Assignment 2 Report

1. LSB Data Hiding

1.1 Bit Plane Examination

1.1.1 MATLAB function to Extract Bit Plane

```
function bitPlane = getBitPlane(image, n)
% Input:
%     image: original image with 8 bit planes.
%     n: the nth least significant bit to process.
% Return:
%     bitPlane: the corresponding bit plane of input image.

    bitPlane = uint8(zeros(size(image)));
    % Use this mask to filt unrelated bits.
    mask = uint8(bitshift(1, n - 1));
    for i = 1 : size(image, 1)
        for j = 1 : size(image, 2);
            bitPlane(i, j) =  bitand(image(i, j), mask);
            % If this bit is set to 1, modify the pixel value to 255,
            % so that the output image could be easier for visual examine.
            if bitPlane(i, j) > 0
                bitPlane(i, j) = 255;
            end
        end
    end

end
```

1.1.2 Questions Related to Noise Bit Planes

For the pepper image, bit planes are examined from MSB to LSB. The noise increases, starting from the 6th MSB, the image is almost not recognizable and the least part of objects can be inferred to be there. While on the 7th bit plane, the image seems to contain only pure noise.

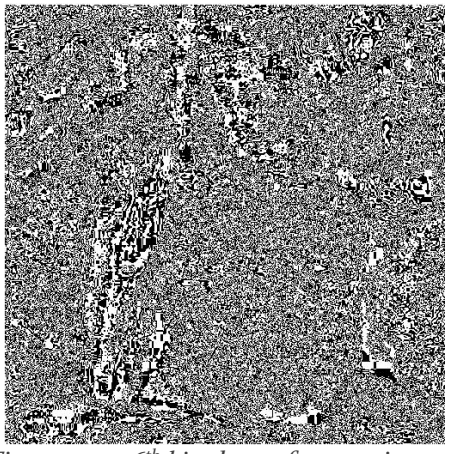

Figure 1.1.2.1 6th bit plane of pepper image

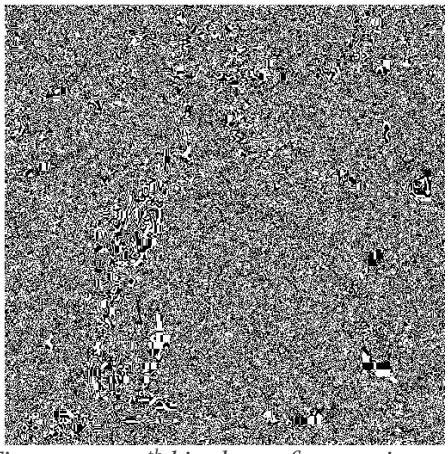

Figure 1.1.2.2 7th bit plane of pepper image

For the baboon image, the 5th MSB bit plane shows a little evidence of the face of baboon, and starts from 6th MSB bit plane, the image represents pure noise.

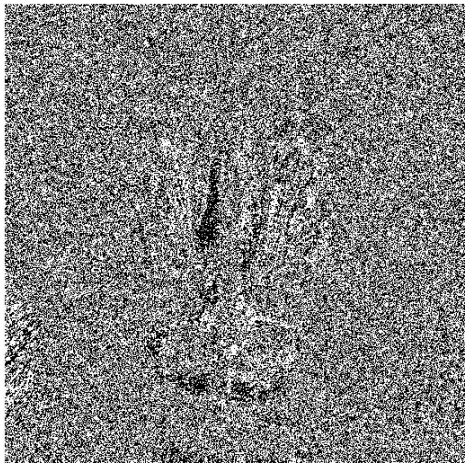

Figure 1.1.2.3 5th bit plane of baboon image

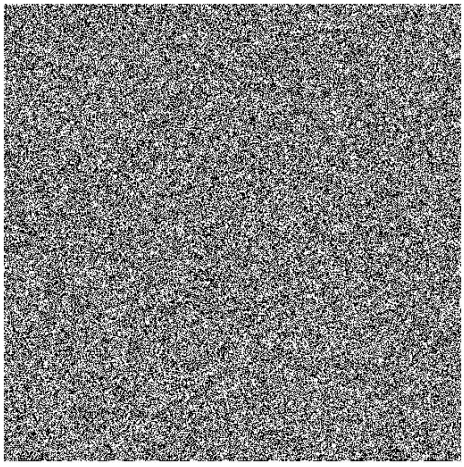

Figure 1.1.2.4 6th bit plane of baboon image

Comparing the pure noise bit planes of the 2 images, they are slightly different. It might because the camera come with different device noise. The reason can also possibly be that different luminance, brightness, or contrast of image. Some certain objects or backgrounds may also cause different LSB noises.

1.2 Examine Bit Planes of Images

LSBwmk1.tiff contains a watermark on 7th MSB bit plane. The watermark is shown below.

Figure 1.2.1 watermark in LSBwmk1.tiff

LSBwmk2.tiff contains a watermark on the 8th MSB / LSB bit plane. The watermark is shown below.

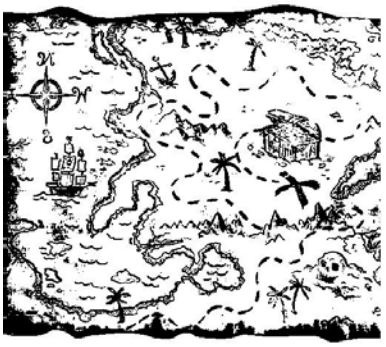

Figure 1.2.2 watermark in LSBwmk2.tiff

LSBwmk3.tiff contains a watermark on 8th MSB / LSB bit plane. The watermark is shown below.

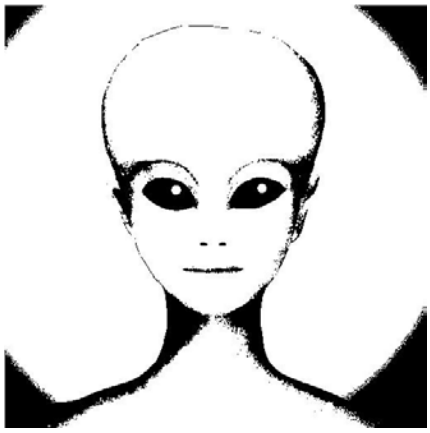

Figure 1.2.3 watermark in LSBwmk3.tiff

1.3 MATLAB Function to Watermark into N LSBs of Original Image

```
function markedImage = embedBit(image, watermark, n)
% Input:
%     image: the original image to be marked.
%     watermark: the image to be embedded into n least significant
%                bits of the original image.
%     n: n: the n least significant bits to process.
% Return:
%     markedImage: the output image with watermark.

    markedImage = uint8(zeros(size(image)));
    % Processing 2 masks for original image and watermark.
    waterMarkMask = uint8(0);
    for i = 9 - n : 8
        % Watermark mask should save n most significant bits of the mark.
        waterMarkMask = bitor(waterMarkMask, uint8(bitshift(1, i - 1)));
    end
    imageMask = uint8(0);
    for i = n + 1 : 8
        % Original image mask erase n least significant bits of it.
        imageMask = bitor(imageMask, uint8(bitshift(1, i - 1)));
    end

    for i = 1 : size(image, 1)
        for j = 1 : size(image, 2)
            % Get most significant bits of watermark.
            watermark(i, j) = bitand(watermark(i, j), waterMarkMask);
            % Move bits to the right.
            watermark(i, j) = bitshift(watermark(i, j), n - 8);
            % Remove least significant bits of original image.
            image(i, j) = bitand(image(i, j), imageMask);
            % Combine original image and watermark.
            markedImage(i, j) = bitor(image(i, j), watermark(i, j));
        end
    end

end
```

1.4 Embedding Watermark into Images

1.4.1 Pepper Image

Retrieve N MSBs from watermark image, replace N LSBs in original pepper image. 8 embedded images, with N from 1 to 8, are shown below.

Figure 1.4.1.1 N = 1

Figure 1.4.1.2 N = 2

Figure 1.4.1.3 N = 3

Figure 1.4.1.4 N = 4

Figure 1.4.1.5 N = 5

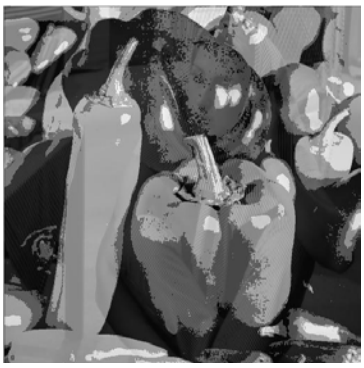

Figure 1.4.1.6 N = 6

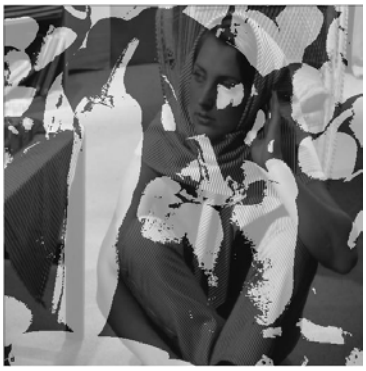

Figure 1.4.1.7 N = 7

Figure 1.4.1.8 N = 8

The original image does not contain visual distortion for up to 3 bit planes. Starting from 6 bit planes, the basic outliner can be visually examined from the embedded image, and it can be examined clearly when there are 7 or more bit planes are embedded.

1.4.2 Baboon Image

Retrieve N MSBs from watermark image, replace N LSBs in original baboon image. 8 embedded images, with N from 1 to 8, are shown below.

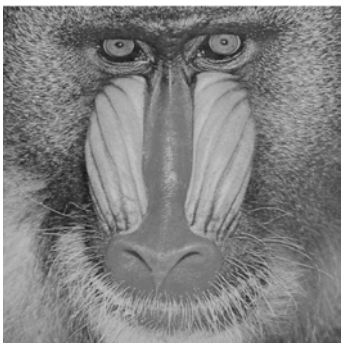

Figure 1.4.2.1 N = 1

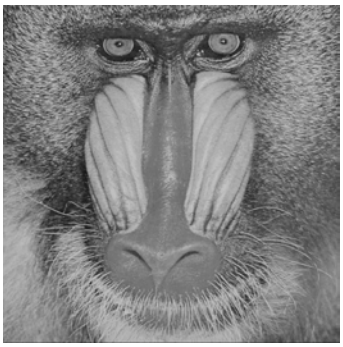

Figure 1.4.2.2 N = 2

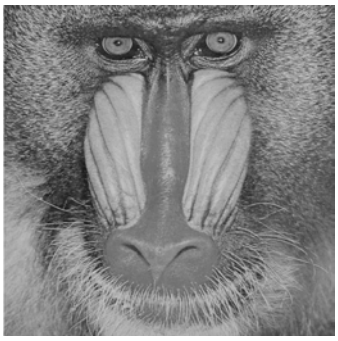

Figure 1.4.1.3 N = 3

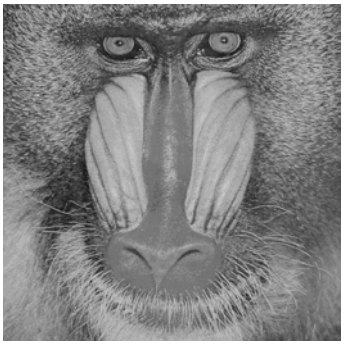

Figure 1.4.2.4 N = 4

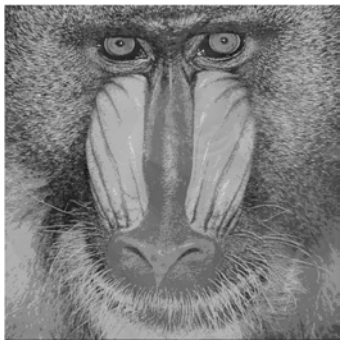

Figure 1.4.2.5 N = 5

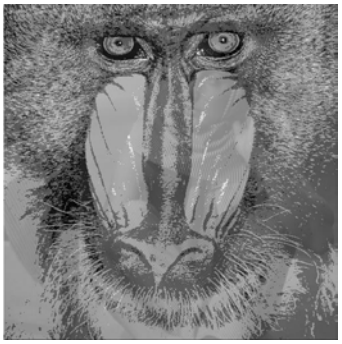

Figure 1.4.2.6 N = 6

Figure 1.4.2.7 N = 7

Figure 1.4.2.8 N = 8

The original image does not contain visual distortion for up to 4 bit planes. Starting from 5 bit planes, the basic outliner can be visually examined from the embedded image, and it can be examined clearly when there are 7 or more bit planes are embedded.

1.4.3 Comparison and Conclusion

The numbers of bits that can be hidden in the original image without visual distortion for the 2 images are different.

I deem that the different luminance, brightness, or contrast may cause the different result. That is, it is possible that for the baboon image, most pixel values in original and watermark images are close to each other, so that after replacement, the embedded image may be more difficult for visual examination.

2. Yeung-Mintzer Watermarking

2.1 MATLAB Function to Embed Y-M Watermark

```
function newImage = YMWaterMark(image, waterMark, key)
% Input:
%     image: original image waiting for water mark
%     waterMark: water mark image
%     key: the key to generate random look up table
% Return:
%     newImage: marked image

    % Seed the random number generator with the user specified key.
    rng(key);
    % Generate look up table values.
    LUTvals = rand(1, 256) > .5;
    % Convert water mark image into binary.
    waterMark = waterMark > .5;

    newImage = uint8(zeros(size(image)));

    for i = 1 : size(image, 1)
        for j = 1 : size(image, 2)
            % Comparing with look up table, if illuminance of water mark
            % region matches look up table.
            if LUTvals(image(i, j)+1) == waterMark(i, j)
                newImage(i, j) = image(i, j);
            % If the water mark region doesn't match, find the matching
            % nearest neighbor.
            else
                newImage(i, j) = nearestNeighbor(LUTvals, ...
                                    image(i, j)+1, waterMark(i, j)) - 1;
            end
        end
    end

end
```

2.2 Embedding Watermark with Y-M Algorithm

2.2.1 Watermark in Pepper and Baboon Image

Watermarked pepper image, by key of 1, is shown below.

Figure 2.2.1.1 LSB bit plane of watermarked pepper image

Watermarked baboon image, by key of 1, is shown below.

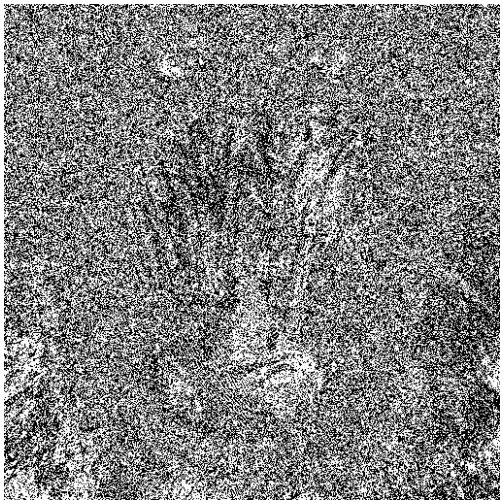

Figure 2.2.1.2 LSB bit plane of watermarked baboon image

2.2.2 Analysis of Watermark

The LSB bit plane of the watermarked image is not visually detectable. LSB algorithm is also utilized for comparison between LSB and Y-M algorithm.

PSNR table is listed below.

Image / PSNR Value	LSB	Yeung-Mintzer
Pepper	53.2993	52.3850
Baboon	53.2760	53.1092

Form 2.2.2.1 PSNR table of different watermark algorithms of images

In general Y-M algorithm introduces more distortion than LSB, which satisfies my expectation. Since LSB algorithm only modifies the LSB of the image, pixel values are modified by at most 1. For Y-M algorithm, since corresponding pixel values needs to be modified to nearest neighbor in the look up table, which is possible to cause difference from original pixel value be greater than 1, so that it introduces more distortion to the image and PSNR is slightly lower.

2.3 MATLAB Function to Extract Y-M Watermark

```
function waterMark = extractYMmark(markedImage, key)
% Input:
%     markedImage: the image contains water mark
%     key: the key to generate random look up table
% Return:
%     waterMark: extracted binary water mark

    % Seed the random number generator with the user specified key
    rng(key);
    % Generate look up table values
    LUTvals = rand(1, 256) > .5;
    waterMark = zeros(size(markedImage));

    for i = 1 : size(markedImage, 1)
```

```
        for j = 1 : size(markedImage, 2)
            % Get binary value by look up table.
            waterMark(i, j) = LUTvals(markedImage(i, j) + 1);
        end
    end

end
```

2.3.1 Extraction of Y-M Watermark from Image

The extracted watermark is shown below, by setting the key to 435.

Figure 2.3.1.1 Extracted watermark

2.4 Robustness Test & Analysis

In this section, bottom half of the LSB and Y-M watermarked baboon images are replaced with top half of original pepper image. Extracted watermarks from 2 versions of mixed images are shown below. Key is set to 1.

Figure 2.4.1 Watermark from LSB image

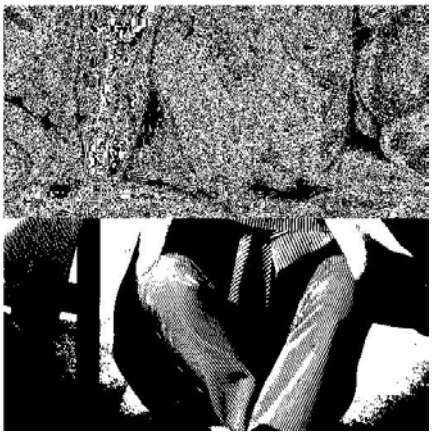

Figure 2.4.2 Watermark from Y-M image

Both of the watermarks are destroyed, while extracted Y-M watermark shows a vague outline of the bottom half of the image for replacement.

To design an attack to preserve LSB embedded watermark, the goal can be easily achieved by extracting LSB bit plane of the marked image first, then embed the mark again after combining of 2 images.

For Yeung-Mintzer watermark, it is impossible to extract the watermark without access to the look up table, or the key to generate the table. So under the assumption described in the problem, it is impossible to preserve the Y-M watermark after the combination of 2 images.

3. Spread Spectrum Watermarking

3.1 MATLAB Function to Embed Spread Spectrum Watermark

```
function markedImage = spectrumWaterMark(image, key, N, alpha)
% Input:
%     image: original image
%     key: key to generate random gaussian water mark
%     N: number of N largest DCT coefficients
%     alpha: water mark embedding strength
% Return:
%     markedImage: water marked image

    [numRow, numCol] = size(image);
    % Perform DCT on original image.
    image = dct2(image);
    image = reshape(image, 1, []);
    markedImage = image;

    % Index are recordered in descending order of DCT coefficients.
    [~, order] = sort(reshape(image, 1, []), 'descend');

    % Seed the random number generator with the user specified key.
    rng(key);
    % Generate i.i.d. Gaussian watermark with mean = 0 and variance = 1.
    wmk = randn(N, 1);

    for i = 1 : N
        % Mark the N highest DCT coefficients excluding DC coefficient.
        markedImage(order(i+1)) = image(order(i+1)) * ...
                                            (1 + alpha * wmk(i));
    end

    markedImage = reshape(markedImage, [numRow numCol]);
    markedImage = uint8(idct2(markedImage));

end
```

3.2 MATLAB Function to Extract Spread Spectrum Watermark

```
function similarityScore = extractSpectrumWaterMark(markedImage, image, N,
alpha, key)
% Input:
%     markedImage: the image containing water mark
%     image: the original reference image
%     N: identify N largest DCT coefficients
%     alpha: embedding strengh
%     key: key to generate random gaussian water mark
% Return:
%     similarityScore: the similarity between extracted water mark
```

```
%                       and original water mark

    % Get DCT of both original and water marked image.
    markedImage = dct2(markedImage);
    markedImage = reshape(markedImage, 1, []);
    image = dct2(image);
    image = reshape(image, 1, []);

    % Get index in descending order of original image coefficients.
    [~, order] = sort(image, 'descend');

    % Seed the random number generator with the user specified key.
    rng(key);
    % Generate i.i.d. Gaussian watermark with mean = 0 and variance = 1.
    oriWmk = randn(N, 1);

    waterMark = zeros(N, 1);
    for i = 1 : N
        % Extracting water mark.
        waterMark(i) = (1 / alpha) * ...
                        (markedImage(order(i+1)) / image(order(i+1)) - 1);
    end

    % Calculating similarity score.
    similarityScore = waterMark' * oriWmk / sqrt(waterMark' * waterMark);

end
```

3.3 Analysis of Spread Spectrum Watermark

The key is set to 1. After calling the function of embedding watermark, the PSNR of marked image and original image is **45.8712**. Then the extraction function is called, and the calculated similarity score of extracted watermark and embedded watermark is **31.5535**. The output similarity score is definitely greater than the detection threshold, 7, described in this problem.

The following codes are MATLAB script to execute random key watermark comparison. Output score table is also provided.

```
% Try 5 different random watermarks and observe similarity score.
similarityTests = zeros(1, 5);
for i = 1 : 5
    rng(key);
    % Original key, 1, is excluded.
    key = randi([2, 2^30], 1, 1);
    similarityTests(i) = extractSpectrumWaterMark(baboonWMK, ...
                                        baboonImage, N, alpha, key);
end
```

Iteration	1	2	3	4	5
Similarity	-1.3800	1.3116	0.4098	1.4422	-0.1110

None of 5 tries generates similarity score greater than 7. This algorithm does not generate false alarm under this circumstance.

3.4 Robustness Test & Analysis

In this section, the watermarked image is performed by certain signal processing. Similarity scores are calculated with original watermark, along with 5 random generated watermarks.

3.4.1 JPEG Compression with Different Quality Factors

Following table contains similarity scores with different watermarks under different quality factors.

Watermarks	Original	Random 1	Random 2	Random 3	Random 4	Random 5
90% Quality	30.9963	-1.3759	1.0549	0.1854	1.8393	0.1398
75% Quality	28.9148	-1.1122	0.8292	-0.3779	1.6189	0.2761
50% Quality	26.9563	-0.6657	0.4448	-0.1352	1.7134	-0.2449

For all 3 JPEG compression processes, the algorithm successfully detects correct watermark. Among 5 random watermarks each time, no false alarms are given, i.e., all similarity scores with random watermarks are less than 7.

3.4.2 Image Smoothing Process

Following table contains the similarity scores between original watermark and watermarks after average and median 3*3 filtered image.

Watermarks	Original	Random 1	Random 2	Random 3	Random 4	Random 5
Average	10.2408	-0.3609	0.0555	0.5990	0.3650	0.6592
Median	16.9004	-0.5901	-0.6423	0.1465	0.0575	1.2023

The algorithm is able to detect the same watermark after average and median filtering. Still no false alarm reported.

3.4.3 Gaussian Noise Contamination

Following table contains the similarity scores between original watermark and watermarks after noise contamination.

Watermarks	Original	Random 1	Random 2	Random 3	Random 4	Random 5
$\sigma = 1$	1.1530	-0.6566	-0.6107	0.1584	0.2307	-0.4386
$\sigma = 5$	0.3551	-0.3483	-0.6647	0.2439	0.2394	-0.5823
$\sigma = 10$	0.0998	-0.2985	-0.6719	0.2856	0.3110	-0.5910

Spread spectrum watermarks are not recognizable after Gaussian noise contamination. As the σ of Gaussian distribution increases, the similarity scores with original watermark becomes closer to 0. Still, there are no false alarms.

According to the pattern of contaminated watermarked image, I believe when σ is controlled within an appropriate small range, the watermark is still recognizable.

Being not able to recognize watermark is reasonable since all given σ values makes the image extremely distorted, and readers cannot identify any object in that image.

3.4.4 Mixture with Unmarked Images

In this section, the bottom half of watermarked image is replaced by bottom half of unmarked image. The following table contains the similarity score with original and processed watermark.

Watermarks	Original	Random 1	Random 2	Random 3	Random 4	Random 5
Mixture	31.5535	-1.3800	1.3116	0.04098	1.4422	-0.1110

Robustness in this section is exceptional since the similarity scores among all watermarks remain exactly the same as that before mixing unmarked image. This algorithm successfully detects same watermark without any false alarm.

3.5 Finding the Best Key

3.5.1 MATLAB Script to Choose Best Key

```
% Read marked image and reference image, and try 100 different keys
% to find the most fit key to this water mark.
ssMarkedImage = uint8(imread('streamSSWmked.tiff'));
ssOriImage = uint8(imread('stream.tiff'));
maxSimilarity = -inf;
bestKey = 0;
for i = 1 : 100
    key = i;
    similarity = extractSpectrumWaterMark(ssMarkedImage, ...
                                          ssOriImage, N, alpha, key);
    if similarity > maxSimilarity
        maxSimilarity = similarity;
        bestKey = key;
    end
end
```

3.5.2 Choice of Key

After running the script above, the best key chosen by the algorithm is **18**.

This key generates the highest similarity score among all keys between 1 and 100, and the score is **30.1133**, which is convincing since the score is significantly greater than 7.