Note of Opencv-Python

Min ChenYang

Hunan University

At October, 2019

目录

[1. Opencv在 pycharm中的配置 2](#_Toc22164861)

[2. 将照片读入到矩阵中，并显示 3](#_Toc22164862)

[3. 窗口操作 3](#_Toc22164863)

[4. waitKey函数 3](#_Toc22164864)

[5. 视频与电脑摄像头输入 4](#_Toc22164865)

[6. 获取图片的信息 4](#_Toc22164866)

[7. 色彩空间转换 4](#_Toc22164867)

[8. Print函数注释 4](#_Toc22164868)

[9. 遍历像素点 4](#_Toc22164869)

[10. 矩阵操纵（创建一幅图像） 5](#_Toc22164870)

[11. 获取程序执行时间 5](#_Toc22164871)

[12. 提取某颜色对应的像素 5](#_Toc22164872)

[13. 图像通道的合并、分离、单通道操作 6](#_Toc22164873)

[14. 图像算术运算、逻辑运算 7](#_Toc22164874)

[15. 调整对比度和亮度 7](#_Toc22164875)

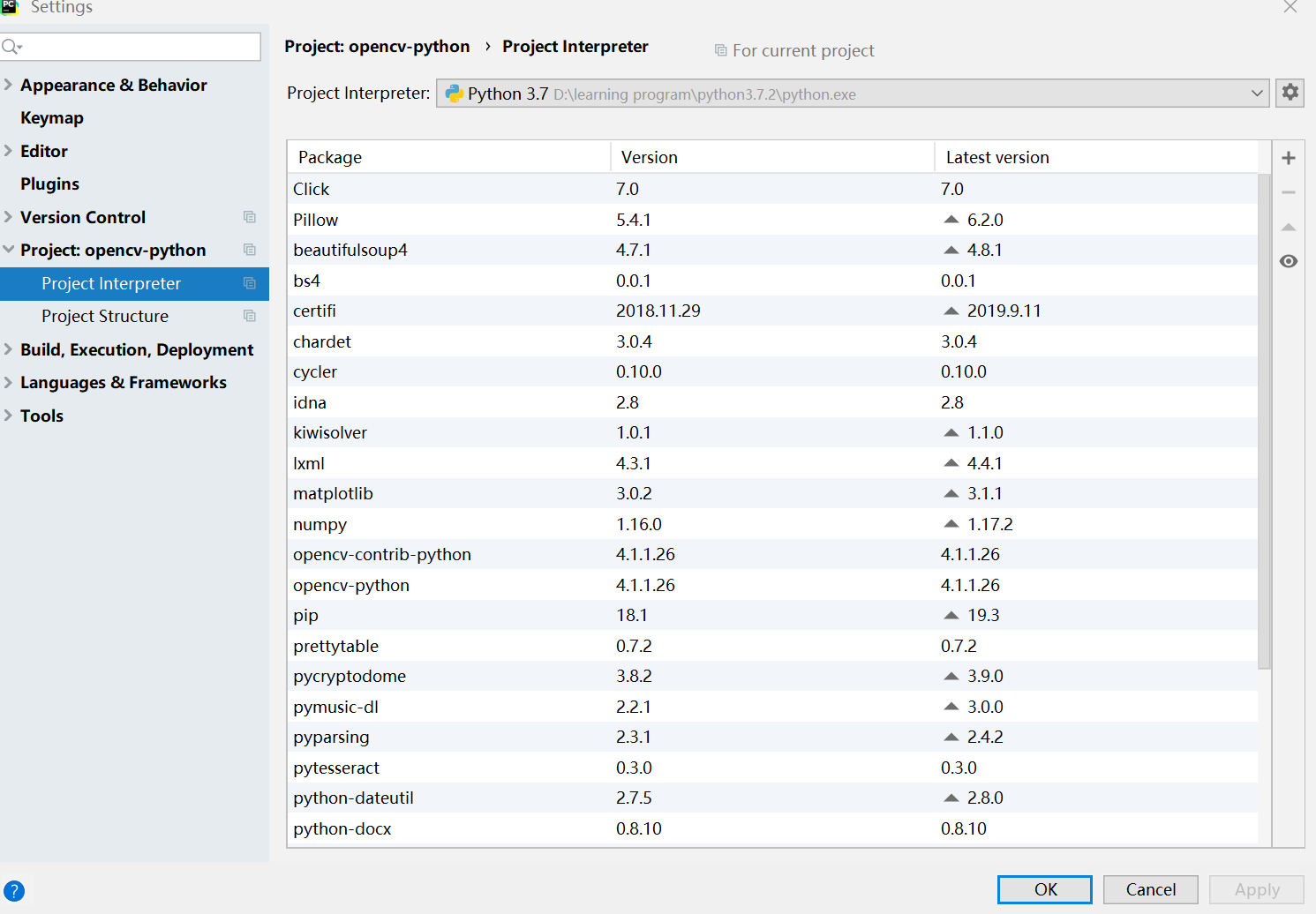
# Opencv在 pycharm中的配置

pip install opencv-python

pip install opencv-contrib-python #扩展库

pip install pytesseract

新建python项目，注意解释器正确配置应该如下：



# 将照片读入到矩阵中，并显示

src = cv.imread("D:/IMG\_20161227\_154705.jpg")

cv.imshow("input", src) #input为窗口名

cv.waitKey(0) 必须要有

cv.imwrite("D:/result.jpg", gray)

# 窗口操作

cv.namedWindow(**"input"**, cv.WINDOW\_AUTOSIZE) 适应图片大小

cv. WINDOW\_NORMAL 窗口大小可调节

CV\_WINDOW\_OPENGL) 支持OpenGL

namedWindow()创建一个窗口。imshow可以直接指定窗口名，可以省去此函数（默认调用），但如果显示图像之前需要其他窗口操作时，需要调用此函数

destroyWindow() 关闭特定窗口 # cv.destroyWindow(**"video"**)

destroyAllWindows()关闭所有的HighGUI窗口

cv.startWindowThread()

在调用cv.startWindowThread();后，即使没有调用waitKey()函数，图片也依然实时刷新。opencv的imshow()函数调用以后，并不立即刷新显示图片，而是等到waitKey()后才会刷新图片显示，所以我猜测cv.startWindowThread();是新开一个线程实时刷新图片显示。

# waitKey函数

* 使用OpenCV的imshow函数显示图片，必须配合waitKey 函数使用，才能将图片显示在windows窗体上。否则，imshow 函数单独使用只能弹出空白窗体，而无法显示图片。
* waitKey的时间延迟，只对Windows窗体有效，而且是 namedWindow 函数创造的OpenCV窗体，对于MFC或者Qt这种GUI窗体是否有效是一种未知结果
* 真正能起到程序暂停的作用的是我们熟悉的Windows API函数Sleep

# 视频与电脑摄像头输入

def video\_demo(): #无输入值

#capture = cv.VideoCapture("D:/IMG\_9764.MP4") #0为读取电脑摄像头，读取的视频无声音，大小有限制

capture = cv.VideoCapture(0)

while(True):

ret, frame = capture.read() #返回值，每一帧

frame1 = cv.flip(frame, 1) #镜像变换 1为左右 -1为上下

frame2 = cv.transpose(frame) #顺时针旋转90°

cv.imshow("video", frame) #每一帧循环显示

cv.imshow("video1", frame1)

cv.imshow("video2", frame2)

c = cv.waitKey(50) #响应用户操作

if c == 27:

break

# 获取图片的信息

def get\_image\_info(image):

print(type(image)) # <class 'numpy.ndarray'>

print(image.shape) #显示高，宽，通道数

print(image.size) #总的像素数据大小=高\*宽\*通道数

print(image.dtype) #显示像素数据类型

pixel\_data = np.array(image) #通过numpy获取像素值

# print(pixel\_data)

print(image) 可以直接打印

# 色彩空间转换

gray = cv.cvtColor(src, cv.COLOR\_BGR2GRAY) #获取灰度图像

dst = cv.bitwise\_not(image) # 通过逻辑非运算来获得负片

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

back\_rgb = cv.cvtColor(gray, cv.COLOR\_GRAY2BGR)

# Print函数注释

打印变量值

print("width : %s, height : %s channels : %s" % (width, height, channels))

打印矩阵

print(image)

# 遍历像素点

def access\_pixels(image):

height = image.shape[0]

width = image.shape[1]

channels = image.shape[2]

print("width : %s, height : %s channels : %s" % (width, height, channels))

for row in range(height):

for col in range(width):

for c in range(channels):

pv = image[row, col, c]

image[row, col, c] = 255 – pv

cv.imshow("demo", image)

# 矩阵操纵（创建一幅图像）

ones创建任意维度和元素个数的数组，其元素值均为1

empty一样，只是它所常见的数组内所有元素均为空

zeros([m,n…],int8) 创建任意维度和元素个数的数组，其元素值均为0

img = np.zeros([400, 400, 3], np.uint8)

#img = np.ones([400, 400, 3]) \* 255

# img[:, :, 2] = np.ones([400, 400])\*255 #对2通道像素平面进行操作

# cv.imshow("new image", img)

matrix = np.ones([6, 6], np.float32) # 有浮点数计算一定选float

fill用来填充矩阵，

reshape可以进行矩阵重组，元素数相同

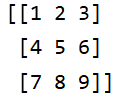
matrix.fill(1625.35)

cv.convertScaleAbs() 可以将浮点数转化为uint8 ，负数转换为绝对值

m2 = matrix.reshape([3, 12])

array生成任意矩阵，可以作为算子

m3 = np.array([[1,2,3], [4,5,6], [7,8,9]],np.int32)



# 获取程序执行时间

t1 = cv.getTickCount()create\_image() #程序t2 = cv.getTickCount()  
time = (t2 - t1) / cv.getTickFrequency()  
print("time = %s ms" % (time \* 1000))

可以通过调用opencv自带的API来减少程序执行时间

# 提取某颜色对应的像素

思路：转换到HSV空间，再参考下表设置inRange函数的参数(红色设置为第二列较佳)



def extract\_object\_demo():

capture = cv.VideoCapture("D:/IMG\_9764.MP4")

while(True):

ret, frame = capture.read() # 先将每一帧读入

if ret == False:

break

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

lower\_hsv = np.array([0, 43, 46]) # 找出白色

upper\_hsv = np.array([10, 255, 255])

mask = cv.inRange(hsv, lowerb=lower\_hsv, upperb=upper\_hsv)

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

cv.imshow("video", frame)

cv.imshow("mask", dst) #将含有红色的像素提取以红色黑色视频中显示

c = cv.waitKey(50)

if c == 27:

break # escape

# 图像通道的合并、分离、单通道操作

b, g, r = cv.split(src)

cv.imshow("blue", b)

cv.imshow("green", g)

cv.imshow("red", r)

src = cv.merge([b, g, r]) # 注意此处的输入

src[:, :, 0] = 0

cv.imshow("changed image", src)

h, w = src.shape[0:2] #获取图像的高与宽，0可以不输入

print(src[30, 30, :]) #打印某位置上的三个像素值

# 图像算术运算、逻辑运算

dst = cv.add(m1, m2) #相加

dst = cv.subtract(m1, m2) #相减

dst = cv.divide(m1, m2) #除

dst = cv.multiply(m1, m2) #乘

M1 = cv.mean(m1) #获取均值

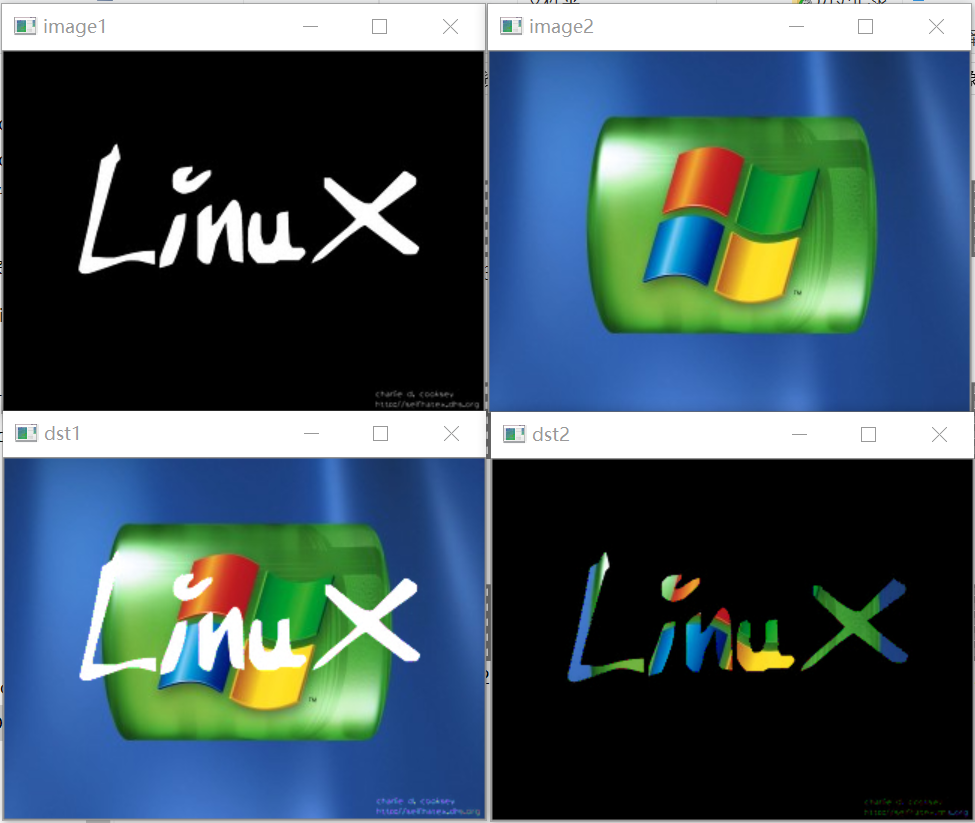
M2, dev2 = cv.meanStdDev(m2) #获取均值和方差

Tips：方差越小，则该图片包含的信息越少，可设置阈值来过滤无意义的图片

dst1 = cv.bitwise\_or(m1, m2)

dst2 = cv.bitwise\_and(m1, m2) # 可以作为一个“遮罩”

dst3 = cv.bitwise\_not(m1) #获得负片



# 调整对比度和亮度

def contrast\_brightness\_demo(image, c, b):

h, w, ch = image.shape #取出shape的前两位【：2】

blank = np.zeros([h, w, ch], image.dtype)

dst = cv.addWeighted(image, c, blank, 1-c, b) #调整对比度和亮度,none的均可以接收，dst = src1\*alpha + src2\*beta + gamma

cv.imshow("con\_bri\_demo", dst)

像素运算式：dst = src1\*alpha + src2\*beta + gamma

1. ROI选择

face = src[50:250, 100:300] # [height, width]

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

backrgb = cv.cvtColor(gray, cv.COLOR\_GRAY2BGR)

src[50:250, 100:300] = backrgb

cv.imshow("face", src)

1. 泛洪填充

def fill\_color\_demo(image):

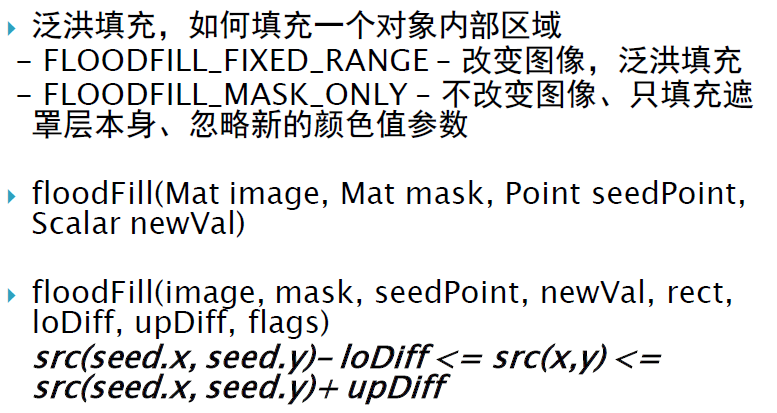
copyImg = image.copy()

h, w = image.shape[:2]

mask = np.zeros([h+2, w+2], np.uint8) #底层代码要求必须这么写

cv.floodFill(copyImg, mask, (30, 30), (0, 255, 255), (100, 100, 100), (50, 50, 50), cv.FLOODFILL\_FIXED\_RANGE)

cv.imshow("fill\_color\_demo", copyImg)



def fill\_binary():

image = np.zeros([400, 400, 3], np.uint8)

image[100:300, 100:300, :] = 255

cv.imshow("fill\_binary", image)

mask = np.ones([402, 402, 1], np.uint8)

mask[100:300, 100:300] = 0

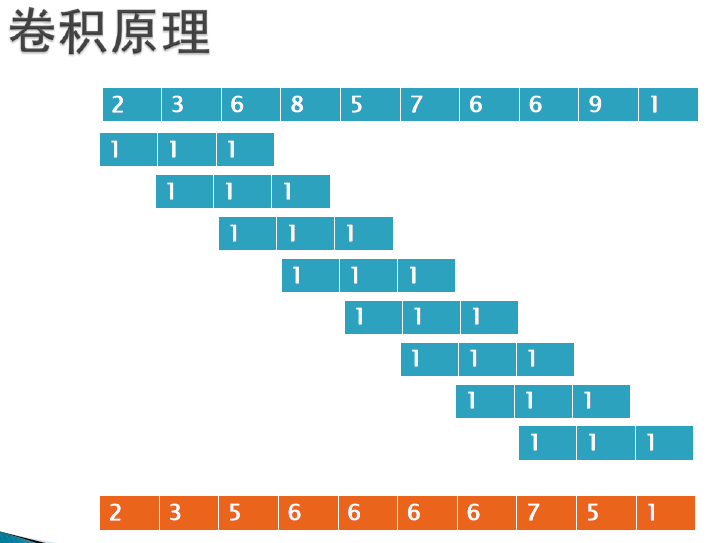
cv.floodFill(image, mask, (200, 200), (255, 255, 0), cv.FLOODFILL\_MASK\_ONLY)#只填充mask标记为（0，0）的像素点

cv.imshow("filled binary", image)

1. 模糊操作

关于算子：元素个数为奇数，总和为0：进行边缘和梯度计算，总和为1进行增强锐化等

Tips：blurry模糊的，不清楚的，污脏的



dst = cv.blur(image, (1, 15)) #均值模糊，模糊只是卷积的表象

dst = cv.medianBlur(image, 5) #中值模糊

def custom\_blur\_demo(image): #自定义卷积核来模糊

# kernel = np.ones([5, 5], np.float32)/25 #最多25个255，防止溢出

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]], np.float32)

dst = cv.filter2D(image, -1, kernel=kernel)

cv.imshow("custom\_blur\_demo" ,dst)