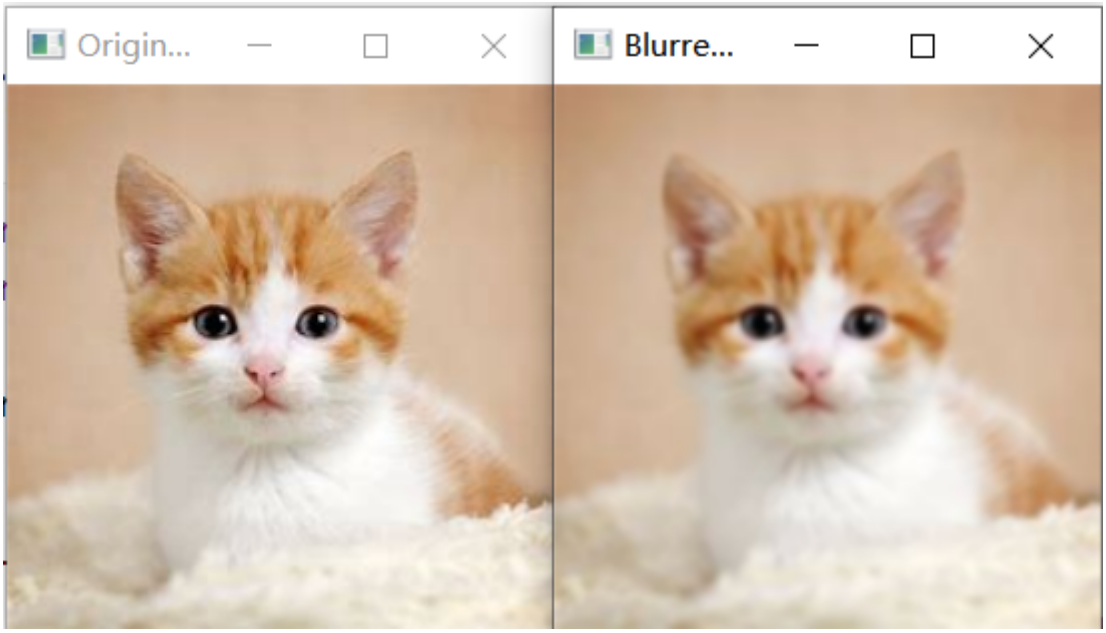
# 模糊和平滑

## 简单平滑

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while(1):

    _, frame = cap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_gray = np.array([0, 0, 31])
    upper_gray = np.array([180, 30, 230])

    mask = cv2.inRange(hsv, lower_gray, upper_gray)
    res = cv2.bitwise_and(frame,frame, mask= mask)
    kernel = np.ones((15,15),np.float32)/225 //我们使用15x15正方形，这意味着我们有225 个总像素
    smoothed = cv2.filter2D(res,-1,kernel)
    cv2.imshow('Original',frame)
    cv2.imshow('Averaging',smoothed)

    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

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

## 高斯模糊

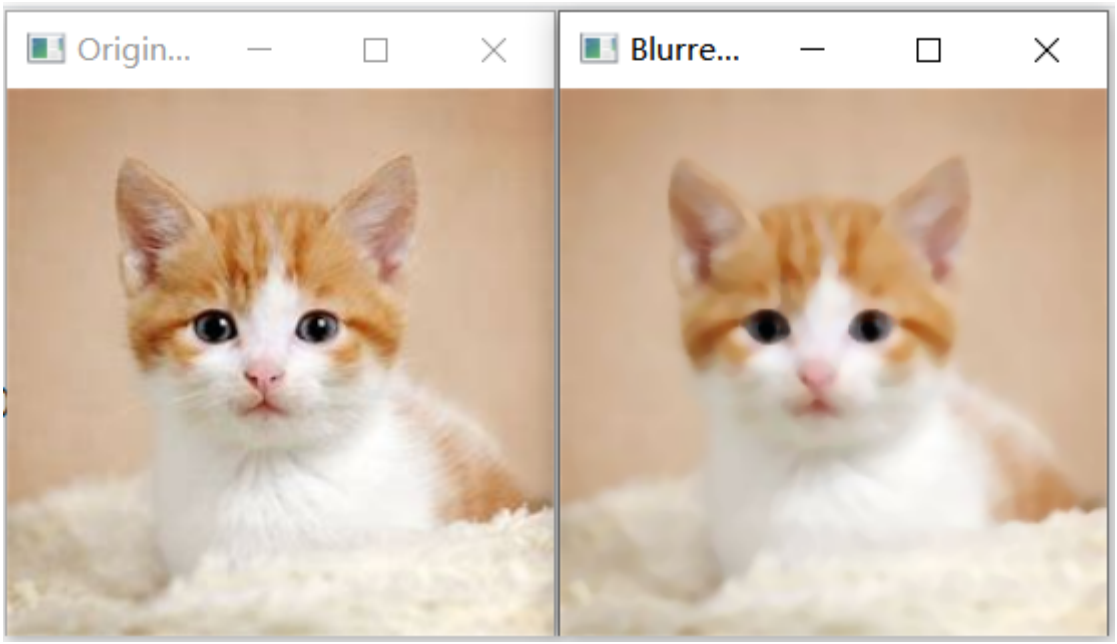
```python
import cv2
import numpy as np

image = cv2.imread("D:/1.png")

# 使用高斯滤波平滑图像
blurred_image = cv2.GaussianBlur(image, (5, 5), 0)
# 这里的(5, 5)是卷积核的大小，可以根据需要进行调整。第三个参数0表示标准差，可以根据需要进行调整。

# 显示原始图像和平滑后的图像
cv2.imshow("Original Image", image)
cv2.imshow("Blurred Image", blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




## 中值模糊

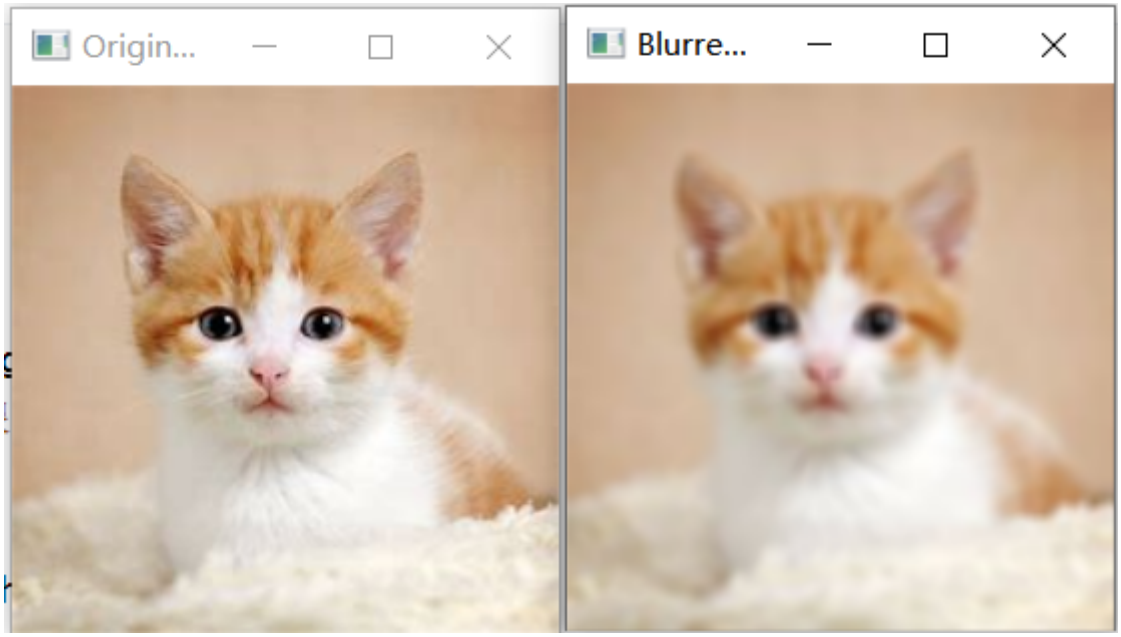
```python
import cv2
import numpy as np

image = cv2.imread("D:/1.png")

# 使用中值滤波平滑图像
blurred_image = cv2.medianBlur(image, 5)
# 这里的5是卷积核的大小，可以根据需要进行调整。

# 显示原始图像和平滑后的图像
cv2.imshow("Original Image", image)
cv2.imshow("Blurred Image", blurred_image)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```




## 双向模糊

```python
import cv2

image = cv2.imread("D:/1.png")

# 水平方向的模糊
blurred_image_horizontal = cv2.blur(image, (5, 1))
# 这里的(5, 1)表示水平方向的卷积核大小，可以根据需要进行调整。

# 垂直方向的模糊
blurred_image = cv2.blur(blurred_image_horizontal, (1, 5))
# 这里的(1, 5)表示垂直方向的卷积核大小，可以根据需要进行调整。

# 显示原始图像和双向模糊后的图像
cv2.imshow("Original Image", image)
cv2.imshow("Blurred Image", blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




*这几个模糊具体选择得看各自所需要的效果而定!!!*

# 形态变换

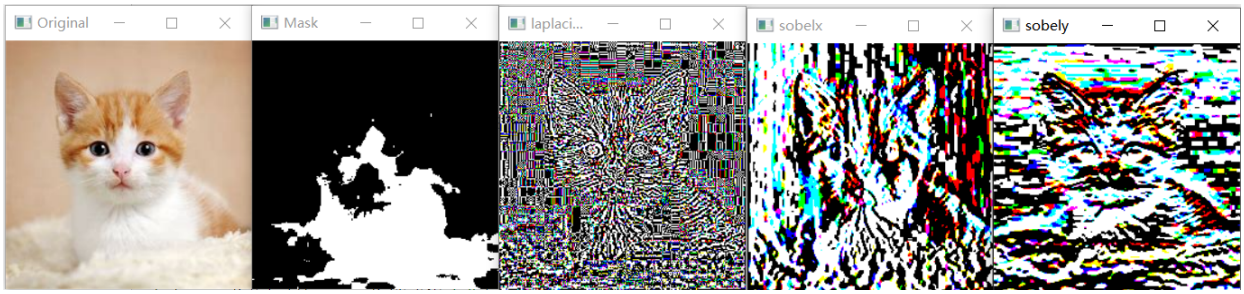
*sorry I cannot find the difference..*

```python
import cv2
import numpy as np

nocap = cv2.VideoCapture(0)

while(1):

    _, frame = nocap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_gray = np.array([0, 0, 31])
    upper_gray = np.array([180, 30, 230])

    mask = cv2.inRange(hsv, lower_gray, upper_gray)
    res = cv2.bitwise_and(frame,frame, mask= mask)

    kernel = np.ones((5,5),np.uint8)
    erosion = cv2.erode(mask,kernel,iterations = 1)
    dilation = cv2.dilate(mask,kernel,iterations = 1)

    cv2.imshow('Original',frame)
    cv2.imshow('Mask',mask)
    cv2.imshow('Erosion',erosion)
    cv2.imshow('Dilation',dilation)

    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 边缘检测和渐变

```
import cv2
import numpy as np

# 读取图像
frame = cv2.imread('D:/1.png')
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

#提取白色
lower_white = np.array([0, 0, 200])#
upper_white = np.array([180, 30, 255])#

mask = cv2.inRange(hsv, lower_white, upper_white)#
res = cv2.bitwise_and(frame,frame, mask= mask)#
#打#这几行不要也可以执行

laplacian = cv2.Laplacian(frame,cv2.CV_64F)
sobelx = cv2.Sobel(frame,cv2.CV_64F,1,0,ksize=5)#cv2.CV_64F是数据类型，ksize是核大小， 我们使用 5，所以每次查询5x5的渔区
sobely = cv2.Sobel(frame,cv2.CV_64F,0,1,ksize=5)

cv2.imshow('Original',frame)
cv2.imshow('Mask',mask)
cv2.imshow('laplacian',laplacian)
cv2.imshow('sobelx',sobelx)
cv2.imshow('sobely',sobely)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



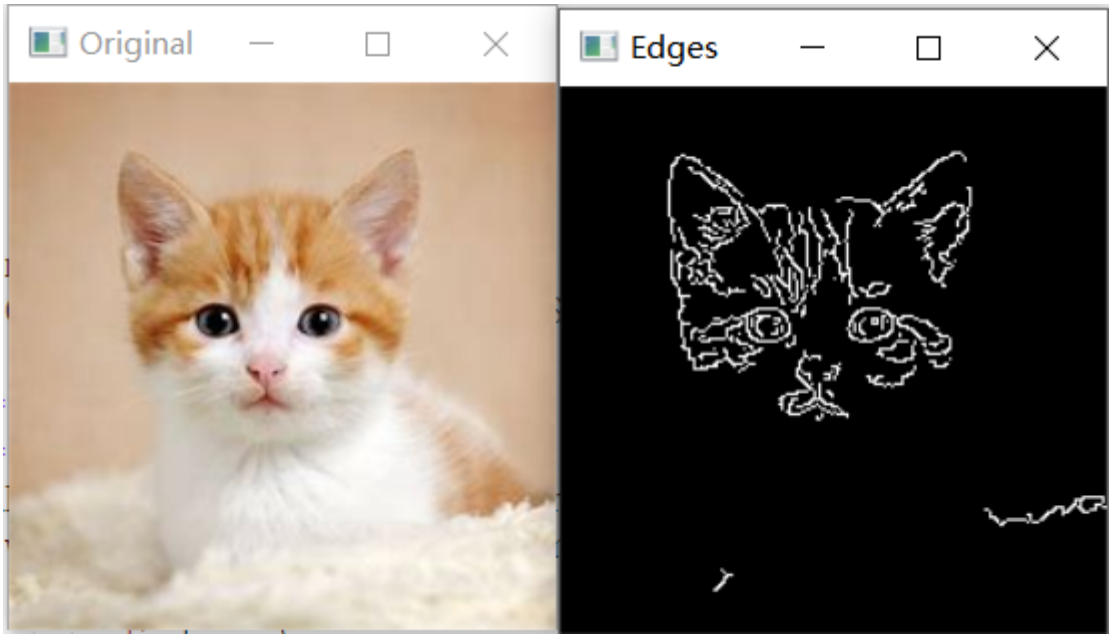

## 不过滤颜色的代码

```
import cv2
import numpy as np

# 读取图像
frame = cv2.imread('D:/1.png')
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

```
7
8  # lower_white = np.array([0, 0, 200])
9  # upper_white = np.array([180, 30, 255])
10 # mask = cv2.inRange(hsv, lower_white, upper_white)
11 # res = cv2.bitwise_and(frame,frame, mask= mask)
12
13 laplacian = cv2.Laplacian(frame,cv2.CV_64F)#读取原图片，所以laplacian和sobelx、
   sobely和上面图片保持一致
14 sobelx = cv2.Sobel(frame,cv2.CV_64F,1,0,ksize=5)
15 sobely = cv2.Sobel(frame,cv2.CV_64F,0,1,ksize=5)
16
17 cv2.imshow('Original',frame)
18 # cv2.imshow('Mask',mask)
19 cv2.imshow('laplacian',laplacian)
20 cv2.imshow('sobelx',sobelx)
21 cv2.imshow('sobely',sobely)
22
23 cv2.waitKey(0)
24 cv2.destroyAllWindows()
```


## Canny 边缘检测

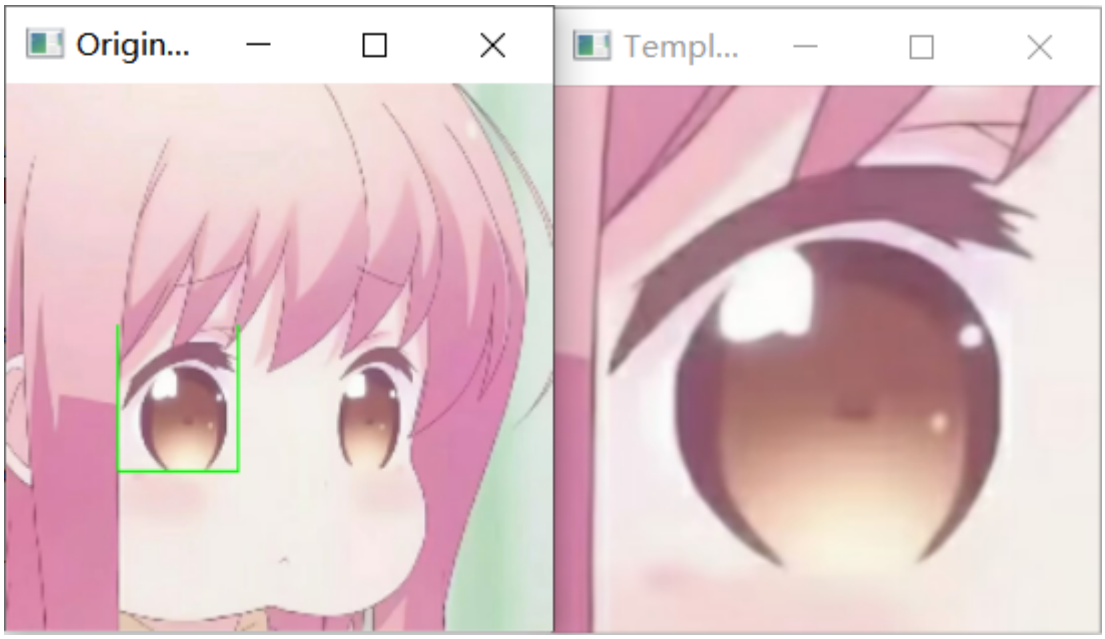
```
1  import cv2
2  import numpy as np
3
4
5  # 读取图像
6  frame = cv2.imread('D:/1.png')
7  hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
8
9  lower_white = np.array([0, 0, 200])
10 upper_white = np.array([180, 30, 255])
11 mask = cv2.inRange(hsv, lower_white, upper_white)
12 res = cv2.bitwise_and(frame,frame, mask= mask)
13
14 cv2.imshow('Original',frame)
15 edges = cv2.Canny(frame,100,200)
16 cv2.imshow('Edges',edges)
17
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```




## 模板匹配

```python
import cv2
import numpy as np

# 读取图像和模板
image = cv2.imread("D:/77.png")
template = cv2.imread("D:/88.png")

# 将图像和模板转换为灰度图像
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray_template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

# 使用模板匹配算法
result = cv2.matchTemplate(gray_image, gray_template, cv2.TM_CCOEFF_NORMED)

# 获取匹配结果的坐标
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
top_left = max_loc
bottom_right = (top_left[0] + template.shape[1], top_left[1] +
template.shape[0])

# 在原始图像上绘制矩形框
cv2.rectangle(image, top_left, bottom_right, (0, 255, 0), 2)

#这里原始图片太大了，我把图片缩小了一些
target_size1=(218,218)
resized_image = cv2.resize(image.astype(np.uint8), target_size1)
target_size2=(218,218)
resized_template = cv2.resize(template.astype(np.uint8), target_size2)
```

```
28
29  # 显示原始图像和匹配结果
30  cv2.imshow("Original Image", resized_image)
31  cv2.imshow("Template Matching Result", resized_template)
32  cv2.waitKey(0)
33  cv2.destroyAllWindows()
```




# GrabCut 前景提取

就是找到前景，并删除背景。想要提取的部分之外全是黑的。

`rect = (start_x, start_y, width, height)`

这是包围我们的主要对象的矩形。要找到适合你的图像的坐标。这很难以用眼睛找，那么我们可以采取交互的办法

```
1  import cv2
2
3  # 读取图像
4  img = cv2.imread('D:/77.png')
5
6  # 选择感兴趣区域
7  rect = cv2.selectROI(img)
8
9  # 打印矩形坐标
10 print(rect)
```

这是我选中的部分

```
import cv2

# 读取图像
img = cv2.imread('D:/77.png')

# 选择感兴趣区域
rect = cv2.selectROI(img)

# 打印矩形坐标
print(rect)
```

```
(217, 394, 294, 365)
```

记下（217，394，294，365）这个坐标

## 完整代码

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

# 读取图像
img = cv2.imread('D:/77.png')

# 创建空白掩模
mask = np.zeros(img.shape[:2], np.uint8)

# 初始化背景模型和前景模型
bgdModel = np.zeros((1, 65), np.float64)
fgdModel = np.zeros((1, 65), np.float64)

# 定义感兴趣区域的矩形坐标
rect = (217, 394, 294, 365)  # 根据实际情况设置矩形坐标

# 执行GrabCut算法
cv2.grabCut(img, mask, rect, bgdModel, fgdModel, 5, cv2.GC_INIT_WITH_RECT)

# 生成二值掩模
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

# 提取前景部分
img = img * mask2[:, :, np.newaxis]

# 显示前景图像
plt.imshow(img)
plt.colorbar()
plt.show()
```

结果会显示在jupyter notebook上

# 角点检测

检测角点的目的是追踪运动，做 3D 建模，识别物体，形状和角色等。

## Harris角点检测

```python
import cv2
import numpy as np

# 读取图像
img = cv2.imread('D:/77.png')

#改图片大小
target_size1=(500,500)
resized_image = cv2.resize(img.astype(np.uint8), target_size1)

# 将图像转换为灰度图像
gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)

# 执行Harris角点检测
dst = cv2.cornerHarris(gray, blockSize=2, ksize=3, k=0.04)

# 对角点进行阈值处理
dst = cv2.dilate(dst, None)
resized_image[dst > 0.01 * dst.max()] = [0, 0, 255]  # 将角点标记为红色

# 显示结果图像
cv2.imshow('Corner Detection', resized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# 特征匹配（单映射）爆破

我们首先使用SIFT算法检测和计算图像中的特征点和描述子。然后，我们使用暴力匹配（brute-force matching）的方法，使用KNN（k-nearest neighbors）算法进行特征匹配。通过比较特征点之间的距离，我们筛选出最佳的匹配点。最后，我们使用OpenCV的`drawMatches`函数将匹配结果可视化。

```python
import cv2

# 读取图像
img1 = cv2.imread('D:/12.jpg', cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread('D:/21.jpg', cv2.IMREAD_GRAYSCALE)

target_size1=(300,300)
resized_img1 = cv2.resize(img1.astype(np.uint8), target_size1)
target_size2=(500,500)
resized_img2 = cv2.resize(img2.astype(np.uint8), target_size2)

# 创建SIFT对象
sift = cv2.SIFT_create()

# 检测关键点和计算描述子
kp1, des1 = sift.detectAndCompute(resized_img1, None)
kp2, des2 = sift.detectAndCompute(resized_img2, None)
```

```python

# 创建BFMatcher对象
bf = cv2.BFMatcher()

# 使用KNN匹配算法进行特征匹配
matches = bf.knnMatch(des1, des2, k=2)

# 应用比率测试，保留最佳匹配
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

# 绘制匹配结果
matching_result = cv2.drawMatches(resized_img1, kp1, resized_img2, kp2, good_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

# 显示匹配结果图像
cv2.imshow('Feature Matching Result', matching_result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## MOG 背景减弱

```python
import cv2

# 创建MOG背景减弱对象
mog = cv2.createBackgroundSubtractorMOG2()

# 读取视频
cap = cv2.VideoCapture('D:/18.mp4')

while True:
    # 读取每一帧
    ret, frame = cap.read()

    if not ret:
        break

    # 对当前帧进行背景减弱
    fg_mask = mog.apply(frame)

    # 显示背景减弱结果
    cv2.imshow('Foreground Mask', fg_mask)

    # 按下'q'键退出循环
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
# 释放视频流和窗口
cap.release()
cv2.destroyAllWindows()
```

*这里出来的是个视频 黑黑*

## 将视频作为新视频导出的代码

```python
import cv2

# 打开视频文件
cap = cv2.VideoCapture('D:/18.mp4')

# 获取视频的宽度和高度
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# 创建输出视频的写入器
output = cv2.VideoWriter('output_video.mp4', cv2.VideoWriter_fourcc(*'mp4v'), 30, (width, height))
#第一个参数是输出视频文件的名称或路径。在示例代码中，输出视频文件名为output_video.mp4。
#第二个参数是FourCC编码，用于指定视频编解码器。FourCC是一个四字符代码，用于唯一标识不同的视频编解码器。在示例代码中，cv2.VideoWriter_fourcc(*'mp4v')使用了MP4V编码器。
#第三个参数是帧率（Frames Per Second，FPS），用于指定视频中的帧数。在示例代码中，帧率被设置为30帧每秒。
#第四个参数是一个元组，指定输出视频的帧大小。在示例代码中，使用(width, height)来指定视频的宽度和高度。


# 创建MOG背景减弱器
mog = cv2.createBackgroundSubtractorMOG2()

while True:
    ret, frame = cap.read()
    if ret:
        # 对每一帧应用MOG背景减弱处理
        fgmask = mog.apply(frame)

        # 写入处理后的帧到输出视频文件
        output.write(fgmask)

        # 显示处理后的帧
        cv2.imshow('Processed Frame', fgmask)

        # 按下 'q' 键退出循环
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

```

```
# 释放视频流和写入器
cap.release()
output.release()

# 关闭所有窗口
cv2.destroyAllWindows()
```

*希望大家的新视频都能顺利打开！*

# Haar Cascade 面部检测

```
import numpy as np
import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades

#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
#加载脸部和眼部的层叠。
face_cascade = cv2.CascadeClassifier('D:/haarcascade_frontalface_default.xml')
#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade_eye.xml
eye_cascade = cv2.CascadeClassifier('D:/haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

#脸部的创建
while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    #拆分面部
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

        #找眼睛
        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

    cv2.imshow('img',img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

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

大脸图片我就不放了

*真的，需要问题问phind会更快一些。黑黑*

参考：https://wizardforcel.gitbooks.io/py-ds-intro-tut/content/opencv.html