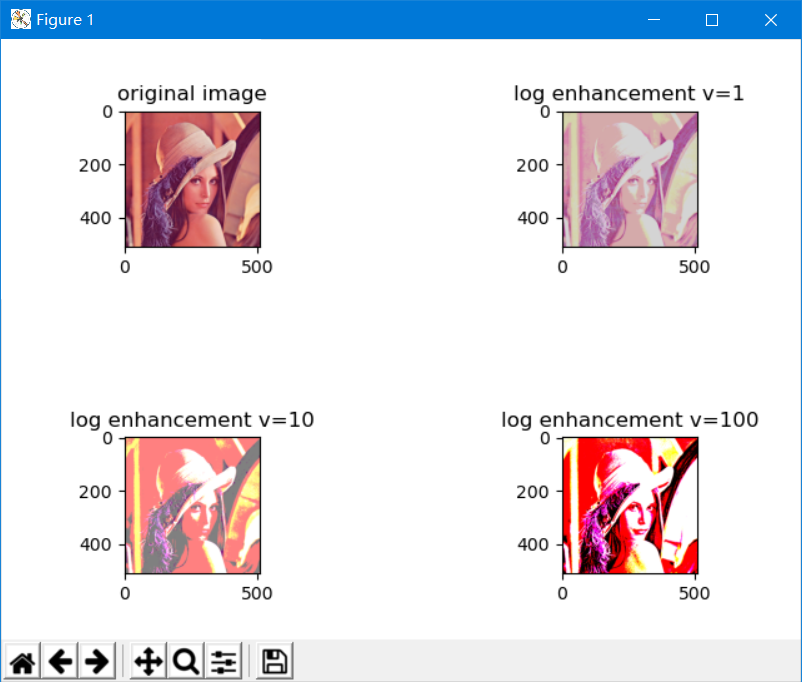
数字图像处理第二次实验

一、Log变换图像增强

【实验原理】

对数变换可以将图像的低灰度值部分扩展，显示出低灰度部分更多的细节，将其高灰度值部分压缩，减少高灰度值部分的细节，从而达到强调图像低灰度部分的目的。变换方法为：

【实验内容】



【源代码】

|  |
| --- |
| import cv2 as cv  import os  import copy  from matplotlib import pyplot as plt  import numpy as np  from numba import jit  ''' è¯»å–å¹¶ç»˜åˆ¶åŽŸå›¾åƒ '''  imgPath = os.path.dirname(\_\_file\_\_) + "\Lena.jpg"  img = cv.imread(imgPath)  img = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  plt.subplot(331)  plt.title("original image")  plt.imshow(img)  @jit  def logTransf(img, v):  ''' logå˜æ¢  @parm img åŽŸå›¾åƒ  @parm v logå˜æ¢å‚æ•°  @return å˜æ¢åŽçš„å›¾åƒ  '''  imgEnhanced = np.zeros(img.shape)  for i in range(img.shape[0]):  for j in range(img.shape[1]):  for k in range(img.shape[2]):  imgEnhanced[i][j][k] = np.log(img[i][j][k]\*v + 1) / np.log(v+1)  return imgEnhanced  ''' v=1 '''  imgEnhanced = np.uint8(logTransf(img, 1))  cv.normalize(imgEnhanced, imgEnhanced, 0, 255, cv.NORM\_MINMAX)  plt.subplot(333)  plt.title("log enhancement v=1")  plt.imshow(imgEnhanced)  ''' v=10 '''  imgEnhanced = np.uint8(logTransf(img, 10))  cv.normalize(imgEnhanced, imgEnhanced, 0, 255, cv.NORM\_MINMAX)  plt.subplot(337)  plt.title("log enhancement v=10")  plt.imshow(imgEnhanced)  ''' v=100 '''  imgEnhanced = np.uint8(logTransf(img, 100))  cv.normalize(imgEnhanced, imgEnhanced, 0, 255, cv.NORM\_MINMAX)  plt.subplot(339)  plt.title("log enhancement v=100")  plt.imshow(imgEnhanced)  plt.show() |

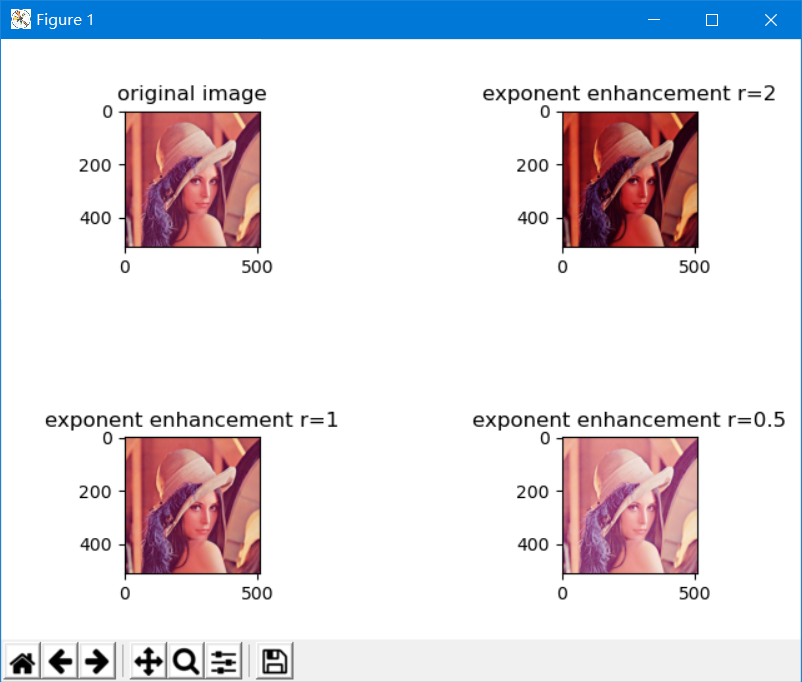
二、指数变换图像增强

【实验原理】

变换公式：

c=1

【实验结果】



【源代码】

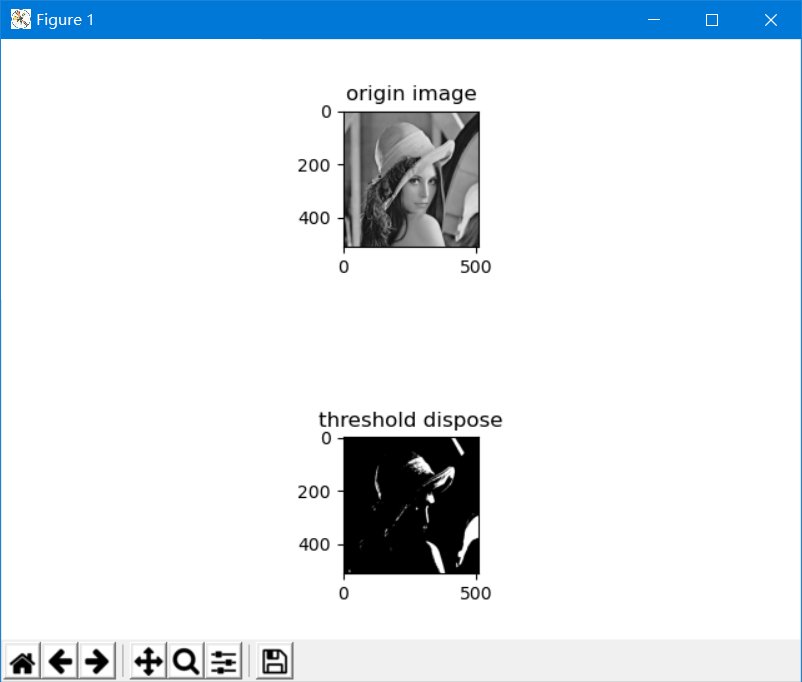
|  |
| --- |
| import cv2 as cv  import os  import copy  from matplotlib import pyplot as plt  import numpy as np  from numba import jit  import math as m  ''' 读取并绘制原图像 '''  imgPath = os.path.dirname(\_\_file\_\_) + "\Lena.jpg"  img = cv.imread(imgPath)  img = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  plt.subplot(331)  plt.title("original image")  plt.imshow(img)  @jit  def exponentTransf(img, r):  ''' 指数变换  @parm img 原图像  @parm r 指数变换参数  @return 变换后的图像  '''  # cv.normalize(img, img, 0, 1, cv.NORM\_MINMAX)  imgEnhanced = np.zeros(img.shape)  for i in range(img.shape[0]):  for j in range(img.shape[1]):  for k in range(img.shape[2]):  imgEnhanced[i][j][k] = np.power(img[i][j][k], r)  cv.normalize(imgEnhanced, imgEnhanced, 0, 255, cv.NORM\_MINMAX)  return imgEnhanced  ''' r=2 '''  imgEnhanced = np.uint8(exponentTransf(img, 2))  plt.subplot(333)  plt.title("exponent enhancement r=2")  plt.imshow(imgEnhanced)  ''' r=1 '''  imgEnhanced = np.uint8(exponentTransf(img, 1))  plt.subplot(337)  plt.title("exponent enhancement r=1")  plt.imshow(imgEnhanced)  ''' r=0.5 '''  imgEnhanced = np.uint8(exponentTransf(img, 0.5))  plt.subplot(339)  plt.title("exponent enhancement r=0.5")  plt.imshow(imgEnhanced)  plt.show() |

三、阈值图像增强

【实验原理】

对于灰度图像的每一个像素，如果灰度值大于阈值，置为255，否则，置为0

【实验结果】



【源代码】

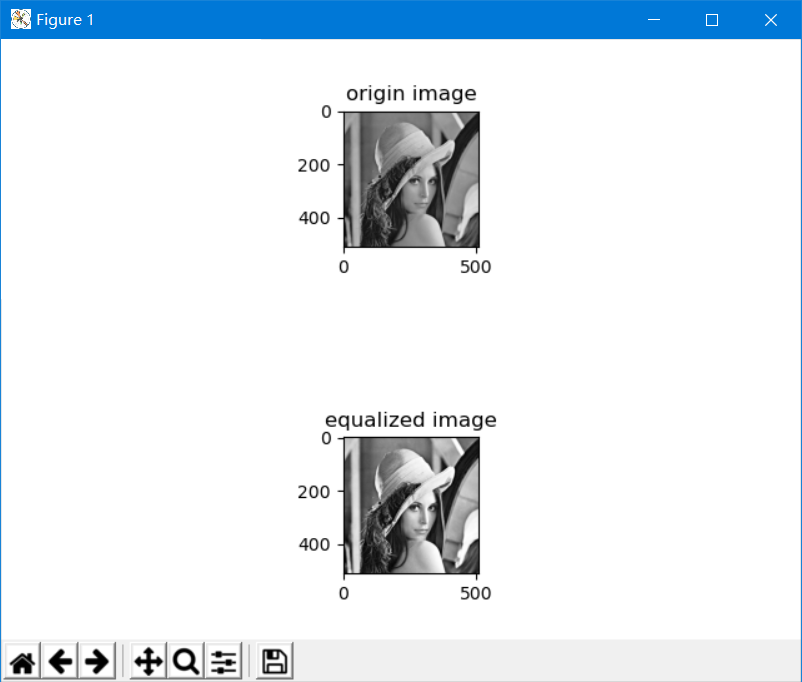
|  |
| --- |
| import cv2 as cv  import os  import copy  from matplotlib import pyplot as plt  import numpy as np  from numba import jit  imgPath = os.path.dirname(\_\_file\_\_) + "\Lena.jpg"  imgGray = cv.imread(imgPath, cv.IMREAD\_GRAYSCALE)  plt.subplot(311)  plt.title("origin image")  plt.imshow(imgGray, cmap='gray')  @jit  def threshold\_dispose(imgGray, threshold):  ''' 对灰度图像imgGray做阈值处理 '''  imgRes = np.zeros(imgGray.shape)  for i in range(imgGray.shape[0]):  for j in range(imgGray.shape[1]):  if imgGray[i][j] > threshold:  imgRes[i][j] = 255  else:  imgRes[i][j] = 0  return imgRes  imgRes = threshold\_dispose(imgGray, 200)  plt.subplot(313)  plt.imshow(imgRes, cmap='gray')  plt.title("threshold dispose")  plt.show() |

四、直方图均衡化

【实验原理】

直方图均衡化是一种简单有效的图像增强技术，通过改变图像的直方图来改变图像中各像素的灰度，主要用于增强动态范围偏小的图像的对比度。原始图像由于其灰度分布可能集中在较窄的区间，造成图像不够清晰。例如，过曝光图像的灰度级集中在高亮度范围内，而曝光不足将使图像灰度级集中在低亮度范围内。采用直方图均衡化，可以把原始图像的直方图变换为均匀分布（均衡）的形式，这样就增加了像素之间灰度值差别的动态范围，从而达到增强图像整体对比度的效果。换言之，直方图均衡化的基本原理是：对在图像中像素个数多的灰度值（即对画面起主要作用的灰度值）进行展宽，而对像素个数少的灰度值（即对画面不起主要作用的灰度值）进行归并，从而增大对比度，使图像清晰，达到增强的目的。

【实验结果】



【源代码】

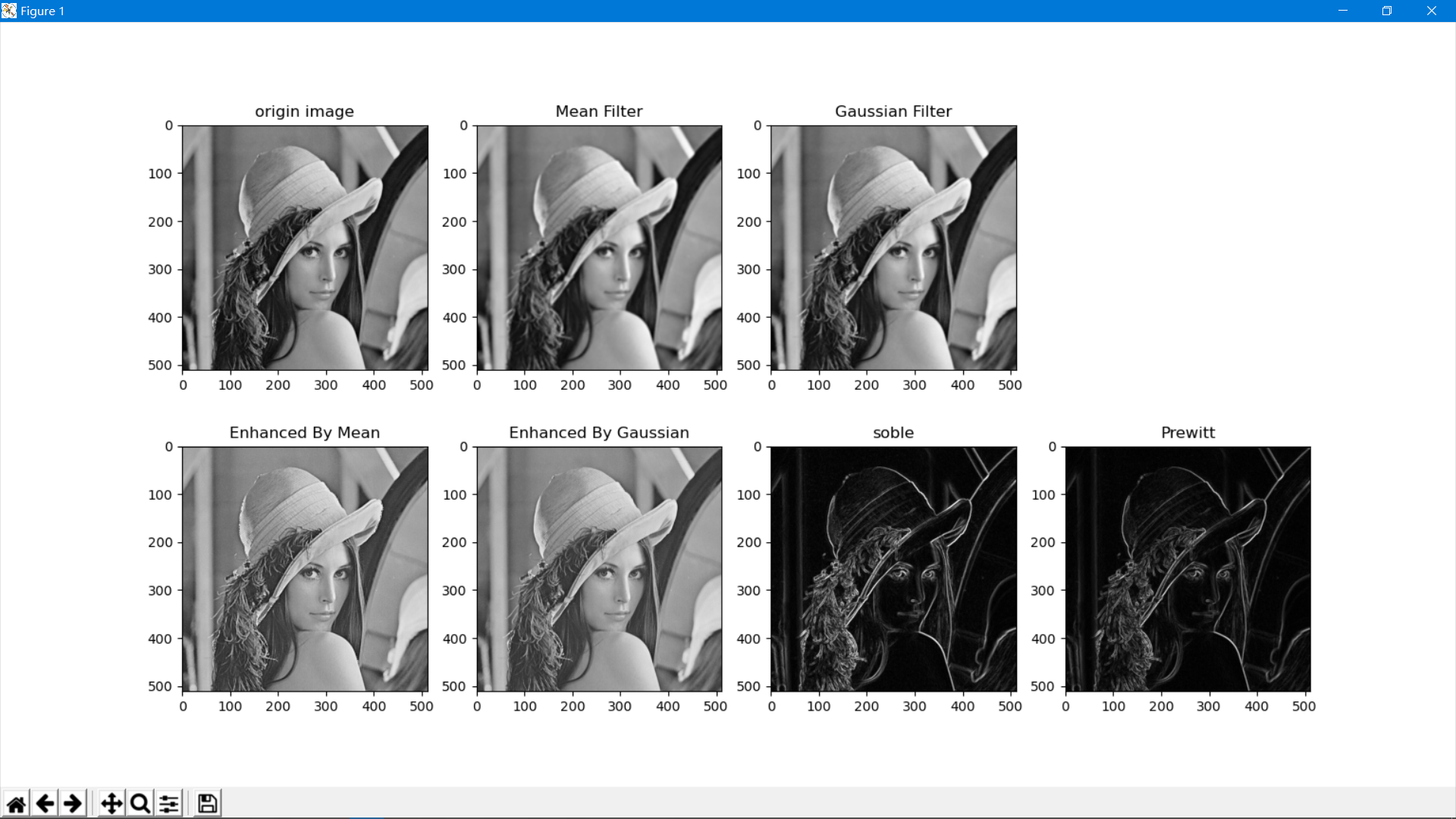
|  |
| --- |
| import cv2 as cv  import os  import copy  import numpy as np  from matplotlib import pyplot as plt  from numba import jit  ''' 读取并绘制原图像 '''  imgPath = os.path.dirname(\_\_file\_\_) + "\Lena.jpg"  imgGray = cv.imread(imgPath, cv.IMREAD\_GRAYSCALE)  plt.subplot(311)  plt.title("origin image")  plt.imshow(imgGray, cmap='gray')  ''' 直方图均衡化处理 '''  equ = cv.equalizeHist(imgGray)  plt.subplot(313)  plt.title("equalized image")  plt.imshow(equ, cmap='gray')  plt.show() |

五、灰度图像常用的平滑、锐化滤波

【实验原理】

均值低通滤波、高斯低通滤波、’Prewitt’算子增强、’Sobel’算子增强

【实验结果】



【源代码】

|  |
| --- |
| import cv2 as cv  import os  import copy  import numpy as np  from matplotlib import pyplot as plt  from numba import jit  ''' 读取并绘制原灰度图像 '''  imgPath = os.path.dirname(\_\_file\_\_) + "\Lena.jpg"  img = cv.imread(imgPath, cv.IMREAD\_GRAYSCALE)  plt.subplot(241)  plt.title("origin image")  plt.imshow(img, cmap='gray')  ''' 均值低通滤波 '''  img2 = cv.blur(img, (5,5))  plt.subplot(242)  plt.title("Mean Filter")  plt.imshow(img2, cmap='gray')  ''' 高斯低通滤波 '''  img3 = cv.GaussianBlur(img, (7, 3), 1)  plt.subplot(243)  plt.title("Gaussian Filter")  plt.imshow(img3, cmap='gray')  ''' 增强图像=原图-均值低通滤波 '''  img4 = img\*2 - img2  plt.subplot(245)  plt.title("Enhanced By Mean")  plt.imshow(img4, cmap='gray')  ''' 增强图像=原图-高斯低通滤波 '''  img5 = img\*2 - img3  plt.subplot(246)  plt.title("Enhanced By Gaussian")  plt.imshow(img5, cmap='gray')  ''' 'sobel'边缘算子增强 '''  x = cv.Sobel(img, cv.CV\_16S, 1, 0)  y = cv.Sobel(img, cv.CV\_16S, 0, 1)  absX = cv.convertScaleAbs(x)  absY = cv.convertScaleAbs(y)  img6 = cv.addWeighted(absX, 0.5, absY, 0.5, 0)  plt.subplot(247)  plt.title("soble")  plt.imshow(img6, cmap='gray')  ''' 'prewitt'边缘算子增强 '''  # Prewitt算子  kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]],dtype=int)  kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]],dtype=int)  x = cv.filter2D(img, cv.CV\_16S, kernelx)  y = cv.filter2D(img, cv.CV\_16S, kernely)  # 转 uint8 ,图像融合  absX = cv.convertScaleAbs(x)  absY = cv.convertScaleAbs(y)  img7 = cv.addWeighted(absX, 0.5, absY, 0.5, 0)  plt.subplot(248)  plt.title("Prewitt")  plt.imshow(img7, cmap='gray')  plt.show() |