

**课 程 实 验 报 告**

**课程名称： 多媒体数据安全实验**

**专业班级： 网安**

**学 号： U201**

**姓 名：**

**指导教师：**

**报告日期： 2022/4/17**

**网络空间安全学院**

目 录

[1 JPEG图像变换域信息隐藏实现 2](#_Toc101184242)

[1 实验目的 2](#_Toc101184243)

[2 实现思路 2](#_Toc101184244)

[2.2 JSTEG算法 3](#_Toc101184245)

[2.3 F4算法 6](#_Toc101184246)

[2.4 F5算法 10](#_Toc101184247)

[3 实验小结 15](#_Toc101184248)

[附录A JPEG图像变换域信息隐藏实现的源程序 17](#_Toc101184249)

[1 信息隐藏stego.m 17](#_Toc101184250)

[2 信息提取extract.m 17](#_Toc101184251)

[3 信息隐藏和提取函数的具体实现 18](#_Toc101184252)

[3.1 JSteg 18](#_Toc101184253)

[3.2 F4 21](#_Toc101184254)

[3.3 F5 25](#_Toc101184255)

# 1 JPEG图像变换域信息隐藏实现

## 1 实验目的

通过实验达到⑴加深对变换域信息隐藏算法原理的理解；⑵熟悉数字图片JPEG压缩格式；(3)比较不同变换域信息隐藏算法在嵌入前后的DCT系数直方图特征。

## 2 实现思路

JPEG图片变换域信息隐藏算法从JSteg、F4到F5逐步改进，目的是尽量保留原始JPEG图片的JPEG系数特征。

信息嵌入流程如图1-1所示。在提取可供修改的AC系数方面，JSteg要求绝对值大于1，F4和F5要求非零；在替换规则方面，JSteg和F4中秘密消息与AC系数的最低位一一对应，具有替换关系，F5采用矩阵编码，每k位秘密消息对应每(2^k-1)位AC系数。

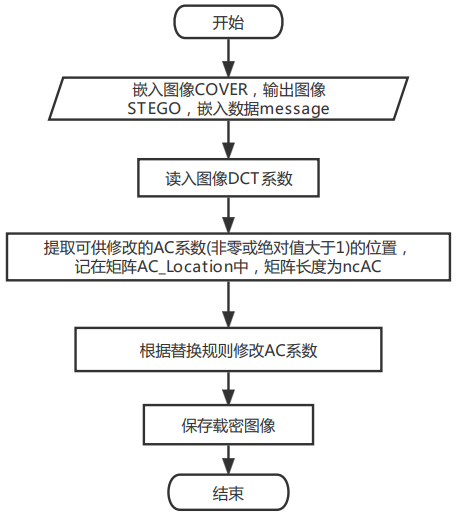


图1-1　信息嵌入流程

2.2 JSTEG算法

2.2.1 算法描述

JSteg信息隐藏算法是LSB替换思想在DCT域的实现。

嵌入过程的关键步骤：将原始图像的AC系数中最低的位平面“替换”为要隐藏的秘密信息。这里的“替换”遵循如下规则：

（1）忽略-1、0、1；

（2）若AC系数为2i，秘密比特为0，该系数不变；

（3）若AC系数为2i，秘密比特为1，该系数变为2i+1；

（4）若AC系数为2i+1，秘密比特为0，该系数变为2i；

（5）若AC系数为2i+1，秘密比特为1，该系数不变。

（6）AC系数为负数时，其二进制的实际含义是正数。(如表2-1所示)

表2-1　JSteg中的系数变化规则

|  |  |  |
| --- | --- | --- |
| 秘密比特  AC系数 | 0 | 1 |
| 正奇数 | -1 | 0 |
| 正偶数 | 0 | +1 |
| 负奇数 | 0 | +1 |
| 负偶数 | -1 | 0 |

如AC系数为(-16)₁₀ = (01111)₂，秘密比特为0时，载密系数是(-17)₁₀ = (01110)₂。

2.2.2 实现思路

JSteg中AC系数提取方式的实现如下：

（1）复制原DCT矩阵，存入AC\_Location矩阵中；

（2）将AC\_Location矩阵中的DC系数置0（即每个8×8矩阵的左上）；

（3）将AC\_Location矩阵中绝对值小于等于1的位置置为0；

（4）find函数找出AC\_Location矩阵中非0的位置，存入AC\_Location矩阵中，此时AC\_Location矩阵中记录的是可修改的AC系数的位置；

JSteg系数替换规则的实现如下：

（1）定义两个循环变量i\_DCT, i\_MSG，初值均为1，分别用来遍历可修改的AC系数和嵌入消息message；

（2）判断i\_MSG是否小于等于message长度，如果否，说明已经嵌入完成，转至（7）算法结束，否则顺序执行；

（3）判断i\_DCT是否大于AC总数，如果是，说明没有更多可供注入密文的AC系数，需要报错，转（7）算法结束；否则顺序执行；

（4）取DCT中第i\_DCT个AC数，存入DCTInfo变量；

（5）判断DCTInfo是正数还是负数，如果是正数，则令原DCT=DCTInfo+message(i\_MSG)-mod(DCTInfo,2)；如果是负数，则相当于DCTInfo的奇偶性发生逆转，令原DCT=DCTInfo+message(i\_MSG)-mod(DCTInfo+1,2)；

（6）i\_DCT=i\_DCT+1, i\_MSG=i\_MSG+1， 转（2），继续循环；

（7）替换算法结束。

JSteg系数提取算法的实现思路：遍历非-1,0,1的AC系数，取最低位；若AC系数是正数，则最低位为信息；若AC系数是负数，则最低位的非为信息。保存为messageHiden文件。

嵌入数据生成思路：MATLAB中有函数randi，可用来生成任意多指定条件的随机数。不妨将其作为“Z”字形扫描得到的比特流。

2.2.3实验结果及性能分析

嵌入信息是使用MATLAB随机生成的28684比特的0,1数组，保存为message文件。嵌入图片是JPG格式的512×512灰度图片。

运行嵌入和提取算法，结果如图2-1,2,3所示。

可以看到，嵌入28684比特数据耗时0.3724秒，提取耗时0.0102秒（如图2-1所示）。原始图像和载密图像相差不大，DCT直方图出现明显的“值对现象”（如图2-2所示），数据能够被成功嵌入和提取（如图2-3所示）。

实际上，经过调试，28684比特也是该图像的最大嵌入容量。选取的图像是512×512的，共262144比特，因此该图像的最大嵌入率约为10.94%。

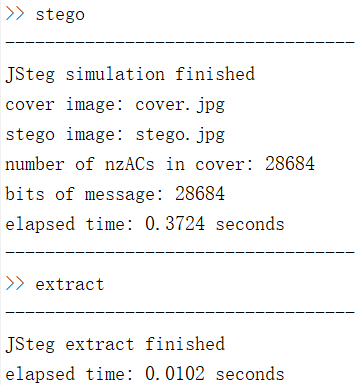


图2-1　JSteg嵌入和提取程序的终端输出

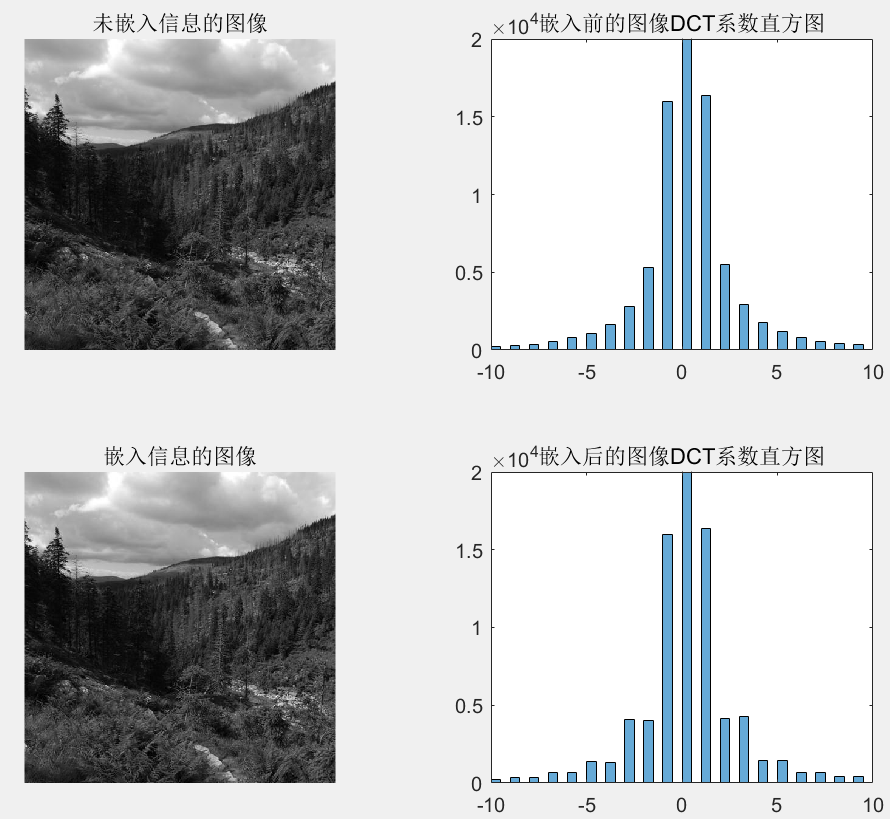


图2-2　JSteg信息隐藏前后图像和DCT系数直方图对比

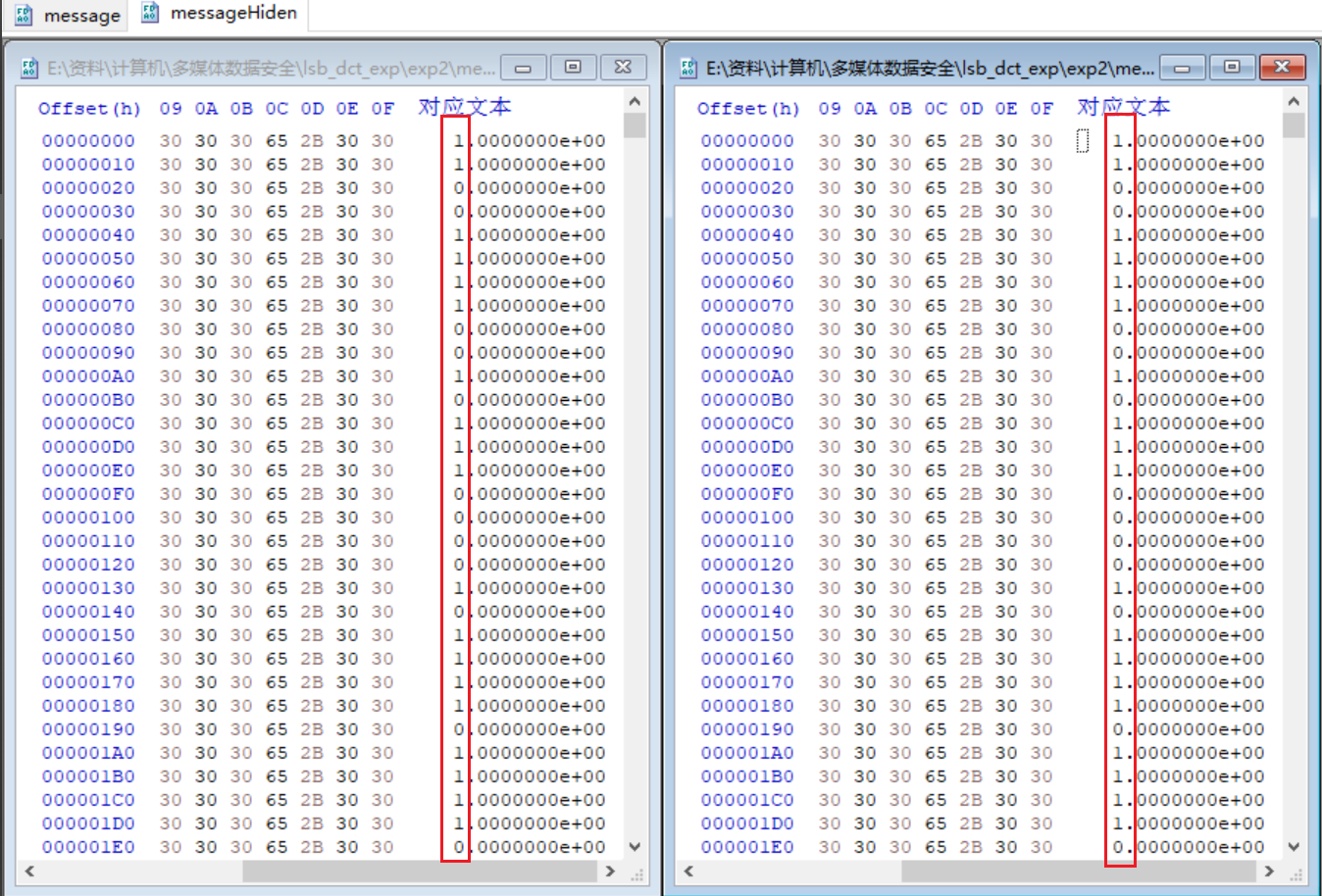


图2-3　JSteg信息隐藏的原信息与提取信息对比

综上，该算法具有简单、容易实现、容量可观的优点，但是由于DCT系数统计直方图特性，它容易被卡方攻击方法检测出秘密消息的存在。

2.3 F4算法

2.3.1 算法描述

F4信息隐藏算法是JSteg的改进。

在F4算法中，当嵌入系数的最低位和密文不符时，将系数的绝对值一律减1，而非对最低位取反，从而较好地避免了值对现象。

嵌入过程的关键步骤：将原始图像的AC系数中最低的位平面“替换”为要隐藏的秘密信息。这里的“替换”遵循如下规则：

（1）忽略0；

（2）当系数被修改成0时，换位置重新嵌入。

（3）正奇数和负偶数代表秘密消息1，正偶数和负奇数代表秘密消息0；

（4）AC系数为负数时，其二进制的实际含义是正数。

（5）具体变化规则如表2-2所示。

表2-2　F4中的系数变化规则

|  |  |  |
| --- | --- | --- |
| 秘密比特  AC系数 | 0 | 1 |
| 正奇数 | -1 | 0 |
| 正偶数 | 0 | -1 |
| 负奇数 | 0 | +1 |
| 负偶数 | +1 | 0 |

2.3.2 实现思路

F4中AC系数提取方式的实现如下：

（1）复制原DCT矩阵，存入AC\_Location矩阵中；

（2）将AC\_Location矩阵中的DC系数置0（即每个8×8矩阵的左上）；

（3）find函数找出AC\_Location矩阵中非0的位置，存入AC\_Location矩阵中，此时AC\_Location矩阵中记录的是可修改的AC系数的位置；

F4系数替换规则的实现如下：

（1）定义两个循环变量i\_DCT, i\_MSG，初值均为1，分别用来遍历DCT中的AC系数和嵌入消息message；

（2）判断i\_MSG是否小于等于message长度，如果否，说明已经嵌入完成，转至（8）算法结束，否则顺序执行；

（3）判断i\_DCT是否大于AC总数，如果是，说明没有更多可供注入密文的AC系数，需要报错，转（7）算法结束；否则顺序执行；

（4）取DCT中第i\_DCT个AC数，存入DCTInfo变量；

（5）判断DCTInfo是正奇数、正偶数、负奇数还是负偶数，正奇数时，令原DCT= DCTInfo+message(i\_MSG)-1；正偶数时，令原DCT= DCTInfo-message(i\_MSG)；负奇数时，令原DCT= DCTInfo+message(i\_MSG)；负偶数时，令原DCT= DCTInfo-message(i\_MSG)+1。

（6）判断嵌入后是否为0，如果是0，则i\_MSG=i\_MSG-1，相当于重新嵌入该秘密消息；

（7）i\_DCT=i\_DCT+1, i\_MSG=i\_MSG+1，转（2），继续循环；

（8）替换算法结束。

F4系数提取算法的实现思路：遍历非0的AC系数，取最低位；若AC系数是正数，则最低位为信息；若AC系数是负数，则最低位的非为信息。保存为messageHiden文件。

嵌入数据生成思路：MATLAB中有函数randi，可用来生成任意多指定条件的随机数。不妨将其作为“Z”字形扫描得到的比特流。

2.3.3实验结果及性能分析

嵌入信息是使用MATLAB随机生成的45000比特的0,1数组，保存为message文件。嵌入图片是JPG格式的512×512灰度图片。

运行嵌入和提取算法，结果如图2-4,5,6,7所示。

可以看到，嵌入45000比特数据耗时0.2246秒，提取耗时0.0108秒（如图2-4所示）。原始图像和载密图像相差不大，DCT直方图没有出现“值对现象”，但是0的数量明显增多（如图2-5所示），数据能够被成功嵌入和提取（如图2-6所示）。

实际上，F4的最大嵌入容量并不是定值，最大嵌入容量与嵌入的秘密信息有关。这是因为算法中将-1或1变成0时需要重新嵌入。不过，通过多次输出F4的最大嵌入比特（如图2-7所示），最大嵌入比特可估算为45000。选取的图像是512×512的，共262144比特，因此该图像的最大嵌入率可估计为17.17%。

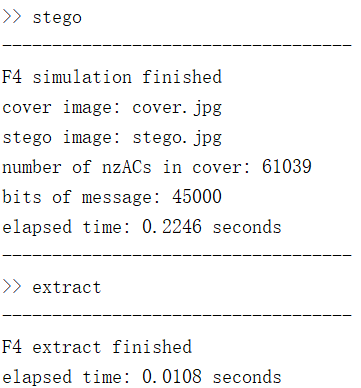


图2-4　F4嵌入和提取程序的终端输出

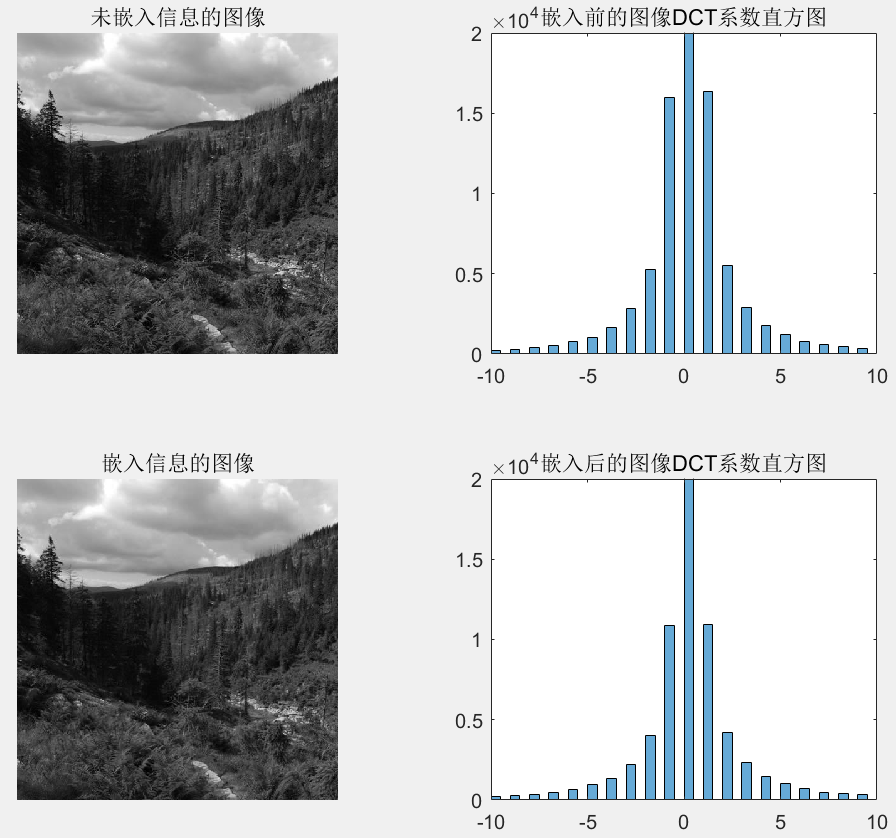


图2-5　F4信息隐藏前后图像和DCT系数直方图对比

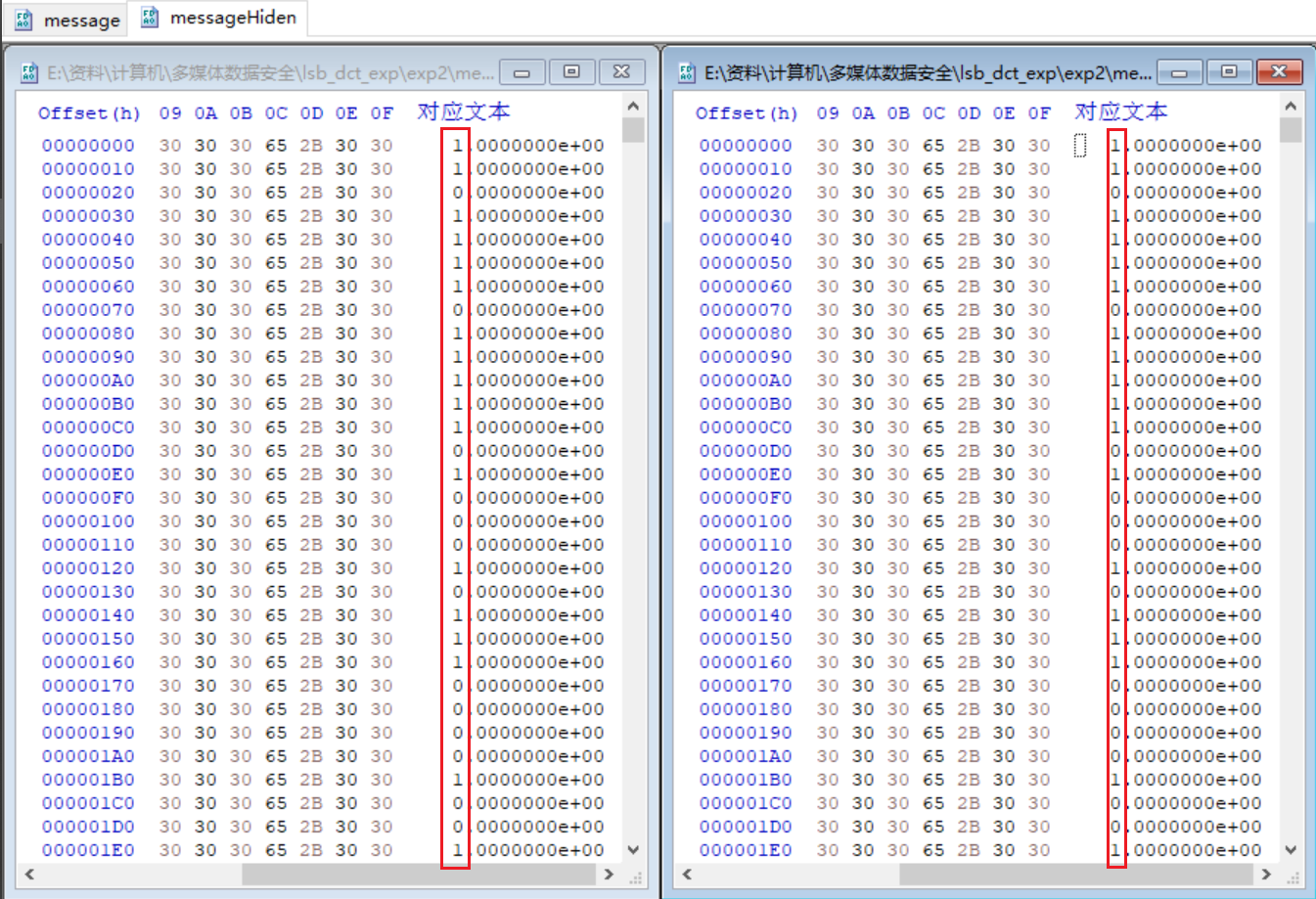


图2-6　F4信息隐藏的原信息与提取信息对比

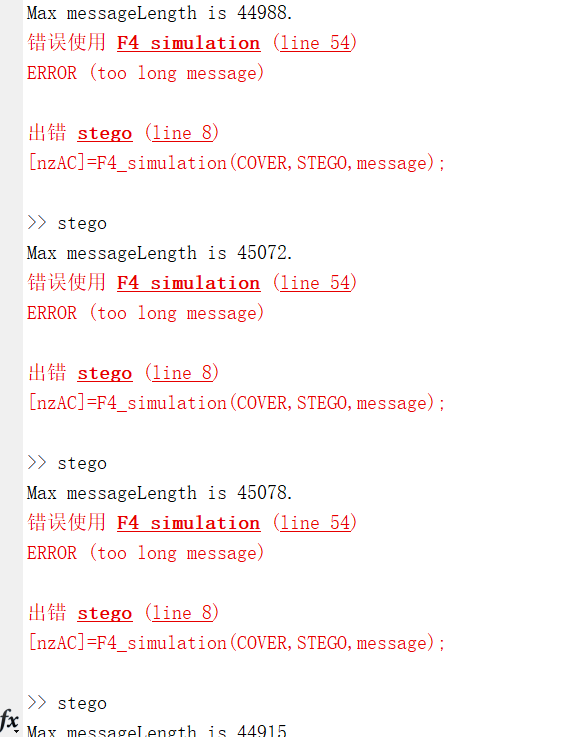


图2-7　F4同一图片的最大嵌入比特

综上，该算法相比JSteg，新增了避免值对现象、增大嵌入容量的优点，但是它将-1、1的值大大减少，也有被检测出来的新特点。并且，修改成0后需要重新嵌入，这一点还有可以改进的空间，能进一步提高嵌入率。

2.4 F5算法

2.4.1 算法描述

F5应用矩阵编码减小数据的修改量，并置乱DCT系数（本实验不要求置乱）。将k比特秘密消息嵌入到(2^k-1)个AC系数中，只修改1个位置。

在本实验中，选择一次嵌入量k为2，涉及的AC系数为3个，分别定义为b1,b2,a1,a2,a3。嵌入过程的关键步骤是矩阵编码，遵循如下规则：

（1）忽略0；

（2）当系数被修改成0时，换位置重新嵌入。

（3）b1=a1⊕a2, b2=a2⊕a3，则不修改数据；

（4）b1≠a1⊕a2, b2=a2⊕a3，则修改a1；

（5）b1＝a1⊕a2, b2≠a2⊕a3，则修改a3；

（6）b1≠a1⊕a2, b2≠a2⊕a3，则修改a2；

（7）正奇数和负偶数代表秘密消息1，正偶数和负奇数代表秘密消息0；

（8）AC系数为负数时，其二进制的实际含义是正数。

提取秘密消息时，只需令b1=a1⊕a2, b2=a2⊕a3。

2.4.2 实现思路

F5中AC系数提取方式的实现与F4相同。

F5系数替换规则的实现如下：

（1）定义两个循环变量i\_DCT, i\_MSG，初值均为1，分别用来遍历DCT中的AC系数和嵌入消息message；

（2）判断i\_MSG+1是否小于等于message长度（因为是每2位一组嵌入密文），如果否，说明已经嵌入完成，转至（8）算法结束，否则顺序执行；

（3）判断i\_DCT+2是否大于AC总数（因为是每3位一组利用AC系数），如果是，说明没有更多可供注入密文的AC系数，需要报错，转（7）算法结束；否则顺序执行；

（4）取DCT中第(i\_DCT到i\_DCT+2)个AC数，存入DCTInfo变量。将所有负数+1，改变最低位。将DCTInfo(1)和DCTInfo(2)取模后按位异或，结果存入xor1变量；将DCTInfo(2)和DCTInfo(3)取模后按位异或，结果存入xor2变量；

（5）定义需要修改的AC系数的位置为tobe\_change，初值为i\_DCT；

（6）若message(i\_MSG)=xor1, message(i\_MSG+1)=xor2，则不修改数据，tobe\_change=-1；若message(i\_MSG)≠xor1, message(i\_MSG+1)=xor2，则tobe\_change=i\_DCT；若message(i\_MSG) = xor1, message(i\_MSG+1)≠xor2，则tobe\_change=i\_DCT+2；若message(i\_MSG)≠xor1, message(i\_MSG+1)≠xor2，则tobe\_change=i\_DCT+1；

（7）修改DCT(AC\_Location(tobe\_change))，如果是正数，则减一，否则加一。

（8）判断嵌入后是否为0，如果是0，则i\_MSG=i\_MSG-2、从AC\_Location矩阵中去除第tobe\_change位、AC总数减1、i\_DCT=i\_DCT-3，相当于重新嵌入该秘密消息；

（9）i\_DCT=i\_DCT+3, i\_MSG=i\_MSG+2，转（2），继续循环；

（10）替换算法结束。

F5系数提取算法的实现思路：每3位遍历非0的AC系数，将所有负数+1，改变最低位。将前两位取模后按位异或、后两位取模后按位异或，结果依次存入为2位秘密消息。保存为messageHiden文件。

嵌入数据生成思路：MATLAB中有函数randi，可用来生成任意多指定条件的随机数。不妨将其作为“Z”字形扫描得到的比特流。

2.4.3 实验结果及性能分析

嵌入信息是使用MATLAB随机生成的32700比特的0,1数组，保存为message文件。嵌入图片是JPG格式的512×512灰度图片。

运行嵌入和提取算法，结果如图2-8,9,10,11所示。

可以看到，嵌入32700比特数据耗时3.0613秒，提取耗时0.1059秒（如图2-8所示）。原始图像和载密图像相差不大，DCT直方图没有出现“值对现象”，0的数量增多，但是比起F4算法，增多不明显（如图2-9所示），数据能够被成功嵌入和提取（如图2-10所示）。

实际上，F5的最大嵌入容量并不是定值，最大嵌入容量与嵌入的秘密信息有关。这是因为算法中将-1或1变成0时需要重新嵌入。不过，通过多次输出F5的最大嵌入比特（如图2-11所示），最大嵌入比特可估算为32700。选取的图像是512×512的，共262144比特，因此该图像的最大嵌入率可估计为12.47%。

最突出的问题是算法的耗时增加了一个数量级。

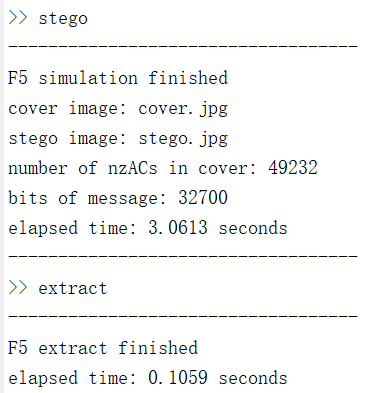


图2-8　F5嵌入和提取程序的终端输出

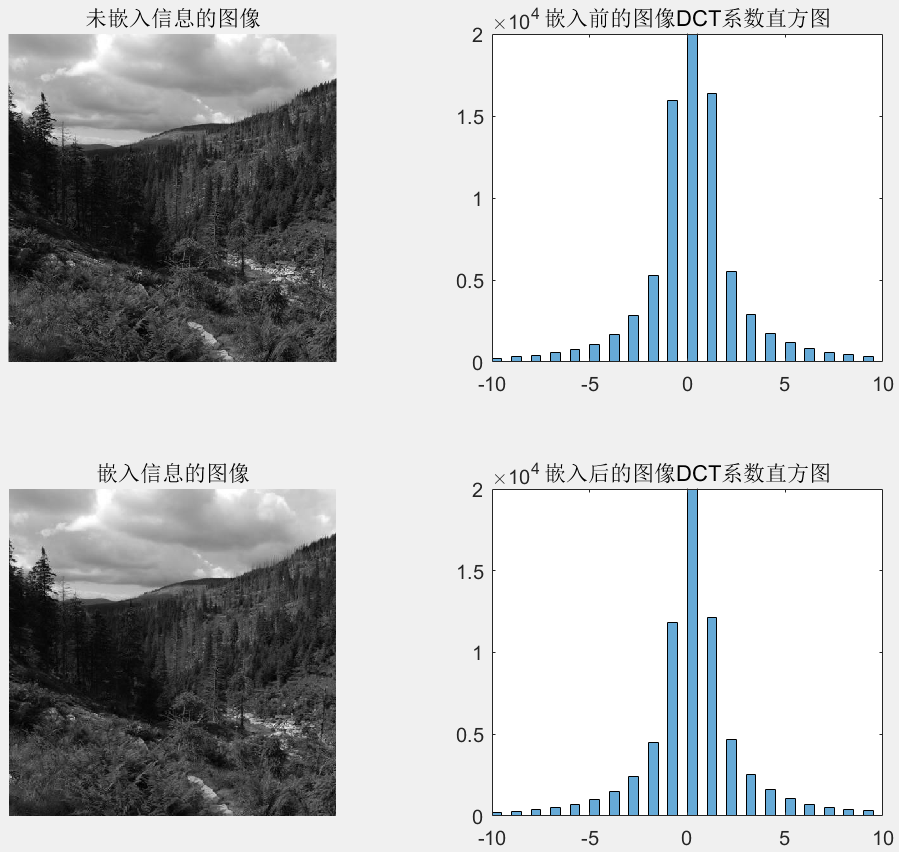


图2-9　F5信息隐藏前后图像和DCT系数直方图对比

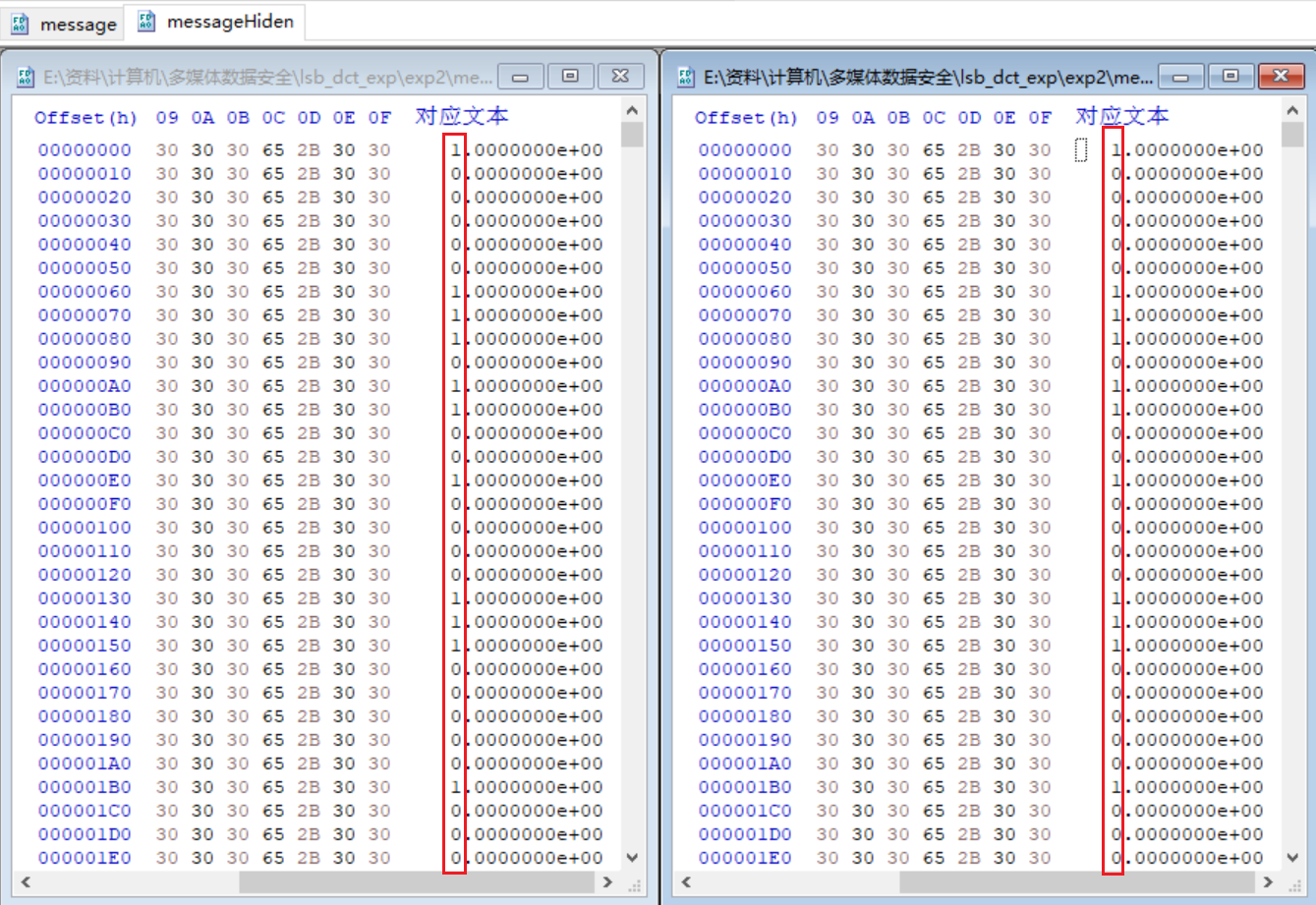


图2-10　F5信息隐藏的原信息与提取信息对比

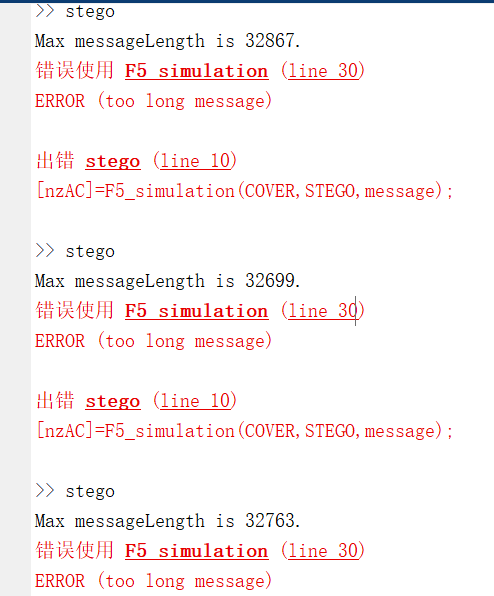


图2-11　F5同一图片的最大嵌入比特

综上，该算法相比JSteg和F4，更加复杂，运行效率远远小于JSteg和F4，但具有避免值对现象、数据修改量小的优点。同时，相比JSteg，它的嵌入容量有所增加；相比F4，它的嵌入容量明显减小。并且，相比F4，它的数据修改量较小，直方图中1和-1的减少更不明显，更难被DCT的统计规律检测出来。

## 3 实验小结

本次实验让我充分熟悉了MATLAB，也直观地感受到LSB信息隐藏和DCT域信息隐藏算法的优缺点。实验设计的难度适宜，收获满满。

第一个实验我在第一次实验课内做完了，第二个实验是课后做的，包括实验报告的撰写，耗时一天多。

我认为我本次编写的代码非常漂亮，逻辑很清楚，是本次实验的亮点。首先是因为MATLAB的语言本身就很美观，其次这是我投入精力最大的部分。

第二个实验的代码编写实际时间不长，几乎一半的时间都在思考怎样才能缩减代码体积、减少冗余的条件判断、复用代码。为此我对三个算法的变量都统一规范命名，并加入了许多基于MATLAB函数的优化。这不仅提高了代码可读性，也有利于我自己后续编写其他的算法。

其中第二个算法的代码，我是先在网上找过的，JSteg算法重点参考了文章<https://blog.csdn.net/WZZ_2363173126/article/details/118723301>，他的命名十分规范，结构清晰易懂，但是他并没有对负数特殊处理，而且程序内部的下标判断有明显的bug，逻辑也有些绕人。博主Dennis F还写了F4的隐藏算法，没有写F5，不过F4和F5我并未参考他的写法。

我修改了JSteg代码中的AC系数提取方式，直接跳过了不可写入的AC系数。并优化了替换规则的实现，减少了替换过程的条件分支。其中起到关键作用的是，MATLAB能够对整个数组的所有元素进行统一变换，比自己实现的循环和条件语句要方便、高效。之后，F4和F5均在JSteg的代码基础上完成。

在发现MATLAB一个又一个新函数和特性、完成一轮又一轮优化后，我很有成就感。不过与此同时，我也感到有些疲倦。

本次实验中，我的缺点是没有进一步分析算法的性能，一直采用同一张灰度图像。我知道我可以用MATLAB做多个图像的分析，多次运行算法，再把运行的数据写入表格中呈现出来。如果以后工具使用得更熟练，我会考虑做更丰富的性能分析。“下次一定！”

指导教师评定意见

一、对实验报告的评语

|  |
| --- |
|  |

二、对实验报告评分

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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

## 1 信息隐藏stego.m

% 更改嵌入算法时，需要将下文JSteg\_simulation替换成其他函数

% 并修改算法名称name、嵌入信息长度messageLen等变量

COVER='cover.jpg';

STEGO='stego.jpg';

name='JSteg';

messageLen=24000;

message=randi([0 1],1,messageLen);%0000); %生成随机数，作为隐藏信息

save('message','message','-ascii'); %保存秘密信息

tic;

[nzAC]=JSteg\_simulation(COVER,STEGO,message);

T=toc;

fprintf('-----------------------------------\n');

fprintf('%s simulation finished\n', name);

fprintf('cover image: %s\n',COVER);

fprintf('stego image: %s\n',STEGO);

fprintf('number of nzACs in cover: %i\n',nzAC);

fprintf('bits of message: %d\n',length(message));

fprintf('elapsed time: %.4f seconds\n',T);

fprintf('-----------------------------------\n');

## 2 信息提取extract.m

% 更改提取算法时，需要将下文JSteg\_extract替换成其他函数

% 并修改算法名称name、嵌入信息长度messageLen等变量

STEGO='stego.jpg';

name='F5';

messageLen=32700;

tic;

messageHiden=JSteg\_extract(STEGO,messageLen);

T=toc;

save('messageHiden','messageHiden','-ascii'); %保存提取出来的秘密信息

fprintf('-----------------------------------\n');

fprintf('%s extract finished\n', name);

fprintf('elapsed time: %.4f seconds\n',T);

fprintf('-----------------------------------\n');

## 3 信息隐藏和提取函数的具体实现

3.1 JSteg

3.1.1 JSteg\_simulation.m

function [ncAC]=JSteg\_simulation(COVER,STEGO,message)

try

jobj=jpeg\_read(COVER); %读取cover图片

PrimeDCT=jobj.coef\_arrays{1};%读取DCT系数

DCT=PrimeDCT;

catch

error('ERROR (problem with the cover image)');

end

AC\_Location=DCT; % 复制DCT

AC\_Location(1:8:end,1:8:end)=false; % 将DC系数置0

AC\_Location(abs(AC\_Location)<=1)=0; % 将绝对值小于等于1的位置置为0

AC\_Location=find(AC\_Location); %找出DCT中不为DC系数、绝对值大于1的位置

ncAC=numel(AC\_Location); % 得到绝对值大于1的AC系数个数

messageLen=length(message);

%信息过长

if(messageLen>ncAC)

error('ERROR (too long message)');

end

i\_DCT=1;

for i\_MSG=1:messageLen

if(i\_DCT>ncAC) % 没有更多可供注入密文的AC系数

fprintf('Max messageLength is %d.\n', i\_MSG-1);

error('ERROR (too long message)');

end

DCTInfo=DCT(AC\_Location(i\_DCT));

if(DCTInfo>0)

DCT(AC\_Location(i\_DCT))=DCTInfo+message(i\_MSG)-mod(DCTInfo,2);

else

DCT(AC\_Location(i\_DCT))=DCTInfo+message(i\_MSG)-mod(DCTInfo+1,2);

end

i\_DCT=i\_DCT+1;

end

%%% save the resulting stego image

try

jobj.coef\_arrays{1} = DCT;

jobj.optimize\_coding = 1;

jpeg\_write(jobj,STEGO);

catch

error('ERROR (problem with saving the stego image)');

end

%显示图像

subplot(2,2,1);imshow(COVER);

title('未嵌入信息的图像');

subplot(2,2,2);histogram(PrimeDCT);axis([-10,10,0,2\*1e4]);

title('嵌入前的图像DCT系数直方图');

subplot(2,2,3);imshow(STEGO);

title('嵌入信息的图像');

subplot(2,2,4);histogram(DCT);axis([-10,10,0,2\*1e4]);

title('嵌入后的图像DCT系数直方图');

3.1.2 JSteg\_extract.m

function message=JSteg\_extract(STEGO,messageLen)

try

jobj=jpeg\_read(STEGO); %读取stego图片

DCT=jobj.coef\_arrays{1};%读取DCT系数

catch

error('ERROR (problem with the cover image)');

end

AC\_Location=DCT; % 复制DCT

AC\_Location(1:8:end,1:8:end)=false; % 将DC系数置0

AC\_Location(abs(AC\_Location)<=1)=0; % 将绝对值小于等于1的位置置为0

AC\_Location=find(AC\_Location); %找出DCT中不为DC系数、绝对值大于1的位置

i\_DCT=1;

for i\_MSG=1:messageLen

DCTInfo=DCT(AC\_Location(i\_DCT));

if(DCTInfo>0)

message(1,i\_MSG)=mod(DCTInfo,2);

else

message(1,i\_MSG)=mod(DCTInfo+1,2);

end

i\_DCT=i\_DCT+1;

end

3.2 F4

3.2.1 F4\_simulation.m

function [ncAC]=JSteg\_simulation(COVER,STEGO,message)

try

jobj=jpeg\_read(COVER); %读取cover图片

PrimeDCT=jobj.coef\_arrays{1};%读取DCT系数

DCT=PrimeDCT;

catch

error('ERROR (problem with the cover image)');

end

AC\_Location=DCT; % 复制DCT

AC\_Location(1:8:end,1:8:end)=false; % 将DC系数置0

AC\_Location=find(AC\_Location); %找出DCT中不为DC系数、不为0的位置

ncAC=numel(AC\_Location); % 得到非0的AC系数个数

messageLen=length(message);

%信息过长

if(messageLen>ncAC)

error('ERROR (too long message)');

end

i\_DCT=1;

i\_MSG=1;

while i\_MSG<=messageLen

if(i\_DCT>ncAC) % 没有更多可供注入密文的AC系数

fprintf('Max messageLength is %d.\n', i\_MSG-1);

error('ERROR (too long message)');

end

DCTInfo=DCT(AC\_Location(i\_DCT));

if(DCTInfo>0)

if(mod(DCTInfo,2)) % 正奇数

DCT(AC\_Location(i\_DCT))=DCTInfo+message(i\_MSG)-1;

else % 正偶数

DCT(AC\_Location(i\_DCT))=DCTInfo-message(i\_MSG);

end

else

if(mod(DCTInfo,2)) % 负奇数

DCT(AC\_Location(i\_DCT))=DCTInfo+message(i\_MSG);

else % 负偶数

DCT(AC\_Location(i\_DCT))=DCTInfo-message(i\_MSG)+1;

end

end

if(DCT(AC\_Location(i\_DCT))==0) % 如果嵌入后是0，该密文重新嵌入

i\_MSG=i\_MSG-1;

end

i\_DCT=i\_DCT+1;

i\_MSG=i\_MSG+1;

end

%%% save the resulting stego image

try

jobj.coef\_arrays{1} = DCT;

jobj.optimize\_coding = 1;

jpeg\_write(jobj,STEGO);

catch

error('ERROR (problem with saving the stego image)');

end

%显示图像

subplot(2,2,1);imshow(COVER);

title('未嵌入信息的图像');

subplot(2,2,2);histogram(PrimeDCT);axis([-10,10,0,2\*1e4]);

title('嵌入前的图像DCT系数直方图');

subplot(2,2,3);imshow(STEGO);

title('嵌入信息的图像');

subplot(2,2,4);histogram(DCT);axis([-10,10,0,2\*1e4]);

title('嵌入后的图像DCT系数直方图');

3.2.2 F4\_extract.m

function message=F4\_extract(STEGO,messageLen)

try

jobj=jpeg\_read(STEGO); %读取stego图片

DCT=jobj.coef\_arrays{1};%读取DCT系数

catch

error('ERROR (problem with the cover image)');

end

AC\_Location=DCT; % 复制DCT

AC\_Location(1:8:end,1:8:end)=false; % 将DC系数置0

AC\_Location=find(AC\_Location); %找出DCT中不为DC系数、不为0的位置

i\_DCT=1;

for i\_MSG=1:messageLen

DCTInfo=DCT(AC\_Location(i\_DCT));

if(DCTInfo>0)

message(1,i\_MSG)=mod(DCTInfo,2);

else

message(1,i\_MSG)=mod(DCTInfo+1,2);

end

i\_DCT=i\_DCT+1;

end

3.3 F5

3.3.1 F5\_simulation.m

function [ncAC]=F5\_simulation(COVER,STEGO,message)

try

jobj=jpeg\_read(COVER); %读取cover图片

PrimeDCT=jobj.coef\_arrays{1};%读取DCT系数

DCT=PrimeDCT;

catch

error('ERROR (problem with the cover image)');

end

AC\_Location=DCT; % 复制DCT

AC\_Location(1:8:end,1:8:end)=false; % 将DC系数置0

AC\_Location=find(AC\_Location); %找出DCT中不为DC系数、不为0的位置

ncAC=numel(AC\_Location); % 得到AC系数总数

messageLen=length(message); % 得到密文的数量

%信息过长

if(messageLen/2>ncAC/3)

error('ERROR (too long message)');

end

i\_DCT=1;

i\_MSG=1;

while i\_MSG+1<=messageLen

if(i\_DCT+2>ncAC) % 没有更多可供注入密文的AC系数

fprintf('Max messageLength is %d.\n', i\_MSG-2);

error('ERROR (too long message)');

end

DCTInfo=DCT(AC\_Location(i\_DCT:i\_DCT+2));

DCTInfo(DCTInfo<0)=DCTInfo(DCTInfo<0)+1; % 将所有负数+1，改变最低位

xor1=bitxor(mod(DCTInfo(1),2),mod(DCTInfo(2),2)); % a1按位异或a2

xor2=bitxor(mod(DCTInfo(2),2),mod(DCTInfo(3),2)); % a2按位异或a3

tobe\_change=i\_DCT;

if(message(i\_MSG)==xor1) % 判断异或值，确定修改位

if(message(i\_MSG+1)==xor2)

tobe\_change=-1;

else

tobe\_change=tobe\_change+2;

end

else

if(message(i\_MSG+1)~=xor2)

tobe\_change=tobe\_change+1;

end

end

if(tobe\_change~=-1) % 需要修改

if(DCT(AC\_Location(tobe\_change))>0) % 正数减1

DCT(AC\_Location(tobe\_change))=DCT(AC\_Location(tobe\_change))-1;

else % 负数加1

DCT(AC\_Location(tobe\_change))=DCT(AC\_Location(tobe\_change))+1;

end

if(DCT(AC\_Location(tobe\_change))==0) % 如果嵌入后是0，这两位密文重新嵌入

i\_MSG=i\_MSG-2;

AC\_Location(tobe\_change)=[]; % 删除这一位置

ncAC=ncAC-1;

i\_DCT=i\_DCT-3; % 复用未被修改的AC系数

end

end

i\_DCT=i\_DCT+3;

i\_MSG=i\_MSG+2; % for循环内循环索引改变不生效

end

%%% save the resulting stego image

try

jobj.coef\_arrays{1} = DCT;

jobj.optimize\_coding = 1;

jpeg\_write(jobj,STEGO);

catch

error('ERROR (problem with saving the stego image)');

end

%显示图像

subplot(2,2,1);imshow(COVER);

title('未嵌入信息的图像');

subplot(2,2,2);histogram(PrimeDCT);axis([-10,10,0,2\*1e4]);

title('嵌入前的图像DCT系数直方图');

subplot(2,2,3);imshow(STEGO);

title('嵌入信息的图像');

subplot(2,2,4);histogram(DCT);axis([-10,10,0,2\*1e4]);

title('嵌入后的图像DCT系数直方图');

3.3.2 F5\_extract.m

function message=F5\_extract(STEGO,messageLen)

try

jobj=jpeg\_read(STEGO); %读取stego图片

DCT=jobj.coef\_arrays{1};%读取DCT系数

catch

error('ERROR (problem with the cover image)');

end

AC\_Location=DCT; % 复制DCT

AC\_Location(1:8:end,1:8:end)=false; % 将DC系数置0

AC\_Location=find(AC\_Location); %找出DCT中不为DC系数、不为0的位置

i\_DCT=1;

for i\_MSG=1:2:messageLen

DCTInfo=DCT(AC\_Location(i\_DCT:i\_DCT+2));

DCTInfo(DCTInfo<0)=DCTInfo(DCTInfo<0)+1; % 将所有负数+1后，改变最低位

message(1,i\_MSG)=bitxor(mod(DCTInfo(1),2),mod(DCTInfo(2),2)); % a1按位异或a2

message(1,i\_MSG+1)=bitxor(mod(DCTInfo(2),2),mod(DCTInfo(3),2)); % a2按位异或a3

i\_DCT=i\_DCT+3;

end