目 录

JPEG 图像变换域信息隐藏实现

1 问题描述

1） 采用 JPEG 格式图像作为载体图像，用 MATLAB 实现 JPEG 图片变换域信息隐藏的嵌入与提取算法：JSTEG, F4, F5。

2） 比较不同变换域信息隐藏算法嵌入前后的视觉效果与 DCT 系数直方图。

2 系统实现

2.1 JSTEG 算法

2.1.1 系统设计

A. 嵌入算法

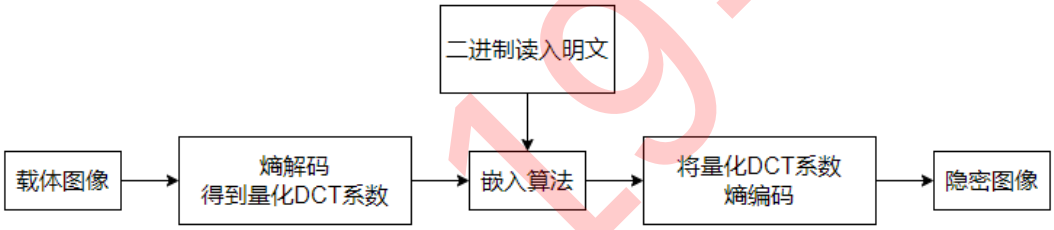

图 2.1-1 JSTEG 嵌入流程图

嵌入位置：原始值为-1、0、1 以外的量化 DCT 系数的 AC 系数。

嵌入方法：按顺序将一个二进制位的隐秘信息嵌入到系数的 LSB 上。在 JPEG 格式中，量化 DCT 系数采用哈夫曼编码。因此正偶数的 LSB 为 0，而负偶数的 LSB 为 1.

图 2.1-2 JSTEG 如何修改系数

B. 提取算法

提取流程为嵌入流程的逆过程，故不在此赘述流程图。

提取位置：值为-1、0、1 以外的量化 DCT 系数。

提取方法：按顺序读取量化 DCT 系数中 AC 系数的 LSB。

2.1.2 系统实现

A. 嵌入算法

将 JSTEG 嵌入算法封装为一个函数。

表 2.1-1 JSTEG 嵌入函数关键变量

	名称	意义
输入参数	COVER	载体图像文件名
	STEGO	隐秘图像文件名
	MSG	秘密信息文件名
函数变量	msg	秘密信息比特流
	DCT	载体图像量化 DCT 系数
	DCT2	隐密图像量化 DCT 系数
	pos	AC 系数在 DCT 中的位置
返回值		无

函数具体实现如下：

1) 将二进制秘密信息比特流存储到 msg 变量，其长度存储到 len 变量。使用 fopen 函数打开秘密信息文件，使用 fread 函数按照二进制方式读取文件内容。

2) 将载体图像的量化 DCT 系数存储到 DCT 变量中。使用 nsF5 算法提供的封装好的 jpeg_read 函数读取载体图像 COVER，函数返回值的成员 coef_arrays{1}即为载体图像的量化 DCT 系数。将 DCT 变量复制一份到 DCT2，用于存储嵌入信息后的量化 DCT 系数。

```
jobj=jpeg_read(COVER);
DCT=jobj.coef_arrays{1};
```

3) 使用 numel 函数计算量化 DCT 系数中的 AC 系数的个数，将其存储到变量 AC。若 AC 小于 len，则无法嵌入全部信息。

```
AC=numel(DCT)-numel(DCT(1:8:end,1:8:end));
```

4) 记录量化 DCT 系数中的非 0 值的位置，将其存储到变量 pos，我们将从这些位置选出可以嵌入信息的比特。首先，利用 size 和 true 函数建立一个和 DCT 变量一样大的全 1 矩阵 pos。由于 jpeg_read 函数将原图分为

8*8 的小块，然后转化为 YCbCr 模型，再对其进行 DCT 变换，因此每个 8*8 的小块的第 1 个系数是 DC 系数，剩余 63 个系数为 AC 系数，而 DC 系数是不能嵌入信息的，故将 pos 变量所有 8*8 小块的第 1 个位置置为 0。最后使用 find 函数，即可得到所有 AC 系数的位置。

```
pos=true(size(DCT));
pos(1:8:end,1:8:end)=false;
pos=find(pos);
```

5) 按顺序遍历量化 DCT 系数，直到所有信息都嵌入完毕。采用一个 i 从 1 变化到 len 的 for 循环，每执行一次 for 循环则嵌入 1 比特信息，则 msg(i) 表示第 i 比特明文信息。在 for 循环函数体中，先使用一个 while 循环来找到绝对值大于 1 的量化 DCT 系数，循环变量为 j，则 pos(j)表示第 j 个 DC 系数的位置，DCT(pos(j))表示第 j 个 DC 系数的值。

```
while(abs(DCT(pos(j)))<=1)
    j=j+1;
end
```

找到绝对值大于 1 的 DC 系数后，即可嵌入信息。根据图 2.1-2，我们将 2i~2i+1 的变换抽象成下面的语句。

```
if(DCT(pos(j))>1)
    DCT2(pos(j))=DCT2(pos(j))-mod(DCT2(pos(j)),2)+msg(i);
else
    DCT2(pos(j))=DCT2(pos(j))-mod(DCT2(pos(j))+1,2)+msg(i);
end
```

6) 使用 jpeg_write 函数对 DCT2 进行熵编码并写入隐密图像 STEGO。

```
jobj.coef_arrays{1}=DCT2;
jobj.optimize_coding=1;
jpeg_write(jobj,STEGO);
```

B. 提取算法

将 JSTEG 提取算法封装为一个函数，该函数与嵌入算法函数十分类似且更简单，我们同样使用 pos 来存储 DCT 系数中的 AC 系数，然后顺序扫描 AC 系

数，如果系数绝对值大于 1，则其 LSB 即为 1 比特秘密信息。

表 2.1-2 JSTEG 提取函数关键变量

	名称	意义
输入参数	STEGO	隐密图像文件名
	len	二进制隐密信息长度
	RES	存储提取出信息的文件名
函数变量	pos	AC 系数在 DCT 中的位置
返回值		无

首先我们使用一个 while 循环跳过绝对值不大于 1 的 AC 系数，对于剩余的 AC 系数，采用如下代码提取密文比特。

```
if(DCT(pos(j))>0)
    fwrite(fp,mod(DCT(pos(j)),2),'ubit1');
elseif(DCT(pos(j))<0)
    fwrite(fp,mod(DCT(pos(j))+1,2),'ubit1');
end
```

2.1.3 实验结果及性能分析

嵌入前后量化 DCT 系数直方图对比，可以看到 0、-1 和 1 的个数不变，其它系数的个数变化，这是因为我们在嵌入时会跳过 0、-1 和 1。

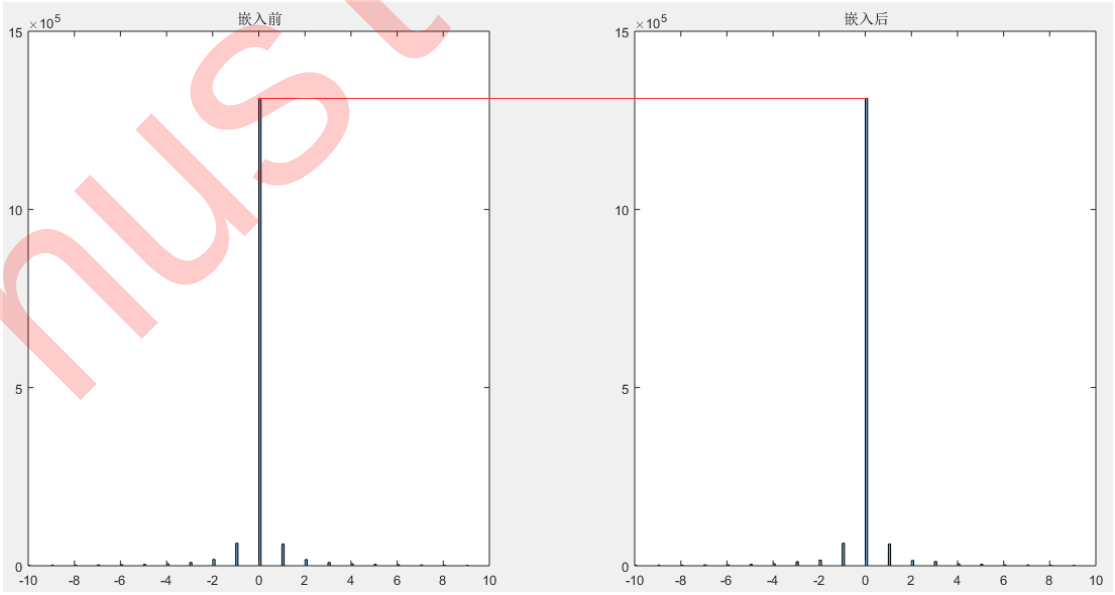

图 2.1-3 量化 DCT 系数直方图 1

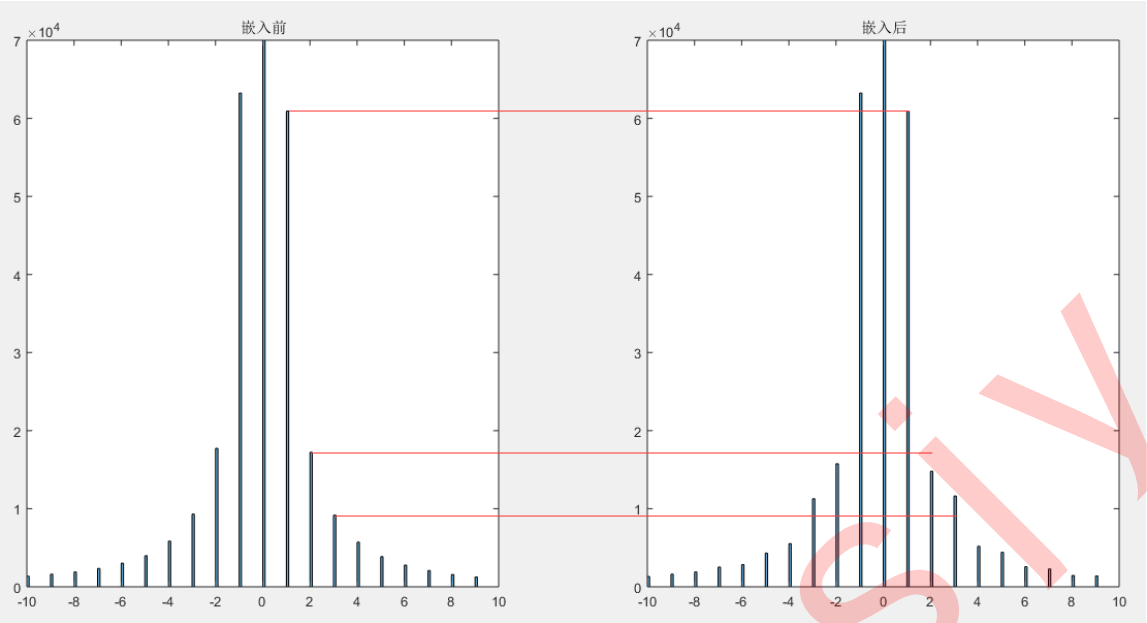

图 2.1-4 量化 DCT 系数直方图 2

观察量化 DCT 系数直方图的局部，发现原本 2i 和 2i+1 的频率有一定差距，而嵌入了大量信息后它们的频率变得很接近，这就是值对现象。

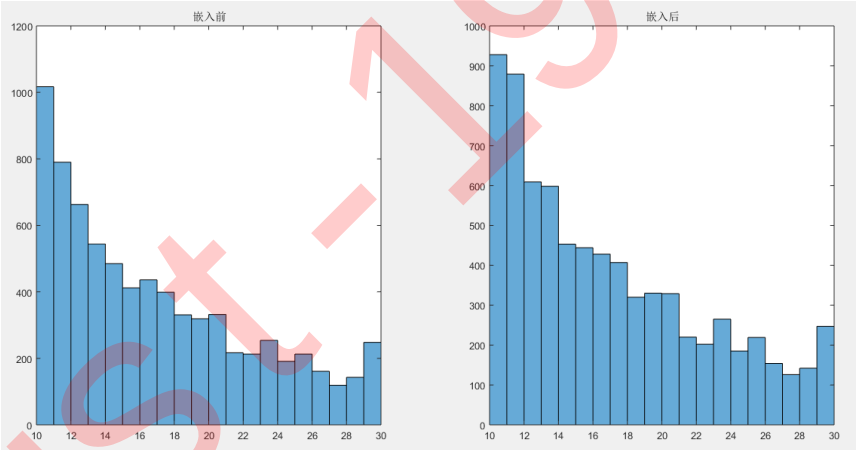

图 2.1-5 值对现象

嵌入前后，图片没有明显变化，不容易被看出嵌入了信息。

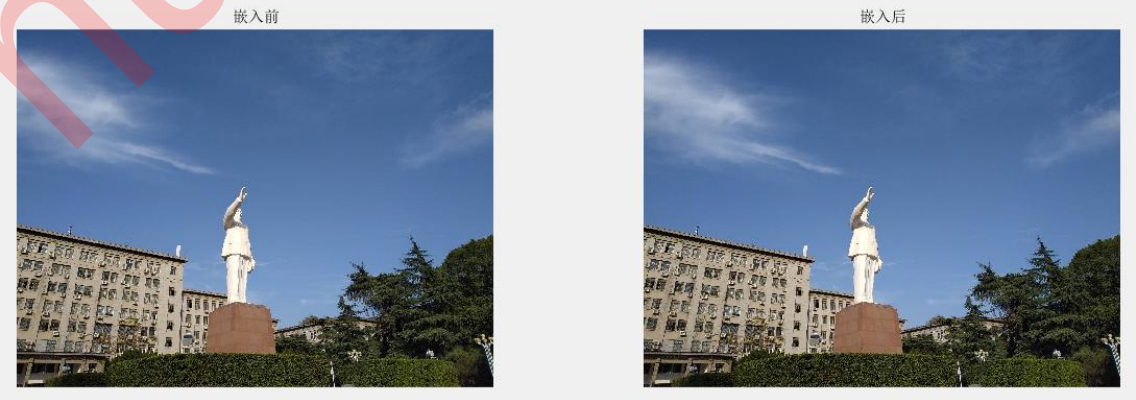

图 2.1-6 图片对比

嵌入和提取的信息一致。

图 2.1-7 嵌入和提取的信息对比

嵌入长度为 160016 的 01 比特流，嵌入时间约为 0.076 秒。

```
>> Jsteg_embed
嵌入时间: 0.076484秒
嵌入长度: 160016
```

图 2.1-8 JSTEG 性能分析

2.2 F4 算法

2.2.1 系统设计

A. 嵌入算法

F4 嵌入流程图与图 2.1-1 一致，只是嵌入算法不同。

嵌入位置：原始值为 0 以外的量化 DCT 系数的 AC 系数。

嵌入方法：如果当前位的 LSB 和隐秘信息比特不同，则将系数的绝对值减 1。同样地，正偶数的 LSB 为 0，而负偶数的 LSB 为 1。注意，如果某个系数嵌入后变为 0，则视作嵌入失败。

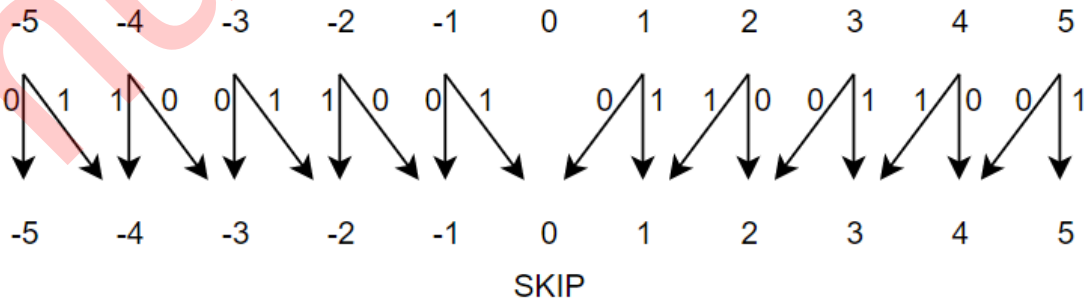

图 2.2-1 F4 如何修改系数

B. 提取算法

提取流程为嵌入流程的逆过程，故不在此赘述流程图。

提取位置：值为 0 的以外的量化 DCT 系数。

提取方法：按顺序读取量化 DCT 系数中 AC 系数的 LSB。

2.2.2 系统实现

A. 嵌入算法

将 F4 嵌入算法封装为一个函数。变量命名和大致流程均与 JSTEG 嵌入函数相同，关键的不同在于嵌入信息的 for 循环。

在这个 for 循环中，同样地，我们先利用一个 while 循环找到可以嵌入信息的 AC 系数，由于嵌入后系数变为 0 的嵌入是无效的，所以 while 循环继续的条件发生了改变。在 F4 中，如果 AC 系数为 1 且秘密比特为 0 或者 AC 系数为-1 且秘密比特为 1 或者 AC 系数为 0，则这次嵌入不会生效，不可以跳出 while 循环，要继续寻找下一个可嵌入的 AC 系数；而在 JSTEG 中只要 AC 系数绝对值大于 1 即可跳出 while 循环。

```
while(1)
    if((DCT(pos(j))==1 && msg(i)==0) || (DCT(pos(j))==-1 && msg(i)==1) ||
(DCT(pos(j))==0))
        DCT2(pos(j))=0;
        j=j+1;
    else
      break;
    end
end
```

利用 while 循环找到可嵌入信息的 AC 系数后，判断其 LSB 和秘密比特是否相同，若不相同则 AC 系数绝对值减 1。

```
if(DCT(pos(j))>0 && msg(i)~=mod(DCT2(pos(j)),2))
      DCT2(pos(j))=DCT2(pos(j))-1;
elseif(DCT(pos(j))<0 && msg(i)~=mod(DCT2(pos(j))+1,2))
     DCT2(pos(j))=DCT2(pos(j))+1;
end
```

B. 提取算法

将 F4 提取算法封装为一个函数，该函数与 JSTEG 提取函数十分类似。同样地，我们顺序扫描 AC 系数，对于不为 0 的 AC 系数，它们都携带着一比特密文。

首先我们使用一个 while 循环跳过为 0 的 AC 系数，对于剩余的 AC 系数，采用如下代码提取密文比特。

```
if(DCT(pos(j))>0)
    fwrite(fp,mod(DCT(pos(j)),2),'ubit1');
elseif(DCT(pos(j))<0)
    fwrite(fp,mod(DCT(pos(j))+1,2),'ubit1');
end
```

2.2.3 实验结果及性能分析

使用 F4 算法在与 JSTAG 算法相同的图片上嵌入相同的内容。

F4 算法嵌入前后量化 DCT 系数直方图对比，可以看到 0 的个数有所增加，见下图，这是因为有些 1 或者-1 嵌入失败变成了 0。而在 JSTEG 算法中，0、-1 和 1 的个数不会变化。

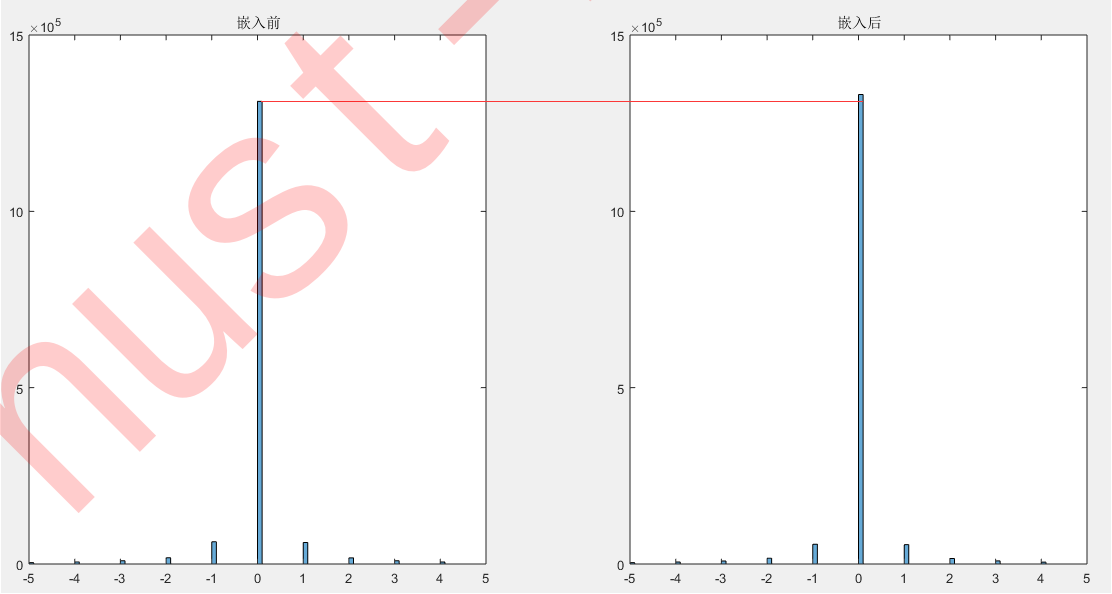

图 2.2-2 量化 DCT 系数直方图 1

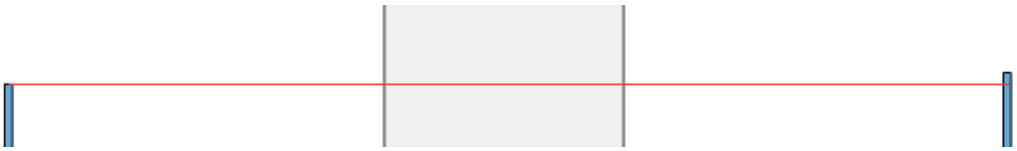

图 2.2-3 量化 DCT 系数直方图 1 放大

而-1 和 1 的个数都变少了，说明变成 0 的 1 比变成 1 的 2 更多，见下图。

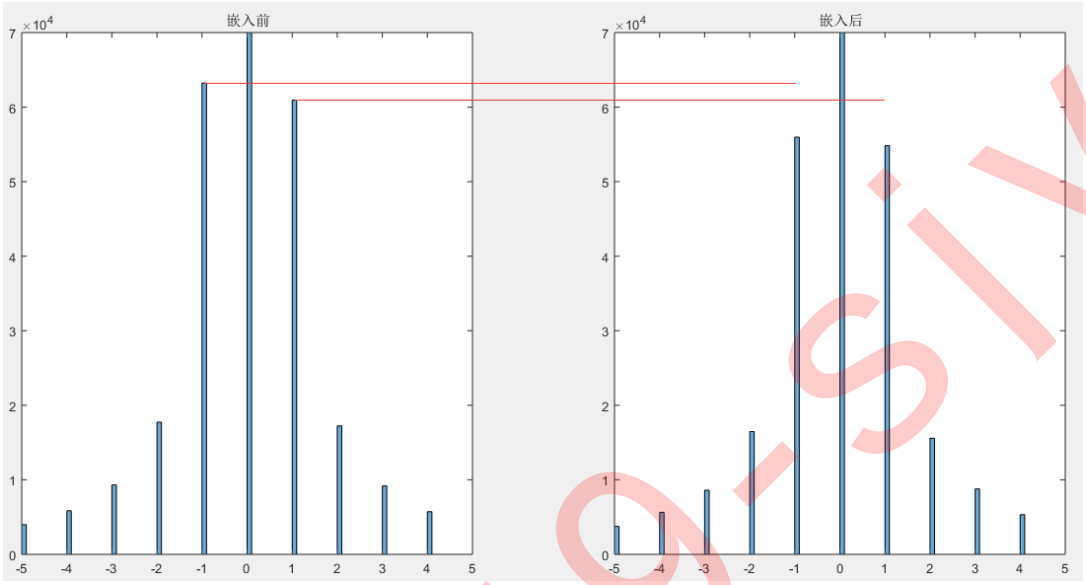

图 2.2-4 量化 DCT 系数直方图 2

观察量化 DCT 系数直方图的局部，发现嵌入前后 2i 和 2i+1 的频率差距几乎差不多，这是因为 F4 算法相当于做平移变换，因此没有值对现象。

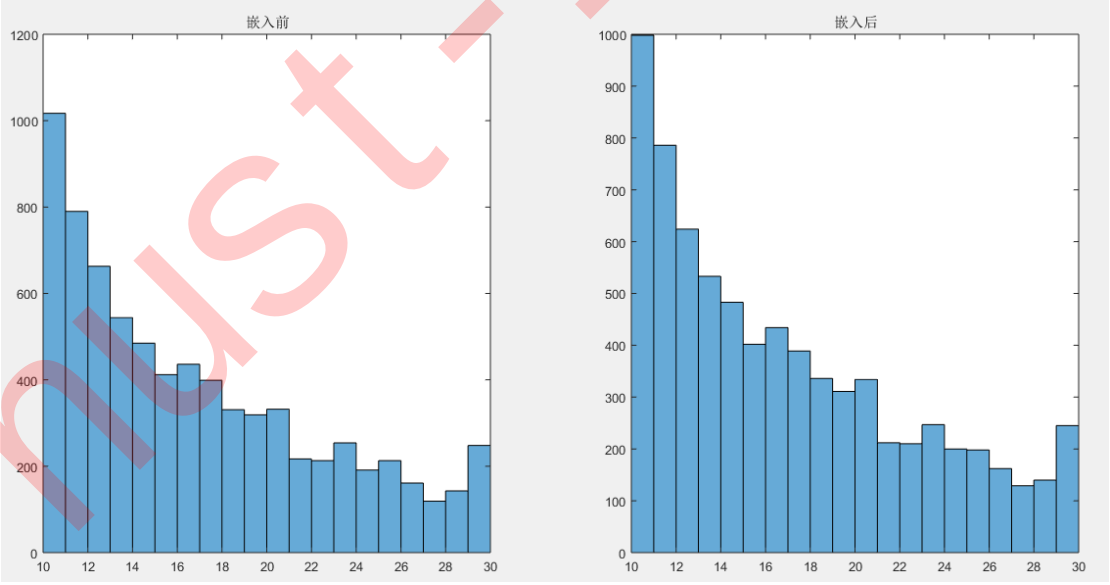

图 2.2-5 无值对现象

嵌入前后，图片没有明显变化，不容易被看出嵌入了信息。

图 2.2-6 图片对比

嵌入和提取的信息一致。

图 2.2-7 嵌入和提取的信息对比

2.3 F5 算法

2.3.1 系统设计

A. 嵌入算法

F5 嵌入流程图与图 2.1-1 一致，只是嵌入算法不同。相比 F4，F5 的两个改变是随机嵌入和矩阵编码，不过为了简化实验，我的 F5 采用的还是顺序嵌入的方法。而所谓矩阵编码，则能显著地减少修改的 AC 系数的个数。在本次实验中，我将 2 比特秘密信息作为一组，记为 b1 和 b2，用 3 个 AC 系数的 LSB 来表示它们，记为 a1, a2 和 a3。

嵌入位置：所有非 0 的 AC 系数都可能被修改，见下表。

表 2.3-1 F5 如何确定修改位置

情况		修改位
$b_1 = a_1 \oplus a_2$	$b_2 \neq a_2 \oplus a_3$	不修改
$b_1 \neq a_1 \oplus a_2$	$b_2 = a_2 \oplus a_3$	修改 a1
$b_1 = a_1 \oplus a_2$	$b_2 \neq a_2 \oplus a_3$	修改 a3
$b_1 \neq a_1 \oplus a_2$	$b_2 = a_2 \oplus a_3$	修改 a2

嵌入方法：与 F4 相同，注意，如果某个系数嵌入后变为 0，则视作嵌入失败。

B. 提取算法

提取位置：每 3 个非 0 的 AC 系数为一组。

提取方法：按顺序读取量化 DCT 系数中 AC 系数的 LSB，$b_1 = a_1 \oplus a_2$，$b_2 = a_2 \oplus a_3$。

2.3.2 系统实现

A. 嵌入算法

将 F5 嵌入算法封装为一个函数，其关键部分还是在于嵌入信息的循环，在 F5 算法中采用 while 循环，而不是前两种算法的 for 循环，这样更好处理嵌入失败的情况。

表 2.3-2 流程图变量说明

变量名	意义
i	当前嵌入到第几比特秘密信息
j	当前使用到第几个非 0 的 AC 系数
pos 矩阵	存储非 0 的 AC 系数的位置
k	这是一组（3 个）AC 系数的第几个
realpos(k)	本组第 k 个 AC 系数在 DCT 矩阵中的位置
a(k)	本组第 k 个 AC 系数的 LSB
thebit	一组（3 个）AC 系数的第几个需要被修改 0 表示没有系数需要被修改

此外，删除 a(thebit)表示将其它 2 个 AC 系数的 LSB 置于 a(1)和 a(2)，这样以后就可以寻找下一个非 0 的 AC 系数。删除 realpos(thebit)同理。

函数流程图如下图所示。

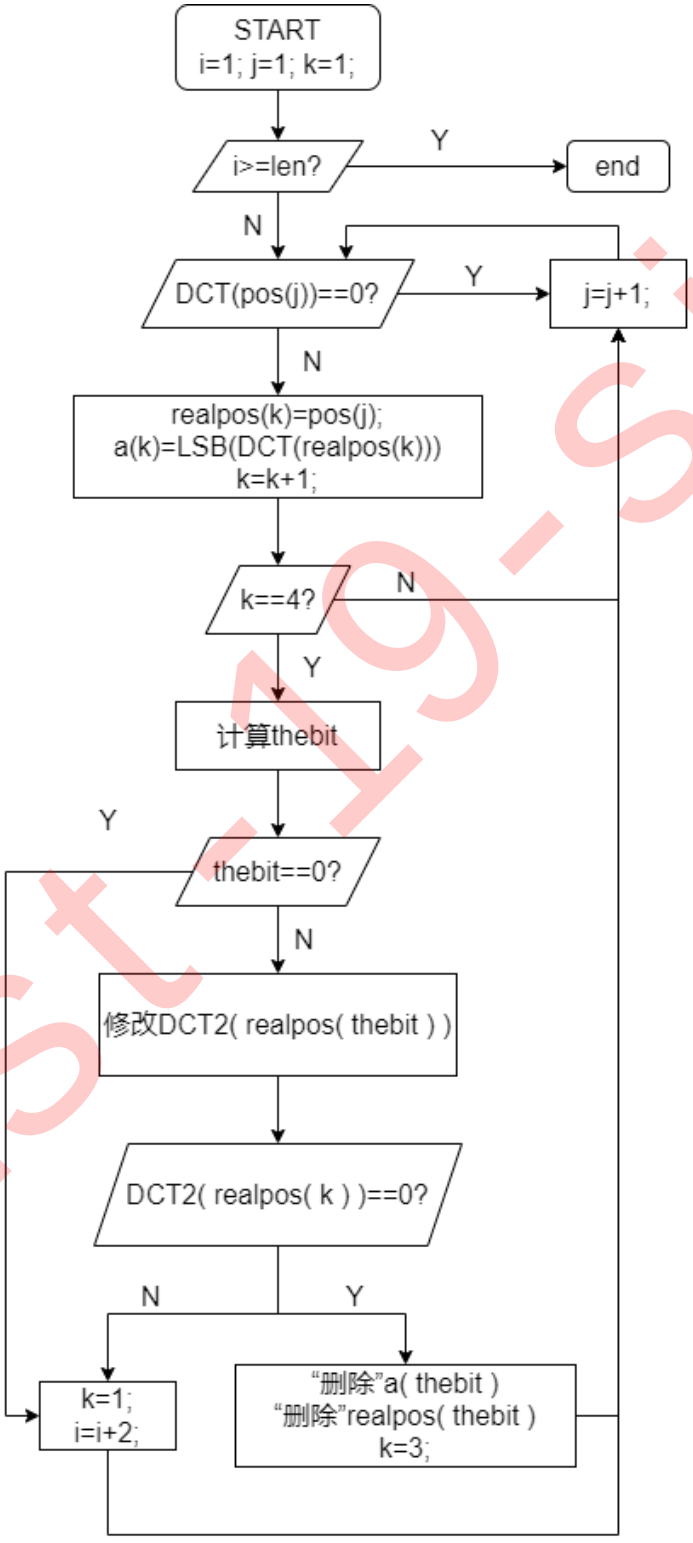

图 2.3-1 F5 嵌入函数流程图

B. 提取算法

将 F5 提取算法封装为一个函数，大体流程与嵌入算法相同，但更简单。我们顺序扫描 AC 系数，对于每 3 个非 0 的 AC 系数，它们携带着 2 比特密文。与嵌入算法相同，使用 a 来存储 3 个非 0 的 AC 系数的 LSB，每找到 3 个非 0 的 AC 系数，就向储存结果的文件写入 2 比特，即$b_1 = a_1 \oplus a_2$，$b_2 = a_2 \oplus a_3$。

2.3.3 实验结果及性能分析

使用 F5 算法在与 F4 算法相同的图片上嵌入相同的内容。

嵌入前后，图片没有明显变化，不容易被看出嵌入了信息。

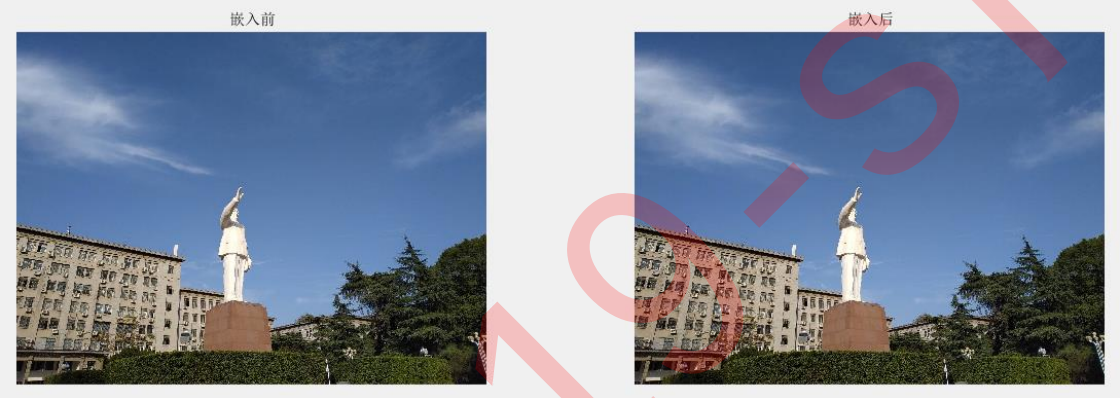

图 2.3-2 图片对比

F5 算法嵌入前后量化 DCT 系数直方图对比，可以看到 0 的个数有所增加，见下图，这是因为有些 1 或者-1 嵌入失败变成了 0，与 F4 类似。

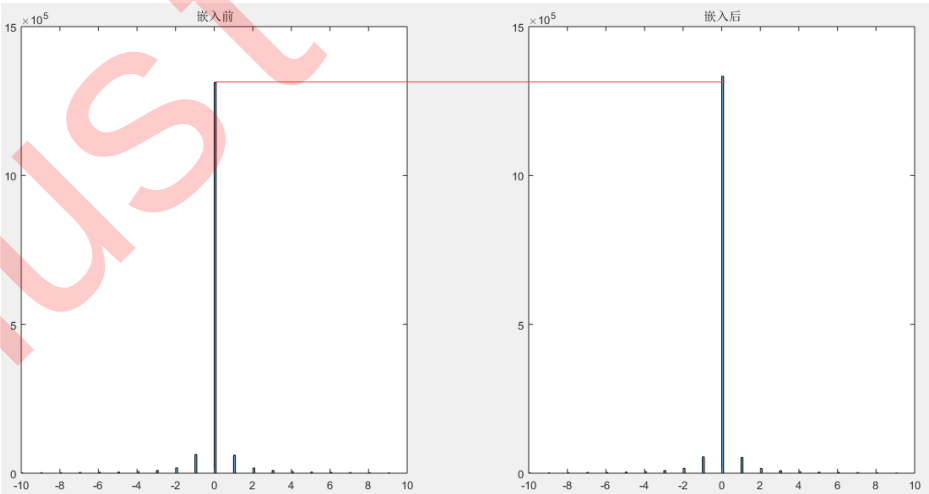

图 2.3-3 量化 DCT 系数直方图 1

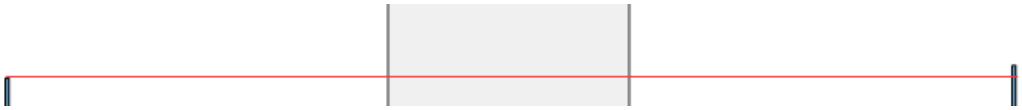

图 2.3-4 量化 DCT 系数直方图 1 放大

而-1 和 1 的个数都变少了，说明变成 0 的 1 比变成 1 的 2 更多，见下图。

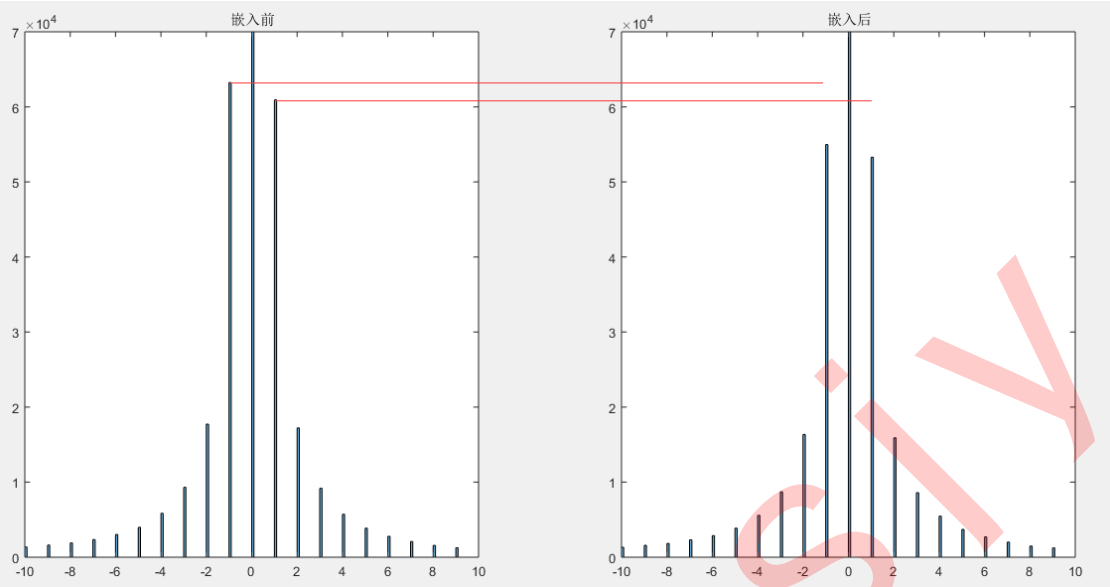

图 2.3-5 量化 DCT 系数直方图 2

观察量化 DCT 系数直方图的局部，发现嵌入前后 2i 和 2i+1 的频率差距几乎差不多，这是因为 F5 算法的修改方法和 F4 一样，相当于做平移变换，因此没有值对现象。

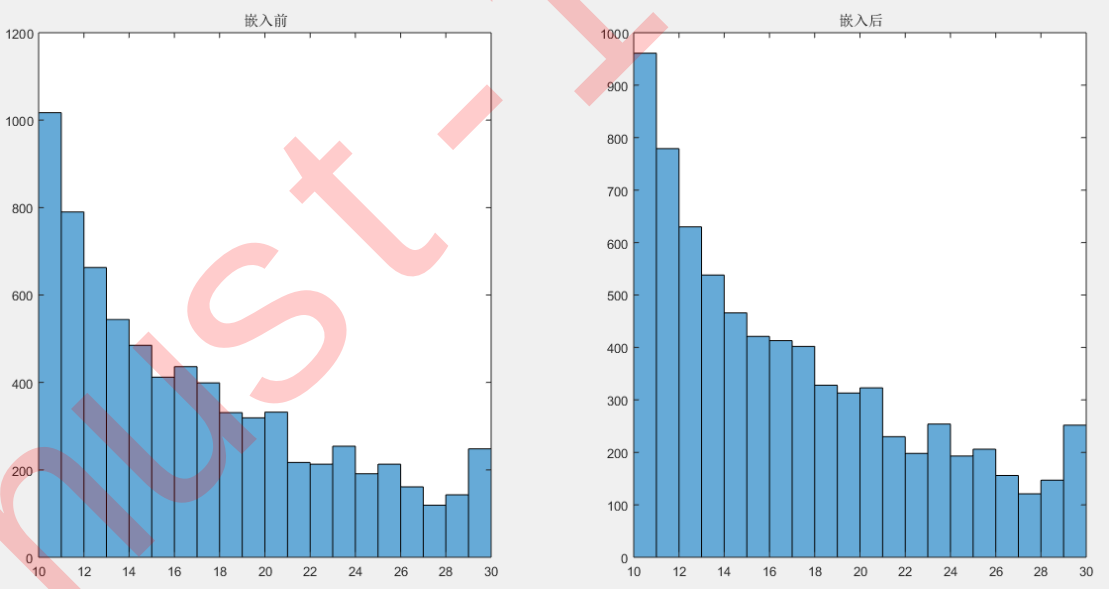

图 2.3-6 无值对现象

嵌入和提取的信息一致。

图 2.3-7 嵌入和提取的信息对比

在 F5 算法中，每修改 1 个 LSB 可以嵌入 2 比特信息，而在 F4 算法中，每修改 1 个 LSB 嵌入 1 比特信息。对于同样的秘密信息和载体图片，分别使用 F4 和 F5 加密，发现 F5 修改的 AC 系数个数的确少于 F4，但并不明显，时间上 F4 略快于 F5。

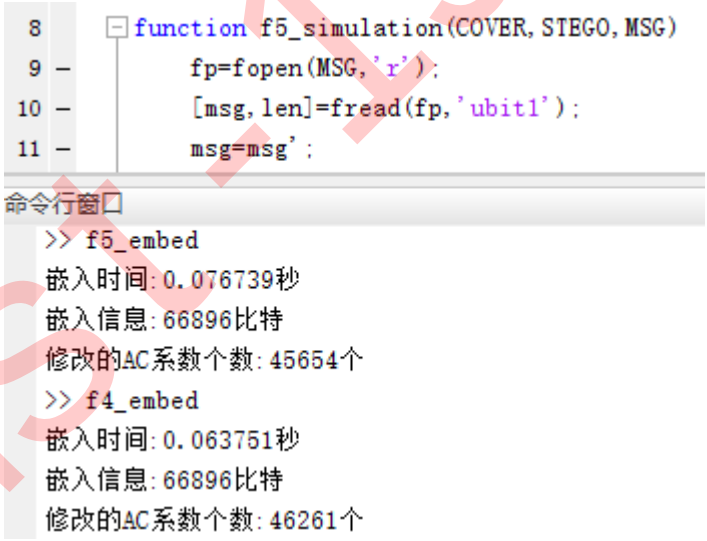

图 2.3-8 F4 和 F5 性能对比

3 实验小结

1) 学习了简单地使用 matlab。例如常见的图片处理函数 imread, imshow, rgb2gray，常见的绘图相关函数 figure, histogram, subplot, axis。掌握了如何在 matlab 中定义自己的函数以及使用封装好的 c 语言函数。
2) 加深了对空域和变换域信息隐藏的理解。空域信息隐藏，即直接替换灰度值的 LSB；变换域信息隐藏，即替换熵解码后量化 DCT 系数的 LSB。在空域进行信息隐藏，算法简单、嵌入容量大，但是在有损压缩后容易丢失信息，鲁棒性差。而变换域信息隐藏，相比前者，算法复杂、嵌入容量小，但由于信息被隐藏在 DCT 系数，鲁棒性更好。
3) JSTEG 算法进行信息隐藏会有值对现象，因此易被发现，而 F4 和 F5 算法都没有值对现象；JSTEG 不修改值为 0, -1 和 1 的 AC 系数，而 F4 和 F5 仅仅不修改值为 0 的 AC 系数，但是将-1 或 1 修改为 0 的嵌入是失败的。对于 F5 算法，每修改 1 个 AC 系数可以隐藏 2bit 秘密信息，而 F4 算法，每修改 1 个 AC 系数只能隐藏 1bit 秘密信息，因此对于同样长度的秘密信息，F5 修改的系数个数更少，时间也更短，但是对载体的利用率更低。

指导教师评定意见

1 对实验报告的评语

2 对实验报告的评语

评分项目（分值）	程序内容（36.8 分）	程序规范（9.2 分）	报告内容（36.8 分）	报告规范（9.2 分）	考勤（8 分）	逾期扣分	合　计（100 分）
得分							

附录 A JPEG 图像变换域信息隐藏实现的源程序

JSTEG_EXTRACT

```
COVER='test2.jpeg';
STEGO='jsteg.jpeg';
MSG='1.txt';

Jsteg_simulation(COVER,STEGO,MSG);

function Jsteg_simulation(COVER,STEGO,MSG)
    fp=fopen(MSG,'r');
    [msg,len]=fread(fp,'ubit1');
    msg=msg';
%     msg是信息的二进制，len是信息长度
    tic;
    try
        jobj=jpeg_read(COVER);
        DCT=jobj.coef_arrays{1};
        DCT2=DCT;
%         得到DCT
    catch
        error('ERROR(problem with the cover image)
');
    end

    AC=numel(DCT)-numel(DCT(1:8:end,1:8:end));
%     AC系数的个数

    if(length(msg)>AC)
        error('ERROR(too long message)');
    end
    pos=true(size(DCT));
    pos(1:8:end,1:8:end)=false;
    pos=find(pos);
%     DCT中非0元素的位置

    j=1;
    for i=1:len
%         找abs>1的AC系数
        while(abs(DCT(pos(j)))<=1)
            j=j+1;
            if(j>=AC)
                break;
            end
        end
%         嵌入第pos(j)个AC系数
        if(DCT(pos(j))>1)
            DCT2(pos(j))=DCT2(pos(j))-mod(DCT2(pos
(j)),2)+msg(i);
        else
            DCT2(pos(j))=DCT2(pos(j))-mod(DCT2(pos
(j))+1,2)+msg(i);
        end
        j=j+1;
        if(j>=AC)
            break;
        end
    end
%     直方图
%     figure();
%     subplot(121);histogram(DCT);axis([-10,10,0,7
*1e4]);title("嵌入前");
%     subplot(122);histogram(DCT2);axis([-10,10,0,
7*1e4]);title("嵌入后");
%     值对现象
%     figure();
%     subplot(121);histogram(DCT,(10:1:30));title
("嵌入前");
%     subplot(122);histogram(DCT2,(10:1:30));title
("嵌入后");
    try
        jobj.coef_arrays{1}=DCT2;
        jobj.optimize_coding=1;
        jpeg_write(jobj,STEGO);
    catch
        error('ERROR(problem with saving the stego
 image)')
    end
    fclose(fp);
```

```
    T=toc;
    fprintf('嵌入时间:%4f秒\n',T);
    fprintf('嵌入长度:%d\n',i);

%     图片对比
%     figure();
%     subplot(121);imshow(COVER);title("嵌入前");
%     subplot(122);imshow(STEGO);title("嵌入后");
end
```

JSTEG_EXTRACT

```
STEGO='jsteg.jpeg';
RES='Jsteg.txt';
len=52928;

Jsteg_simulation(STEGO,len,RES);

function Jsteg_simulation(STEGO,len,RES)
    fp=fopen(RES,'a');
    try
        jobj=jpeg_read(STEGO);
        DCT=jobj.coef_arrays{1};
    catch
        error('ERROR(problen with the STEGO imag)
');
    end

    pos=true(size(DCT));
    pos(1:8:end,1:8:end)=false;
    pos=find(pos);
    j=1;
    for i=1:len
        while(abs(DCT(pos(j)))<=1)
            j=j+1;
        end
        if(DCT(pos(j))>0)
            fwrite(fp,mod(DCT(pos(j)),2),'ubit1');
        elseif(DCT(pos(j))<0)
            fwrite(fp,mod(DCT(pos(j))+1,2),'ubit1
');
        end
        j=j+1;
    end
    fclose(fp);
end
```

F4_EMBED

```
COVER='test.jpeg';
STEGO='f4.jpeg';
MSG='1.txt';
f4_simulation(COVER,STEGO,MSG);

function f4_simulation(COVER,STEGO,MSG)
    fp=fopen(MSG,'r');
    [msg,len]=fread(fp,'ubit1');
    msg=msg';
%     计时、计数
    tic;
    sjy=0;
    try
        jobj=jpeg_read(COVER);
        DCT=jobj.coef_arrays{1};
        DCT2=DCT;
    catch
        error('ERROR(problem with the cover image)
');
    end

    AC=numel(DCT)-numel(DCT(1:8:end,1:8:end));

    if(length(msg)>AC)
        error('ERROR(too long message)');
    end

    pos=true(size(DCT));
    pos(1:8:end,1:8:end)=false;
    pos=find(pos);

    j=1;
    for i=1:len
        while(1)
            if((DCT(pos(j))==1 && msg(i)==0) || (D
CT(pos(j))==-1 && msg(i)==1) || (DCT(pos(j))==0))
                if(DCT(pos(j))~=0)
                    sjy=sjy+1;
                end
                DCT2(pos(j))=0;
                j=j+1;
                if(j>=AC)
                    break;
                end
            else
                break;
            end
        end
%         嵌入第pos(j)个AC系数
        if(j>=AC)
            break;
        end
        if(DCT(pos(j))>0 &&i)~=mod(DCT2(pos(j)),
2))
            sjy=sjy+1;
            DCT2(pos(j))=DCT2(pos(j))-1;
        elseif(DCT(pos(j))<0 && msg(i)~=mod(DCT2(p
os(j))+1,2))
            sjy=sjy+1;
            DCT2(pos(j))=DCT2(pos(j))+1;
        end
        j=j+1;
        if(j>=AC)
            break;
        end
    end
%     直方图
%     figure();
%     subplot(121);histogram(DCT);axis([-10,10,0,7
*1e4]);title("f4嵌入前");
%     subplot(122);histogram(DCT2);axis([-10,10,0,
7*1e4]);title("f4嵌入后");
%     无值对现象
%     figure();
%     subplot(121);histogram(DCT,(10:1:30));title
("f4嵌入前");
%     subplot(122);histogram(DCT2,(10:1:30));title
("f4嵌入后");
    try
        jobj.coef_arrays{1}=DCT2;
```

```
        jobj.optimize_coding=1;
        jpeg_write(jobj,STEGO);
    catch
        error('ERROR(problem with saving the stego
 image)')
    end
%     图片对比
%     figure();
%     subplot(121);imshow(COVER);title("嵌入前");
%     subplot(122);imshow(STEGO);title("嵌入后");
    T=toc;
    fprintf('嵌入时间:%4f秒\n',T);
    fprintf('嵌入信息:%d比特\n',i);
    fprintf('修改的AC系数个数:%d个\n',sjy);
    fclose(fp);
end
```

F4_EXTRACT

```
STEGO='f4.jpeg';
len=52928;
RES='f4.txt';

f4_simulation(STEGO,len,RES);

function f4_simulation(STEGO,len,RES)
    fp=fopen(RES,'a');
    try
        jobj=jpeg_read(STEGO);
        DCT=jobj.coef_arrays{1};
    catch
        error('ERROR(problen with the STEGO imag)
');
    end

    pos=true(size(DCT));
    pos(1:8:end,1:8:end)=false;
    pos=find(pos);
    j=1;
    for i=1:len
        while(abs(DCT(pos(j)))==0)
            j=j+1;
        end
        if(DCT(pos(j))>0)
            fwrite(fp,mod(DCT(pos(j)),2),'ubit1');
        else
            fwrite(fp,mod(DCT(pos(j))+1,2),'ubit1
');
        end
        j=j+1;
    end
    fclose(fp);
end
```

F5_EMBED

```
COVER='test.jpeg';
STEGO='f5.jpeg';
MSG='1.txt';

f5_simulation(COVER,STEGO,MSG);

function f5_simulation(COVER,STEGO,MSG)
    fp=fopen(MSG,'r');
    [msg,len]=fread(fp,'ubit1');
    msg=msg';
    tic;
    sjy=0;
    try
        jobj=jpeg_read(COVER);
        DCT=jobj.coef_arrays{1};
        DCT2=DCT;
    catch
        error('ERROR(problem with the cover image)
');
    end

    AC=numel(DCT)-numel(DCT(1:8:end,1:8:end));

    if(length(msg)>AC)
        error('ERROR(too long message)');
    end
    pos=true(size(DCT));
    pos(1:8:end,1:8:end)=false;
    pos=find(pos);

    i=1;
    j=1;
    k=1;
    thebit=1;
    a=[0,0,0];
    realpos=[0,0,0];
    while(i<=len)
        while(1)
            if(DCT(pos(j))==0)
                j=j+1;
                if(j>=AC)
                    break;
                end
            else
                realpos(k)=pos(j);
                if(DCT(pos(j))>0)
                    a(k)=mod(DCT(pos(j)),2);
                else
                    a(k)=mod(DCT(pos(j))+1,2);
                end
                k=k+1;
                j=j+1;
                if(k==4)
                    k=k-3;
                    break;
                end
            end
        end
        if(j>=AC)
            break;
        end
%         嵌入这三个DCT系数中的哪一个
        if(xor(a(1),a(2))~=msg(i) && xor(a(2),a
(3))==msg(i+1))
            sjy=sjy+1;
            thebit=1;
        elseif(xor(a(1),a(2))~=msg(i) && xor(a(2),
a(3))~=msg(i+1))
            sjy=sjy+1;
            thebit=2;
        elseif(xor(a(1),a(2))==msg(i) && xor(a(2),
a(3))~=msg(i+1))
            sjy=sjy+1;
            thebit=3;
        else
            k=1;
            i=i+2;
            continue;
        end
%         开始嵌入
```

```
        if( DCT(realpos(thebit))>0 )
            DCT2(realpos(thebit))=DCT2(realpos(the
bit))-1;
        elseif( DCT(realpos(thebit))<0)
             DCT2(realpos(thebit))=DCT2(realpos(th
ebit))+1;
        end
%          嵌入失败
       if(DCT2(realpos(thebit))==0)
           k=3;
           if(thebit==2)
               a(2)=a(3);
               realpos(2)=realpos(3);
           end
           if(thebit==1)
              a(1)=a(2);
              realpos(1)=realpos(2);
              a(2)=a(3);
              realpos(2)=realpos(3);
           end
       else
%             嵌入成功
           k=1;
           i=i+2;
       end
    end
%     直方图
%     figure();
%     subplot(121);histogram(DCT);axis([-10,10,0,7
*1e4]);title("f5嵌入前");
%     subplot(122);histogram(DCT2);axis([-10,10,0,
7*1e4]);title("f5嵌入后");
%     值对现象
%     figure();
%     subplot(121);histogram(DCT,(10:1:30));title
("f5嵌入前");
%     subplot(122);histogram(DCT2,(10:1:30));title
("f5嵌入后");
    try
        jobj.coef_arrays{1}=DCT2;
        jobj.optimize_coding=1;
        jpeg_write(jobj,STEGO);
    catch
        error('ERROR(problem with saving the stego
 image)')
    end
    T=toc;
    fprintf('嵌入时间:%4f秒\n',T);
    fprintf('嵌入信息:%d比特\n',i-1);
    fprintf('修改的AC系数个数:%d个\n',sjy);
    fclose(fp);
end
```

F5_EXTRACT

```
STEGO='f5.jpeg';
len=52928;
RES='f5.txt';
f5_simulation(STEGO,len,RES);

function f5_simulation(STEGO,len,RES)
    fp=fopen(RES,'a');
    try
        jobj=jpeg_read(STEGO);
        DCT=jobj.coef_arrays{1};
    catch
        error('ERROR(problen with the STEGO imag)
');
    end

    pos=true(size(DCT));
    pos(1:8:end,1:8:end)=false;
    pos=find(pos);
    a=[0,0,0];
    i=1;
    j=1;
    k=1;
    while(i<=len)
        while(1)
```

```
if(DCT(pos(j))==0)
    j=j+1;
else
    if(DCT(pos(j))>0)
        a(k)=mod(DCT(pos(j)),2);
    else
        a(k)=mod(DCT(pos(j))+1,2);
    end
    k=k+1;
    j=j+1;
    if(k==4)
                    k=k-3;
                    break;
                end
            end
        end
        fwrite(fp,xor(a(1),a(2)),'ubit1');
        fwrite(fp,xor(a(3),a(2)),'ubit1');
        i=i+2;
    end
        fclose(fp);
end
```