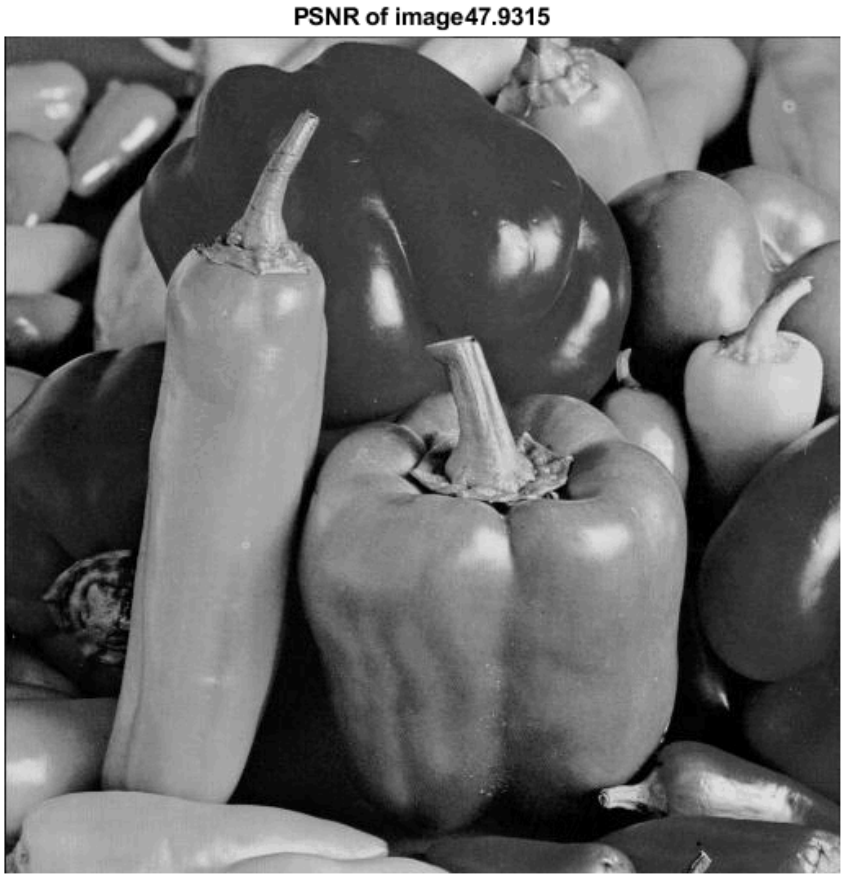
Table of Contents


```
%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

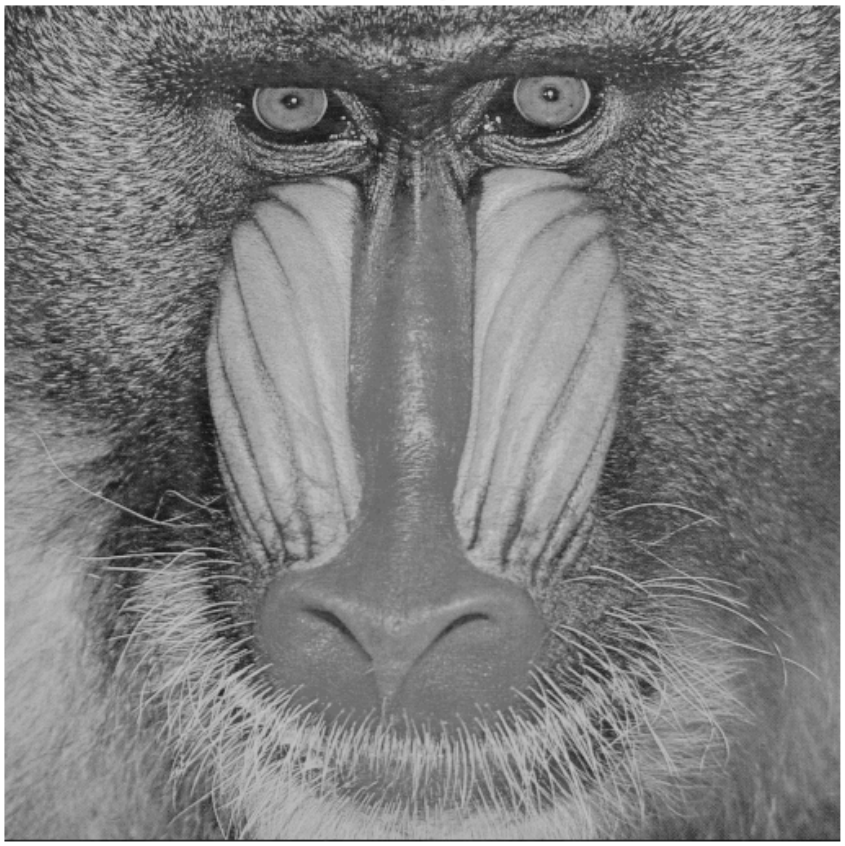
Bit Plane #5

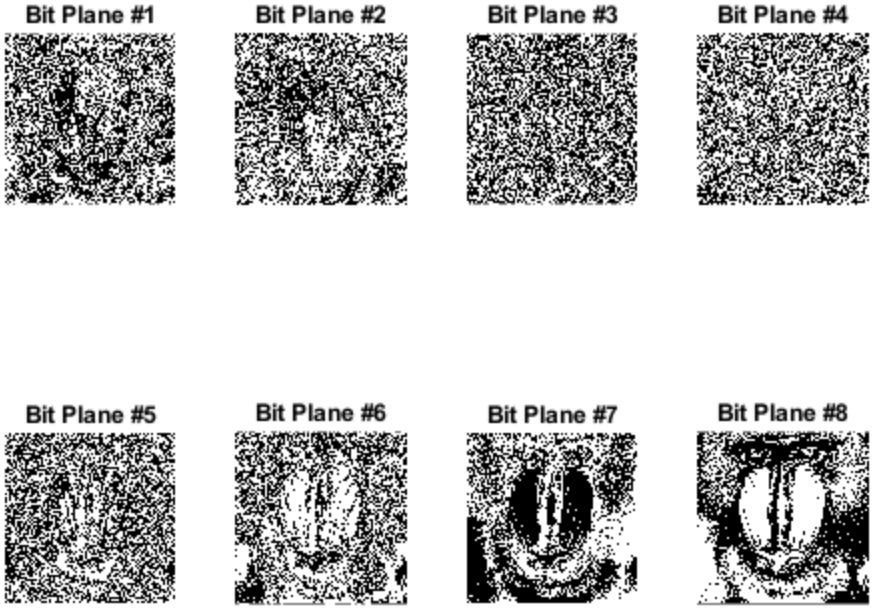
Bit Plane #6

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')
```

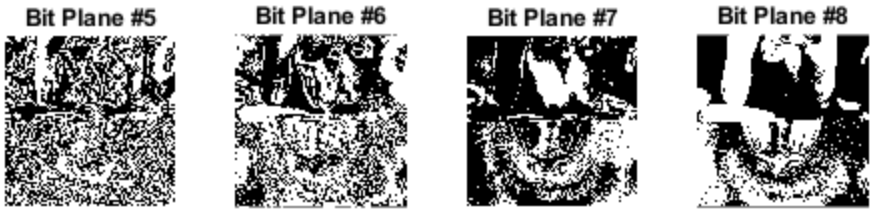
YMwmkedKey435.tiff

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