信息隐藏实验报告

Outlines

1. 算法描述
2. 系统实现
3. 测试结果
4. 结果分析
5. 实验评价和小结

算法描述

lsb算法是通过替换bmp低位为秘密信息，原图像的7个高位平面和秘密信息共同组成含隐蔽信息的新图像。由于像素的最低位对视觉的影响很小，LSB可以通过这种方法实现较好的视觉隐蔽性。

隐写分析是对可疑的载体进行分析检测，判断是否可能存在秘密信息甚至提取秘密信息的技术。在传统的隐写-检测模型中，Warden负责检测Alice向Bob传递的载体。

目前针对LSB隐藏技术的主要分析方法有chi-square分析、信息量估计、RS分析和GPC分析等。本次试验主要实现LSB隐写和针对LSB的chi-square分析和RS分析。

Preliminaries

载体图片为512*512的灰阶bmp图片，深度为8位，即像素值$x_{i,j} \in [0, 255)$

Chi-square Analysis

设图像中像素值为x的像素数量为h_x，其中$x \in [0, 256), h_x \in [0, 512 * 512]$。一般而言，未经隐写的图片的像素值分布情况比较固定，即h_{2i}和h_{2i+1}的值差距很大，其中$i \in Z$，$2i + 1 < 256$。而如果将嵌入秘密视作随机的01比特流（在秘密信息长度足够长的情况下），则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$h_{2i} + h_{2i+1}$的值,因为像素值要么不改变，要么就在h_{2i},和h_{2i+1}之间改变。嵌入后h_{2i}和h_{2i+1}的比较关系将会发生很明显的变化，即$h_{2i} \approx h_{2i+1}$。通过衡量值对的“相同性”，可以估计出图片经过LSB隐写的可能性，这就是chi-square analysis的核心思想。

令

$$h_{2i}^* = (h_{2i} + h_{2i+1})/2$$
$$q = (h_{2i} - h_{2i+1})/2$$

则h_{2i}^*在隐写前后不变,而q会在隐写后变小。接下来量化值对的相同性。如果像素的位置为i，则对q的贡献为$1/2$；否则为$-1/2$。假设每个像素都包含1bit的敏感信息，则所有对于所有灰度值为$2i$和$2i + 1$的点，包含0或者1的概率是均等的，和i的取值无关。而未嵌入的自然图片可以认为其LSB的值和高位存在相关性，并不存在均匀分布的性质。

接下来进行量化分析。当每个像素都包含1bit的敏感信息并具有上述性质时，根据中心极限定理：

$$\frac{h_{2i} - h_{2i+1}}{\sqrt{2h_{2i}^*}} = \sqrt{2} \cdot \frac{h_{2i} - h_{2i}*}{\sqrt{h_{2i}^*}} \sim N(0, 1)$$

则：

$$r = \sum \frac{(h_{2i} - h_{2i}^*)^2}{h_{2i}^*}$$

服从卡方分布，则载体被嵌入的可能性估计值为

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\gamma} \exp(-t/2) t^{\frac{k-1}{2}-1} dt$$

如果p接近1，则说明载体很大可能有秘密信息。

Chi-square分析基于一个统计学的假设：如果图片被LSB嵌入，则被嵌入的像素为的最低位均匀分布，即这些像素的值在$(2i, 2i+1)$上服从均匀分布且和i无关。所以在统计所有像素的相关性时，如果被嵌入的像素占比越高，统计效果越好。因此，chi-square分析在嵌入率低时效果不好。

RS Analysis

RS分析是对LSB隐写的一种新的算法。RS算法在较低嵌入率的情况下也能得出正确的结果，同时还可以估计嵌入率。

RS分析是基于像素间的相关性进行分析的。对于相邻的若干个像素，定义平滑度量化函数：

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|$$

该函数定义了像素组的平滑度。f越小，说明像素之间的空间相关度越强。如果图像经过了LSB隐写，一般情况下f值会显著增加。

定义三种变换：

$$F_1(x) = i \leftrightarrow i+1$$
$$F_{-1}(x) = i-1 \leftrightarrow i$$
$$F_0(x) = x$$

定义：

- R组为变换后平滑度上升的像素组
- S组为变换后平滑度下降的像素组
- R_M为经过$+M$变换后R组的数量
- S_M为经过$+M$变换后S组的数量
- R_{-M}, S_{-M}相同定义

如果图片经过了LSB隐写，则对图片进行F1翻转与进行F-1翻转会对图片平滑度造成不同的影响，进而导致R_M, S_M, R_{-M}, S_{-M}的不对称变化，由此可以推断出进行了隐写。F_1隐写相当于在最低位添加了噪声，但对于像素的影响最多为1，如果引入了F_{-1}变换作为对比，则噪声引入像素的影像提升至2.

如果待检测图像没有经过LSB隐写，则对像素进行的F_1还是F_{-1}，都会同等程度增加图像块的混乱程度，即如果$R_{-M} - S_{-M} > R_{+M} - S_{+M}$，则判断载体有进行LSB隐写，否则没有。

系统设计与测试

本次实验实现一个基于命令行的LSB隐写/分析工具。源码已开源于iridium-soda/Steg_Experiment_Master_HUST: Information Hiding and Digital Watermarking Experiment of Master, HUST, Cyber Security, 2022 (github.com)

主要功能和命令参数说明如下：

```
usage: main.py [-h] [-e | -c | -r] path

Process some images. Embed, chi-square analysis and LS
analysis.

positional arguments:
  path         Input or output path

options:
  -h, --help   show this help message and exit
  -e, --embed  Steg image by LSB
  -c, --chi    Analysis image by chi-square
  -r, --rs     Analysis image by RS
```

- `-e`对`path`指向的路径图片进行LSB隐写，并输出为`embed_{embed rate}_{timestamp}.bmp`
- `-c`使用卡方分析`path`指向的路径图片，输出p值

- `-r`使用RS分析`path`指向的路径图片，输出$R_{-M} - S_{-M} > R_{+M} - S_{+M}$并判断是否嵌入

- `main.py`：程序主入口，初始化module并使用`argparse`解析参数
- `bmputil.py`：实现一个读取并操作bmp图像的类
- `logger.py`：日志相关初始化配置
- `lsb.py`：LSB隐写相关代码
- `RSanalysis.py`：RS分析相关代码
- `chisquare.py`：卡方分析相关代码

关键函数流程简介

LSB

1. 实例化`BmpUtil`类，读取原图片
2. 设定嵌入率`embed_rate`和随机种子
3. 生成长度为像素数*嵌入率的随机0/1流
4. 逐行进行嵌入，通过下面的计算：

```
img.image[img_index_row, img_index_col] =
img.image[img_index_row, img_index_col] >> 1 << 1 |
int(b)
```

5. 嵌入后的像素矩阵写入新图片

Chi-square

1. 实例化`BmpUtil`类，读取待分析图片
2. 计数h_{2i}和h_{2i+1}
3. 计算h^*_{2i}
4. 计算r：

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

5. 计算p：

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

RS

1. 实例化`BmpUtil`类，读取待分析图片
2. 设定组大小`n=4`，mask矩阵`mode=[1,0,0,1]`
3. 划分像素到组中
4. 对每个组按照mask进行F_1或F_{-1}操作
5. 反转mask，对原分组进行F_1或F_{-1}操作
6. 分别计算第4步和第5步产生的矩阵的每一组的光滑度
7. 计数，计算参数R和参数S
8. 计算$(R_{-M} - S_{-M}) - (R_{+M} - S_{+M})$，得出结果

测试结果

程序测试以标准512Lena灰度图作为素材。

LSB嵌入

```
python main.py -e "512lena.bmp"
```

嵌入率在源代码中手动修改，随机密文流的生成种子也在代码中指定：

```
# lsb.py
def embed(path: str):
    #...
    embed_rate = 1
    random.seed(1024)  # Set seed as 1024
```

输出下面的日志信息，输出嵌入后的图片：

```
- INFO - Embed finished:./embed_1_1668563775.bmp
```

Chi-square分析

对嵌入后的图片进行chi-square分析：

```
 python main.py -c embed_1_1668563775.bmp
```

程序输出计算后的P值：当p接近1的时候可以认为图片经过隐写。

```
- INFO - Calculated p value is 0.9999999379217502
```

RS分析

对嵌入后的图片进行RS分析：

```
 python main.py -r embed_1_1668563775.bmp
```

程序输出计算后的R值：经检验$R_{-M} - S_{-M} > R_{+M} - S_{+M}$，判断为有嵌入信息。

```
- INFO - RS Analysis:   R+:16381        R-:29021
S+:16422        S-:9243
- INFO - Analysis finished: embedding approved
```

结果分析、实验评价和小结

RS分析和卡方分析都是用于分析LSB隐写的方法。卡方分析对于低嵌入率性能较差，而RS分析不仅可以处理嵌入率较低的场景，还可以估算大概的嵌入率。而且整个过程RS分析更加简易。两种方法都存在一个基础假设，即嵌入的密文信息是随机的二进制流，0和1的出现频率接近0.5-0.5 。如果密文的规模较小，或者0/1有出现上的特点，这个假设将不能成立。

RS分析

影响RS检测能力（包括误报、漏报等情况）等主要因素有下面几点：

1. 嵌入率。一般嵌入率越高，检验等准确度越高。
2. 掩码的设置。根据实验得出的经验性结论，当$n = 4$时掩码取`[0,1,1,0]`效果最好。
3. 嵌入信息的位置。当信息接近随机分布时，分析的效果较好；但当信息集中于局部区域，则大幅影响RS分析的准确度。

卡方分析

一般地，当计算p值> 0.95时，可以认为嵌入了秘密信息。但当嵌入率小的情景下，不能保持较好的性能。根据实验得出的经验性结论，当嵌入率只有70%时，卡方分析的效果已经出现较大波动。此外，0.95这一阈值有时候并不是保证最低错误率的设定。因此需要具体情况具体分析，得出经验性的结论。