ECES 435

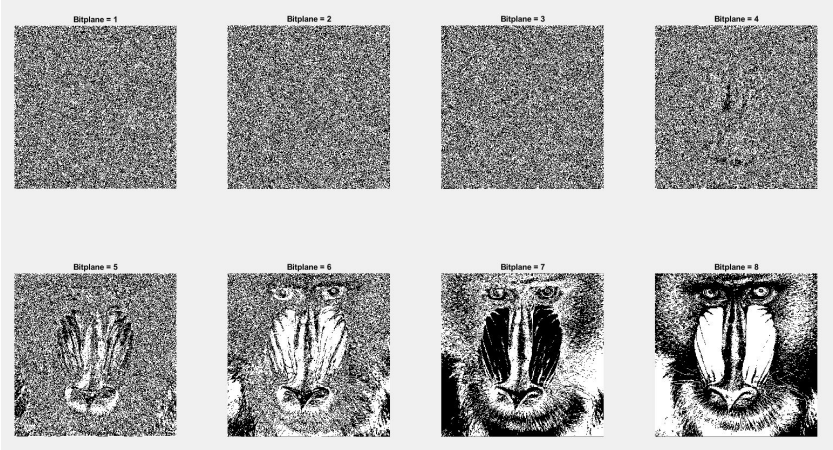
Professor Matthew Stamm

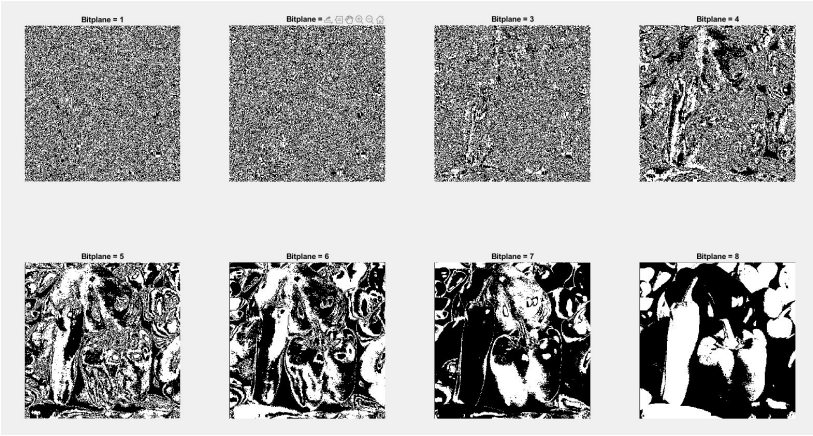
Assignment 3

Wanyu Li and John R Seitz IV

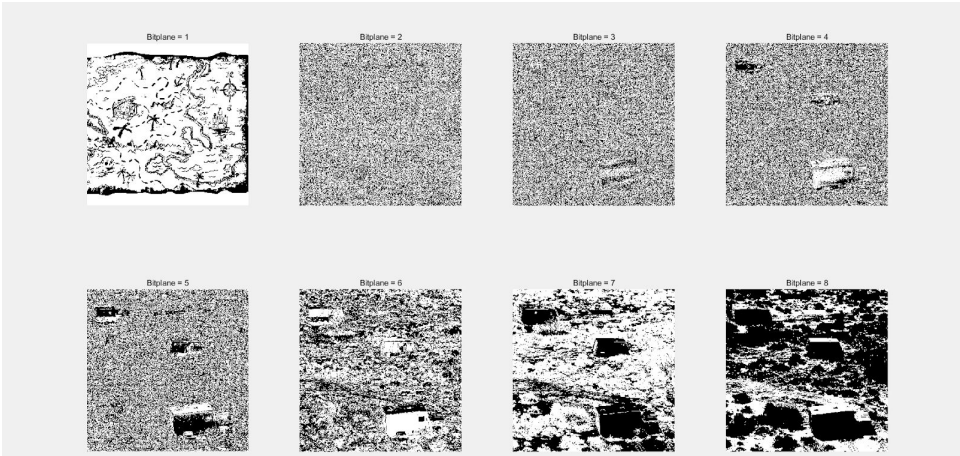
Part 1: Least Significant Bit Data Hiding

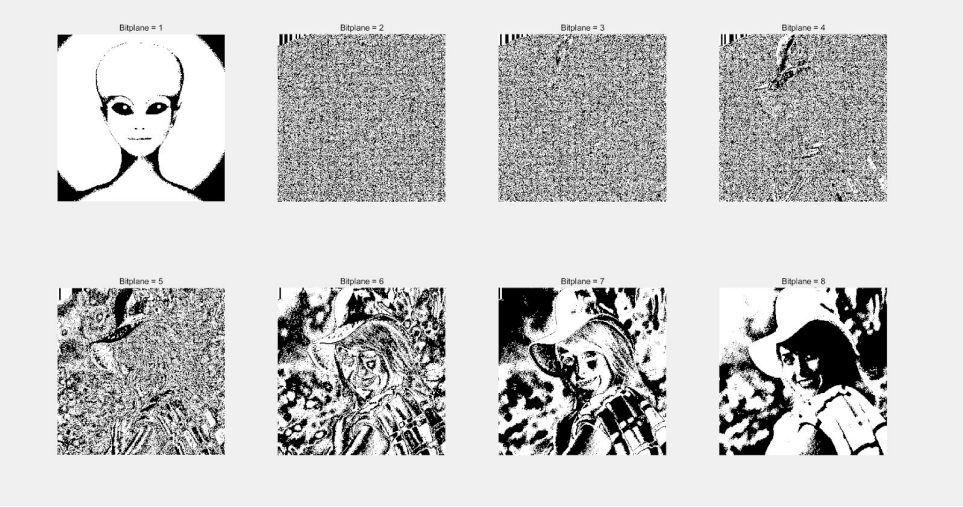
Below are the 8 bitplanes for the Baboon and Peppers images. In the “Baboon” image, the 5th bit plane and below no longer resemble the image content, but it can be seen in the 6, 7, 8 bit planes. Whereas for the “Peppers” image, the 4th bitplane is when the image content is no longer visible. They are slightly different due to images’ the frequency content, resolution, and other factors.



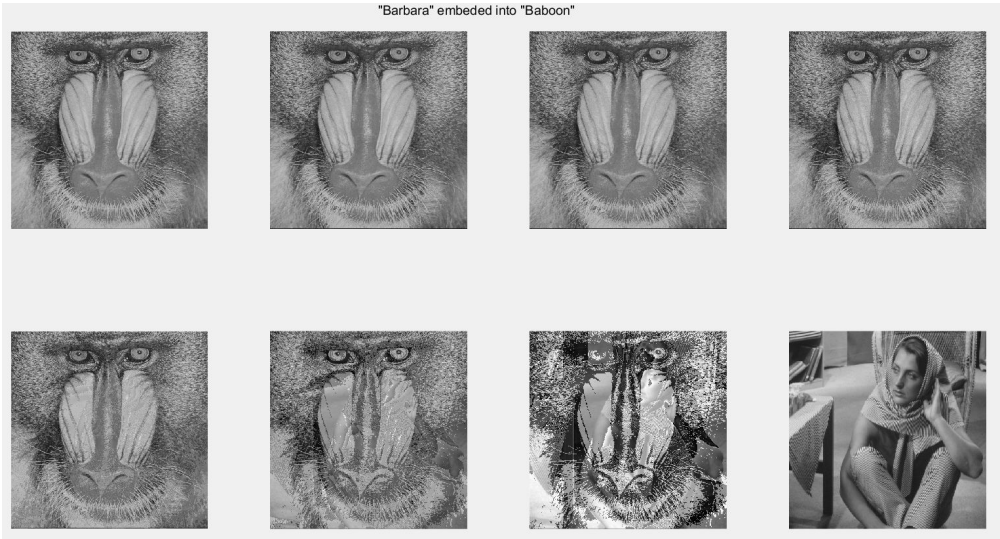


For the images LSBwmk1, LSBwmk2, LSBmk3, all three images have hidden content but in different bit planes. LSBwmk1 has the hidden content of a Drexel University logo at the 2nd bit plane. LSBwmk2 has the hidden content of a treasure map at the 1st bit plane. LSBwmk3 has the hidden content of an alien at the 1st bit plane. All 3 images' bit planes can be seen below.



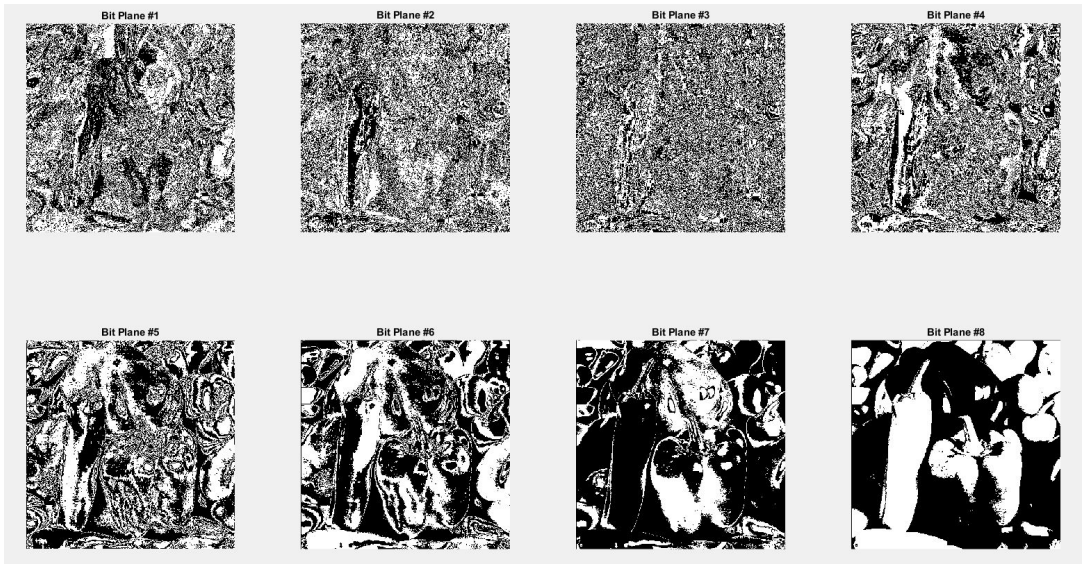


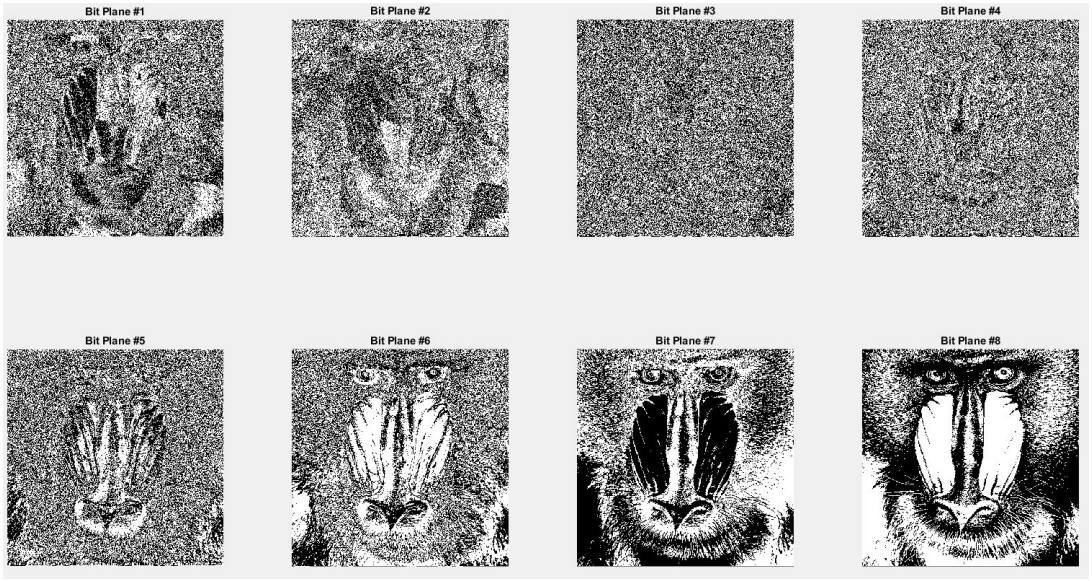
The top N bit planes of the “Barbara” image were then embedded in the “Peppers” and “Baboon” images’ bit planes. In Peppers, distortion is not clearly noticed until the 4th bit plane, and the hidden content of Barbara is not obvious until the 7th bit plane. Whereas in Baboon, distortion is not clearly noticed until the 5th bit plane and the hidden content of Barbara is not obvious until the 8th (last) bit plane. While they are very similar, the small differences come from the host images’ properties: mostly the varying frequency content.



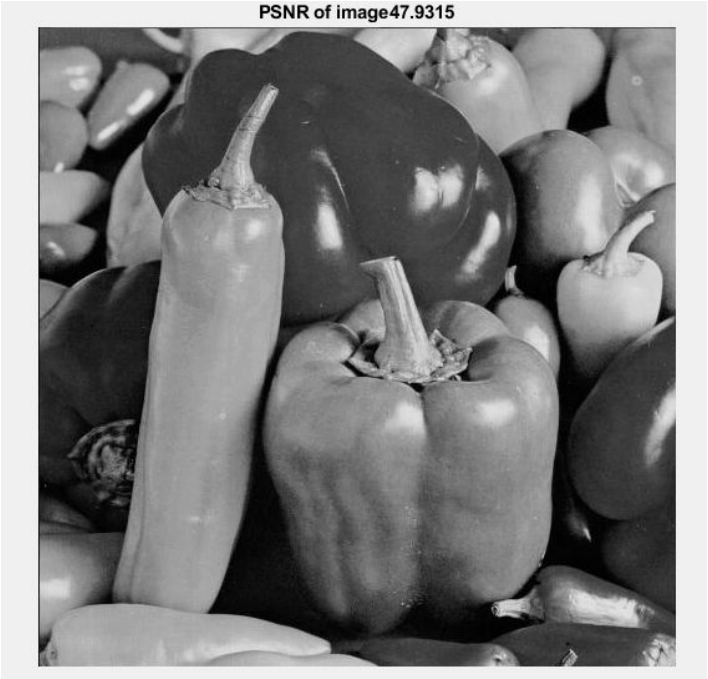
Part 2: Yeung-Mintzer Watermarking

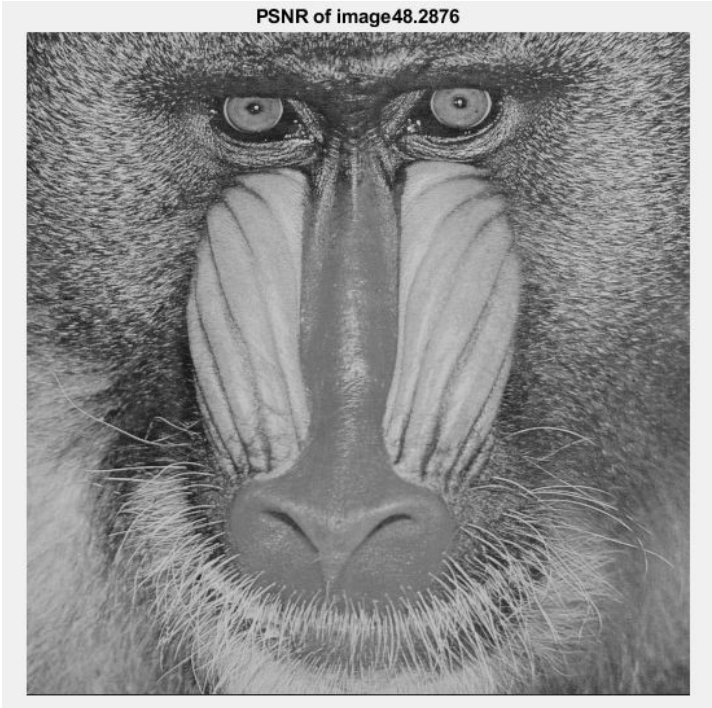
After performing the Yeung-Mintzer watermarking on both Peppers and Baboon images, the results can be seen below. The Barbara watermark is not visually detectable in both images, but a slight silhouette of it can be seen in the Pepper's image first bit plane.



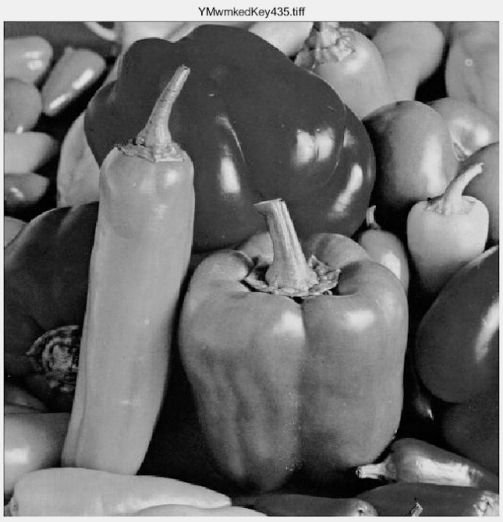


As seen in the images below, the Peak Signal to Noise Ratio (PSNR) of the Peppers image vs watermarked is 47.93, and the PSNR of the Baboon image vs watermarked is 48.28. These values are expectedly lower than that of the algorithm because it makes use of the lookup table values and it is an improved watermark technique.





Below is the extracted watermark from the given “YMwmkedKey435” image. It is an alternating sequence of Drexel logos with different black and white compositions.



First, the top half of the baboon image was replaced with the bottom half of the peppers image so that both images' bottoms are together. This is seen in the image below.

The extracted watermark of these images can be seen below. The baboon image includes the LSB Barbara watermark, and the peppers image watermark cannot be seen visually identified.

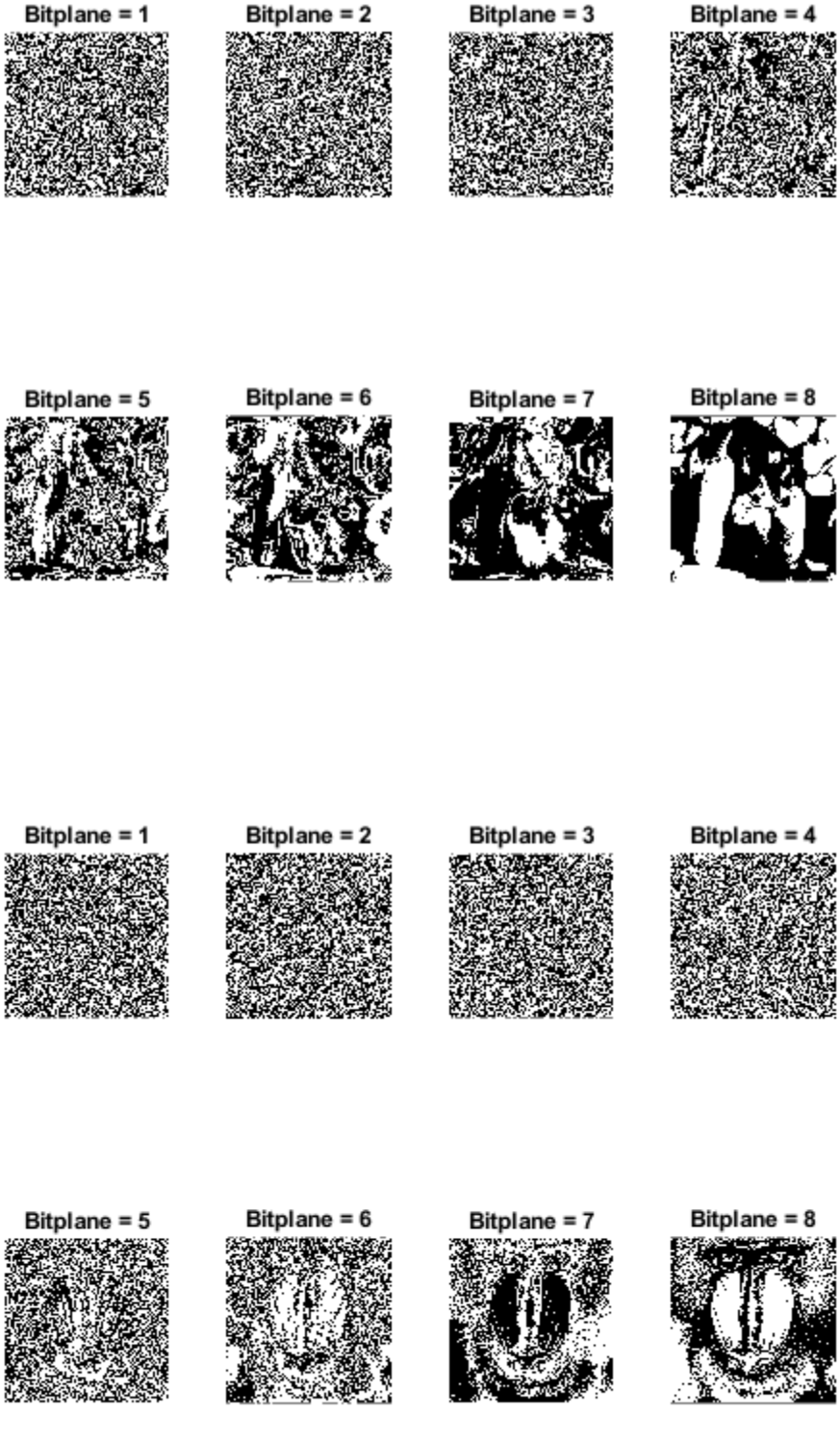
After designing an attack, the top half of the baboon image was replaced with the bottom half of the peppers image, was seen on the left side. From this, you will then see the LSB watermark of the Barbara image in both, as seen on the right side. This same attack could be done on the Yeung-Mintzer watermarked image if the attacker knew the key and look up table values. Otherwise, it should not be possible for this attack to be done.

Table of Contents


```
%ECES435 Assignment 3 Part 1 - By Wanyu Li and John Seitz
close all; clear all; clc;
```

Part 1 - 1st Task - LSB Least Significant Bit-plane Hiding

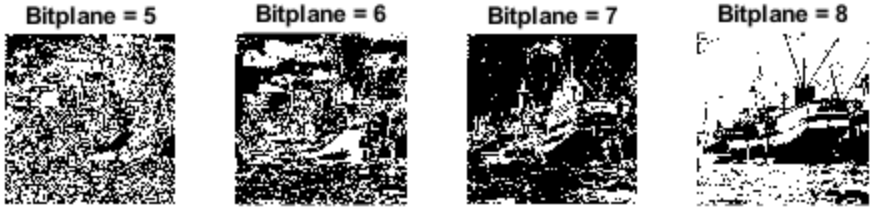
```
%Part 1 - 1st Task
P = imread('peppers.tif'); %Read in image as unit8
B = imread('baboon.tif'); %Read in image as unit8

figure (1);
for I = 1:8
P_Bitplane = get_bitplane(P,I); %extract 1st bitplane of image P,
 using function
subplot(2,4,I)
imshow(P_Bitplane) %show bitplane
title (['Bitplane = ',sprintf('%d',I)])
end

figure (2);
for I = 1:8
B_Bitplane = get_bitplane(B,I); %extract 1st bitplane of image P,
 using function
subplot(2,4,I)
imshow(B_Bitplane) %show bitplane
title (['Bitplane = ',sprintf('%d',I)])
end
```


Bitplane = 1
Bitplane = 2
Bitplane = 3
Bitplane = 4
Bitplane = 5
Bitplane = 6
Bitplane = 7
Bitplane = 8
Bitplane = 1
Bitplane = 2
Bitplane = 3
Bitplane = 4
Bitplane = 5
Bitplane = 6
Bitplane = 7
Bitplane = 8

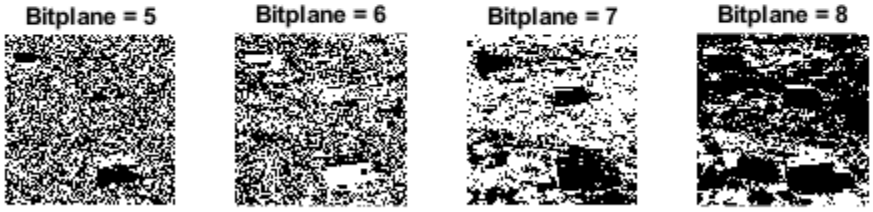
Part 1 - 2nd Task - Examine bitplanes of images

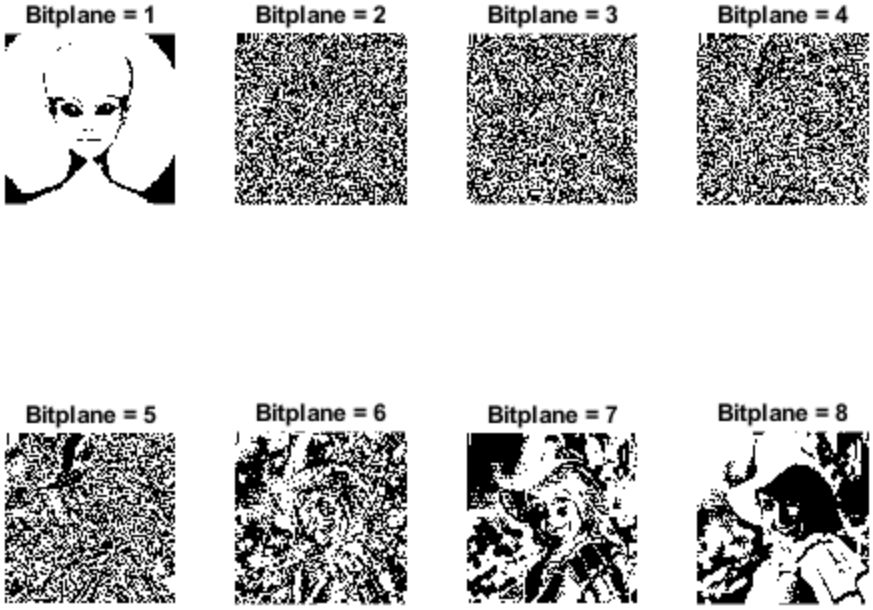
```
wmk1 = imread('LSBwmk1.tiff'); %Read in image as unit8
wmk2 = imread('LSBwmk2.tiff'); %Read in image as unit8
wmk3 = imread('LSBwmk3.tiff'); %Read in image as unit8



figure(3)

for I = 1:8
wmk1_Bitplane = get_bitplane(wmk1,I); %extract 1st bitplane of image
 P, using function
subplot(2,4,I)
imshow(wmk1_Bitplane) %show bitplane
title (['Bitplane = ',sprintf('%d',I)])
end

figure(4)
for I = 1:8
wmk2_Bitplane = get_bitplane(wmk2,I); %extract 1st bitplane of image
 P, using function
subplot(2,4,I)
imshow(wmk2_Bitplane) %show bitplane
title (['Bitplane = ',sprintf('%d',I)])
end

figure(5)

for I = 1:8
wmk3_Bitplane = get_bitplane(wmk3,I); %extract 1st bitplane of image
 P, using function
subplot(2,4,I)
imshow(wmk3_Bitplane) %show bitplane
title (['Bitplane = ',sprintf('%d',I)])
end
```


Bitplane = 1
Bitplane = 2
Drexel
UNIVERSITY
Bitplane = 3
Bitplane = 4


Bitplane = 5
Bitplane = 6
Bitplane = 7
Bitplane = 8


Bitplane = 1
Bitplane = 2
Bitplane = 3
Bitplane = 4


Bitplane = 5
Bitplane = 6
Bitplane = 7
Bitplane = 8

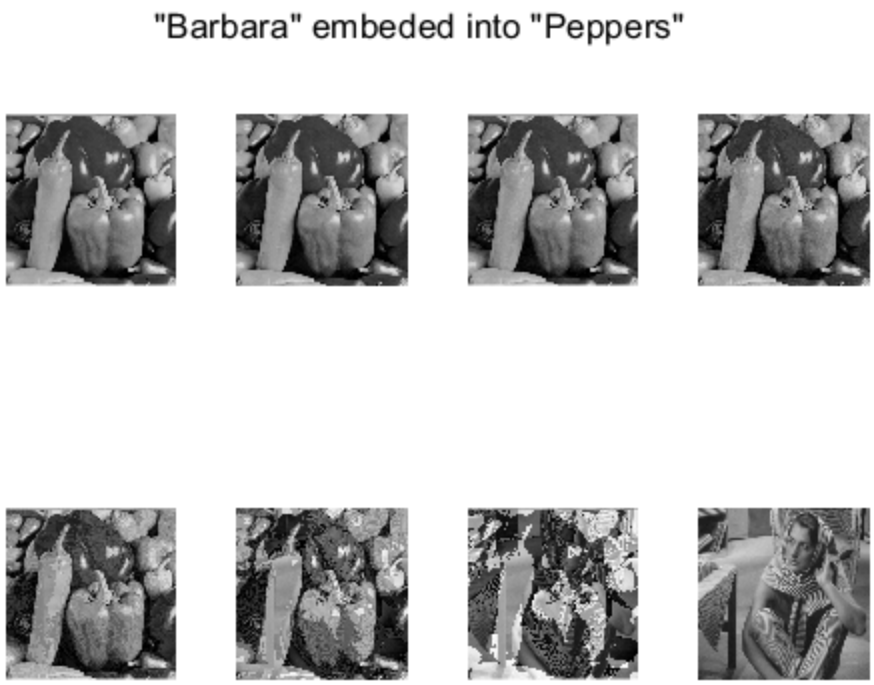


Part 1 - 3rd Task - Embed one image bitplanes into another's

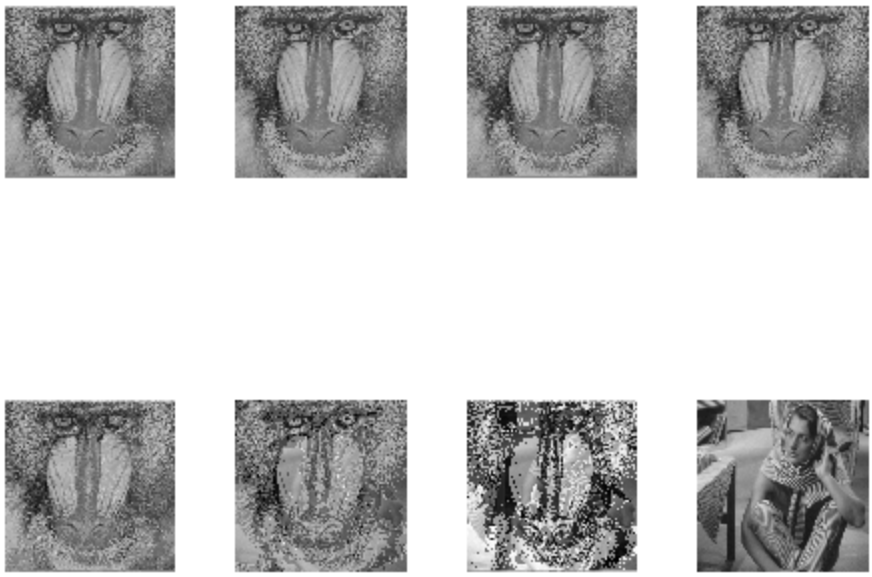
```
Pep = imread('peppers.tif'); %Read in image as unit8
Bab = imread('baboon.tif'); %Read in image as unit8
Barb = imread('Barbara.bmp'); %Read in image as unit8


%Use watermark function to put the bitplanes of Barbera into the
 bitplanes
%of Peppers


figure(6)


subplot(2,4,1)
newimg = watermark2(Pep,Barb,1,1);
imshow(newimg)

subplot(2,4,2)
newimg = watermark2(newimg,Barb,2,2);
imshow(newimg)
```

```
subplot(2,4,3)
newimg = watermark2(newimg,Barb,3,3);
imshow(newimg)

subplot(2,4,4)
newimg = watermark2(newimg,Barb,4,4);
imshow(newimg)

subplot(2,4,5)
newimg = watermark2(newimg,Barb,5,5);
imshow(newimg)

subplot(2,4,6)
newimg = watermark2(newimg,Barb,6,6);
imshow(newimg)

subplot(2,4,7)
newimg = watermark2(newimg,Barb,7,7);
imshow(newimg)

subplot(2,4,8)
newimg = watermark2(newimg,Barb,8,8);
imshow(newimg)

sgtitle('"Barbara" embeded into "Peppers"')
%Do the same for the Baboon Image
figure(7)

subplot(2,4,1)
newimg = watermark2(Bab,Barb,1,1);
imshow(newimg)

subplot(2,4,2)
newimg = watermark2(newimg,Barb,2,2);
imshow(newimg)

subplot(2,4,3)
newimg = watermark2(newimg,Barb,3,3);
imshow(newimg)

subplot(2,4,4)
newimg = watermark2(newimg,Barb,4,4);
imshow(newimg)

subplot(2,4,5)
newimg = watermark2(newimg,Barb,5,5);
imshow(newimg)

subplot(2,4,6)
newimg = watermark2(newimg,Barb,6,6);
imshow(newimg)

subplot(2,4,7)
newimg = watermark2(newimg,Barb,7,7);
```

```
imshow(newimg)

subplot(2,4,8)
newimg = watermark2(newimg,Barb,8,8);
imshow(newimg)

sgtitle('"Barbara" embeded into "Baboon"')
```



"Barbara" embeded into "Baboon"



Functions Below

```
type watermark2.m
type get_bitplane.m


function [newimg] = watermark2(img1,img2,NIbp,WMbp)
%Function to replace a specific bitplane of one image with another
%specified bitplane of another image

%img1 = double(img1);
%img2 = double(img2); %watermark

bp = get_bitplane(img2,WMbp);
newimg = bitset(img1,NIbp,bp);

end

function [Bitplane_img] = get_bitplane(img,bitplane)
%This function outputs the specified bitplane of the input img
img = double(img); %Convert unit8 image to double
Bitplane_img = bitget(img,bitplane); %Extract specified bit plane from
 img
end
```

Published with MATLAB® R2019b

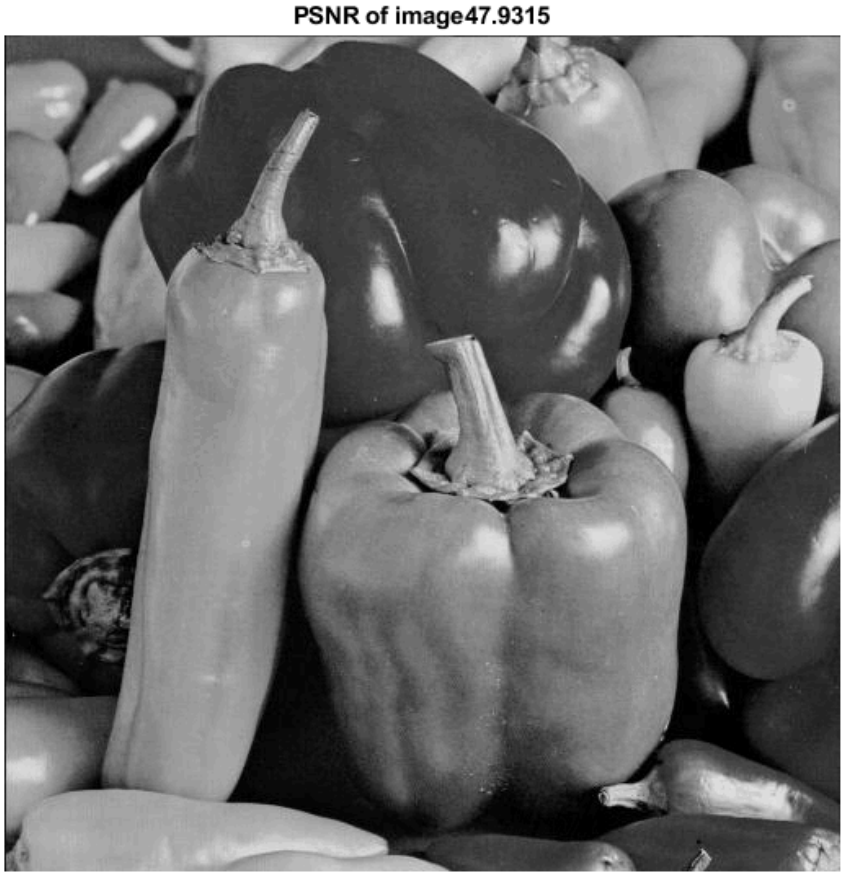
Table of Contents


```
%ECES435 Assignment 3 Part 2 - By Wanyu Li and John Seitz
close all; clear all; clc;
```

Part 2 - 1st Task - Yeung-Mintzer embedding function to hide the most signicant bit plane of the Barbara image in both the peppers and baboon images

```
imgs = {'peppers.tif','baboon.tif'}; % Read in the two host images,
 peppers and baboons
wmkimages = uint8(zeros(512,512,length(imgs)));

for i = 1:length(imgs)
    img = imread(imgs{i});
    wmk = get_bitplane(imread('Barbara.bmp'),8); % Read in most
 significant bitplane of watermark image to embed in others

    figure(2*i-1) % create figures of length i*2-1
    key = 0; % initialize key value to zero
    [wmkimages(:,:,i)] = YME(img,wmk,key); %Use the created function
 to implement the Yeung-Mintzer altorithm to embed a binary watermark
 in an image
    imshow(wmkimages(:,:,i)) % display the new watermarked images

    imgs{i}
    peaksnr = psnr(wmkimages(:,:,i),img) % calculate the psnr of the
 watermark images vs the original
    title(['PSNR of image',num2str(peaksnr)])
    figure(2*i)
    splitimg(wmkimages(:,:,i));
end

imwrite(wmkimages(:,:,1),'peppers_ymwmk.tiff','tiff'); % save the
 Yeung-Mintzer watermarked image as tiff
imwrite(wmkimages(:,:,2),'baboon_ymwmk.tiff','tiff');  % save the
 Yeung-Mintzer watermarked image as tiff
```

```
ans =

    'peppers.tif'

peaksnr =

   47.9315

ans =

    'baboon.tif'

peaksnr =

   48.2876
```



Bit Plane #1

Bit Plane #2

Bit Plane #3

Bit Plane #4

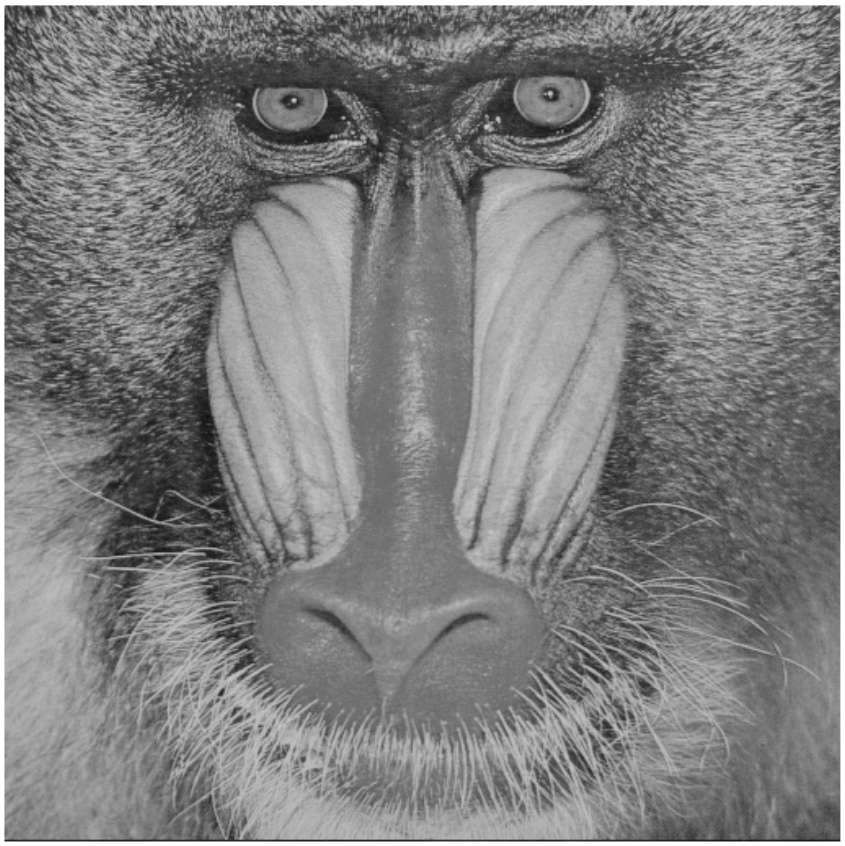
Bit Plane #5

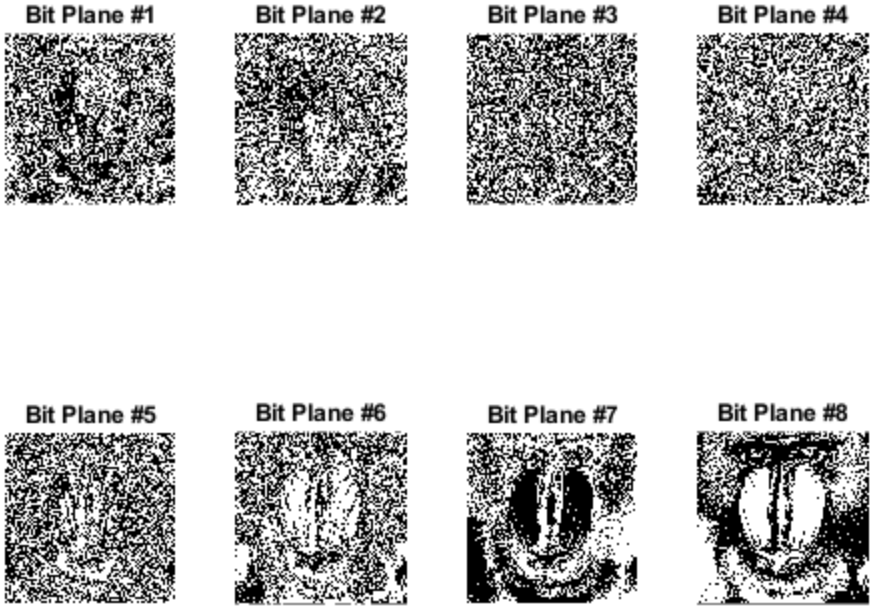
Bit Plane #6

Bit Plane #7

Bit Plane #8

PSNR of image48.2876





Part 2 - 2nd Task - Extract the watermark embedded in the given image using key 435

```
figure(2*i+1) % create figures of length i*2+1

ym_img = imread('YMwmkedKey435.tiff');
newimg = YMD(ym_img,435); % Use the specified key in the assignment =
 435

subplot(1,2,1)
imshow(ym_img)
title('YMwmkedKey435.tiff')

subplot(1,2,2)
imshow(newimg)
title('Extracted Watermark')
```

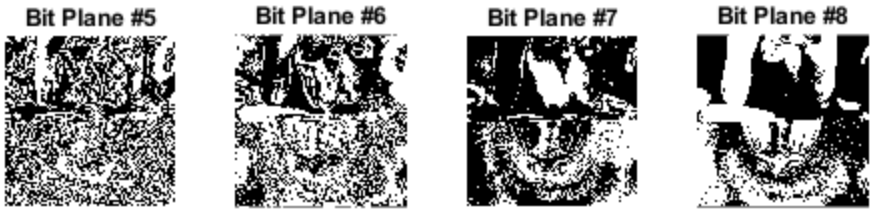
YMwmkedKey435.tiff

Extracted Watermark

Part 2 - 3rd Task - Extract the watermark embedded in the given image using key 435

```
baboon_ymwmk = imread('baboon_ymwmk.tiff'); % Read in Yeung-Mintzer
 watermarked image
peppers_ymwmk = imread('peppers_ymwmk.tiff'); % Read in Yeung-Mintzer
 watermarked image

bab = imread('baboon.tif'); % Read in original host images
pep = imread('peppers.tif'); % Read in original host images

wmk = imread('Barbara.bmp'); % Read in Watermark image


pep_lsb = watermark_2(pep, wmk,3);
bab_lsb = watermark_2(bab, wmk,3);
both_lsb = [pep_lsb(257:512,:);bab_lsb(257:512,:)];

figure(6)
imshow(both_lsb)
title('LSB Watermark Manipulation')

figure(7);
splitimg(both_lsb);
```

LSB Watermark Manipulation



Yeung-Mintzer watermarked image

```
both_img = [pep(257:512,:);baboon_ymwmk(257:512,:)]; % Image with both
 halves put together
wmk_img = YMD(both_img,0);

figure(8)
subplot(1,2,1)
imshow(both_img)
title('Peppers and Baboons Watermark')
subplot(1,2,2)
imshow(wmk_img);

test_ym = [peppers_ymwmk(257:512,:);baboon_ymwmk(257:512,:)]; % Test
 out an attack with the Yeung-Mintzer
YME_wmk = YMD(test_ym,0);

figure(9)
subplot(1,2,1)
imshow(test_ym)
title('Peppers Watermark and Baboons Watermark')
subplot(1,2,2)
imshow(YME_wmk);
```

Peppers and Baboons Watermark

Peppers Watermark and Baboons Watermark

Functions Below

```
type YME.m
type YMD.m
type splitimg.m
type pixcorrect.m
```

```
function [newimg] = YME(img,wmk,key)
% This function implements the Yeung-Mintzer altorithm to embed a
 binary watermark in an image.
%For this function to work properly,the host image and watermark image
 must have the same size
%%
rng(key); %Use the rng command in Matlab
LUTvals = rand(1,256) > .5; %to generate an array of 256 uniformly
 distributed random numbers, then compare them to a threshold of 0.5
%%
[x,y] = size(img);
zz = zeros(x,y);
for i = 1:x
    for j = 1:y
        zz(i,j) = LUTvals(img(i,j)+1); % Extract the watermark image
        W(i,j) = wmk(i,j);
        if (zz(i,j) ~= W(i,j)) % use if statement to compare both
 images
            img(i,j) = pixcorrect(img(i,j),LUTvals); %use the
 pixcorrect function to perform pixel correction
        end
    end
end
newimg = img;
end
```

```
function [ZZ] = YMD(img,key)
% This function implements the Yeung-Mintzer altorithm to decode/
extract a binary watermark in an image.

rng(key); %Use the rng command in Matlab
LUTvals= rand(1,256) > .5; %to generate an array of 256 uniformly
 distributed random numbers, then compare them to a threshold of 0.5

[x,y] = size(img);
ZZ = zeros(x,y);
% Use look up table to decode the Yeung-Mintzer image
for i = 1:x
    for j = 1:y
        ZZ(i,j) = LUTvals(img(i,j)+1); % By adding 1, the range is
 changed to 1-256, the same range as the look up value table
    end
end
```

```
end


function [] = splitimg(img)
% This function splits a specific image into bitplanes

for i = 1:8
    newimg = get_bitplane(img,i); %split each bit plane and then
    subplot(2,4,i); % plot each bitplane in a 2x4
    imshow(newimg); % display the bitplane image
    title(['Bit Plane #',num2str(i)]); %title!
end
end


function [newpix] = pixcorrect(oldpix,LUTvals)
% This function corrects by using the look up table values going from
 right to left

    oldpix = oldpix + 1; %correct the range to 1-256 to match the look
 up table values
    newvalue = ~LUTvals(oldpix);  % assign the new desired value by
 interverting the original
    P = 1;
    while 1
        if (oldpix+P <= 256) && (LUTvals(oldpix+P) == newvalue)
            newpix = oldpix + P-1; % reset the range to the original
 0-255
            break;
        elseif (oldpix-P >= 1)&&(LUTvals(oldpix-P) == newvalue)
            newpix = oldpix - P-1;
            break;
        else
        P=P+1;
    end
end
```

Published with MATLAB® R2019b