**实验一：信号处理基础**

# 实验目的

1. 掌握基于python的数字音频及图片常用操作方法
2. 了解音频和图像变换系数特点，掌握离散余弦和时空域基本操作
3. 了解LSB隐写技术原理，掌握LSB隐写及提取基本操作

# 实验环境

1. Windows操作系统
2. Python3.6或以上科学计算软件
3. 图像文件
4. 音频文件

# 实验内容

## 数字音频常用操作

### 实验目的

掌握基于python的数字音频常用操作方法。

### 实验内容

利用python解码音频，读取并观察音频内容（音轨、采样率、时间等），并绘制相应波形图。

### 实验参考

import audioread  
import matplotlib.pyplot as plt  
import numpy as np  
  
filename = "1.wav"  
with audioread.audio\_open(filename) as f:  
 print(f.channels, f.samplerate, f.duration)  
 for buf\_i, buf in enumerate(f):  
 if buf\_i < 100:  
 continue  
 data = np.frombuffer(buf, np.int16)  
 plt.plot(range(buf\_i\*len(data)//2, (buf\_i+1)\*len(data)//2), data[0::2])  
 plt.plot(range(buf\_i\*len(data)//2, (buf\_i+1)\*len(data)//2), data[1::2])  
 if buf\_i > 120:  
 break  
 plt.show()

说明：

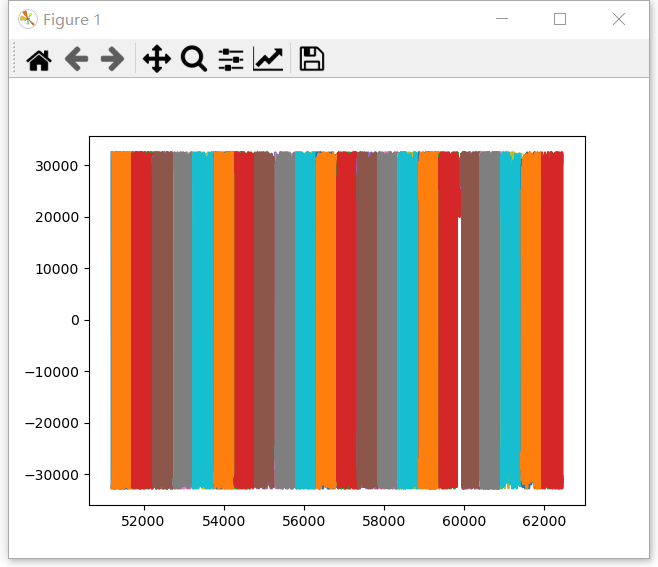
1. 使用函数audioread可读取全部/指定长度/指定位置及长度的音频样点，函数返回值为样点数组、采样率等。
2. 利用下述代码可以进行音频内容的读取，首先获得的是：

Channels: 音轨

Samplerate: 采样率

Duration: 持续时间

1. 设置音频样点读取范围，上述代码设置为100-120，可自行选取
2. 函数plt.plot中的range嵌套是为了计算这段data的时间长度，然后将时间对x轴进行覆盖，再赋到y轴上，从而形成波形图片。
3. 函数plt.show是展示图像的一种办法
4. 上述代码示例波形图如下：



### 课堂练习（请注意，课堂练习无需写入实验报告，仅仅是完成练习即可）

1. 选取文件夹中的音频“1.wav”，利用python解码音频，读取并记录音频内容（音轨、采样率、时间等）。
2. 修改音频样点读取范围为180-200，绘制相应波形图。

## 数字图像常用操作

### 实验目的

掌握基于Python的数字图像常用操作。

### 实验内容

1. 选取一幅图片，使用Python将其转换为512\*512的灰度图，保存为BMP图像文件，命名为flag.bmp。
2. 使用Python处理图像，将部分DCT系数置零后重构图像，有失真的图像保存为stego.bmp。
3. 完成实验，绘制出原始图像和失真图像，并记录其峰值信噪比。
4. 请思考：修改系数时，当选取的位置或数量不同，峰值信噪比是否相同？为什么？（建议结合实际数据分析）

### 实验参考

1. **准备灰度图像文件**

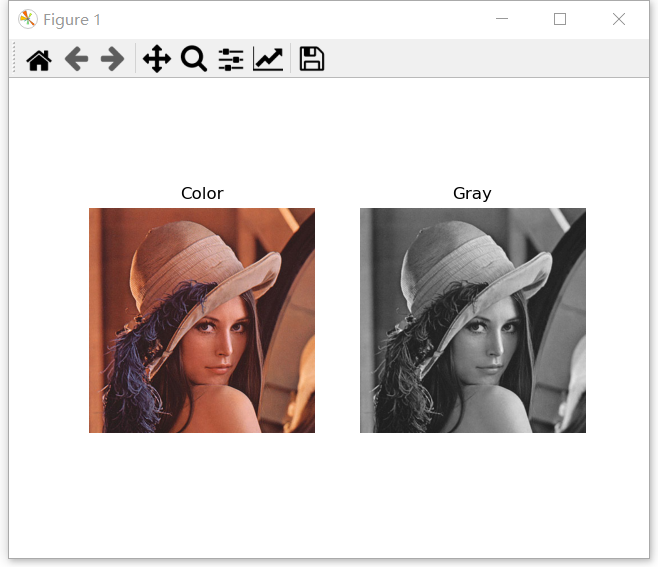
**需要安装CV2的包** pip install opencv-python

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg

img\_c = mpimg.imread('lena.bmp')  
img = cv2.imread('lena.bmp', cv2.IMREAD\_GRAYSCALE)  
img = cv2.resize(img,dsize=(512,512),fx=1,fy=1,interpolation=cv2.INTER\_LINEAR)  
cv2.imwrite('flag.bmp',img)  
plt.subplot(121)  
plt.imshow(img\_c), plt.title('Color'), plt.axis('off')  
plt.subplot(122)  
plt.imshow(img,'gray'), plt.title('Gray'), plt.axis('off')  
plt.show()

说明：

1. 为便于入门，这里选取512\*512大小的BMP灰度图为后续算法处理对象。为此，单独练习如何将常见的彩色BMP图像转换为灰度BMP图像，以及如何修改图片尺寸。上述示例使用图片为Lena.bmp。
2. 函数imread用于读取图像像素，函数返回值为图像像素及其调色板（可选）。
3. **参数cv2.IMREAD\_GRAYSCALE用于将彩色图像像素转换灰度图像像素。**
4. 函数resize通过插值的方式改变图片尺寸，dsize为期望的输出图像大小尺寸，fx和fy分别代表水平和竖直方向上的缩放系数。
5. 函数imwrite用于将图像保存为BMP文件（格式可选）。
6. 原始彩图图像与转换后的灰度图像对比如下：

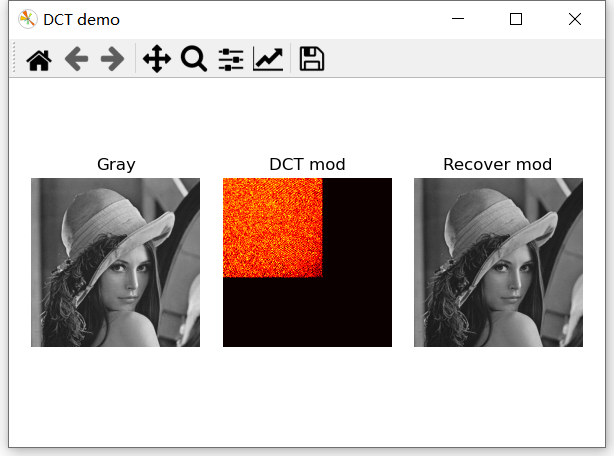


1. **使用Python处理图像：对图片进行DCT变换及逆变换，查看图片效果**

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
  
img\_c = mpimg.imread('Lena.bmp')  
img = cv2.imread('Lena.jpg', cv2.IMREAD\_GRAYSCALE)  
img = cv2.resize(img,dsize=(512,512),fx=1,fy=1,interpolation=cv2.INTER\_LINEAR)  
cv2.imwrite('flag.bmp',img)  
  
height,width = img.shape  
img\_dct = cv2.dct(np.array(img, np.float32))  
for i in range(0,width):  
 for j in range(0,height):  
 if i > 300 or j > 300:  
 img\_dct[i,j] = 0  
img\_r = np.array(cv2.idct(img\_dct), np.uint8)  
cv2.imwrite('stego.bmp',img\_r)  
  
fig = plt.figure('DCT demo', figsize=(4,2))  
plt.subplot(131)  
plt.imshow(img, 'gray'), plt.title('Gray'), plt.axis('off')  
plt.subplot(132)  
plt.imshow(np.array(img\_dct, np.uint8), cmap='hot'), plt.title('DCT mod'), plt.axis('off')  
plt.subplot(133)  
plt.imshow(img\_r, 'gray'), plt.title('Recover mod'), plt.axis('off')  
plt.tight\_layout()  
plt.show()

说明：

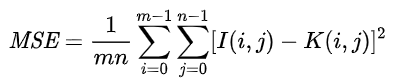
1. OpenCV提供离散余弦变换及逆变换的函数，分别为dct()及idct()，其操作对象均为numpy数组。
2. 上述代码将（300,300）以外的DCT系数置零，置零范围可自行选取。
3. 以下是样例图运行效果，变换后DCT系数能量主要集中在左上角，并且可见逆IDCT转换后的重构图片相较模糊：



1. **计算原始图像与失真图像的峰值信噪比**
2. **计算公式：**

峰值信噪比（英语：Peak signal-to-noise ratio，常缩写为PSNR）是一个表示信号最大可能功率和影响它的表示精度的破坏性噪声功率的比值的工程术语。由于许多信号都有非常宽的动态范围，峰值信噪比常用对数分贝单位来表示。

计算PSNR要先知道MSE(均方误差)的计算。两个m×n单色图像I和K，如果一个为另外一个的噪声近似，那么它们的的均方误差定义为：



PSNR就是通过MSE得出来的，其中，MAXI是表示图像点颜色的最大数值，如果每个采样点用 8 位表示，那么就是 255。公式如下：



因此，MSE越小，则PSNR越大，代表着图像质量越好。

1. **参考代码**

import math  
import numpy as np  
import cv2  
  
def psnr(img1, img2):  
 img1 = np.float64(img1)  
 img2 = np.float64(img2)  
 mse = np.mean((img1 / 1.0 - img2 / 1.0) \*\* 2)  
 if mse < 1.0e-10:  
 return 100  
 PIXEL\_MAX = 255.0  
 return 20 \* math.log10(PIXEL\_MAX / math.sqrt(mse))  
  
original = cv2.imread('flag.bmp')  
contrast = cv2.imread('stego.bmp')  
res = psnr(original, contrast)  
print(res)

### 课堂练习（请注意，课堂练习无需写入实验报告，仅仅是完成练习即可）

1. 选取文件夹中图片“test.bmp”，将其转换为512\*512的灰度图，保存为BMP图像文件，命名为“flag.bmp”。
2. 使用Python处理图像，将（300,300）范围以外的DCT系数置零后重构图像，有失真的图像保存为“stego.bmp”。
3. 完成实验，绘制出原始图像和失真图像，并记录其峰值信噪比。
4. 请思考：修改系数时，当选取的位置或数量不同，峰值信噪比是否相同？为什么？（建议结合实际数据分析）

## 图像LSB隐写

### 实验原理

首先，介绍图像的数据表示：

1 2 3 4 5 6 7 8

8

7

6

5

4

3

2

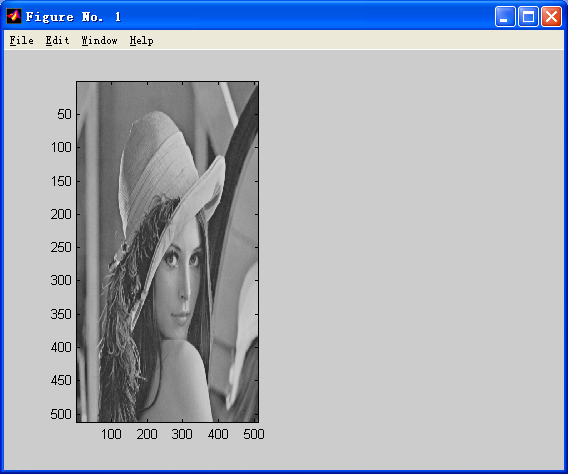
1

1

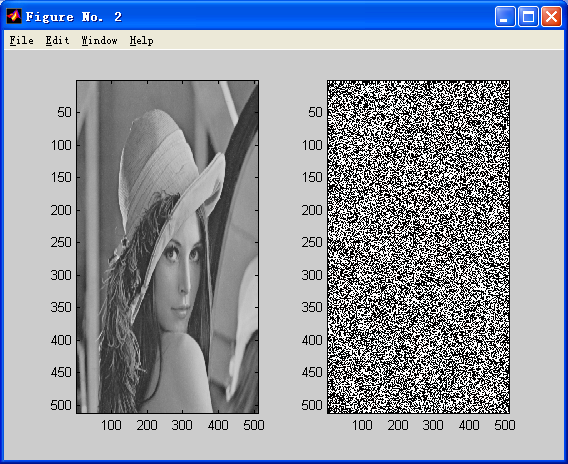
8

图像象素的灰度表示

如上图所示，以一个8×8的灰度图像为例，共有64个象素点，每一个象素点的取值为0到255，可以用8比特表示，图中每一个横截面代表一个位平面，第一个位平面由每一个象素最低比特位组成，第八个位平面由每一个象素的最高比特位组成。因此这八个位平面在图像中所代表的重要程度是不同的。以Lena图像为例。



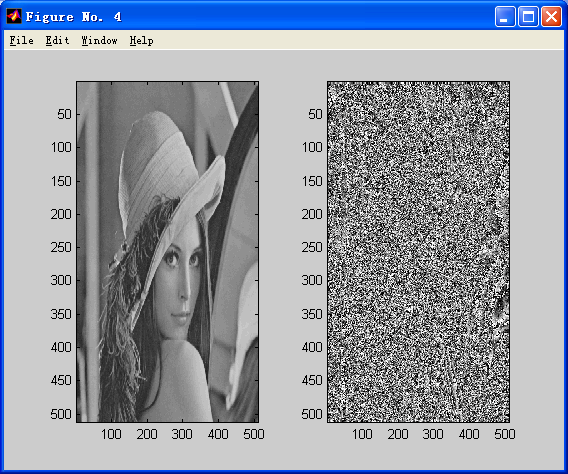
（a）原始图像（8bit 灰度BMP图像）



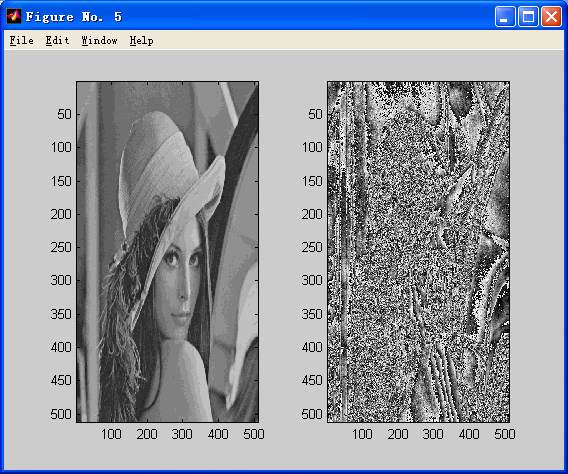
（b）去掉第一个位平面的Lena图像和第一个位平面



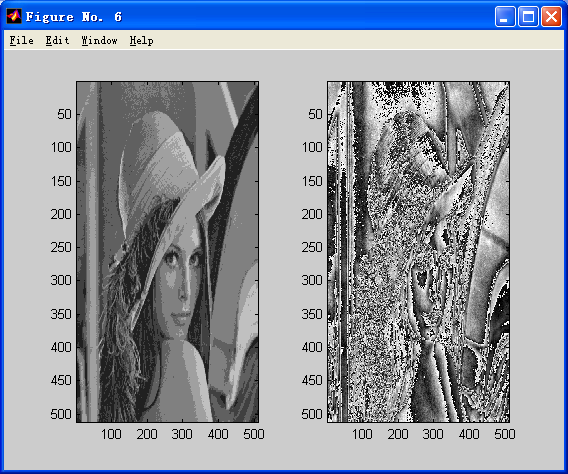
（c）去掉第1－2个位平面的Lena图像和第1－2个位平面



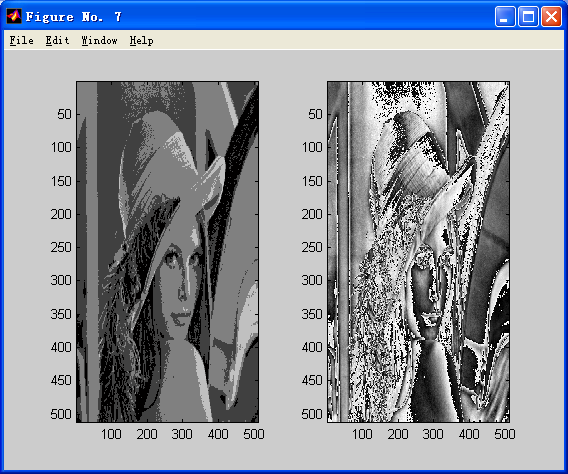
（d）去掉第1－3个位平面的Lena图像和第1－3个位平面



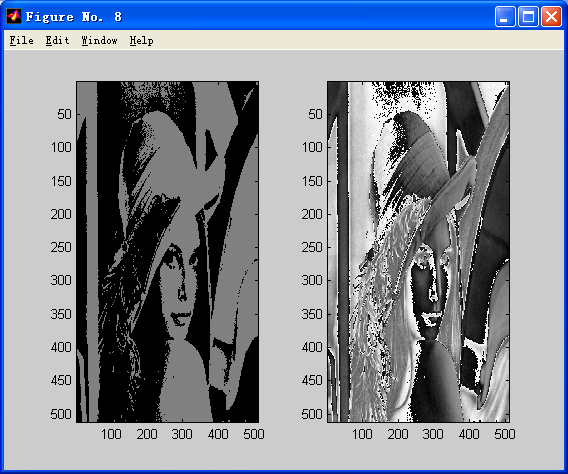
（e）去掉第1－4个位平面的Lena图像和第1－4个位平面



（f）去掉第1－5个位平面的Lena图像和第1－5个位平面



（g）去掉第1－6个位平面的Lena图像和第1－6个位平面



（h）去掉第1－7个位平面的Lena图像（即第八个位平面）和第1－7个位平面

Lena图像各个位平面示意图

图（b）和（c）中，最低的两个位平面反映的基本上是噪声，没有携带图像的有用信息；而加入第3个位平面后，则噪声信息显得不均匀，已经包含了一些图像信息；而1至4个位平面所携带的信息已经有了明显的不均匀，可以注意到已经不是均匀的噪声了，而去掉第1至4个位平面的Lena图像已经出现了可见的误差；往后的几幅图（f，g，h）则变化越来越明显。

从这个例子可以看出每一个位平面对图像能量的贡献大小，也可以帮助我们理解如何选择信息隐藏的位置，达到不易被察觉的目的。

彩色图像有三个位平面，分别是R，G和B三个颜色通道。

0-255

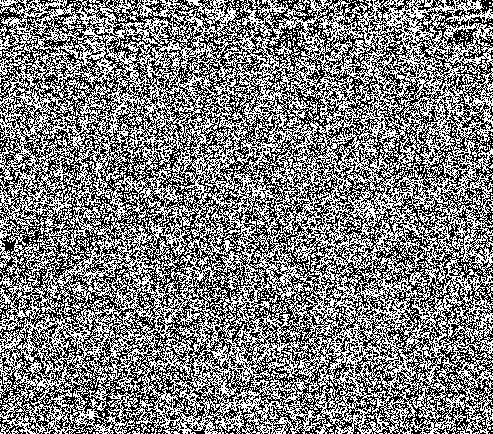
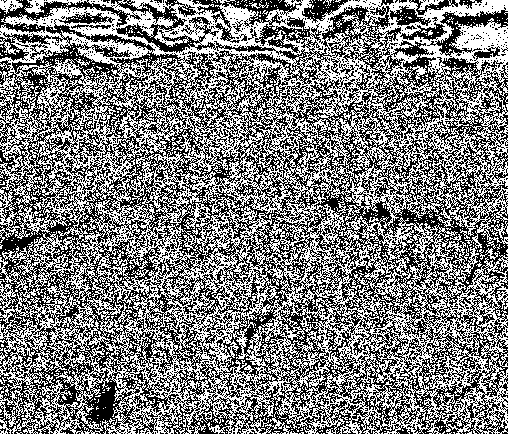
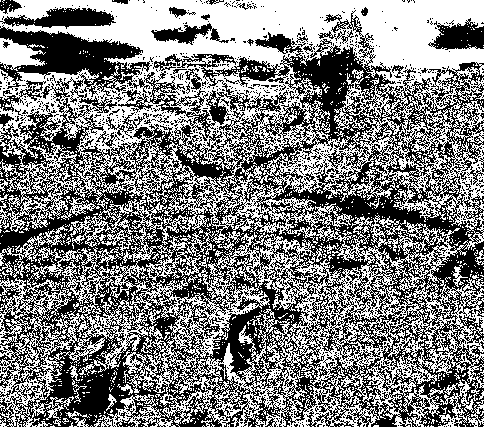
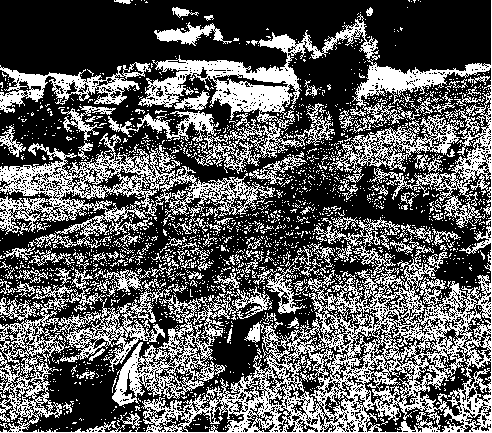


**红色通道**

**绿色通道**

**蓝色通道**

### 图像各个位平面的作用不同，不同的位平面包含的能量不同，比如绿色通道的1-6位平面。



最低有效位 (LSB) 是一种技术，其中每个像素的最后一位被修改并替换为秘密消息的数据位。每个像素包含红、绿、蓝（RGB）三个值，这些值的范围从0 到 255，换句话说，它们是 8 位值。LSB隐写即是修改每个颜色值的最低一位，将其替换为我们想要嵌入的信息中的内容，以此来实现数据隐藏。

例如有七个像素，其RGB矩阵相应像素的值为(225, 12, 99), (155, 2, 50), (99, 51, 15), (15, 55, 22),(155, 61, 87), (63, 30, 17), ( 1, 55, 19)。使用ASCII 表，我们可以将秘密消息转换为十进制值，然后转换为二进制值：0110100 0110101。现在，我们对像素值一一进行替换，将它们转换为二进制后，将每个最低有效位替换为该消息的比特值（例如 225 是 11100001，我们用第一个数据位替换最后一位，依此类推）。这只会将像素值 +1 或 -1，对颜色值的修改不是很明显，肉眼无法察觉其中的差异。执行LSB后得到的像素值如下：[(224, 13, 99),(154, 3, 50),(98, 50, 15),(15, 54, 23),(154, 61, 87),(63, 30, 17),( 1, 55, 19)

LSB算法可嵌入的信息极多。例如，一幅彩色图像是 512×512 的RGB位图，如果只嵌入最低位，最多可以嵌入512×512×3=786,432 bit = 96KB数据。因为彩色图像有R，G，B三个颜色通道，每个颜色通道可以嵌入一位信息。

应用LSB算法的图像格式需为位图形式，即图像不能经过压缩。故LSB算法多应用于无损压缩的png、bmp等格式，而较少应用于有损压缩的jpg格式。下面以灰度图像来说明图像和声音信号中可以用来隐藏信息的地方。

### 实验内容

1. 选取一幅RGB图片，思考的问题是：RGB彩色图像有R，G和B三个通道，LSB选取其中的一个通道隐藏，因此首先提取三个颜色通道，输出三个颜色通道的图。分别命名为R，G和B。
2. 每个颜色通道都有8个位平面，输出R通道的1-8位平面。
3. 对其使用LSB隐写技术隐藏秘密信息，对比隐写前后的图像，在屏幕中一行两列输出。注意：图片须为 RGB格式。提取隐藏的LSB隐写信息，对比原信息，判断是否提取成功。
4. 计算原始载体和隐藏秘密信息后的图像的峰值信噪比。

### 实验参考

**1、 LSB隐写信息：**

from PIL import Image

import matplotlib.pyplot as plt  
import matplotlib.image as mpimg

def get\_msg(msg):  
 return msg.zfill(8)  
def generate(msg): # 将信息转置为2进制。  
 result=''  
 for i in msg:  
 if isinstance(i,int):  
 result +=get\_msg(bin(i)).replace('0b','')  
 else:  
 result+= get\_msg(bin(ord(i)).replace('0b',''))  
 return result  
  
img=Image.open('Lena.bmp')  
tmp\_msg='informationhiding'  
hide\_msg=generate(tmp\_msg)  
width,height=img.size  
length=len(hide\_msg)  
tmp=''  
tmp2=''  
count=0# 将信息隐藏到其最低位。  
for i in range(0,width):  
 if count == length:  
 break  
 for j in range(0,height):  
 pixels=img.getpixel((i,j))  
 a,b,c=pixels  
 if count==length:  
 break  
 tmp+=str(a%2)  
 a-=a%2+int(hide\_msg[count])  
 tmp2+=str(a%2)  
 count+=1  
 if count == length:  
 img.putpixel((i, j), (a,b,c))  
 break  
 tmp += str(b % 2)  
 b -= b % 2 + int(hide\_msg[count])  
 tmp2 += str(b % 2)  
 count += 1  
 if count == length:  
 img.putpixel((i, j), (a, b, c))  
 break  
 tmp += str(c % 2)  
 c -= c % 2 + int(hide\_msg[count])  
 tmp2 += str(c % 2)  
 count += 1  
 if count == length:  
 img.putpixel((i, j), (a, b, c))  
 break  
 if count % 3 == 0:  
 img.putpixel((i, j), (a, b, c))  
img.save('encode.bmp')

plt.subplot(121)  
plt.imshow(mpimg.imread('Lena.bmp')), plt.title('Original'), plt.axis('off')  
plt.subplot(122)  
plt.imshow(mpimg.imread('encode.bmp')), plt.title('Encoded'), plt.axis('off')  
plt.show()

**2、LSB提取信息：**

from PIL import Image  
img=Image.open('encode.bmp')  
width,height=img.size  
count=0  
result=''  
result1=''

msg\_len=4 # LSB隐写数据长度   
length= msg\_len \*8  
for i in range(0,width):  
 if count==length:  
 break  
 for j in range(0,height):  
 pixel=img.getpixel((i,j))  
 if count == length:  
 break  
 if count%3==0:  
 count+=1  
 result1+=str(int(pixel[0])%2)  
 if count==length:  
 break  
 if count%3==1:  
 count+=1  
 result1+=str(int(pixel[1])%2)  
 if count==length:  
 break  
 if count%3==2:  
 count+=1  
 result1+=str(int(pixel[2])%2)  
 if count==length:  
 break  
 if count==length:  
 break  
print("Secret:"+result1)  
for i in range(0,len(result1),8):  
 result+=chr(int(result1[i:i+8],2))  
  
print(result)

### 实验作业（需将此部分内容写入实验报告，实验报告中的步骤从这一步开始，也就是实验 作业完成LSB图像隐写）

1. 选取文件夹中的图片“test.bmp”，将其转为灰度图。在屏幕中一行两列输出。第一行第一列输出彩色图，标注的名称为“彩色图像”，第一行第二列为灰度图像，标注的名称为“灰度图像”。**同时这个输出的名称为“本人的学号+姓名+彩色图和灰度图对比”，将输出截图出入到实验报告中。本图为图1。**
2. 在灰度图中使用LSB隐写技术隐藏秘密信息“BUPT”（大写），嵌入的位置是从第一个像素开始，顺序嵌入（嵌入多少个比特？）。要求将BUPT的ASCII码值转成二进制的bit位输出，截图插入到实验报告中，应该是32bit的秘密信息。**（需要编写代码将BUPT的ASCII码输出，输出在命令行中，输出要求为“本人的学号+姓名嵌入秘密信息的ASCII码为 ”，此图为图2. 然后将二进制的值输出，输出在命令行中，命令行的输出为应该是32bit的秘密信息，输出要求为“本人的学号+姓名嵌入秘密信息的二进制为 ”，此图为图3.**
3. 对比隐写前后的图像，在屏幕中一行两列输出。第一行第一列输出原始灰度图，标注的名称为“载体图像”，第一行第二列为嵌入秘密信息后的图像，标注的名称为“携密图像”。**同时这个输出的名称为“本人的学号+姓名+载体图和**携密图对比”**。此图为图4.**
4. 提取隐藏的LSB隐写信息，对比原信息，判断是否提取成功。要求将提取的隐藏信息32bit位全部显示出来（和第二步骤中BUPT的ASCII码值的32bit比较是否完全一致，必须完全一致），再转成文本，看提取的文本信息是不是BUPT。将提取的隐藏信息32bit位全部显示出来**，在命令行输出，命令行的输出为应该是32bit的秘密信息。然后再转成文本，输出的文本为BUPT，输出要求为：“本人的学号+姓名提取出来的32位秘密信息为 ”；此图为图5.输出要求为：“本人的学号+姓名提取出来的秘密文本为 ”；此图为图6.**
5. 计算携密图像的峰值信噪比。**峰值信噪比取小数点后四位，命令行方式输出的结果要求“本人的学号+姓名计算出来的峰值信噪比为 ”，必须有本人姓名和学号的信息。此图为图7.**

课外练习：此部分内容作为学有余力的同学课后完成（此部分 内容也不需要写入实验报告）

1. **选取文件夹中的图片“test.bmp”，将彩色彩色图转成灰度图graytest.bmp，灰度图的每个像素点的值为0-255之间。提取graytest.bmp灰度图的最低位平面，次低位平面，第三位平面，第四位平面，第五位平面，第六位平面，第七位平面，第八位平面，将这8副图像分为两行四列显示。**
2. 对**graytest.bmp**使用LSB隐写技术隐藏秘密信息“BUPT”，隐写后的灰度图为graystego.bmp，对比隐写前后的图像，在屏幕中一行两列输出。注意：隐写的载体为灰度图。
3. 提取隐藏的LSB隐写信息，对比原信息，判断是否提取成功。
4. 计算原始载体和隐藏秘密信息后的图像的峰值信噪比。