DIGITAL IMAGE PROCESSING

FINAL PROJECT

Adel P. Bordbari

(**********)

Prof DR. *****

WINTER - 1401

TABLE OF CONTENTS

I.

INTRODUCTION

An overview of the final project, requirements, motives and goals

INTRODUCTION

Includes two main tasks:

compression and watermarking

Highlights the applications of DWT

Compression

- "Art and science of reducing the amount of required data for representation"
- Exists because data can be redundant and/or unnecessary
- Useful since a lot of data exists!
- Less requirements to save, share and process
- Can be lossy or lossless

INTRODUCTION

Watermarking

- Adding extra data for a purpose
- Somehow the opposite of compression
- Useful since a lot of data exists, again!
- Used for authentication and copyright issues

INTRODUCTION

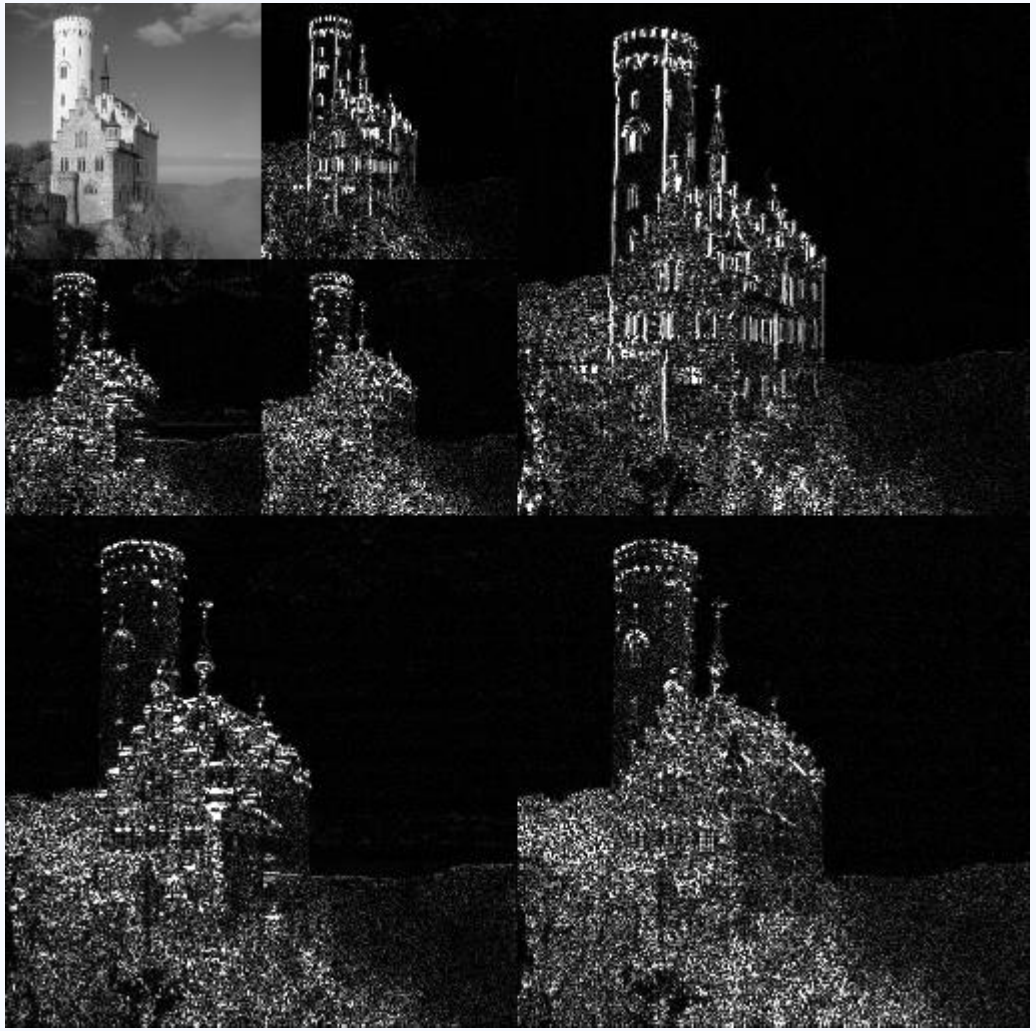
DWT

- Both tasks require DWT
- Discrete wavelet transform
- Sampling different wavelets separately
- Works like FFT; but better because considers temporal resolution (freq. and location)
- Based on mother wavelets: Haar, Daubechies, etc.
- Used in many methods: JPEG2000, JPEG XS, DjVu,

INTRODUCTION

Example:

coefficient matrices for2-D, 2-level DWT



II.

COMPRESSION

Thoughts, results and details on the first main task: Image compression using DWT

HIGHLIGHTS

Method

- Read 3 images
- Use 2-D DWT (Haar and Daubechies) to compress (or 3-D for color image)
- Quantize coefficients (local/global)
- Compare original and compressed images using MSE and PSNR metrics
- Show results in every step
- Global threshold is 100

Challenges

- Input images need to be diverse, include different frequency bands
- Some representations might be difficult to distinguish on computer screens
- Setting global / calculating local threshold
- Processing color channel (read as 2D or 3D?)

Notes

- One image is small and mostly include lower frequencies
- Two real life image, large, include many frequencies and edges
- All color images, all square aspect ratio (1:1)

Steps

1. **Haar 3-L**
    1.1. with local thresholding
    1.2. with global thresholding
2. **Daubechies 3-L**
    2.1. with local thresholding
    2.2. with global thresholding
3. **Haar 5-L**
    3.1. with local thresholding
    3.2. with global thresholding
4. **Daubechies 5-L**
    4.1. with local thresholding
    4.2. with global thresholding
5. **Results**

Image 1:

circ.jpg

1789x1789px

300dpi

24bit-depth

1.35Mb

Image 2:

rail.jpg

3000x3000px

96dpi

24bit-depth

3.37Mb

Image 3:

disco.jpg

576x576px

96dpi

24bit-depth

78.7Kb

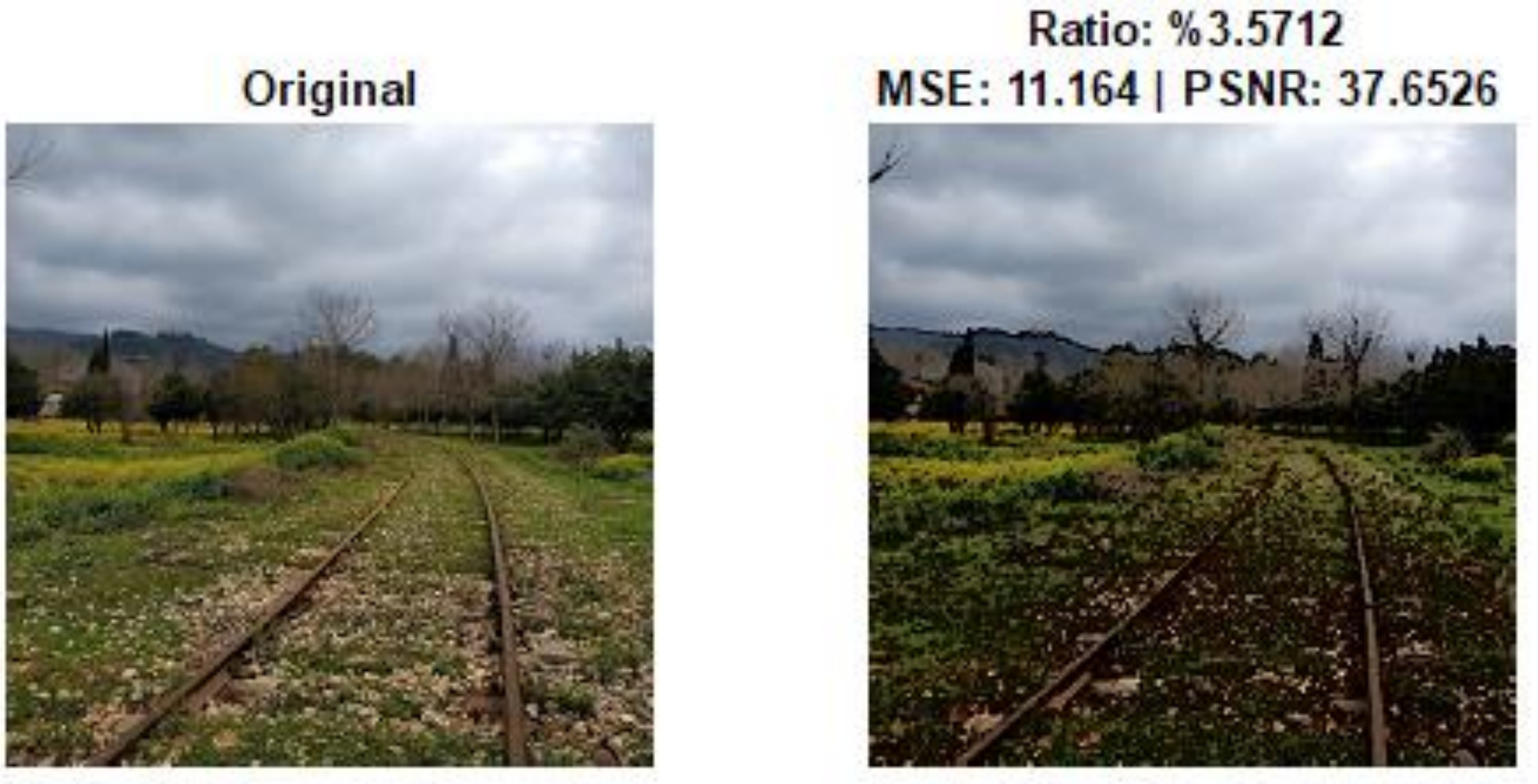
II. 1

HAAR 3-L

1. Threshold = local
2. Threshold = 100 (global)

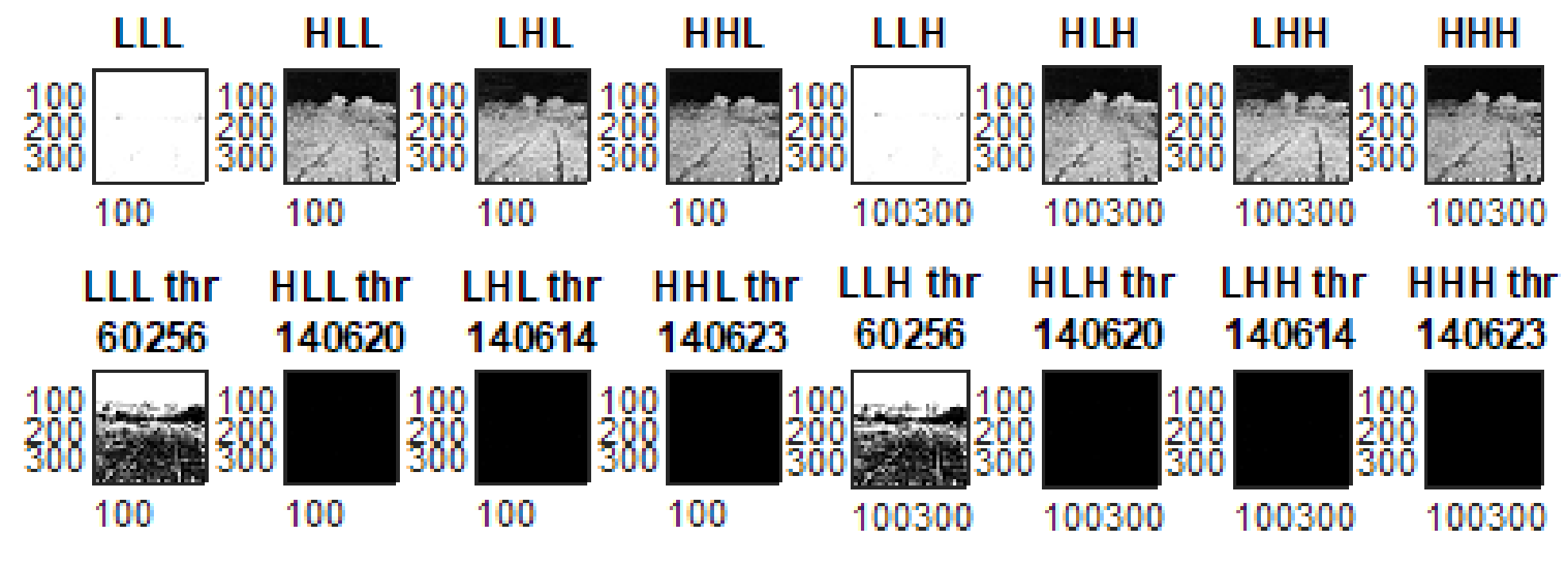
RESULT

1.1 *WITH LOCAL THRESHOLDING*



COEFFS

1.1 *WITH LOCAL THRESHOLDING*



RESULT

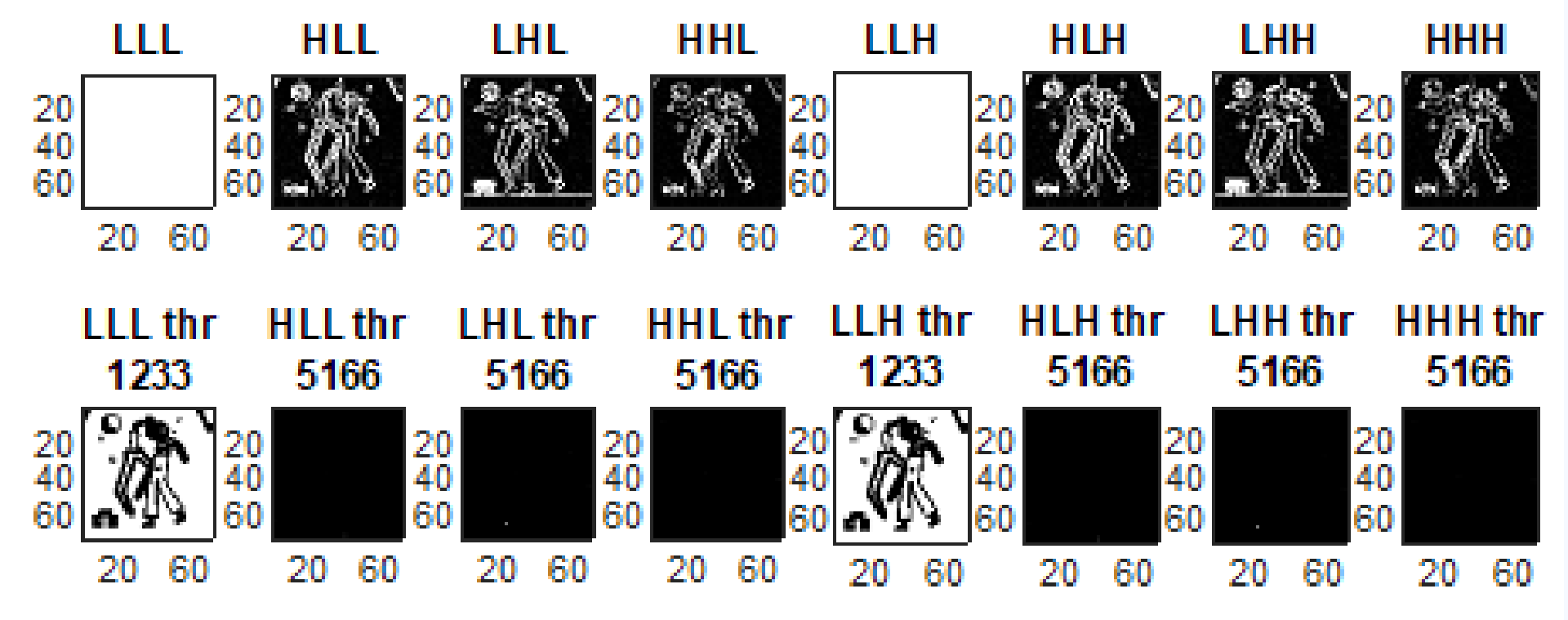
1.1 *WITH LOCAL THRESHOLDING*

Original

Ratio: %3.3619
MSE: 2.0231 | PSNR: 45.0706

COEFFS

1.1 WITH LOCAL THRESHOLDING



RESULT

1.1 *WITH LOCAL THRESHOLDING*

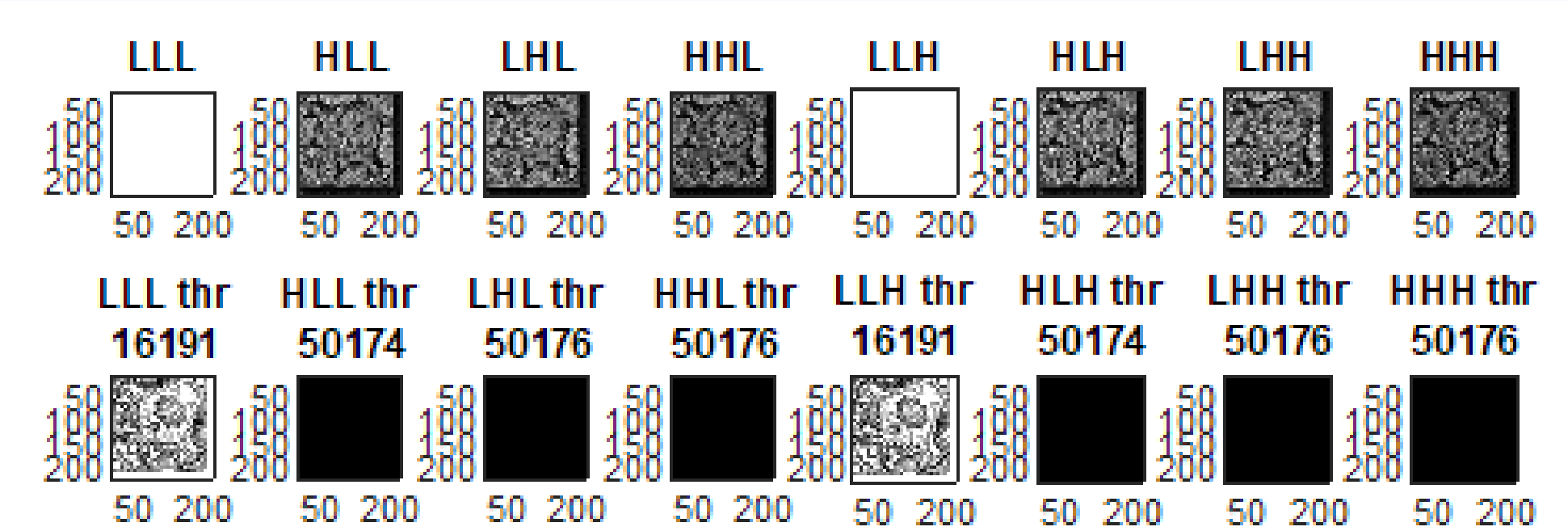
Original

Ratio: %3.4727

MSE: 7.6394 | PSNR: 39.3002

COEFFS

1.1 *WITH LOCAL THRESHOLDING*



RESULT

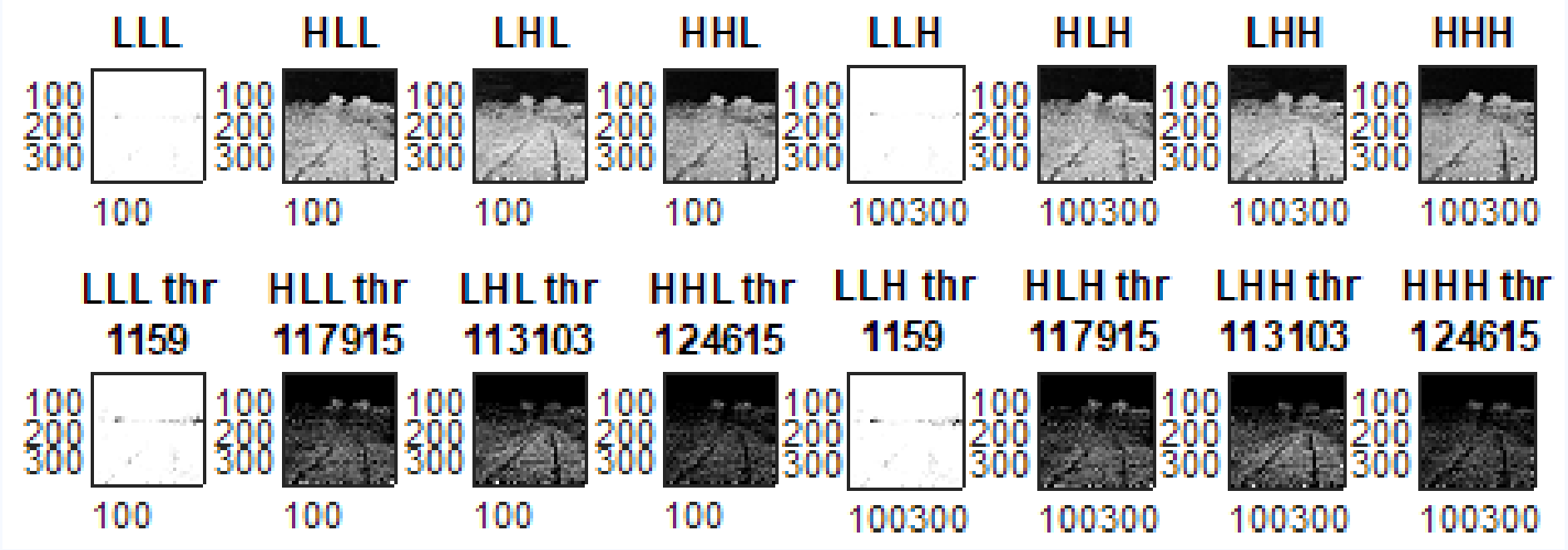
1.2 *WITH GLOBAL THRESHOLDING*

Original

Ratio: %2.6429
MSE: 33.1961 | PSNR: 32.9199

COEFFS

1.2 WITH GLOBAL THRESHOLDING



RESULT

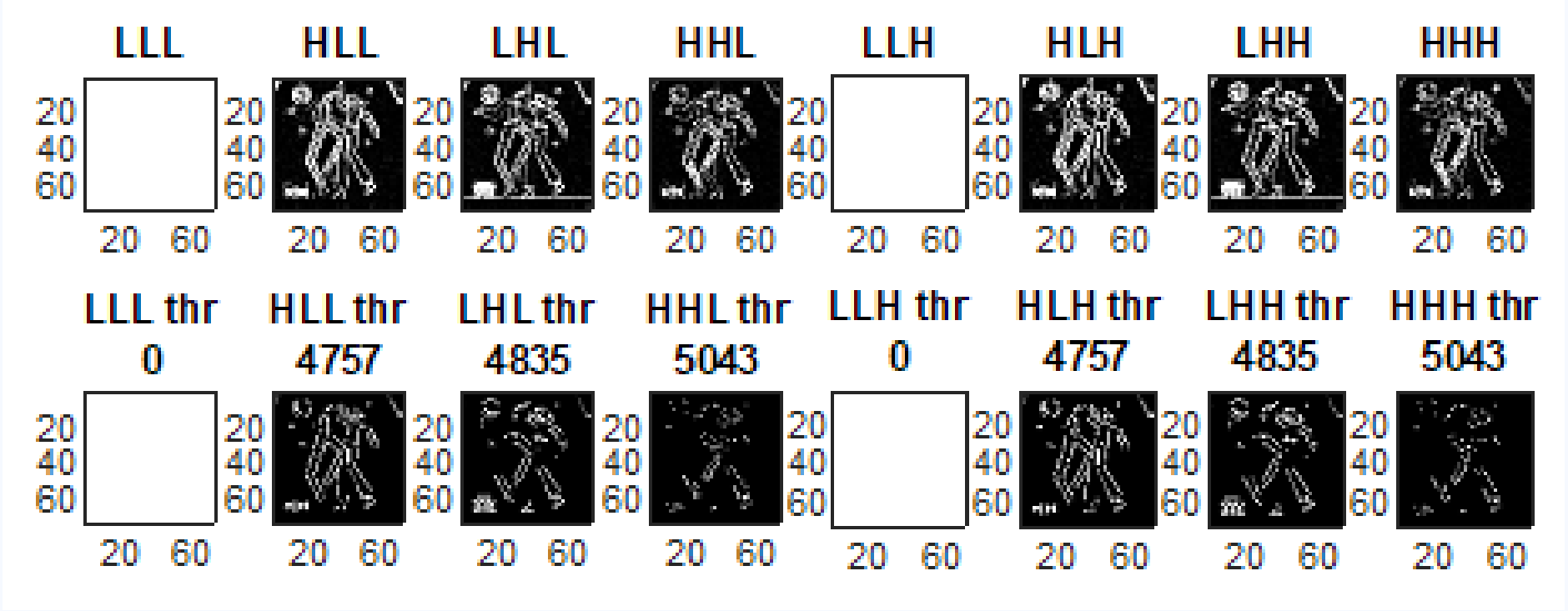
1.2 *WITH GLOBAL THRESHOLDING*

Original

Ratio: %2.9407
MSE: 12.1942 | PSNR: 37.2693

COEFFS

1.2 WITH GLOBAL THRESHOLDING



RESULT

1.2 *WITH GLOBAL THRESHOLDING*

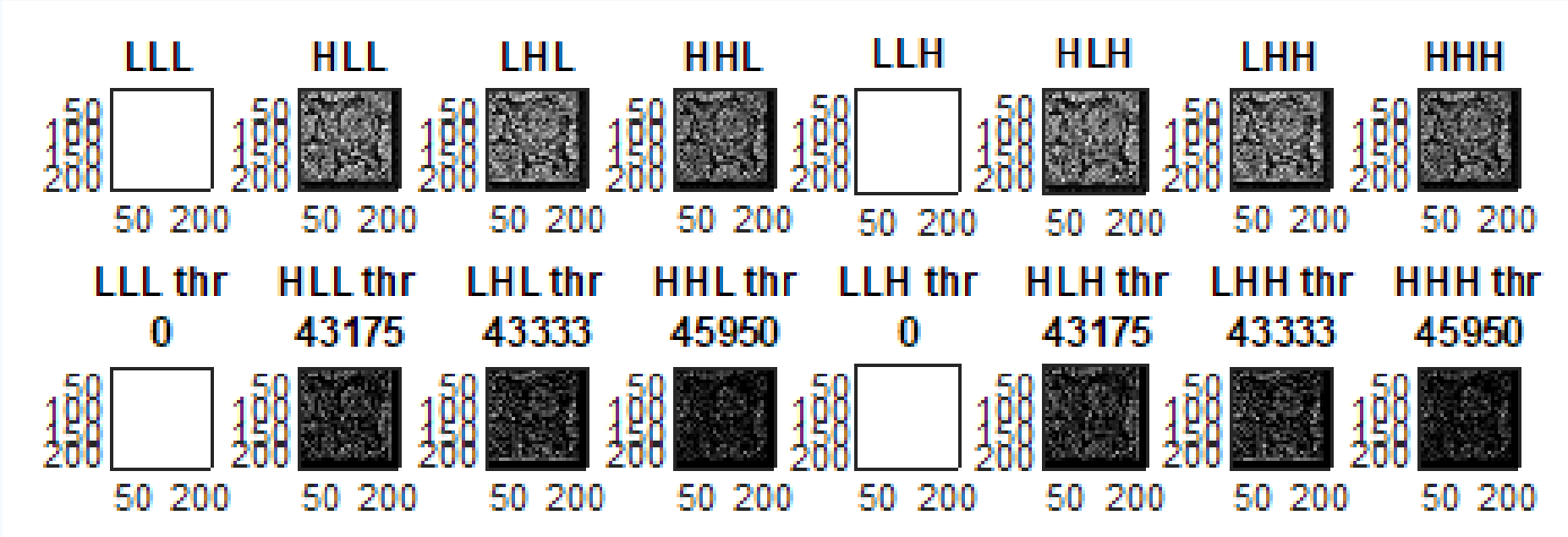
Original

Ratio: %2.7591

MSE: 21.3435 | PSNR: 34.8381

COEFFS

1.2 WITH GLOBAL THRESHOLDING



II. 2

DAUBECHIES 3-L

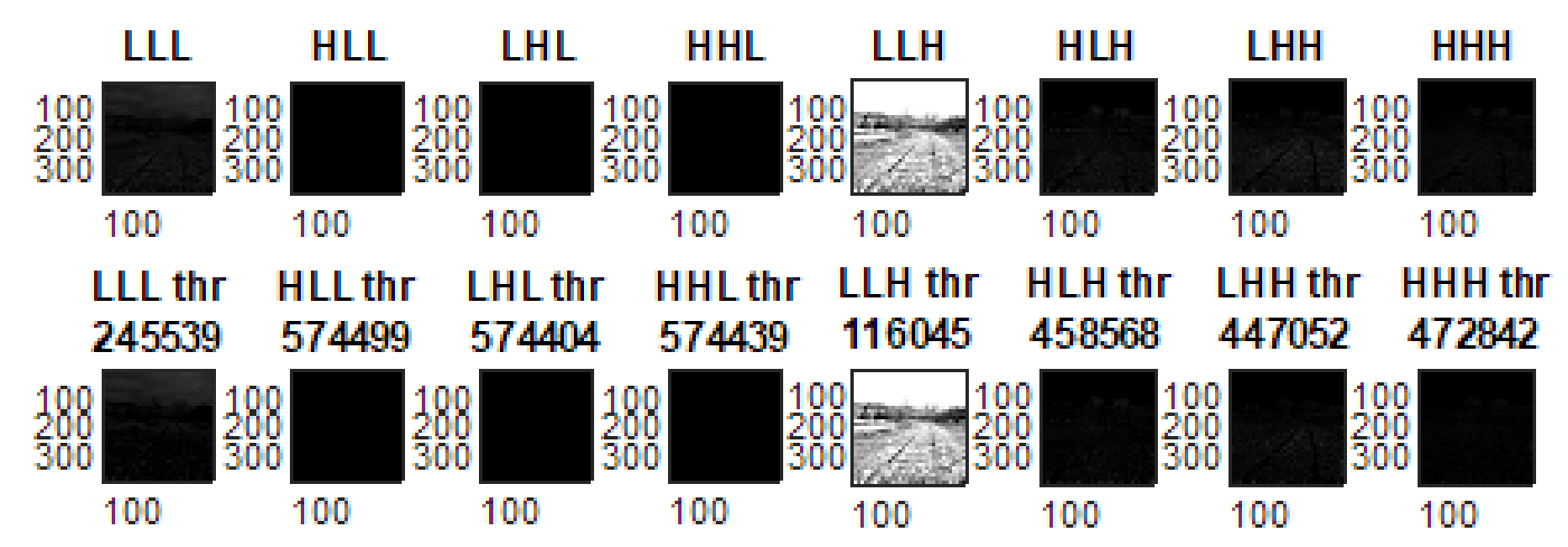
1. Threshold = local
2. Threshold = 100 (global)

RESULT

2.1 *WITH LOCAL THRESHOLDING*

COEFFS

2.1 WITH LOCAL THRESHOLDING



RESULT

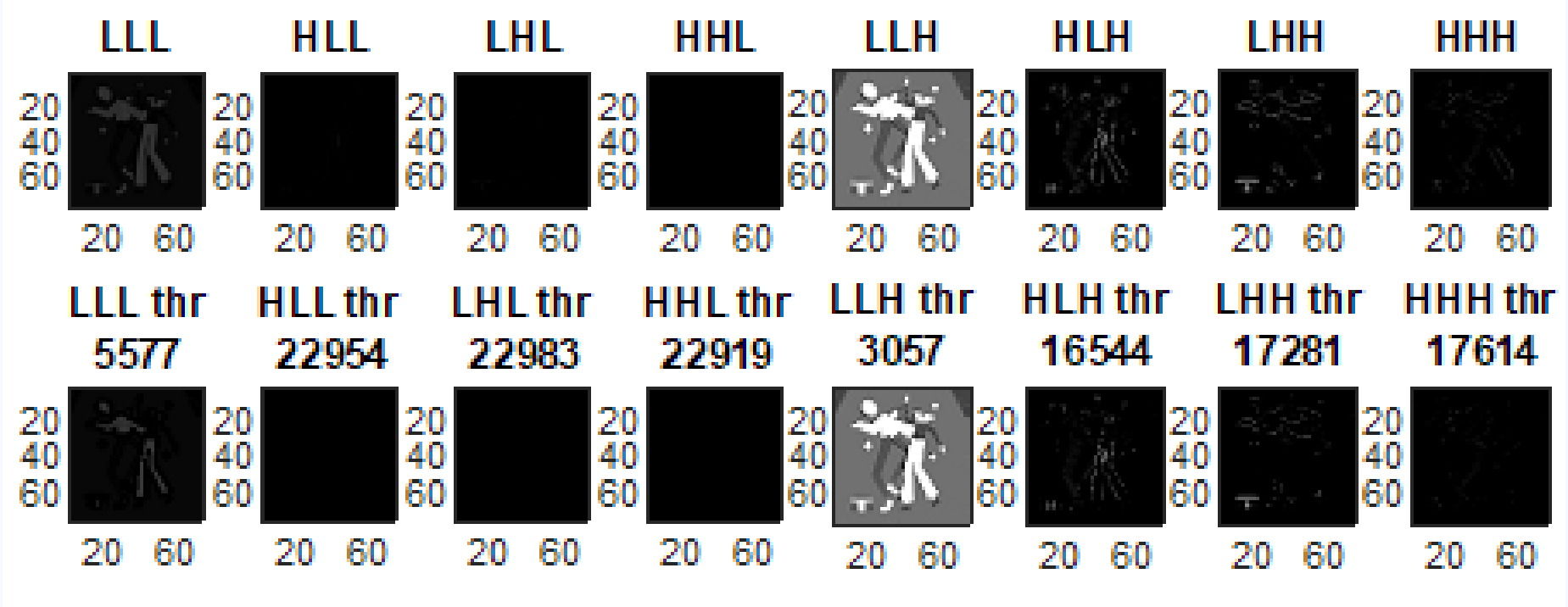
2.1 *WITH LOCAL THRESHOLDING*

Original

Ratio: %12.9534
MSE: 0.27474 | PSNR: 53.7416

COEFFS

2.1 WITH LOCAL THRESHOLDING



COEFFS

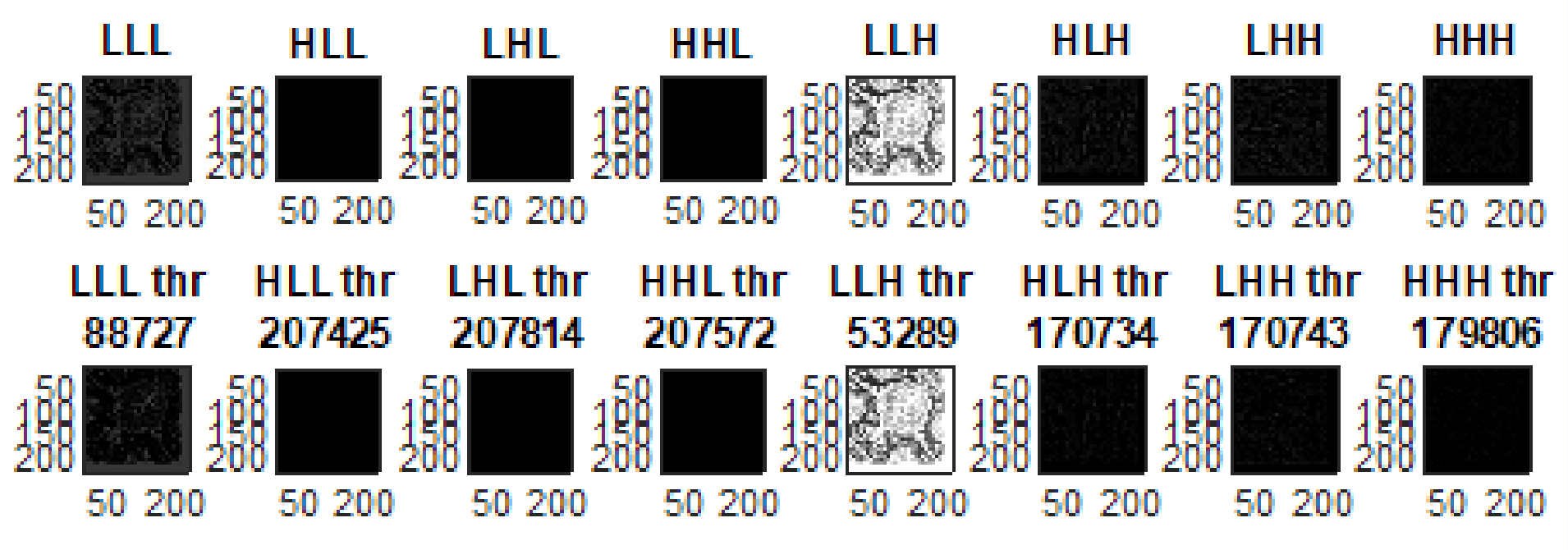
2.1 WITH LOCAL THRESHOLDING

Original

Ratio: %13.3948
MSE: 0.41226 | PSNR: 51.9791

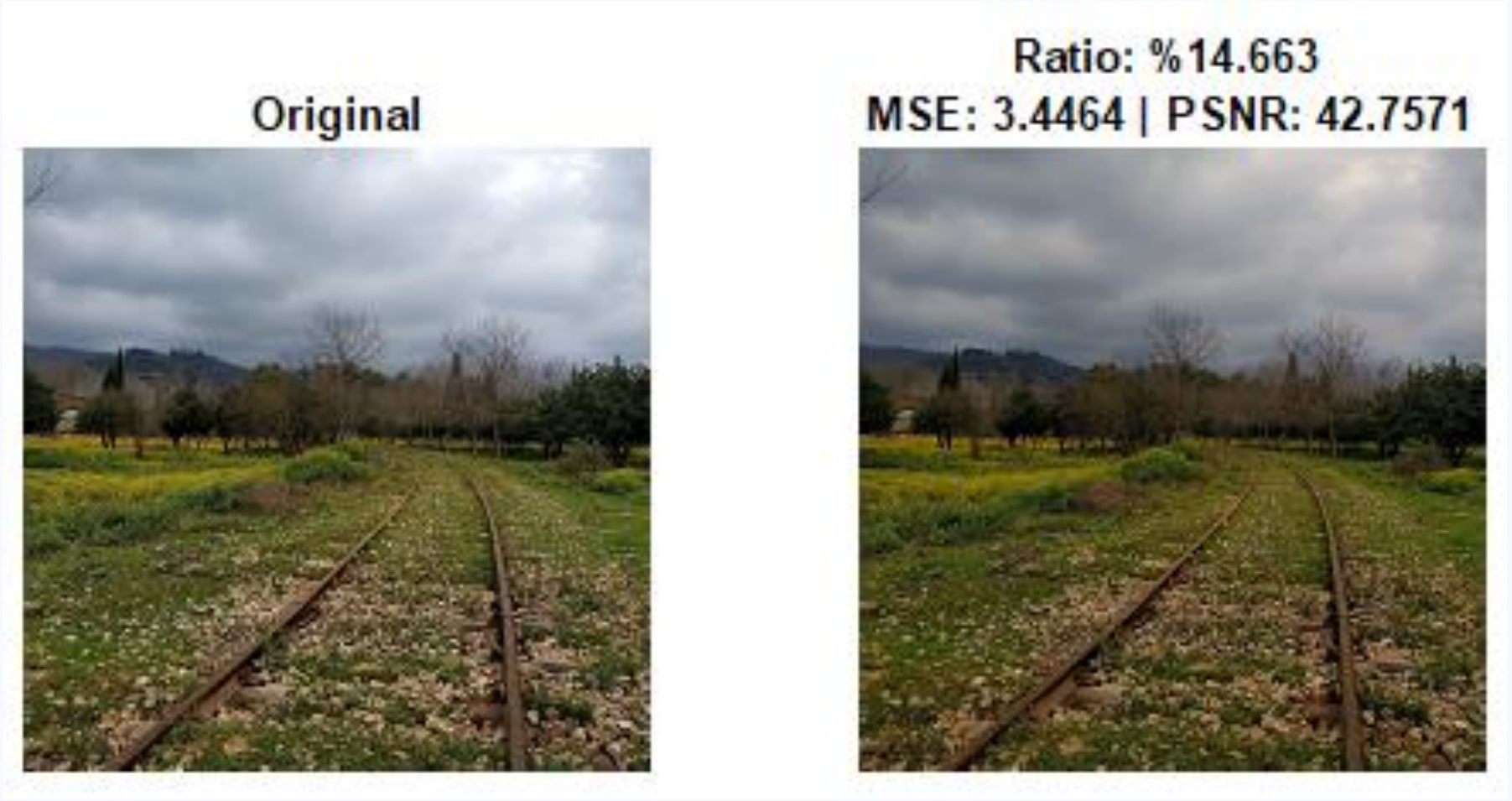
COEFFS

2.1 WITH LOCAL THRESHOLDING



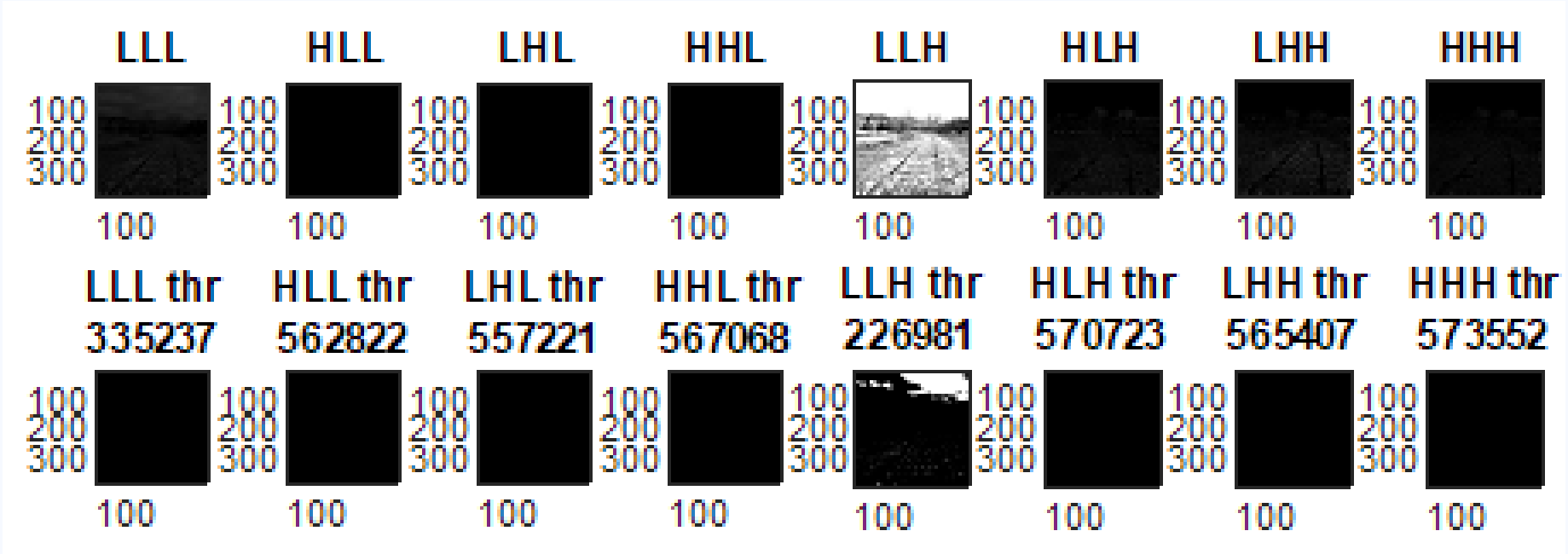
RESULT

2.2 WITH GLOBAL THRESHOLDING



COEFFS

2.2 WITH GLOBAL THRESHOLDING



RESULT

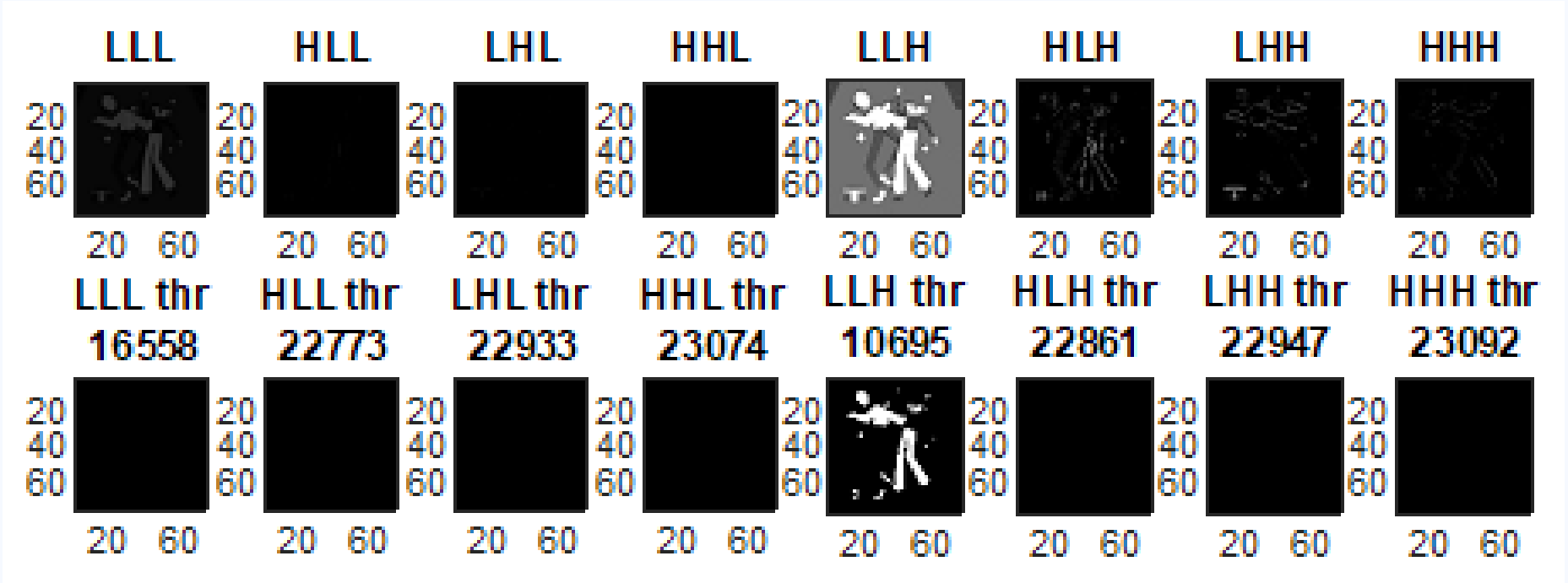
2.2 WITH GLOBAL THRESHOLDING

Original

Ratio: %16.5707
MSE: 1.6135 | PSNR: 46.0531

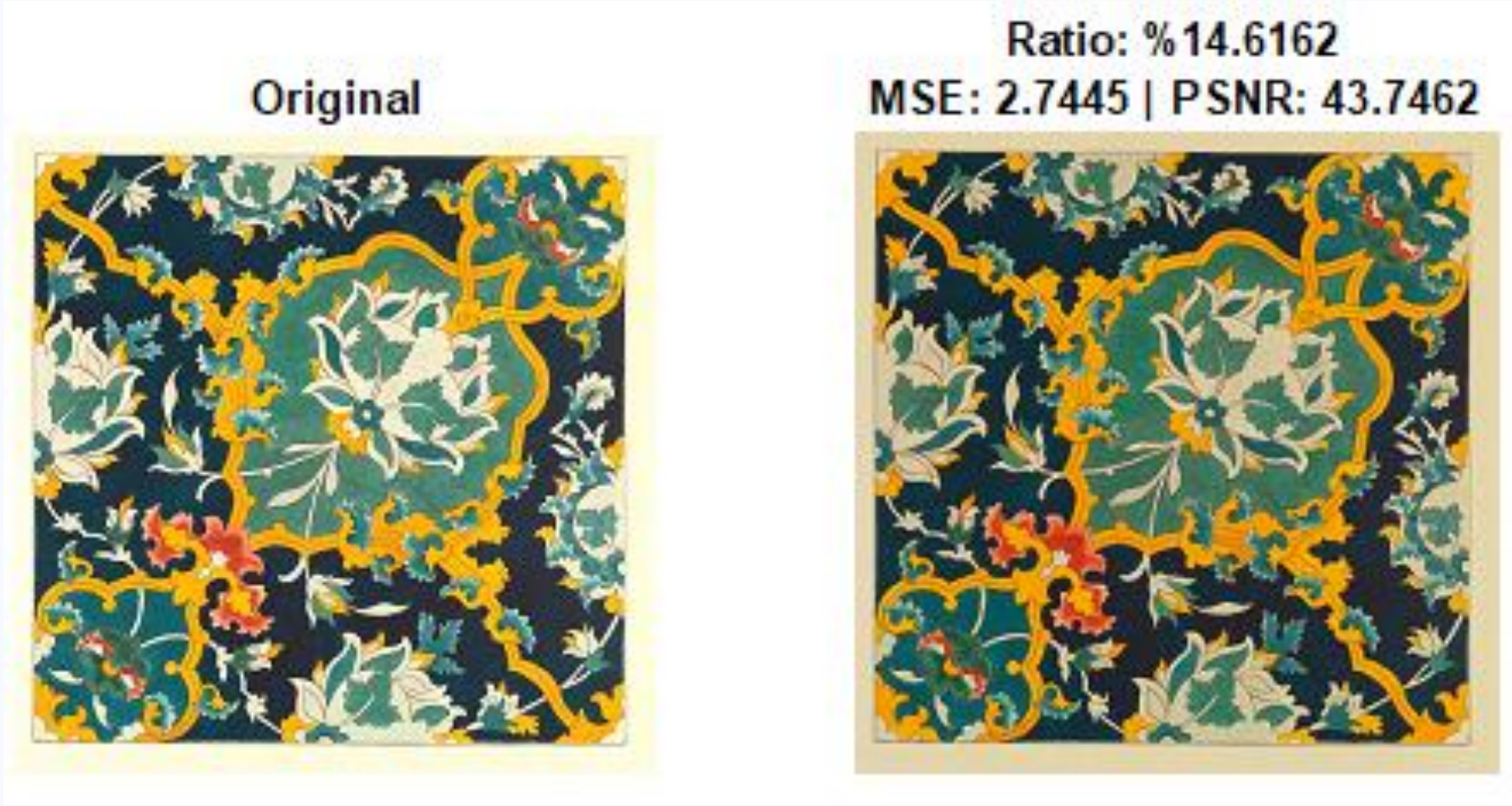
COEFFS

2.2 WITH GLOBAL THRESHOLDING



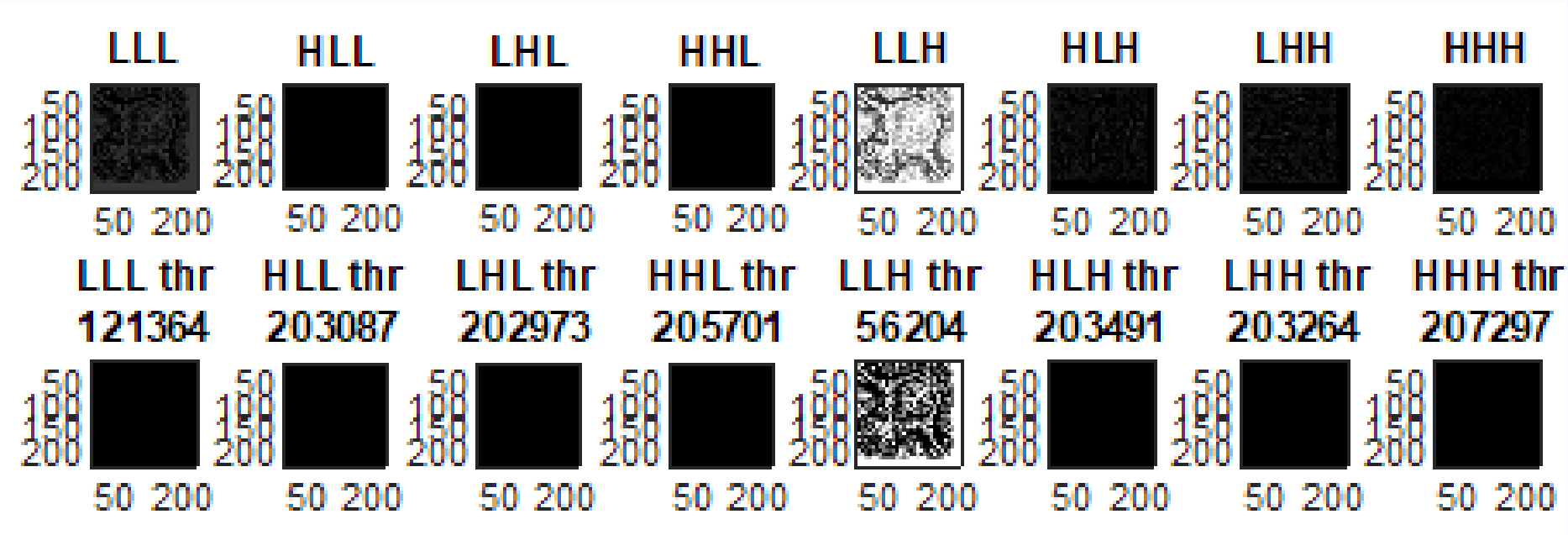
RESULT

2.2 WITH GLOBAL THRESHOLDING



COEFFS

2.2 WITH GLOBAL THRESHOLDING



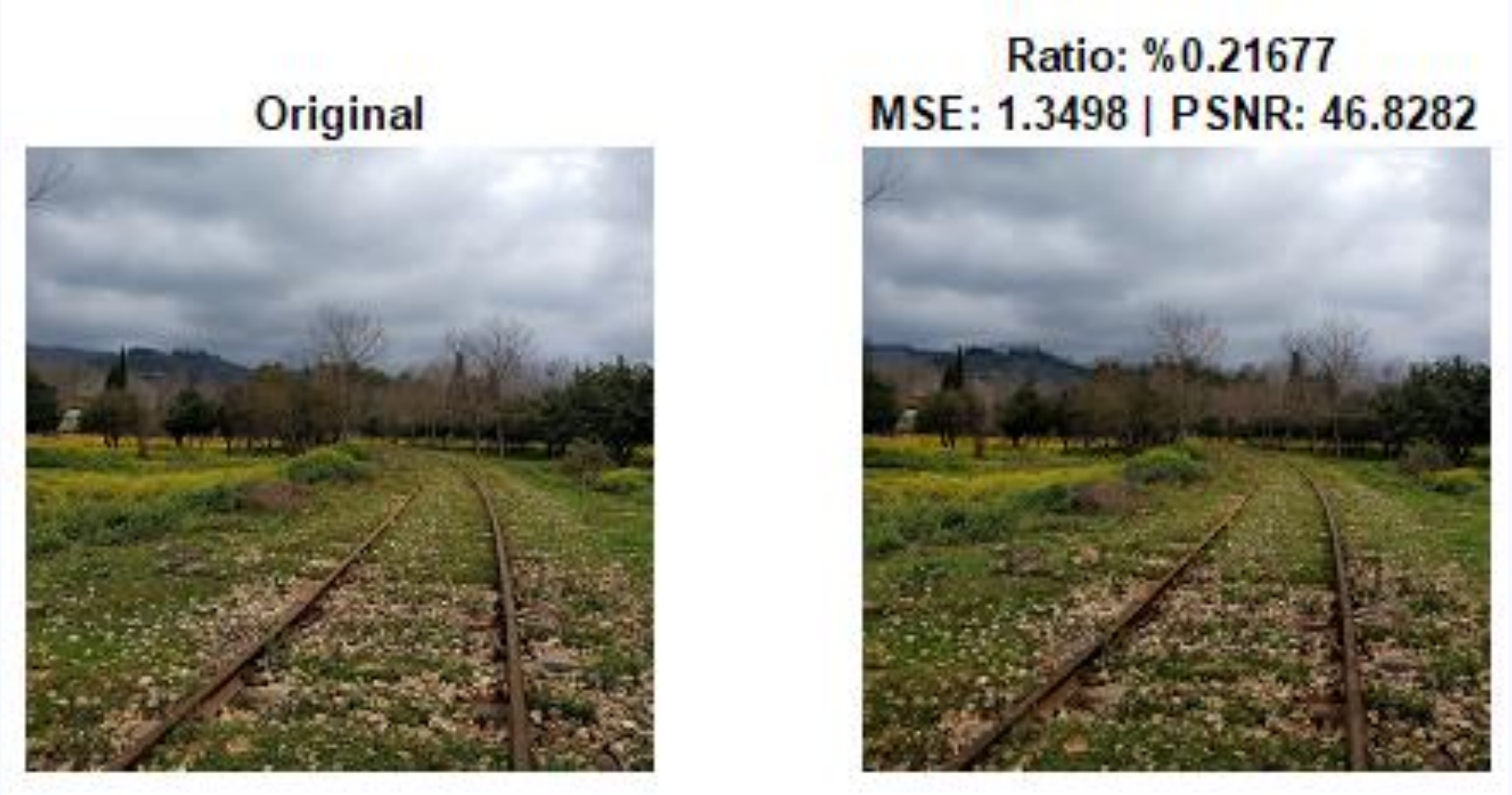
II. 3

HAAR 5-L

1. Threshold = local
2. Threshold = 100 (global)

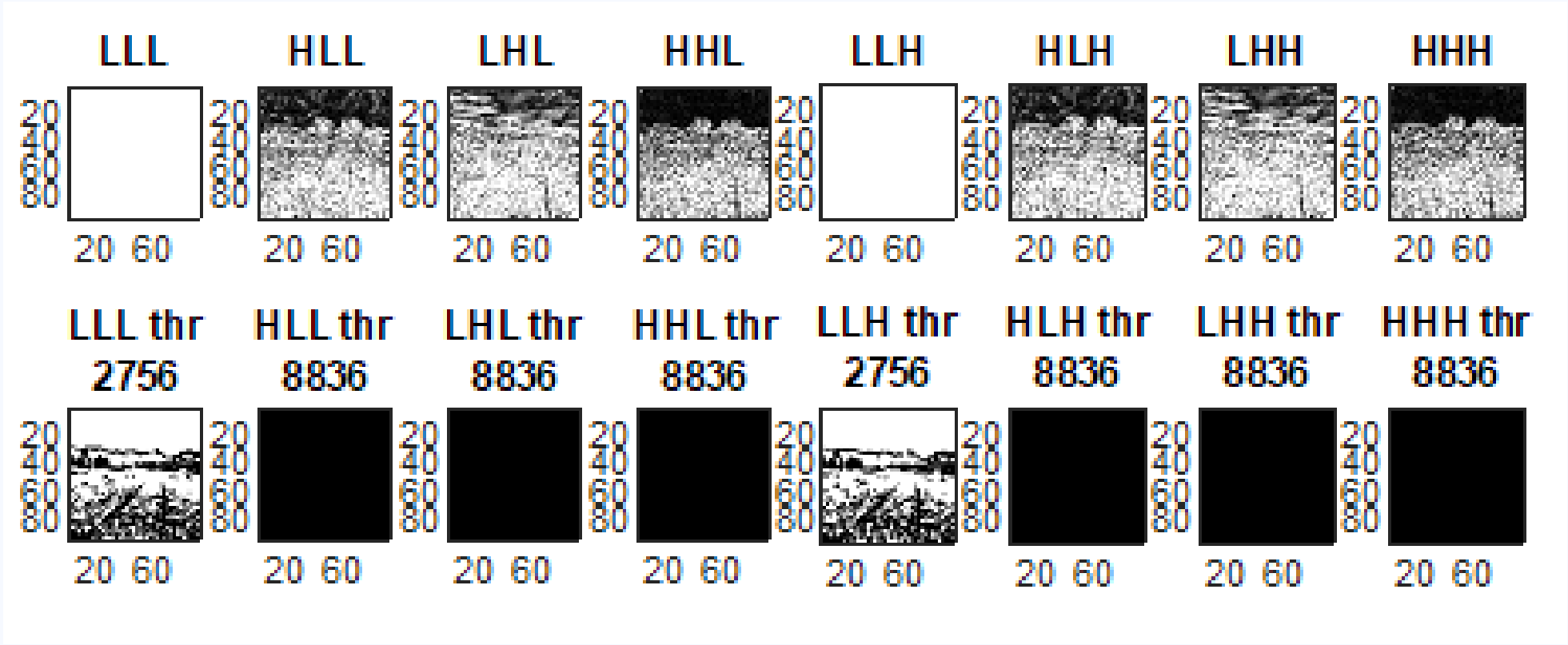
RESULT

3.1 *WITH LOCAL THRESHOLDING*



COEFFS.

3.1 *WITH LOCAL THRESHOLDING*

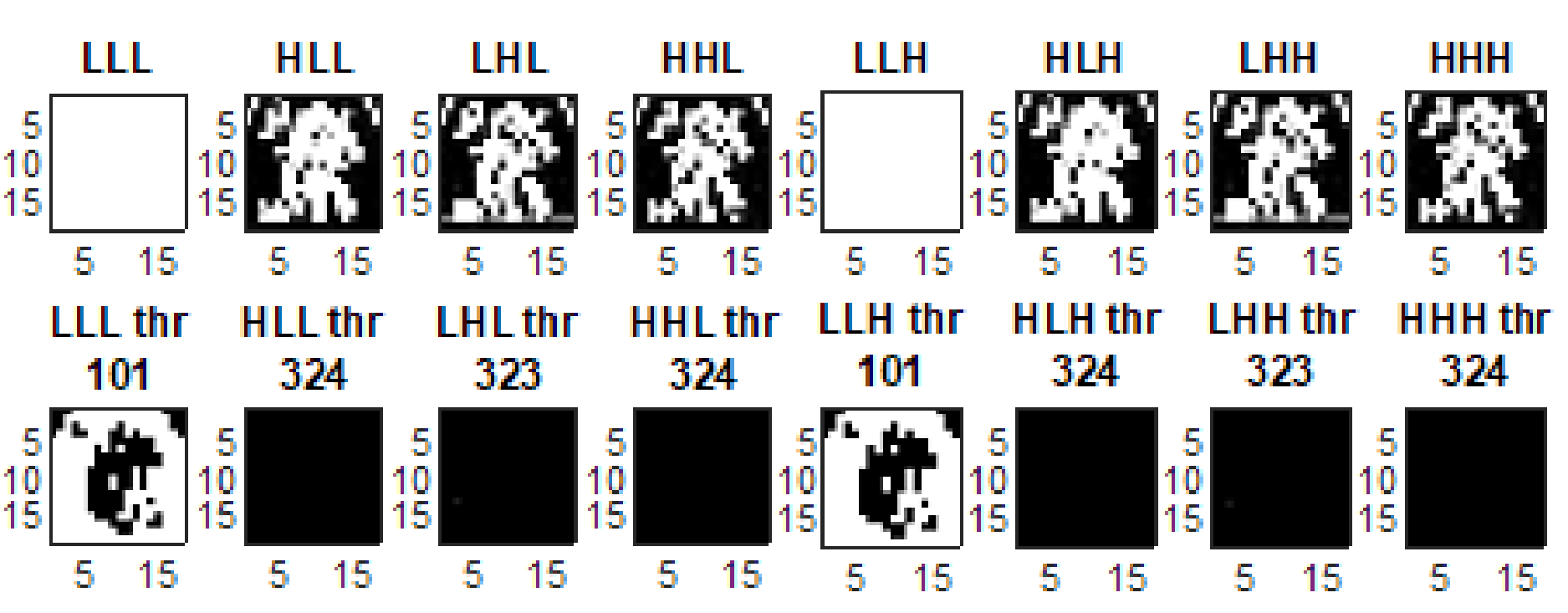


RESULT

3.1 WITH LOCAL THRESHOLDING

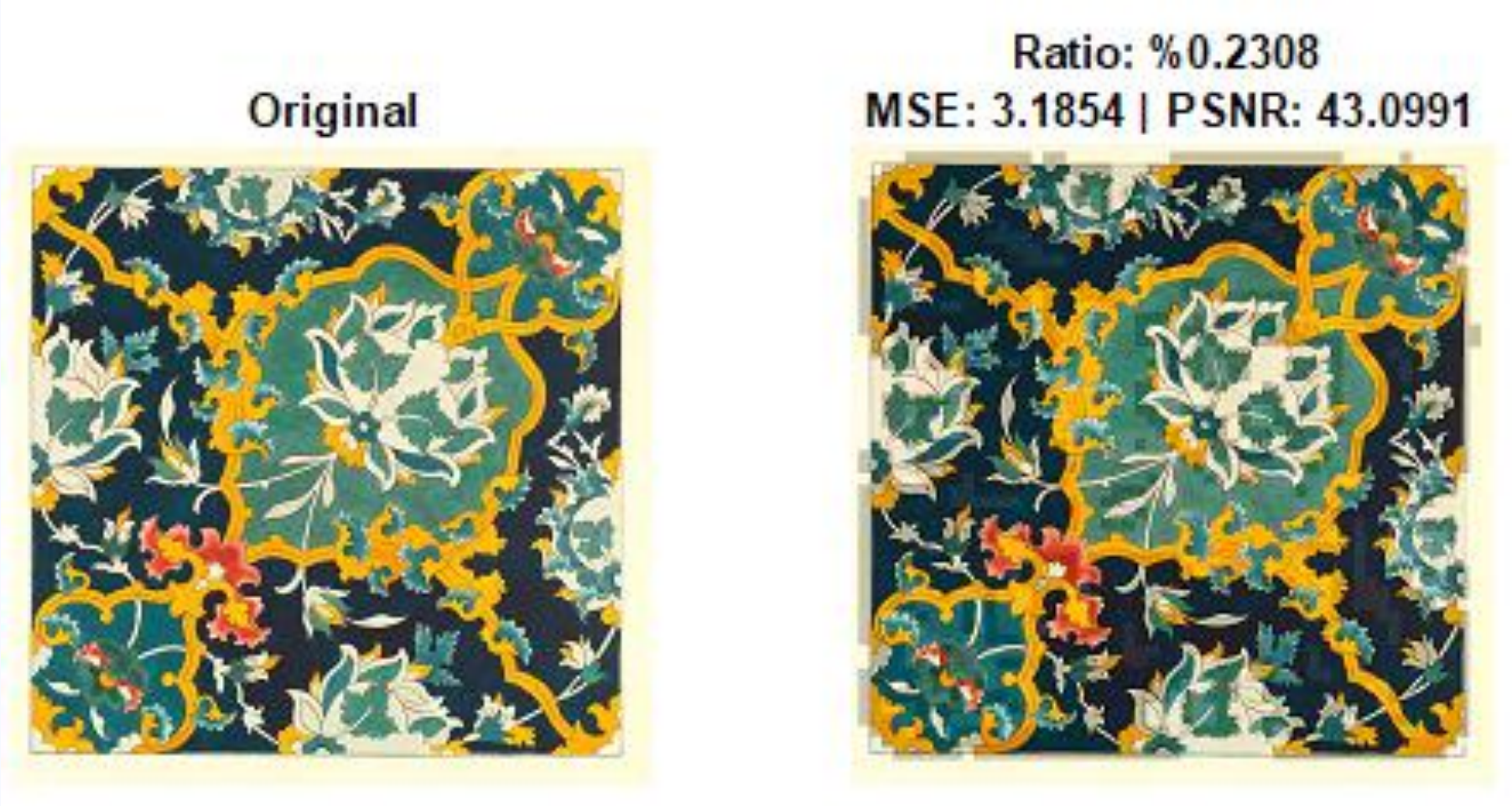
COEFFS.

3.1 *WITH LOCAL THRESHOLDING*



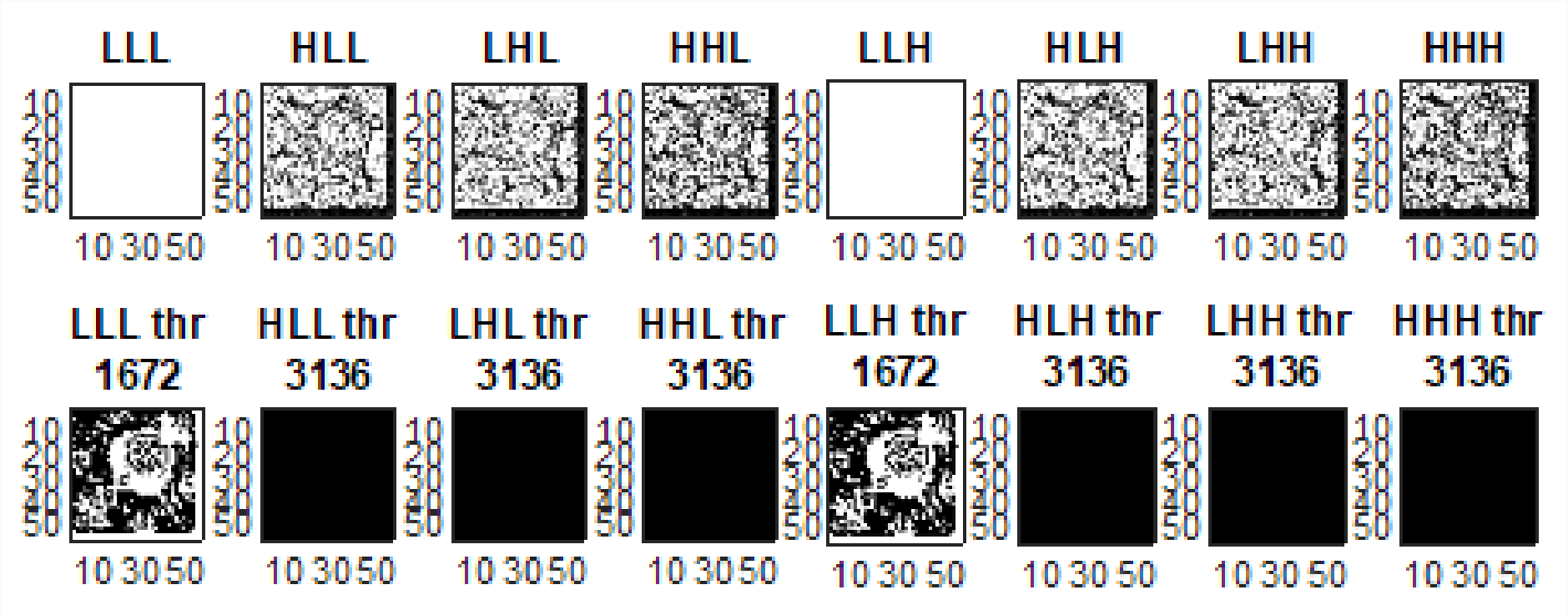
RESULT

3.1 WITH LOCAL THRESHOLDING



COEFFS.

3.1 *WITH LOCAL THRESHOLDING*



RESULT

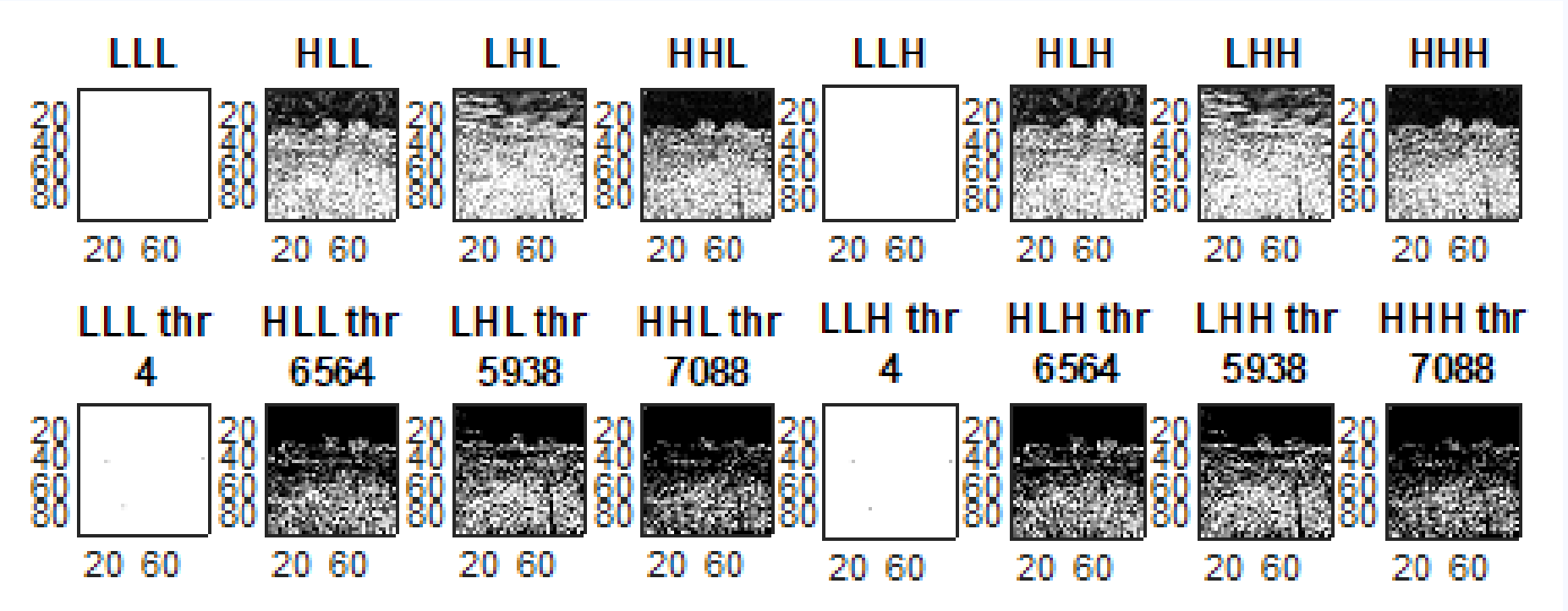
3.2 WITH GLOBAL THRESHOLDING

Original

Ratio: %0.14514
MSE: 5.9422 | PSNR: 40.3913

COEFFS.

3.2 WITH GLOBAL THRESHOLDING

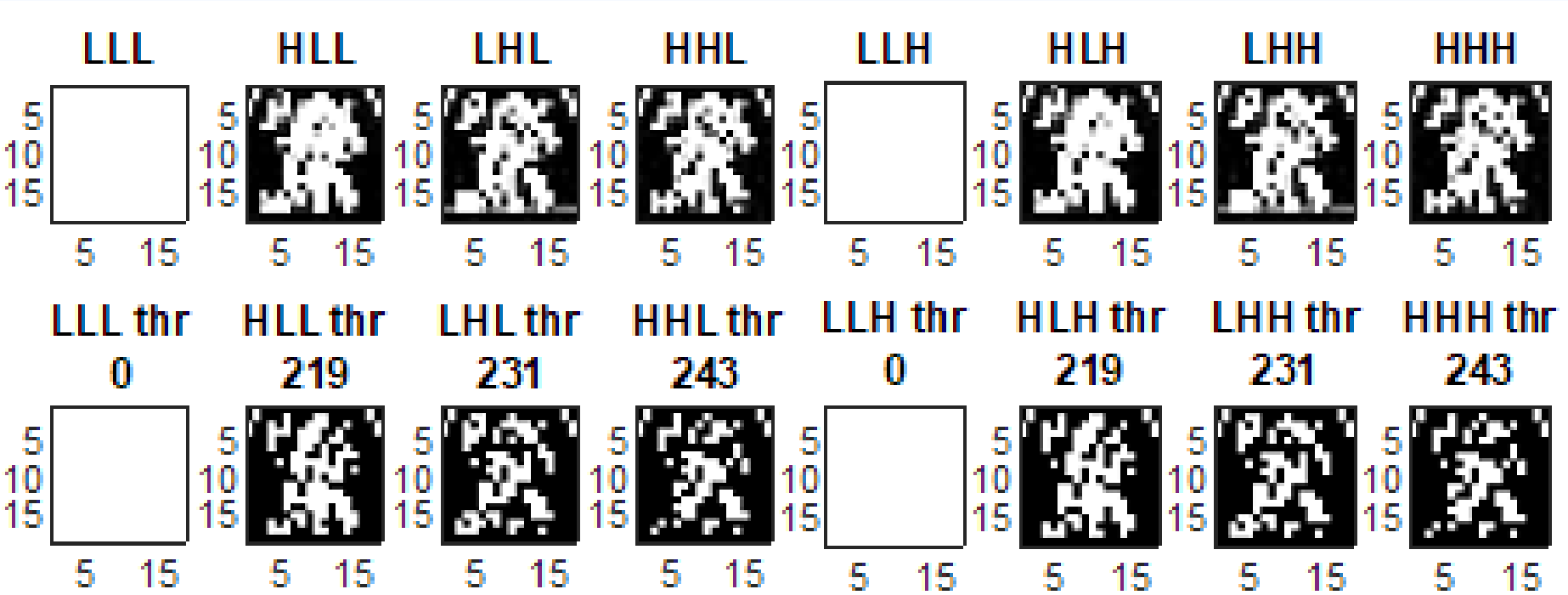


RESULT

3.2 *WITH GLOBAL THRESHOLDING*

COEFFS.

3.2 *WITH GLOBAL THRESHOLDING*



RESULT

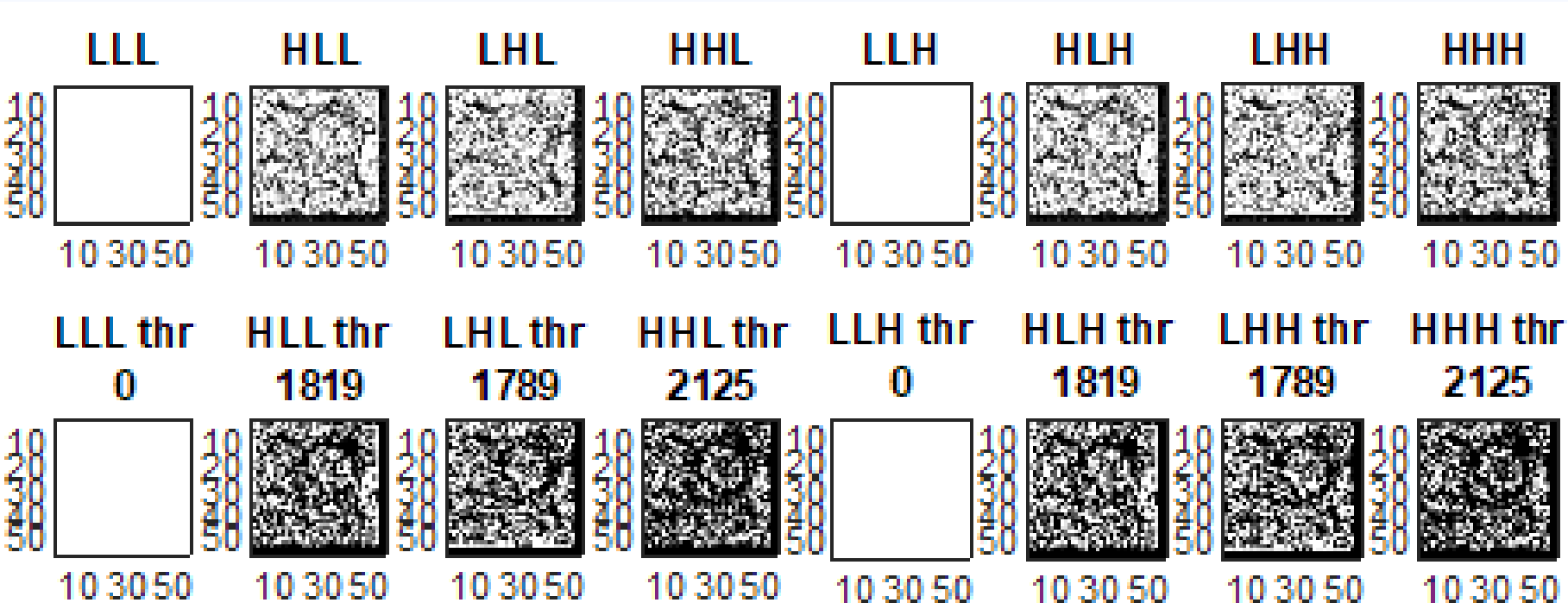
3.2 *WITH GLOBAL THRESHOLDING*

Original

Ratio: %0.11942
MSE: 17.5141 | PSNR: 35.6969

COEFFS.

3.2 WITH GLOBAL THRESHOLDING



II. 4

DAUBECHIES 5-L

1. Threshold = local
2. Threshold = 100 (global)

RESULT

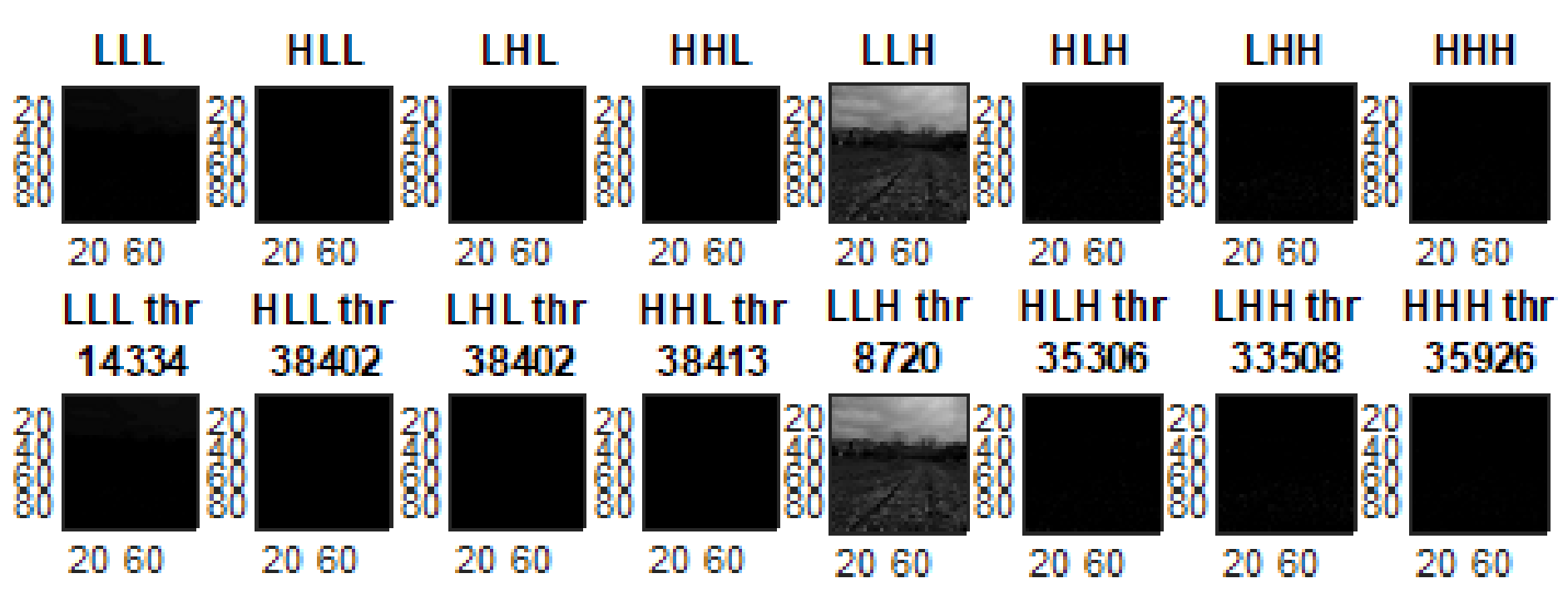
4.1 WITH LOCAL THRESHOLDING

Original

Ratio: %0.90004
MSE: 0.0018664 | PSNR: 75.4208

COEFFS.

4.1 *WITH LOCAL THRESHOLDING*

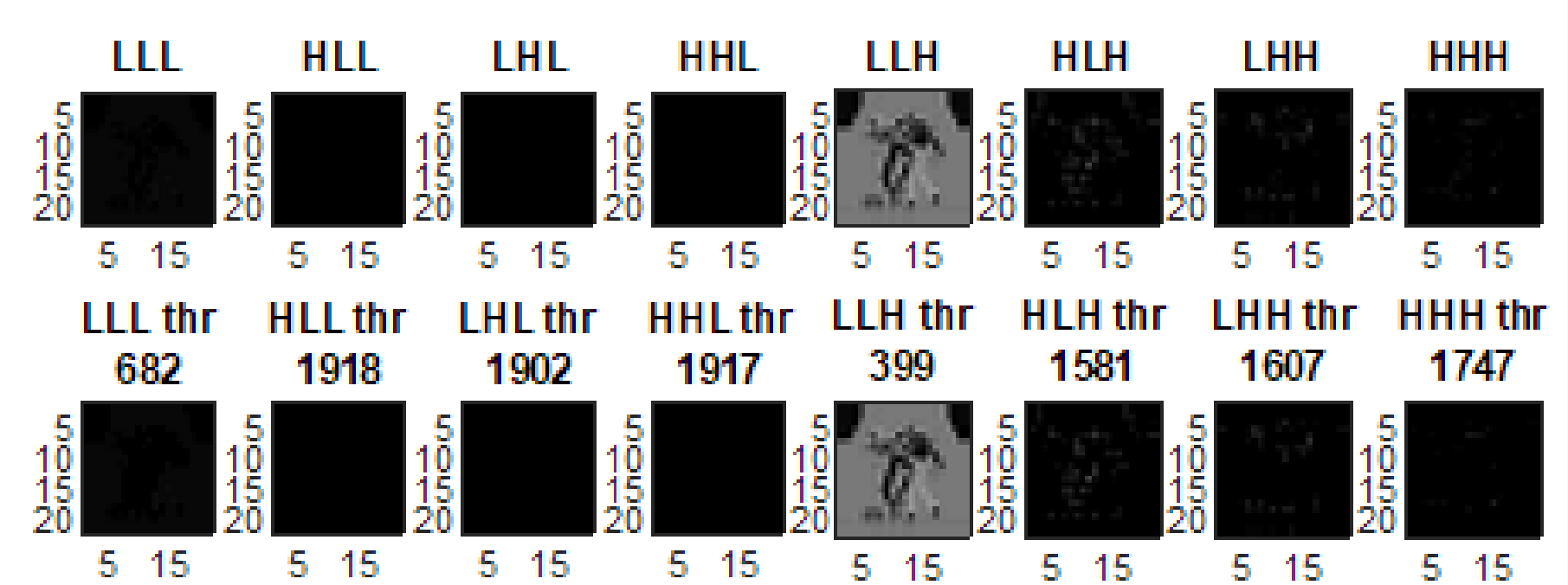


RESULT

4.1 WITH LOCAL THRESHOLDING

COEFFS.

4.1 *WITH LOCAL THRESHOLDING*



RESULT

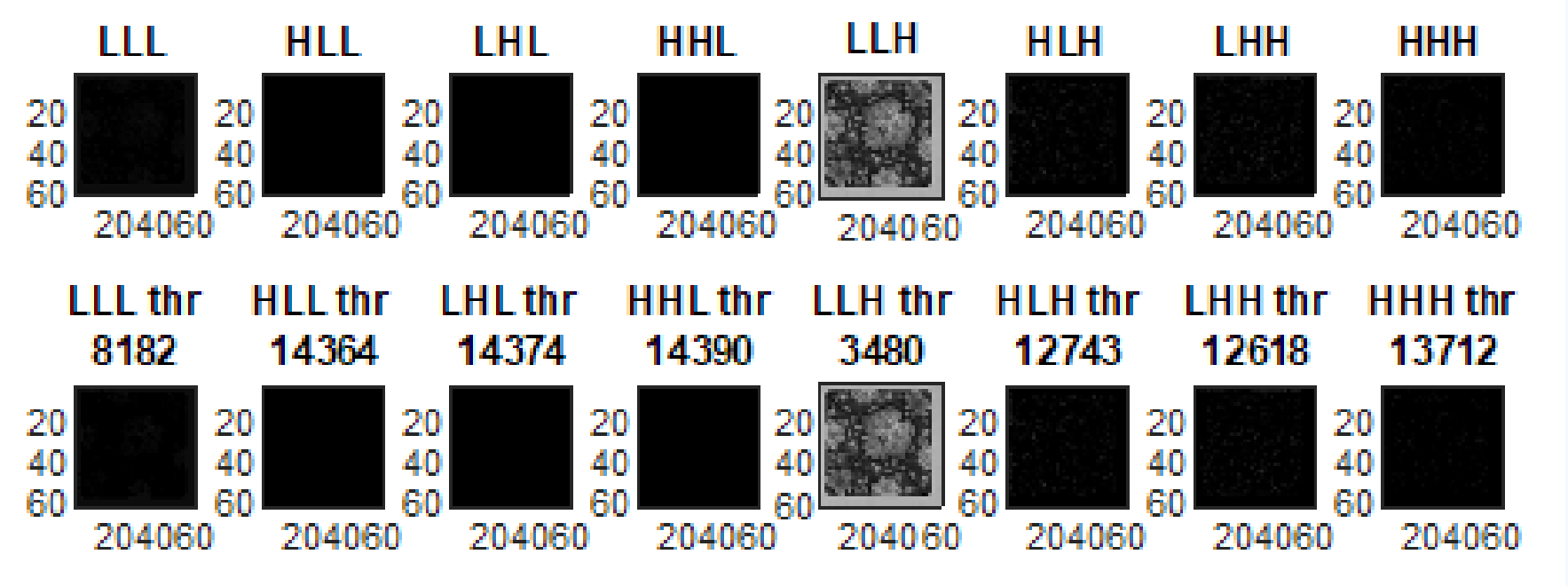
4.1 WITH LOCAL THRESHOLDING

Original

Ratio: %0.97758
MSE: 0.00030974 | PSNR: 83.2208

COEFFS.

4.1 WITH LOCAL THRESHOLDING



RESULT

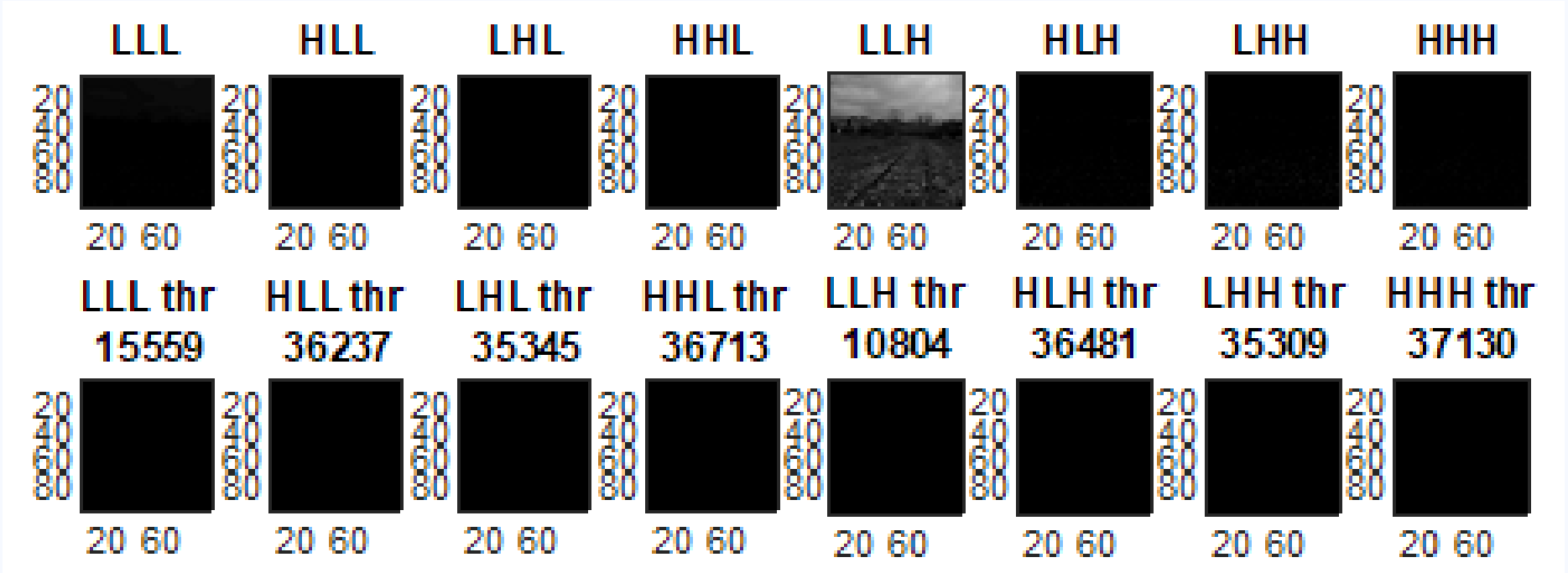
4.2 WITH GLOBAL THRESHOLDING

Original

Ratio: %0.90214
MSE: 0.014976 | PSNR: 66.3769

COEFFS.

4.2 WITH GLOBAL THRESHOLDING

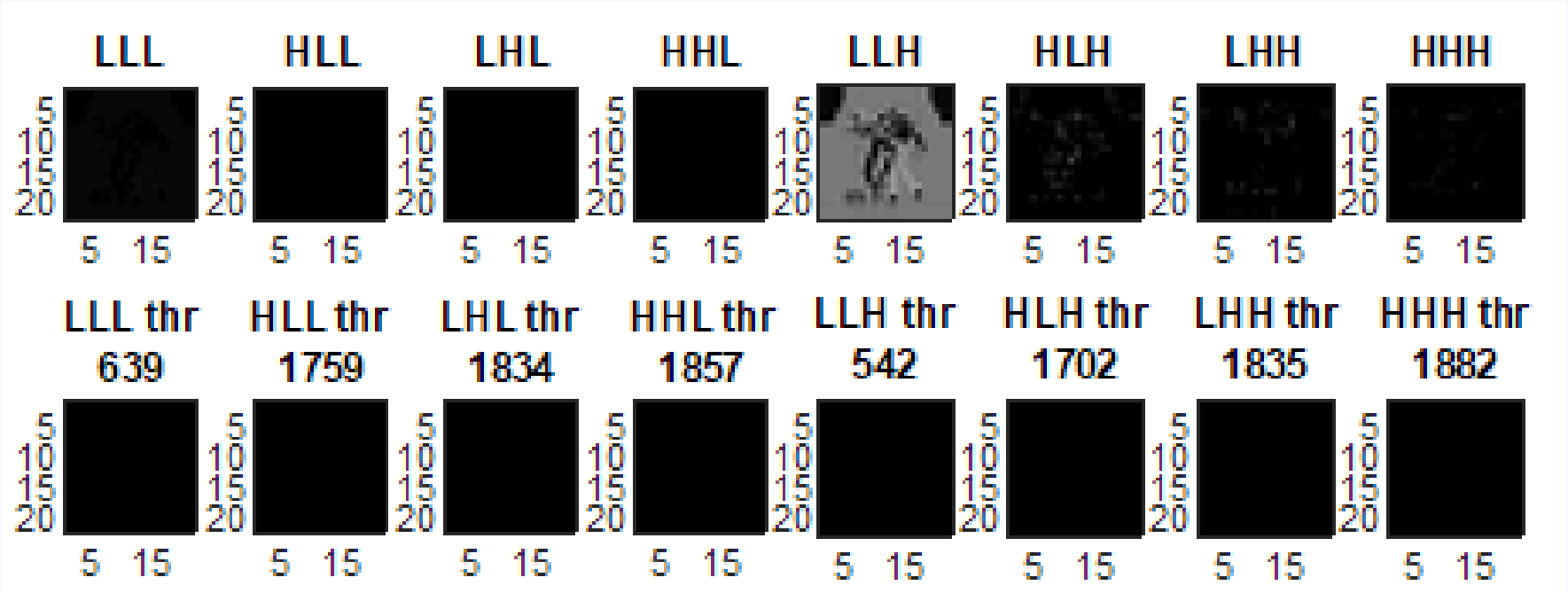


RESULT

4.2 WITH GLOBAL THRESHOLDING

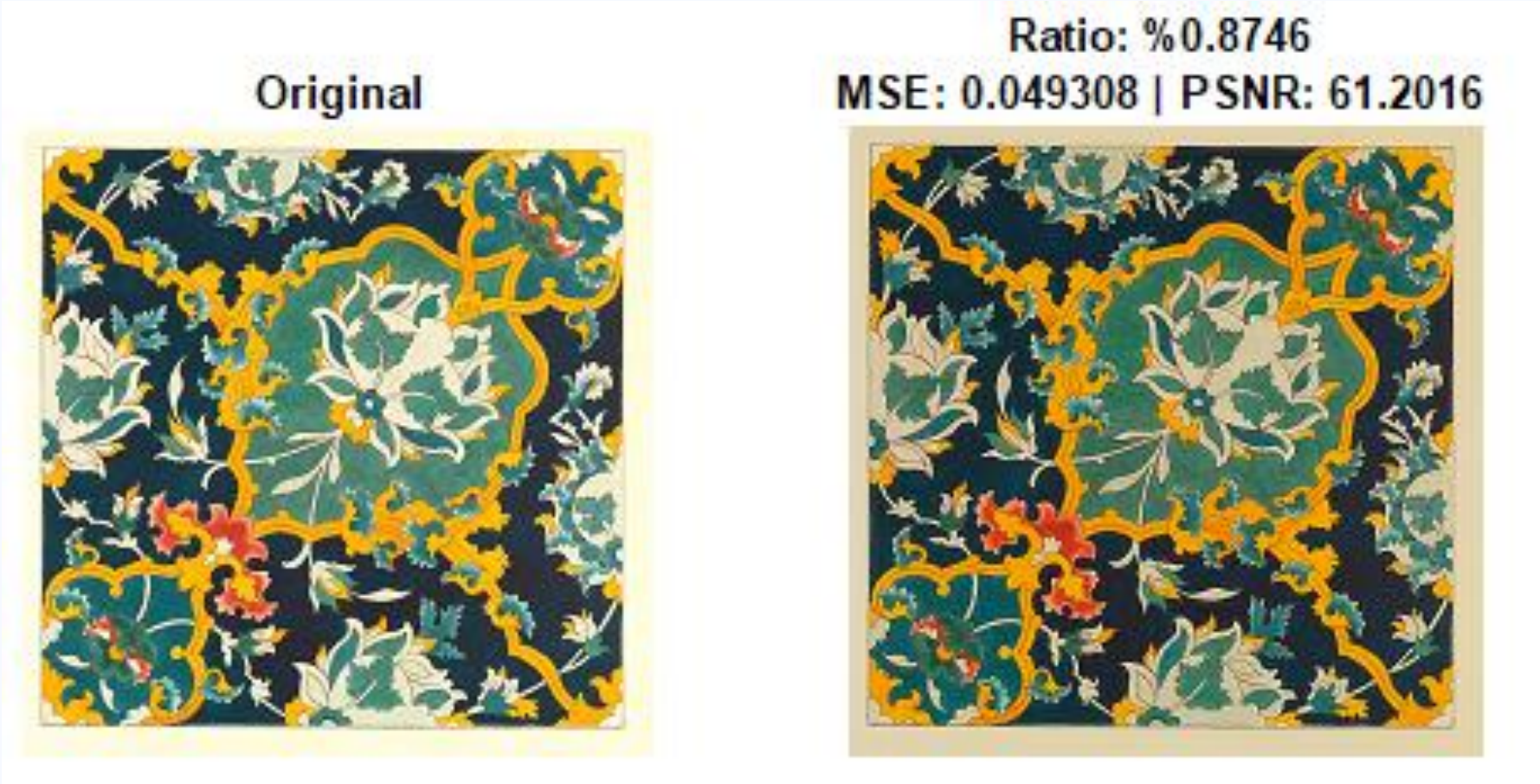
COEFFS.

4.2 *WITH GLOBAL THRESHOLDING*



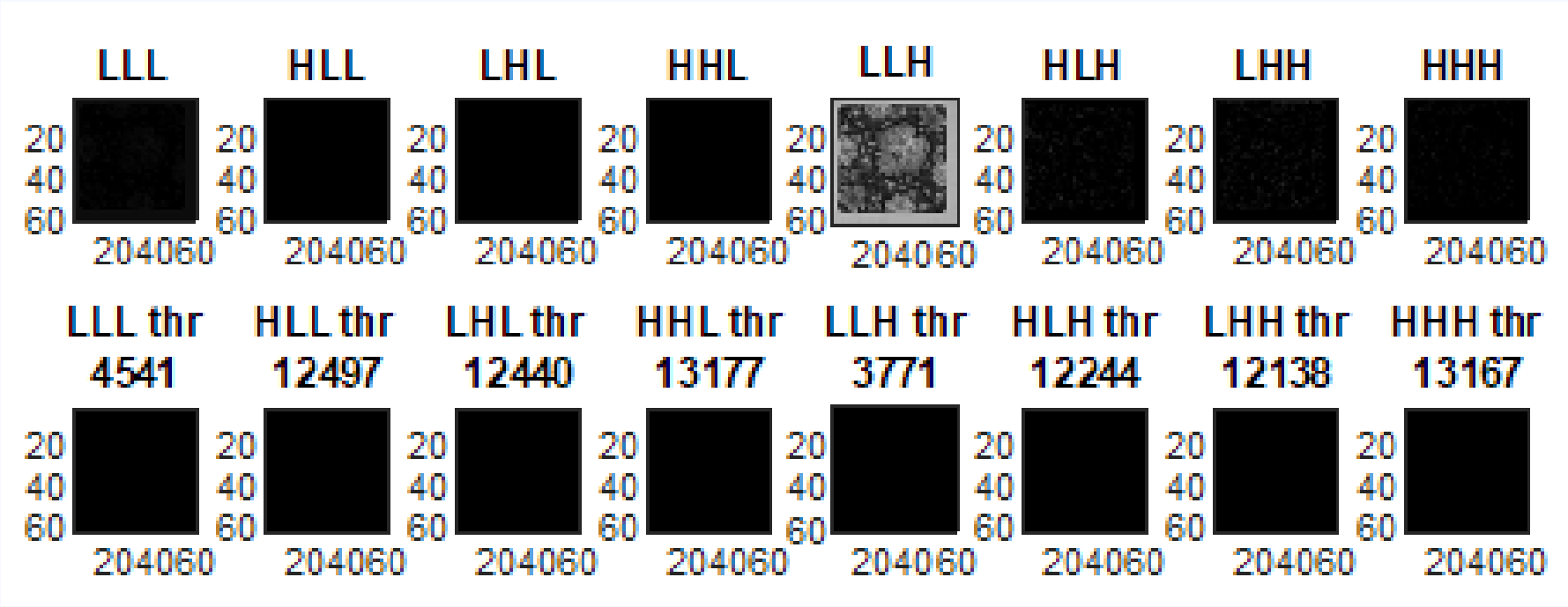
RESULT

4.2 *WITH GLOBAL THRESHOLDING*



COEFFS.

4.2 WITH GLOBAL THRESHOLDING



II. 5

CUMULATIVE RESULTS

H-3	***rail.jpg***		***disco.jpg***		***circ.jpg***	
THR type	*local*	*global*	*local*	*global*	*local*	*global*
C. Ratio	3.57	2.64	3.36	2.94	3.47	2.75
MSE	11.16	33.19	2.02	12.19	7.63	21.34
PSNR	37.65	32.91	45.07	37.26	39.3	34.83
DB-3	***rail.jpg***		***disco.jpg***		***circ.jpg***	
THR type	*local*	*global*	*local*	*global*	*local*	*global*
C. Ratio	12.82	14.66	12.95	16.57	13.39	14.61
MSE	0.69	3.44	0.27	1.61	0.41	2.74
PSNR	49.7	42.75	53.74	46.05	51.97	43.74

H-5	***rail.jpg***		***disco.jpg***		***circ.jpg***	
THR type	*local*	*global*	*local*	*global*	*local*	*global*
C. Ratio	0.21	0.14	0.21	0.13	0.23	0.11
MSE	1.34	5.94	2.22	13.04	3.18	17.51
PSNR	46.82	40.39	44.65	36.97	43.09	35.69
DB-5	***rail.jpg***		***disco.jpg***		***circ.jpg***	
THR type	*local*	*global*	*local*	*global*	*local*	*global*
C. Ratio	0.9	0.9	1.18	1.21	0.97	0.87
MSE	0.001	0.01	0.003	0.07	0.0003	0.04
PSNR	75.42	66.37	73.17	59.34	83.22	61.2

III.

WATERMARKING

Thoughts, results and details on the second main task: Image watermarking using DWT

HIGHLIGHTS

Method

- Read base and watermark images
- Use 2-D DWT (Haar and Daubechies) to decompose images
- Recompose watermark and base images together
- Save result
- De-watermark the result from previous step

Challenges

- Proper Q-K coefficients pair must be found experimentally
- Watermark visibility rate
- Tile or upscale watermark? what if base image is smaller?
- Some representations might be difficult to distinguish on computer screens

Notes

- Images are in 1:1 ratio (square)
- Watermark image is small and mostly include lower frequencies (20kB, 100x100px)
- Base image also mostly includes lower frequencies (78kB, 576x576px)
- Both color images, but processed in grayscale

Steps

1. Haar 3-L
   - 1.1. (k, q) = (0.2, 0.009)
   - 1.2. (k, q) = (0.6, 0.009)
   - 1.3. (k, q) = (1, 0.009)
   - 1.4. (k, q) = (1.4, 0.009)
   - 1.5. (k, q) = (1.8, 0.009)
   - 1.6. (k, q) = (0.2, 0.01)
   - 1.7. (k, q) = (0.6, 0.01)
   - 1.8. (k, q) = (1, 0.01)
   - 1.9. (k, q) = (1.4, 0.01)
   - 1.10. (k, q) = (1.8, 0.01)

2. Haar 3-L
   - 2.1. (k, q) = (0.2, 0.009)
   - 2.2. (k, q) = (0.6, 0.009)
   - 2.3. (k, q) = (1, 0.009)
   - 2.4. (k, q) = (1.4, 0.009)
   - 2.5. (k, q) = (1.8, 0.009)
   - 2.6. (k, q) = (0.2, 0.01)
   - 2.7. (k, q) = (0.6, 0.01)
   - 2.8. (k, q) = (1, 0.01)
   - 2.9. (k, q) = (1.4, 0.01)
   - 2.10. (k, q) = (1.8, 0.01)

3. Results

Cover Image:

circ.jpg

1789x1789px

300dpi

24bit-depth

1.35Mb

Watermark Image:

disco.jpg

576x576px

96dpi

24bit-depth

78.7Kb

III. 1

HAAR 3-L

RESULTS

1.1 HAAR 3-L
K, Q = 0.2, 0.009

Original

Watermarked
MSE: 14503.3142 | PSNR: 6.5161

EXTRACTION

1.1 HAAR 3-L
K, Q = 0.2, 0.009

Original Watermark

Extracted Watermark
MSE: 8692.6086 | PSNR: 8.7393

RESULTS

1.1 HAAR 3-L
K, Q = 0.6, 0.009

Original

Watermarked
MSE: 3676.2338 | PSNR: 12.4768

EXTRACTION

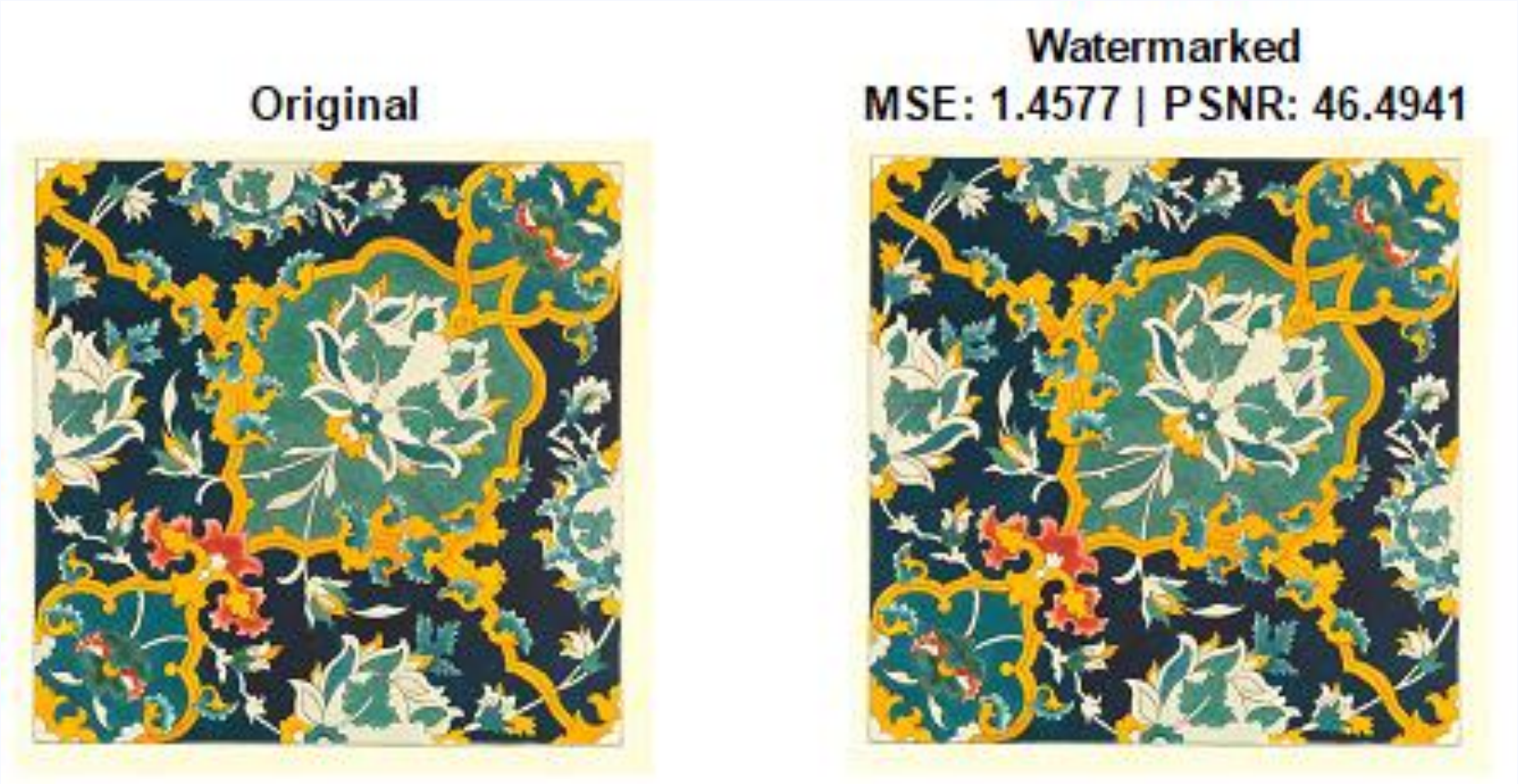
1.1 HAAR 3-L
K, Q = 0.6, 0.009

Original Watermark

Extracted Watermark
MSE: 2577.2543 | PSNR: 14.0192

RESULTS

1.1 HAAR 3-L
K, Q = 1.0, 0.009



EXTRACTION

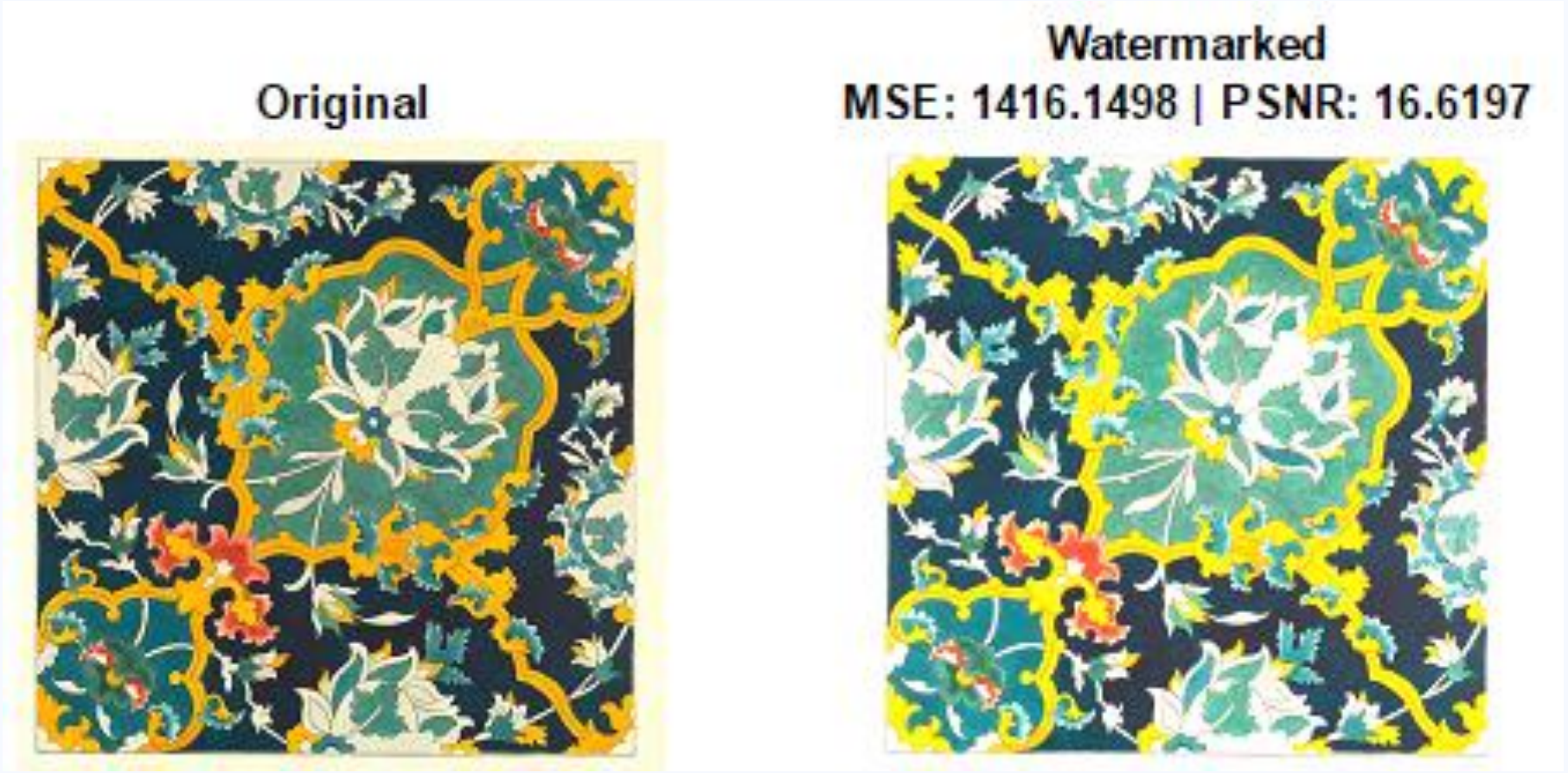
1.1 HAAR 3-L
K, Q = 1.0, 0.009

Original Watermark

Extracted Watermark
MSE: 1965.0851 | PSNR: 15.197

RESULTS

1.1 HAAR 3-L
K, Q = 1.4, 0.009



EXTRACTION

1.1 HAAR 3-L
K, Q = 1.4, 0.009

Original Watermark

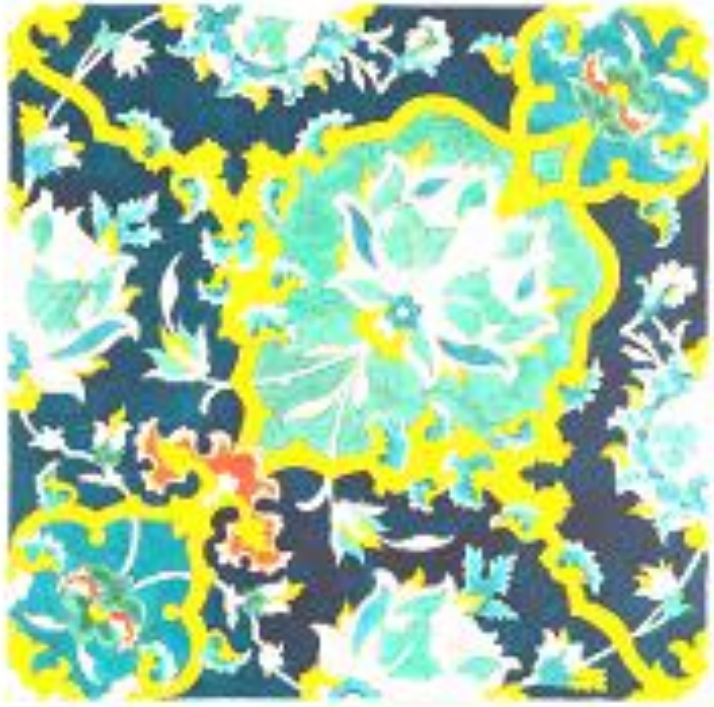
Extracted Watermark
MSE: 5915.4772 | PSNR: 10.4109

RESULTS

1.1 HAAR 3-L
K, Q = 1.8, 0.009

Original

Watermarked
MSE: 3762.985 | PSNR: 12.3755

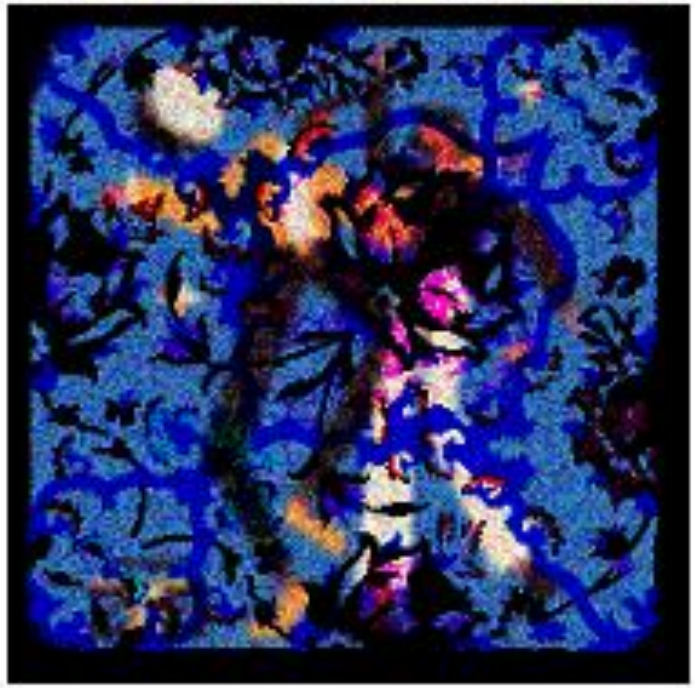


EXTRACTION

1.1 HAAR 3-L
K, Q = 1.8, 0.009

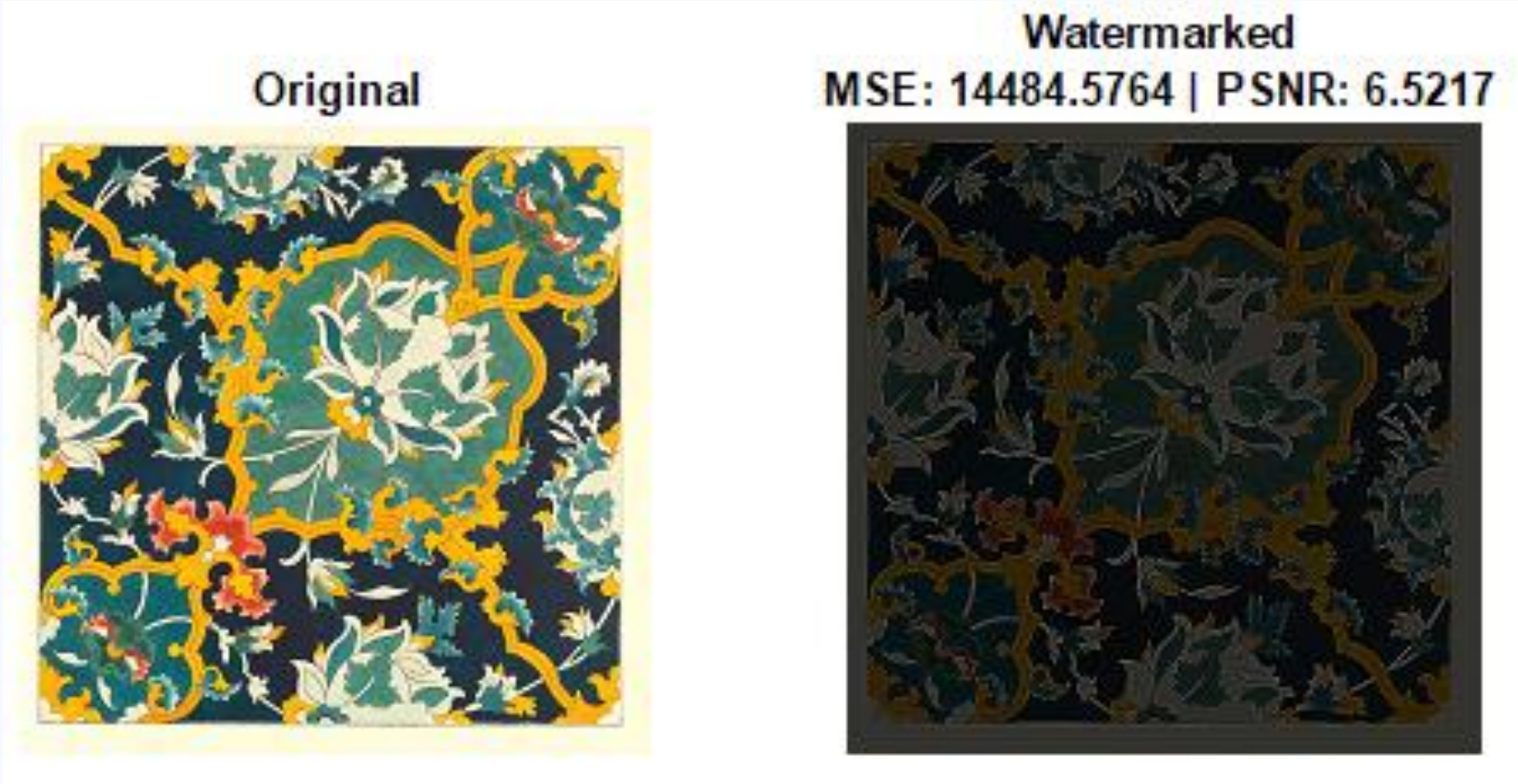
Original Watermark

Extracted Watermark
MSE: 7058.8844 | PSNR: 9.6434



RESULTS

1.2 HAAR 3-L
K, Q = 0.2, 0.01



EXTRACTION

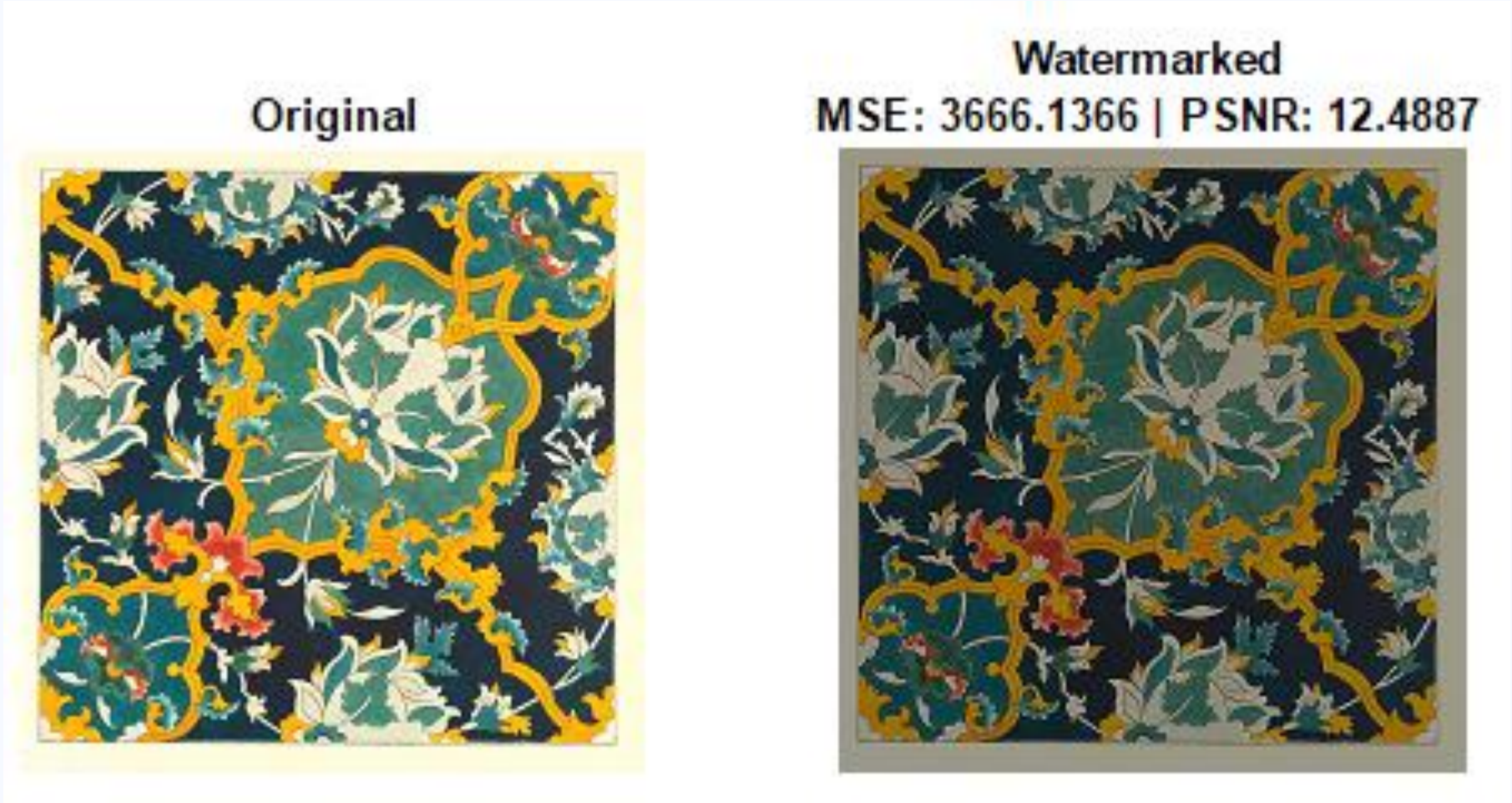
1.2 HAAR 3-L
K, Q = 0.2, 0.01

Original Watermark

Extracted Watermark
MSE: 8377.1873 | PSNR: 8.8998

RESULTS

1.2 HAAR 3-L
K, Q = 0.6, 0.01



EXTRACTION

1.2 HAAR 3-L
K, Q = 0.6, 0.01

Original Watermark

Extracted Watermark
MSE: 2325.8551 | PSNR: 14.465

RESULTS

1.2 HAAR 3-L
K, Q = 1.0, 0.01

EXTRACTION

1.2 HAAR 3-L
K, Q = 1.0, 0.01

Original Watermark

Extracted Watermark
MSE: 1512.3034 | PSNR: 16.3344

RESULTS

1.2 HAAR 3-L
K, Q = 1.4, 0.01

Original

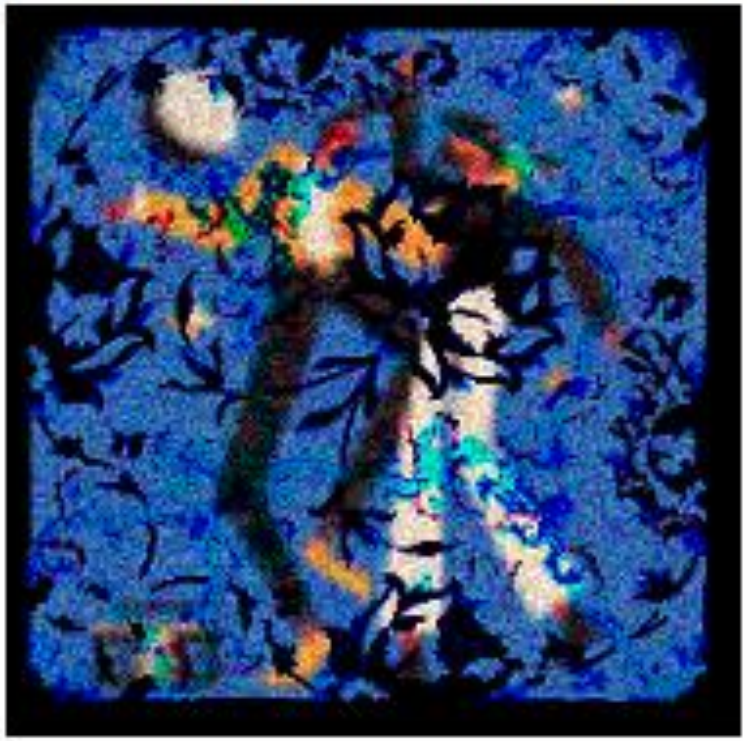
Watermarked
MSE: 1421.1176 | PSNR: 16.6045

EXTRACTION

1.2 HAAR 3-L
K, Q = 1.4, 0.01

Original Watermark

Extracted Watermark
MSE: 5798.6589 | PSNR: 10.4975



RESULTS

1.2 HAAR 3-L
K, Q = 1.8, 0.01

Original

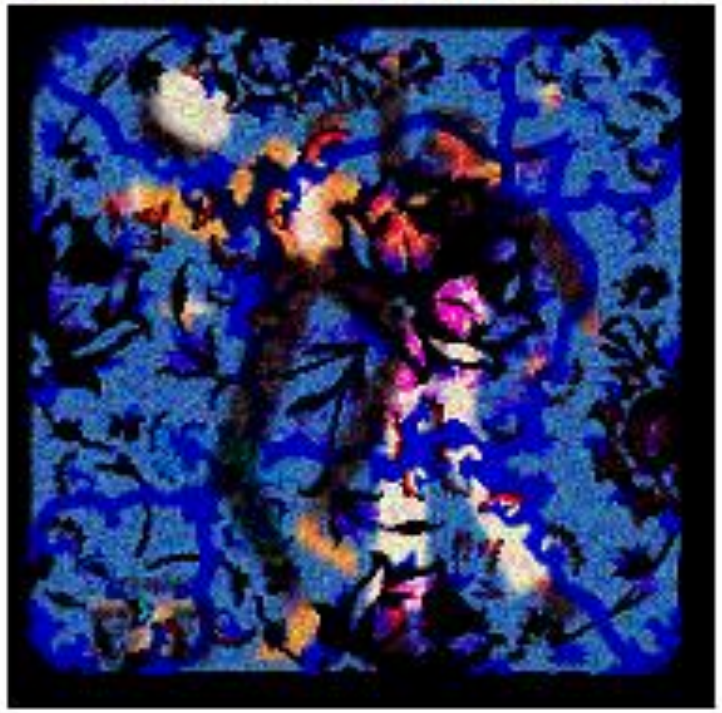
Watermarked
MSE: 3770.7284 | PSNR: 12.3666

EXTRACTION

1.2 HAAR 3-L
K, Q = 1.8, 0.01

Original Watermark

Extracted Watermark
MSE: 6954.8252 | PSNR: 9.7079



III. 2

DAUBECHIES 3-L

RESULTS

2.1 DB 3-L
K, Q = 0.2, 0.009

Original

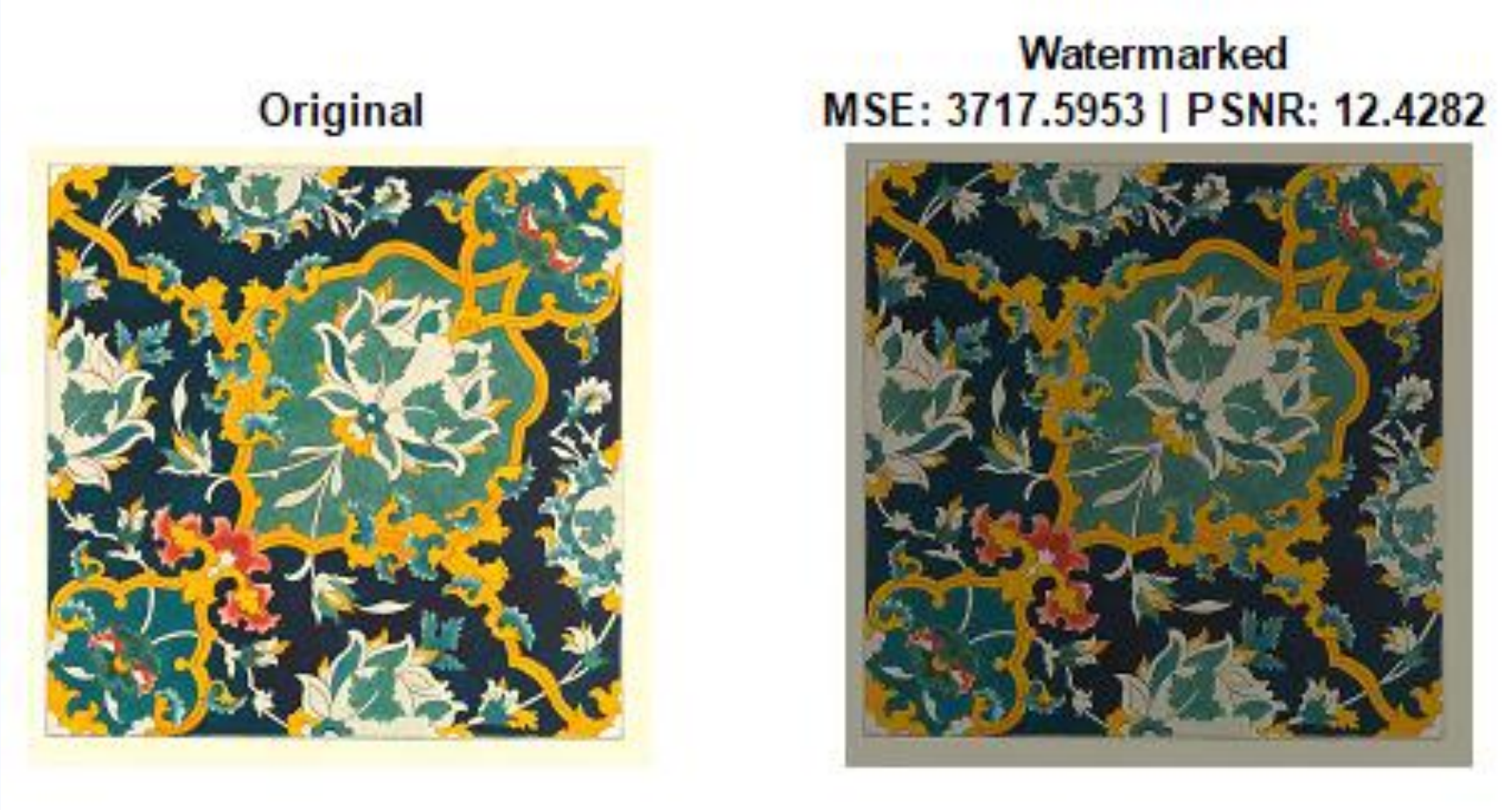
Watermarked
MSE: 14776.6294 | PSNR: 6.435

EXTRACTION

2.1 DB 3-L
K, Q = 0.2, 0.009

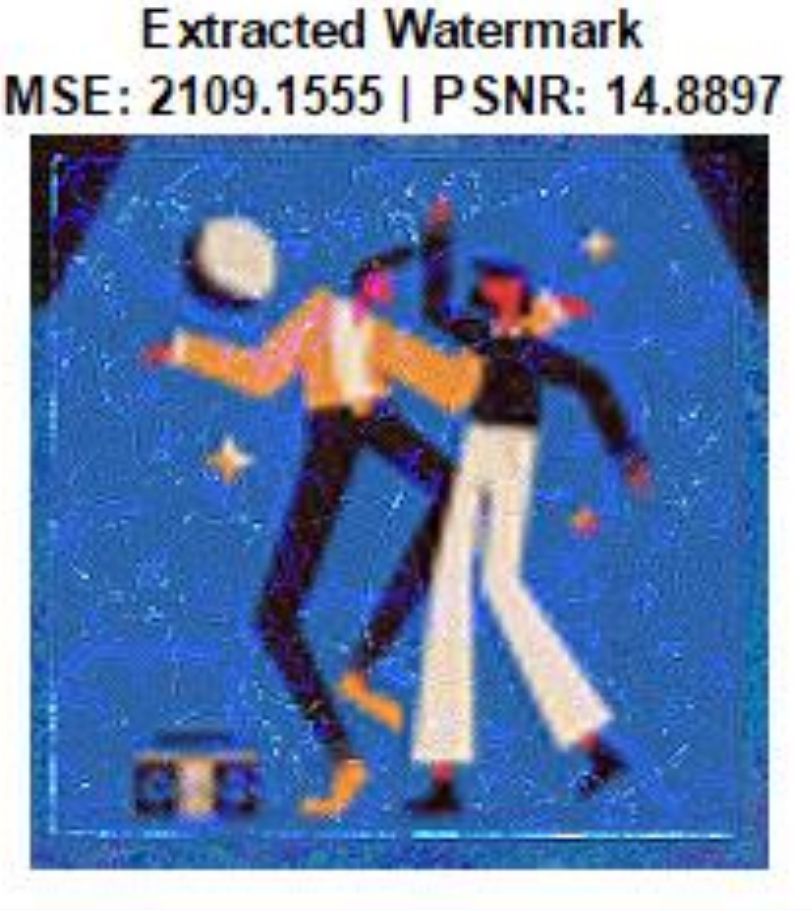
RESULTS

2.1 DB 3-L
K, Q = 0.6, 0.009



EXTRACTION

2.1 DB 3-L
K, Q = 0.6, 0.009

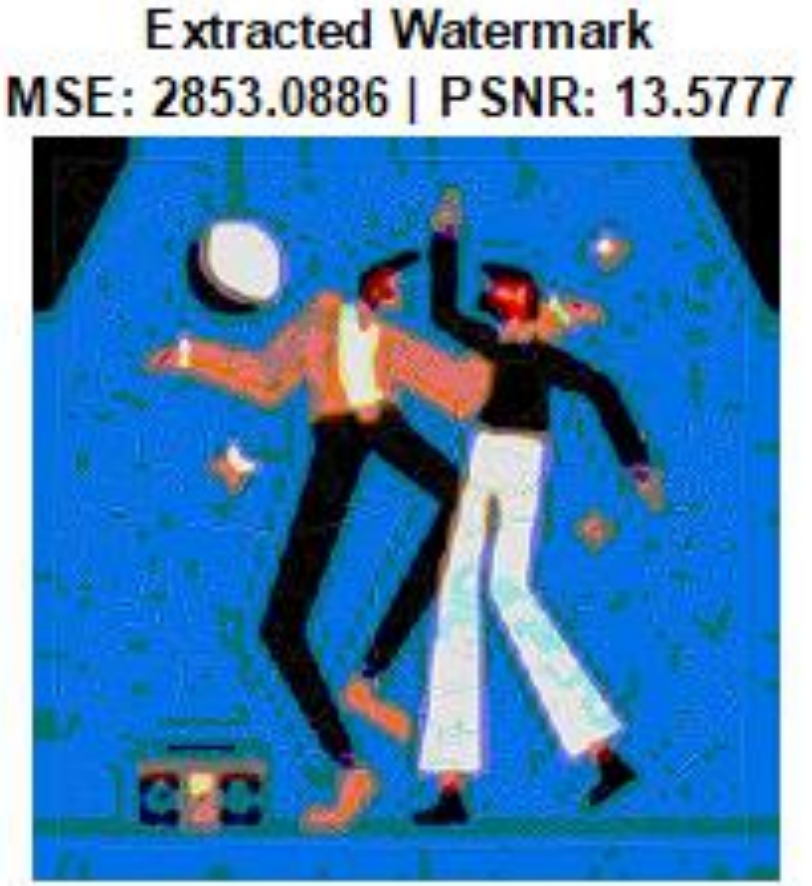


RESULTS

2.1 DB 3-L
K, Q = 1.0, 0.009

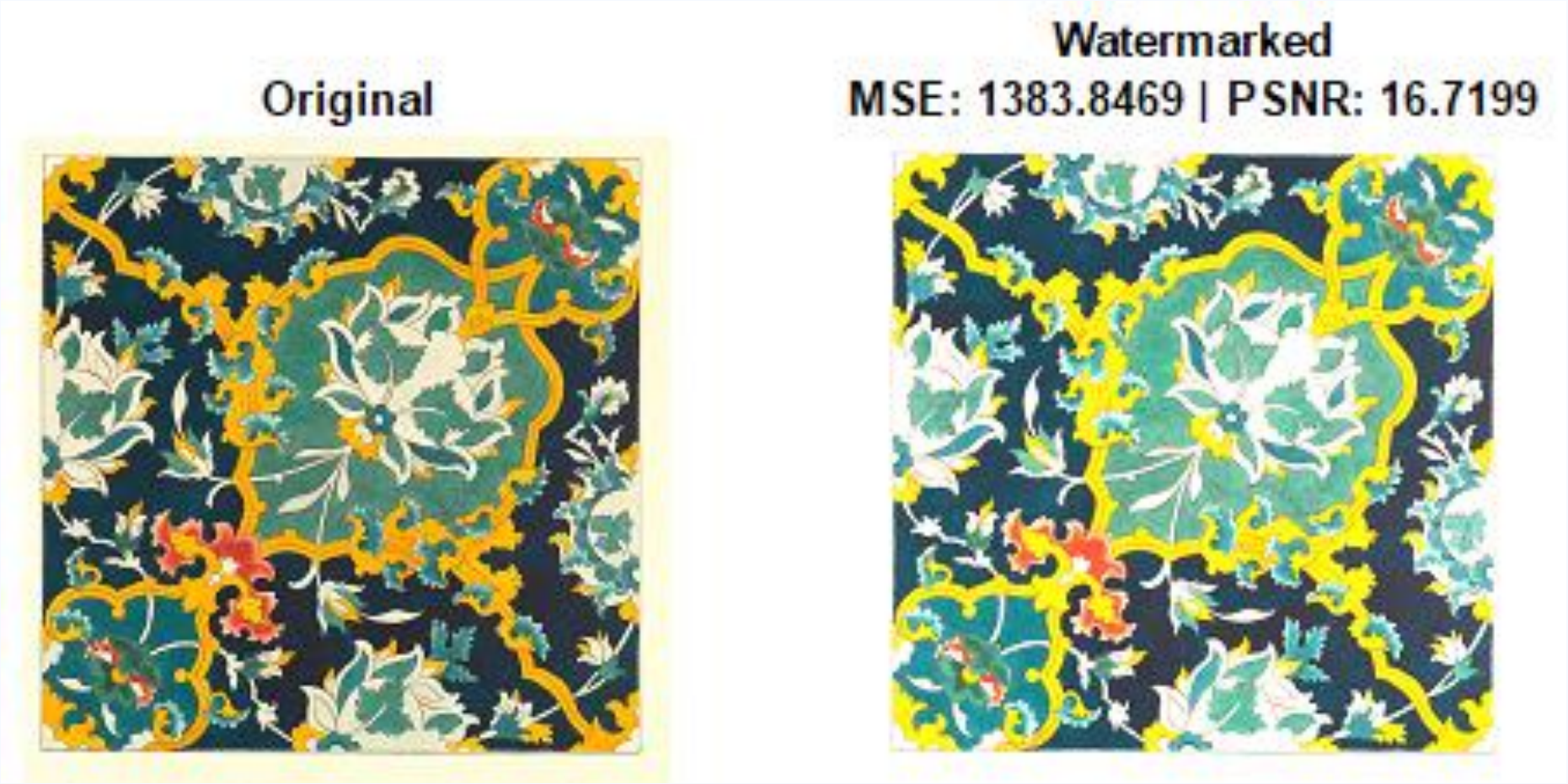
EXTRACTION

2.1 DB 3-L
K, Q = 1.0, 0.009



RESULTS

2.1 DB 3-L
K, Q = 1.4, 0.009



EXTRACTION

2.1 DB 3-L
K, Q = 1.4, 0.009

Original Watermark

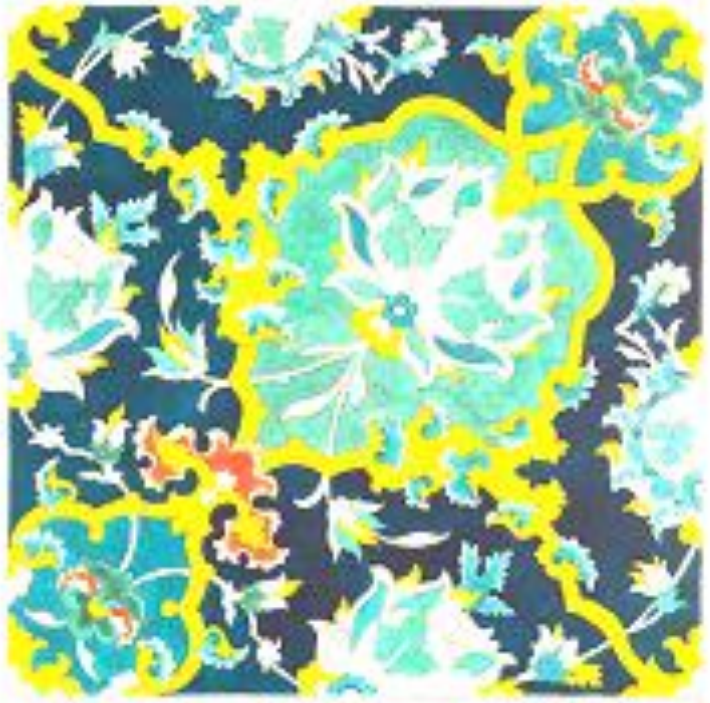
Extracted Watermark
MSE: 8818.93 | PSNR: 8.6766

RESULTS

2.1 DB 3-L
K, Q = 1.8, 0.009

Original

Watermarked
MSE: 3630.9183 | PSNR: 12.5306



EXTRACTION

2.1 DB 3-L
K, Q = 1.8, 0.009

Original Watermark

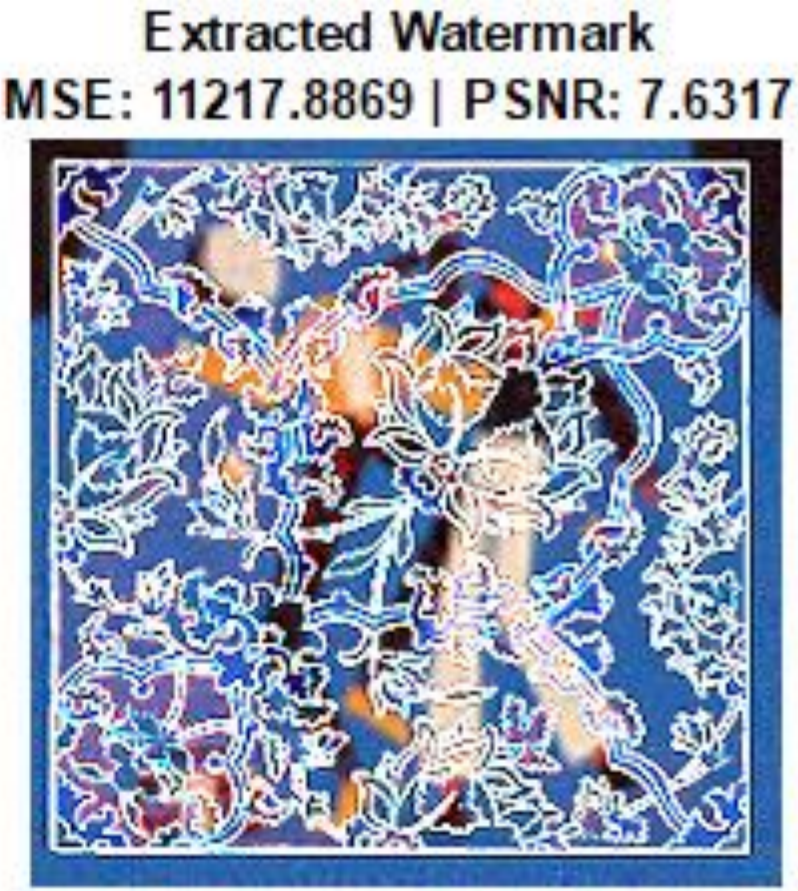
Extracted Watermark
MSE: 10874.3101 | PSNR: 7.7668

RESULTS

2.2 DB 3-L
K, Q = 0.2, 0.01

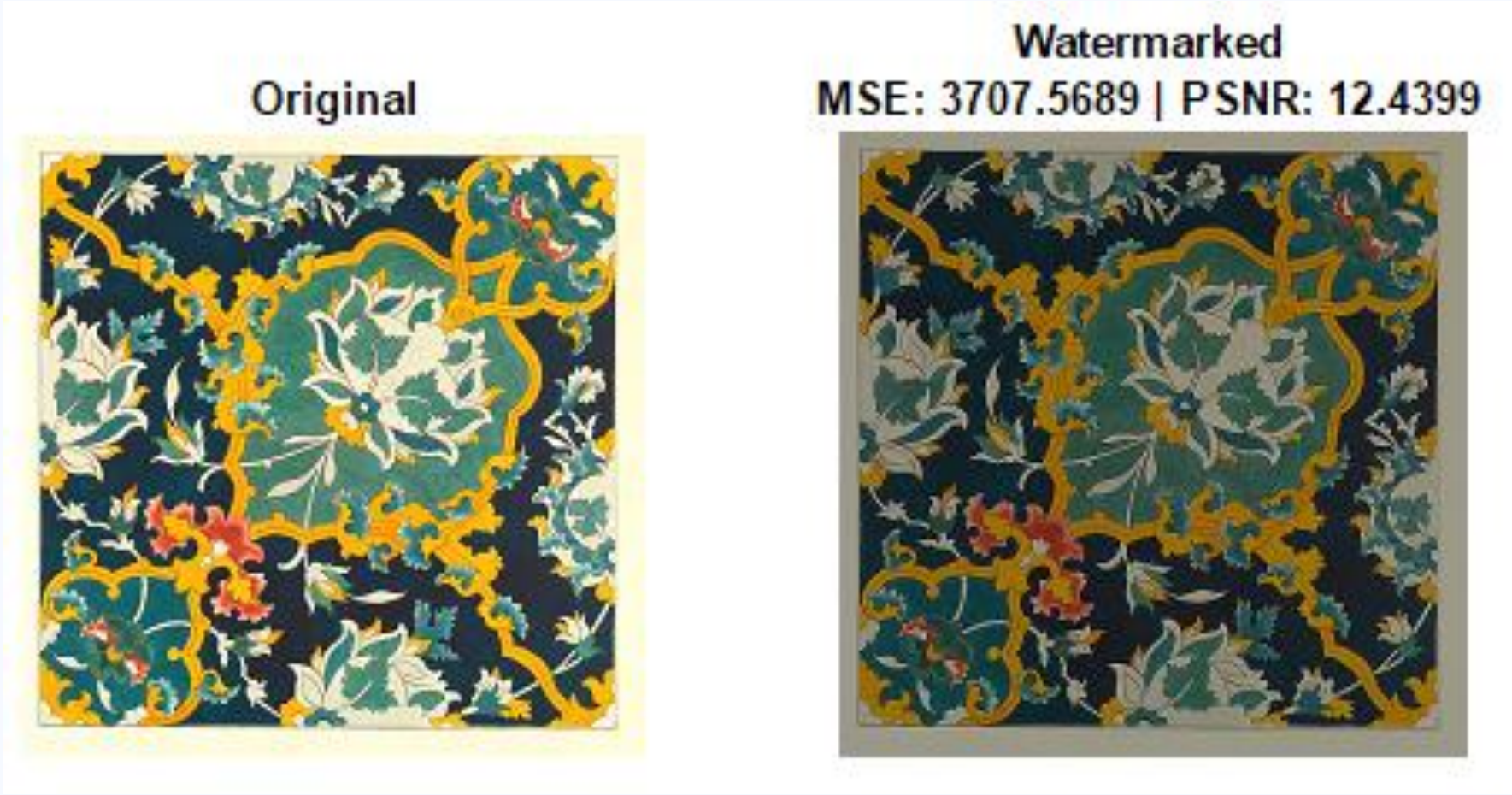
EXTRACTION

2.2 DB 3-L
K, Q = 0.2, 0.01



RESULTS

2.2 DB 3-L
K, Q = 0.6, 0.01

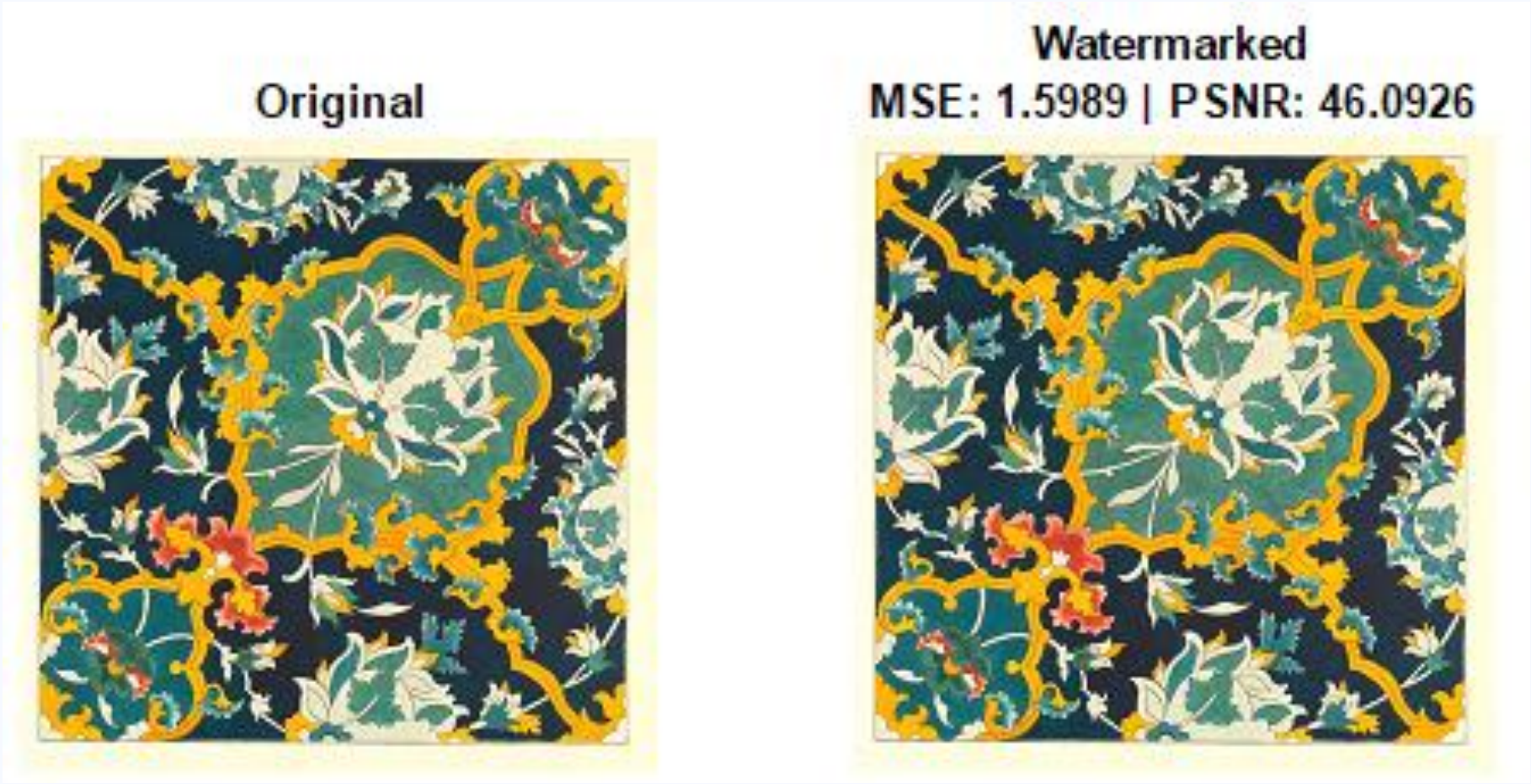


EXTRACTION

2.2 DB 3-L
K, Q = 0.6, 0.01

RESULTS

2.2 DB 3-L
K, Q = 1.0, 0.01



EXTRACTION

2.2 DB 3-L
K, Q = 1.0, 0.01

RESULTS

2.2 DB 3-L
K, Q = 1.4, 0.01

Original

Watermarked
MSE: 1388.7691 | PSNR: 16.7045

EXTRACTION

2.2 DB 3-L
K, Q = 1.4, 0.01

Original Watermark

Extracted Watermark
MSE: 8682.1471 | PSNR: 8.7445

RESULTS

2.2 DB 3-L
K, Q = 1.8, 0.01

Original

Watermarked
MSE: 3638.4822 | PSNR: 12.5216

EXTRACTION

2.2 DB 3-L
K, Q = 1.8, 0.01

Original Watermark

Extracted Watermark
MSE: 10755.7642 | PSNR: 7.8144

III. 3

CUMULATIVE RESULTS

	Base Image		Watermark	
H-3	***MSE***	***PSNR***	***MSE***	***PSNR***
0.2, 0.009	14503	6.5	8692	8.7
0.6, 0.009	3676	12.4	2577	14.01
1.0, 0.009	1.45	46.49	1965	15.19
1.4, 0.009	1416	16.6	5915	10.41
1.8, 0.009	3762	12.37	7058	9.64
0.2, 0.01	14484	6.5	8377	8.89
0.6, 0.01	3666	12.48	2325	14.46
1.0, 0.01	1.61	46.05	1512	16.33
1.4, 0.01	1421	16.6	5798	10.49
1.8, 0.01	3770	12.36	6954	9.7

	Base Image		Watermark	
DB-3	***MSE***	***PSNR***	***MSE***	***PSNR***
0.2, 0.009	14776	6.4	11592	7.48
0.6, 0.009	3717	12.42	2109	14.88
1.0, 0.009	1.43	46.55	2853	13.57
1.4, 0.009	1383	16.71	8818	8.67
1.8, 0.009	3630	12.5	10847	7.76
0.2, 0.01	14757	6.44	11217	7.63
0.6, 0.01	3707	12.43	2039	15.03
1.0, 0.01	1.59	46.09	2402	14.32
1.4, 0.01	1388	16.7	8682	8.74
1.8, 0.01	3638	12.52	10755	7.81

IV.

CODE

Explaining the codes used on this project: functions, procedures, conversions and representations

4.2. Compression

```
in = 2;
fam = 'db3';
lvl = 5;
thr_type = 1;
gthr = 100;
```

```
if in == 1
    x = 'rail.jpg';
elseif in == 2
    x = 'disco.jpg';
else
    x = 'circ.jpg';
end
base = imread(x);
```

1

- شماره تصویر انتخابی
- خانواده wavelet
- سطح یا عمق DWT
- نوع آستانه (1: گلوبال، 0: لوکال)
- مقدار آستانه گلوبال (در صورت 1 بودن خط بالا استفاده میشود)

2

- با توجه به شماره عکس، آدرس عکس را برای خوانده شدن پیدا میکند.
- عکس را میخواند.

4.1. Compression

```
dc = wavedec3(base, lvl, fam);

lll_base = abs(dc.dec{1});
hll_base = abs(dc.dec{2});
lhl_base = abs(dc.dec{3});
hhl_base = abs(dc.dec{4});
llh_base = abs(dc.dec{5});
hlh_base = abs(dc.dec{6});
lhh_base = abs(dc.dec{7});
hhh_base = abs(dc.dec{8});
```

3

- با توجه به نوع خانوده، تصویر خوانده شده و سطح، DWT سه بعدی اعمال میکند. (چون تصویر را رنگی خواندیم.)
- ماتریس های ضرایب را ذخیره میکند.
- (ترتیب ماتریس ها بر اساس وبسایت متلب است.)

4.1. Compression

```
lll_std = stdfilt(lll_base);
hll_std = stdfilt(hll_base);
lhl_std = stdfilt(lhl_base);
hhl_std = stdfilt(hhl_base);
llh_std = stdfilt(llh_base);
hlh_std = stdfilt(hlh_base);
lhh_std = stdfilt(lhh_base);
hhh_std = stdfilt(hhh_base);
```

4

- انحراف از معیار را برای همه ماتریس های ضرایب پیدا میکنیم. (برای استفاده در فرمول آستانه لوکال: sigma)

```
lll_n = numel(lll_base);
hll_n = numel(hll_base);
lhl_n = numel(lhl_base);
hhl_n = numel(hhl_base);
llh_n = numel(llh_base);
hlh_n = numel(hlh_base);
lhh_n = numel(lhh_base);
hhh_n = numel(hhh_base);
```

5

- تعداد اعضا را برای همه ماتریس های ضرایب پیدا میکنیم. (برای استفاده در فرمول آستانه لوکال: N)

4.1. Compression

```
lll_t = lll_std * (sqrt(2 * log2(lll_n)));
hll_t = hll_std * (sqrt(2 * log2(hll_n)));
lhl_t = lhl_std * (sqrt(2 * log2(lhl_n)));
hhl_t = hhl_std * (sqrt(2 * log2(hhl_n)));
llh_t = llh_std * (sqrt(2 * log2(llh_n)));
hlh_t = hlh_std * (sqrt(2 * log2(hlh_n)));
lhh_t = lhh_std * (sqrt(2 * log2(lhh_n)));
hhh_t = hhh_std * (sqrt(2 * log2(hhh_n)));
```

6

- اعمال فرمول با مقادیر به دست آمده از قبل و پیدا کردن آستانه لوکال برای هر ماتریس ضرایب.

```
if thr_type == 1
    lll_t = gthr;
    hll_t = gthr;
    lhl_t = gthr;
    hhl_t = gthr;
    llh_t = gthr;
    hlh_t = gthr;
    lhh_t = gthr;
    hhh_t = gthr;
end
```

7

- اگر بنا به استفاده از آستانه گلوبال از، مرحله قبل را نادیده بگیر و همه آستانه ها را مساویِ آستانه گلوبال کن.

4.1. Compression

```
lll_size = size(lll_base);
hll_size = size(hll_base);
lhl_size = size(lhl_base);
hhl_size = size(hhl_base);
llh_size = size(llh_base);
hlh_size = size(hlh_base);
lhh_size = size(lhh_base);
hhh_size = size(hhh_base);
```

8

- ابعاد همه ماتریس های ضرایب را حساب کن.

```
lll_zeros = sum(lll_base==0,'all');
hll_zeros = sum(hll_base==0,'all');
lhl_zeros = sum(lhl_base==0,'all');
hhl_zeros = sum(hhl_base==0,'all');
llh_zeros = sum(llh_base==0,'all');
hlh_zeros = sum(hlh_base==0,'all');
lhh_zeros = sum(lhh_base==0,'all');
hhh_zeros = sum(hhh_base==0,'all');
```

9

- تعداد صفرهای هر ماتریس را پیدا کن.
- (برای محاسبه نرخ فشرده سازی براساس مقادیر صفر شده. این مقدار اولیه است که بعدا مقایسه میشود.)

4.1. Compression

```
lll_new = lll_base .* double(lll_base > lll_t);
hll_new = hll_base .* double(hll_base > lll_t);
lhl_new = lhl_base .* double(lhl_base > lll_t);
hhl_new = hhl_base .* double(hhl_base > lll_t);
llh_new = llh_base .* double(llh_base > lll_t);
hlh_new = hlh_base .* double(hlh_base > lll_t);
lhh_new = lhh_base .* double(lhh_base > lll_t);
hhh_new = hhh_base .* double(hhh_base > lll_t);
```

10

- اعمال آستانه. اگر مقدار قبل از آستانه بزرگتر بود در ماتریس های جدید نگه دار، وگرنه حذف کن.)

4.1. Compression

```
lll_new_zeros = sum(lll_new==0,'all');
hll_new_zeros = sum(hll_new==0,'all');
lhl_new_zeros = sum(lhl_new==0,'all');
hhl_new_zeros = sum(hhl_new==0,'all');
llh_new_zeros = sum(llh_new==0,'all');
hlh_new_zeros = sum(hlh_new==0,'all');
lhh_new_zeros = sum(lhh_new==0,'all');
hhh_new_zeros = sum(hhh_new==0,'all');
```

11

- تعداد صفرهای ماتریس های جدید را بشمار.
- (یعنی صفرهای بعد از اعمال آستانه.)

4.1. Compression

```
lll_zeros_diff = abs(lll_zeros - lll_new_zeros);
hll_zeros_diff = abs(hll_zeros - hll_new_zeros);
lhl_zeros_diff = abs(lhl_zeros - lhl_new_zeros);
hhl_zeros_diff = abs(hhl_zeros - hhl_new_zeros);
llh_zeros_diff = abs(llh_zeros - llh_new_zeros);
hlh_zeros_diff = abs(hlh_zeros - hlh_new_zeros);
lhh_zeros_diff = abs(lhh_zeros - lhh_new_zeros);
hhh_zeros_diff = abs(hhh_zeros - hhh_new_zeros);
```

12

- اختلاف صفرهای جدید و قدیم را حساب کن.

4.1. Compression

```
figure
colormap(gray);
subplot(2,8,1); image(lll_base(:,:,1)); title('LLL');axis square;
subplot(2,8,2); image(hll_base(:,:,1)); title('HLL');axis square;
subplot(2,8,3); image(lhl_base(:,:,1)); title('LHL');axis square;
subplot(2,8,4); image(hhl_base(:,:,1)); title('HHL');axis square;
subplot(2,8,5); image(llh_base(:,:,1)); title('LLH');axis square;
subplot(2,8,6); image(hlh_base(:,:,1)); title('HLH');axis square;
subplot(2,8,7); image(lhh_base(:,:,1)); title('LHH');axis square;
subplot(2,8,8); image(hhh_base(:,:,1)); title('HHH');axis square;
```


- نمایش ماتریس های ضرایب اولیه، بدون آستانه گذاری.

4.1. Compression

14

- نمایش ماتریس های ضرایب پس از آستانه گذاری.

```
subplot(2,8,9); image(lll_new(:,:,1));
title({'LLL thr';lll_zeros_diff});axis square;
subplot(2,8,10);image(hll_new(:,:,1));
title({'HLL thr';hll_zeros_diff});axis square;
subplot(2,8,11); image(lhl_new(:,:,1));
title({'LHL thr';lhl_zeros_diff});axis square;
subplot(2,8,12); image(hhl_new(:,:,1));
title({'HHL thr';hhl_zeros_diff});axis square;
subplot(2,8,13); image(llh_new(:,:,1));
title({'LLH thr';llh_zeros_diff});axis square;
subplot(2,8,14); image(hlh_new(:,:,1));
title({'HLH thr';hlh_zeros_diff});axis square;
subplot(2,8,15); image(lhh_new(:,:,1))
 title({'LHH thr';lhh_zeros_diff});axis square;
subplot(2,8,16); image(hhh_new(:,:,1));
title({'HHH thr';hhh_zeros_diff});axis square;
```

4.1. Compression

```
dc.dec{1} = lll_new;
dc.dec{2} = hll_new;
dc.dec{3} = lhl_new;
dc.dec{4} = hhl_new;
dc.dec{5} = llh_new;
dc.dec{6} = hlh_new;
dc.dec{7} = lhh_new;
dc.dec{8} = hhh_new;
```

15

- جایگذاری ماتریس های ضرایب آستانه گذاری شده برای بازسازی.

```
cmp = waverec3(dc);
cmp = uint8(cmp);

D = abs(cmp - base) .^2;
mse  = sum(D(:))/numel(base);
psnr = 10*log10(255*255/mse);
```

16

- بازسازی سه بعدی ماتریس های ضرایب، تبدیل به تصویر فشرده شده
- محاسبه مقادیر mse و psnr بین تصویر اولیه و فشرده شده.

4.1. Compression

```
all_zeros = lll_zeros_diff +
hll_zeros_diff +lhl_zeros_diff +
hhl_zeros_diff + llh_zeros_diff +
hlh_zeros_diff + lhh_zeros_diff +
hhh_zeros_diff;

all_elements = numel(base);
cr = 100 * (all_zeros /
all_elements);
```

17

- محاسبه مجموع همه ضرایبِ صفر شده.
- محاسبه تعداد همه ضرایب اَولیه در تصویر.
- محاسبه نرخ فشرده سازی براساس تعداد ضرایب صفر شده و درصدگیری از آن.

```
figure
subplot(1,2,1);
imshow(base); axis square;
title("Original");
subplot(1,2,2);
imshow(cmp); axis square;
msg = strcat("MSE: ", num2str(mse),
" | PSNR: ", num2str(psnr));
title({strcat("Ratio: %",
num2str(cr));msg});
imwrite(cmp, 'compressed.jpg');
```

18

- نمایش تصویر اصلی و فشرده شده. همچنین mse، psnr و نرخ فشرده سازی به دست آمده از قبل.
- ذخیره نتیجه فشرده سازی در حافظه داخلی.

IV.2

WATERMARKING

Codes explained

4.2. Watermarking

```
cover = imread('circ.jpg');
wm = imread('disco.jpg');
fam = 'db3';
lvl = 3;
k = 1.8;
q = 0.01;
```

1

- تعیین تصویر کاور و واترمارک
- تعیین موجک مادر
- تعیین سطح موجک
- تعیین متغیرهای k و q

```
figure
subplot(1,2,1);
imshow(cover);
title("Cover");
subplot(1,2,2);
imshow(wm);
title("Watermark");
```

2

- نمایش تصویر کاور و واترمارک

4.2. Watermarking

```
[c_cover, s_cover] = wavedec2(cover, lvl, fam);
ll_cover = appcoef2(c_cover, s_cover, fam, lvl);
[lh_cover, hl_cover, hh_cover] = detcoef2('all',
c_cover, s_cover, lvl);
```

3

- اعمال موجک روی تصویر کاور
- استخراج ماتریس ضرایب LL
- استخراج سایر ماتریس های ضرایب (LH، HL و HH)

4.2. Watermarking

```
rng = size(cover, 1);
ll_cover_scaled = wcodemat(ll_cover, rng, 'mat', lvl);
lh_cover_scaled = wcodemat(lh_cover, rng, 'mat', lvl);
hl_cover_scaled = wcodemat(hl_cover, rng, 'mat', lvl);
hh_cover_scaled = wcodemat(hh_cover, rng, 'mat', lvl);
```

4

- متوازن کردن گستره رنگی ماتریس های ضرایب برای نمایش بهتر
- *(مهم نیست. جایی استفاده نمیشه. فقط واسه نمایش ماتریس هاست. حتی میشه حذفش کرد و اهمیتی نداشته باشه.)*

4.2. Watermarking

```
[c_wm, s_wm] = wavedec2(wm, lvl, fam);
ll_wm = appcoef2(c_wm, s_wm, fam, lvl);
[lh_wm, hl_wm, hh_wm] = detcoef2('all', c_wm, s_wm, lvl);
wm_rng = size(wm, 1);
ll_wm_scaled = wcodemat(ll_wm, wm_rng, 'mat', 3);
lh_wm_scaled = wcodemat(lh_wm, wm_rng, 'mat', 3);
hl_wm_scaled = wcodemat(hl_wm, wm_rng, 'mat', 3);
hh_wm_scaled = wcodemat(hh_wm, wm_rng, 'mat', 3);
```

5

- انجام مراحل 3 و 4، این بار برای تصویر واترمارک:
- اعمال تبدیل موجک، استخراج ماتریس های ضرایب، توازن رنگ

4.2. Watermarking

```
figure
colormap pink(255);
subplot(2,4,1); imagesc(ll_wm_scaled); title('WM LL'); axis square;
subplot(2,4,2); imagesc(lh_wm_scaled); title('WM LH'); axis square;
subplot(2,4,3); imagesc(hl_wm_scaled); title('WM HL'); axis square;
subplot(2,4,4); imagesc(hh_wm_scaled); title('WM HH'); axis square;
subplot(2,4,5); imagesc(ll_cover_scaled); title('ORG LL'); axis square;
subplot(2,4,6); imagesc(lh_cover_scaled); title('ORG LH'); axis square;
subplot(2,4,7); imagesc(hl_cover_scaled); title('ORG HL'); axis square;
subplot(2,4,8); imagesc(hh_cover_scaled); title('ORG HH'); axis square;
```

6

- نمایش ماتریس های ضرایب

4.2. Watermarking

7

- تبدیل نوع ضرایب k و q به نوع double
- پیدا کردن ابعاد باند LL برای کاور و واترمارک

```
k = double(k); q = double(q);
[x,y,z] = size(ll_cover); ll_cover_size = [x, y];
[x,y,z] = size(ll_wm); ll_wm_size = [x, y];
```

```
ll_wm_resized = imresize(ll_wm, ll_cover_size);
ll_result = (k * ll_cover) + (q * ll_wm_resized);
ll_result_1d = ll_result(:)'; ll_result_1d_size = size(ll_result_1d);
```

8

- تغییر اندازه واترمارک به اندازه کاور (تا در ادامه روی همه سطح تصویر قرار بگیرد.)
- اعمال فرمول اصلی با توجه به ضرایب k و q و باندهای LL تصویر کاور و واترمارک
- تبدیل باند LL حاصل از فرمول به آرایه افقی (چون آرایه c، خروجی wavedec و ورودی waverec یک بعدی است و LL قرار است قطعه اول آن باشد.)
- پیدا کرد ابعاد ماحصل خط

4.2. Watermarking

```
c_result = c_cover;
s_result = s_cover;
for id=1:ll_result_1d_size(2)
      c_result(1,id) = ll_result_1d(1,id);
end
```

9

- کپی گرفتن از ماتریس های c و s تصویر کاور و جایگزینی (موقت) به عنوان c و s تصویر خروجی
- جایگذاری مقادیر باند LL خروجی در ماتریس c تصویر خروجی

```
result = waverec2(c_result, s_result, fam);
wimage = uint8(result);
mse  = immse(cover, wimage);
psnr = psnr(cover, wimage);
```

10

- بازسازی تصویر خروجی با استفاده از waverec2. عملا اعمال IDWT.
- تبدیل تصویر به تصویر 3 بیتی.
- محاسبه معیارهای MSE و PSNR بین تصویر واترمارک شده و کاور اولیه.

4.2. Watermarking

```
figure
subplot(1,2,1);
imshow(cover); title("Original"); axis square;
subplot(1,2,2);
imshow(wimage); colormap(gray); axis square;
msg = strcat("MSE: ", num2str(mse), " | PSNR: ", num2str(psnr));
title({"Watermarked";msg});
imwrite(wimage, 'watermarked.png');
```

11

- نمایش تصویر واترمارک شده و تصویر اصلی
- نمایش معیارهای MSE و PSNR
- ذخیره تصویر خروجی در حافظه

4.2. De-Watermarking

```
clearvars -except k q lvl fam ll_cover wm rng;
wimage = imread('watermarked.png');
[c_wi, s_wi] = wavedec2(wimage, lvl, fam);
ll_wi = appcoef2(c_wi, s_wi, fam, lvl);
[lh_wi, hl_wi, hh_wi] = detcoef2('all', c_wi, s_wi, lvl);
```

1

- پاک کردن همه متغیرها به جز موارد مذکور
- خواندن تصویر واترمارک شده
- اعمال dwt روی تصویر
- استخراج ماتریس ضرایب تخمین LL
- استخراج سایر ماتریس های ضرایب (LH، HL، HH)

4.2. De-Watermarking

```
ll_ext = (ll_wi - (k * ll_cover)) / q;
ll_ext_1d = ll_ext(:)';
ll_ext_1d_size = size(ll_ext_1d);
```

2

- اعمال فرمول موجود برای استخراج واترمارک اولیه
- تبدیل ماتریس باند LL به آرایه افقی (برای استفاده و بازسازی c)
- پیداکردن ابعاد ماحصل خط قبل

```
c_ext = c_wi;
s_ext = s_wi;
for id = 1:ll_ext_1d_size(2)
    c_ext(id) = ll_ext_1d(id);
end
```

3

- کپی گرفتن از ماتریس های c و s
- جایگزینی ضرایب به دست آمده از فرمول استخراج به جای باند LL تصویر خروجی

4.2. De-Watermarking

4

```
c_ext_size = size(c_ext);
for id = ll_ext_1d_size(2)+1:c_ext_size(2)
        c_ext(id) = 0;
end
```

- صفر کردن ضرایب سایر باندهای موجود در c

5

```
ext = waverec2(c_ext, s_ext, fam);
ext = uint8(ext);
[x,y,z] = size(ext);
ext_size = [x, y];
wm_resized = imresize(wm, ext_size);
```

- اعمال IDWT روی واترمارک استخراج شده
- تبدیل آن به تصویر 3 بیتی
- تغییر اندازه آن به اندازه اولیه
- (چون در فرآیند واترمارک کردن، به اندازه کاور تغییر یافته بود.)

4.2. De-Watermarking

6

- محاسبه معیارهای MSE و PSNR بین واترمارک اصلی و استخراج شده

```
mse  = immse(wm_resized, ext);
psnr = psnr(wm_resized, ext);
```

7

- نمایش واترمارک اولیه و استخراج شده
- نمایش معیارهای MSE و PSNR
- ذخیره واترمارک استخراج شده در حافظه

```
figure
subplot(1,2,1);
imshow(wm_resized); title("Original
Watermark"); axis square;
subplot(1,2,2);
imshow(ext); colormap(gray); axis square;
msg = strcat("MSE: ", num2str(mse), " |
PSNR: ", num2str(psnr));
title({"Extracted Watermark";msg});
imwrite(ext, 'extracted_watermark.png');
```

V.

FUTURE WORKS

What needs to be done, suggested revisions and extensions on method and functionality

5. FUTURE WORK

- Try other wavelet families: `bior`, `syml`, `spiht`, etc.
- Try higher DWT levels
- Try both global and local thresholding for coeff. Matrices
  - Using `ddencom`, `adaptthresh`, etc.
- Try larger image
- Try tiling the watermark
- Add exception catching for bad inputs

VI.

REFERENCES

Books, links, blogposts, etc.

6. REFERENCES

1. https://www.mathworks.com/help/wavelet/ref/wcompress.html
2. https://www.mathworks.com/help/wavelet/ref/wavedec2.html?searchHighlight=wavedec2&s_tid=srchtitle_wavedec2_1
3. https://www.mathworks.com/help/wavelet/ref/waverec2.html?searchHighlight=waverec2&s_tid=srchtitle_waverec2_1
4. https://www.mathworks.com/help/wavelet/ref/dwt2.html?searchHighlight=dwt2&s_tid=srchtitle_dwt2_1
5. https://www.mathworks.com/help/wavelet/ref/idwt2.html?searchHighlight=idwt2&s_tid=srchtitle_idwt2_1

6. REFERENCES

6. Discrete wavelet transform. (2022, December 15). In *Wikipedia*. https://en.wikipedia.org/wiki/Discrete_wavelet_transform
7. https://www.scribd.com/document/436856865/Concise-Introduction-to-Wavelets
8. S. Mallat, A Wavelet Tour of Signal Processing, 2nd ed. San Diego, CA: Academic, 1999
9. Gonzales, Rafael C., and Paul Wintz. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc., 2008.


THANK YOU