基于MATLAB的空域替换算法的信息隐写系统(LSB/MLSB/随机替换/调色板替换)

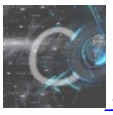
原创

芸兮 已于 2022-04-15 08:47:38 修改 2514 收藏 29

分类专栏： 信息安全 文章标签： matlab 密码学

于 2020-06-09 09:55:12 首次发布

本文链接：https://blog.csdn.net/baiduwaimai/article/details/106634308



文章目录

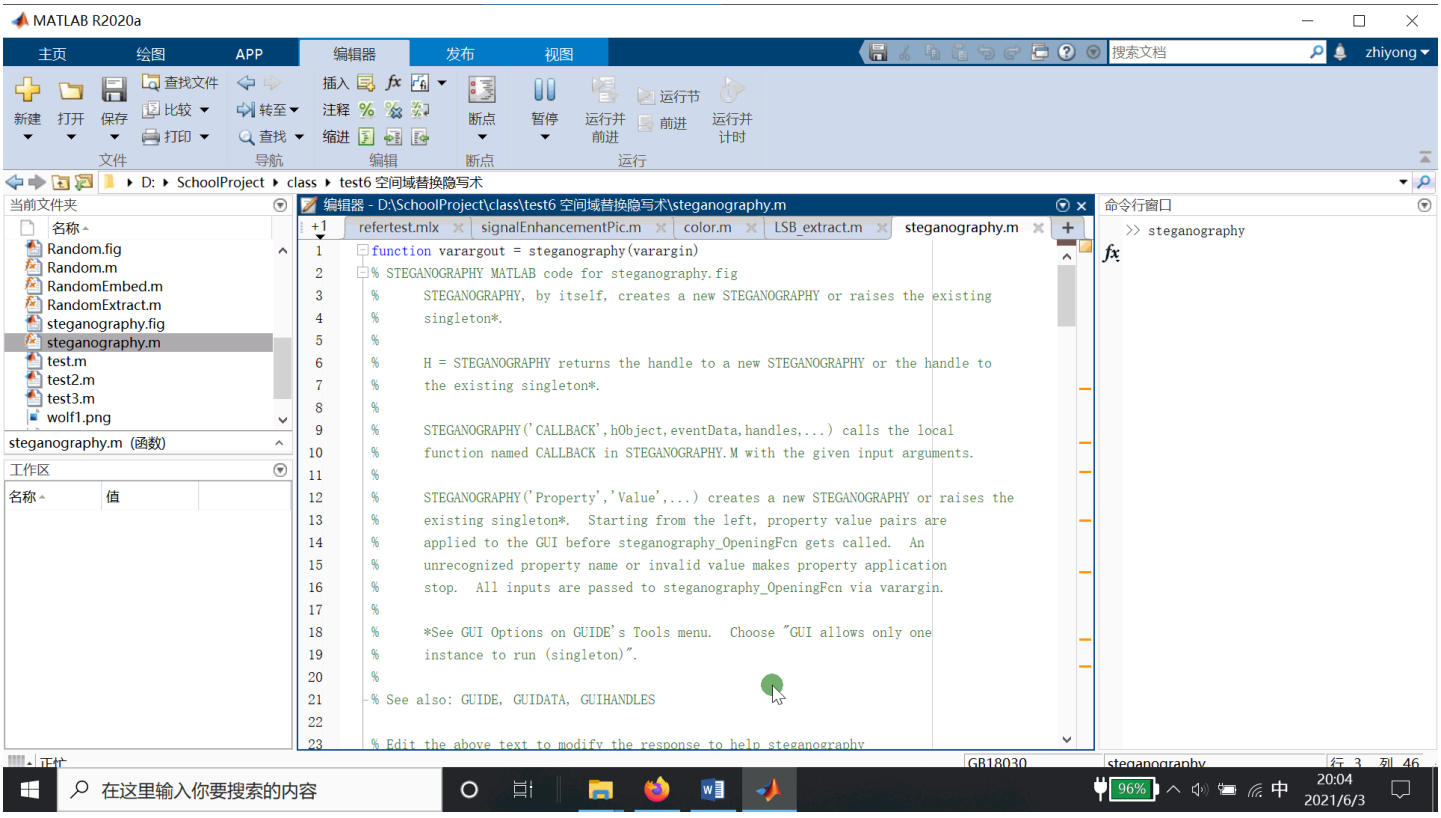
序

一、LSB/MLSB隐写算法

二、随机替换算法

三、 调色板换序算法

（源码可以有常私我vx:xdsqczkyqs713，仅源码40r，
加设计说明报告书50r）



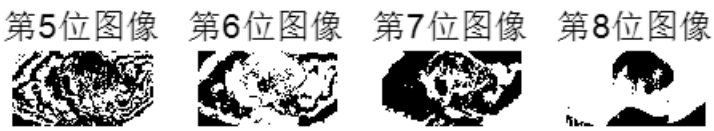
序

隐写术是一门关于信息隐藏的技巧与科学，其中利用数字图像作为隐藏载体来伪装的方法称为数字图像隐写算法。本文探究一种基于空域替换算法的信息隐写系统，支持常见的LSB算法，MLSB算法，伪随机替换算法，索引图像调色板换序算法，说明空域替换图像隐写术的原理，并在matlab平台实践隐写术实施的具体流程。

一、LSB/MLSB隐写算法

LSB（Least Significant Bit），最低有效位，该算法的基本原理是将载体图像的LSB篡改为信息者想要发送的秘密信息的二进制位。LSB隐写算法是一种简单且高效的信息隐藏技术。对于PNG或者BMP图像，一般是由RGB三原色（红绿蓝）组成，每种颜色占用8bit即一个字节的空间，每个像素的像素值范围为 0x00~0xFF，对于彩色图来说，总共有 256^3种颜色。所以如果用LSB方法隐藏信息，每个像素可以隐藏3Bit的信息。LSB隐写通过修改RGB颜色分量的最低二进制位（LSB），即使图像像素当中最不重要的那一位发生改变也无法使肉眼察觉出来。
以下边这幅灰度图为例：

将他的每个比特位代表的像素值单独拿出来就得到下边这幅图：



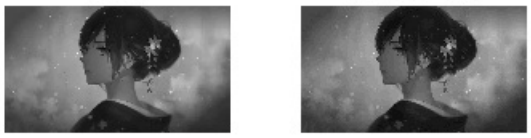
从上图可以看出，低比特位像素数据对反映图像信息贡献不大，因此可以用最低位平面作为我们隐藏信息的图层。
而MLSB旨在不被人眼察觉出异常的情况下，尽可能的多藏几位信息，从而提高消息隐藏的效率和容量。从下边的这幅图我们可以看到，在保留低三位图的时候仍然没有相应的图像轮廓出现，即对图像成型贡献不大，应此MLSB指最大隐藏图像位数，一般取3位。

低1位图像 低2位图像 低3位图像 低4位图像

低5位图像 低6位图像 低7位图像 低8位图像

为了进一步证实，可将低三位的位图像去掉，与原图进行对比，用人眼观察是否有明显不同，如下所示：

原图 去掉低三位



下边在matlab上实现隐写术工作流程：
首先强调的一点是载体图片的格式问题，png图片是一种无损压缩的位图片形格式，也只有在无损压缩或者无压缩的图片（BMP）上实现lsb隐写。如果图像是jpg图片的话，就没法使用lsb隐写了，原因是jpg图片对像数进行了有损压缩，我们修改的信息就可能会在压缩的过程中被破坏。而png图片虽然也有压缩，但却是无损压缩，这样我们修改的信息也就能得到正确的表达，不至于丢失。BMP的图片也是一样的，是没有经过压缩的。BMP图片一般是特别的大的，因为BMP把所有的像数都按原样储存，没有进行压缩。png图片中的图像像数一般是由RGB三原色（红绿蓝）组成，每一种颜色占用8位，取值范围为0x00~0xFF，即有256种颜色，一共包含了256的3次方的颜色，即16777216种颜色。而人类的眼睛可以区分约1000万种不同的颜色，这就意味着人类的眼睛无法区分余下的颜色大约有6777216种。

代码流程：

1、LSB/MLSB消息嵌入编写函数实现图片变量中最低位嵌入隐藏消息
函数输入:图片变量名、消息内容、需要隐藏到最低几位、隐藏在RGB三图层的哪一个
函数输出：嵌入消息的图片变量
首先对message进行编码，在此使用‘UTF-8’编码，为了方便后续信息提取，设置一个结束位使用strcat函数连接在message后边。实验中设置的结束标志位为char‘4’，并使用unicode2native函数将Unicode码转为数字字节。
编码后的消息是以字符char的格式存储的，且形式为十进制，接下来，将字符消息转换为二进制，并把这些二进制编码整合到一个数组msg中。
将msg中的‘0’、‘1’按序隐藏在图片的低lsb（设置的隐藏位数）位中，每个像素点隐藏lsb个，则所需的像素点载体为length(msg)/lsb,如果不足则在消息末尾再次补位，注意这里补位的内容还是结束标志位char‘4‘。
像素点的数据格式为uint8（无符号整型),表示范围为0~255的使用八位二进制数表示的十进制数据，对数据取模2^lsb，则得到数据的低lsb位，将得到的低lsb位数据减去，加上msg隐藏信息（将lsb位隐藏信息转为十进制后再运算），这样就完成了低位数据的替换。返回替换后的，既隐藏了消息的图片变量。下边给出LSB/MLSB嵌入程序：

```
function [image_result] = LSBembed(image, message, lsb, color)
% lsbembed
% image: the picture's matrix name
% message: the data you want to hide in the picture
% lsb: lsb-rightmost LSBs
% color: 1-red, 2-green, 3-blue

msg_origin = unicode2native(strcat(message, char(4)), 'UTF-8');  % UTF-8 encode, 'EOT' is the end tag
msg_bin = dec2bin(msg_origin, 8);  % convert to binary
msg = strjoin(cellstr(msg_bin)','');

len = length(msg) / lsb;
while len ~= fix(len)
    strcat(msg, char(4));
    len = length(msg) / lsb;
end
tmp = blanks(len);
for i = 1 : len
    tmp(i) = char(bin2dec(msg((i - 1) * lsb + 1 : i * lsb)));
end
tmp=double(tmp);

% use Red, Green or Blue
layer = image(:, :, color);
for i = 1 : len
    layer(i) = layer(i) - mod(layer(i), 2^lsb) + tmp(i);% only to be consistent with front
end
% save the picture
image_result = image;
image_result(:, :, color) = layer;
% imshow(image_result);
% imwrite(image_result, 'result.png');  %
end
```

2、LSB/MLSB消息提取 在上边嵌入程序的基础上编写函数LSB_extract进行信息的提取。 函数输入：嵌入了消息的图片变量名、需要隐藏到最低几位、隐藏在RGB三图层的哪一个 函数输出：消息内容 从消息的角度来说，消息的提取基本上是消息嵌入的逆过程（图片载体不可逆）

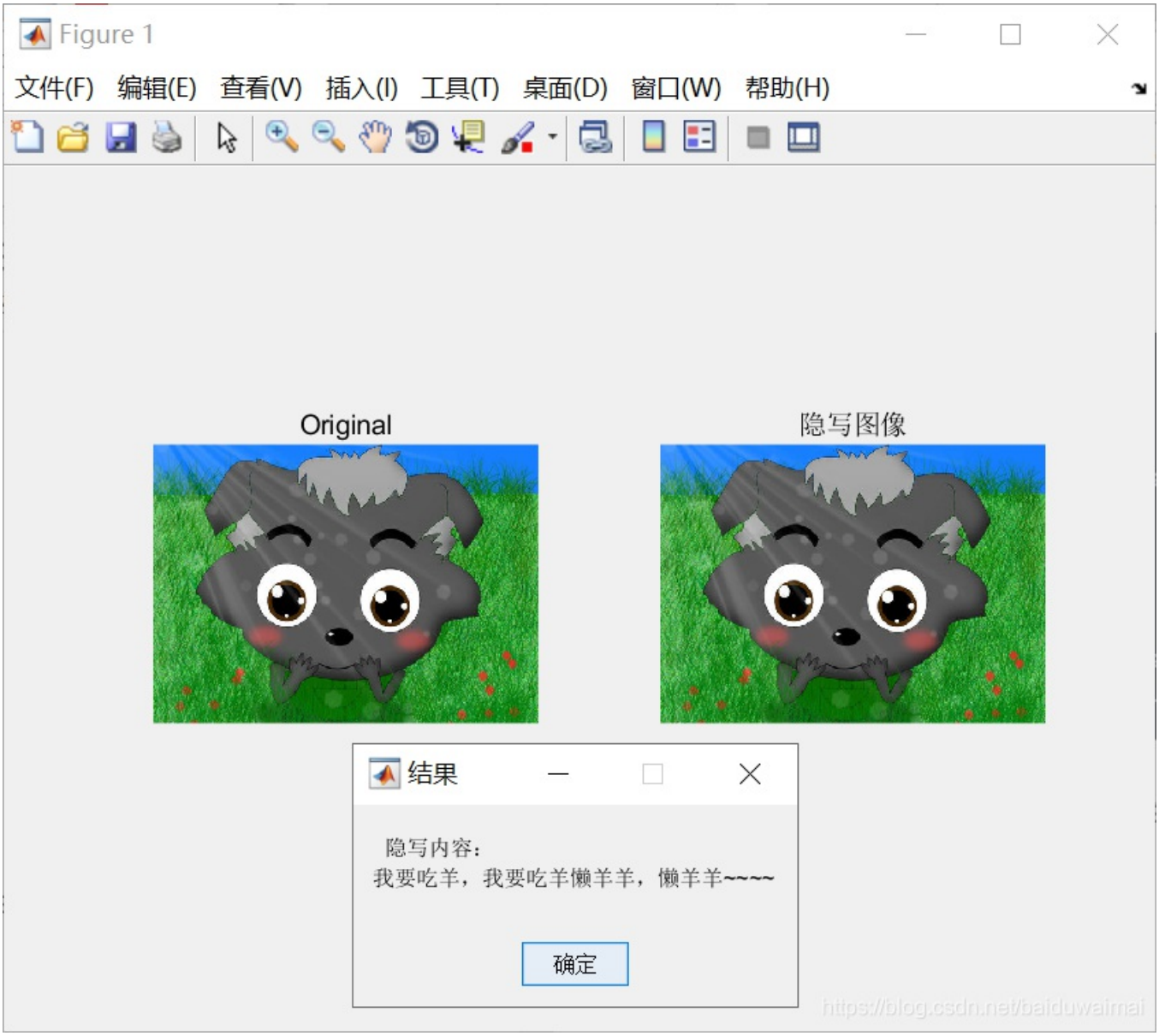
先获取每个像素点的低lsb位，因为编码采用的’UTF-8’，因此每提取出8位，整合到一起，转化为十进制，作为消息信息的一个字节。提取消息检测到结束标志位‘4’时，停止消息提取。然后使用native2unicode将数字字节转换为unicode编码，完成消息的显示。
下边是消息提取函数：

```
function [msg_origin] = LSB_extract(image, lsb, color)
% LSB_extract(name, lsb)  LSB in steganography (extract)
% name: the picture's path and name
% lsb: lsb-rightmost LSBs
% color: 1-red, 2-green, 3-blue

layer = image(:, :, color);
tmp = blanks(0);
n = prod(size(layer));
% if lsb ~= 2, then you need to change something below
for i = 1 : n * lsb / 8
    tmp((i - 1) * 4 + 1 : i * 4) = mod(layer((i - 1) * 4 + 1 : i * 4), 2^lsb);
    msg((i - 1) * 8 + 1 : i * 8) = dec2bin(tmp((i - 1) * 4 + 1 : i * 4), lsb)';
    msg_origin(i) = bin2dec(msg((i - 1) * 8 + 1 : i * 8));
    if msg_origin(i) == 4  % EOT is the end tag
        break;
    end
end

msg_origin = native2unicode(msg_origin,'UTF-8');
msg_origin = msg_origin(1:end-1);
end
```

3、隐写测试 调用上边的消息嵌入和提取函数，编写程序进行一个小小的隐写术测试。 测试时选择隐藏两位，选取B色道。 测试程序如下：

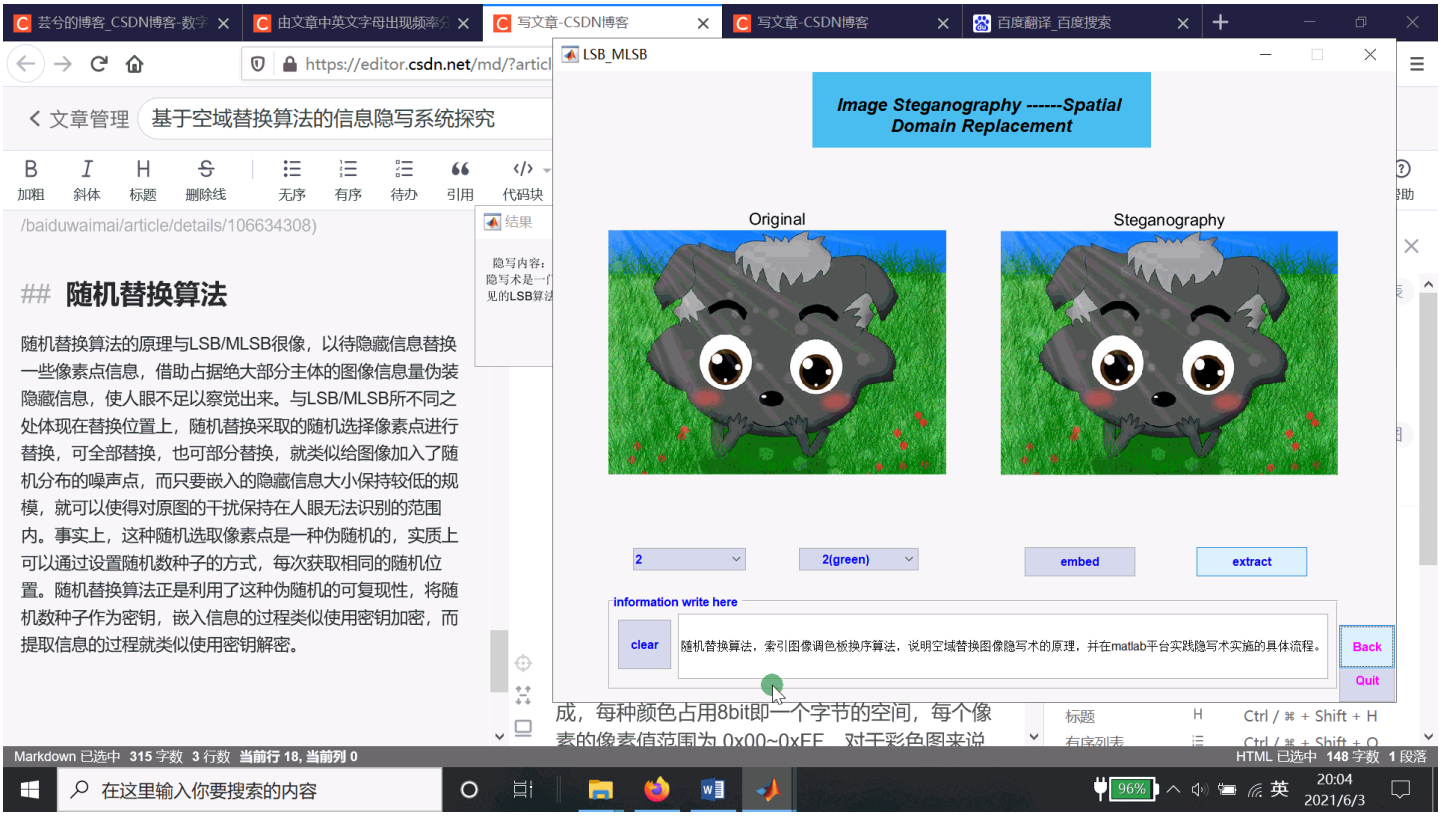
```
%LSB 隐写术
%
clear
close all
clc
message=input('请输入隐写内容: message=');
image=imread('wolf1.png');
subplot(121)
imshow(image)
title('Original')
image_result=LSBembed(image,message,2,3);
subplot(122)
imshow(image_result)
title('隐写图像')
msg_origin=LSB_extract(image_result,2,3);
msgbox({'  隐写内容:     ',msg_origin,'       '},'结果');
```

message输入：‘我要吃羊，我要吃羊懒羊羊，懒羊羊~~~~’
结果截图：



二、随机替换算法

随机替换算法的原理与LSB/MLSB很像，以待隐藏信息替换一些像素点信息，借助占据绝大部分主体的图像信息量伪装隐藏信息，使人眼不足以察觉出来。与LSB/MLSB所不同之处体现在替换位置上，随机替换采取的随机选择像素点进行替换，可全部替换，也可部分替换，就类似给图像加入了随机分布的噪声点，而只要嵌入的隐藏信息大小保持较低的规模，就可以使得对原图的干扰保持在人眼无法识别的范围内。事实上，这种随机选取像素点是一种伪随机的，实质上可以通过设置随机数种子的方式，每次获取相同的随机位置。随机替换算法正是利用了这种伪随机的可复现性，将随机数种子作为密钥，嵌入信息的过程类似使用密钥加密，而提取信息的过程就类似使用密钥解密。



三、 调色板换序算法

索引图像像素点存放的是调色板中的下标，具有颜色不容易失真，存储文件小的优势。基于调色板换序的算法，是指不改变调色板中颜色的顺序，只是改变调色板的排列，用调色板的排序方式对信息进行编码。因为有N!种不同的方式对调色板进行排序，所以可以选择一个，用来对一个短信息进行编码。基于调色板换序的隐写算法，没有改变颜色信息，只是将调色板的顺序及对应像素的索引进行了同步换序，因此所得到的隐写图像与原图像从视觉上看是一模一样的。一种比较简单的应用就是用调色板的顺序来作为信息加密的密文传输信息。

假设有密码本：

1~36对应

abcdefghijklmnopqrstuvwxyz0123456789

要传递一个短消息“fireat5”（5点开火）

密文即：6 9 18 5 1 20 32

假设原调色板顺序为1~n，

设置结束标志位flag（flag选任意1~n之间不与密文重复的数）

根据我们的密文将调色板顺序换为：6 9 18 5 1 20 32 flag……

那么跟据隐写图像的调色板与原调色板对比，即可确定上述密文，然后解密。下边给的程序是GUI代码中的一部分，在这里对程序流程做一个简单的说明：

其中index作为要传输的密文（6 9 18 5 1 20 32），然后将密文转为调色板中的顺序，其余的没有用的序号，依次放在后边，这就得到了新的调色板的顺序，记作lin（lin中前几个索引原来所在位置为我们的密文6 9 18 5 1 20 32），由这个lin，将原图像的像素点索引进行同步的换序，方法是使用find函数返回索引位置，将其替换为lin指示的索引。提取消息的时候，就根据调色板顺序比对，得到lin，然后解密。

```
global index  map I new_map new_I lin;
new_I=I;
ind=(1:64);
new_map=zeros(1,64);
tmp=zeros(64,3);
for i=1:length(index)
    key=index(i);
    tmp(i,:)=map(key,:);
end
dif=setdiff(ind,index);
j=1;
for i= (length(index)+1:64)
    k=dif(j);
    j=j+1;
    tmp(i,:)=map(k,:);
end
lin=[index,dif];
new_map=tmp;
for i=1:64
    pos=find(I==i);
    pos2=find(lin==i);
    if sum(pos)~=0
        new_I(pos)=lin(pos2);
    end
end
axes(handles.axes2);
imshow(new_I,map)
title('Steganography')
```

调色板换序的程序演示GIF太大放不了，就算啦！

下面给出一个综合的性能分析表：

表 6.1 算法性能分析

算法	容量	隐蔽性	鲁棒性
LSB/MLSB	n/8（n 位平面）	优	差
伪随机替换	相对 LSB/MLSB 低	良	差
调色板换序	极小（调色板颜色数）	最优	差