Table of Contents

[Image processing in OpenCV 2](#__RefHeading___Toc106_2567912753)

[Changing Colorspaces 2](#__RefHeading___Toc108_2567912753)

[Geometric(几何的) Transformations of Images 2](#__RefHeading___Toc110_2567912753)

[Image Thresholding(阈值) 2](#__RefHeading___Toc112_2567912753)

[Smoothing Images　图像平滑处理　 2](#__RefHeading___Toc114_2567912753)

[Morphological(形态) Transformations，　变粗、变细，... 3](#__RefHeading___Toc116_2567912753)

[Image Gradients(梯度) 3](#__RefHeading___Toc118_2567912753)

[Canny Edge Detection　Canny边界检测算法 3](#__RefHeading___Toc120_2567912753)

[Image Pyramids 金字塔 3](#__RefHeading___Toc122_2567912753)

[Contours(外形，轮廓，等高线) 3](#__RefHeading___Toc124_2567912753)

[Histograms(直方图) in OpenCV 4](#__RefHeading___Toc126_2567912753)

[Image Transforms in OpenCV 图像变换（傅里叶变换） 5](#__RefHeading___Toc128_2567912753)

[Template Matching 模板匹配 6](#__RefHeading___Toc130_2567912753)

[Hough Line Transform 6](#__RefHeading___Toc132_2567912753)

[Hough Circle Transform 6](#__RefHeading___Toc134_2567912753)

[Image Segmentation with Watershed(分水岭，转折点) Algorithm 6](#__RefHeading___Toc136_2567912753)

[Interactive(互动的) Foreground Extraction using GrabCut(抓斗) Algorithm 6](#__RefHeading___Toc138_2567912753)

[Raspberry pi 4 7](#__RefHeading___Toc140_2567912753)

# Image processing in OpenCV

## Changing Colorspaces

cv2.cvtColor(),

cv2.inRange() 判断一些点的颜色是否在某个范围内，返回一个0,1矩阵，可用作掩码。

　　cv2.bitwise\_\* 对像素点进行位操作

　　BGR 颜色空间与其他颜色空间的转化技巧：green = np.uint8([0,255,0]) hsvgreen=cv2.cvtColor(green,cv2.COLOR\_BGR2HSV)

## Geometric(几何的) Transformations of Images

　　cv2.resize()改变图像的大小，用算法对图像缩放处理

　　cv2.wrapAffine() 仿射变换。在向量空间中进行一次线性变换（乘以一个矩阵）加上一个平移。矩阵实例：M = np.float32([1,0,100],[0,1,50]), 合适的矩阵可以实现图像的旋转效果，cv2.getRotationMatrix2D((centerX,centerY),degree,scale)函数可以得到这样的一个矩阵。合适的矩阵可以实现图像的不规则变形（三个点的坐标变换到另一组新的位置），这个矩阵也可以用过cv2.getAffineTransform(oldPos,newPos)得到。

　　cv2.warpPerspective() 透视变换。基于透视原理进行图像的变换（至少需要四个点的对应位置），有点类似用上面的getAffineTransform得到的矩阵进行的放射变换。

## Image Thresholding(阈值)

　　cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

区别：

THRESH\_BINARY:当前点值大于阈值时，取Maxval,否则取０

THRESH\_BINARY\_INV: 当前点值大于阈值时，取０,否则取Maxval

THRESH\_TRUNC:当前点值大于阈值时，设置为阈值，否则不改变

THRESH\_TOZERO: 当前点值大于阈值时，不改变，否则设置为0

THRESH\_TOZERO\_INV: 当前点值大于阈值时，设置为0，否则不改变

THRESH\_MASK:

THRESH\_OTSU: 自适应阈值，可与其他选项或。

THRESH\_TRIANGLE:用三角算法选取阈值，可与其他选项或，不用自己指定阈值。

　　自适应的阈值算法

　　cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY or THRESH\_BINARY\_INV,blockSize,c)

　　ADAPTIVE\_THRESH\_MEAN\_C：局部邻域块的均值，减去常数C

　　ADAPTIVE\_THRESH\_GAUSSIAN\_C：局部邻域块的高斯加权和，减去常数C

## Smoothing Images　图像平滑处理

cv2.filter2D(img,-1,kernel), kernel 为一单通道浮点数矩阵，用于卷积计算。如何得到这个矩阵呢？

　　cv2.blur(img,(5,5))，用均值模糊，(5,5)为内核大小为５Ｘ５

　　cv2.GaussianBlur(img,(5,5),0)，高斯模糊

　　cv2.medianBlur(img,5)　中值滤波模糊（消除椒盐噪声有奇效），5为模板大小，大于１的奇数

　　cv2.bilateralFilter(img,9,75,75)双边滤波(边缘保存)

## Morphological(形态) Transformations，　变粗、变细，...

cv2.erode(img,kernel,iterations = 1) 侵蚀，腐蚀。kernel可以用getStructuringElement函数获取。

cv2.dilate(img,kernel,iterations = 1) 扩大，膨胀

cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)，先膨胀后侵蚀，可以消除外围噪点。

cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)，先侵蚀后膨胀，可消除物体上的噪声

cv2.morphologyEx(img, cv2.MORPH\_GRADIENT, kernel),物体的空心轮廓线

cv2.morphologyEx(img, cv2.MORPH\_TOPHAT, kernel),礼帽运算

cv2.morphologyEx(img, cv2.MORPH\_BLACKHAT, kernel),黑帽运算

cv2.getStructuringElement(cv2.MORPH\_RECT,(5,5))得到上述kernel的算法

## Image Gradients(梯度)

找到图像的坡度，边缘, 三种滤波器：High-pass filters(高通), Sobel(索贝尔), Scharr and Laplacian(沙尔和拉普拉斯)

laplacian = cv2.Laplacian(img,cv2.CV\_64F) 拉普拉斯变换

sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)　索贝尔变换X

sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5)　索贝尔变换Y

需要注意上述的数据类型，如采用cv2.CV\_8U则可能损失信息。

## Canny Edge Detection　Canny边界检测算法

1986年开发的多级边缘检测：

.应用高斯滤波来平滑图像，目的是去除噪声

.找寻图像的强度梯度（intensity gradients）

.应用非最大抑制（non-maximum suppression）技术来消除边误检（本来不是但检测出来是）

.应用双阈值的方法来决定可能的（潜在的）边界

.利用滞后技术来跟踪边界

edges = cv2.Canny(img,100,200)，100,200为threshold1和threshold2,上述滞后处理中的两个阈值.

## Image Pyramids 金字塔

同一张图片的不同分辨率，案例：融合两张图片。

cv2.pyrDown(img)

cv2.pyrUp(img)

## Contours(外形，轮廓，等高线)

找出轮廓线

im2, contours, hierarchy = cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)经过单色，阈值处理后.

画出轮廓线

cv2.drawContours(img, contours, -1, (0,255,0), 3)

## Histograms(直方图) in OpenCV

使用openCV计算单通道图像的直方图，了解每个像素点颜色的分布图

hist = cv2.calcHist([img],[0],None,[256],[0,256])

也可以使用NumPy的函数

hist,bins = np.histogram(img.ravel(),256,[0,256])

使用matplotlib绘制

plt.hist(img.ravel(),256,[0,256]); plt.show()

如果只想获取某一区域的直方图，使用一个该区域的Mask

mask = np.zeros(img.shape[:2], np.uint8)

mask[100:300, 100:400] = 255

masked\_img = cv2.bitwise\_and(img,img,mask = mask)

hist\_mask = cv2.calcHist([img],[0],mask,[256],[0,256])

直方图的均衡化，将原始的直方图拉伸，增强对比度，其思想是把原始图像的的灰度直方图从比较集中的某个区域变成在全部灰度范围内的均匀分布。旨在使得图像整体效果均匀，黑与白之间的各个像素级之间的点更均匀一点。

img = cv2.imread('wiki.jpg',0)

equ = cv2.equalizeHist(img)

CLAHE(Contrast Limited Adaptive Histgram Equalization)限制对比度自适应直方图均衡

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

cl1 = clahe.apply(img)

2D直方图，表示两个元素的相互影响

hist = cv2.calcHist([hsv], [0, 1], None, [180, 256], [0, 180, 0, 256])

参数：

[0,1]表示0和1通道

None表示没有mask

[180,256]表示0通道的bins为180，1通道的bins为256

[0,180,0,256]表示0通道的range为0到180，1通道的range为0到256

hist, xbins, ybins = np.histogram2d(h.ravel(),s.ravel(),[180,256],[[0,180],[0,256]])

直方图反向投影：首先计算某一特征的直方图模型，然后使用模型去寻找测试图像中存在的该特征。也就是根据样本的直方图，找到图像与样本相似的地方，多半在HSV色彩空间

https://blog.csdn.net/shuiyixin/article/details/80331839

幅图像的反向投影利用了其原始图像（或目标区域）的直方图，将该直方图作为一张查找表来找对应像素点的像素值，即将目标图像像素点的值设置为原始图像（或目标区域）直方图上对应的bin值。该bin值代表了（目标区域）上该像素值出现的概率。从而得到一幅图像的概率值。从而我们可以通过这幅概率图可以得知在这幅图像中，目标出现可能出现的位置。

反向投影

　image=

0 1 2 3

4 5 6 7

8 9 10 11

8 9 14 15

Histogram = 4 4 6 2 （bin的区间为[0,3],[4,7],[8,11],[12,16]）

反向投影back\_project=

4 4 4 4

4 4 4 4

6 6 6 6

6 6 2 2

　根据图像每个点所在直方图中的位置，用直方图中对应的值设置为像素点对应的值。

　一幅图像的反向投影利用了其原始图像（或目标区域）的直方图，将该直方图作为一张查找表来找对应像素点的像素值，即将目标图像像素点的值设置为原始图像（或目标区域）直方图上对应的bin值。该bin值代表了（目标区域）上该像素值出现的概率。从而得到一幅图像的概率值。从而我们可以通过这幅概率图可以得知在这幅图像中，目标出现可能出现的位置。

肤色检查中效果明显

## Image Transforms in OpenCV 图像变换（傅里叶变换）

频域里的分析，高频信号和低频信号，边界和噪声是高频信号，图像主要部分是在低频分量

NumPy中的傅里叶变化：

f = np.fft.fft2(img)

#默认结果中心点位置在左上角，通过下面函数转移到中间

fshift = np.fft.fftshift(f)

#fshift是复数，求绝对值结果才是振幅

magnitude\_spectrum = 20\*np.log(np.abs(fshift))

逆变换，

f1shift = np.fft.ifftshift(fshift)

img\_back = np.fft.ifft2(f1shift)

#出来的复数，无法显示，转成实数

img\_back = np.abs(img\_back) #在频域里面做一些处理，再逆变回来，可以和原图比较差异

OpenCV中的傅里叶变换cv2.dft() cv2.idft()

dft = cv2.dft(np.float32(img),flags = cv2.DFT\_COMPLEX\_OUTPUT)

dft\_shift = np.fft.fftshift(dft)

经过傅里叶变换，在频域里处理，再逆变回来的例子

rows, cols = img.shape

crow,ccol = rows/2 , cols/2

#Mask

mask = np.zeros((rows,cols,2),np.uint8)

mask[crow-30:crow+30, ccol-30:ccol+30] = 1

#频域里处理

fshift = dft\_shift\*mask

#逆变回来

f\_ishift = np.fft.ifftshift(fshift)

#shift反操作

img\_back = cv2.idft(f\_ishift)

#图像复原 magnitude用来计算二维矢量的幅值

img\_back = cv2.magnitude(img\_back[:,:,0],img\_back[:,:,1])

傅里叶变换的效率，cv2.getOptimalDFTSize() 可用来选取合适的尺寸

img = cv2.imread('messi5.jpg',0)

rows,cols = img.shape

nrows = cv2.getOptimalDFTSize(rows)

ncols = cv2.getOptimalDFTSize(cols)

nimg = np.zeros((nrows,ncols))

#拷贝原图像到新的这个Size中

nimg[:rows,:cols] = img

#下面的傅里叶变换效率大大提高近4倍

cv2.dft(np.float32(nimg),flags=cv2.DFT\_COMPLEX\_OUTPUT)

滤波器的运用

## Template Matching 模板匹配

cv2.matchTemplate(img,template,method),其中method可为：

cv2.TM\_CCOEFF

cv2.TM\_CCOEFF\_NORMED

cv2.TM\_CCORR

cv2.TM\_CCORR\_NORMED

cv2.TM\_SQDIFF

cv2.TM\_SQDIFF\_NORMED

字符串转代码技巧:

meth='cv2.TM\_COFF'

method = eval(meth)

cv2.matchTemplate(img,template,method)

如需匹配多个目标，需要在返回结果中根据指定一个阈值来获取:

res = cv2.matchTemplate(img\_gray,template,cv2.TM\_CCOEFF\_NORMED)

threshold = 0.8

loc = np.where( res >= threshold)

...

## Hough Line Transform（霍夫变换-直线）

一种检测直线的方法，首先需要对图像进行边缘检测处理，从平面空间到极坐标空间。

img = cv2.imread('sudoku.png')

#颜色空间转换

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

#边缘检测处理

edges = cv2.Canny(gray,50,150,apertureSize = 3)

#直线变换

lines = cv2.HoughLines(edges,1,np.pi/180,200)

for line in lines:

rho,theta = line[0] #极坐标角度和极径， 可得到对应的x,y坐标

另一种方法：

lines = cv2.HoughLinesP(edges,1,np.pi/180,100,minLineLength=100,maxLineGap=100)

## Hough Circle Transform

## Image Segmentation with Watershed(分水岭，转折点) Algorithm

## Interactive(互动的) Foreground Extraction using GrabCut(抓斗) Algorithm

.

# Raspberry pi 4

. Download OS image(with desktop) from official website

. Formate SD card with SD Card formatter tool

. Write OS image downloaded in step 1 with Win32DiskImager

. Start Raspberry pi

. HDMI 0 will display desktop and begin to set up

. run Raspberry Pi Configuration from raspberry menu in left-top

. select Interface tab, and enable ssh and vnc

Install OpenCV

. python3 and python2 are installed already

. sudo apt-update

. apt list python\*opencv\*, below list shows an OpenCV version for raspberry Pi4:

python-opencv-apps/stable 1.12.0-2 all

python-opencv/stable 3.2.0+dfsg-6 army

python3-opencv-apps/stable 1.12.0-2 all

python3-opencv/stable,now 3.2.0+dfsg-6 army

. Install OpenCV: sudo apt install python3-opencv

. try again and again if it failed because missing some package.

Solve the problem of "Cannot currently show the dekstop"

. Log in Raspberry Pi4

. sudo rasps-config

. change the resolution to a lower value and connect VNC again.

how to get the CPU temperature?

cat /sys/class/thermal/thermal\_zone0/temp

how to add a user and change his group information?

sudo adduser XXX

sudo usermod -a -G GROUPNAME USERNAME

reboot it to take effect

how to use an existed conda python environment in pycharm?

. select "Existing interpreter" for "Project interpreter",select the python which located in conda environment path, for example: /home/XXXX/anaconda3/bin/python