****

计算机视觉实验

**实验一**

**专 业：**

**班 级：**

**姓 名：**

**学 号：**

**指导教师：**

**成 绩：**

**计算机学院**

**年 月 日**

# 实验目的

安装、配置、熟悉Python+Opencv开发环境

图像读、写等的基本操作

选取一张自己的生活照，实现：

直方图归一化和均衡化（基础实现，如果实现了自适应，加分）

快速傅里叶变换和高斯滤波（使用不同的sigma,分析sigma大小对图像清晰度的影响，至少使用三种不同的sigma）。

# 实验过程

Pycharm终端输入pip opencv后却不能编译有opencv函数的代码

电脑上原来有个python环境，编译选项是原来的环境，但opencv库安装在conda

设置里把编译器改成conda

## 直方图归一化和均衡化

归一化

实验图片

所用代码

import cv2

import numpy as np

import matplotlib.pyplot as plt

def hist\_normalization(img, a=0, b=255):

c = img.min()

d = img.max()

out = img.copy()

# normalization

out = (b - a) / (d - c) \* (out - c) + a

out[out < a] = a

out[out > b] = b

out = out.astype(np.uint8)

return out

if \_\_name\_\_ == '\_\_main\_\_':

img =cv2.imread('img/in.jpeg').astype(np.uint8)

arr=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

shape=arr.shape

# 因为手机拍摄照片灰度范围是0~255，直接归一化没有效果，所以修改像素灰度值来缩小灰度范围，使效果明显

for i in range(0,shape[0]):

for j in range(0,shape[1]):

if arr[i,j]<50:

arr[i,j]\*=1.5

elif arr[i,j]>200:

arr[i,j]\*=0.8

out=hist\_normalization(arr)

plt.hist(out.ravel(),bins=255,rwidth=0.8,range=(0,255))

plt.savefig('img/out3.png')

cv2.imwrite("img/out.jpg",out)

实验效果

代码分析及现象

直方图相比原来，被拉开的更散

均衡化

实验图片

所用代码

import cv2

import numpy as np

import matplotlib.pyplot as plt

if \_\_name\_\_ == '\_\_main\_\_':

img =cv2.imread('img/in.jpeg').astype(np.uint8)

img=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

equalization = cv2.equalizeHist(img)

cv2.imshow('Equalization', equalization)

cv2.waitKey(0)

实验效果

代码分析及现象

图片灰度变得更均匀

## 快速傅里叶变换和高斯低通滤波

快速傅里叶变化

实验图片

所用代码

import cv2

import numpy as np

if \_\_name\_\_ == '\_\_main\_\_':

img =cv2.imread('img/in.jpeg',cv2.IMREAD\_GRAYSCALE)

# 2. 进行快速傅里叶变换，数据类型需转换为单精度

fft = np.fft.fft2(np.float32(img))

# 3. 频谱中心化

fft\_shift = np.fft.fftshift(fft)

# 4. 灰度值缩放，便于显示

fft\_shift = 15 \* np.log(np.abs(fft\_shift))

# 5. 数据类型转换

fft\_shift = fft\_shift.astype(np.uint8)

# 6. 显示结果

cv2.imwrite('img/out.jpg', fft\_shift)

cv2.imshow('Image', img)

cv2.imshow('FFT', fft\_shift)

cv2.waitKey(0)

实验效果



代码分析及现象

原图变成了中间一个光源向四处发散的图像

高斯低通滤波

实验图片

所用代码

import numpy as np

import cv2

import numpy.random as random

def add\_salt\_noise(image, prob=0.1):

output = np.zeros(image.shape, np.uint8)

for i in range(image.shape[0]):

for j in range(image.shape[1]):

if random.random() < prob:

if random.random() < 0.5:

output[i, j] = 0

else:

output[i, j] = 255

else:

output[i, j] = image[i, j]

return output

if \_\_name\_\_ == '\_\_main\_\_':

# 1. 读取图像文件

image\_path = 'img/in.jpeg'

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

# 2. 添加椒盐噪声

noise = add\_salt\_noise(image, prob=0.1)

# 3. 高斯滤波

kernel\_size = 3

gaussian = cv2.GaussianBlur(noise, (kernel\_size, kernel\_size), sigmaX=1)

for i in range(0,9):

gaussian = cv2.GaussianBlur(gaussian, (kernel\_size, kernel\_size), sigmaX=1)

# 4. 显示图像

cv2.imshow('Image', image)

cv2.imshow('Noise', noise)

cv2.imshow('Gaussian', gaussian)

cv2.waitKey(0)

实验效果

Sigma=1

Sigma=0.1

Sigma=0.5

代码分析及现象

随着sigma增大，去噪效果增强，不过sigma在1之后继续增大时效果不明显

# 实验心得

熟悉了opencv的基本操作，深入得理解了怎样对图片处理使其质量更好，对于其中出现的问题也能通过bing等搜索引擎找到答案