#OpenCV库：

pip install opencv-python==3.4.2.17

import cv2 as cv

实例：

# 导入 OpenCV 库  
**import** cv2  
  
# 1. 读取图像  
# 替换为实际的图像路径，这里是当前目录下的 "bird.jpg"  
image\_path = "./bird.jpg"  
image = cv2.imread(image\_path)  
  
# 检查图像是否成功读取  
**if** image **is** None:  
    **print**("错误：无法加载图像，请检查路径是否正确。")  
    exit()  
  
# 2. 显示图像  
# 创建一个名为 "Display Image" 的窗口，并在其中显示图像  
cv2.imshow("Display Image", image)  
  
# 3. 等待用户按键  
# 参数 0 表示无限等待，直到用户按下任意键  
key = cv2.waitKey(0)  
  
# 4. 根据用户按键执行操作  
**if** key == ord('s'):  # 如果按下 's' 键  
    # 保存图像  
    output\_path = "saved\_image.jpg"  
    cv2.imwrite(output\_path, image)  
    **print**(f"图像已保存为 {output\_path}")  
**else**:  # 如果按下其他键  
    **print**("图像未保存。")  
  
# 5. 关闭所有窗口  
cv2.destroyAllWindows()

图像的基本属性：

1.numpy的array数组(三维),**axis=0为高,axis=1为宽**,axis=2为三位颜色通道(RGB);

2.数据类型一般是**uint8**(0~255).

3.可以通过img[**y , x**]访问像素(x,y)的RGB值，通过img[**y1:y2,x1:x2**]选定ROI区域的RGB值。

4.可以通过 img[**y, x, c**] 来访问特定通道 c 的像素值，其中 c = 0（蓝b）、1（绿g）或 2（红r）。

可以用b,g,r=**cv.split**(img)来通道分离，用merged\_img=**cv.merge**([b,g,r])来通道合并。

5.RGB的三个数为[255,255,255]时为白色,[0,0,0]为黑色。

读取图像：img = cv.imread(path,flags=1),flags的选择：

1 = cv.IMREAD’COLOR彩色，默认

0 = cv.IMREAD’GRAYSCALE灰色

-1 = cv.IMREAD\_UNCHANGED包括alpha通道的加载图像模式

显示图像：cv.imshow(‘ThisImage’,img)显示一个标题为ThisImage的窗口

等待按键：key=cv.waitKey(delay=0)延时delay秒，0表示无限等待

用户按键类型：ord(‘s’)按下s键,返回q的ASCII码值

输出图像：cv.imwrite(path,img)

关闭窗口：cv.destroyAllWindows()

图像的缩放、旋转、平移、翻转：

1.缩放：cv.**resize**(img,<shape>)其中shape是二元组

2.旋转：

两个函数：

cv.**getRotationMatrix2D**(center,angle,scale)获取旋转或缩放的**变换矩阵**

cv.**warpAffine**(img,M,(w,h)) **仿射变换**

eg：

h,w=img.shape[:2]

center=(w//2,h//2) #获取中心,注意先x后y

M=cv.getRotationMatrix2D(center,45,1.0) #获取变换矩阵

rotated\_img=cv2.warpAffine(img,M,(w,h)) #仿射变换获取旋转的矩阵

3.平移：

M=np.float32([[1,0,100],

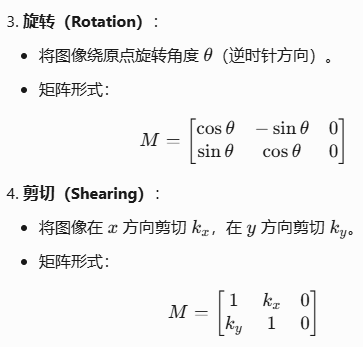
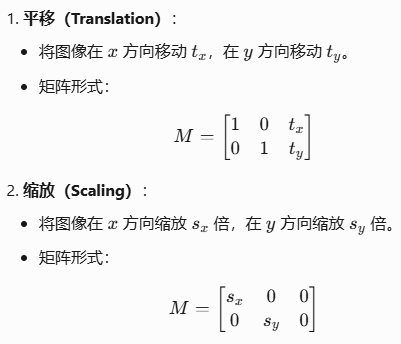
[0,1,50]]) #获取平移矩阵,此矩阵表示右移100下移50

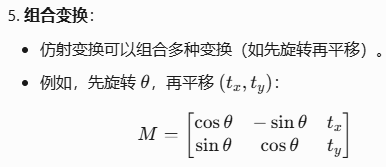
moved\_img=cv.**warpAffine**(img,M,(w,h))

4.翻转(镜像):

flipped\_img=cv.**flip**(img,1) #1为水平镜像，0为垂直镜像，(0,1)中心对称

图像的仿射变换矩阵M的常见结论：





注：平移变换比较简单因此可以直接写出M，缩放也可以直接写出M，而旋转一般使用

cv.getRotationMatrix2D()来获取M.

图像的算术运算：

cv.add(img1,img2)+ subtract- multiply\* divide/

bitwisw\_and-- & bitwise\_or-- | bitwise\_not-- ~ bitwise\_xor-- ^

权重混合：

cv.addWeighted(img1,alpha,img2,beta)将img1以alpha权重,img2以beta权重混合

图像的阈值处理：

retval,dst=cv.**threshold**(img,thresh,maxval,type)

src: 输入图像，通常为灰度图像。

thresh: 设定的阈值。

maxval: 当像素值超过（或小于，根据类型）阈值时，赋予的新值。

type: 阈值处理的类型，常见的类型有：

0--cv.THRESH\_BINARY: 如果像素值大于阈值，则赋予 maxval，否则赋予 0。

1--cv.THRESH\_BINARY\_INV: 与 cv2.THRESH\_BINARY 相反，如果像素值大于阈值，则赋予 0， 否则赋予 maxval。

2--cv.THRESH\_TRUNC: 如果像素值大于阈值，则赋予阈值，否则保持不变。

3--cv.THRESH\_TOZERO: 如果像素值大于阈值，则保持不变，否则赋予 0。

4--cv.THRESH\_TOZERO\_INV: 与 cv2.THRESH\_TOZERO 相反，如果像素值大于阈值，则赋予 0， 否则保持不变。

8--cv.THRESH\_OTSU:Otsu阈值法，自动计算最佳阈值，通常和0,1一起使用(+)

16-cv.THRESH\_TRIANGLE:Triangle阈值法，自动计算最佳阈值，通常和3,4一起使用

返回值：

retval: 实际使用的阈值（在某些情况下可能与设定的阈值不同）。

dst: 处理后的图像。

图像的平滑处理：

均值滤波：cv.**blur**(img,(5,5))对img的每个像素滤波(在滤波核(5,5)范围内取平均值)

这个函数有个缺点，边缘会变得模糊。

高斯滤波：cv.**GaussianBlur**(img,(5,5),sigmaX=0)有效避免边缘模糊问题

中值滤波：cv.**medianBlur**(img,5)有效去除图像中出现的黑白点，滤波核必须是奇数

双边滤波：cv.**bolateralFilter**(img,9,75,75)去除噪声的同时保留图像的边缘信息