# LSB信息隐藏的卡方分析

## 【实验目的】

了解什么是隐写分析（steganalysis），隐写分析与信息隐藏和数字水印的关系。掌握基于图像的LSB隐写的分析方法，设计并实现一种基于图像的LSB卡方隐写分析方法。

## 【实验环境】

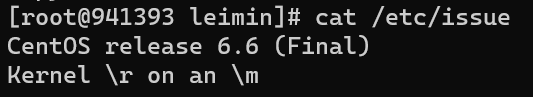
(1) Windows10或11系统

(2) Python 3.7或以上

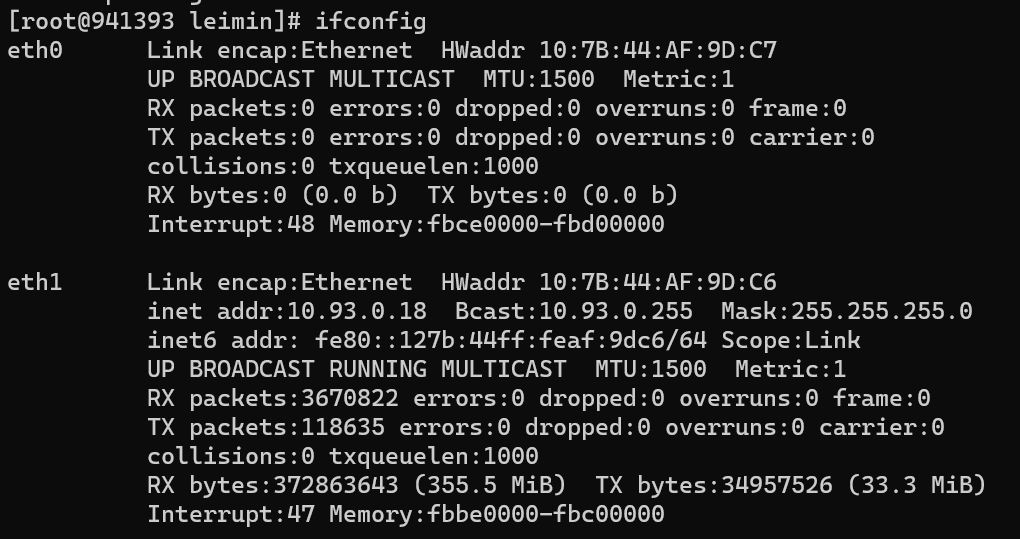
(3) 图像文件man.bmp

在本次实验中同学需要使用FTP客户端登录提供的服务器，并在服务器上下载codes.zip文件进行实验。FTP的**账号密码均为【本人学号】**

服务器的操作系统为centos6.6。



服务器的**IP地址为：10.93.0.18。**



## 【原理简介】

隐写术和隐写分析技术从本质上来说是互相矛盾的，但是两者实际上又是相互促进的。隐写分析是指对可疑的载体信息进行攻击以达到检测、破坏，甚至提取秘密信息的技术，它的主要目标是为了揭示媒体中隐蔽信息的存在性，甚至只是指出媒体中存在秘密信息的可疑性。

图像LSB信息隐藏的方法是用嵌入的秘密信息取代载体图像的最低比特位，原来图像的7 个高位平面与代表秘密信息的最低位平面组成含隐蔽信息的新图像。虽然LSB 隐写在隐藏大量信息的情况下依然保持良好的视觉隐蔽性，但使用有效的统计分析工具可判断一幅载体图像中是否含有秘密信息。

目前对于图像LSB信息隐藏主要分析方法有卡方分析、信息量估算法、RS分析法和GPC分析法等。本节介绍卡方分析方法。卡方分析的步骤是：

设图像中灰度值为j的象素数为，其中0≤ *j* ≤255。如果载体图像未经隐写，和的值会相差得很远。秘密信息在嵌入之前往往经过加密，可以看作是0、1 随机分布的比特流，而且值为0与1 的可能性都是1/2。如果秘密信息完全替代载体图像的最低位，那么和的值会比较接近，可以根据这个性质判断图像是否经过隐写。下面定量分析载体图像最低位完全嵌入秘密信息的情况。嵌入信息会改变直方图的分布，由差别很大变得近似相等，但是却不会改变的值，因为样值要么不改变，要么就在和之间改变。令 , ，显然这个值在隐写前后是不会变的。

如果某个样值为，那么它对参数q的贡献为1/2；如果样值为，对参数q的贡献为-1/2。载体音频中共有个样点的值为或，若所有样点都包含1比特的秘密信息，那么每个样点为或的概率就是0.5。当较大时，根据中心极限定理，式(6.2.1)成立。

 (式6.2.1)

其中表示近似服从正态分布。因此

 (式6.2.2)

服从卡方分布。式(6.2.2)中，k等于和所组成数字对的数量，为0的情况不计在内。r越小表示载体含有秘密信息的可能性越大。结合卡方分布的密度计算函数计算载体被隐写的可能性为：

 (式6.2.3)

如果p接近于1，则说明载体图像中含有秘密信息。

## 【实验步骤】

### 1.灰度图LSB嵌入和直方图变化

将彩色图转为灰度图，然后对灰度图像进行LSB嵌入，并比较嵌入秘密信息前后的直方图变化。

源代码lsb.py如下：

#lsb.py

#PIL图像处理库

from PIL import Image

#表格绘制库

import matplotlib.pyplot as plt

#数学库

import numpy as np

#图像的基本信息

img = Image.open("man.bmp")

width = img.size[0]

height = img.size[1]

#rgb彩色图像转灰度图

def rgb2gray(img\_):

img\_ = img\_.convert("L")

return img\_

#生成随机信息

def randomMsg(percent):

if percent>0 and percent<=1:

row = round(width \* percent)

col = round(height \* percent)

return np.random.randint(0,2,(col,row))

else:

raise Exception("传入的值必须属于(0,1]")

#将信息写入

def lsbWritein(img,msg):

for y in range(len(msg)):

for x in range(len(msg[0])):

color = img.getpixel((x,y))

temp = bin(color).replace('0b','')

#不满足8bit长度的在高位补0

for j in range(8-len(temp)):

temp = '0' + temp

temp = temp[0:7]+str(msg[y][x])

img.putpixel((x,y),int(temp,2))

return img

#主函数

def main():

plt.figure("pixel")

rt = 1

img\_gray = rgb2gray(img)

martix\_gray = np.array(img\_gray)

msg = np.array(randomMsg(rt))

img\_lsb = lsbWritein(img\_gray,msg)

martix\_lsb = np.array(img\_lsb)

#表格绘制

plt.subplot(211)

plt.title("img\_gray")

plt.hist(martix\_gray.flatten(),bins=np.arange(0.5,257))

plt.subplot(212)

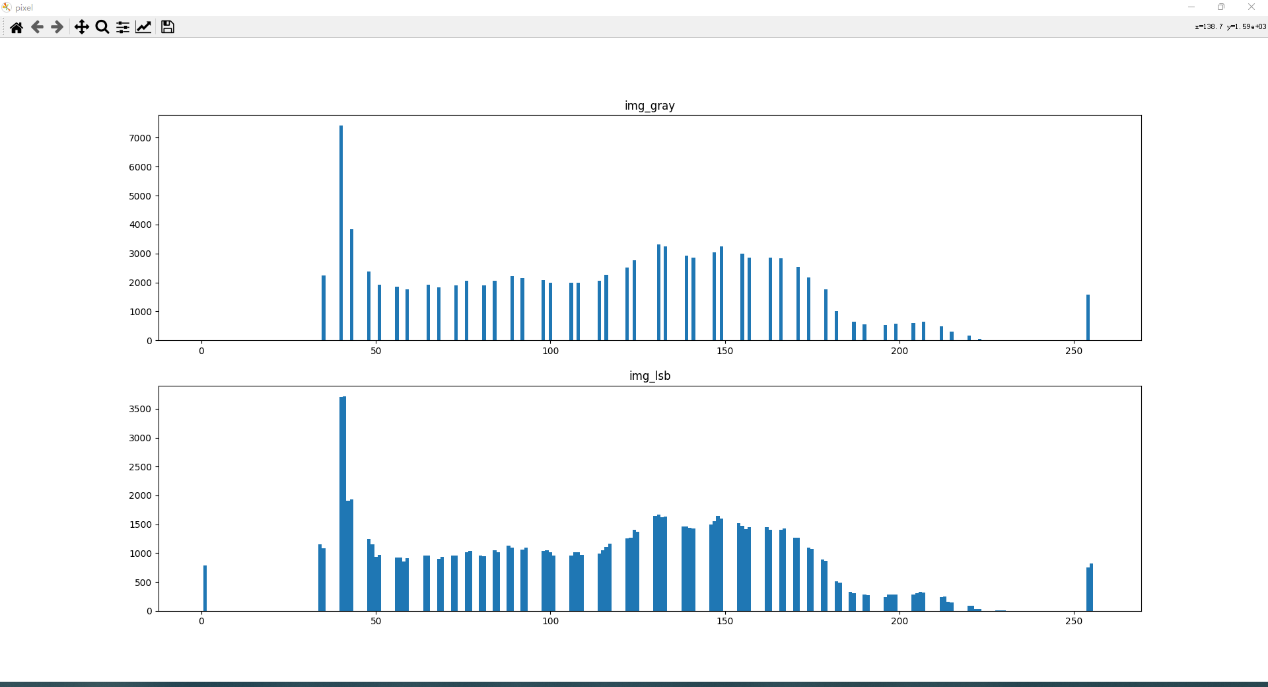
plt.title("img\_lsb")

plt.hist(martix\_lsb.flatten(),bins=np.arange(0.5,257))

plt.show()

main()

绘制的结果如下



**2.卡方分析函数**

源代码function.py如下：

#function.py

#统计学库

from scipy.stats import chi2

#数学库

import numpy as np

#卡方分析

def stgPrb(martix):

count = np.zeros(256,dtype=int)

for i in range(len(martix)):

for j in range(len(martix[0])):

count[martix[i][j]] += 1

h2i = count[2:255:2]

h2is = (h2i+count[3:256:2])/2

filter= (h2is!=0)

k = sum(filter)

idx = np.zeros(k,dtype=int)

for i in range(127):

if filter[i]==True:

idx[sum(filter[1:i])]=i

r=sum(((h2i[idx]-h2is[idx])\*\*2)/(h2is[idx]))

p = 1-chi2.cdf(r,k-1)

return p

**3.LSB卡方分析源代码**

源代码LSB卡方分析.py如下：

#LSB卡方分析.py

from PIL import Image

from function import stgPrb

import numpy as np

#图像的基本信息

img = Image.open("man.bmp")

width = img.size[0]

height = img.size[1]

#rgb彩色图像转灰度图

def rgb2gray(img):

img = img.convert("L")

return img

#生成随机信息

def randomMsg(percent):

if percent>0 and percent<=1:

row = round(width \* percent)

col = round(height \* percent)

return np.random.randint(0,2,(col,row))

else:

raise Exception("传入的值必须属于(0,1]")

#将信息写入

def lsbWritein(img,msg):

for y in range(len(msg)):

for x in range(len(msg[0])):

color = img.getpixel((x,y))

temp = bin(color).replace('0b','')

#不满足8bit长度的在高位补0

for j in range(8-len(temp)):

temp = '0' + temp

temp = temp[0:7]+str(msg[y][x])

img.putpixel((x,y),int(temp,2))

return img

#主函数

def main():

p = np.zeros((3,91))

for k in range(3):

img\_gray = rgb2gray(img)

**#根据隐写率大小生成秘密信息，隐写率为0.3,0.5,0.7三种**

rt = 0.3 + 0.2 \* k

msg = randomMsg(rt)

#lsb隐写

img\_lsb = lsbWritein(img\_gray,msg)

img\_lsb.save("man\_{}%.bmp".format(rt\*100))

martix = np.array(img\_lsb)

#循环，确定一个隐写率区间对图片进行分析

i = 0

for rto in range(10,101):

row = round(width \* (rto/100))

col = round(height \* (rto/100))

p[k][i] = stgPrb(martix[0:row,0:col])

i+=1

#输出

for i in range(3):

for j in range(91):

print(p[i][j],end=',')

print()

main()

## 【实验作业】

1. 利用FTP客户端登录服务器下载代码，登录服务器和下载代码的过程请截图插入。每个同学的账号没有删除权限。
2. 在灰度图man.bmp利用LSB方法随机嵌入0和1 的比特，嵌入率为100%，后生成mangraystego.bmp，在一行两列中输出mangray.bmp和mangraystego.bmp，截图插入到实验报告中。
3. 并计算隐写后的图像的峰值信噪比，从主观（视觉效果）和客观（峰值信噪比）上来查看隐写后图像的透明性。将峰值信噪比的结果截图插入到实验报告中。
4. 比较嵌入秘密信息前后的直方图变化，将直方图显示在屏幕上，截图插入到实验报告中。直方图输出的时候要求输出灰度值为40-60的灰度值，不需要输出全部的0-255的灰度值的数量。
5. 利用卡方分析方法判断隐写后的图像是否隐藏秘密信息，计算p，将结果截图插入到实验报告中。要求是学号+姓名+卡方分析得到的P值为：
6. 将本人实验代码压缩成为一个目录，目录名为自己的学号，上传到服务器本人的目录下，并截图证明自己上传成功。

## 【思考题】

如果秘密信息没有嵌满所有像素，并且嵌入位置随机分布于整个图像，而分析者很难知道秘密信息所在位置，卡方分析方法就很难生效。可以使用伪随机序列随机在载体的最低比特位嵌入信息，使用上述卡方分析方法判断载体是否携密，验证卡方分析方法在上述隐写算法情况下是否有效。

## 【课后练习】