



南开大学
Nankai University

南 开 大 学

网 络 空 间 安 全 学 院

信息隐藏技术实验六

LSB 隐藏法实验

穆禹宸 2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：李朝晖

2023 年 4 月 4 日

目录

一、 实验要求	1
(一) 实验目的	1
(二) 实验环境	1
(三) 实验要求	1
二、 实验原理	1
(一) LSB 方法及特点	1
(二) LSB 算法相关函数	2
三、 实验步骤	2
(一) 将二值图像嵌入到位图中	2
1. 实验代码	2
2. 实验结果	5
(二) 将学号 (一个整数) 嵌入到位图中	7
1. 实验代码	7
2. 实验结果	11
四、 实验心得体会	12

一、实验要求

(一) 实验目的

1. 实现将二值图像嵌入到位图中
2. 实现将学号 (一个整数) 嵌入到位图中

(二) 实验环境

所需实验环境

主要有如下内容

- (1) 运行系统: Windows11
- (2) 实验工具: Matlab2022a
- (3) 数据: **PNG** 格式图像

(三) 实验要求

1. 在 Matlab 之中完成
2. 编写实验代码和报告, 并给出截图
3. QQ 群提交作业

二、实验原理

(一) LSB 方法及特点

LSB(LeastSignificantBit) 方法是用秘密信息 (比特) 替换掉最低有效位的数据。具体来说, LSB 算法是一种隐写术, 用于将秘密信息嵌入到数字图像的最低有效位中, 以实现隐蔽传输。该算法的基本思想是将秘密信息的二进制码嵌入到数字图像的最低有效位中, 因为最低有效位的变化对图像的视觉效果影响最小, 所以不容易被察觉。需要注意的是, LSB 算法虽然隐蔽性较好, 但也容易被攻击者检测到。因此, 在实际应用中, 需要结合其他隐写术和加密算法, 以提高信息的安全性。

优点: 简单, 容易实现, 容量大;

缺点: 安全性不高, 不能抵抗叠加噪声, 有损压缩等破坏。

提高安全性

主要有如下措施:

- (1) 对秘密信息进行加密后再隐藏
- (2) 多次重复嵌入
- (3) 引入纠错编码技术。先进行纠错编码, 再进行隐藏。

(二) LSB 算法相关函数

相关函数

主要有如下函数：

- (1) 获取图像 x 的行数和列数
- (2) 嵌入/提取图像 bit 位的值

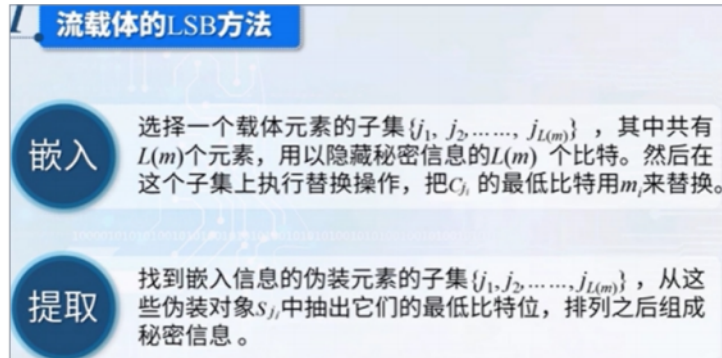


图 1: 原理

三、 实验步骤

总述：水印图像是二值图像，载体图像是与水印图像大小相同的 256 级灰度图像，进行 LSB 水印图像的嵌入和提取读取。

总述 LSB 步骤

主要有如下五个步骤：

- (1) 将秘密信息转换为二进制码。
- (2) 将数字图像的像素点转换为二进制码。
- (3) 将秘密信息的二进制码依次嵌入到数字图像像素点的最低有效位中。
- (4) 将修改后的像素点重新组合成数字图像。
- (5) 接收方通过提取数字图像的最低有效位，即可获取秘密信息的二进制码，再将其转换为原始信息。

(一) 将二值图像嵌入到位图中

1. 实验代码

步骤一

读取灰度图像和二值图像。这里选择 Lena。

首先，使用 `imread` 函数读取两张数字图像，分别为载体图像和水印图像。其中，载体图像为“Lena.bmp”，水印图像为“lion.bmp”。

接着，使用 `imshow` 函数分别显示载体图像和水印图像。其中，`[]` 表示将图像的像素值映射到 `[0,1]` 之间，以便于显示。

然后，调用 `Hide` 函数将水印图像嵌入到载体图像中，生成一个新的水印图像。`Hide` 函数是一个自定义函数，实现了基于 LSB 算法的图像水印嵌入。

最后，调用 `Extract` 函数从新生成的水印图像中提取出原始的水印图像。`Extract` 函数也是一个自定义函数，实现了基于 LSB 算法的图像水印提取。

```
1 function ImageHiding()
2     x=imread("Lena.bmp"); % 载体图像
3     m=imread("lion.bmp"); % 水印图像
4     imshow(x,[])
5     imshow(m,[]);
6     WaterMarked=Hide(x,m);
7     watermark=Extract(WaterMarked);
8 end
```

步骤二

将隐藏图像嵌入到载体图像的最低位平面，实现信息的隐藏。

下面这个函数实现了基于 LSB 算法的图像水印嵌入过程。具体解释如下：

首先，获取载体图像的大小，即行数和列数，分别为 `Mc` 和 `Nc`。

然后，创建一个与载体图像大小相同的全零矩阵 `WaterMarked`，用于存储嵌入水印后的图像。

接着，使用两层 `for` 循环遍历载体图像的每个像素点，将水印图像的每个像素点的最低有效位嵌入到载体图像的对应像素点的最低有效位中。具体实现是使用 `bitset` 函数将载体图像的对应像素点的最低有效位替换为水印图像的对应像素点的最低有效位。

嵌入完成后，使用 `imwrite` 函数将嵌入水印后的图像保存为“lsb_watermarked.bmp”文件。

最后，使用 `imshow` 函数显示嵌入水印后的图像，并设置 `title` 为“WaterMarked Image”。

```
1 function WaterMarked = Hide(origin,watermark)
2     [Mc,Nc]=size(origin);
3     WaterMarked=uint8(zeros(size(origin)));
4
5     for i=1:Mc
6         for j=1:Nc
7             WaterMarked(i,j)=bitset(origin(i,j),1,watermark(i,j));
8         end
9     end
10
11     imwrite(WaterMarked,'lsb_watermarked.bmp','bmp');
12     figure;
13     imshow(WaterMarked,[]);
14     title("WaterMarked Image");
```

```
15 end
```

步骤三

提取伪装对象的最低位平面，恢复隐藏的图像

这个函数实现了基于 LSB 算法的图像水印提取过程。具体解释如下：

首先，获取嵌入水印后的图像的大小，即行数和列数，分别为 Mw 和 Nw。

然后，创建一个与嵌入水印后的图像大小相同的全零矩阵 WaterMark，用于存储提取出的水印图像。

接着，使用两层 for 循环遍历嵌入水印后的图像的每个像素点，提取出每个像素点的最低有效位，并将其存储到 WaterMark 矩阵的对应位置中。具体实现是使用 bitget 函数获取嵌入水印后的图像的对应像素点的最低有效位。

提取完成后，使用 imshow 函数显示提取出的水印图像，并设置 title 为"WaterMark"。

```
1 function WaterMark=Extract(WaterMarked)
2     [Mw,Nw]=size(WaterMarked);
3     WaterMark=uint8(zeros(size(WaterMarked)));
4
5     for i=1:Mw
6         for j=1:Nw
7             WaterMark(i,j)=bitget(WaterMarked(i,j),1);
8         end
9     end
10
11     figure;
12     imshow(WaterMark,[]);
13     title("WaterMark");
14 end
```

最后，完整代码如下：

```
1 function ImageHiding()
2     x=imread("Lena.bmp"); % 载体图像
3     m=imread("lion.bmp"); % 水印图像
4     imshow(x,[])
5     imshow(m,[]);
6     WaterMarked=Hide(x,m);
7     watermark=Extract(WaterMarked);
8 end
9
10 function WaterMarked = Hide(origin,watermark)
11     [Mc,Nc]=size(origin);
12     WaterMarked=uint8(zeros(size(origin)));
13
```

```
14     for i=1:Mc
15         for j=1:Nc
16             WaterMarked(i,j)=bitset(origin(i,j),1,watermark(i,j));
17         end
18     end
19
20     imwrite(WaterMarked,'lsb_watermarked.bmp','bmp');
21     figure;
22     imshow(WaterMarked,[]);
23     title("WaterMarked Image");
24 end
25
26 function WaterMark=Extract(WaterMarked)
27     [Mw,Nw]=size(WaterMarked);
28     WaterMark=uint8(zeros(size(WaterMarked)));
29
30     for i=1:Mw
31         for j=1:Nw
32             WaterMark(i,j)=bitget(WaterMarked(i,j),1);
33         end
34     end
35
36     figure;
37     imshow(WaterMark,[]);
38     title("WaterMark");
39 end
```

2. 实验结果

首先展示要隐藏的水印和载体图像。



图 2: 要隐藏的水印



图 3: 载体图像

然后展示恢复以后的水印图像:



图 4: 恢复后图像

可以看到实验取得圆满成功!

(二) 将学号 (一个整数) 嵌入到位图中

实验步骤

主要有如下六个步骤:

- (1) 将要嵌入的整数转换为二进制码。
- (2) 读取载体图片, 并将其转换为二进制码。
- (3) 将要嵌入的整数的二进制码依次嵌入到载体图片像素点的最低有效位中。具体实现是使用 `bitset` 函数将载体图片的对应像素点的最低有效位替换为要嵌入的整数的对应二进制位。
- (4) 将修改后的像素点重新组合成图片。
- (5) 嵌入完成后, 保存修改后的图片。
- (6) 接收方通过提取图片的最低有效位, 即可获取嵌入的整数的二进制码, 再将其转换为原始整数。

1. 实验代码

主函数

读取 Lena 灰度图像并调整大小, 然后调用函数操作。

先编写一个函数，实现了将一个整数通过 LSB 算法嵌入到数字图像中，并从嵌入后的图像中提取出原始整数的过程。

首先，使用 `imread` 函数读取载体图像，即“Lena.bmp”。

然后，将要嵌入的信息 `m` 设置为 2012026，即一个整数（我的学号）。

接着，使用 `imshow` 函数显示载体图像。其中，`[]` 表示将图像的像素值映射到 $[0,1]$ 之间，以便于显示。

然后，调用 `Hide` 函数将整数 `m` 嵌入到载体图像中，生成一个新的嵌入信息后的图像。`Hide` 函数是一个自定义函数，实现了将整数通过 LSB 算法嵌入到数字图像中的过程。

嵌入完成后，调用 `Extract` 函数从新生成的嵌入信息后的图像中提取出原始的整数。`Extract` 函数也是一个自定义函数，实现了从数字图像中提取出嵌入的整数的过程。

最后，将提取出的整数赋值给变量 `watermark`，并输出到命令窗口。

```
1 function IntHiding()
2     x=imread("Lena.bmp"); % 载体图像
3     m=2012026; % 要嵌入的信息
4     imshow(x, [])
5     WaterMarked=Hide(x,m);
6     watermark=Extract(WaterMarked)
7 end
```

嵌入函数

嵌入学号 2012026 后得到水印图像。

注意，由于学号只有七位，因此我们这里只将整数的二进制码的前 21 位嵌入到图像中已经足够了。

首先，获取载体图像的大小，即行数和列数，分别为 `Mc` 和 `Nc`。

然后，创建一个与载体图像大小相同的全零矩阵 `WaterMarked`，用于存储嵌入水印后的图像。

使用两层 `for` 循环遍历载体图像的每个像素点，如果当前像素点在第一行且列数小于等于 21，则将要嵌入的整数的对应二进制位嵌入到该像素点的最低有效位中。具体实现是使用 `bitset` 函数将载体图像的对应像素点的最低有效位替换为要嵌入的整数的对应二进制位。如果当前像素点不在第一行或列数大于 21，则直接将该像素点的值赋给 `WaterMarked` 矩阵的对应位置。

嵌入完成后，使用 `imwrite` 函数将嵌入水印后的图像保存为“lsb_int_watermarked.bmp”文件。

最后，使用 `imshow` 函数显示嵌入水印后的图像，并设置 `title` 为“WaterMarked Image”。

```
1 function WaterMarked = Hide(origin,watermark)
2     [Mc,Nc]=size(origin);
3     WaterMarked=uint8(zeros(size(origin)));
4
5     for i=1:Mc
6         for j=1:Nc
7             if i==1 && j<=21
8                 tem=bitset(watermark,j);
```

```

9         WaterMarked(i,j)=bitset(origin(i,j),1,tem);
10     else
11         WaterMarked(i,j)=origin(i,j);
12     end
13 end
14 end

15
16 imwrite(WaterMarked,'lsb_int_watermarked.bmp','bmp');
17 figure;
18 imshow(WaterMarked,[]);
19 title("WaterMarked Image");
20 end

```

提取函数

从水印图像中提取学号【注意学号的二进制位数】

这个函数实现了从一个数字图像中提取出嵌入的整数的过程，但与普通的 LSB 算法不同的是，这里只提取整数的二进制码的前 21 位。具体解释如下：

首先，将提取出的整数 WaterMark 初始化为 0。

使用 for 循环遍历嵌入水印后的图像的第一行的前 21 个像素点，即整数的二进制码的前 21 位。

对于每个像素点，使用 bitget 函数获取其最低有效位，并将其存储到 tem 变量中。

然后，使用 bitset 函数将 tem 存储到 WaterMark 变量的对应二进制位中。

循环结束后，WaterMark 变量中存储的就是从嵌入水印后的图像中提取出的整数的二进制码的前 21 位。

最后，将提取出的整数 WaterMark 输出到命令窗口。

```

1 function WaterMark=Extract(WaterMarked)
2     WaterMark=0;
3     for j=1:21
4         tem=bitget(WaterMarked(1,j),1);
5         WaterMark=bitset(WaterMark,j,tem);
6     end
7 end

```

完整代码如下所示：

```

1 function IntHiding()
2     x=imread("Lena.bmp"); % 载体图像
3     m=2012026; % 要嵌入的信息
4     imshow(x,[])
5     WaterMarked=Hide(x,m);
6     watermark=Extract(WaterMarked)
7 end

```

```
8
9 function WaterMarked = Hide(origin,watermark)
10     [Mc,Nc]=size(origin);
11     WaterMarked=uint8(zeros(size(origin)));
12
13     for i=1:Mc
14         for j=1:Nc
15             if i==1 && j<=21
16                 tem=bitget(watermark,j);
17                 WaterMarked(i,j)=bitset(origin(i,j),1,tem);
18             else
19                 WaterMarked(i,j)=origin(i,j);
20             end
21         end
22     end
23
24     imwrite(WaterMarked,'lsb_int_watermarked.bmp','bmp');
25     figure;
26     imshow(WaterMarked,[]);
27     title("WaterMarked Image");
28 end
29
30 function WaterMark=Extract(WaterMarked)
31     WaterMark=0;
32     for j=1:21
33         tem=bitget(WaterMarked(1,j),1);
34         WaterMark=bitset(WaterMark,j,tem);
35     end
36 end
```

2. 实验结果

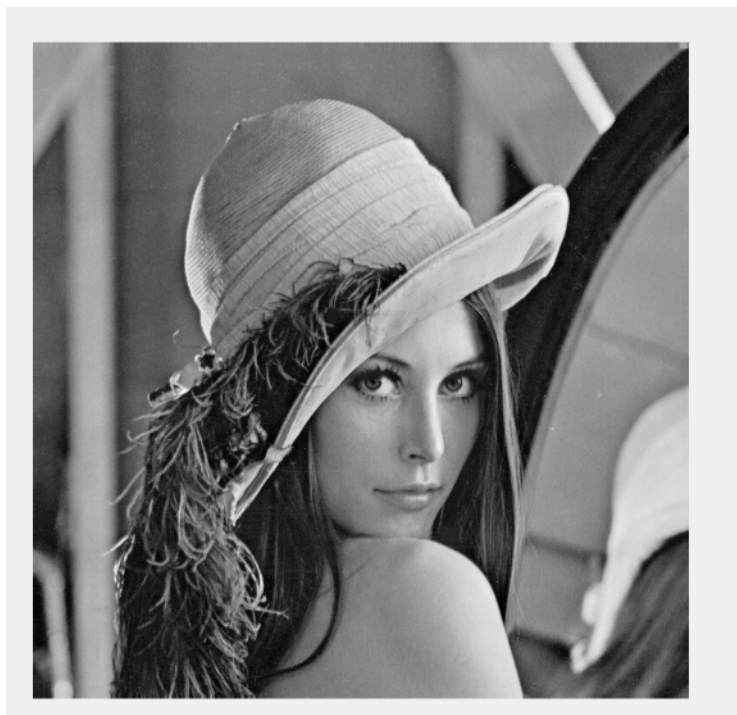


图 5: 载体图像

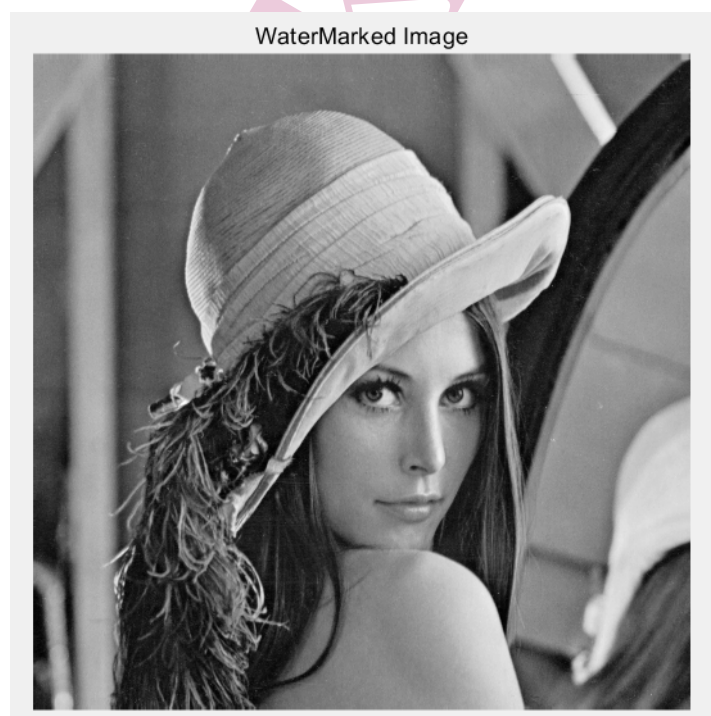
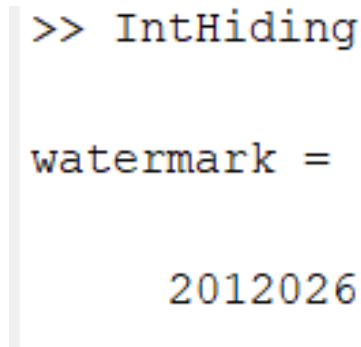


图 6: 隐藏后的图像



```
>> IntHiding  
  
watermark =  
  
2012026
```

图 7: 恢复出来的结果

这里我们可以看到我的学号已经恢复成功了!

四、 实验心得体会

在进行 LSB 实验的过程中,我深刻地认识到了数字图像隐写术的重要性和应用价值。LSB 算法是数字图像隐写术中最基础、最常用的一种方法,其原理简单、实现方便,可以用于数字图像的版权保护、身份认证等方面。在实验中,我通过编写 MATLAB 代码,实现了将一个整数通过 LSB 算法嵌入到数字图像中,并从嵌入后的图像中提取出原始整数的过程。同时,我也了解到了 LSB 算法的局限性,即隐蔽性较差,容易被攻击者检测到,因此需要结合其他隐写术和加密算法,以提高信息的安全性。

在实验中,我还学习了 MATLAB 的基本语法和图像处理函数的使用方法。通过编写代码,我深入理解了数字图像的存储方式和像素点的组成,以及如何对图像进行读取、显示、修改和保存等操作。同时,我也学会了如何使用 MATLAB 的位运算函数,如 `bitget` 和 `bitset`,来实现 LSB 算法的嵌入和提取过程。这些知识和技能对于我今后从事数字图像处理和隐写术方面的研究和应用都具有重要的意义。

此外,通过实验,我还深刻认识到了数字图像隐写术的应用价值。数字图像隐写术可以用于数字图像的版权保护、身份认证、信息隐藏等方面。例如,在数字图像的版权保护方面,可以将版权信息嵌入到数字图像中,以防止他人盗用或篡改;在身份认证方面,可以将身份信息嵌入到数字图像中,以便于身份的验证和识别;在信息隐藏方面,可以将秘密信息嵌入到数字图像中,以实现信息的隐蔽传输。这些应用都具有重要的现实意义和社会价值。

总的来说,通过 LSB 实验,我不仅学习了数字图像隐写术的基本原理和实现方法,还深刻认识到了数字图像隐写术的应用价值和局限性。这些知识和体会对于我今后从事数字图像处理和隐写术方面的研究和应用都具有重要的意义。

本次实验我受益匪浅,对信息隐藏技术掌握得更加深刻了,而且本次实验非常直观,我学到了很多!