LSB 图像隐写与提取算法

原创

iO快到碗里来 于 2021-10-17 00:23:24 发布 2267 收藏 9

分类专栏： 信息内容安全 文章标签： 图像处理 信息安全

0x00 信息内容安全实验报告

【实验目的】

了解信息隐藏中最常用的LSB算法特点，掌握LSB算法原理，设计并实现一种基于图像的LSB隐藏算法；了解如何通过峰值信噪比来对图像质量进行客观评价，并计算峰值信噪比。

【实验环境】

(1) Windows XP 或 Vista 操作系统；

(2) Matlab7.1 科学计算软件；

(3) BMP 灰度图像文件。

【原理简介】

任何多媒体信息，在数字化时，都会产生物理随机噪，而人的感观系统对这些随机噪声不敏感。信息隐藏技术就是利用这个原理，通过使用秘密信息比特替换随机噪声，从而完成信息隐藏目标。

BMP 灰度图像的每个像素值为 8bit 二进制值，表示该点亮度。不同位平面对视觉影响不同，图像高位平面对图像感官质量起主要作用，去除图像最低位平面并不会造成画面质量的明显下降。利用这个原理可用秘密信息(或称水印信息)替代载体图像低位平面以实现信息嵌入。

本实验中算法选用最低位平面来嵌入秘密信息。最低位平面对图像的视觉效果影响最轻微，但很容易受噪声影响和攻击，解决办法可采用冗余嵌入的方式来增强稳健性。即在一个区域(多个像素)中嵌入相同的信息，提取时根据该区域中的所有像素判断。

因为暂时还没来得及学习 Matlab 的使用，所以先简单地用 python 实现一下，且以 RGB 图像为例。

0x01 Hide

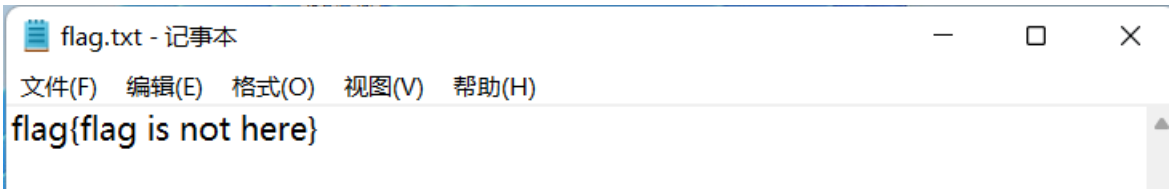
```
from PIL import Image

def mod(x,y):
    return x % y

def bin_ord(flag):
 string = ""
 with open(flag) as f:
  txt = f.read()
  for i in range(len(txt)):
   string = string + bin(ord(txt[i])).replace('0b','').zfill(8)
 return string

def hide(pic,flag,new_pic):
 count = 0
 im = Image.open(pic)
 width = im.size[0]
 height = im.size[1]
 string = bin_ord(flag)
 for h in range(height):
  for w in range(width):
   pixel = im.getpixel((w,h))
   x = pixel[0]
   y = pixel[1]
   z = pixel[2]
   if count == len(string):
    break
   x = x - mod(x,2) + int(string[count])
   im.putpixel((w,h),(x,y,z))
   count = count + 1
   if count == len(string):
    break
   y = y - mod(y,2) + int(string[count])
   im.putpixel((w,h),(x,y,z))
   count = count + 1
   if count == len(string):
    break
   z = z - mod(z,2) + int(string[count])
   im.putpixel((w,h),(x,y,z))
   count = count + 1
 im.save(new_pic)

pic = r"C:\Users\lyz\Desktop\test.png"

flag = r"C:\Users\lyz\Desktop\flag.txt"

new_pic = r"C:\Users\lyz\Desktop\new.png"

hide(pic,flag,new_pic)
```

虽然标准 ASCII 是 7 位编码，但由于计算机基本处理单位为字节（ 1byte = 8bit ），所以一般仍以一个字节来存放一个 ASCII 字符。

隐写前的图片：

隐写后的图片：

隐写的内容：



0x02 Extract

```
from PIL import Image

def mod(x,y):
 return x % y

def extract(pic,lenth,hide):
 binary = ""
 string = ""
 count = 0
 im = Image.open(pic)
 width = im.size[0]
 height = im.size[1]
 for h in range(height):
  for w in range(width):
   pixel = im.getpixel((w,h))
   x = pixel[0]
   y = pixel[1]
   z = pixel[2]
   if count == lenth:
    break
   binary = binary + str(mod(x,2))
   count = count + 1
   if count == lenth:
    break
   binary = binary + str(mod(y,2))
   count = count + 1
   if count == lenth:
    break
   binary = binary +str(mod(z,2))
   count = count + 1
 with open(hide,"w",encoding="UTF-8") as f:
  for i in range(0,len(binary),8):
   string = string + chr(int(binary[i:i+8],2))
  f.write(string)

pic = r"C:\Users\lyz\Desktop\new.png"

lenth = 200

hide = r"C:\Users\lyz\Desktop\hide.txt"

extract(pic,lenth,hide)
```

提取的内容：

0x03 PSNR

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。
峰值信噪比经常用作图像压缩等领域中信号重建质量的测量方法，它常简单地通过均方误差（MSE）进行定义。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，那么它们的均方误差定义为：

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

峰值信噪比定义为：

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right),$$

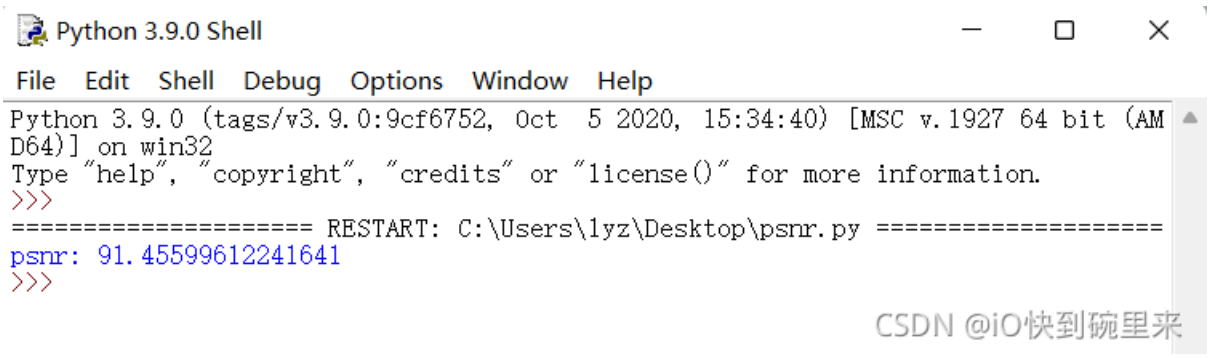
其中，MAXI是表示图像点颜色的最大数值，如果每个采样点用 8 位表示，那么就是 255。更为通用的表示是，如果每个采样点用B位线性脉冲编码调制表示，那么MAXI就是：

$$2^B - 1.$$

对于每点有RGB三个值的彩色图像来说峰值信噪比的定义类似，只是均方误差是所有方差之和除以图像尺寸再除以3。
图像压缩中典型的峰值信噪比值在 30 到 40dB 之间，愈高愈好。

```python
import cv2 as cv
import math
import numpy as np


def psnr(old_pic,new_pic):
 mse = np.mean((old_pic/1.0-new_pic/1.0)**2)
 if mse < 1e-10:
  return 100
 psnr = 20*math.log10(255/math.sqrt(mse))
 return psnr

old_pic = cv.imread(r"C:\Users\lyz\Desktop\test.bmp")

new_pic = cv.imread(r"C:\Users\lyz\Desktop\new.bmp")

if __name__ == '__main__':
 res= psnr(old_pic,new_pic)
 print("psnr:",res)
```

运行结果如下：



0x04 Summary

网上的图片大多数都是 jpg 格式。在实验过程中，jpg 后缀的图片无法完成隐写，而将图片后缀改为 png 或 bmp 后，均能成功完成隐写。

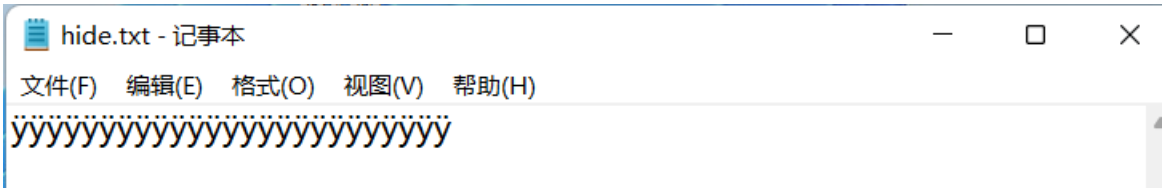
格式	压缩模式	透明支持
JPG	有损压缩	不支持
PNG	无损压缩	支持
BMP	无压缩	不支持

JPG：使用的一种失真压缩标准方法，24 bit真彩色，不支持动画、不支持透明色。JPEG的压缩方式通常是破坏性资料压缩（lossy compression），即在压缩过程中图像的品质会遭受到可见的破坏。一张图片多次上传下载后，图片逐渐会失真。

PNG：格式是无损数据压缩的，PNG格式有8位、24位、32位三种形式，其中8位PNG支持两种不同的透明形式（索引透明和alpha透明），24位PNG不支持透明，32位PNG在24位基础上增加了8位透明通道（32-24=8），因此可展现256级透明程度。

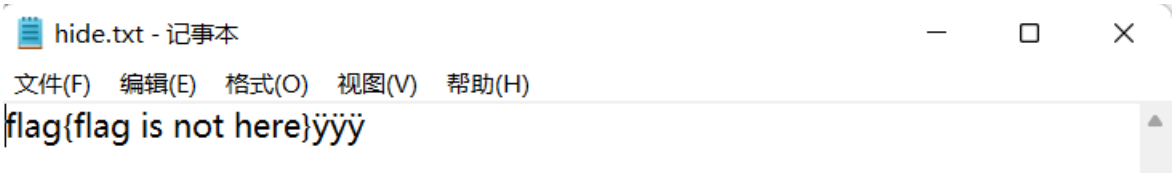
BMP：是一种与硬件设备无关的图像文件格式，使用非常广。它采用位映射存储格式，除了图像深度可选以外，不采用其他任何压缩，因此，BMP图像所占用的空间很大。BMP文件的图像深度可选lbit、4bit、8bit及24bit。BMP文件存储数据时，图像的扫描方式是按从左到右、从下到上的顺序。

JPG:


hide.txt - 记事本
文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

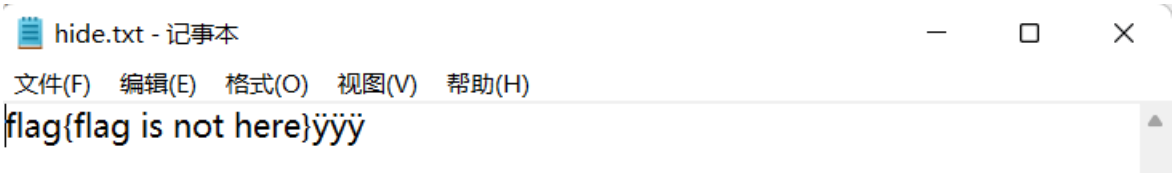
PNG:


new.png 属性
常规 安全 详细信息 以前的版本
new.png
文件类型: PNG 文件 (.png)
打开方式: Windows 照片查看器
更改(C)...
位置: C:\Users\lyz\Desktop
大小: 375 KB (384,732 字节)
占用空间: 376 KB (385,024 字节)
CSDN @iO快到碗里来


hide.txt - 记事本
文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)
flag{flag is not here}ÿÿÿ

BMP:


new.bmp 属性
常规 安全 详细信息 以前的版本
new.bmp
文件类型: BMP 文件 (.bmp)
打开方式: 照片
更改(C)...
位置: C:\Users\lyz\Desktop
大小: 1.84 MB (1,935,414 字节)
占用空间: 1.84 MB (1,937,408 字节)
CSDN @iO快到碗里来



LSB算法

.

全称为Least Significant Bit，在二进制数中意为最低有效位，一般来说，MSB(最高有效位)位于二进制数的最左侧，LSB位于二进制数的最右侧。

.

由于图像的每一个像素点都是由RGB（红、绿、蓝）三原色组成，而这三种颜色又可以组合成各种其它颜色，每个颜色占8位(如#FFFFFF)，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。

.

一个像素点包含三种颜色，每个颜色修改最后1位，这样一个像素点就可以携带3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，如LSB算法多应用于png、bmp等格式，而jpg格式较少。

.

详细参考：https://wenku.baidu.com/view/ff590e9d5f0e7cd1842536d7.html

.

Python Imaging Library

.

Python Imaging
Library（简称PIL）为Python解释器提供了图像处理的功能，PIL提供了广泛的文件格式支持、高效的内部表示以及相当强大的图像处理功能。PIL图像处理库的核心被设计成为能够快速访问以几种基本像素类型表示的图像数据，它为通用图像处理工具提供了一个坚实基础。

.

结合PIL可以方便的编写Python脚本处理图片隐写问题。

.

StegSolve

.

StegSolve是一款基于Java开发的流行图片隐写分析软件，其支持常见的图片文件格式，可以对不同的文件进行结合（包括XOR、ADD、SUB等操作），可以对图片文件格式进行分析，可以提取GIF文件中的帧等，覆盖了基本的图片隐写分析需求。