该系列文章是讲解Python OpenCV图像处理知识，前期主要讲解图像入门、OpenCV基础用法，中期讲解图像处理的各种算法，包括图像锐化算子、图像增强技术、图像分割等，后期结合深度学习研究图像识别、图像分类应用。希望文章对您有所帮助，如果有不足之处，还请海涵~

该系列在github所有源代码：https://github.com/eastmountyxz/ImageProcessing-Python
PS：请求帮忙点个Star，哈哈，第一次使用Github，以后会分享更多代码，一起加油。

同时推荐作者的C++图像系列知识：
[数字图像处理] 一.MFC详解显示BMP格式图片
[数字图像处理] 二.MFC单文档分割窗口显示图片
[数字图像处理] 三.MFC实现图像灰度、采样和量化功能详解
[数字图像处理] 四.MFC对话框绘制灰度直方图
[数字图像处理] 五.MFC图像点运算之灰度线性变化、灰度非线性变化、阈值化和均衡化处理详解
[数字图像处理] 六.MFC空间几何变换之图像平移、镜像、旋转、缩放详解
[数字图像处理] 七.MFC图像增强之图像普通平滑、高斯平滑、Laplacian、Sobel、Prewitt锐化详解

前文参考：
[Python图像处理] 一.图像处理基础知识及OpenCV入门函数
[Python图像处理] 二.OpenCV+Numpy库读取与修改像素
[Python图像处理] 三.获取图像属性、兴趣ROI区域及通道处理
[Python图像处理] 四.图像平滑之均值滤波、方框滤波、高斯滤波及中值滤波
[Python图像处理] 五.图像融合、加法运算及图像类型转换
[Python图像处理] 六.图像缩放、图像旋转、图像翻转与图像平移
[Python图像处理] 七.图像阈值化处理及算法对比
[Python图像处理] 八.图像腐蚀与图像膨胀
[Python图像处理] 九.形态学之图像开运算、闭运算、梯度运算
[Python图像处理] 十.形态学之图像顶帽运算和黑帽运算
[Python图像处理] 十一.灰度直方图概念及OpenCV绘制直方图

前面的第六篇文章讲解了图像缩放、图像旋转、图像翻转和图像平移的几何变换，本篇文章主要讲解图像仿射变换和图像透视变换，通过Python调用OpenCV函数实。基础性知识希望对您有所帮助。

1.图像仿射变换
2.图像透视变换
3.基于图像透视变换的图像校正
4.图像几何变换总结

PS：文章参考自己以前系列图像处理文章及OpenCV库函数，同时，本篇文章涉及到《计算机图形学》基础知识，请大家下来补充。

参考文献：

Python下opencv使用笔记（三）（图像的几何变换）

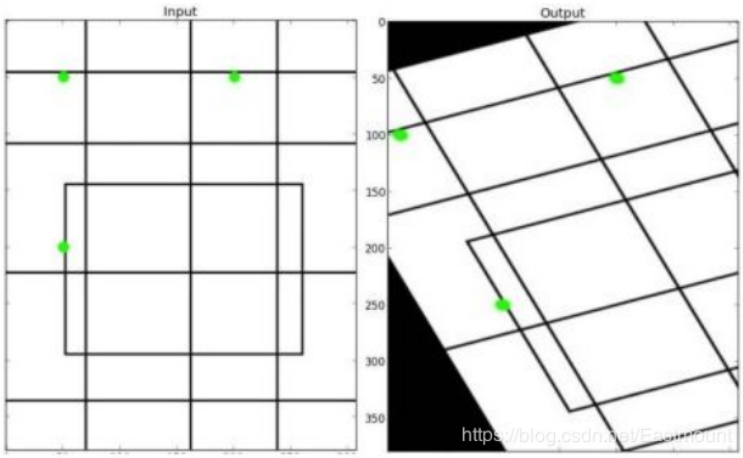
数字图像处理——图像的几何变换

图像校正-透视变换——t6_17

一.图像仿射变换

图像仿射变换又称为图像仿射映射，是指在几何中，一个向量空间进行一次线性变换并接上一个平移，变换为另一个向量空间。通常图像的旋转加上拉升就是图像仿射变换，仿射变换需要一个M矩阵实现，但是由于仿射变换比较复杂，很难找到这个M矩阵.

OpenCV提供了根据变换前后三个点的对应关系来自动求解M的函数——cv2.getAffineTransform(pos1,pos2)，其中pos1和pos2表示变换前后的对应位置关系，输出的结果为仿射矩阵M，接着使用函数cv2.warpAffine()实现图像仿射变换。图5-14是仿射变换的前后效果图。



图像仿射变换的函数原型如下：

M = cv2.getAffineTransform(pos1,pos2)

- pos1表示变换前的位置
- pos2表示变换后的位置

cv2.warpAffine(src, M, (cols, rows))

- src表示原始图像
- M表示仿射变换矩阵
- (rows,cols)表示变换后的图像大小，rows表示行数，cols表示列数

实现代码如下所示：

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取图片
src = cv2.imread('test.bmp')

#获取图像大小
rows, cols = src.shape[:2]

#设置图像仿射变换矩阵
pos1 = np.float32([[50,50], [200,50], [50,200]])
pos2 = np.float32([[10,100], [200,50], [100,250]])
M = cv2.getAffineTransform(pos1, pos2)

#图像仿射变换
result = cv2.warpAffine(src, M, (cols, rows))

#显示图像
cv2.imshow("original", src)
cv2.imshow("result", result)

#等待显示
cv2.waitKey(0)
cv2.destroyAllWindows()
```

输出效果图如下所示：

二.图像透视变换

图像透视变换（Perspective Transformation）的本质是将图像投影到一个新的视平面，同理OpenCV通过函数cv2.getPerspectiveTransform(pos1,pos2)构造矩阵M，其中pos1

和pos2分别表示变换前后的4个点对应位置。得到M后在通过函数cv2.warpPerspective(src,M,(cols,rows))进行透视变换。

图像透视变换的函数原型如下：

M = cv2.getPerspectiveTransform(pos1, pos2)

- pos1表示透视变换前的4个点对应位置
- pos2表示透视变换后的4个点对应位置

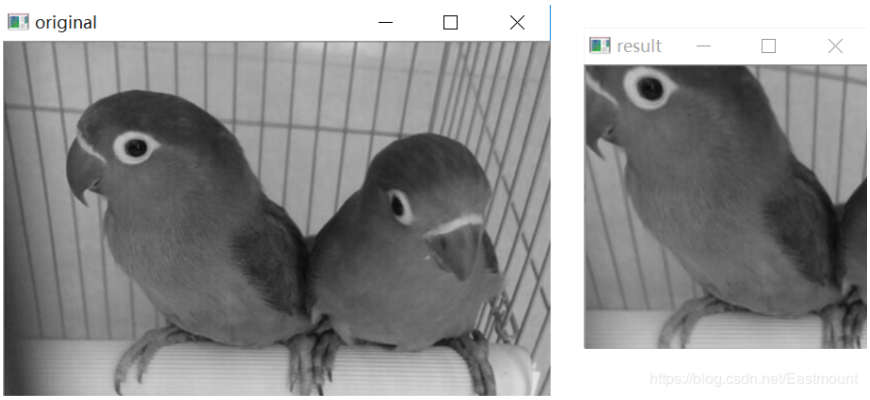
cv2.warpPerspective(src,M,(cols,rows))

- src表示原始图像
- M表示透视变换矩阵
- (rows,cols)表示变换后的图像大小，rows表示行数，cols表示列数

代码如下：

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取图片
src = cv2.imread('test01.jpg')

#获取图像大小
rows, cols = src.shape[:2]

#设置图像透视变换矩阵
pos1 = np.float32([[114, 82], [287, 156], [8, 322], [216, 333]])
pos2 = np.float32([[0, 0], [188, 0], [0, 262], [188, 262]])
M = cv2.getPerspectiveTransform(pos1, pos2)

#图像透视变换
result = cv2.warpPerspective(src, M, (190, 272))

#显示图像
cv2.imshow("original", src)
cv2.imshow("result", result)

#等待显示
cv2.waitKey(0)
cv2.destroyAllWindows()
```

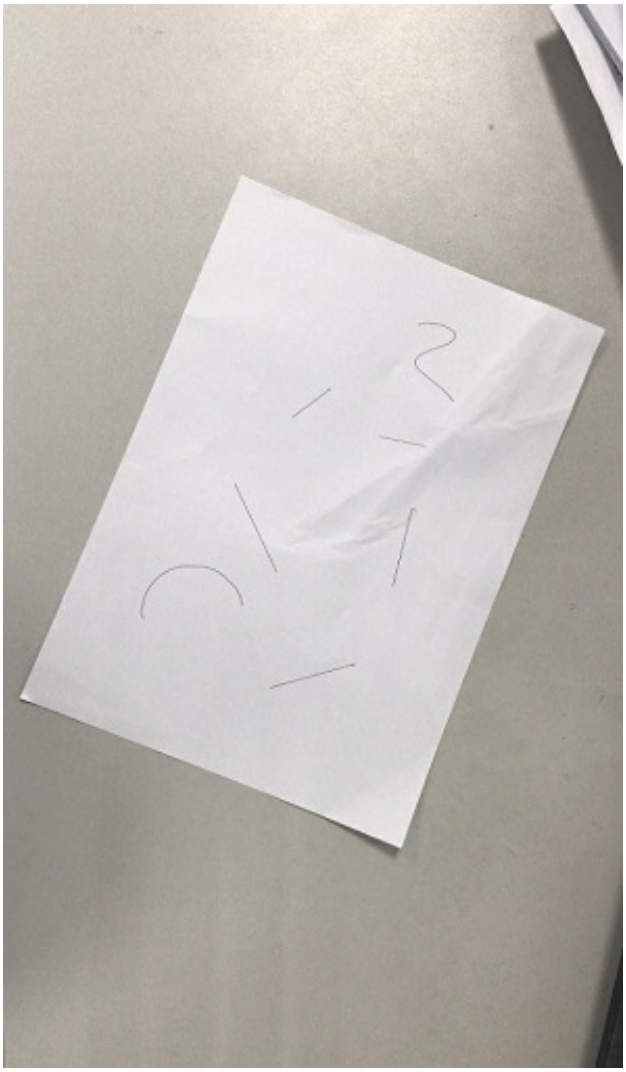
输出结果如下图所示：



三.基于图像透视变换的图像校正

下面参考 t6_17大神 的文章，通过图像透视变换实现图像校正功能。

假设现在存在一张A4纸图像，现在需要通过调用图像透视变换校正图像。



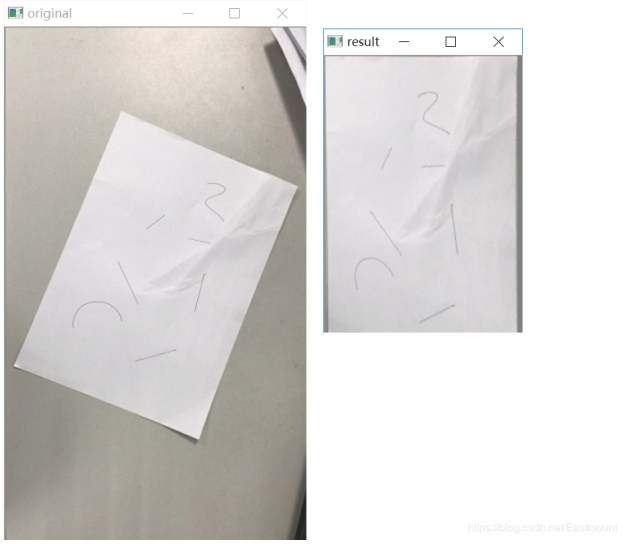
代码如下所示:

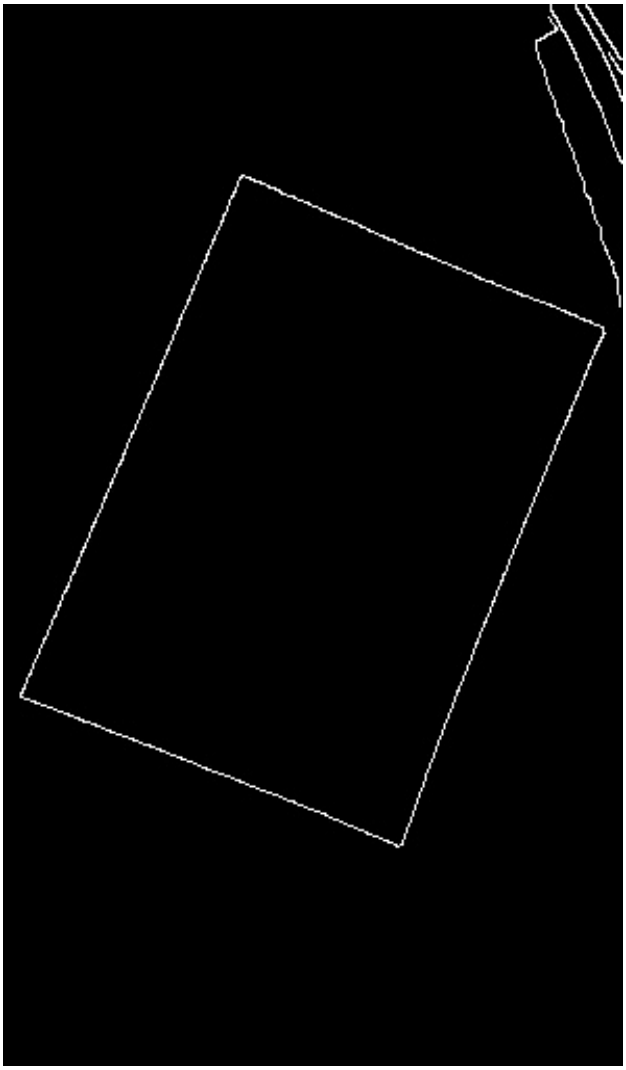
```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取图片
src = cv2.imread('test01.jpg')

#获取图像大小
rows, cols = src.shape[:2]

#将源图像高斯模糊
img = cv2.GaussianBlur(src, (3,3), 0)
#进行灰度化处理
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

#边缘检测（检测出图像的边缘信息）
edges = cv2.Canny(gray,50,250,apertureSize = 3)
```

```
cv2.imwrite("canny.jpg", edges)

#通过霍夫变换得到A4纸边缘
lines = cv2.HoughLinesP(edges,1,np.pi/180,50,minLineLength=90,maxLineGap=

#下面输出的四个点分别为四个顶点
for x1,y1,x2,y2 in lines[0]:
    print(x1,y1),(x2,y2)
for x1,y1,x2,y2 in lines[1]:
    print(x1,y1),(x2,y2)

#绘制边缘
for x1,y1,x2,y2 in lines[0]:
    cv2.line(gray, (x1,y1), (x2,y2), (0,0,255), 1)

#根据四个顶点设置图像透视变换矩阵
pos1 = np.float32([[114, 82], [287, 156], [8, 322], [216, 333]])
pos2 = np.float32([[0, 0], [188, 0], [0, 262], [188, 262]])
M = cv2.getPerspectiveTransform(pos1, pos2)

#图像透视变换
result = cv2.warpPerspective(src, M, (190, 272))

#显示图像
cv2.imshow("original", src)
cv2.imshow("result", result)

#等待显示
cv2.waitKey(0)
cv2.destroyAllWindows()
```

运行结果如下图所示:

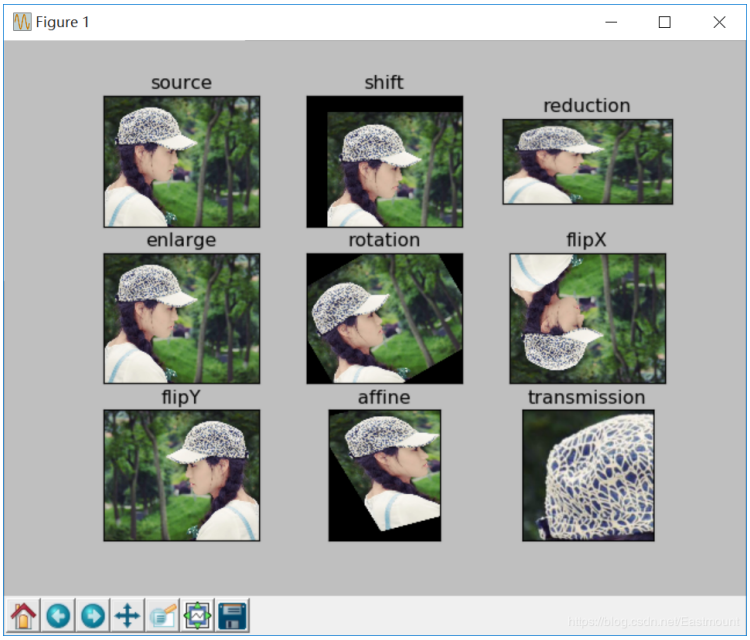
```
>>> ================================ RESTART ================
===
>>>
(8, 332) (114, 82)
(216, 333) (287, 156)
```





四.图像几何变换总结

最后补充图像几何代码所有变换，希望读者能体会下相关的代码，并动手实践下。输出结果以女神为例：



完整代码如下：

```
#encoding:utf-8
import cv2
import numpy as np
import matplotlib.pyplot as plt

#读取图片
img = cv2.imread('test3.jpg')
image = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

#图像平移矩阵
M = np.float32([[1, 0, 80], [0, 1, 30]])
rows, cols = image.shape[:2]
img1 = cv2.warpAffine(image, M, (cols, rows))

#图像缩小
img2 = cv2.resize(image, (200,100))

#图像放大
img3 = cv2.resize(image, None, fx=1.1, fy=1.1)

#绕图像的中心旋转
#源图像的高、宽 以及通道数
rows, cols, channel = image.shape
#函数参数：旋转中心 旋转度数 scale
M = cv2.getRotationMatrix2D((cols/2, rows/2), 30, 1)
```

```
#函数参数：原始图像 旋转参数 元素图像宽高
img4 = cv2.warpAffine(image, M, (cols, rows))

#图像翻转
img5 = cv2.flip(image, 0)   #参数=0以X轴为对称轴翻转
img6 = cv2.flip(image, 1)   #参数>0以Y轴为对称轴翻转

#图像的仿射
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])
M = cv2.getAffineTransform(pts1,pts2)
img7 = cv2.warpAffine(image, M, (rows,cols))

#图像的透射
pts1 = np.float32([[56,65],[238,52],[28,237],[239,240]])
pts2 = np.float32([[0,0],[200,0],[0,200],[200,200]])
M = cv2.getPerspectiveTransform(pts1,pts2)
img8 = cv2.warpPerspective(image,M,(200,200))


#循环显示图形
titles = [ 'source', 'shift', 'reduction', 'enlarge', 'rotation', 'flipX
images = [image, img1, img2, img3, img4, img5, img6, img7, img8]
for i in xrange(9):
   plt.subplot(3, 3, i+1), plt.imshow(images[i], 'gray')
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

希望文章对大家有所帮助，如果有错误或不足之处，还请海涵。最近连续奔波考博，经历的事情太多，有喜有悲，需要改变自己好好对家人，也希望读者与我一起加油。
（By：Eastmount 2019-03-20 早上12点 https://blog.csdn.net/Eastmount/）