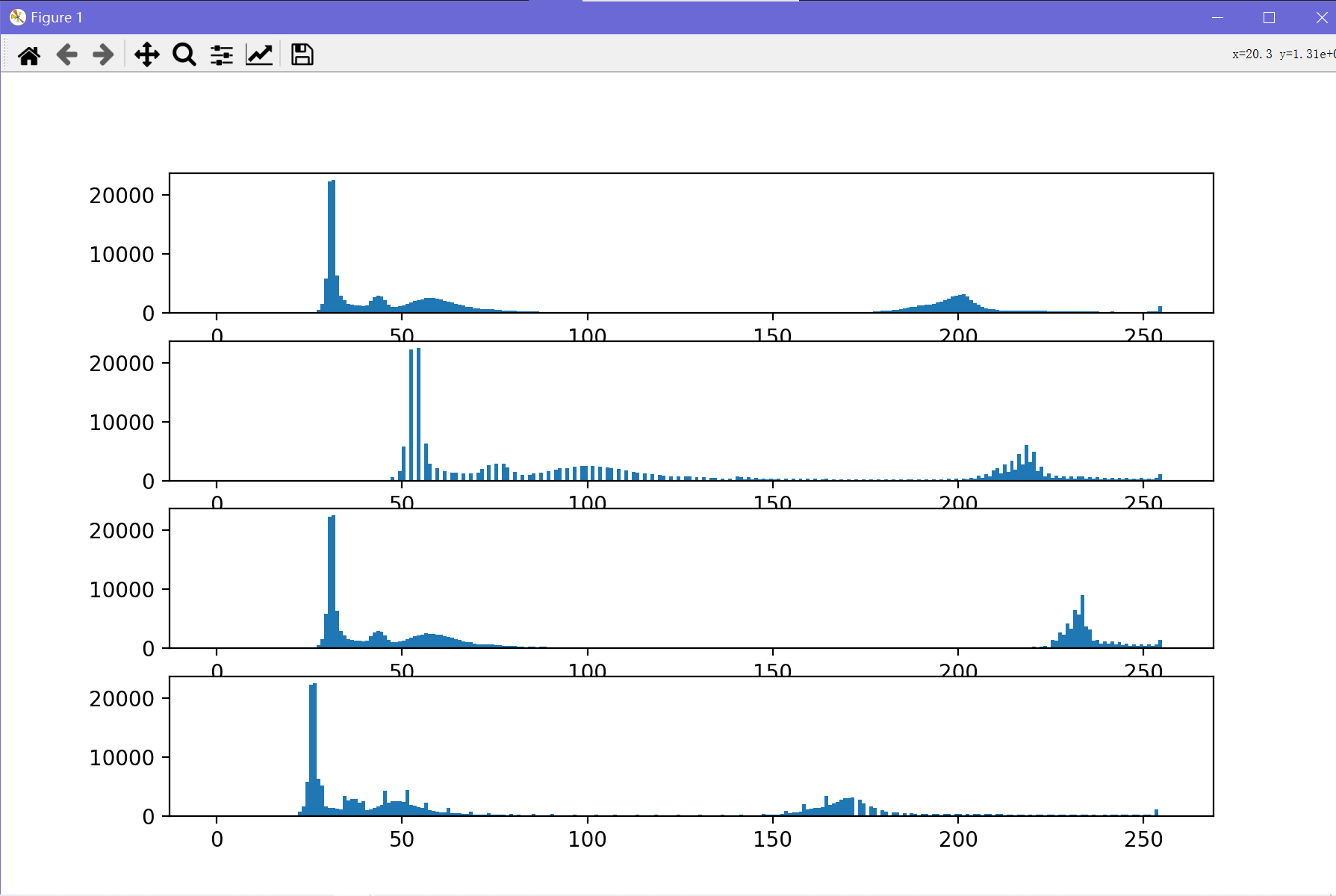
1. 实验目的
   1. 肺部动态范围拉伸
   2. 肌肉动态范围拉伸
   3. 骨骼动态范围拉伸
   4. 肺部灰级窗开窗显示
   5. 肌肉灰级窗开窗显示
   6. 骨骼灰级窗开窗开始
2. 实验结果
   1. 动态拉伸
      1. 直方图变化

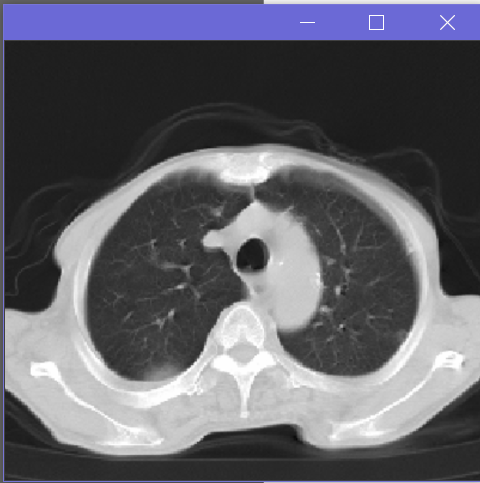
最上方为原始图像的灰度直方图，之后自上到下依次是肺部动态拉伸后的灰度直方图，肌肉动态范围拉伸后的灰度直方图，骨骼动态范围后的灰度直方图。

肺部动态范围拉伸是将0~80的灰度值拉至0~120

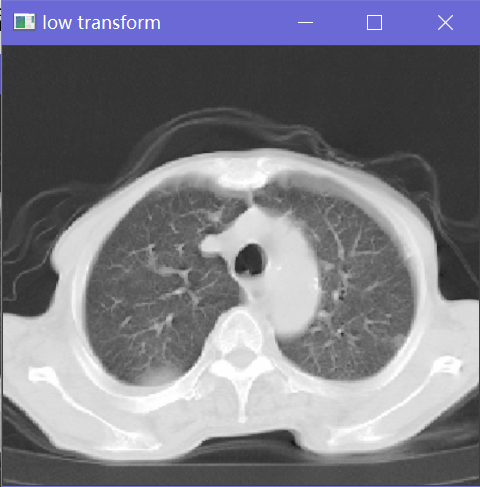
肌肉动态范围拉伸是将80~110的灰度值拉伸至120~175

骨骼动态范围拉伸是150~210的灰度值拉伸至175~255

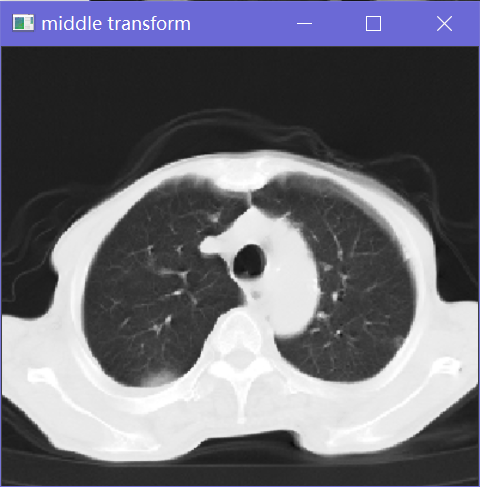
原始图像：



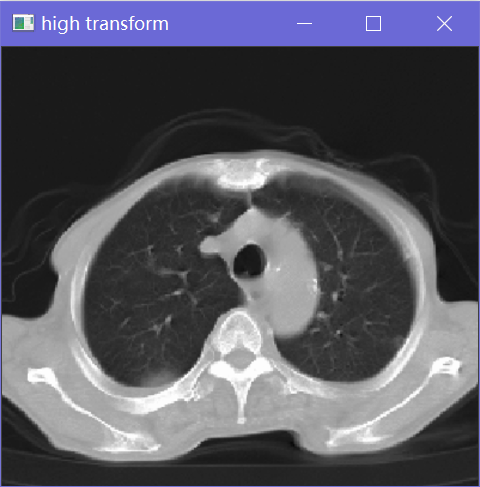
肺部动态拉伸后：



在上图基础上进行肌肉动态拉伸后：

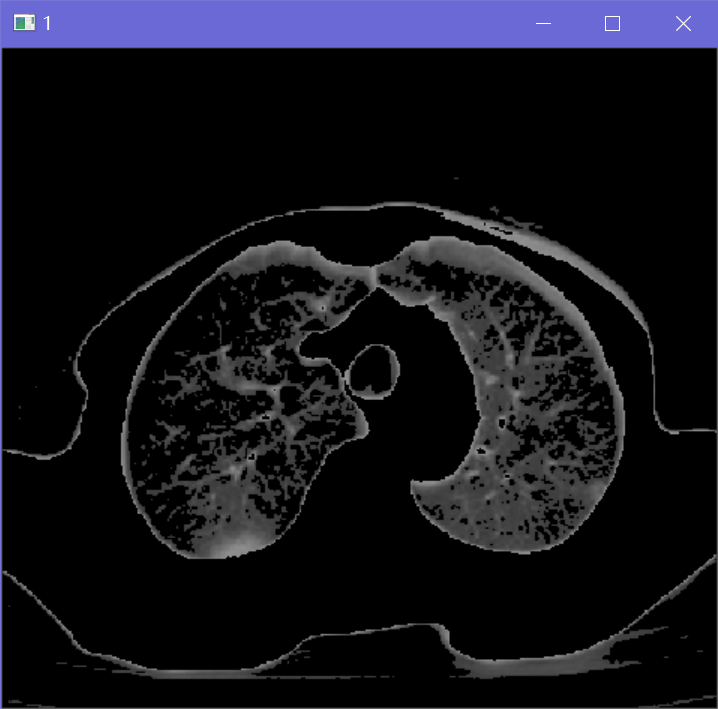


在上图基础上进行骨骼动态拉伸后：



* 1. 开窗

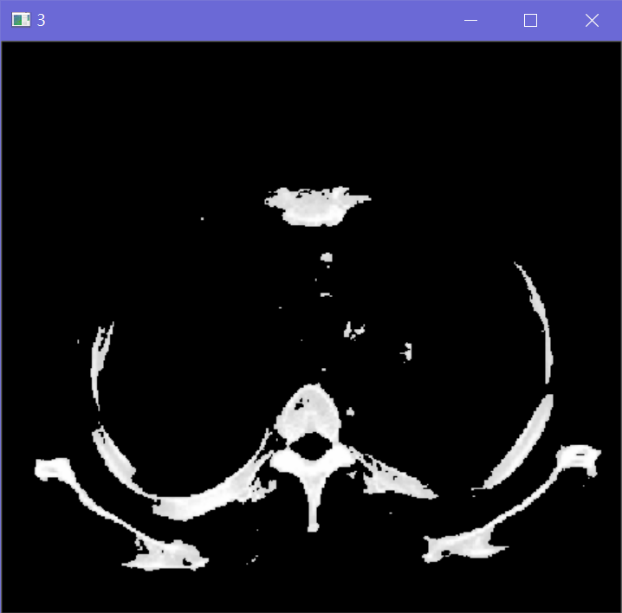
肺部开窗：



肌肉开窗：



骨骼开窗：



* 1. LBP编码
  2. LBP编码
     1. 8领域编码



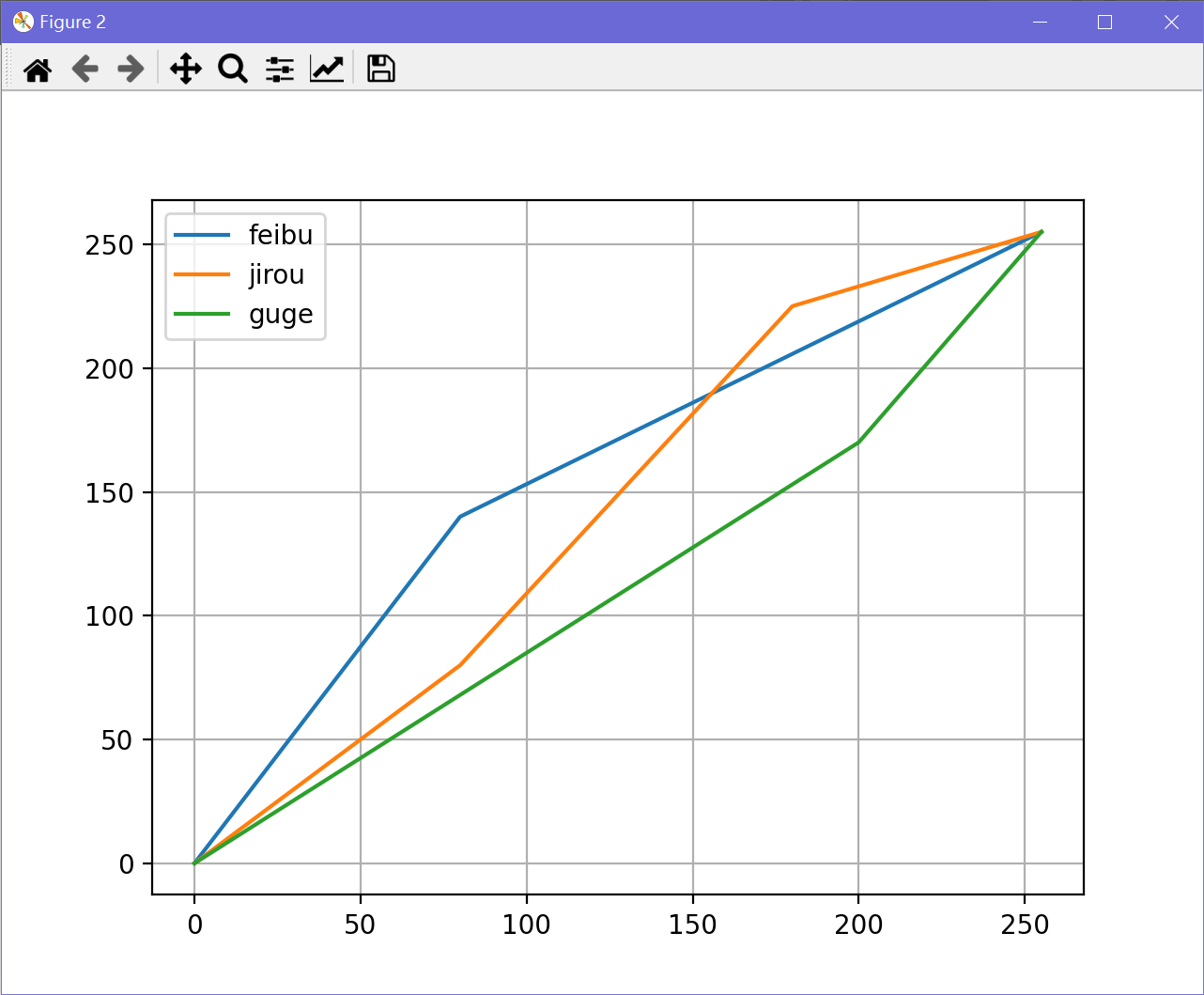
上图为原始图像，下图为编码后图像

* + 1. 半径为3的圆形领域编码



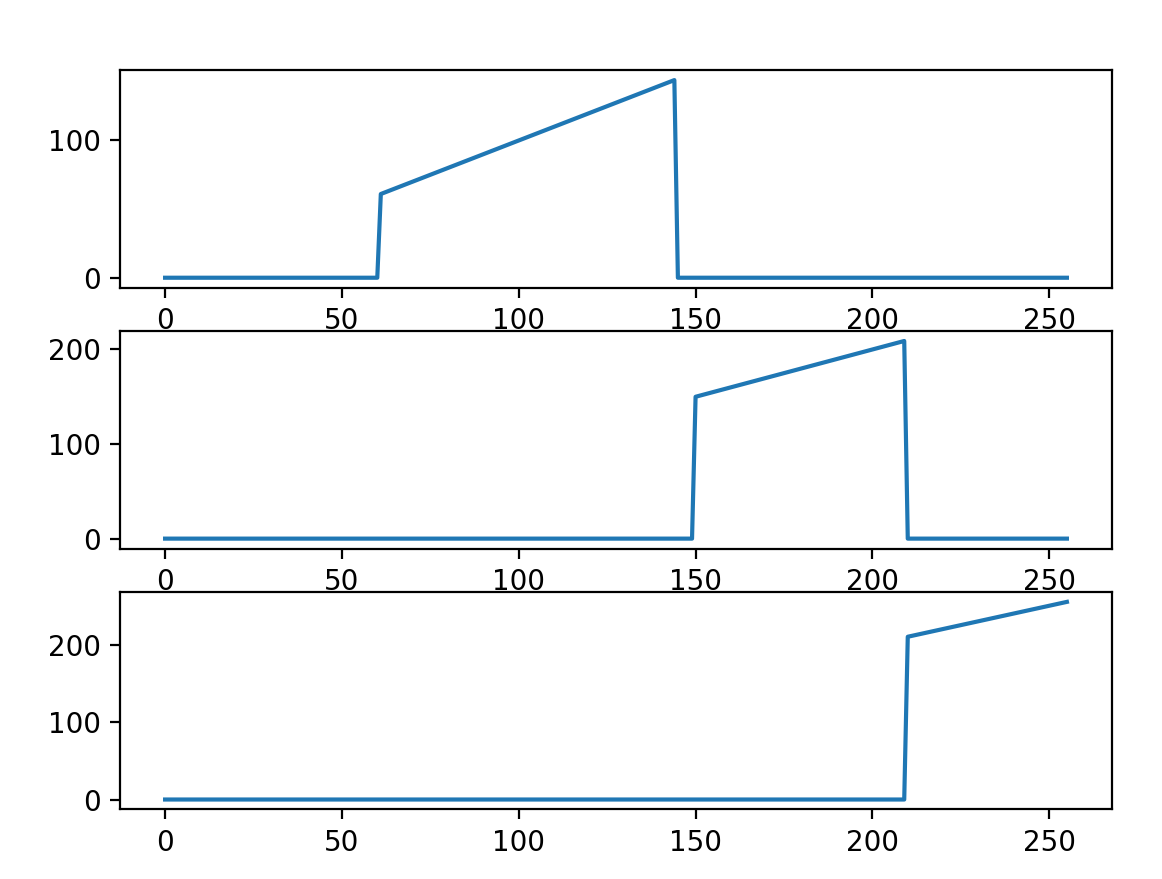
上图为原始图像，下图为编码后图像

1. 实验原理
   1. 动态拉伸

灰度值映射曲线

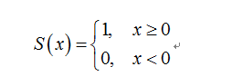
* 1. 开窗

灰度值映射曲线



从上至下依次是肺部开窗，肌肉开窗，骨骼开窗

* 1. LBP编码



1. 实现代码
   1. 动态范围拉伸

import numpy as np

import matplotlib.pyplot as plt

import cv2

img = cv2.imread("lung.png", 0).astype(np.float64)

cv2.imshow("origin", img/255)

def transform(origin\_img, x1, x2, x3, x4, i):

img = origin\_img.copy()

mask = (img>=x1) & (img<=x2)

mask1 = (img<x1)

mask2 = (img>x2)

if i == 1:

img[mask] = x1 + (img[mask] - x1) / (x2 - x1) \* (x4 - x3)

img[~mask] = x4 + (img[~mask] - x2) / (255 - x2) \* (255-x4)

elif i == 2:

img[mask1] = 0 + (img[mask1] - 0 )/(x1- 0) \* (x3 - 0)

img[mask] = x3 + (img[mask] - x1) / (x2 - x1) \* (x4 - x3)

img[mask2] = x4 + (img[mask2] - x2) / ( 255 - x2) \* (255 - x4)

elif i == 3:

img[mask] = x3 + (img[mask] - x1) / (x2 - x1) \* (x4 - x3)

img[~mask] = 0 + (img[~mask] - 0) / (x1 - 0) \* (x3 - 0)

else:

pass

return img

fig, ax = plt.subplots(4, 1, figsize=(20, 30))

ax[0].hist(img.ravel(), 256, [0, 256])

origin\_img = img.copy()

img1 = transform(origin\_img, 0, 80, 0, 140,1 )

img2 = transform(origin\_img, 80, 180, 80, 225, 2)

img3 = transform(origin\_img, 200, 255, 170, 255, 3)

ax[1].hist(img1.ravel(), 256, [0, 256])

ax[2].hist(img2.ravel(), 256, [0, 256])

ax[3].hist(img3.ravel(), 256, [0, 256])

img1 /= 255

img2 /= 255

img3 /= 255

array = np.linspace(0, 255, num=1000)

array2 = array.copy()

transformed\_array\_1 = transform(array,0, 80, 0, 140, 1)

transformed\_array\_2 = transform(array, 80, 180, 80, 225, 2)

transformed\_array\_3 = transform(array, 200, 255, 170, 255, 3)

plt.figure()

plt.plot(array2, transformed\_array\_1, label="feibu")

plt.plot(array2, transformed\_array\_2, label="jirou")

plt.plot(array2, transformed\_array\_3, label="guge")

plt.grid()

plt.legend()

cv2.imshow("low transform", img1)

cv2.imshow("middle transform", img2)

cv2.imshow("high transform", img3)

plt.show()

cv2.waitKey(0)

* 1. 开窗

import numpy as np

import matplotlib.pyplot as plt

import cv2

img = cv2.imread("lung.png", 0).astype(np.float64)

cv2.imshow("origin", img/255)

mask1 = (img >= 60) & (img <= 145)

img1 = img.copy()

img1[~mask1] = 0

mask2 = (img > 150) & (img <= 210)

img2 = img.copy()

img2[~mask2] = 0

mask3 = (img > 210)

img3 = img.copy()

img3[~mask3] = 0

cv2.imshow("1", img1/255)

cv2.imshow("2", img2/255)

cv2.imshow("3", img3/255)

cv2.waitKey(0)

* 1. LBP编码
     1. 8领域编码

import numpy as np

import matplotlib.pyplot as plt

import cv2

img = cv2.imread("lung.png", 0).astype(np.float64)

fig, ax = plt.subplots(2, 1)

ax[0].imshow(img, cmap="gray\_r")

ax[0].axis("off")

size = img.shape

img = np.pad(img, [[1,1],[1,1]])

power = np.array([[2\*\*0, 2\*\*1, 2\*\*2], [2\*\*7, 0, 2\*\*3], [2\*\*6, 2\*\*5, 2\*\*4]])

origin\_img = img.copy()

for i in range(1, size[0]):

for j in range(1, size[1]):

mask = origin\_img[i:i+3, j:j+3] < img[i+1, j+1]

img[i+1, j+1] = np.sum(np.multiply(power, mask))

ax[1].imshow(img, cmap="gray\_r")

ax[1].axis("off")

plt.show()

* + 1. 半径为3原型领域编码

import numpy as np

import matplotlib.pyplot as plt

import cv2

img = cv2.imread("lung.png", 0).astype(np.float64)

fig, ax = plt.subplots(2, 1)

ax[0].imshow(img, cmap="gray\_r")

ax[0].axis("off")

size = img.shape

img = np.pad(img, [[1, 1], [1, 1]])

origin\_img = img.copy()

for i in range(1, size[0]):

for j in range(1, size[1]):

sub\_matrix = origin\_img[i:i + 3, j:j + 3]

point = sub\_matrix[ 1, 1]

num = 0

for c, theta in enumerate(np.linspace(-np.pi, -np.pi \* 2, 8)):

x, y = np.cos(theta), np.sin(theta)

fx1 = (1 - x) / 2 \* sub\_matrix[0, 0] + (x + 1) / 2 \* sub\_matrix[0, 2]

fx2 = (1 - x) / 2 \* sub\_matrix[2, 0] + (x + 1) / 2 \* sub\_matrix[2, 2]

f = (1 - y) / 2 \* fx1 + (y + 1) / 2 \* fx2

num += 2 \*\* c if f > point else 0

img[i+1, j+1] = num

ax[1].imshow(img, cmap="gray\_r")

ax[1].axis("off")

plt.show()