

图像处理基础

（作业2）



学 院（系）： 电子信息与电气工程学部

专 业： 计算机科学与技术

班 级： 电计1704

学 生 姓 名： 谢玉宁

学 号： 201792260

完 成 日 期： 2020年3月24日

实验目标：实现图像旋转

实现原理:

图像旋转是指图像按照某个位置转动一定角度的过程，旋转中图像仍保持这原始尺寸。图像旋转后图像的水平对称轴、垂直对称轴及中心坐标原点都可能会发生变换，因此需要对图像旋转中的坐标进行相应转换。

假设图像逆时针旋转θ，则根据坐标转换可得旋转转换为：

首先定义Xrot函数，输入原图片点的集合，通过计算并输出旋转后点的集合。

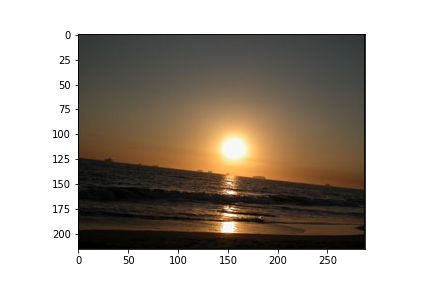
再定义Irot函数，输入原图片，将原图片坐标变换为以图片正中心为原点的笛卡尔坐标系，并将图片灰度值记录在lvector中，将新坐标点集输入Xrot得到旋转后图片的坐标集合。

然后通过坐标集合中x,y坐标最值得到新图片输出的尺寸，并用meshgrid得到输出图片的坐标网格，在通过griddata向输出图像中填充旋转后的灰度图像，得到输出图像。

最后定义show\_rot函数通过matplotlib.pyplot中的ginput函数获得点击图像的两个像素点，根据两个像素坐标由公式计算得到两点连成直线的斜率，进而得到与水平面的旋转角，再用Irot函数输出新的图像，是两点连线呈水平线。

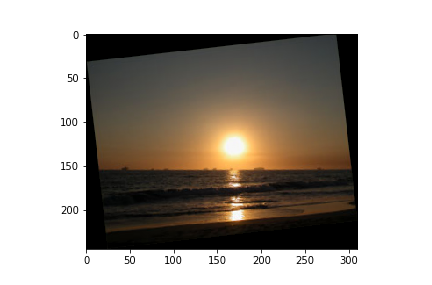
输入与输出对比:

输入图像：

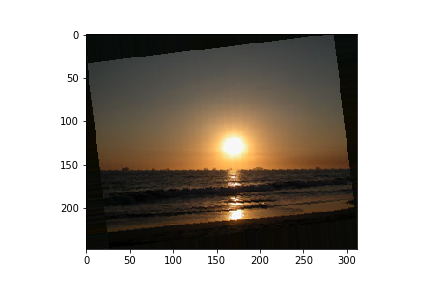


图片 1输入原始图像

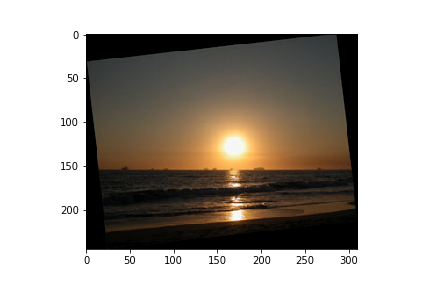
输出图像：



图片 2 旋转图像（linear插值）



图片 3 旋转图像（nearest插值）



图片 4 旋转图像（cubic插值）

结果分析:

实验最终成功实现图片旋转的操作。旋转后图像较原图稍有失真。

在图像旋转测试中，分别采用3种不同的插值方法——nearest，linear，cubic。不同的插值方法得到的图像效果有肉眼可见的差异，

Nearest是根据每个像素最近的像素灰度值（RGB值）得到该像素的灰度值（RGB值），因此用该方法插值后的得到的图像失真较严重，不仅在图形“边缘”产生了锯齿状，还在图像外部黑色区域出现了彩色条带，是由黑边像素与距离黑色边最近的图像像素灰度一致。

Linear插值法相当于之前实现的bilinear，插值后得到的图像效果较nearest好得多，图像的边缘变得非常平滑，外部黑色区域也没有出现条带。因此旋转效果非常好。

Cubic插值法是一种非线性插值，如果将图片放大来看，它得到的图像效果最好，颜色过渡非常平滑舒适，但是该方法计算量更大，因此得到旋转图像所需时间更长，对于实时性较高的图像处理任务该方法较为不实用。

代码：

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 24 15:30:02 2020

@author: Lenovo

"""

from scipy.interpolate import griddata

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as pli

im1\_path = r'C:\Users\Lenovo\Desktop\crooked\_horizon.jpg'

im1 = pli.imread(im1\_path)

def Xrot(X, angle):

anglePi = angle \* np.pi / 180.0

cos = np.cos(anglePi)

sin = np.sin(anglePi)

Rot\_M = np.array([[cos,-sin],

[sin, cos]])

Y = np.matmul(Rot\_M,X)

return Y

def Irot(Image,Angle):

h = Image.shape[0]

w = Image.shape[1]

x = np.linspace(h/2-0.5,-h/2+0.5,h)

y = np.linspace(-w/2+0.5,w/2-0.5,w)

xcoord,ycoord = np.meshgrid(y,x)

X = np.concatenate((xcoord.flatten()[np.newaxis,:],ycoord.flatten()[np.newaxis,:]),axis = 0)

lvector = Image.reshape(Image.shape[0]\*Image.shape[1],Image.shape[2])

Y = Xrot(X,Angle)

scale\_x = int(np.floor(Y[0].min())),int(np.ceil(Y[0].max()))

scale\_y = int(np.floor(Y[1].min())),int(np.ceil(Y[1].max()))

x\_range,y\_range = scale\_x[1]-scale\_x[0],scale\_y[1]-scale\_y[0]

x = np.linspace(x\_range/2-0.5,-x\_range/2+0.5,x\_range)

y = np.linspace(-y\_range/2+0.5,y\_range/2-0.5,y\_range)

new\_xcoord,new\_ycoord = np.meshgrid(-x,-y)

grid\_z = griddata(Y.T, lvector/255, (new\_xcoord,new\_ycoord), method=‘’linear’)

plt.imshow(grid\_z)

return grid\_z

def show\_rot(Image):

I = Image.copy()

#while True:

plt.imshow(I)

pos = plt.ginput(2)

angle = np.arctan((pos[0][1]-pos[1][1])/(pos[0][0]-pos[1][0]))

print(angle)

angle=angle\*180/np.pi

print(angle)

if((pos[0]<pos[1]).all() or (pos[0]>pos[1]).all()):

Irot(I,angle)

else:

Irot(I,-angle)

#rot(im1,60)

show\_rot(im1)