# WAV音频的LSB信息隐藏

## 【实验目的】

了解WAV文件格式，掌握利用WAV格式音频文件实现LSB信息隐藏原理，设计并实现一种基于WAV文件的LSB信息隐藏算法，同时自行练习设计实现一种次第有效位的音频隐写算法。

了解归一化相关系数NC的原理，利用NC对嵌入的水印图像和提取的图像水印进行比较。

## 【实验环境】

(1) Windows或Linux操作系统

(2) Python3环境

(3) Python的wave、numpy、matplotlib、opencv-python库

(4) wave音频文件和二值水印图像文件

## 【原理简介】

WAV是Microsoft Windows本身提供的音频格式，该格式通常都保存一些没有压缩的音频。对于数字音频，其最低比特位或者最低几个比特位的改变，对于整个声音没有明显的影响，因此替换掉这些不重要的部分，可以隐藏秘密信息。

如果在音频信号中嵌入的水印为图像，则来定义评价提取的水印与原始水印的相似性，可采用归一化相关系数（NC）作为评价标准，其定义为

 (式4.1.1)

(式4.1.1)式中：W为原始水印，为提取的水印，它们的大小为。

## 【实验步骤】

### （1）随机水印

**1.隐藏算法**

源代码hide.py如下：

# 导入wave音频文件处理库

import wave

# 导入数学计算库

import numpy as np

# 导入绘图库

import matplotlib.pyplot as plt

# 读取载体音频

wav = wave.open('carrier.wav', 'rb')

nchannels, sampwidth, framerate, nframes, comptype, compname = wav.getparams()

time = nframes / framerate

# 以字节方式读取载体音频的数据

frames = wav.readframes(nframes)

# 产生随机的80比特长的水印数据

wm\_random = np.random.randint(0, 2, 80)

print(f'Random Array: {wm\_random}')

wav\_embedded = wave.open('embedded.wav', 'wb')

wav\_embedded.setparams(wav.getparams())

# 将字节数据转换为numpy数组

data = np.frombuffer(frames, dtype=np.uint8)

**# LSB嵌入水印**

**data\_embedded = data.copy()**

**for i in range(len(wm\_random)):**

**data\_embedded[i] -= data\_embedded[i] % 2**

**data\_embedded[i] += wm\_random[i]**

**# 次低位嵌入水印  
data\_embedded = data.copy()  
for i in range(len(wm\_random)):  
data\_embedded[i]-=data\_embedded[i]%4-data\_embedded[i]%2  
data\_embedded[i]+=wm\_random[i] \* 2**

**# 次低有效位嵌入水印  
data\_embedded = data.copy()  
for i in range(len(wm\_random)):  
    bit = 1  
    data\_embedded[i] &= ~(1 << bit)  
    data\_embedded[i] |= wm\_random[i] << bit**

# 写入音频数据

wav\_embedded.writeframes(data\_embedded)

# 展示原音频和嵌入音频的波形

plt.figure(figsize=(14, 6))

plt.subplot(121)

plt.plot(data)

plt.title('Original Audio')

plt.xticks([]), plt.yticks([])

plt.subplot(122)

plt.plot(data\_embedded)

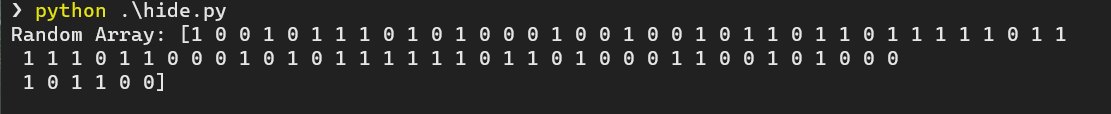
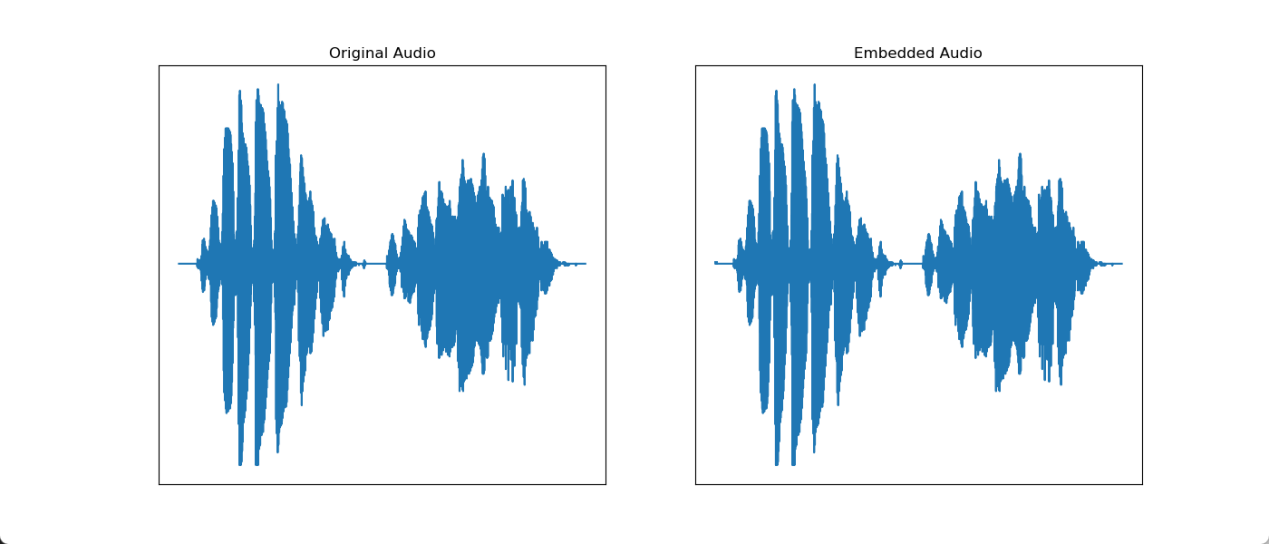
plt.title('Embedded Audio')

plt.xticks([]), plt.yticks([])

plt.show()

## 实验结果

完成LSB嵌入之后，首先对LSB嵌入前后的音频文件进行听觉上的区分，二者靠人耳听不出任何差别。下图是原始音频波形图和携秘音频波形图，从这两幅图的对比中可以看出， LSB信息隐藏后，对原始音频的波形影响也非常小，几乎看不出任何差别。LSB隐藏的音频透明性非常好。



**2.随机水印提取**

源代码extract.py如下：

# 导入wave音频文件处理库

import wave

# 导入数学计算库

import numpy as np

# 读取携密音频

wav = wave.open('embedded.wav', 'rb')

nchannels, sampwidth, framerate, nframes, comptype, compname = wav.getparams()

time = nframes / framerate

# 以字节方式读取携密音频的数据

frames = wav.readframes(nframes)

# 将字节数据转换为numpy数组

data = np.frombuffer(frames, dtype=np.uint8)

# LSB提取水印，提取前80比特

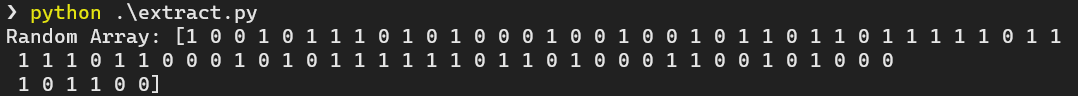
wm = np.zeros(80, dtype=np.uint8)

for i in range(len(wm)):

wm[i] = data[i] % 2

print(f'Random Array: {wm}')

## 实验结果



### （2）图像水印

**1.图像水印隐藏**

水印信息为二值图像bupt64.bmp，图像的大小为64\*64，共4096个像素，原始载体是音频，从音频中截取4096个字节，每个字节中隐藏一个比特。

源代码hide.py如下：

# 导入wave音频文件处理库

import wave

# 导入图像处理库

import cv2

# 导入数学计算库

import numpy as np

# 导入绘图库

import matplotlib.pyplot as plt

# 读取载体音频

wav = wave.open('carrier.wav', 'rb')

nchannels, sampwidth, framerate, nframes, comptype, compname = wav.getparams()

time = nframes / framerate

# 以字节方式读取载体音频的数据

frames = wav.readframes(nframes)

# 以灰度图模式读取水印图像

wm = cv2.imread('bupt64.bmp', cv2.IMREAD\_GRAYSCALE)

# 从二维矩阵转为一维并二值化

wm = wm.flatten() > 128

#对numpy数组使用运算符比较特殊，他会对数组中的每个元素

#都进行运算，比如[0,0,0,0]+1，结果就是[1,1,1,1]  
#进行大小比较也一样，[0,0,255,255] > 128 的结果就

#是 [False, False,True, True]

#.flatten()就是将二维转换为一维  
#比如[[0,1],[2,3]].flatten()就是[0,1,2,3]

wav\_embedded = wave.open('embedded.wav', 'wb')

wav\_embedded.setparams(wav.getparams())

# 将字节数据转换为numpy数组

data = np.frombuffer(frames, dtype=np.uint8)

# LSB嵌入水印

data\_embedded = data.copy()

for i in range(len(wm)):

data\_embedded[i] -= data\_embedded[i] % 2

data\_embedded[i] += wm[i]

# 写入音频数据

wav\_embedded.writeframes(data\_embedded)

# 展示原音频和嵌入音频的波形

plt.figure(figsize=(14, 6))

plt.subplot(121)

plt.plot(data)

plt.title('Original Audio')

plt.xticks([]), plt.yticks([])

plt.subplot(122)

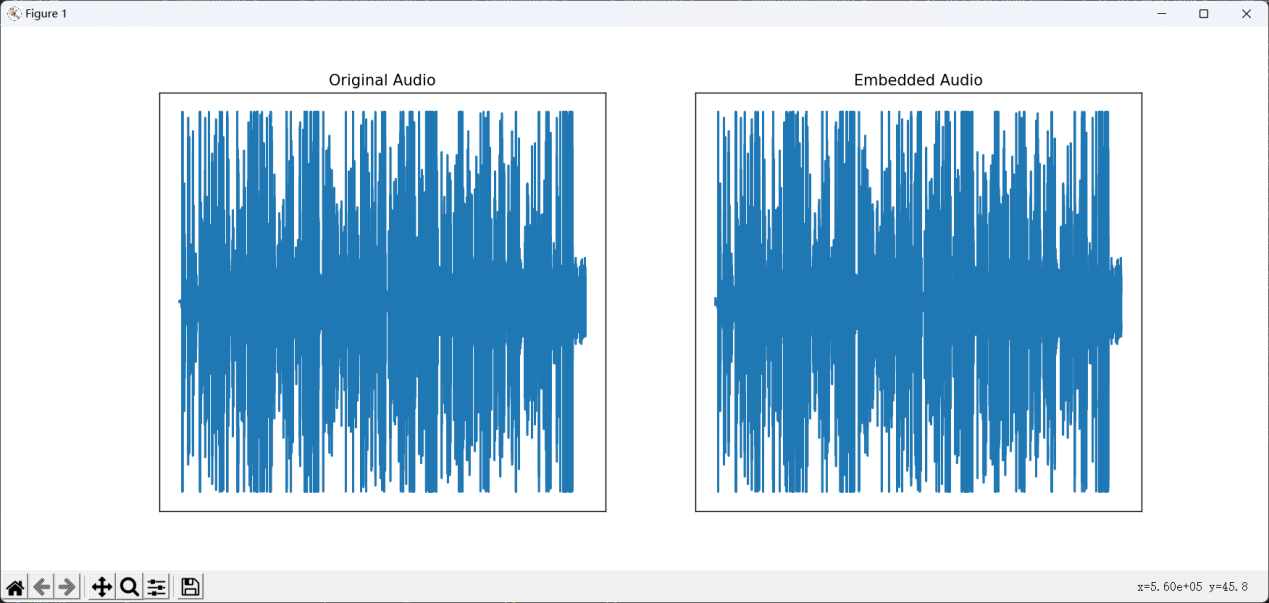
plt.plot(data\_embedded)

plt.title('Embedded Audio')

plt.xticks([]), plt.yticks([])

plt.show()

## 实验结果



**2.图像水印提取**

在实践过程中，含有水印信息的音频信号从编码到解码之间可能有很多传播途径，主要有以下的四种方式：

1. 声音文件从一台机器拷贝到另外一台机器，其中没有任何形式的改变。编码方和解码方的采样率完全相同；
2. 第二种情形是信号仍然保持数字的形式，但是采样率发生变化。这一变化保持大多数信号的幅值和相位值，但是改变了信号的时域特征；
3. 第三种情况是信号被转换成为模拟形式，通过模拟形式传送，在终端被重新采样。在此过程中信号的幅值、量化方式和时域采样都得不到保持，这种情况下信号的相位值可以得到保持；
4. 第四种情形是信号在空气中传播，经过麦克重新采样。此时信号受到未知的非线性改变，会导致相位变化、幅值变化、不同频率成分的漂移和产生回声等；

当某一段音频文件嵌入水印后以某种方式传播，到达终端的时候会发生一些变化。提取水印后和原始水印进行比较，采用归一化函数计算提取的水印信息和原始水印信息之间的差别。

提取水印的源代码extract.py如下：

# 导入wave音频文件处理库

import wave

# 导入图像处理库

import cv2

# 导入数学计算库

import numpy as np

# 导入绘图库

import matplotlib.pyplot as plt

# 计算NC值的函数

def NC(template, img):

template = template.astype(np.uint8)

img = img.astype(np.uint8)

return cv2.matchTemplate(img, template, cv2.TM\_CCORR\_NORMED)[0][0]

# 设置水印图像的宽高

wm\_height = 64

wm\_width = 64

# 读取携密音频

wav = wave.open('embedded.wav', 'rb')

nchannels, sampwidth, framerate, nframes, comptype, compname = wav.getparams()

time = nframes / framerate

# 以字节方式读取携密音频的数据

frames = wav.readframes(nframes)

# 将字节数据转换为numpy数组

data = np.frombuffer(frames, dtype=np.uint8)

# LSB提取水印

wm = np.zeros(wm\_height \* wm\_width, dtype=np.uint8)

for i in range(len(wm)):

wm[i] = data[i] % 2 \* 255

# 从一维转为二维矩阵

wm = np.reshape(wm, (wm\_height, wm\_width))

# 以灰度图模式读取水印图像

wm\_original = cv2.imread('bupt64.bmp', cv2.IMREAD\_GRAYSCALE)

# 计算NC值

nc = NC(wm\_original, wm)

print(f'NC = {nc \* 100} %')

# 保存提取出的水印图像

cv2.imwrite('wm.bmp', wm)

# 展示嵌入图像、水印原图像和提取出的水印图像

plt.figure(figsize=(15, 6))

plt.subplot(131)

plt.plot(data)

plt.title('Embedded Audio')

plt.xticks([]), plt.yticks([])

plt.subplot(132)

plt.imshow(wm\_original, 'gray')

plt.title('Original Watermark')

plt.xticks([]), plt.yticks([])

plt.subplot(133)

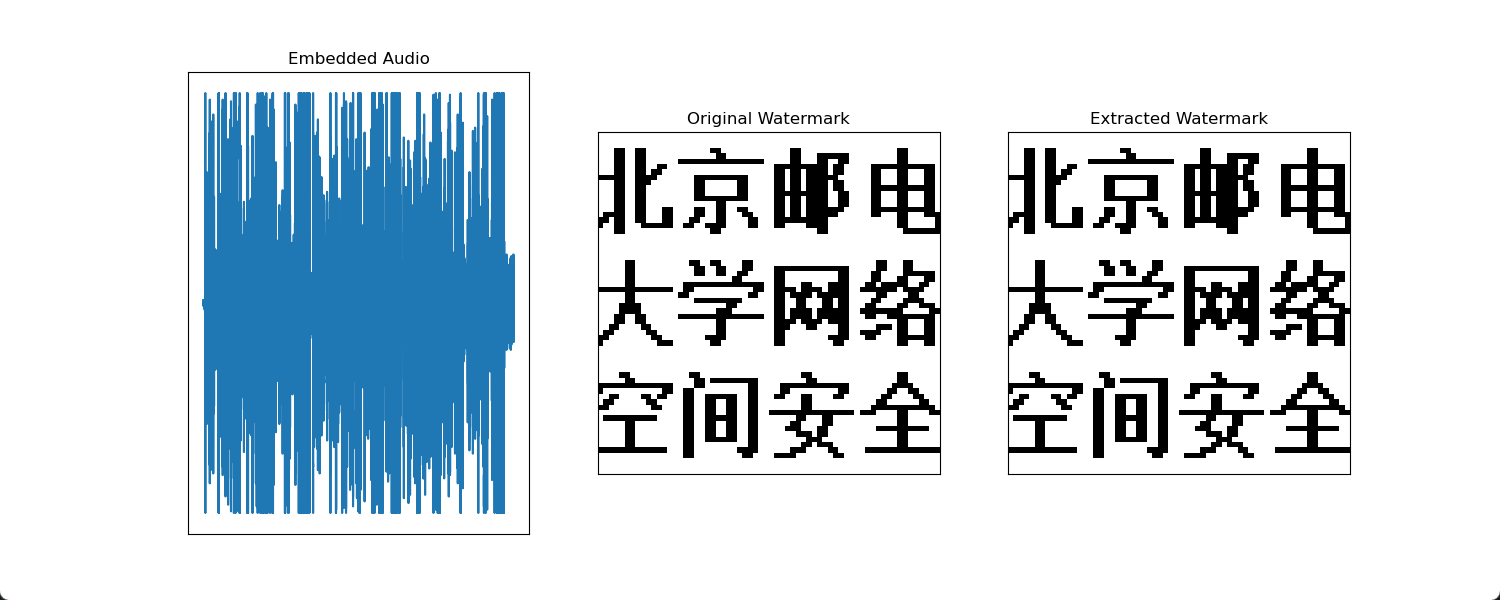
plt.imshow(wm, 'gray')

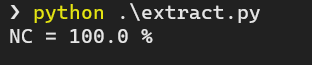
plt.title('Extracted Watermark')

plt.xticks([]), plt.yticks([])

plt.show()

## 实验结果





### (3)计算NC

图像水印提取后和原始水印进行归一化比较。在本例中，携密音频未发生任何变化，也就是说未对携密音频进行任何形式的攻击。因此提取出来的水印信息和原始的水印信息完全相同，归一化函数的值为1。

归一化函数的函数NC如下：

def NC(template, img):

template = template.astype(np.uint8)

img = img.astype(np.uint8)

return cv2.matchTemplate(img, template, cv2.TM\_CCORR\_NORMED)[0][0]

## 【实验作业】

（1）利用载体音频river.wav隐藏嵌入秘密信息bupt64.bmp图像，采取次低有效位嵌入，不是在最低有效，而是在倒数第二有效位嵌入。

（2）在一行两列中输出原始音频和携密音频的波形图。

（3）在没有任何攻击的情况下，从嵌入秘密信息后的音频中提取水印图像，在一行两列中输出原始图像水印和提取的水印图像。比较原始水印图像和提取水印图像的NC的值，这个NC的值应为为1。

（4）撰写实验报告，并提交实验报告。

## 【思考题】

1.将某个二值图像利用LSB算法嵌入到语音载体中，将语音载体添加高斯噪声，添加高斯噪声后将水印信息提取，使用归一化函数计算提取的水印信息和原始水印信息之间的差别。

2.LSB方式信息隐藏算法鲁棒性较差，为加大水印攻击的难度，可采用一段伪随机序列来控制水印的嵌入位置。伪随机序列可由伪随机序列发生器来产生，不同的初始值产生不同的伪随机序列。这样收发两方只需要秘密地传送一个初始值（也就是密钥），就可得到一个相同的伪随机序列而不需要传送整个的伪随机序列。

3.LSB信息隐藏替换技术可分为五种处理情况：

* + 秘密信息在最低位平面连续嵌入至结束，余下部分不作任何处理，比较典型的软件是MandelSteg；
  + 秘密信息在最低位平面连续嵌入至结束，余下部分随机化处理（也称沙化处理），比较典型的软件是PGMStealth；
  + 秘密信息在最低位平面和次底位平面连续嵌入，同时嵌入到最低位平面和次底位平面；
  + 秘密信息在最低位平面嵌入，等最低位平面完全潜入后，再开始嵌入到次底位平面；
  + 秘密信息在最低位平面随机嵌入；

不同的嵌入方式，隐藏的信息容量不同，而且这几种算法鲁棒性差别也比较大。可自行设计实现这几种基于音频的LSB隐藏。