# -\*- coding:utf-8 -\*-  
# demo.py  
# zhengyinloong  
# 2023/04/09 04:50  
  
import numpy as np  
import matplotlib.pyplot as plt  
import cv2 as cv  
  
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['axes.unicode\_minus'] = False  
  
# 读取图像  
imgFile = 'img2.jpg'  
img1 = cv.imread(imgFile, cv.IMREAD\_COLOR)  
img2 = cv.imread(imgFile, cv.IMREAD\_GRAYSCALE)  
H, W, D = np.shape(img1)  
# print(W, H)  
  
# 缩放  
w = int(W / 4)  
h = int(H / 4)  
img3 = cv.resize(img1, (w, h))  
img4 = cv.resize(img2, (w, h))  
  
# 显示缩放后图像  
cv.imshow('Color', img3)  
cv.imwrite('Color.jpg', img3)  
cv.imshow('Gray', img4)  
cv.imwrite('Gray.jpg', img4)  
  
# 直方图  
hist = np.zeros(256, dtype=np.uint64)  
for i in range(0, 256):  
 hist[i] = np.sum(img4 == i)  
plt.bar(range(0, 256), hist)  
plt.title('直方图')  
plt.show()  
  
# 归一化  
s = np.sum(hist)  
hist\_nor = hist / s  
  
# 累计直方图  
hist\_cum = np.copy(hist\_nor)  
for i in range(1, 256):  
 hist\_cum[i] = hist\_cum[i - 1] + hist\_cum[i]  
plt.bar(range(0, 256), hist\_cum)  
plt.title('累计直方图')  
plt.show()  
  
# 均衡直方图  
hist\_eq = np.round(hist\_cum \* 255)  
# plt.bar(range(0,256),hist\_eq)  
plt.bar(hist\_eq, hist)  
plt.title('均衡后直方图')  
plt.show()  
  
# 灰度映射  
img5 = np.zeros((h, w), dtype=np.uint8)  
for i in range(0, 256):  
 k = img4 == i  
 img5[k] = hist\_eq[i]  
# 方法二  
# img5 = np.copy(img4)  
# for i, line in enumerate(img4):  
# for j, pixel in enumerate(line):  
# img5[i][j] = zft\_list\_jh[img4[i][j]]  
  
# 显示  
cv.imshow('Image after equalization', img5)  
cv.imwrite('ImageAfterEqualization.jpg', img5)  
  
# 平滑滤波  
filter1 = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]]) # 均值滤波  
filter2 = np.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]]) # 高斯滤波  
filter3 = np.array([[3, 5, 3], [5, 8, 5], [3, 5, 3]])  
# filter3 = np.array([[0.09474166, 0.11831801, 0.09474166],  
# [0.11831801, 0.14776132, 0.11831801],  
# [0.09474166, 0.11831801, 0.09474166]])  
  
def imgFiltering(img, myfilter):  
 *"""  
 滤波函数（忽略边界）  
  
 :param img: 图像  
 :param myfilter: 滤波器  
 :return: 滤波后图像  
 """* newimg = np.copy(img)  
 img\_h, img\_w = np.shape(img)  
 myfilter = myfilter / np.sum(myfilter)  
  
 for i, line in enumerate(img):  
 for j, pixel in enumerate(line):  
 if (1 < i < img\_h - 1) and (1 < j < img\_w - 1):  
 newimg[i, j] = np.sum(img[i - 1:i + 2, j - 1:j + 2] \* myfilter)  
  
 return newimg  
  
  
img6 = imgFiltering(img4, filter1)  
img7 = imgFiltering(img4, filter2)  
img8 = imgFiltering(img4, filter3)  
cv.imshow('Filtered image 1', img6)  
cv.imwrite('FilteredImage1.jpg', img6)  
cv.imshow('Filtered image 2', img7)  
cv.imwrite('FilteredImage2.jpg', img7)  
cv.imshow('Filtered image 3', img8)  
cv.imwrite('FilteredImage3.jpg', img8)  
  
# cv函数滤波  
img9 = cv.blur(img4, (3, 3))  
img10 = cv.GaussianBlur(img4, (3, 3), 1)  
# cv.imshow('Filtered image 4',img9)  
cv.imwrite('FilteredImage4.jpg', img9)  
# cv.imshow('Filtered image 5',img10)  
cv.imwrite('FilteredImage5.jpg', img10)  
  
cv.waitKey(0)  
cv.destroyAllWindows()