# cv2.warpAffine() 仿射变换

基于该方法去寻找相关资料，得到仿射变换的基本概念，从二维坐标到二维坐标之间的线性变换，并且要保持二维图形的“平直性” 和 “平行性”。

仿射变换中，包含平移，缩放，翻转，旋转，剪切。

该方法的语法结构如下：

dst = cv2.warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])

参数说明：

* src：输入图像
* M：2\*3 transformation matrix (转变矩阵)
* dsize：输出图像的大小，格式为(cols,rows)，width 对应 cols，height 对应 rows
* flags： 可选参数，插值方法的组合（int 类型），默认值 INTER\_LINEAR
* borderMode：可选参数，边界像素模式（int 类型），默认值 BORDER\_CONSTANT
* borderValue：可选参数，边界填充值; 默认情况下，默认值 Scalar()即 0

官方可查阅的英文资料如下：



插值方式（flags）有如下类型：

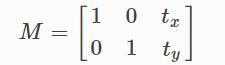
* INTER\_LINEAR 线性插值
* INTER\_NEAREST 最近邻插值
* INTER\_AREA 区域插值
* INTER\_CUBIC 三次样条插值
* INTER\_LANCZOS4 Lanczos 插值

一般情况下，使用 wrapAffine 前三个参数，即 warpAffine(img,M,(rows,cols)) 实现基本的仿射变换效果，但是这种情况会出现 黑边 现象。

最后一个参数为 borderValue，边界填充的颜色，默认为黑色，可以自行设置为其它颜色。

**具体代码**

图像平移是图像位置的移动。在平移前，需要先转换矩阵 M，，其中 tx，ty 表示在(x,y)方向上的位移，如下所示：



测试代码如下，重要部分在注释中进行说明。

import numpy as np

import cv2 as cv

img = cv.imread('10.jpg', 1)

rows, cols, channels = img.shape

# [1,0,100] 右移 100， [0,1,50] 下移 50

M = np.float32([[1, 0, 100], [0, 1, 50]])

res = cv.warpAffine(img, M, (cols, rows))

cv.imshow("img", res)

cv.waitKey()

特别说明：  
cv.warpAffine 方法的第三个参数是输出图像的大小，其形式应为（width，height）。记住 width =列数（cols），height =行数（rows）。

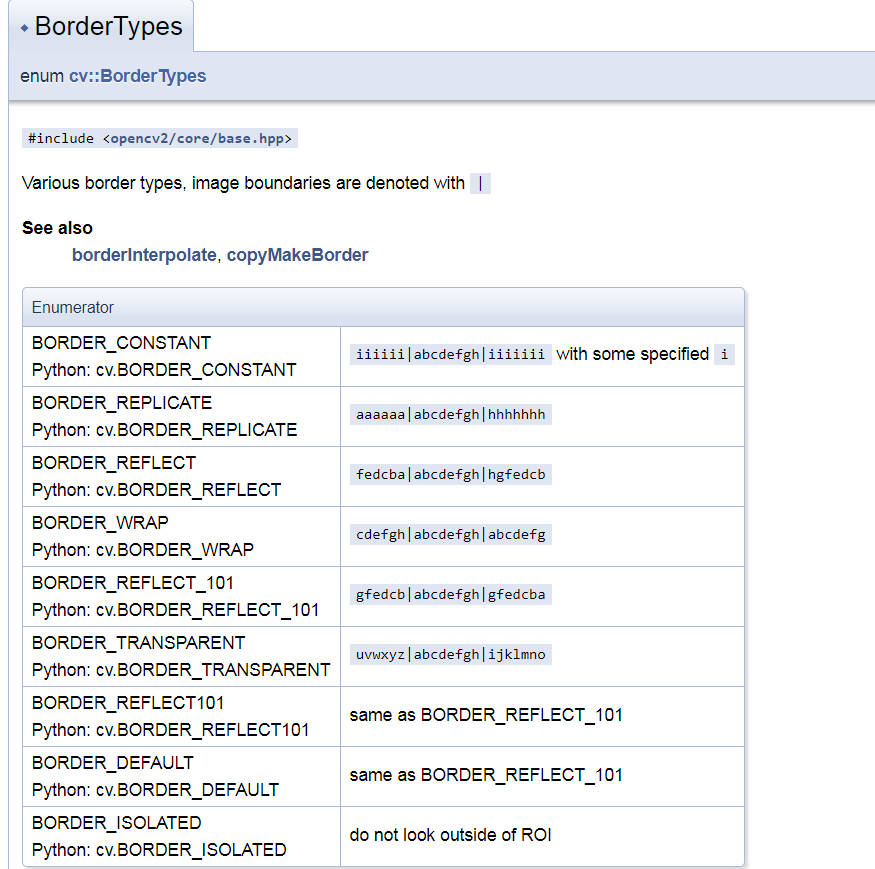
以下为代码运行结果。



关于 borderMode 参数值的说明：  
该部分内容涉及边缘处理，翻阅手册过程找到的说明指向 BorderTypes。翻阅了一下，依旧是有些超纲，可以先记录下相关内容，日后在进行回顾，参考地址。

所有值罗列如下：

* BORDER\_CONSTANT = 0
* BORDER\_REPLICATE = 1
* BORDER\_REFLECT = 2
* BORDER\_WRAP = 3
* BORDER\_REFLECT\_101 = 4
* BORDER\_TRANSPARENT = 5
* BORDER\_REFLECT101 = BORDER\_REFLECT\_101
* BORDER\_DEFAULT = BORDER\_REFLECT\_101
* BORDER\_ISOLATED = 16



borderValue：当图像边界处理方式为 BORDER\_CONSTANT 时的填充值。

测试代码如下：

import numpy as np

import cv2 as cv

img = cv.imread('10.jpg', 1)

rows, cols, channels = img.shape

M = np.float32([[1, 0, 100], [0, 1, 50]])

res = cv.warpAffine(img, M, (cols, rows),

borderMode=cv.BORDER\_CONSTANT, borderValue=(200, 66, 66))

cv.imshow("img", res)

cv.waitKey()