第一次作业

学生： 彭相阳

班级：自动化钱 61

学号： 2160800060

提交日期： 2019.03.04

摘要：

首先对bmp图片格式进行了初步的了解；然后完成了改变图片的灰度级并显示的功能：将原始图片lena的8灰度级依次降低为7-1；计算了图片lena的平均值和方差；理解并实现了将图片放大过程中的仿射变换以及灰度内插的三种方法：最近邻、双线性、双三次；并在此基础上进一步完成了水平错切和旋转的仿射变换与zoom的叠加实现

1. Bmp图像格式简介,以7.bmp为例说明

BMP文件格式，又称为Bitmap（位图）或是DIB(Device-Independent Device，设备无关位图)，是Windows系统中广泛使用的图像文件格式。

BMP文件的数据按照从文件头开始的先后顺序分为四个部分：

Ø bmp文件头(bmp file header)：提供文件的格式、大小等信息

Ø 位图信息头(bitmap information)：提供图像数据的尺寸、位平面数、压缩方式、颜色索引等信息

Ø 调色板(color palette)：可选，如使用索引来表示图像，调色板就是索引与其对应的颜色的映射表

Ø 位图数据(bitmap data)：图像数据

使用python读取bmp格式的图像数据可以使用函数 cv2.imread(),返回一个numpy的array数组，比如7.bmp读取后得到如下数组：

array([[[ 82, 82, 82], [ 82, 82, 82], [ 73, 73, 73], [ 59, 59, 59], [ 55, 55, 55], [ 80, 80, 80], [ 90, 90, 90]],

[[ 97, 97, 97], [ 89, 89, 89], [ 90, 90, 90], [ 95, 95, 95], [ 71, 71, 71],

[ 40, 40, 40], [ 69, 69, 69]],

[[104, 104, 104], [ 71, 71, 71], [ 63, 63, 63], [105, 105, 105], [ 93, 93, 93],

[ 76, 76, 76], [ 42, 42, 42]],

[[ 88, 88, 88], [ 75, 75, 75], [ 85, 85, 85], [101, 101, 101], [ 90, 90, 90],

[ 91, 91, 91], [ 70, 70, 70]],

[[ 97, 97, 97], [ 92, 92, 92], [ 91, 91, 91], [ 99, 99, 99], [ 72, 72, 72],

[ 71, 71, 71], [ 82, 82, 82]],

[[ 98, 98, 98], [101, 101, 101], [102, 102, 102], [ 86, 86, 86], [ 69, 69, 69],

[ 71, 71, 71], [ 95, 95, 95]],

[[103, 103, 103], [ 99, 99, 99], [100, 100, 100], [ 84, 84, 84], [ 86, 86, 86],

[ 98, 98, 98], [ 98, 98, 98]]], dtype=uint8)

这是一个7\*7\*3的矩阵，代表了图像中7\*7个像素的三个通道的值。

2. 把lena 512\*512图像灰度级逐级递减8-1显示

将图像灰度级递减显示可以发现，伪轮廓开始出现，图像的质量降低

原图：



处理结果如下：







3. 计算lena图像的均值方差

均值：99.05

方差：2796.03

4. 把lena图像用近邻、双线性和双三次插值法zoom到2048\*2048

从近邻到双线性到双三次，在直边缘的处理和保持细节方面的效果逐渐增强

近邻 双线性 双三次

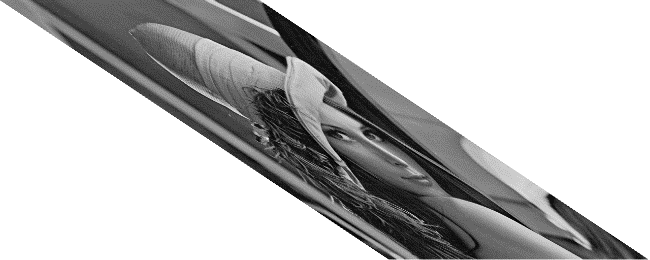


5. 把lena和elain图像分别进行水平shear（参数可设置为1.5，或者自行选择）和旋转30度，并采用用近邻、双线性和双三次插值法zoom到2048\*2048

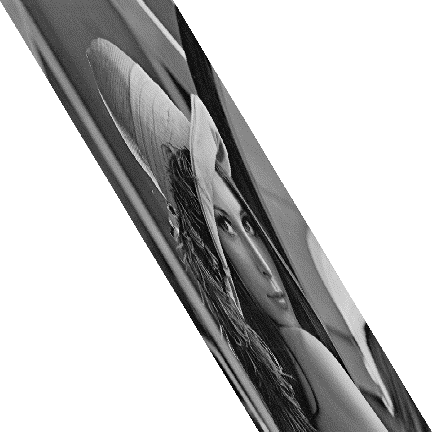
Lena：

近邻插值法：

水平shear:

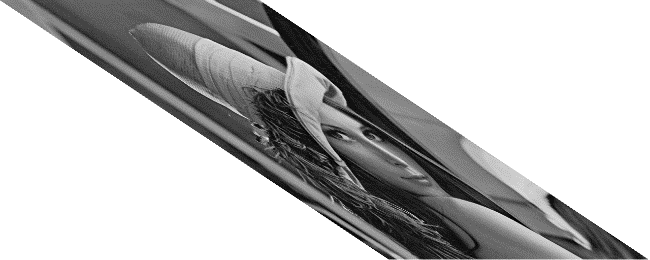


Zoom到2048\*2048：

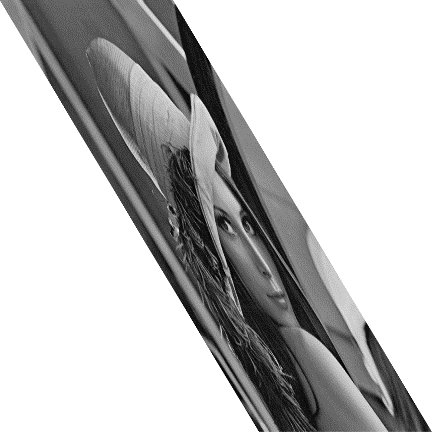


双线性插值法：

水平shear:

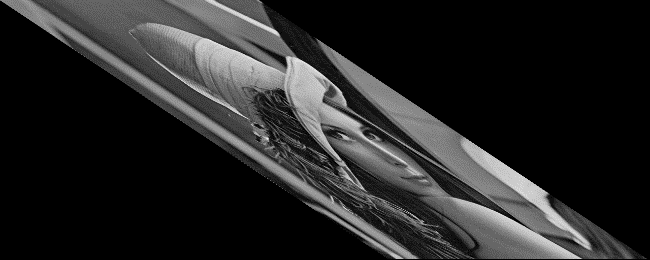


Zoom到2048\*2048：

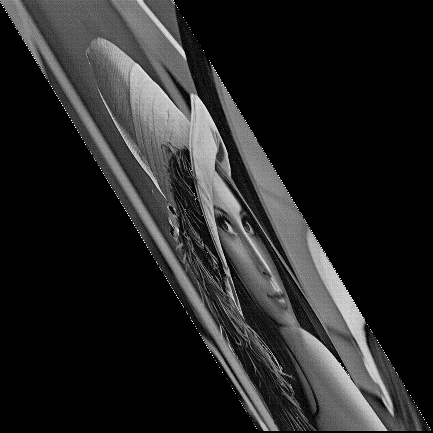


双三次插值法：

水平shear:



Zoom到2048\*2048：



Elain

近邻插值法：

旋转30度 Zoom到2048\*2048：

双线性插值法

旋转30度 Zoom到2048\*2048：

双三次插值法

旋转30度 Zoom到2048\*2048：

附录一：源代码

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

"""

Created on Tue Feb 26 22:02:06 2019

@author: Administrator

"""

from numba import jit

from tqdm import tqdm #显示迭代进度

import math

import cv2 as cv

import numpy as np

from numpy import \*

def calMeanVar(img):

'''计算图片的均值方差

Args:

img:图片

'''

print("mean = ",np.mean(img))

print("var = ",np.var(img) )

def changeGrayValue(img,n,name):

'''改变图片的灰度值

Args:

img:原始图片

n :目标灰度级 0<i<8

name:新图片名字

'''

i=8-n

img=(img/2\*\*i).astype(np.uint8)

img=img\*2\*\*i

cv.imshow(str(n),img)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+str(n)+".bmp",img)

def W(x):

'''BiCubic函数

Args:

x

Return:

W(x)

'''

a=-0.5

if abs(x)<=1:

return (a+2)\*abs(x)\*\*3-(a+3)\*abs(x)\*\*2+1

else:

if abs(x)<2:

return a\*abs(x)\*\*3-5\*a\*abs(x)\*\*2+8\*a\*abs(x)-4\*a

else:

return 0

#@jit

#python 中的函数参数传递是 值传递和引用传递 的综合

#增加一层函数调用之后运算速度变慢是怎么回事呢

def nearestInterplotation(img,target,T):

'''最近邻灰度内插

Args:

img :原始图片

target:目标图片

T :仿射矩阵

'''

(X,Y,Z) = img.shape

(x,y,Z) = target.shape

\_T = T.I

for i in tqdm(range(x)):

for j in range(y):

xy = mat([i,j,1])\*\_T

ii = xy[0,0]

jj = xy[0,1]

I = int(ii)

U = ii-I

J = int(jj)

V = jj-J

#将 dis 的计算提到 k 循环外减少工作量加快速度

if 0<=I<X and 0<=J<Y:

dis = np.asarray([math.sqrt(U\*\*2+V\*\*2),math.sqrt(U\*\*2+(1-V)\*\*2),math.sqrt((1-U)\*\*2+V\*\*2),math.sqrt((1-U)\*\*2+(1-V)\*\*2)])

for k in range(Z):

try:

value = np.asarray([img[I,J,k],img[I,J+1,k],img[I+1,J,k],img[I+1,J+1,k]])

target[i,j,k] = value[np.argmin(dis)]

except:

target[i,j,k] = 255

else:

target[i,j] = [255,255,255]

def bilinearityInterplotation(img,target,T):

'''双线性灰度内插

Args:

img :原始图片

target:目标图片

T :仿射矩阵

'''

(X,Y,Z) = img.shape

(x,y,Z) = target.shape

\_T = T.I

for i in tqdm(range(x)):

for j in range(y):

xy = mat([i,j,1])\*\_T

ii = xy[0,0]

jj = xy[0,1]

I = int(ii)

U = ii-I

J = int(jj)

V = jj-J

if 0<=I<X and 0<=J<Y:

weight = [(1-U)\*(1-V),(1-U)\*V,U\*(1-V),U\*V]

for k in range(Z):

try:

target[i,j,k] = weight[0]\*img[I,J,k]+weight[1]\*img[I,J+1,k]+weight[2]\*img[I+1,J,k]+weight[3]\*img[I+1,J+1,k]

except:

target[i,j,k] = 255

else:

target[i,j] = [255,255,255]

def bicubicInterplotation(img,target,T):

'''双三次灰度内插

Args:

img :原始图片

target:目标图片

T :仿射矩阵

'''

(X,Y,Z) = img.shape

(x,y,Z) = target.shape

\_T = T.I

for i in tqdm(range(x)):

for j in range(y):

xy = mat([i,j,1])\*\_T

ii = xy[0,0]

jj = xy[0,1]

I = int(ii)

U = ii-I

J = int(jj)

V = jj-J

if 0<=I<X and 0<=J<Y:

a = np.asarray([[W(1+U),W(U),W(1-U),W(2-U)],[W(1+V),W(V),W(1-V),W(2-V)]])

for k in range(1):#3个通道要半小时 word天

try:

value = np.asarray([[img[I-1,J-1,k],img[I-1,J,k],img[I-1,J+1,k],img[I-1,J+2,k]],

[img[I,J-1,k],img[I,J,k],img[I,J+1,k],img[I,J+2,k]],

[img[I+1,J-1,k],img[I+1,J,k],img[I+1,J+1,k],img[I+1,J+2,k]],

[img[I+2,J-1,k],img[I+2,J,k],img[I+2,J+1,k],img[I+2,J+2,k]]])

target[i,j,k] = 0

for m in range(4):

for n in range(4):

target[i,j,k] += a[0,m]\*a[1,n]\*value[m,n]

except:

target[i,j,k] = 0

target[i,j,1]=target[i,j,0]

target[i,j,2]=target[i,j,0]

else:

#等于255的话会导致条纹 就算是纯白的图片这样放大后也会出现杂点 应该是权重设计的不好

target[i,j] = [0,0,0]

def enlargePicture(img,x,y,name,method="all"):

'''放大图片

Args:

img:原始图片

x :x边放大后的像素

y :y边放大后的像素

name:新图片名字

method:插值方法 "all" "nearest" "bilinearity" "bicubic"

'''

if method not in ["all","nearest","bilinearity","bicubic"]:

raise Exception("method must be one of [all,nearest,bilinearity,bicubic]")

(X,Y,Z) = img.shape

target = np.empty([x,y,Z],dtype=np.uint8)

x\_rate = x/X

y\_rate = y/Y

T=mat([[x\_rate,0,0],[0,y\_rate,0],[0,0,1]])

if method == "all" or method == "nearest":

nearestInterplotation(img,target,T)

cv.imshow("nearest",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"nearest.bmp",target)

if method == "all" or method == "bilinearity":

bilinearityInterplotation(img,target,T)

cv.imshow("bilinearity",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"bilinearity.bmp",target)

if method == "all" or method == "bicubic":

bicubicInterplotation(img,target,T)

cv.imshow("bicubic",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"bicubic.bmp",target)

def horiShearPicture(img,a,name,method="all"):

'''水平错切图片

Args:

img:原始图片

a :错切参数

name:新图片名字

method:插值方法 "all" "nearest" "bilinearity" "bicubic"

'''

if method not in ["all","nearest","bilinearity","bicubic"]:

raise Exception("method must be one of [all,nearest,bilinearity,bicubic]")

(X,Y,Z) = img.shape

x=X

y=int(Y+X\*a)

target = np.empty([x,y,Z],dtype=np.uint8)

T=mat([[1,a,0],[0,1,0],[0,0,1]])

if method == "all" or method == "nearest":

nearestInterplotation(img,target,T)

cv.imshow("nearest",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"nearest.bmp",target)

if method == "all" or method == "bilinearity":

bilinearityInterplotation(img,target,T)

cv.imshow("bilinearity",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"bilinearity.bmp",target)

if method == "all" or method == "bicubic":

bicubicInterplotation(img,target,T)

cv.imshow("bicubic",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"bicubic.bmp",target)

def rotatePicture(img,a,name,method="all"):

'''旋转图片

Args:

img:原始图片

a :旋转角度

name:新图片名字

method:插值方法 "all" "nearest" "bilinearity" "bicubic"

'''

if method not in ["all","nearest","bilinearity","bicubic"]:

raise Exception("method must be one of [all,nearest,bilinearity,bicubic]")

(X,Y,Z) = img.shape

(x1,y1) = (X\*cos(a)-Y\*sin(a),X\*sin(a)+Y\*cos(a))

(x2,y2) = (X\*cos(a),X\*sin(a))

(x3,y3) = (Y\*sin(a)\*(-1),Y\*cos(a))

x = int(max(abs(x1),abs(x2-x3)))

y = int(max(abs(y1),abs(y2-y3)))

dx= min(0,x1,x2,x3)\*-1

dy= min(0,y1,y2,y3)\*-1

target = np.empty([x,y,Z],dtype=np.uint8)

T=mat([[math.cos(a),math.sin(a),0],[math.sin(a)\*(-1),math.cos(a),0],[0,0,1]])\*mat([[1,0,0],[0,1,0],[dx,dy,1]])

if method == "all" or method == "nearest":

nearestInterplotation(img,target,T)

cv.imshow("nearest",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"nearest.bmp",target)

if method == "all" or method == "bilinearity":

bilinearityInterplotation(img,target,T)

cv.imshow("bilinearity",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"bilinearity.bmp",target)

if method == "all" or method == "bicubic":

bicubicInterplotation(img,target,T)

cv.imshow("bicubic",target)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("./"+name+"bicubic.bmp",target)

'''

tt=np.ones([100,100,3],dtype=np.uint8)\*255

cv.imshow("tt",tt)

cv.waitKey(0)

cv.destroyAllWindows()

newname=tt

enlargePicture(tt,2048,2048,newname,"bicubic")

'''

imgname = "elainRotatebicubic"

lena = cv.imread("./"+imgname+".bmp")

#calMeanVar(lena)

'''

newname = "lenaG"

for i in range(7):

changeGrayValue(lena,7-i,newname)

'''

newname = "elainRZ"

enlargePicture(lena,2048,2048,newname,"bicubic")

#enlargePicture(lena,2048,2048,"nearest")

#enlargePicture(lena,2048,2048,"bilinearity")

#enlargePicture(lena,2048,2048,"bicubic")

'''

newname = "NewlenaShear"

horiShearPicture(lena,1.5,newname,"bicubic")

'''

#horiShearPicture(lena,1.5,"nearest")

#horiShearPicture(lena,1.5,"bilinearity")

#horiShearPicture(lena,1.5,"bicubic")

'''

newname = "elainRotate"

angle = pi/6

rotatePicture(lena,angle,newname,"all")

'''

#rotatePicture(lena,1,"nearest")

#rotatePicture(lena,1,"bilinearity")

#rotatePicture(lena,1,"bicubic")