基本

二值图像隐藏

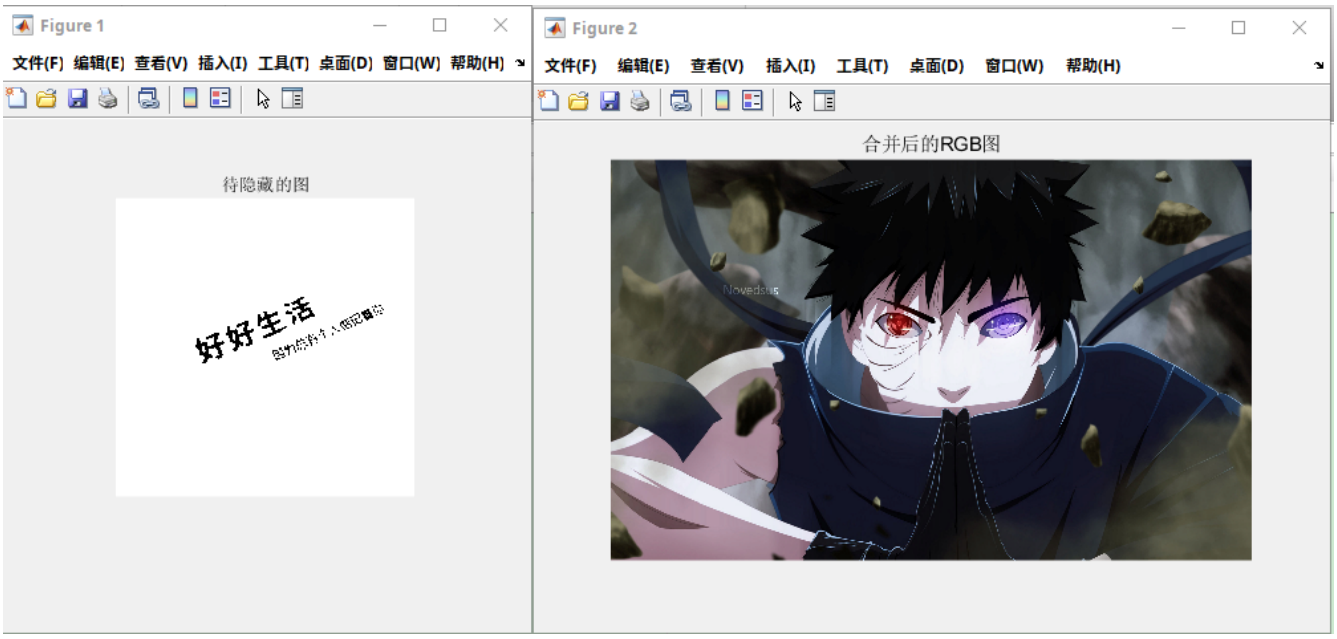
关键词：存储容量小，操作简单，

代码

```
% merge.m
% 读取原图
Img = imread('D:\Matlab\document\temp1\1.png');
% 读取待隐藏的图
Imgmark = imread('D:\Matlab\document\temp1\4.png');
[M, N, Z ] = size(Img); %求某一矩阵所有维度的最大长度，M是图片的高，N是图片的宽
[A, B, C] = size(Imgmark);%同上
Img = double(Img);
ImgR2 = Img(:,:,1);
ImgG2 = Img(:,:,2);
ImgB2 = Img(:,:,3);

% 转为灰度图
Imgmark = rgb2gray(Imgmark);
% 转为二值图
Imgmark = im2bw(Imgmark);
figure;imshow(Imgmark,[]);title('待隐藏的图')


for i = 1 : M
for j = 1 : N
        % 清空图A第零位平面
        if mod(ImgR2(i,j), 2) == 1
            ImgR2(i,j) = ImgR2(i,j) - 1;
        end
        if mod(ImgG2(i,j), 2) == 1
            ImgG2(i,j) = ImgG2(i,j) - 1;
        end
        if mod(ImgB2(i,j), 2) == 1
            ImgB2(i,j) = ImgB2(i,j) - 1;
        end
        if i>A||j>B
           Imgmark(i,j)= 0;
        end
        % 加上图B的像素值
        ImgR2(i,j) = ImgR2(i,j) + Imgmark(i, j);
        ImgG2(i,j) = ImgG2(i,j) + Imgmark(i, j);
        ImgB2(i,j) = ImgB2(i,j) + Imgmark(i, j);
    end
end

% 三通道合并
ImgNew = zeros(M, N, Z);
```

```
ImgNew(:,:,1) = ImgR2;
ImgNew(:,:,2) = ImgG2;
ImgNew(:,:,3) = ImgB2;


figure;imshow(uint8(ImgNew),[]);title('合并后的RGB图');
imwrite(uint8(ImgNew), 'D:\Matlab\document\temp1\3.png'); % 保存图片
```

实验效果:



上图为原图，下图是处理后的待隐藏图片和合并后的RGB图像

提取出来的图像

提取图像的代码：

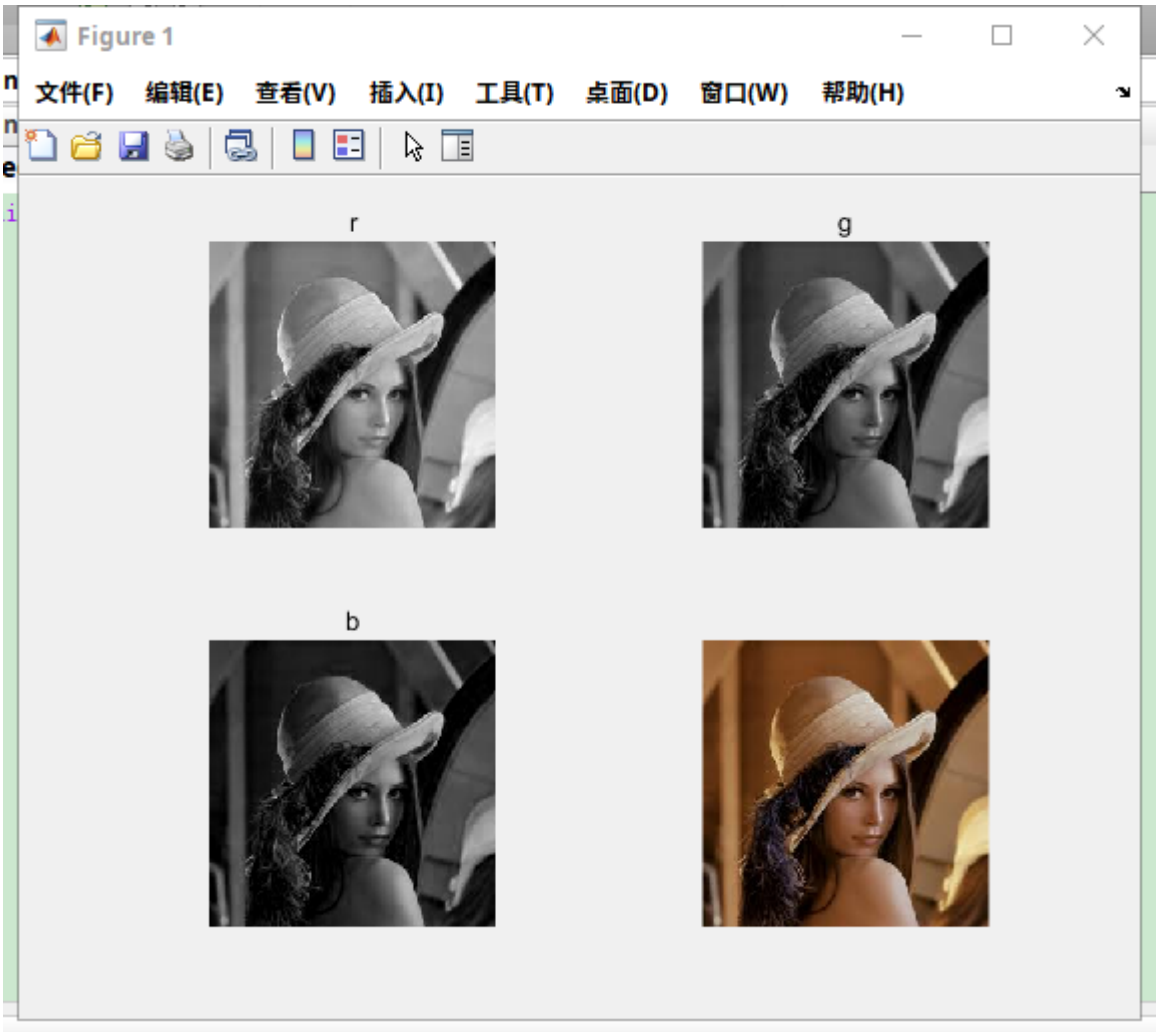
```
% extract.m
% 读取合并后的RGB图
Img = imread('D:\Matlab\document\temp1\3.png');
[M, N, Z] = size(Img);
Img = double(Img);
ImgR2 = Img(:,:,1);
ImgG2 = Img(:,:,2);
ImgB2 = Img(:,:,3);

Imgmarkextract = zeros(M, N, Z);
ImgmarkextractR = Imgmarkextract(:,:,1);
ImgmarkextractG = Imgmarkextract(:,:,2);
ImgmarkextractB = Imgmarkextract(:,:,3);
for i = 1 : M
    for j = 1 : N
        % 读取第零位平面
        if mod(ImgR2(i,j), 2) == 1
            ImgmarkextractR(i,j) = 1;
        end
        if mod(ImgG2(i,j), 2) == 1
            ImgmarkextractG(i,j) = 1;
```

```
        end
        if mod(ImgB2(i,j), 2) == 1
            ImgmarkextractB(i,j) = 1;
        end
    end
end
Imgmarkextract(:,:,1) = ImgmarkextractR;
Imgmarkextract(:,:,2) = ImgmarkextractG;
Imgmarkextract(:,:,3) = ImgmarkextractB;
figure;imshow(Imgmarkextract,[]);title('提取出的隐藏图');
imwrite(Imgmarkextract, 'D:\Matlab\document\2\4.png'); % 保存图片
```

tips

为什么代码中读取第零位面是这种形式

答:



一张正常的彩色图片是由RGB三个维度的图像合成的。上面的前三张图片就是将正常的图片拆成三个单一像素值的图片。从另一个角度来讲，一张RGB图片可以看作一个三维矩阵，而图片r，g，b分别是正常图片的三个维度的矩阵。

正如代码中:

```
ImgR2 = Img(:,:,1);%提取第一个维度的矩阵，Img是一个三维矩阵，ImgR2是一个二维矩阵
ImgG2 = Img(:,:,2);%同上
ImgB2 = Img(:,:,3);%同上
```

mod函数是取余函数，这两个处理过程中，只用到最低有效位。

这样来理解，目标是将最低有效位置零，0-255 即为 00000000-11111111，最低位的变化对于图像的影响最小，（人眼几乎完全无法分辨）对像素值进行取余：如果最低有效位是1，例如000000001（1），那么用2取余的结果也是1。所以将这个最低有效位置零。若为10111110（190），那么用 2 取余结果为0，不需要更多操作。

代码：

```
for i = 1 : M
for j = 1 : N
        % 清空图A第零位平面
        if mod(ImgR2(i,j), 2) == 1
            ImgR2(i,j) = ImgR2(i,j) - 1;
        end
        if mod(ImgG2(i,j), 2) == 1
            ImgG2(i,j) = ImgG2(i,j) - 1;
        end
        if mod(ImgB2(i,j), 2) == 1
            ImgB2(i,j) = ImgB2(i,j) - 1;
        end
        if i>A||j>B %如果待隐藏图像的尺寸更小，
           Imgmark(i,j)= 0;% 超过待隐藏图像的尺寸的像素点的值置零。
        end
        % 加上图B的像素值
        ImgR2(i,j) = ImgR2(i,j) + Imgmark(i, j);
        ImgG2(i,j) = ImgG2(i,j) + Imgmark(i, j);
        ImgB2(i,j) = ImgB2(i,j) + Imgmark(i, j);
    end
end
```

灰度图像隐藏

关键词：大容量，多通道操作

实现代码：

```
% main_merge2.m
% 各通道肉眼可接受位差
yr = 4;
yg = 5;
yb = 3;

% 读取原图
Img = imread('C:\Users\admin\Desktop\test\2.png');
figure;imshow(Img,[]);title('原图');

% 读取待隐藏的图
Imgmark = imread('C:\Users\admin\Desktop\test\3.png');

% 转为灰度图
Imgmark = rgb2gray(Imgmark);
figure;imshow(Imgmark,[]);title('待隐藏的图');
Imgmark = double(Imgmark);
```

```
[markm, markn] = size(Imgmark);
% 将灰度图的二维数组转成一列
Imgmarkline = Imgmark(:);

% 这一列再转化为更长的一列，二进制八位表示
Imgmarklinebin = zeros(markm*markn*8,1);
for ii = 1 : markm*markn
    [Imgmarklinebin(8*ii-7), Imgmarklinebin(8*ii-6), Imgmarklinebin(8*ii-5),
Imgmarklinebin(8*ii-4), Imgmarklinebin(8*ii-3),...
        Imgmarklinebin(8*ii-2), Imgmarklinebin(8*ii-1), Imgmarklinebin(8*ii)] =
Find8bits(Imgmarkline(ii));
end

%%
% 获得RGB各通道分量图
Img = double(Img);
ImgR = Img(:,:,1);
ImgG = Img(:,:,2);
ImgB = Img(:,:,3);
% 嵌入
% 对于红色通道
embedNumsed = 0; % 已嵌入个数
[M, N, Z] = size(Img);
y = zeros(8, 1);
flag = 0; % 辅助跳出的标志

ImgRline = ImgR(:); % 转换为一列
ImgRlineNew = ImgRline; % 嵌入后
for ii = 1 : M*N
    if flag == 1 % 跳出外层循环
       break;
    end

    [y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)] = Find8bits(ImgRline(ii));
    posNzreo = FindNotZero(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));
    embedNums = posNzreo - yr; % 能嵌入的个数
    if  embedNums > 0 %符合嵌入条件
        for jj = 1 : embedNums
            embedNumsed = embedNumsed + 1; % 已嵌入个数
            if embedNumsed > markm*markn*8 % 嵌入完成
               flag = 1; % 设置标识，使外层循环也跳出
               break;
            end

            y(jj) = Imgmarklinebin(embedNumsed);% 嵌入
        end
    end
    ImgRlineNew(ii) = bin2dec_trans(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));% 嵌入后
的
end
ImgR2 = reshape(ImgRlineNew, [M, N]);
```

```
% 对于G通道
ImgGline = ImgG(:); % 转换为一列
ImgGlineNew = ImgGline; % 嵌入后
for ii = 1 : M*N
    if flag == 1; % 跳出外层循环
       break;
    end

    [y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)] = Find8bits(ImgGline(ii));
    posNzreo = FindNotZero(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));
    embedNums = posNzreo-yg; % 能嵌入的个数
    if  embedNums > 0 % 符合嵌入条件
        for jj = 1 : embedNums
            embedNumsed = embedNumsed + 1; % 已嵌入个数
            if embedNumsed > markm*markn*8 % 嵌入完成
               flag = 1; % 设置标识，使外层循环也跳出
               break;
            end

            y(jj) = Imgmarklinebin(embedNumsed); % 嵌入
        end
    end
    ImgGlineNew(ii) = bin2dec_trans(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)); % 嵌入
后的
end
ImgG2 = reshape(ImgGlineNew, [M, N]);

% 对于B通道
ImgBline = ImgB(:); % 转换为一列
ImgBlineNew = ImgBline; % 嵌入后
for ii = 1 : M*N
    if flag == 1; % 跳出外层循环
       break;
    end

    [y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)] = Find8bits(ImgBline(ii));
    posNzreo = FindNotZero(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));
    embedNums = posNzreo - yb; % 能嵌入的个数
    if  embedNums > 0 % 符合嵌入条件
        for jj = 1 : embedNums
            embedNumsed = embedNumsed + 1; % 已嵌入个数
            if embedNumsed > markm*markn*8 % 嵌入完成
               flag = 1; % 设置标识，使外层循环也跳出
               break;
            end

            y(jj) = Imgmarklinebin(embedNumsed); % 嵌入
        end
    end
    ImgBlineNew(ii) = bin2dec_trans(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)); % 嵌入
后的
end
ImgB2 = reshape(ImgBlineNew, [M, N]);
```

```
ImgNew = zeros(M, N, Z);
ImgNew(:,:,1) = ImgR2;
ImgNew(:,:,2) = ImgG2;
ImgNew(:,:,3) = ImgB2;

figure;imshow(uint8(ImgNew),[]);title('合并后的图片');
imwrite(uint8(ImgNew), 'C:\Users\admin\Desktop\test\4.png'); % 保存图片
```

代码中用到的工具函数：

```
% bin2dec_trans.m
% 二进制转十进制
function Data=bin2dec_trans(y7,y6,y5,y4,y3,y2,y1,y0)
   Data=y7*128+y6*64+y5*32+y4*16+y3*8+y2*4+y1*2+y0;
end

% Find8bits.m，转换成8个二进制数
function [y7,y6,y5,y4,y3,y2,y1,y0]=Find8bits(Data)
y0=mod(Data,2);
y7=fix(Data/128);Data=Data-y7*128;
y6=fix(Data/64); Data=Data-y6*64;
y5=fix(Data/32); Data=Data-y5*32;
y4=fix(Data/16); Data=Data-y4*16;
y3=fix(Data/8);  Data=Data-y3*8;
y2=fix(Data/4);  Data=Data-y2*4;
y1=fix(Data/2);  Data=Data-y1*2;
end

% FindNotZero.m
%找出第一个不为零的数位 从最高位(第八位)开始
function posNzreo=FindNotZero(y7,y6,y5,y4,y3,y2,y1,y0)
    if y7~=0      posNzreo=8;
    elseif y6~=0  posNzreo=7;
    elseif y5~=0  posNzreo=6;
    elseif y4~=0  posNzreo=5;
    elseif y3~=0  posNzreo=4;
    elseif y2~=0  posNzreo=3;
    elseif y1~=0  posNzreo=2;
    else          posNzreo=1;
    end
end
```

代码太多，详细意义太累了不想写。到时候讲一下思路就行。

提取图像的代码：

```
% main_extract2.m
% 各通道肉眼可接受位差
yr = 4;
yg = 5;
```

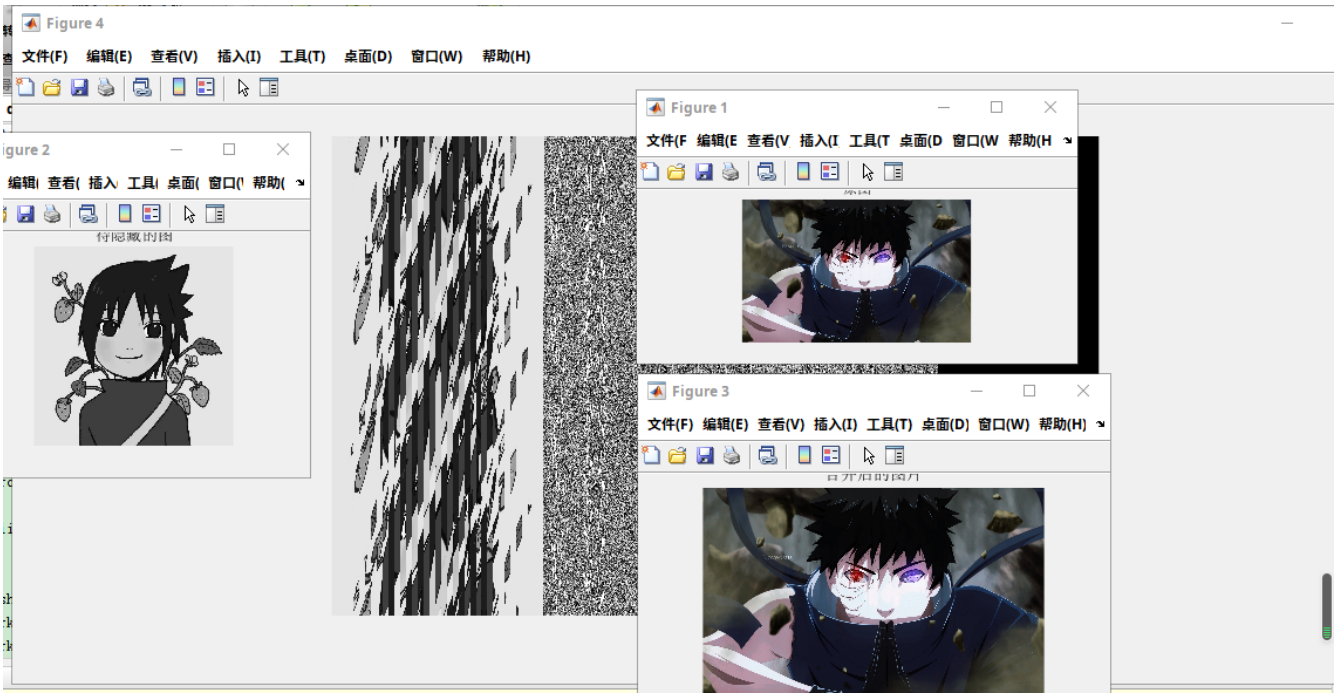
```
yb = 3;

% 读取合并后的RGB图
Img = imread('C:\Users\admin\Desktop\test\4.png');
[M, N, Z] = size(Img);
Img = double(Img);
ImgR2 = Img(:,:,1);
ImgG2 = Img(:,:,2);
ImgB2 = Img(:,:,3);

% 提取嵌入图像
flag = 0;
Imgmark_extractlinebin = zeros(M*N*8, 1);
extractNumsed = 0; % 已提取个数

% R通道
ImgRline2 = ImgR2(:); % 转换为一列
for ii = 1 : M*N
    if flag == 1; % 跳出外层循环
       break;
    end

    [y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)] = Find8bits(ImgRline2(ii));
    posNzreo = FindNotZero(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));
    embedNums = posNzreo - yr; % 已嵌入的个数
    if  embedNums > 0 % 符合嵌入条件
        for jj = 1 : embedNums

            extractNumsed = extractNumsed + 1; % 已提取个数
            if extractNumsed > M*N*8 % 提取完成
               flag = 1; % 设置标识，使外层循环也跳出
               break;
            end

           Imgmark_extractlinebin(extractNumsed) = y(jj); % 提取
        end
    end
end

% G通道
ImgGline2 = ImgG2(:); % 转换为一列
for ii = 1 : M*N
    if flag == 1; % 跳出外层循环
       break;
    end

    [y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)] = Find8bits(ImgGline2(ii));
    posNzreo = FindNotZero(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));
    embedNums = posNzreo - yg; % 已嵌入的个数
    if embedNums > 0 % 符合嵌入条件
        for jj = 1:embedNums

            extractNumsed = extractNumsed + 1; % 已提取个数
```

```
            if extractNumsed > M*N*8 % 提取完成
               flag = 1; % 设置标识，使外层循环也跳出
               break;
            end

           Imgmark_extractlinebin(extractNumsed) = y(jj);% 提取
        end
    end
end

%  G通道
ImgBline2 = ImgB2(:); % 转换为一列
for ii = 1:M*N
    if flag == 1; % 跳出外层循环
       break;
    end

    [y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1)] = Find8bits(ImgBline2(ii));
    posNzreo = FindNotZero(y(8), y(7), y(6), y(5), y(4), y(3), y(2), y(1));
    embedNums = posNzreo - yb; % 已嵌入的个数
    if embedNums > 0 % 符合嵌入条件
        for jj = 1 : embedNums

            extractNumsed = extractNumsed + 1; % 已提取个数
            if extractNumsed > M*N*8 % 提取完成
               flag = 1; % 设置标识，使外层循环也跳出
               break;
            end

           Imgmark_extractlinebin(extractNumsed) = y(jj); % 提取
        end
    end
end

% 二进制转十进制
Imgmarklinedec = zeros(M*N, 1); % 转化为十进制
for ii = 1 : M*N
    Imgmarklinedec(ii) = bin2dec_trans(Imgmark_extractlinebin(8*ii-7),
Imgmark_extractlinebin(8*ii-6), Imgmark_extractlinebin(8*ii-5),
Imgmark_extractlinebin(8*ii-4),...
                                    Imgmark_extractlinebin(8*ii-3),
Imgmark_extractlinebin(8*ii-2), Imgmark_extractlinebin(8*ii-1),
Imgmark_extractlinebin(8*ii));
end
Imgmarkextract = reshape(Imgmarklinedec, [M, N]);
figure;imshow(Imgmarkextract,[]);title('提取出的隐藏图');
imwrite(uint8(Imgmarkextract), 'C:\Users\admin\Desktop\test\5.png'); % 保存图片
```

实际效果：

第一次失败：

第二次失败



根据第二次的失败情况（提取出来的图片中有一个变形的小佐助），推测是提取时两图片尺寸不一致的问题

调整之后就成功了：

修改代码为：

```
%读取被隐藏图片的尺寸，还原的时候要用
image = imread('C:\Users\admin\Desktop\test\3.png');
[A,B,C] = size(image);
%原先的代码：
% 二进制转十进制
Imgmarklinedec = zeros(M*N, 1); % 转化为十进制
for ii = 1 : M*N
    Imgmarklinedec(ii) = bin2dec_trans(Imgmark_extractlinebin(8*ii-7),
Imgmark_extractlinebin(8*ii-6), Imgmark_extractlinebin(8*ii-5),
Imgmark_extractlinebin(8*ii-4),...
                                    Imgmark_extractlinebin(8*ii-3),
Imgmark_extractlinebin(8*ii-2), Imgmark_extractlinebin(8*ii-1),
Imgmark_extractlinebin(8*ii));
end
Imgmarkextract = reshape(Imgmarklinedec, [M, N]);
figure;imshow(Imgmarkextract,[]);title('提取出的隐藏图');
imwrite(uint8(Imgmarkextract), 'C:\Users\admin\Desktop\test\5.png'); % 保存图片
%！！！！改为：
Imgmarklinedec = zeros(A*B, 1); % 转化为十进制！！！！！修改点
for ii = 1 : A*B%！！！！！修改点
    Imgmarklinedec(ii) = bin2dec_trans(Imgmark_extractlinebin(8*ii-7),
Imgmark_extractlinebin(8*ii-6), Imgmark_extractlinebin(8*ii-5),
Imgmark_extractlinebin(8*ii-4),...
                                    Imgmark_extractlinebin(8*ii-3),
Imgmark_extractlinebin(8*ii-2), Imgmark_extractlinebin(8*ii-1),
Imgmark_extractlinebin(8*ii));
end
Imgmarkextract = reshape(Imgmarklinedec, [A,B]);%！！！！！修改点
figure;imshow(Imgmarkextract,[]);title('提取出的隐藏图');
imwrite(uint8(Imgmarkextract), 'C:\Users\admin\Desktop\test\5.png'); % 保存图片
```

原代码未考虑到图片尺寸不一致的情况，复原时按照'原图'的尺寸进行复原，产生了拉伸。

将复原时尺寸修改为”待隐藏图像“的尺寸，复原时就不存在拉伸的情况。因此能够正常提取出图片

彩色图像隐藏

方案一：大容量隐藏算法是可以用来隐藏彩色图像的。

大容量隐藏算法的处理步骤是：将彩图转化成灰度图像，然后将灰度图像插入原图。通过查询资料得知：单一通道的图像可以当作灰度图来处理。 所以灰度图像隐藏算法也可以用来隐藏彩色图像。将彩色图像的三个单通道图像逐个插入，类比灰度图像。 但是要求是原图的尺寸要足够大，但是这个范围并不能很好的被估计 灰度图像隐藏时，待隐藏图像的尺寸小于等于原图的尺寸，有很大的概率成功。 但是将三个单通道图像全部隐藏进去，就不能确定原图的尺寸到底要有多大，比待隐藏图像大多少才可以。具有不确定性。

方案二：尾部附加法。将文件附在载体图片之后，利用BMP文件的特殊性质(系统在读取BMP文件的时候，是读取它的第3～6个字节为文件长度，对超出这个长度的部分会忽略)，将目标图片的二进制文件直接附着在载体之后，来实现文件的隐藏，此方法简单易行，不会破坏载体与目标图片的任何信息，且对隐藏文件的大小没有限制，但隐蔽性有待加强。 通过对BMP图像文件的数据结构的分析在BMP图像的头文件中有指示文件大小的数值bfSIZE。它指定了一般的图片浏览器所能读取的范围。若不改变该数值的大小，则一般的图片浏览器只能够读取原文件。即便在该文件的尾部接上其他的文件，图片浏览器所显示的仍然只是原文件。这就相当于把后续文件给屏蔽掉了，可以达到隐藏信息的目的。一旦图片被截获者截获，一般的图片浏览器对图片的读取也只能进行到载体部分，目标图片不会被暴露出来，达到隐藏的效果。 （有待实现）