

W-SVD 数字水印

W-SVD水印算法简介

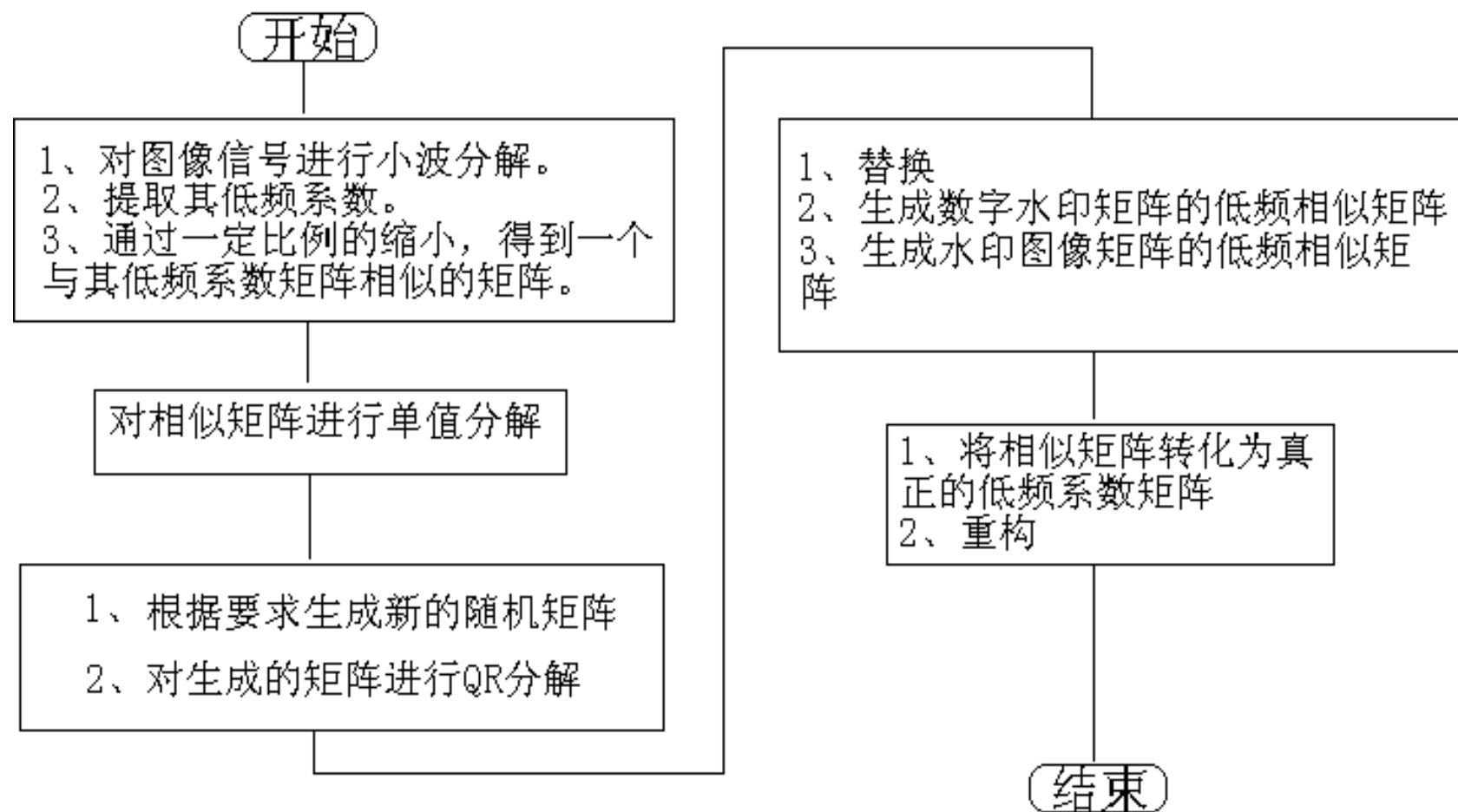
W-SVD数字水印算法是美国Syracuse大学数学系和美国空军实验室通信遥感部联合于1998年发布的。该算法属于小波变换域数字水印算法，具有良好的水印不可见性和鲁棒性等特点。尽管这一算法也并不是十分完美的，而且也不具备现代数字水印自适应和盲检测的要求，但无论如何，W-SVD的经典性是无庸置疑的。

第一部分



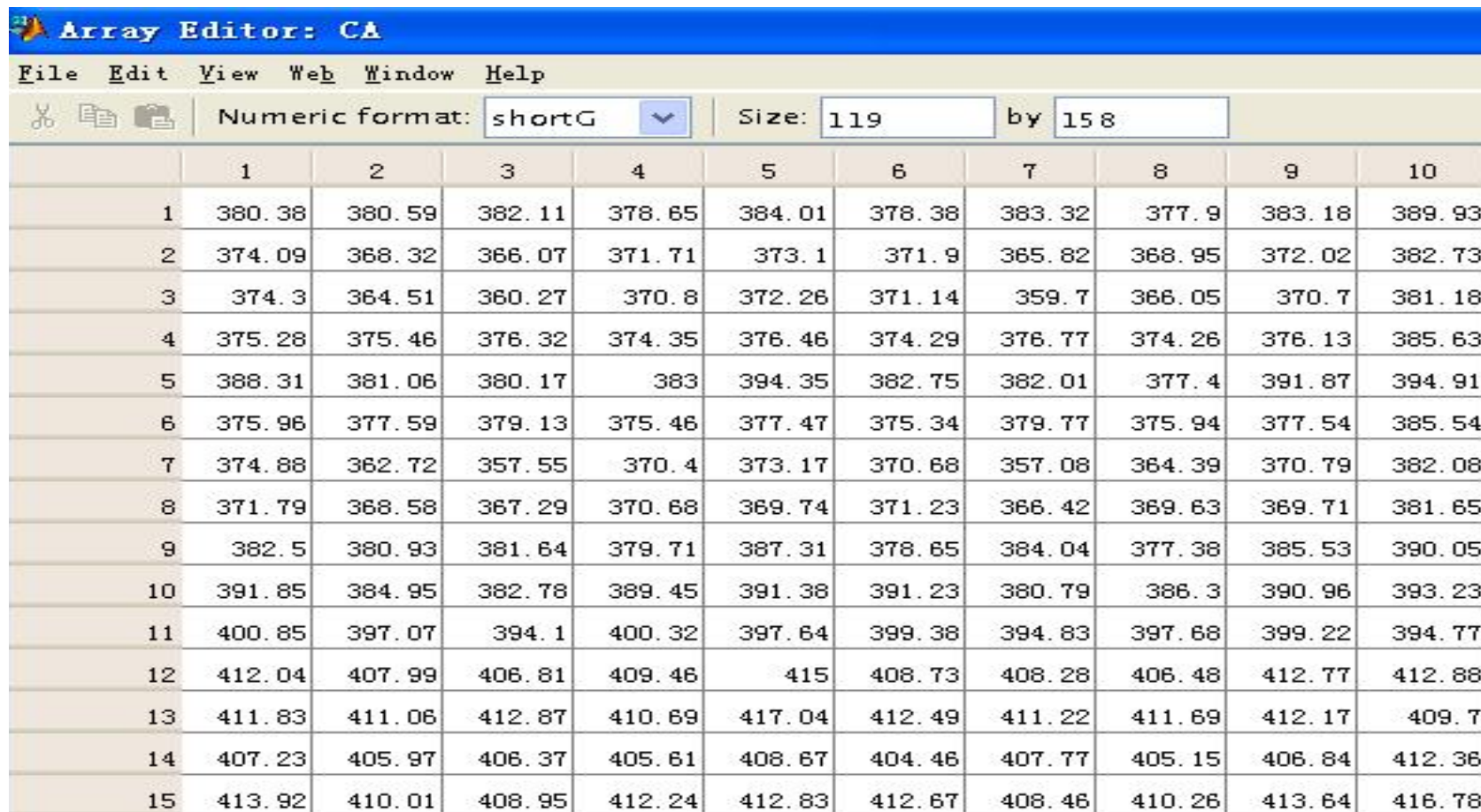
水印生成和嵌入

水印生成和嵌入策略流程图



步骤一：图像小波分解及低频系数归一化

- 利用MATLAB的wavedec2函数对图像进行小波分解，提取低频小波系数。

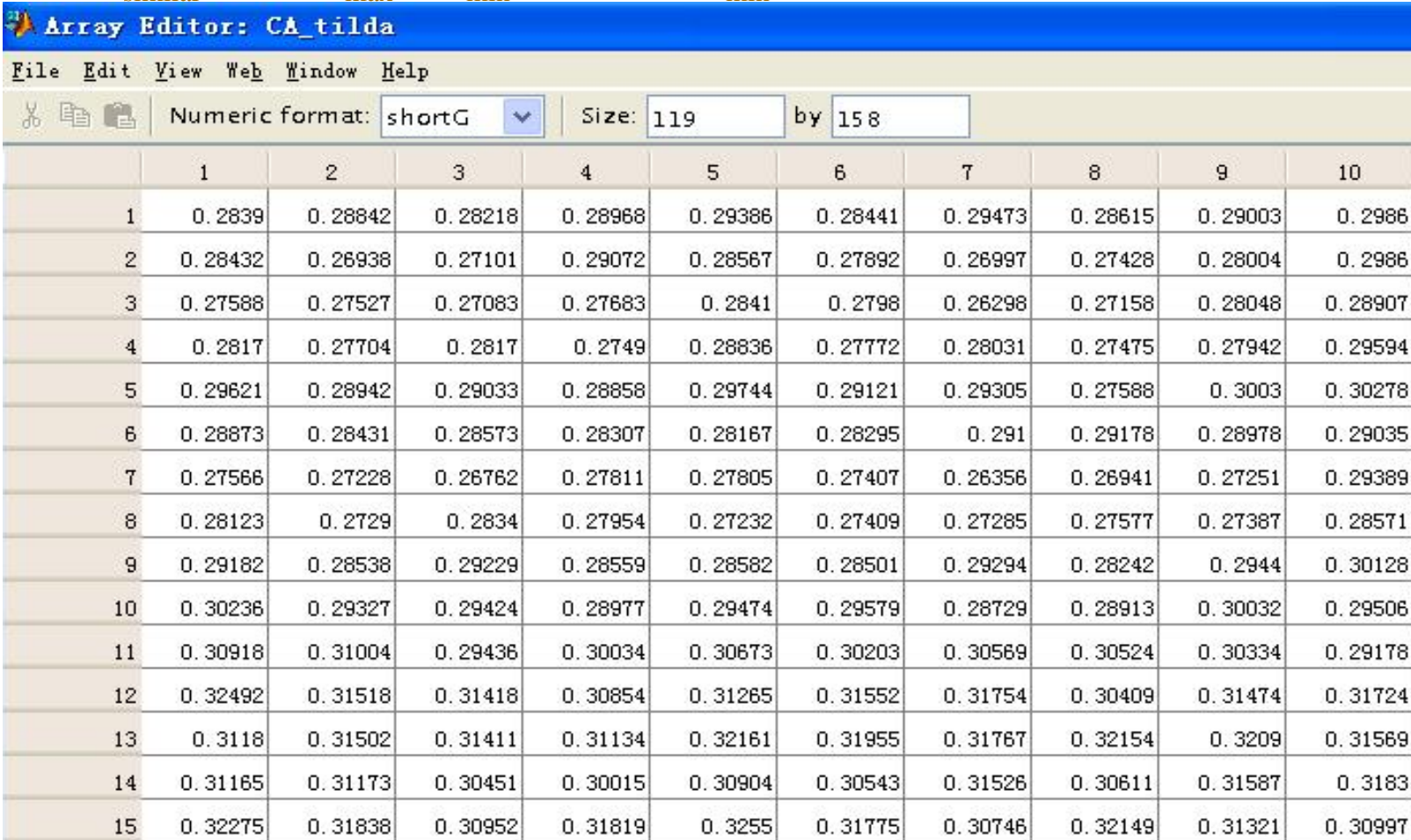


The image shows a screenshot of the MATLAB Array Editor window. The title bar reads "Array Editor: CA". The menu bar includes "File", "Edit", "View", "Web", "Window", and "Help". Below the menu bar, there are icons for cut, copy, and paste, followed by a "Numeric format:" dropdown set to "shortG", and "Size:" fields showing "119" by "158". The main area displays a 15x10 matrix of numerical values, representing wavelet coefficients. The values are rounded to two decimal places.

	1	2	3	4	5	6	7	8	9	10
1	380.38	380.59	382.11	378.65	384.01	378.38	383.32	377.9	383.18	389.93
2	374.09	368.32	366.07	371.71	373.1	371.9	365.82	368.95	372.02	382.73
3	374.3	364.51	360.27	370.8	372.26	371.14	359.7	366.05	370.7	381.18
4	375.28	375.46	376.32	374.35	376.46	374.29	376.77	374.26	376.13	385.63
5	388.31	381.06	380.17	383	394.35	382.75	382.01	377.4	391.87	394.91
6	375.96	377.59	379.13	375.46	377.47	375.34	379.77	375.94	377.54	385.54
7	374.88	362.72	357.55	370.4	373.17	370.68	357.08	364.39	370.79	382.08
8	371.79	368.58	367.29	370.68	369.74	371.23	366.42	369.63	369.71	381.65
9	382.5	380.93	381.64	379.71	387.31	378.65	384.04	377.38	385.53	390.05
10	391.85	384.95	382.78	389.45	391.38	391.23	380.79	386.3	390.96	393.23
11	400.85	397.07	394.1	400.32	397.64	399.38	394.83	397.68	399.22	394.77
12	412.04	407.99	406.81	409.46	415	408.73	408.28	406.48	412.77	412.88
13	411.83	411.06	412.87	410.69	417.04	412.49	411.22	411.69	412.17	409.7
14	407.23	405.97	406.37	405.61	408.67	404.46	407.77	405.15	406.84	412.36
15	413.92	410.01	408.95	412.24	412.83	412.67	408.46	410.26	413.64	416.78

步骤一：图像小波分解及低频系数归一化

放缩： $CA_{\text{similar}} = (1/(CA_{\text{max}} - CA_{\text{min}})) \times (CA - CA_{\text{min}})$ 得到了归一化后的原始图像低频矩阵。



Array Editor: CA_tilda

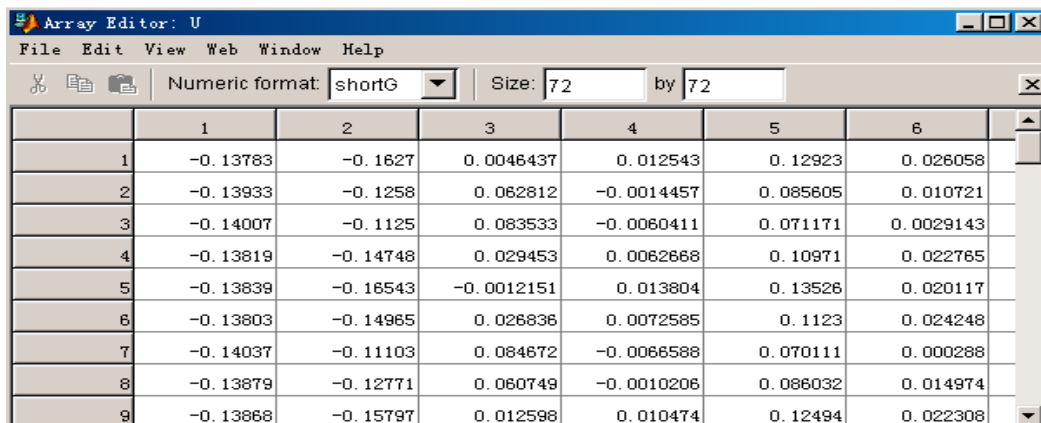
File Edit View Web Window Help

Numeric format: shortG Size: 119 by 158

	1	2	3	4	5	6	7	8	9	10
1	0.2839	0.28842	0.28218	0.28968	0.29386	0.28441	0.29473	0.28615	0.29003	0.2986
2	0.28432	0.26938	0.27101	0.29072	0.28567	0.27892	0.26997	0.27428	0.28004	0.2986
3	0.27588	0.27527	0.27083	0.27683	0.2841	0.2798	0.26298	0.27158	0.28048	0.28907
4	0.2817	0.27704	0.2817	0.2749	0.28836	0.27772	0.28031	0.27475	0.27942	0.29594
5	0.29621	0.28942	0.29033	0.28858	0.29744	0.29121	0.29305	0.27588	0.3003	0.30278
6	0.28873	0.28431	0.28573	0.28307	0.28167	0.28295	0.291	0.29178	0.28978	0.29035
7	0.27566	0.27228	0.26762	0.27811	0.27805	0.27407	0.26356	0.26941	0.27251	0.29389
8	0.28123	0.2729	0.2834	0.27954	0.27232	0.27409	0.27285	0.27577	0.27387	0.28571
9	0.29182	0.28538	0.29229	0.28559	0.28582	0.28501	0.29294	0.28242	0.2944	0.30128
10	0.30236	0.29327	0.29424	0.28977	0.29474	0.29579	0.28729	0.28913	0.30032	0.29506
11	0.30918	0.31004	0.29436	0.30034	0.30673	0.30203	0.30569	0.30524	0.30334	0.29178
12	0.32492	0.31518	0.31418	0.30854	0.31265	0.31552	0.31754	0.30409	0.31474	0.31724
13	0.3118	0.31502	0.31411	0.31134	0.32161	0.31955	0.31767	0.32154	0.3209	0.31569
14	0.31165	0.31173	0.30451	0.30015	0.30904	0.30543	0.31526	0.30611	0.31587	0.3183
15	0.32275	0.31838	0.30952	0.31819	0.3255	0.31775	0.30746	0.32149	0.31321	0.30997

步骤二：对低频相似矩阵做SVD变换

- 矩阵SVD变换：对于任意 $M \times N$ 矩阵 B ，都可以写成 $B = U\Sigma V^T$ ，其中 U 和 V 分别是 $M \times M$ 和 $N \times N$ 的正交矩阵。 Σ 是 $M \times N$ 的对角矩阵。
- SVD变换通过函数svd完成。（右图从上到下依次为 U 、 Σ 、 V 、）

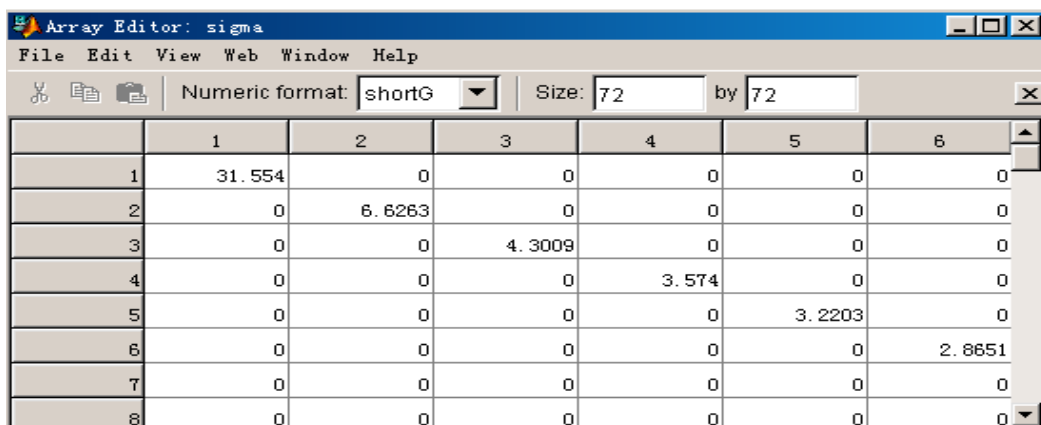


Array Editor: U

File Edit View Web Window Help

Numeric format: shortG Size: 72 by 72

	1	2	3	4	5	6
1	-0.13783	-0.1627	0.0046437	0.012543	0.12923	0.026058
2	-0.13933	-0.1258	0.062812	-0.0014457	0.085605	0.010721
3	-0.14007	-0.1125	0.083533	-0.0060411	0.071171	0.0029143
4	-0.13819	-0.14748	0.029453	0.0062668	0.10971	0.022765
5	-0.13839	-0.16543	-0.0012151	0.013804	0.13526	0.020117
6	-0.13803	-0.14965	0.026836	0.0072585	0.1123	0.024248
7	-0.14037	-0.11103	0.084672	-0.0066588	0.070111	0.000288
8	-0.13879	-0.12771	0.060749	-0.0010206	0.086032	0.014974
9	-0.13868	-0.15797	0.012598	0.010474	0.12494	0.022308

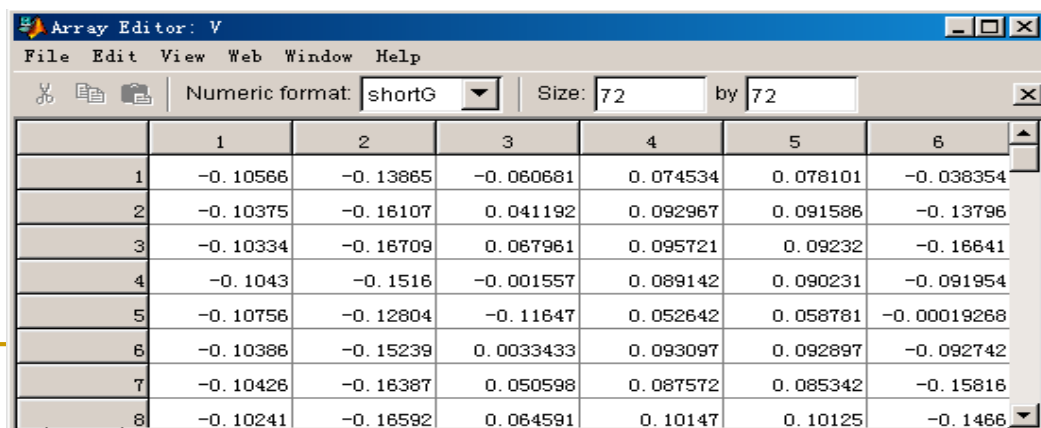


Array Editor: sigma

File Edit View Web Window Help

Numeric format: shortG Size: 72 by 72

	1	2	3	4	5	6
1	31.554	0	0	0	0	0
2	0	6.6263	0	0	0	0
3	0	0	4.3009	0	0	0
4	0	0	0	3.574	0	0
5	0	0	0	0	3.2203	0
6	0	0	0	0	0	2.8651
7	0	0	0	0	0	0
8	0	0	0	0	0	0



Array Editor: V

File Edit View Web Window Help

Numeric format: shortG Size: 72 by 72

	1	2	3	4	5	6
1	-0.10566	-0.13865	-0.060681	0.074534	0.078101	-0.038354
2	-0.10375	-0.16107	0.041192	0.092967	0.091586	-0.13796
3	-0.10334	-0.16709	0.067961	0.095721	0.09232	-0.16641
4	-0.1043	-0.1516	-0.001557	0.089142	0.090231	-0.091954
5	-0.10756	-0.12804	-0.11647	0.052642	0.058781	-0.00019268
6	-0.10386	-0.15239	0.0033433	0.093097	0.092897	-0.092742
7	-0.10426	-0.16387	0.050598	0.087572	0.085342	-0.15816
8	-0.10241	-0.16592	0.064591	0.10147	0.10125	-0.1466

步骤三：

正交随机矩阵的生成

在W-SVD算法中，我们需要生成 \bar{U} 和 \bar{V} 两个随机的正交矩阵。

对于一个 $M \times N$ 矩阵 C ，其QR分解是将一个 $M \times M$ 的正交矩阵 Q 和一个 $M \times N$ 的矩阵 R 。

在MATLAB中，QR变换是通过内置函数qr完成的。（右图从上到下依次为一般随机矩阵、 Q 和 R ）

The figure displays three screenshots of the MATLAB Array Editor, showing the results of generating random matrices and their QR decomposition. Each window is titled 'Array Editor: [matrix name]' and shows a 72x72 matrix in 'shortG' format.

Array Editor: randmatrix

	1	2	3	4	5	6
1	0.95013	0.8385	0.13377	0.96689	0.70357	0.81212
2	0.23114	0.56807	0.20713	0.66493	0.48496	0.61011
3	0.60684	0.37041	0.6072	0.87038	0.11461	0.70149
4	0.48598	0.70274	0.62989	0.0099273	0.66486	0.092196
5	0.8913	0.54657	0.37048	0.13701	0.36537	0.42489
6	0.7621	0.44488	0.57515	0.81876	0.14004	0.37558
7	0.45647	0.69457	0.45142	0.43017	0.56677	0.16615
8	0.018504	0.62131	0.043895	0.89032	0.82301	0.83315
9	0.82141	0.79482	0.027185	0.73491	0.67395	0.83864

Array Editor: Q

	1	2	3	4	5	6
1	-0.18973	-0.013943	0.18366	0.11654	-0.027394	0.016731
2	-0.046157	-0.12312	0.18358	0.12832	-0.0027349	0.089436
3	-0.12118	0.041651	-0.098114	0.12942	-0.17928	0.08632
4	-0.097047	-0.03883	-0.087324	-0.23303	0.13658	-0.14245
5	-0.17799	0.063352	0.051227	-0.19064	0.010937	-0.017672
6	-0.15219	0.053915	-0.055353	0.061331	-0.17428	-0.092755
7	-0.091153	-0.13406	-0.025742	-0.041957	0.037969	-0.14736
8	-0.003695	-0.13729	0.055382	0.26042	0.091255	0.18446
9	-0.16403	-0.038137	0.19702	0.063494	0.012652	0.08331

Array Editor: R

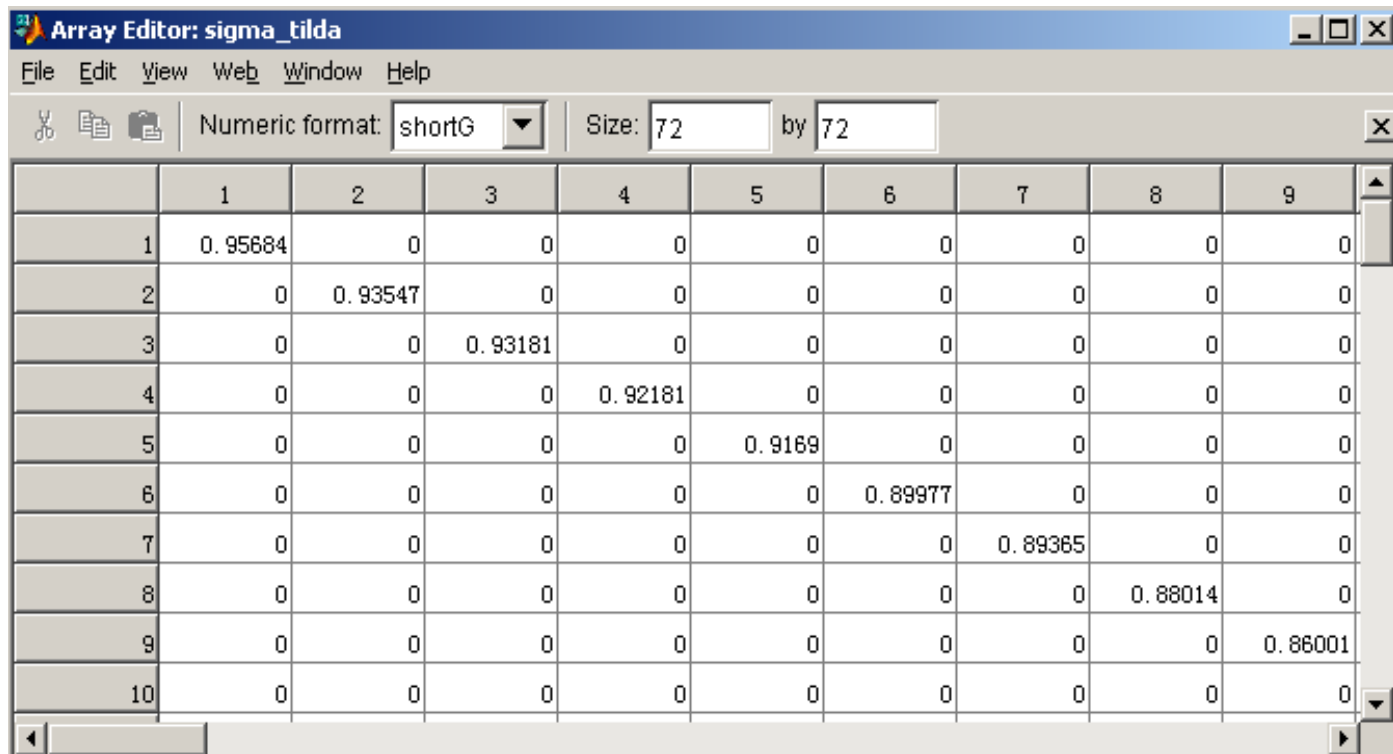
	1	2	3	4	5	6
1	-5.0077	-4.1126	-3.1592	-4.1972	-3.7927	-3.8887
2	0	-3.0722	-0.90957	-1.2827	-1.45	-1.298
3	0	0	-2.6729	-0.84616	-0.57768	-0.35624
4	0	0	0	2.5664	1.1677	0.7092
5	0	0	0	0	2.7466	0.13404
6	0	0	0	0	0	2.0893
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

步骤四：随机对角矩阵的生成

- 随机对角矩阵生成：

$\text{sigma_tilda} = \alpha * \text{diag}(\text{flipud}(\text{sort}(\text{rand}(d,1))));$

- sigma_tilda 就是算法中的 Σ 。
- α 就是强度因子 α ， d 是根据 d/n 计算出的要替换的行数。



Array Editor: sigma_tilda

File Edit View Web Window Help

Numeric format: shortG Size: 72 by 72

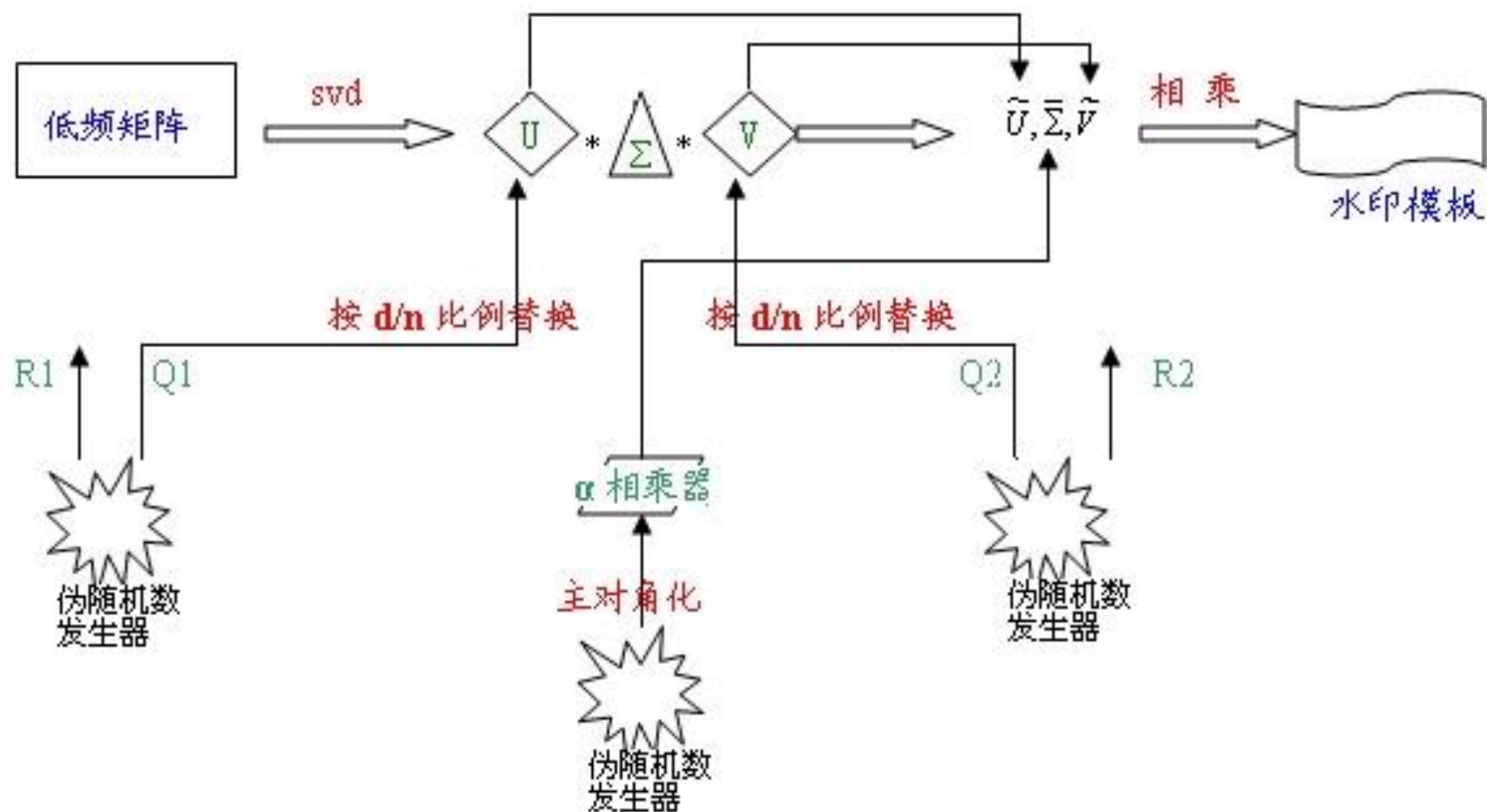
	1	2	3	4	5	6	7	8	9
1	0.95684	0	0	0	0	0	0	0	0
2	0	0.93547	0	0	0	0	0	0	0
3	0	0	0.93181	0	0	0	0	0	0
4	0	0	0	0.92181	0	0	0	0	0
5	0	0	0	0	0.9169	0	0	0	0
6	0	0	0	0	0	0.89977	0	0	0
7	0	0	0	0	0	0	0.89365	0	0
8	0	0	0	0	0	0	0	0.88014	0
9	0	0	0	0	0	0	0	0	0.86001
10	0	0	0	0	0	0	0	0	0

步骤五：替换

水印模板的产生。用种子控制的随机正交矩阵 \bar{U} 和 \bar{V} 的后 d 列（行）来替换原始低频系数分解矩阵 U 和 V 的后 d 列（行），得到矩阵 U 和 V 。

d 是一个由比例因子 d/n 决定的整数。

步骤五: 替换



步骤六：用替换后的低频系数矩阵重构图像

- 利用MATLAB的waverec2函数对图像进行重构。
- 考虑到MATLAB精度对实验结果的影响，建议将重构后的图像以16位图（如png格式）的方式存储。

W-SVD数字水印算法函数代码

```
% 文件名:wavemarksvd.m  
% 函数功能:本函数将完成 W-SVD 模型下数字水印的嵌入  
% 参数说明:  
% input 为输入原始图像  
% seed 为随机数种子  
% wavelet 为使用的小波函数  
% level 为小波分解的尺度  
% alpha 为水印强度  
% ratio 为算法中  $d/n$  的比例  
% watermarkimagergb 为加有水印的结果  
% watermarkimage 为单层加水印的结果  
% waterCA 为加有水印模板的低频分解系数  
% watermark2 为由水印模板直接重构得到的水印形态
```

```
% correlationU, correlationV 为替换正交矩阵后与未替换的正交矩阵的相关系数
function
[ watermarkimagergb, watermarkimage, waterCA, watermark2, correlationU, correlationV ] = wavemarksvd( input, goal, seed, wavelet, level, alpha, ratio)
% function watermark = wavemarksvd( input, goal, seed, wavelet, level, alpha, ratio)
% 读取原始图像
data = imread( input );
data = double( data )/255;
datared = data( :, :, 1 ); % 在 R 层加水印

% 对原始图像的 R 层进行小波分解记录原始大小,并将其补成正方形
[ C, Sreal ] = wavedec2( datared, level, wavelet );
[ row, list ] = size( datared );
standard1 = max( row, list );
new = zeros( standard1, standard1 );
```

```
if row <= list
    new(1:row,:) = datared;
else
    new(:,1:list) = datared;
end
% 正式开始加水印
% 小波分解,提取低频系数
[ C,S] = wavedec2( new,level,wavelet);
CA = appcoef2( C,S,wavelet,level);
% 对低频系数进行归一化处理
[ M,N] = size( CA);
CAmin = min( min( CA));
CAmax = max( max( CA));
CA = ( 1/( CAmax - CAmin)) * ( CA - CAmin);
d = max( size( CA));
```

% 对低频率系数单值分解

[U, sigma, V] = svd(CA);

% 按输出参数得到要替换的系数的数量

np = round(d * ratio);

% 以下是随机正交矩阵的生成

rand('seed', seed);

M_V = rand(d, np) - 0.5;

[Q_V, R_V] = qr(M_V, 0);

M_U = rand(d, np) - 0.5;

[Q_U, R_U] = qr(M_U, 0);


```
% 替换
V2 = V; U2 = U;
V(:, d - np + 1 : d) = Q_V(:, 1 : np);
U(:, d - np + 1 : d) = Q_U(:, 1 : np);
sigma_tilda = alpha * flipud( sort( rand( d, 1 ) ) );
correlationU = corr2( U, U2 ); % 计算替换的相关系数
correlationV = corr2( V, V2 );
% 生成水印
watermark = U * diag( sigma_tilda, 0 ) * V';
% 重构生成水印的形状, 便于直观认识, 本身无意义
watermark2 = reshape( watermark, 1, S( 1, 1 ) * S( 1, 2 ) );
waterC = C;
waterC( 1, 1 : S( 1, 1 ) * S( 1, 2 ) ) = watermark2;
watermark2 = waverec2( waterC, S, wavelet );
```

% 调整系数生成嵌入水印后的图像

```
CA_tilda = CA + watermark;
```

```
over1 = find( CA_tilda > 1 );
```

```
below0 = find( CA_tilda < 0 );
```

```
CA_tilda( over1 ) = 1;
```

```
CA_tilda( below0 ) = 0; % 系数调整, 将过幅系数与负数修正
```

```
CA_tilda = ( CAmix-Camin) * CA_tilda + CAmix; % 系数还原到归一化以前的范围
```

% 记录加有水印的低频系数

```
waterCA = CA_tilda;
```

```
if row <= list
```

```
    waterCA = waterCA( 1:Sreal(1,1),: );
```

```
else
```

```
    waterCA = waterCA( :, 1:Sreal(1,2) );
```

```
end
```

% 重构

```
CA_tilda = reshape( CA_tilda,1,S(1,1) * S(1,2));
```

```
C(1,1:S(1,1) * S(1,2)) = CA_tilda;
```

```
watermarkimage = waverec2( C,S,wavelet);
```

% 将前面补上的边缘去掉

```
if row <= list
```

```
    watermarkimage = watermarkimage( 1: row, : );
```

```
else
```

```
    watermarkimage = watermarkimage( : , 1: list );
```

```
end
```

```
watermarkimagergb = data;
```

```
watermarkimagergb( : , : , 1 ) = watermarkimage;
```

```
imwrite( watermarkimagergb,goal,'BitDepth',16);% 通过写回修正过幅系数
```

```
watermarkimagergb2 = imread( goal );
```



W-SVD水印效果

原始图片



嵌入水印后的RGB图片



水印



R层图片



嵌入水印后的R层图片



$\alpha=0.1$
 $d/n=0.99$
wavelet=db6
level=2
seed=10

原始图片



嵌入水印后的 RGB 图片



水印形态图



R 层图片



嵌入水印后的 R 层图片



$\alpha=0.3$
 $d/n=0.5$
wavelet=db6
level=2
seed=10

第二部分

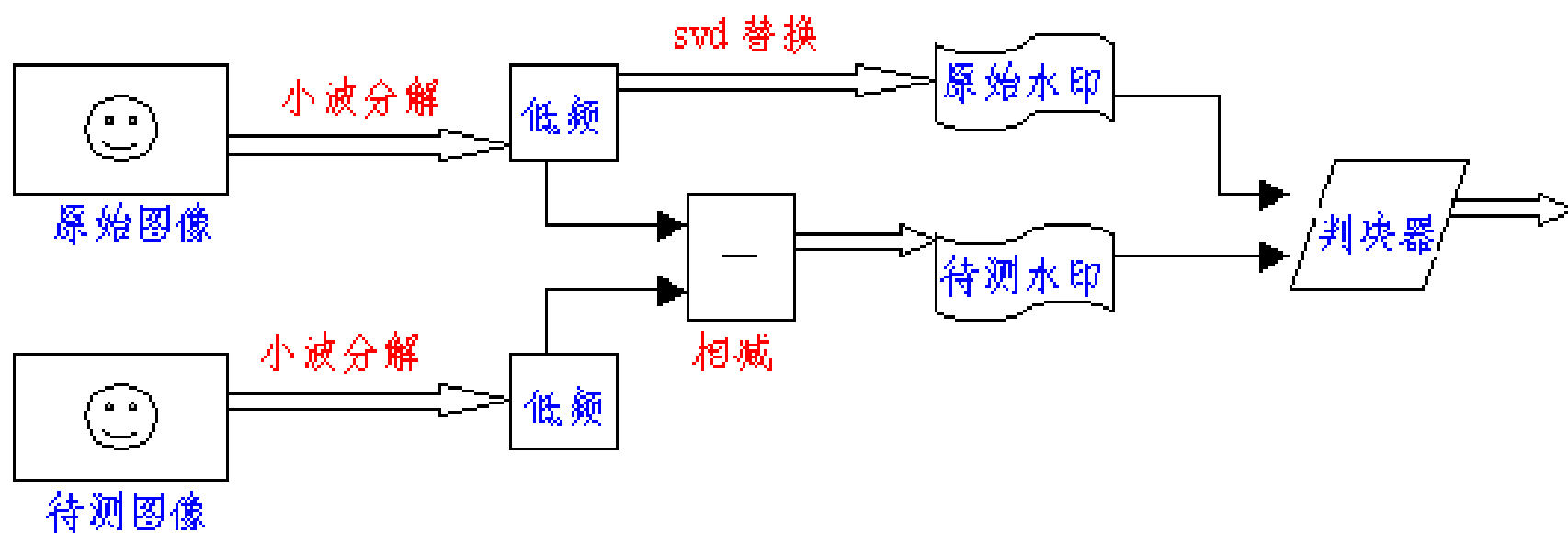


水印的检测

W-SVD水印的检测策略

W-SVD算法采用非盲检测手段对图像进行检测。其思路为：利用原始图像生成一个理论上存在的水印模板（**原始水印**），从待测图像中提取可能存在的水印模板（**待测水印**），继而计算二者的相关性。当二者高度相关时，我们认为待测图像含有水印；反之则检测不出水印。

W-SVD水印的检测策略



计算检测相关性值的方法

■ 常规检测直接相关性值 d :

$$d = \frac{|\langle W, W' \rangle|}{\|W\| \cdot \|W'\|}$$

■ DCT检测相关性值 \hat{d} :

$$\hat{d} = \frac{|\langle \hat{W}, \hat{W}' \rangle|}{\|\hat{W}\| \cdot \|\hat{W}'\|}$$

W 和 W' 分别表示原始水印和待测水印。

M 和 N 为水印模板的大小。

$$\langle W, W' \rangle = \sum_{i=1}^M \sum_{j=1}^N w_{ij} w'_{ij}$$

$$\|W\| = \sqrt{\langle W, W \rangle}$$

$\hat{\cdot}$ 表示经过 DCT 后的系数矩阵

水印检测的具体步骤

- 给原始图像加水印并提取其加有水印图像的小波低频系数 A 。
- 提取待测图像的小波低频系数 B 。
- 提取原始图像的小波低频系数 C 。
- 生成原始水印 $(A-C)$ 和待测水印 $(B-C)$ 。
- 计算相关性值。

一个问题的提出：

计算出了相关性值如何判定是否有水印？



- 根据一个检测阈值 (Test_threshold) 来决定水印的有无。
- 在数字水印检测中，会出现两类错误：**虚警错误**和**漏警错误**。前者是指将没有水印的图像判定为有水印，后者则是将有水印的图像判定为无水印。当检测阈值选取过大时，就会造成漏警概率过大；而当检测阈值选取过小时，就会造成虚警概率过大。
- 绘制“种子—相关性值图” (SC图) 是我们分析水印系统的一个重要手段。

W-SVD模型下数字水印检测

```
% 文件名:wavedetect.m
% 函数功能:本函数将完成 W-SVD 模型下数字水印的检测
% 输入格式举例:
[corr_coef,corr_DCTcoef] = wavedetect('c:\test.png','c:\lenna.jpg',1983,
                                     'db6',2,0.1,0.99)

% 参数说明:
% input 为输入原始图像
% seed 为随机数种子
% wavelet 为使用的小波函数
% level 为小波分解的尺度
% alpha 为水印强度
% ratio 为算法中 d/n 的比例
% corr_coef,corr_DCTcoef 分别为不同方法下检测出的相关性值
function [corr_coef,corr_DCTcoef] = wavedetect(test,original,seed,wavelet,level,
                                              alpha,ratio)
```

```
dataoriginal = imread( original ) ;  
datatest = imread( test ) ;  
dataoriginal = double( dataoriginal ) / 255 ;  
datatest = double( datatest ) / 65535 ;  
  
dataoriginal = dataoriginal( : , : , 1 ) ;  
datatest = datatest( : , : , 1 ) ;  
  
% 提取加有水印的图像的小波低频系数  
[ watermarkimagergb , watermarkimage , waterCA , watermark2 , correlationU , correlationV ] = wavemarksvd( original , 'temp. png' , seed , wavelet , level , alpha , ratio ) ;
```

```

% 提取待测图像的小波低频系数
[ C,S] = wavedec2( datatest,level,wavelet) ;
CA_test = appcoef2( C,S,wavelet,level) ;
% 提取原始图像的小波低频系数
[ C,S] = wavedec2( dataoriginal,level,wavelet) ;

realCA = appcoef2( C,S,wavelet,level) ;

% 生成两种水印
realwatermark = waterCA-realCA ;
testwatermark = CA_test-realCA ;

% 计算相关性值
corr_coef = trace( realwatermark' * testwatermark ) /
              ( norm( realwatermark, 'fro') * [ testwatermark, 'fro'] ) ;

```

% DCT 系数比较

DCTrealwatermark = dct2(waterCA-realCA);

DCTtestwatermark = dct2(CA_test-realCA);

DCTrealwatermark = DCTrealwatermark (1 : min (32 , max (size (DCTrealwatermark))) , 1 : min (32 , max (size (DCTrealwatermark)))) ;

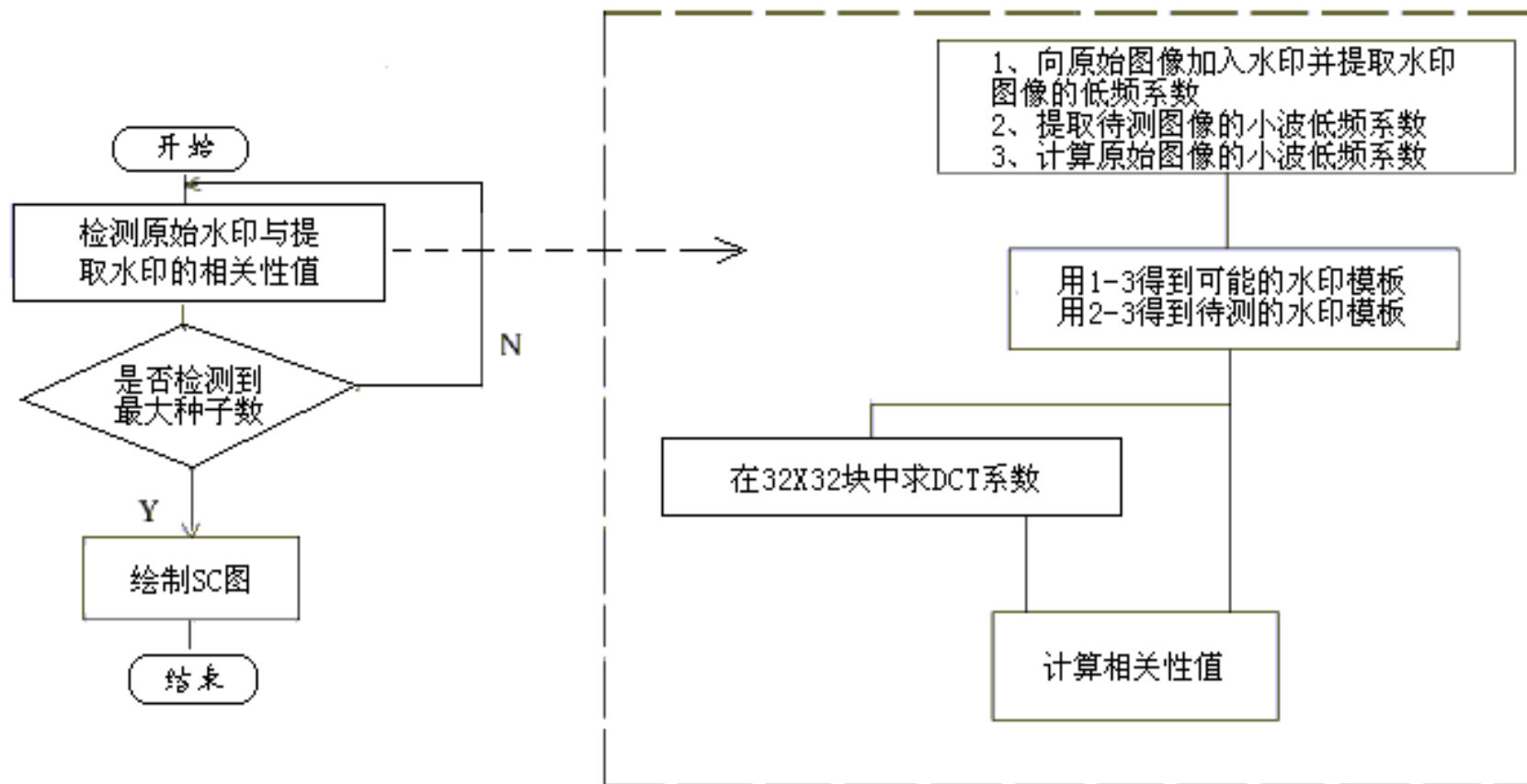
DCTtestwatermark = DCTtestwatermark (1 : min (32 , max (size (DCTtestwatermark))) , 1 : min (32 , max (size (DCTtestwatermark)))) ;

DCTrealwatermark(1,1) = 0 ;

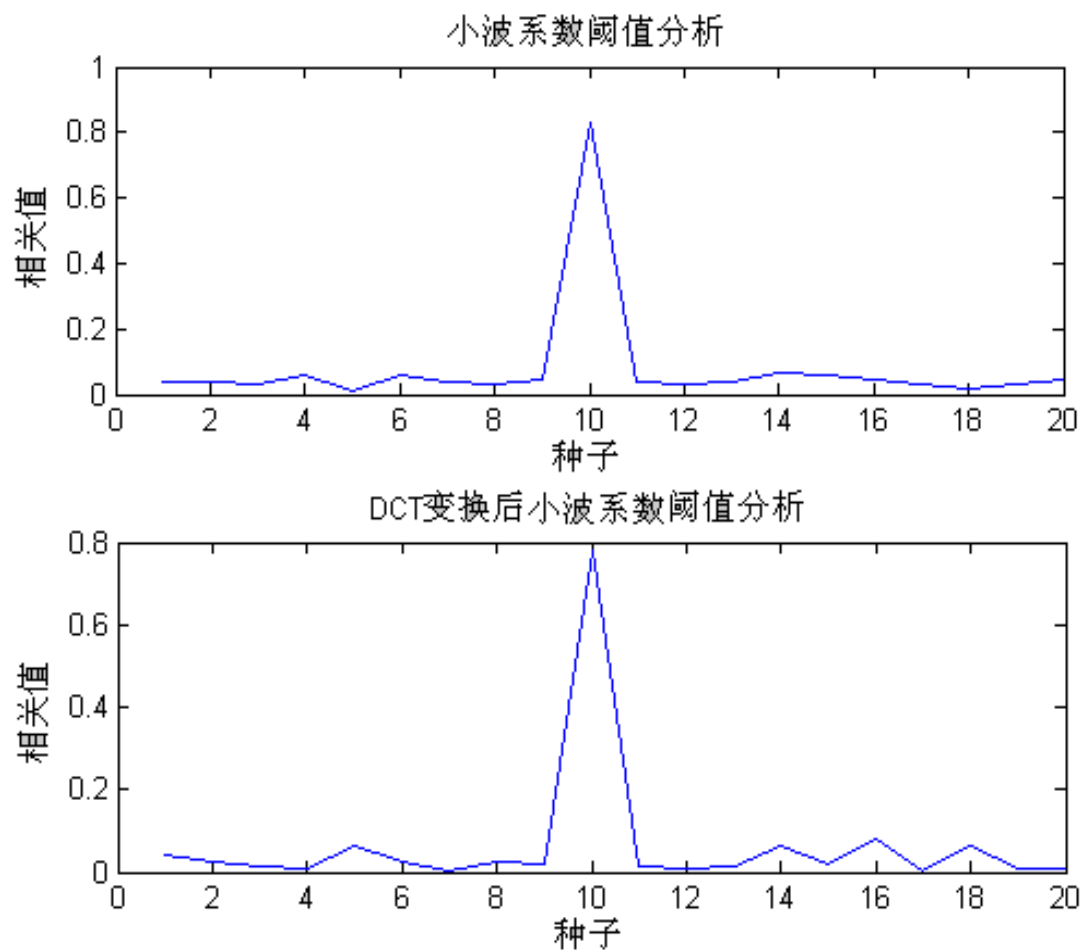
DCTtestwatermark(1,1) = 0 ;

corr_DCTcoef = trace (DCTrealwatermark' * DCTtestwatermark) / (norm (DCTrealwatermark , 'fro') * norm (DCTtestwatermark , 'fro')) ;

SC图的绘制流程图



两种检测方法所绘制的SC图



SC图的特点及作用

- 当一个确有水印的图像检测绘制的SC图出现明显且唯一的峰值时，则表明检测手段是理想的。
- SC图可以给我们选择检测阈值提供依据。
- 检测阈值给我们对水印系统的鲁棒性测试提供了保证。
- SC图可以给我们提供判定算法参数与水印鲁棒性的关系的依据。
- SC图是攻击水印系统的有效手段。

第三部分



W-SVD水印系统性能分析

从以下几个方面进行性能测试

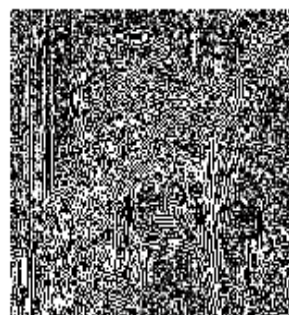
- 算法各参数与水印生成的关系
- 算法各参数与水印鲁棒性的关系
- 算法各参数与水印安全性的关系
- 算法各参数与水印不可见性的关系

算法各参数指：① α ② d/n ③使用的小波 ④小波分解的尺度 ⑤随机数种子

算法各参数与水印生成的关系

- d/n 取值越小，表示原图像特征系数被替换得越少，水印形态与原始图像越相象。
- α 越大水印的能量越大。
- 不同的小波基分解和同一小波不同尺度下的分解生成的水印在形态、与原始图像的相关性、信息容量和随机性等各方面也不同。

算法各参数与水印生成的关系



level=1



level=2



level=3



level=4



level=5



level=6

选择不同尺度分解下的水印形态图。“水印形态图”是指以水印模板直接作为图像重构的低频系数而获得的重构图像，其与原图像越相象表明水印能量越低。

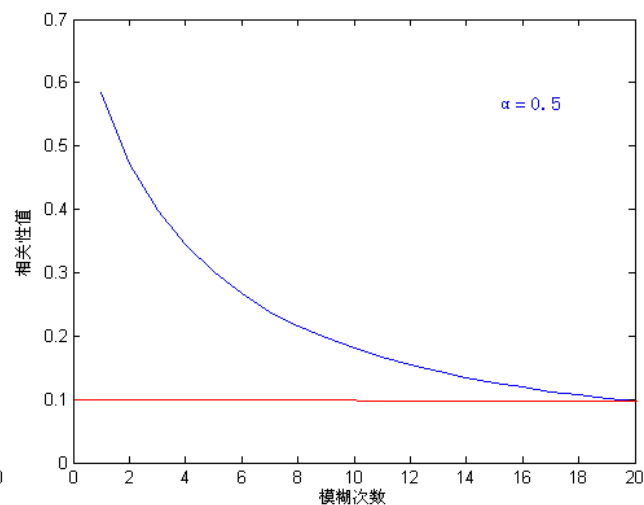
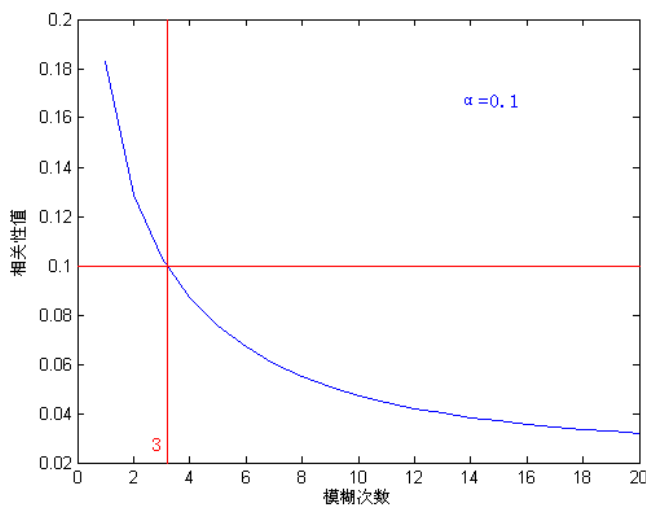
算法各参数与水印鲁棒性的关系

- 在鲁棒性测试中，我们将使用到①JPEG压缩 ②图像模糊化 ③图像中值滤波 ④图像马赛克处理这4种攻击方法。
- 利用SC图和攻击强度-相关性值图分析水印鲁棒性。

算法各参数与水印鲁棒性的关系

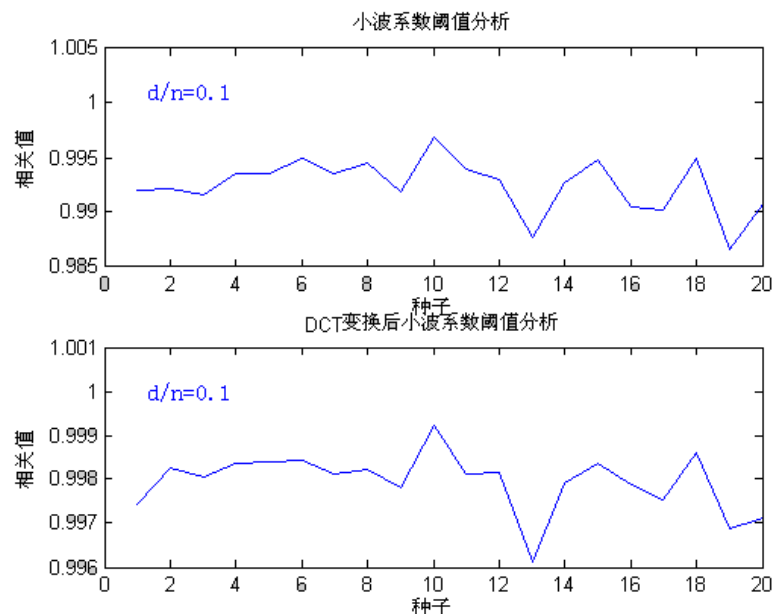
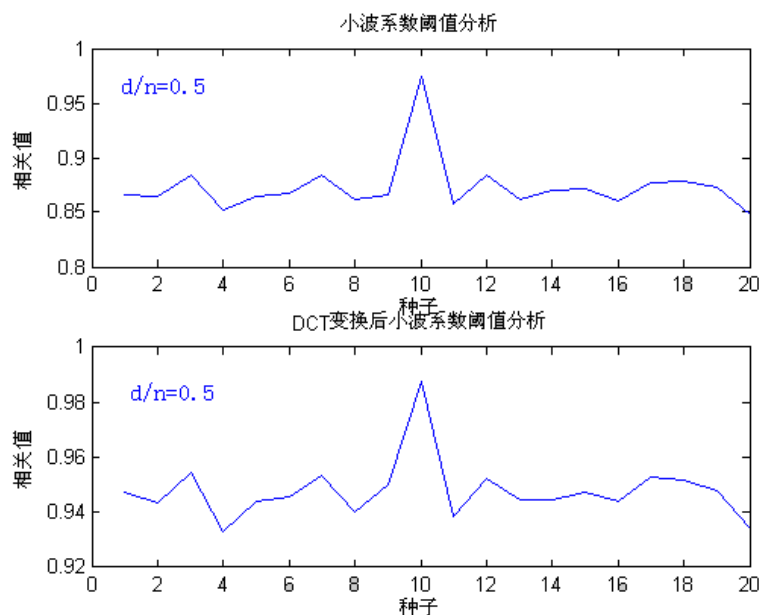
参数 α 是与水印鲁棒性密切相关。事实上， α 的取值略微改变一点，水印性能都会发生很大变化。

参数 α 越大，水印的抗攻击性能越好。



算法各参数与水印鲁棒性的关系

d/n 越大则水印检测的效果越好，对检测更有利。当 d/n 过小时，其检测SC图上已无法找到明显的峰值数据，所以也就谈不上检测阈值了。



算法各参数与水印鲁棒性的关系

小波分解的尺度越大，与之相关的水印信息越少，检测越困难。小波分解的尺度越大，水印越能嵌入到图像的高能量部分（低频部分），水印鲁棒性越强。（这是一对矛盾，在具体实验时要注意）

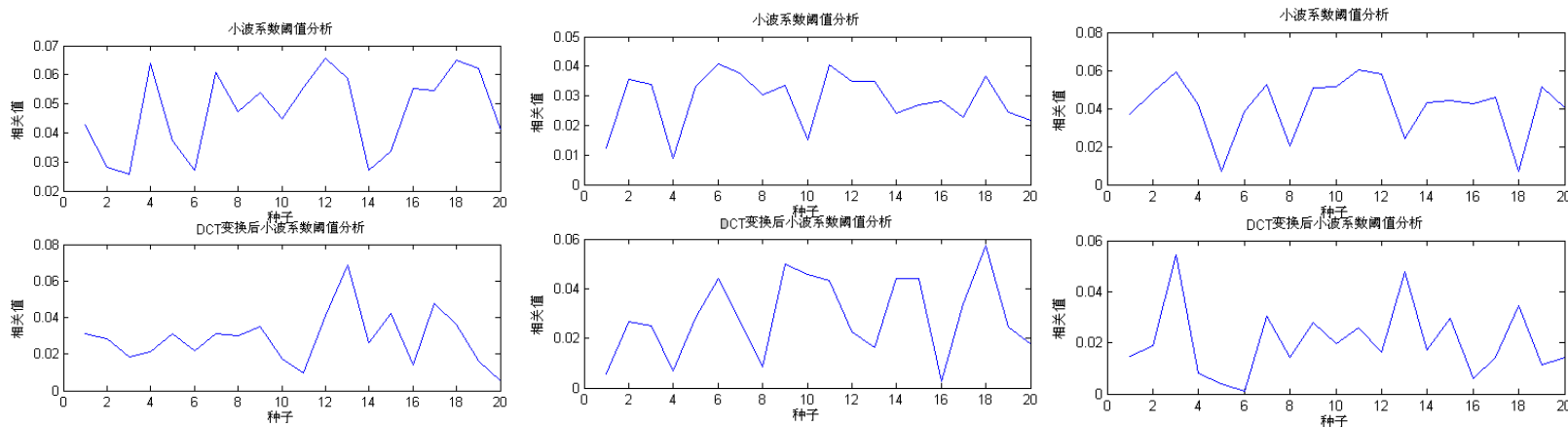
我们认为在W—SVD中取小波分解尺度为2或3是合适的。

算法各参数与水印安全性的关系

- Kerckhoffs准则认为：一个安全保护系统的安全性不是建立在它的算法对于对手来说是保密的，而是应该建立在它所选择的密钥对于对手来说是保密的。
- 一部分学者认为对所要加入的信息进行加密，直接引入密码学中的密钥为密钥；另一部分学者则认为水印嵌入的位置和相关参数也应该是密钥而不是算法的部分。
- 通过绘制SC图我们发现，在W-SVD中，将算法各参数均认为是密钥的一部分是合适的。

算法各参数与水印安全性的关系

以下是在不知道小波基、分解尺度、 α 和 d/n 的情况下随意定义这些参数对水印进行穷举检测绘制的SC图。显然，除了在不知道参数 α 外均无法认定图像是否嵌有水印。（下图从左到右依次为不知道小波基、分解尺度和 d/n ）



算法各参数与水印不可见性的关系

- 随着水印强度因子 α 的不断增大，水印对原始图像的破坏也越来越大，水印的不可见性降低。
- d/n 对水印不可见性影响不大。
- 随着小波分解尺度的增加，水印更为集中在图像能量高的部分，对图像的感知质量造成的影响越来越大。