图像信息伪装技术

图像信息伪装技术

- 图像中隐写另一幅图像
  - 4bit替换法
- 图像置乱
  - 变化模版形状
  - 幻方变换
  - Hash置乱
  - 隐藏置乱图像的优点

图像的冗余空间（低4bit）

图像的低4bit，我们通常认为可以做隐藏信息的空间，也就是载体的冗余空间。

因此，这里要介绍的方法都是对图像像素值的低4bit进行操作的。

讨论图像的冗余空间（低4bit）

下面是woman图像及其部分矩阵：

Array Editor: X

File Edit View Web Window Help

Numeric format: shortG Size: 256 by

	1	2	3	4	5	6
1	125	79	111	98	111	79
2	98	112	91	112	79	111
3	98	125	98	111	112	98
4	106	98	79	112	79	125
5	79	125	98	91	98	98
6	98	111	112	98	94	111
7	112	79	111	94	98	79
8	111	112	98	111	112	91
9	106	98	79	112	79	98
10	111	112	111	91	112	111

讨论图像的冗余空间（低4bit）

去掉低4bit后的woman图像及其部分矩阵：

Array Editor: X1

File Edit View Web Window Help

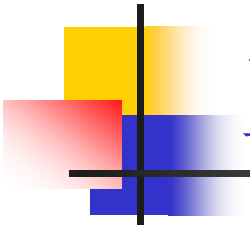
Numeric format: shortG Size: 256 by

	1	2	3	4	5	6
1	112	64	96	96	96	64
2	96	112	80	112	64	96
3	96	112	96	96	112	96
4	96	96	64	112	64	112
5	64	112	96	80	96	96
6	96	96	112	96	80	96
7	112	64	96	80	96	64
8	96	112	96	96	112	80
9	96	96	64	112	64	96

Array Editor: X Array Editor: X1

讨论图像的冗余空间（低4bit）

由上面的对比可以看到，虽然图像矩阵的每个像素值都去掉了低4bit（如第（1,1)像素点），但改变后的图像和原图像在给人的视觉上并没有太大的变化，因此可以认为低4bit是冗余空间，可以改变，这是可以进行图像嵌入伪装的理论基础。



直接4bit替换

所谓直接4bit替换法，就是直接用秘密图像像素值的高4bit去替换载体图像像素值的低4bit。

第一部分

秘密图像和载体图像的处理

提取图像信息并分层

```
%提取woman和lenna图像，
cover = imread(‘lenna.bmp');
data=cover;
msg=imread('woman.bmp' );
[row,col]=size(cover);

cover1=cover(:,:,1);
```

lenna原图像

woman原图像

分层后，lenna图像的R层就做为载体图像。

lenna红（R）层图像

处理载体图像和秘密图像的低4bit

%置载体图像R层的低4bit为0
cover1=bitand(cover1,240);

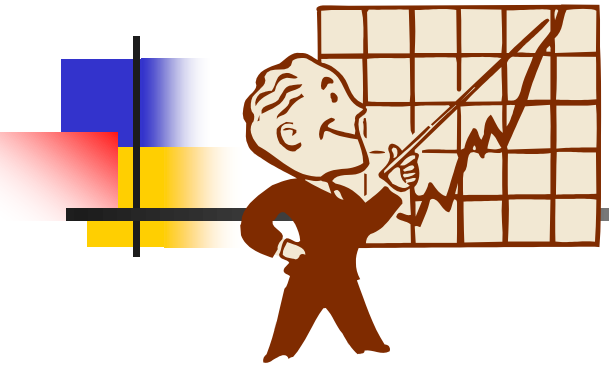
%置秘密图像的低4bit为0
takemsg4=bitand(msg,240);

%将秘密图像的高4bit右移4位
shiftmsg4=bitshift(takemsg4,-4);

去掉低4位的lenna的R层图像

去掉低4位的woman图像

woman图像高4bit右移4位



第二部分

图像隐藏

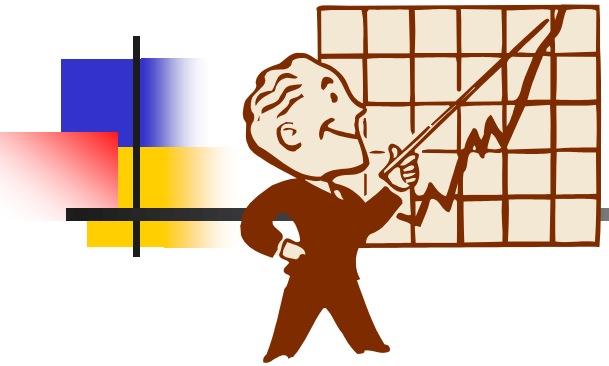
图像隐藏并保存

```
%图像隐藏
cover1=bitor(cover1,shiftmsg4);

%写回并保存
data(:,:,1)=cover1;
imwrite(data, 'mix.bmp");
```

woman隐藏在lenna的R层后的图像

最终隐藏后的图像



第三部分

效果检测

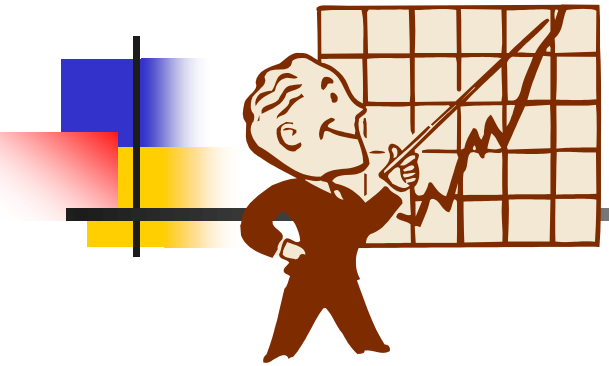
秘密图像提取

```
%提取秘密图像信息
data=imread('mix.bmp');
[row,col]=size(data);
A=data(:,:,1);

%提取隐蔽图像的低4bit
A=bitand(A,15);

%将提取的秘密图像的低4bit左
    移4位
A=bitshift(A,4);
```

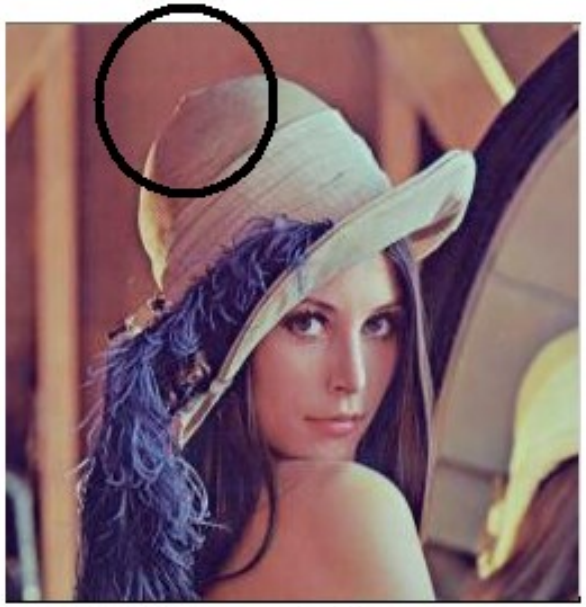
提取出的秘密图像



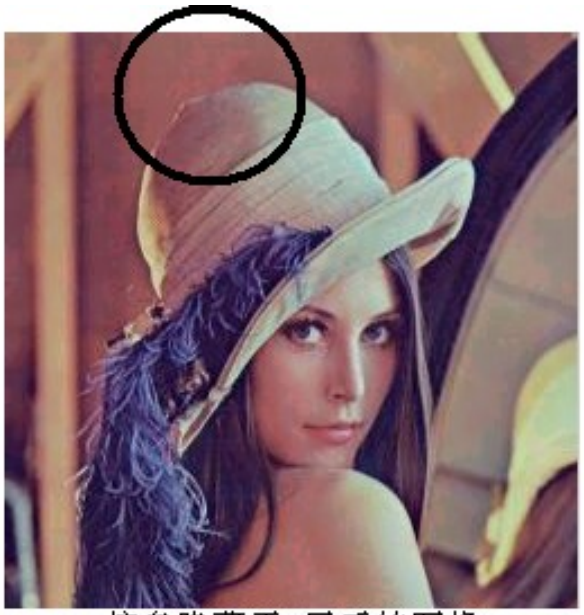
第四部分

讨论

将秘密图像分别藏于RGB载体图像不同层



lenna原图像



信息隐藏于R层后的图像

信息隐藏于G层后的图像

信息隐藏于B层后的图像

对隐藏效果的观察（不可见性）

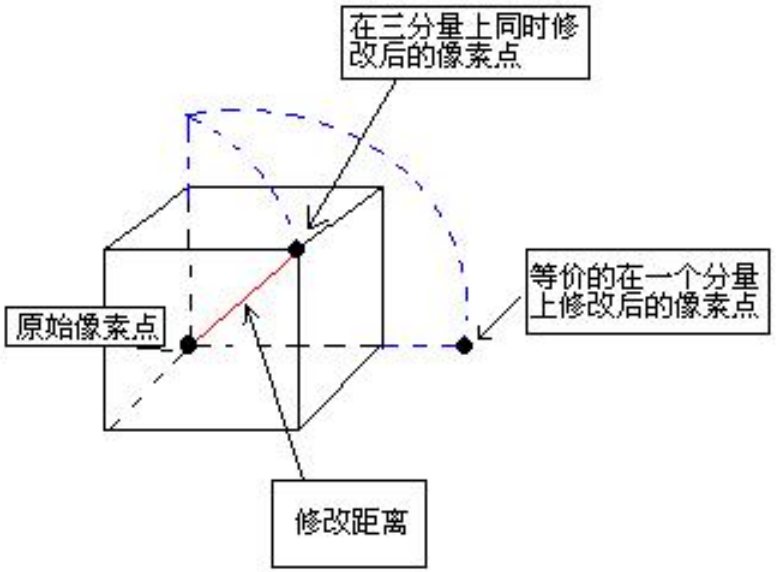
在上页图中我们看到，秘密图像隐藏于不同的层后与原载体图像的差别大小是不同的。对于隐藏在R、G、B各层的隐蔽载体，图中圈内的部分分别产生泛红、泛绿和泛蓝的现象。

对隐藏效果的分析

根据RGB图像像素的原理，一个像素的颜色由R、G、B三个分量值确定。考虑到第一章中我们阐述的RGB颜色模型，将秘密图像隐藏在一层中，容易导致该点的色彩向相应的坐标上发生绝对偏移，从而使得该像素点的色彩的相应分量突出。

隐藏方法的改进1

可将秘密图像像素值的高4bit分别藏于载体图像R、G、B层像素值的最低位或次低位，即可将秘密图像的高2bit藏于R层，另外两bit分别藏于G和B层，此时像素色彩的改变就不是沿一个坐标方向而改变，而是在整个RGB空间中发生偏移，改变后的颜色坐标点与改变前的颜色坐标点的距离（数学上的范数）比单纯在一个分量上改变的两点距离要小，这样对载体图像的影响会更小。在实际应用中，还应该考虑隐藏的鲁棒性等问题。



隐藏方法的改进2

3bit替换+4bit考察（4bit选择性替换）。

可对图像进行3bit替换。3bit替换对图像视觉效果造成的影响远小于4bit替换，不可见性程度大幅提高。但缺点是秘密图像嵌入的信息量大幅减少。解决办法，在3bit替换的基础上，选择性的进行4bit替换。

隐藏方法的改进2

所谓选择性替换就是对一部分像素进行4bit替换，一部分像素不进行替换。选择替换的依据是两幅图像局部区域的相似度。

相似度定义：如果两图像同一坐标下的像素中第4bit相同占一块图像（8x8的小块内）全部像素的比例。

$$\mu = \frac{s}{64}$$

其中，s为第4bit相同的像素数量，64为8x8块中的总像素数量。

隐藏方法的改进2

根据μ的值来确定该块各像素第4bit的隐藏策略：

1）如果μ大于某一阈值T，则可直接用秘密图像的第4bit替换载体图像的第4bit；

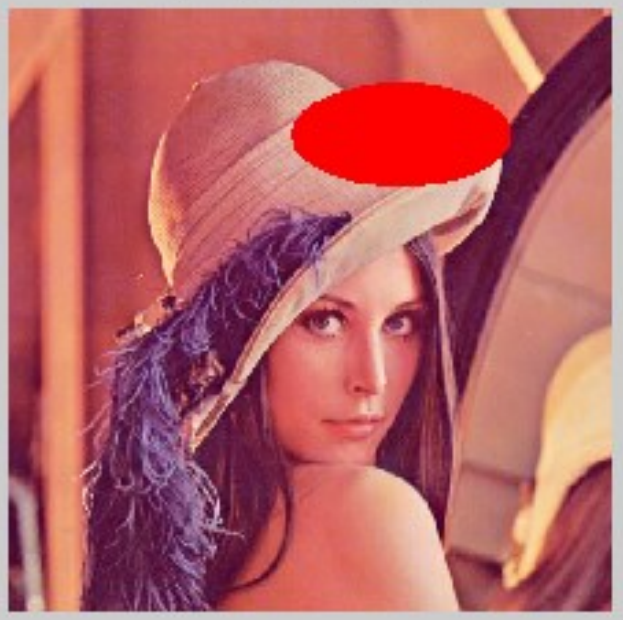
2）如果μ小于阈值1-T，则将秘密图像的第4bit取反后再替换；

3）如果μ介于1-T和T之间，则不进行替换。

当然，要用一个替换表对第4bit进行替换或取反替换了的块进行记录，并且将此表也嵌入到载体图像中。

对鲁棒性和安全性的考察

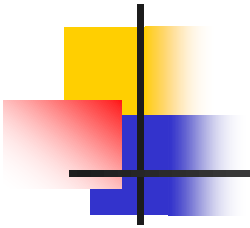
隐蔽载体图像



提取的秘密图像

鲁棒性弱、安全性差（无密钥机制）

图像置乱

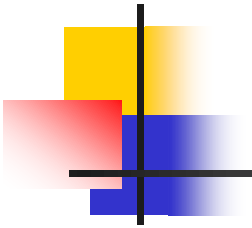


图像置乱的概念

所谓“置乱”，就是将图像的信息次序打乱，将a像素移动到b像素的位置上，b像素移动到c像素的位置上……使其变换成杂乱无章难以辨认的图像。

图像置乱的优点

置乱实际上就是图像的加密，与加密保证安全性不同的是，将置乱的图像作为秘密信息再进行隐藏，在提高系统安全性的同时可以很大限度的提高隐蔽载体的鲁棒性。所以图像置乱是信息隐藏中非常常用的一项技术。



变化模板形状的图像置乱

变化模板形状的图像置乱（算法）

变化模板形状的图像置乱算法的思想如下：

(1) 对原图像取一个固定模板，模板中像素位置排列（如图4.10所示）；

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

图4.10 原图像模板

变化模板形状的图像置乱（算法）

(2) 做一个与原图像模板不同的置乱模板（如图4.11），在置乱模板中把图像模板中的像素位置按一定次序填入（图4.11的模板中按从上到下，从左到右的次序依次填入）；

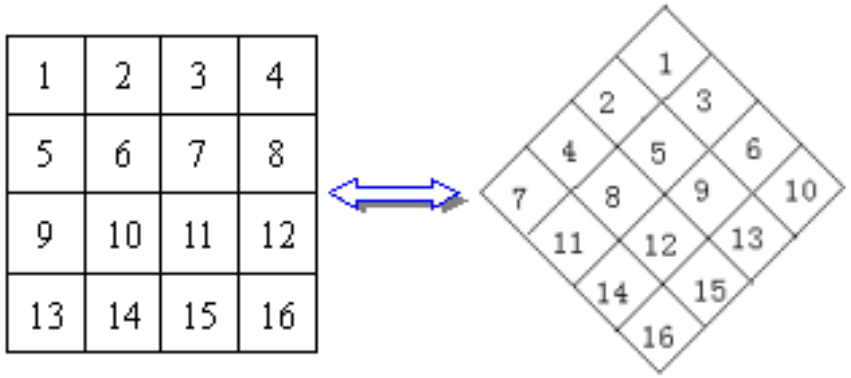


图4.10 原图像模板　　图4.11 置乱模板

变化模板形状的图像置乱（算法）

(3) 将置乱模板中的像素位置再按一定的次序填回到原图像模板中就得到了置乱后的图像模板（图4.12的模板是按从左到右，从上到下的次序依次读取置乱模板中像素位置）。

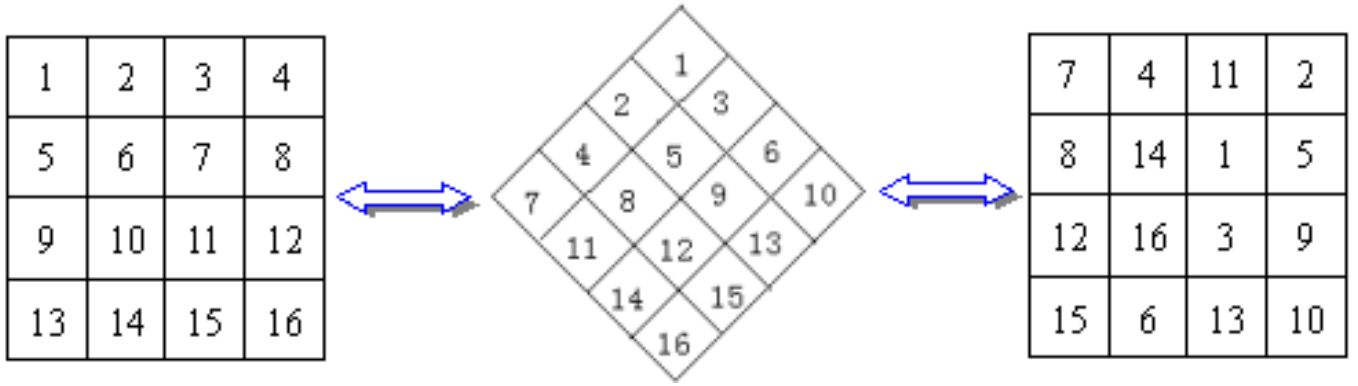
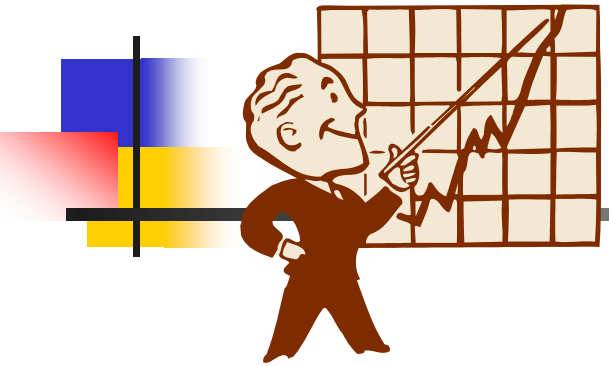


图4.10 原图像模板　　图4.11 置乱模板　　图4.12 置乱后模板



第一部分

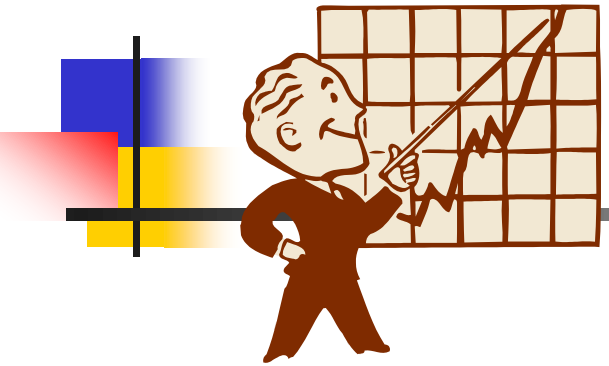
模板制作

模板要求

我们由于我们固定了置乱模板的大小，所以在对图像置乱前我们要对其进行边界修补。比如取置乱模板为32×32，则要求秘密图像的尺寸为32×32，64×64，128×128……。假设一幅图像的尺寸为32×31，则应该对其增加一列数据。

制作置乱变换表

由于我们实验中选取了固定的模板（菱形），我们可以采取查表的方法编写程序，32×32的菱形置乱变换表（行表和列表各一个）见教材。例如，置乱行列表（1,1）坐标下的数据分别为16和17，就表示原图像（16,17）坐标下的像素将被置乱到（1,1）坐标下。

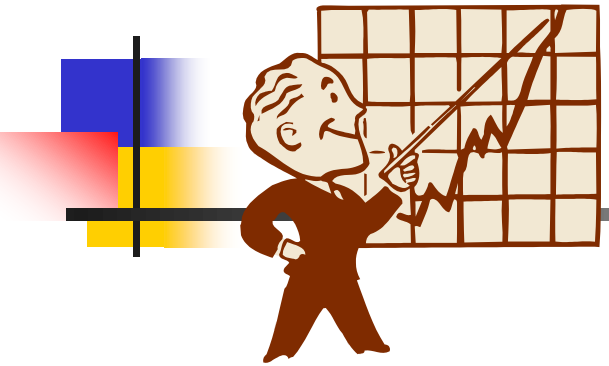


第二部分

置乱过程

置乱过程

- 首先分析原图像尺寸并补遗
- 在图像置乱机制中引入一个简单的密钥控制。将由密钥生成的第一个[128，255]的随机整数与置乱的结果进行异或（模2加）
- 调用查表函数进行查表置乱
- 还原置乱：重复一次置乱操作



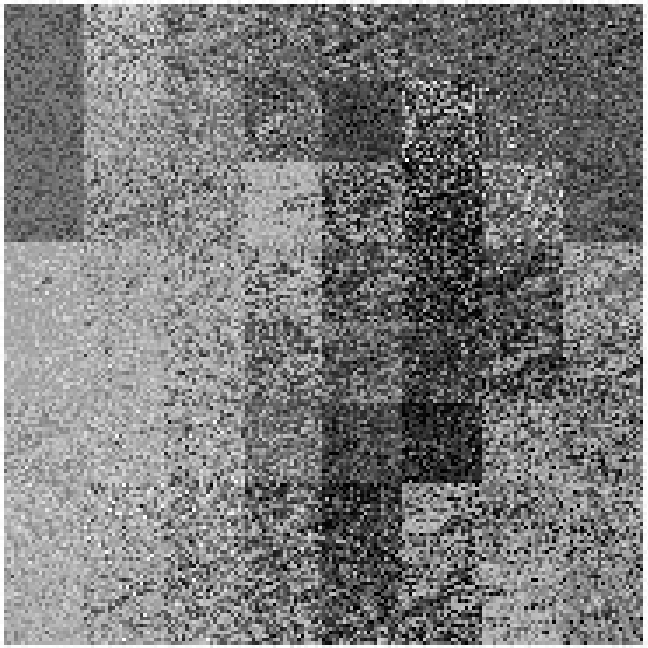
第三部分

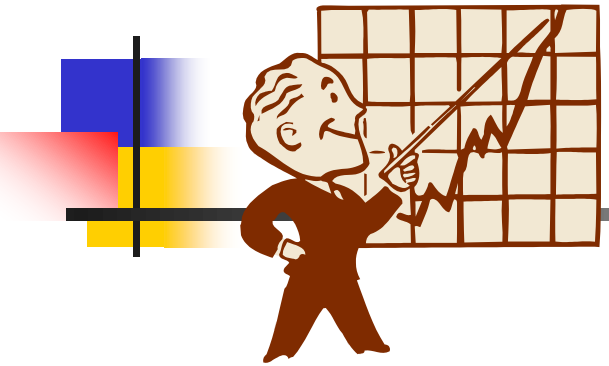
置乱效果

对woman进行置乱的效果

原始图像

置乱效果



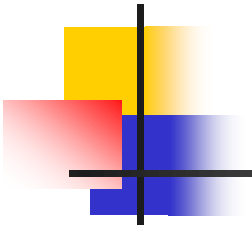


第四部分

分析结论

结论

可以选取不同形状的置乱模板，如与原图像模板宽度相同或不同的矩形，星形等。总之一句话，将图像的原始信息破坏得越大越好，不过，这种破坏一定是可以复原的。



图像的幻方置乱

图像的幻方变换

定义（幻方） n阶方阵M=$\{m_{ij}\}$，i,j=1,2,3…n为二维幻方，当且仅当

（1） $\forall K \in \{1,2...n\}$ 有 （c为仅与n有关的常数）

（2） $$\sum_{i=j} m_{ij} = \sum_{i+j-1=n} m_{ij} = c$$

特别的，当$m_{ij} \in \{1,2,3\ldots n^2\}$且两两不同时，称M为**n阶标准幻方**。此时，$c = \frac{n(n^2+1)}{2}$ 。

3阶幻方（《数术记遗》九宫图）

2	9	4
7	5	3
6	1	8

$\Sigma=15$

取阶为5,7,11下的标准幻方示意图（不唯一）

标准5阶幻方

20	3	11	24	7
21	9	17	5	13
2	15	23	6	19
8	16	4	12	25
14	22	10	18	1

Σ= 65

标准7阶幻方

26	36	4	21	31	48	9
34	44	12	22	39	7	17
42	3	20	30	47	8	25
43	11	28	38	6	16	33
2	19	29	46	14	24	41
10	27	37	5	15	32	49
18	35	45	13	23	40	1

Σ= 175

标准11阶幻方

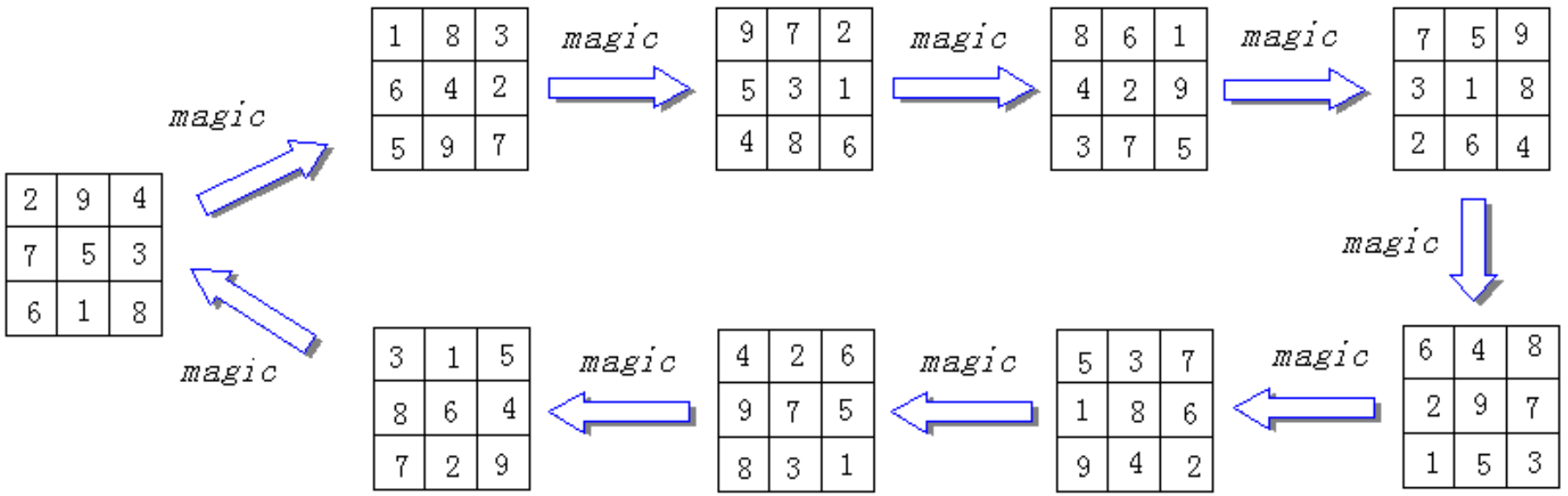
38	52	66	69	83	97	100	114	7	21	24
61	75	78	92	106	120	2	16	30	44	47
84	98	101	115	8	22	25	39	53	56	70
107	121	3	17	31	34	48	62	76	79	93
9	12	26	40	54	57	71	85	99	102	116
32	35	49	63	77	80	94	108	111	4	18
55	58	72	86	89	103	117	10	13	27	41
67	81	95	109	112	5	19	33	36	50	64
90	104	118	11	14	28	42	45	59	73	87
113	6	20	23	37	51	65	68	82	96	110
15	29	43	46	60	74	88	91	105	119	1

Σ= 671

幻方置乱算法

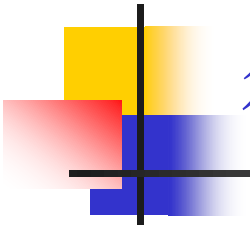
```
for r=1:n
for c=1:n
      M(r,c)=M(r,c)-1;
      if M(r,c)==0
            M(r,c)= n*n;
      end
end
end
```

3阶标准幻方9次变换复原示意图



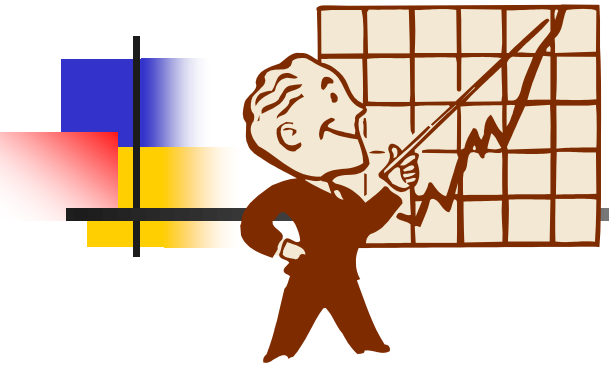
幻方变换复原

幻方置乱运算具有准对合性。假设记n阶图像块（对应于n阶标准幻方）I的幻方置乱为$I_1=magic(I)$，则相应的对I_1再进行幻方置乱得到$I_2=magic(I_1)=magic^2(I)$，依此类推，一般的有$I_{n^2}=magic^{n^2}(I)=I$，图像还原。



得到密钥

由此，我们可以得到密钥控制的方法。设算法加密钥为ekey，则解密钥为dkey分别表示输入数据执行幻方置乱的次数，仅要求ekey+dkey=n^2即可。

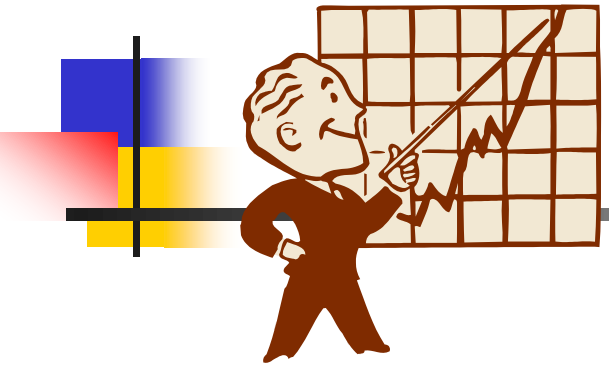


第一部分

模板制作

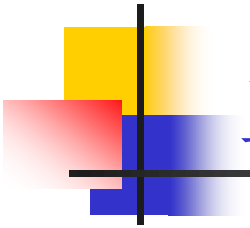
模板选取并制作变换表

我们这里选取11阶标准幻方作置乱模板，因为11阶幻方不唯一，所以这个置乱模板也不唯一。将11阶幻方改成行列查找表（见教材）。



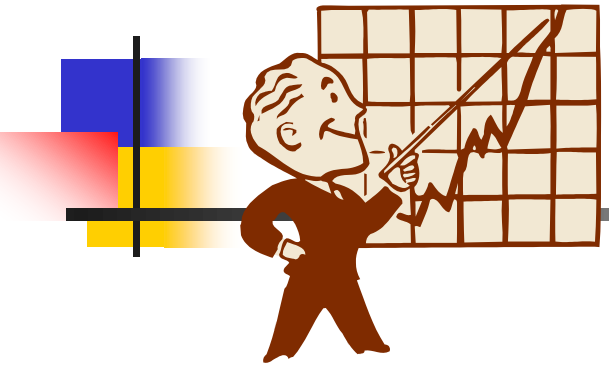
第二部分

置乱过程



置乱过程

- 分析原图像尺寸并补遗
- 密钥生成随机数
- 进行幻方变幻
- 调用查表函数进行查表置乱



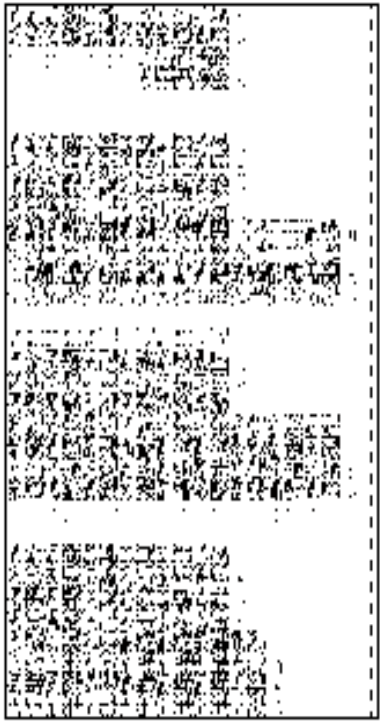
第三部分

置乱效果

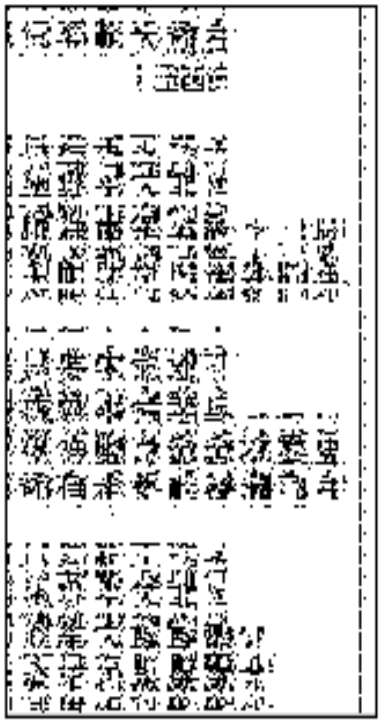
文本图像进行幻方置乱后的效果

1次幻方置乱的结果

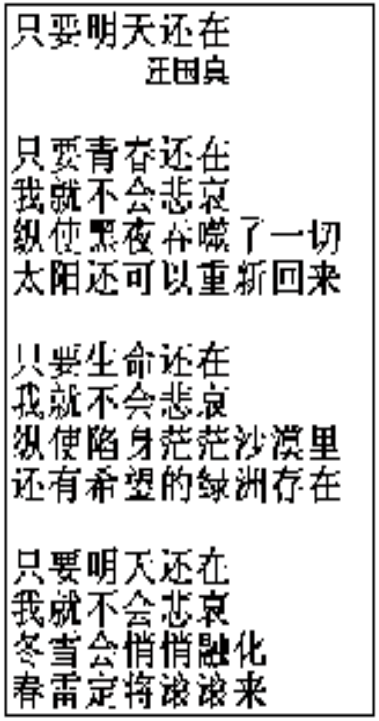
50次幻方置乱的结果

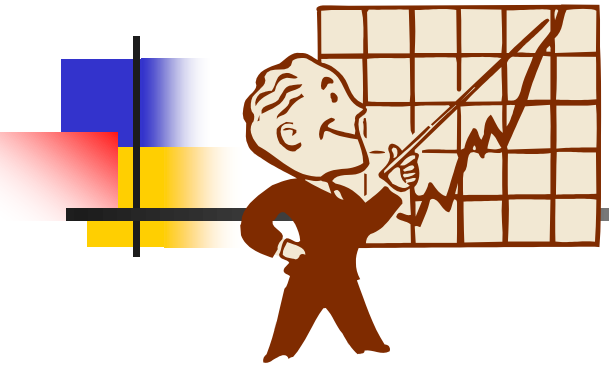


100次幻方置乱的结果



121次幻方置乱的结果



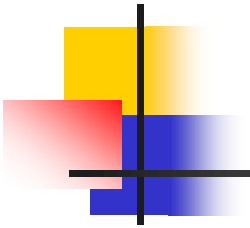


第四部分

分析结论

结论

这种方法对于文本图像的置乱效果还是比较好的，在置乱的过程中文字内容是完全不可读的。经过121次幻方置乱后文本图像就还原了，如果利用教材中的magicsquares.m函数自己编程推算出高阶标准幻方，相信效果会更好。另外，将低阶幻方作置乱模板置乱的结果再以图像块为单位进一步置乱，也能取得良好的置乱效果。



图像的Hash置乱

Hash置乱

对于m×n个像素点，我们要求使用随机置换函数置换m×n个，就完成了图像的Hash置乱。

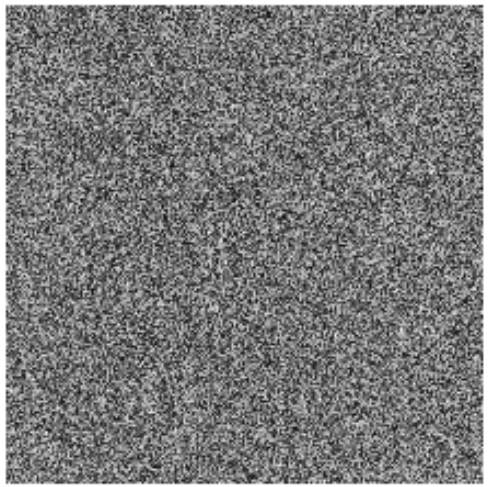
这个算法具有无冲突（collision）和强密钥控制的特点，是一个良好的图像置乱算法。

但这种算法不是对合的，所以在实现上较前两种复杂一些。另外，其算法执行起来也比较费时间。

woman图像hash置乱后的效果

原始图像

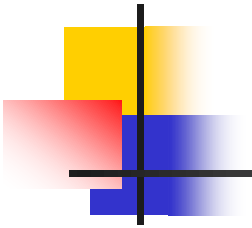
置乱后的结果



复原的结果

使用Hash进行置乱的特点

- 这个算法具有无冲突（collision）和强密钥控制的优点。
- 可以进行全幅度的置乱，因此，原始图像信息荡然无存，效果很好。
- 缺点：算法较复杂，速度慢



使用置乱后图像隐藏的优点

使用置乱方法可以增加图像伪装的鲁棒性

首先，将图像置乱后，将得到一幅杂乱无章的图像，这个图像无色彩、无纹理、无形状，从中无法读取任何信息，那么，将这样一幅图嵌入到另一幅普通图像时就不易引起那幅图色彩、纹理、形状的太大改变，甚至不会发生改变，这样人眼就不易识别，从而逃出了第三方的视线。

使用置乱方法可以增加图像伪装的鲁棒性

其次，由于秘密图像是置乱后的图像，根据上述的图像的“三无”特征，第三方根本不可能对其进行色彩、纹理、形状等的统计分析，即便他们截取到了秘密图像，也是无能为力。

使用置乱方法可以增加图像伪装的鲁棒性

而且，如果第三者企图对秘密图像进行反置乱，这也是不可能的，由于图像置乱有很多种方法，每种方法又可以使用不同的置乱模板算法，设置不同的参数，使用者有很大的自由度，他可以根据自己的想法得到不同的结果，相反，这给企图截获秘密信息的第三方带来了很大的困难，使他们需要耗费巨大的计算量来穷举测试各种可能性。

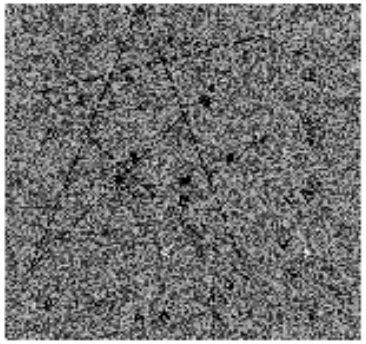
使用置乱方法可以增加图像伪装的鲁棒性

最后，可以抵抗第三方的恶意攻击。这是因为对秘密图像进行反置换的过程，就使得第三方在图像上所涂、画的信息分散到画面的各个地方，形成了点状的随机噪声，对视觉影响的程度不大。当然，为了使提取的信息更为清晰，最好对破坏严重的图像进行边界保持的中值滤波等方面的处理，以去除随机噪声。

置乱图像隐藏的抗恶意攻击性能

对隐蔽载体的恶意攻击1

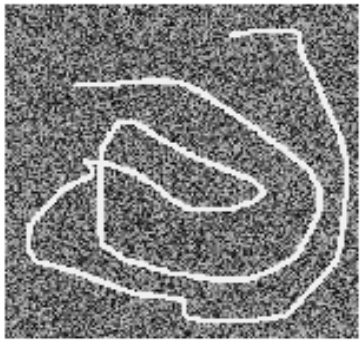
提取的置乱后的图像



还原的效果

对隐蔽载体的恶意攻击2

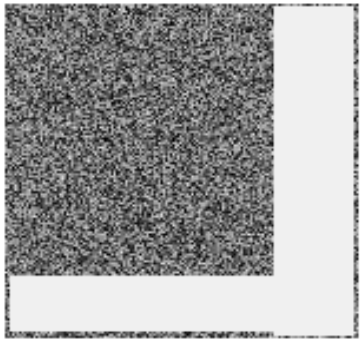
提取的置乱后的图像



还原的效果

对隐蔽载体的恶意攻击3

提取的置乱后的图像



还原的效果