《数字图像处理》期末课程设计

姓名：李志成

学号：2022190503030

题目：

1. **将图像的灰度级分辨率调整至128，64，32，16，8，4，2 ，并在同一个figure窗口上将它们显示出来。**

思路：假设原灰度级分辨率为256，若将其调整为m，则意味着将256个灰度级分为m份，每份长度为。假设第（i，j）的像素灰度值为x，改变后为 ，则：

其中，，为相邻每份的灰度差值。

python代码：

|  |
| --- |
| from skimage import io import matplotlib.pyplot as plt  img\_path = 'img/lena.bmp' img = io.imread(img\_path)  # convert the greyscale to certain value def convert\_greyscale\_to(img, greyscale):  interval = 256 // greyscale  delta\_grey = 255 // (greyscale - 1)  rows, cols = img.shape  for i in range(rows):  for j in range(cols):  img[i][j] = (img[i][j] // interval) \* delta\_grey   return img   plt.figure(figsize=(8, 4))  # draw the oiginal image plt.subplot(2, 4, 1) plt.title('Original Image') plt.axis('off') io.imshow(img)  for i, value in enumerate([128, 64, 32, 16, 8, 4, 2]):  plt.subplot(2, 4, i+2)  plt.title('{}'.format(value))  plt.axis('off')  out = convert\_greyscale\_to(img, value)  io.imshow(out)  plt.tight\_layout() plt.show() |



图表 1原图与转化后图像

1. **往图像中叠加不同类型的噪声，并设计一个频域低通滤波器来去除之**

思路：利用skimage的内置函数分别加入gaussian, localvar, poisson, salt, pepper, s&p, speckle噪声，最后再用截止频率比例为0.05、阶数为1的Butterworth滤波器进行低通滤波：

|  |
| --- |
| import skimage import matplotlib.pyplot as plt from skimage.filters import butterworth  img\_path = 'img/lena.bmp' img = skimage.io.imread(img\_path)  noise = skimage.util.random\_noise(img, mode='gaussian', var=0.01) noise\_mode = ['gaussian', 'localvar', 'poisson', 'salt', "pepper", 's&p', 'speckle']  plt.figure(figsize=(8, 8)) plt.subplot(4, 4, 1) plt.title('Original Image') skimage.io.imshow(img) plt.axis('off')  for i, mode in enumerate(noise\_mode):  plt.subplot(4, 4, i+2)  plt.title('{}'.format(mode))  plt.axis('off')  noise = skimage.util.random\_noise(img, mode=mode)  skimage.io.imshow(noise)   plt.subplot(4, 4, i + 10)  plt.title('{}+LPF'.format(mode))  plt.axis('off')  low\_pass = butterworth(noise, 0.05, False, 1, channel\_axis=-1)  skimage.io.imshow(low\_pass)   plt.tight\_layout() plt.show() |



图表 2原图、插入噪声后与加上低通滤波后的图

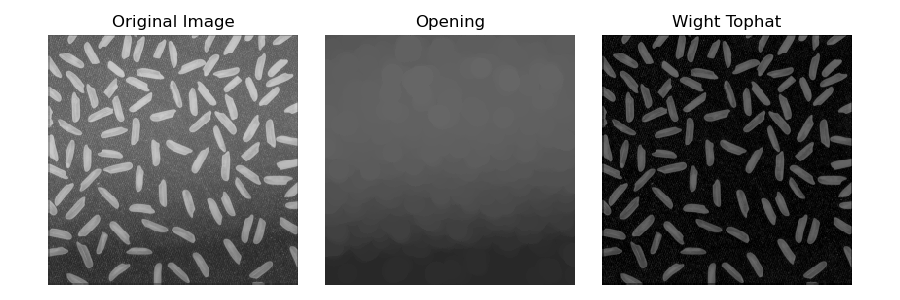
1. **举例说明顶帽变换在图像阴影校正方面的应用。**

思路：顶帽变换可以用于暗背景上的亮物体，以去除掉非均匀光照下的背景从而分离出目标主体。

顶帽变化=原图像-原图像的开操作

在本题目中，开操作可以理解为一个半径为20的球紧贴原图像所构成的平面的下表面滚动，此时球体任何部分所能达到的最低点即为开操作的平面。在与原图像相减之后，背景不均匀的光照就可以被去除了。

|  |
| --- |
| import skimage from skimage import morphology, io import matplotlib.pyplot as plt  img\_path = 'img/rice.bmp' img = skimage.io.imread(img\_path)  plt.figure(figsize=(9, 3))  plt.subplot(1, 3, 1) plt.title('Original Image') plt.axis('off') io.imshow(img)  se = morphology.disk(20) opening = morphology.opening(img, se) white\_tophat = img - opening  plt.subplot(1, 3, 2) plt.title('Opening') plt.axis('off') io.imshow(opening)  plt.subplot(1, 3, 3) plt.title('Wight Tophat') plt.axis('off') io.imshow(white\_tophat)  plt.tight\_layout() plt.show() |



图表 3原图、开操作与顶帽运算

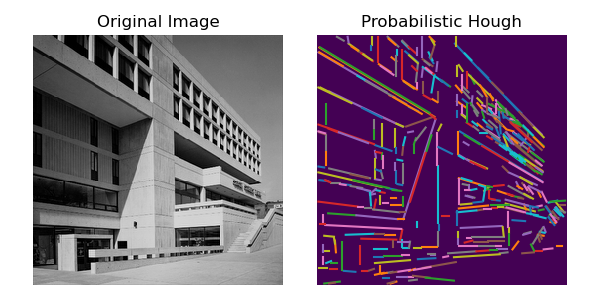
1. **利用Hough变换来检测图像中的直线，与变换过程相关的系列约束条件（线段的最小长度等）可自行叠加。**

思路：先通过Canny边缘检测算法获取图像边缘，将其输入霍夫变换中即可得到图像中的直线。

Canny算子原理：先通过高斯滤波进行降噪，在利用Sobel算子计算梯度大小和方向，最后设定一个或多个阈值来分离最可能的边缘（极值）。

Hough变换原理：笛卡尔坐标系中一条直线，对应霍夫空间的一个点。如果笛卡尔坐标系的点共线，这些点在霍夫空间对应的直线交于一点。最后从中选择由尽可能多直线汇成的点，就完成了霍夫变换的求解。

|  |
| --- |
| import skimage from skimage import io from skimage.feature import canny from skimage.transform import probabilistic\_hough\_line import numpy as np import matplotlib.pyplot as plt  img\_path = 'img/bank.bmp' img = skimage.io.imread(img\_path)  plt.figure(figsize=(6, 3)) plt.subplot(1, 2, 1) plt.title('Original Image') plt.axis('off') io.imshow(img)  plt.subplot(1, 2, 2) plt.title('Probabilistic Hough') plt.axis('off') edges = canny(img, 2, 1, 25) lines = probabilistic\_hough\_line(edges, threshold=10, line\_length=10, line\_gap=3)  plt.imshow(edges \* 0) for line in lines:  p0, p1 = line  plt.plot((p0[0], p1[0]), (p0[1], p1[1]))  plt.tight\_layout() plt.show() |

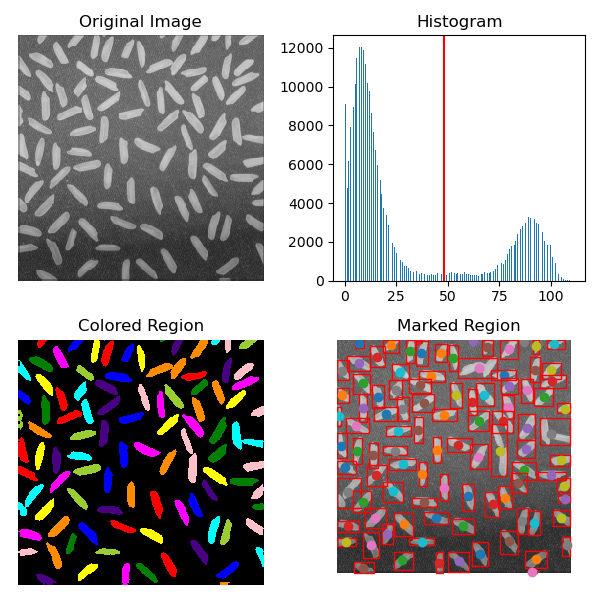


图表 4原图与Hough变换后

1. **对图像执行阈值分割操作并统计出每一个区域块的属性，然后，将每个区域的中心和外接矩形给标注出来。**

思路：先用顶帽对图像进行预处理，再用ostu 进行分块，最后统计相关属性

|  |
| --- |
| import skimage from skimage import io, morphology, measure, color from skimage.filters import threshold\_otsu import matplotlib.pyplot as plt import matplotlib.patches as mpatches import pandas as pd  img\_path = 'img/rice.bmp' img = skimage.io.imread(img\_path)  plt.figure(figsize=(6, 6))  plt.subplot(2, 2, 1) plt.title('Original Image') plt.axis('off') io.imshow(img)  se = morphology.disk(20) opening = morphology.opening(img, se) white\_tophat = img - opening  thresh = threshold\_otsu(white\_tophat) bi = white\_tophat > thresh  labels = measure.label(bi, connectivity=2) labels = morphology.remove\_small\_objects(labels, min\_size=10, connectivity=1) colored\_region = color.label2rgb(labels)  plt.subplot(2, 2, 2) plt.title('Histogram') plt.hist(white\_tophat.ravel(), bins=256) plt.axvline(thresh, color='r')  plt.subplot(2, 2, 3) plt.title('Colored Region') plt.axis('off') io.imshow(colored\_region)  ax = plt.subplot(2, 2, 4) plt.title('Marked Region') plt.axis('off') io.imshow(img) for region in measure.regionprops(labels):  minr, minc, maxr, maxc = region.bbox  rect = mpatches.Rectangle((minc, minr), maxc - minc, maxr - minr,  fill=False, edgecolor='red', linewidth=1)  ax.add\_patch(rect)  ax.scatter(region.centroid[1], region.centroid[0])  print(region.centroid)  # ax.plot(region.centroid)   stat = measure.regionprops\_table(labels, properties=('label', 'bbox', 'centroid', 'perimeter', 'eccentricity'), separator=',') stat = pd.DataFrame(stat) stat.to\_csv('stat.csv', sep=',')  plt.tight\_layout() plt.show() |



图表 5原图、灰度柱状图与阈值、分割后区域、标记后的图像

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Idx** | **label** | **bbox,0** | **bbox,1** | **bbox,2** | **bbox,3** | **centroid,0** | **centroid,1** | **perimeter** | **eccentricity** |
| 0 | 1 | 0 | 40 | 18 | 60 | 7.3 | 48.06666667 | 69.55634919 | 0.774681125 |
| 1 | 2 | 0 | 102 | 28 | 138 | 11.81782946 | 118.5930233 | 111.8700577 | 0.908405873 |
| 2 | 3 | 0 | 152 | 50 | 172 | 24.3091922 | 161.3133705 | 130.9705627 | 0.944397348 |
| 3 | 4 | 0 | 178 | 4 | 182 | 1.2 | 179.7 | 8 | 0.781543864 |
| 4 | 5 | 0 | 296 | 6 | 308 | 2.5 | 300.9285714 | 34.24264069 | 0.898548914 |
| 5 | 6 | 0 | 320 | 36 | 344 | 15.24782609 | 330.1173913 | 107.8345238 | 0.911768202 |
| 6 | 7 | 0 | 360 | 42 | 398 | 19.54301075 | 378.6397849 | 140.9411255 | 0.938703272 |
| 7 | 8 | 0 | 404 | 12 | 422 | 5.181818182 | 412.2727273 | 52.48528137 | 0.784907432 |
| 8 | 9 | 0 | 432 | 30 | 446 | 12.93956044 | 438.3241758 | 80.48528137 | 0.872491459 |
| 9 | 10 | 0 | 462 | 20 | 502 | 9.62962963 | 478.3888889 | 106.627417 | 0.928519892 |
| 10 | 11 | 2 | 174 | 58 | 200 | 29.59039191 | 186.3084703 | 147.1126984 | 0.949549274 |

图表 6部分区域块属性（具体可见附件）

1. **设计一个简易的Matlab GUI界面程序，要求其具有如下的功能：①打开与保存图像时均打开文件名设置对话框；②当下拉菜单中的条目被选中时，列表框之中实时的记录下当前的选择；③通过编辑框来实现相关参数的交互式输入；④将输入图像及处理后的结果显示在相应的坐标轴之上；⑤含有工具栏和菜单栏，当选择其下的组件成分时，要有相应的图像处理行为发生；⑥将figure窗口的“Name”属性修改为自己的姓名和学号；⑦将所设计的GUI程序编译为“.exe”形式的可执行文件**

思路：内容较多，具体可见源代码。