



南開大學  
Nankai University

网络空间安全学院  
信息隐藏技术实验报告

二值图像隐藏法

姓名：孙露

学号：2112060

专业：信息安全

2024 年 4 月 30 日

# 目录

<b>1 实验要求</b>	<b>2</b>
<b>2 实验原理</b>	<b>2</b>
2.1 二值图像中的信息隐藏	2
2.2 Zhao—Koch 方案	2
2.2.1 嵌入	2
2.2.2 识别无效块	3
2.2.3 提取	3
2.2.4 简易算法	3
2.2.5 简易算法的嵌入过程 (1)	4
2.2.6 简易算法的嵌入过程 (2)	4
2.2.7 简易算法的提取过程	5
2.2.8 简易算法的核心代码	5
2.2.9 运行结果	6
2.3 利用游程编码的方法	6
<b>3 实验步骤</b>	<b>6</b>
3.1 嵌入过程	6
3.1.1 CalculateBlack	7
3.1.2 嵌入 encode	7
3.1.3 Matlab 中的 bitget 函数	11
3.1.4 Matlab 中的 bitset 函数	11
3.2 提取过程	12
<b>4 实验结果展示</b>	<b>13</b>
4.1 嵌入 encode	13
4.2 提取过程	14
<b>5 实验心得体会</b>	<b>14</b>

## 1 实验要求

实验 8：二值图像隐藏法

内容：

- 1、隐藏：利用二值图像隐藏法，实现将秘密信息（可以是图像、文字等信息）嵌入到位图中；
- 2、提取：将秘密信息提取出来。

## 2 实验原理

### 2.1 二值图像中的信息隐藏

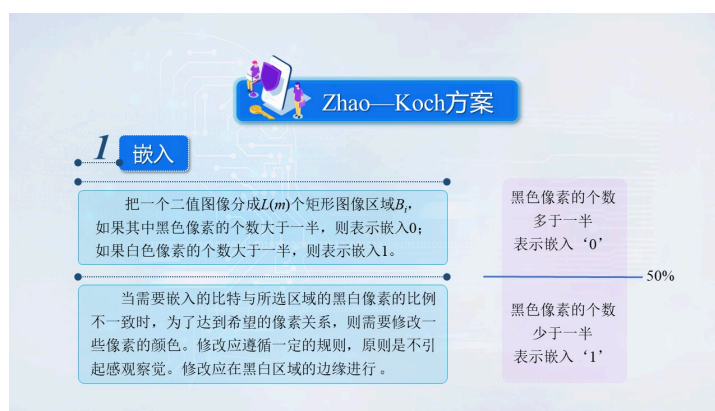
二值图像是由黑白两种颜色的像素组成的图像，通常利用图像区域中黑色像素个数相对于区域中全部像素个数的百分比来对秘密信息进行编码。

### 2.2 Zhao—Koch 方案

#### 2.2.1 嵌入

把一个二值图像分成  $L(m)$  个矩形图像区域  $B_i$ ，如果其中黑色像素的个数大于一半，表示嵌入 0；如果白色像素的个数大于一半，表示嵌入 1。

当需要嵌入的比特与所选区域的黑白像素的比例不一致时，为了达到希望的像素关系，需要修改一些像素的颜色。修改应遵循一定的规则，原则是不引起感官察觉。修改应在黑白区域的边缘进行。



选择图像块时，应考虑一定的冗余度，确定有效区域

- 确定两个阈值  $R_1 > 50\%$  和  $R_0 < 50\%$ ，以及一个健壮性参数  $\lambda$
- 隐藏 0 时，该块的黑色像素的个数应该属于  $[R_1, R_1 + \lambda]$
- 隐藏 1 时，该块的黑色像素的个数应该属于  $[R_0 - \lambda]$

**2 需注意的细节**

选择图像块时，应考虑有一定的冗余度，确定有效区域：

- 确定两个阈值  $R_1 > 50\%$  和  $R_0 < 50\%$ ，以及一个健壮性参数  $\lambda$ 。
- 隐藏0时，该块的黑色像素的个数应属于  $[R_1, R_1 + \lambda]$ ；
- 隐藏1时，该块的黑色像素的个数应属于  $[R_0 - \lambda, R_0]$ 。

### 2.2.2 识别无效块

如果为了适应所嵌入的比特，目标块必须修改太多的像素，就把该块设为无效。

方法：将无效块中的像素进行少量修改，使得其中黑色像素的百分比大于  $R_1 + 3\lambda$ ，或者小于  $R_0 - 3\lambda$ 。

### 2.2.3 提取

判断每一个图像块黑色像素的百分比，如果大于  $R_1 + 3\lambda$ ，或者小于  $R_0 - 3\lambda$ ，则跳过这样的无效块。

如果在  $[R_1, R_1 + \lambda]$  或者  $[R_0 - \lambda, R_0]$  的范围内，则正确提取出秘密信息 0 或 1。

### 2.2.4 简易算法

将原图划分为  $1 \times 4$  的矩形像素块，每个区域有四个连续的像素点，这些像素点的取值情况可以分为 5 类，全白，1 个黑像素点，2 个黑像素点，3 个黑像素点和全黑。

我们要隐藏的信息为文本文档中的字符串，不过需要注意的是：

- 嵌入的信息长度不能过大，不能超过图片大小所能复旦的度量。
- 为了简化过程，可以规定接受者已知秘密信息的长度。

**5 简易算法**

将原图划分为  $1 \times 4$  的矩形像素块，每个区域有四个连续的像素点。这些像素点的取值情况可以分为 5 类：全白，1 个黑像素点，2 个黑像素点，3 个黑像素点和全黑。

黑像素个数	0	1	2	3	4
像素分布	全白	1黑3白	两黑两白	3黑1白	全黑
含义	无效块	隐藏“1”	不能出现	隐藏“0”	无效块

我们要隐藏的信息为文本文档中的字符串，需要注意的是：

- 嵌入的信息长度不能过大，不能超过图片大小所能负担的度量；
- 为了简化过程，可以规定接收者已知秘密信息的长度。

### 2.2.5 简易算法的嵌入过程 (1)

遍历原图中的每个  $1 \times 4$  矩形区域 (1) 如果我们要嵌入的信息为 0, 则需要将当前区域的黑像素点数量调整到 3 个。

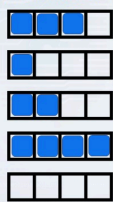
- 如果我们当前区域黑像素点数量正好为 3 个, 则不需要修改
- 如果当前区域黑像素点数量为 1 或 2 或 4 个, 则需要进行修改, 使其黑像素点数量变为 3 个, 同时需要注意的是, 对原有黑像素直接利用, 位置不做修改, 为的是嵌入秘密信息的过程中, 对图片的修改尽量少。
- 如果原区域全白, 则舍弃这一块不做修改, 否则变化可能会被直观视觉所感受到。

**6 简易算法的嵌入过程(1)**

**嵌入过程** 遍历原图中的每个  $1 \times 4$  矩形区域。

(1) 如果我们要嵌入的信息为 0, 则需要将当前区域的黑像素点数量调整到 3 个。

- 如果当前区域黑像素点数量正好为 3 个, 则不需要修改;
- 如果当前区域黑像素点数量为 1 或 2 或 4 个, 则需进行修改, 使其黑色像素点数量变为 3 个, 同时要注意的是, 对原有黑色像素直接利用, 位置不做修改, 为的是嵌入秘密信息的过程中, 对图片的修改尽量少。
- 如果原区域全白, 则舍弃这一块不做修改, 否则变化可能会被直观视觉所感受到。



### 2.2.6 简易算法的嵌入过程 (2)

遍历原图中的每个  $1 \times 4$  矩形区域 (1) 如果我们要嵌入的信息为 2, 则需要将当前区域的黑像素点数量调整到 2 个。

- 如果我们当前区域黑像素点数量正好为 2 个, 则不需要修改
- 如果当前区域黑像素点数量为 0 或 2 或 3 个, 则需要进行修改, 使其黑像素点数量变为 1 个, 同时需要注意的是, 对原有黑像素直接利用, 多余的翻转为白像素, 为的是嵌入秘密信息的过程中, 对图片的修改尽量少。
- 如果原区域全黑, 则舍弃这一块不做修改, 否则变化可能会被直观视觉所感受到。

### 7 简易算法的嵌入过程(2)

(2) 如果我们要嵌入的信息为1，则需要将当前区域的黑像素点数量调整到1个。

- 如果当前区域黑像素点数量正好为1个，则不需要修改；
- 如果当前区域黑像素点数量为0或2或3个，则需进行修改，使其黑色像素点数量变为1个，同时要注意的是，对原有黑色像素直接利用，多余的翻转为白像素，为的是嵌入秘密信息的过程中，对图片的修改尽量少。
- 如果原区域全黑，则舍弃这一块不做修改，否则变化可能会被直观视觉所感受到。



### 2.2.7 简易算法的提取过程

嵌入过信息的图像中每个区域的黑色像素点个数只有 4 个取值：0、1、3、4

遍历携带秘密信息图像的每个  $1 \times 4$  区域，如果黑色像素点个数为 1 或 3 则提取信息，1 个黑色像素点对应 '1'，3 个黑色像素点对应 '0'，黑色像素为 0 或 4 为未嵌入信息的区域。

### 8 简易算法的提取过程

嵌入过信息的图像中每个区域的黑色像素点个数只有4个取值：



提取过程

遍历携带图像的每个  $1 \times 4$  区域，如果黑色像素点个数为1或3则提取信息，1个黑色像素点对应 '1'，3个黑色像素点对应 '0'，黑色像素为0或4为未嵌入信息的区域。

### 2.2.8 简易算法的核心代码

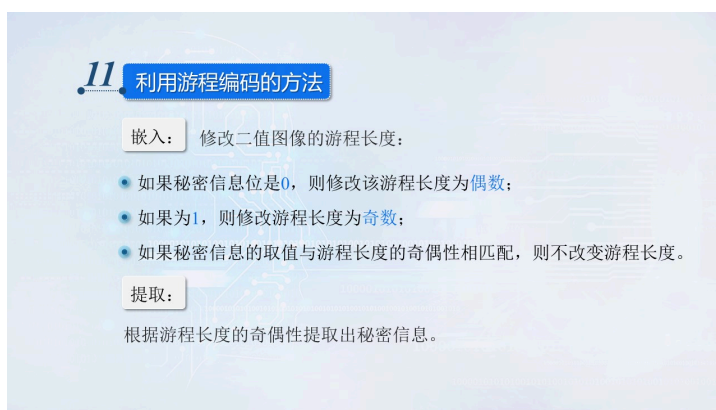
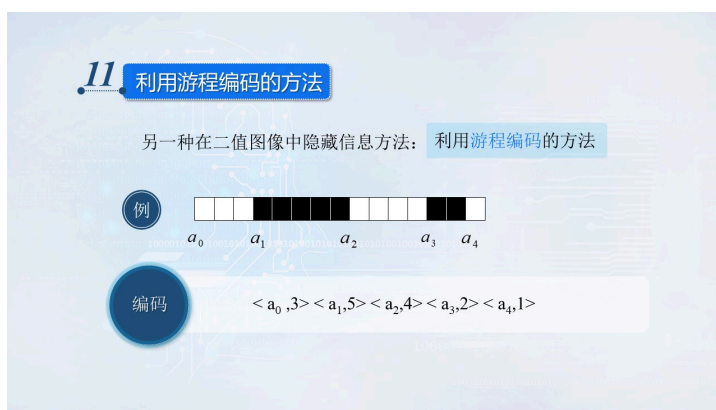
### 9 简易算法的核心代码

```
for j=c:cc %统计区域内黑块个数
    if bw(r,j)==0
        blk=blk+1;
    end
end
%要嵌入消息为0，修改使黑色变为3个
if bin_msg(ind)==0
    if blk==1 %1个黑像素的情况
        if bw(r,c)==0 %黑像素位于第1个位置
            bw(r,c+1)=0;
            bw(r,c+2)=0;
        elseif bw(r,c+1)==0 %黑像素位于第2个位置
            bw(r,c)=0;
            bw(r,c+2)=0;
        elseif %黑像素位于第3个位置
            %黑像素位于第4个位置
        end
    elseif blk==2 %2个黑像素的情况
    elseif blk==3 %3个黑像素的情况
    elseif blk==4 %4个黑像素的情况
        randnum=round(rand(1,1)*3);
        if randnum==0
            bw(r,c)=1;
        elseif randnum==1
            bw(r,c+1)=1;
        end
    elseif blk==0 %黑色为0个，舍弃这一块
        ind=ind-1;
    end
else %要嵌入的消息为1，需要修改使黑色为1个
    %
end
```

### 2.2.9 运行结果



## 2.3 利用游程编码的方法



## 3 实验步骤

### 3.1 嵌入过程

嵌入过程主要分为以下 3 个步骤:



- 给二值图像划定区域，一  $1 \times 4$  为一个区域进行位图的划分
- 统计区域内黑块的个数
- 循环嵌入秘密文本信息

### 3.1.1 CalculateBlack

CalculateBlack 用以统计黑色像素的数量。计算从  $x$  第  $i$  个元素开始的连续 4 个元素中，值为 0 的元素个数，以 4 格为单位遍历原始图像然后输出黑色像素的数量

Listing 1: CalculateBlack

```
1
2 function out = CalculateBlack(x, i)
3 out = 0;
4 for a = i : i + 3
5     if x(a) == 0
6         out = out + 1;
7     end
8 end
9 end
```

### 3.1.2 嵌入 encode

秘密信息为学号，encode 首先读取图片并转换二值图像，使用 bitget 函数对学号进行二进制提取。学号转换成二进制后，也就是需要隐藏的信息，可以发现至少需要 24 位，因此 for 循环执行 24 次。

2,112,060	
HEX	20 3A3C
DEC	2,112,060
OCT	10 035 074
BIN	0010 0000 0011 1010 0011 1100

然后需要对秘密进行嵌入。

首先判断嵌入的秘密信息是 0 还是 1，如果是 0，那么需要 3 个黑色像素点和 1 个白色像素点；如果是 1，那么需要 3 个白色像素点和 1 个黑色像素点，通过 CalculateBlack 进行获取。

最后保存原图和加了水印的图

Listing 2: encode

```
1
2 clc;
3 clear;
4 d = imread('./pic.bmp');
```



```
5 d = im2bw(d);
6 imwrite(d, 'black1.bmp', 'bmp')
7 subplot (1, 2, 1); imshow (d, []); title (' 原始图片 ');
8 secret = 2112060;
9
10 for t = 1:24
11     s(t) = bitget(secret, t);
12 end
13
14 num = 1;
15 t = 1;
16
17 while t < 24
18
19     if s(t) == 0
20
21         switch (CalculateBlack(d, num))
22             case 0
23                 t = t - 1;
24                 num = num + 4;
25             case 1
26                 temp = 1;
27                 startnum = num;
28
29                 while temp < 3
30
31                     if d(startnum) == 1
32                         d(startnum) = 0;
33                         temp = temp + 1;
34                         startnum = startnum + 1;
35                     end
36
37                 end
38
39                 num = num + 4;
40             case 2
41                 temp = 2;
42                 startnum = num;
43
44                 while temp < 3
45
46                     if d(startnum) == 1
```

```
47         d(startnum) = 0;
48         temp = temp + 1;
49         startnum = startnum + 1;
50     end
51
52 end
53
54     num = num + 4;
55 case 3
56     num = num + 4;
57 case 4
58     temp = 4;
59     startnum = num;
60
61     while temp > 3
62
63         if d(startnum) == 0
64             d(startnum) = 1;
65             temp = temp - 1;
66             startnum = startnum + 1;
67         end
68
69     end
70
71     num = num + 4;
72 end
73
74 else
75     a = CalculateBlack(d, num)
76
77     switch a
78     case 0
79         temp = 4;
80         startnum = num;
81
82         while temp > 3
83
84             if d(startnum) == 1
85                 d(startnum) = 0;
86                 temp = temp - 1;
87                 startnum = startnum + 1;
88             end
```

```
89
90         end
91
92         num = num + 4;
93
94     case 1
95         num = num + 4;
96     case 2
97         temp = 2;
98         startnum = num;
99
100         while temp < 3
101
102             if d(startnum) == 0
103                 d(startnum) = 1;
104                 temp = temp + 1;
105                 startnum = startnum + 1;
106             end
107
108         end
109
110         num = num + 4;
111     case 3
112         temp = 1;
113         startnum = num;
114
115         while temp < 3
116
117             if d(startnum) == 0
118                 d(startnum) = 1;
119                 temp = temp + 1;
120                 startnum = startnum + 1;
121             end
122
123         end
124
125         num = num + 4;
126     case 4
127         t = t - 1;
128         num = num + 4;
129     end
130
```

```
131     end
132
133     t = t + 1;
134 end
135
136 imwrite(d, 'black2.bmp', 'bmp')
137
138 subplot (1, 2, 2); imshow (d, []); title (' 水印图片 ');
```

### 3.1.3 Matlab 中的 bitget 函数

- 语法

b = bitget(A,bit)

b = bitget(A,bit,assumedtype)

- 说明

b = bitget(A,bit) 返回整数数组 A 中位于位置 bit 的位值。

b = bitget(A,bit,assumedtype) 假定 A 属于 assumedtype。

tmp(i,j) = bitget(X(i,j),k);

bitget 函数首先将 X(i,j) 处灰度值分解为二进制串，然后取第 k 位

### 3.1.4 Matlab 中的 bitset 函数

bitset 函数的功能是设置指定位置的位。

- 语法

intout = bitset(A,bit)

intout = bitset(A,bit,assumedtype)

intout = bitset(A,bit,V)

intout = bitset(A,bit,V,assumedtype)

- 说明

intout = bitset(A,bit) 返回 A 的值，并将位置 bit 设置为 1（打开）。

intout = bitset(A,bit, assumedtype) 假定 A 的类型为 assumedtype。

intout = bitset(A,bit,V) 返回 A，并将位置 bit 设置为 V 的值。

a) 如果 V 为零，则位的位置 bit 将设置为 0（关闭）。

b) 如果 V 为非零，则位的位置 bit 将设置为 1（打开）。

intout = bitset(A,bit,V, assumedtype) 假定 A 的类型为 assumedtype。

### 3.2 提取过程

读取载入信息后的伪装图像矩阵，使用 CalculateBlack 遍历获取区域内黑色像素的个数，统计区域内黑像素个数是 1、3 还是 0、4，从而恢复出秘密信息 0 或 1。

Listing 3: decode

```
1
2 %% decode
3 clc;
4 clear;
5 d = imread('black2.bmp');
6
7 for t = 1:24
8     s(t) = bitget(0, t);
9 end
10
11 t = 1;
12 num = 1;
13
14 while t < 24
15     a = CalculateBlack(d, num);
16     switch a
17         case 0
18             num = num + 4;
19         case 1
20             s(t) = 1;
21             t = t + 1;
22             num = num + 4;
23         case 3
24             s(t) = 0;
25             t = t + 1;
26             num = num + 4;
27         case 4
28             num = num + 4;
29     end
30 end
31
32 sum = 0;
33
34 for t = 1:24
35     sum = sum + s(t) * 2^(t - 1);
36 end
37
```

```
38 fprintf("秘密信息是:  %d\n", sum);
```

## 4 实验结果展示

### 4.1 嵌入 encode

原始图像:



嵌入秘密信息的图像:



## 4.2 提取过程

```
秘密信息是: 2112060  
fx >>
```

## 5 实验心得体会

实验过程中,我们首先需要准备一张载体图像和一段秘密信息。载体图像应该是一张二值图像,可以使用 MATLAB 中的 `im2bw` 函数将彩色图像转化为二值图像。秘密信息可以是一段文本或一张二值图像,需要将其转化为二进制码流。在转化过程中,需要注意数据存储的方式,如大端小端方式等,以确保发送者和接收者使用相同的方式进行数据存储和解析。接下来,我们需要将秘密信息嵌入到载体图像的像素中。具体方法是将秘密信息的二进制码流按照一定的规则嵌入到载体图像的像素中。常见的嵌入规则包括 LSB (最低有效位) 嵌入法、PVD (像素值差分) 嵌入法等。LSB 嵌入法是将秘密信息的二进制码流按照顺序依次嵌入到载体图像的像素的最低有效位中,从而实现信息的隐蔽传输。PVD 嵌入法是将秘密信息的二进制流按照顺序依次嵌入到载体图像的像素的像素值差分中,从而实现信息的隐蔽传输。在嵌入过程中,需要注意控制隐藏对象的大小和容量与秘密信息的匹配,以及图像行列的动态变化,避免信息的丢失和损坏。最后,我们需要将隐藏的秘密信息从载体图像中提取出来。具体方法是按照嵌入规则逆向操作,将载体图像中隐藏的秘密信息提取出来,并将其转化为原始的二进制码流,再根据数据存储的方式进行解析,得到原始的秘密信息。在提取过程中,需要注意发送者和接收者提前进行提取和隐藏方法的沟通,以确保信息的正确提取和解析。总的来说,二值图像隐藏法是一种简单有效的信息隐藏技术,可以用于保护敏感信息的隐私和安全。在实践中,需要注意控制隐藏对象的大小和容量与秘密信息的匹配,以及图像行列的动态变化,避免信息的丢失和损坏。

本次实验不仅帮助我加深了对数字水印嵌入和提取的理解,还提高了我在 MATLAB 编程方面的实践能力。通过实验,我深刻体会到了理论知识与实践操作的结合对于知识的巩固和应用的重要性,也更加深刻地认识到了信息安全在数字化时代的重要性。