Kleanthis E Karavangelas, Basem Saleh

Assignment 3, ECES 435

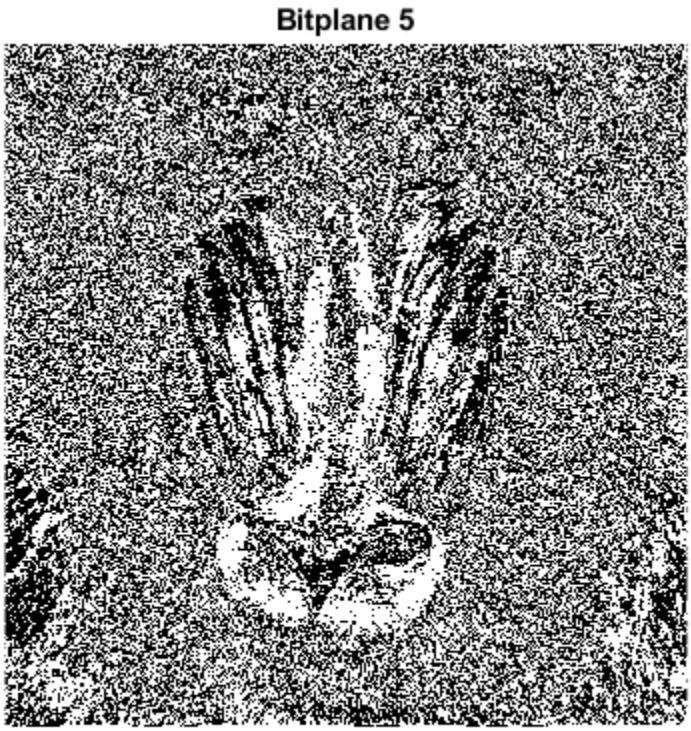
Part 1: Least Significant Bit Data Hiding

peppers bit plane

```
figure(1);
GetBitPlane(4, 'peppers.tif');
```

baboon bit plane

```
figure(2);
GetBitPlane(5, 'baboon.tif');
```



For the peppers.tif image, the highest bit plane that does not resemble image content based on our group's opinion, is bit plane 4. For the baboon.tif image, the highest bit plane that doesn't resemble image content, is bit plane 5, where you can barely recognize the baboon. These are two different pictures, so it makes sense that not each bit plane will represent an equal amount of information. The peppers image has more details such as edges, that can be preserved and seen through a greater amount of bit planes. Instead, the baboon has less of those features, which makes the noticable content be a lot less visible after fewer bit planes.

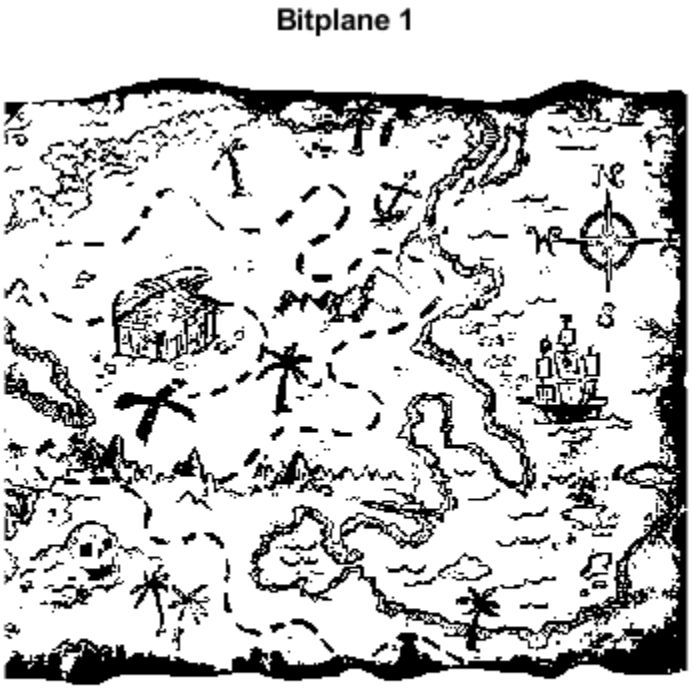
LSBwmk1.tiff

```
figure(3);
I1 = imread('LSBwmk1.tiff');
imshow(I1);

figure(4);
GetBitPlane(2, 'LSBwmk1.tiff');
```

The hidden image is the Drexel logo and is found on bit plane 2.

LSBwmk2.tiff

```
figure(5);
I2 = imread('LSBwmk2.tiff');
imshow(I2);

figure(6);
GetBitPlane(1, 'LSBwmk2.tiff');
```



The hidden image is a treasure map and is found on bit plane 1.

LSBwmk3.tiff

```
figure(7);
I3 = imread('LSBwmk3.tiff');
imshow(I3);

figure(8);
bp = GetBitPlane(1, 'LSBwmk3.tiff');
```

Bitplane 1

The hidden image is an alien and is found on bit plane 1.

LSB

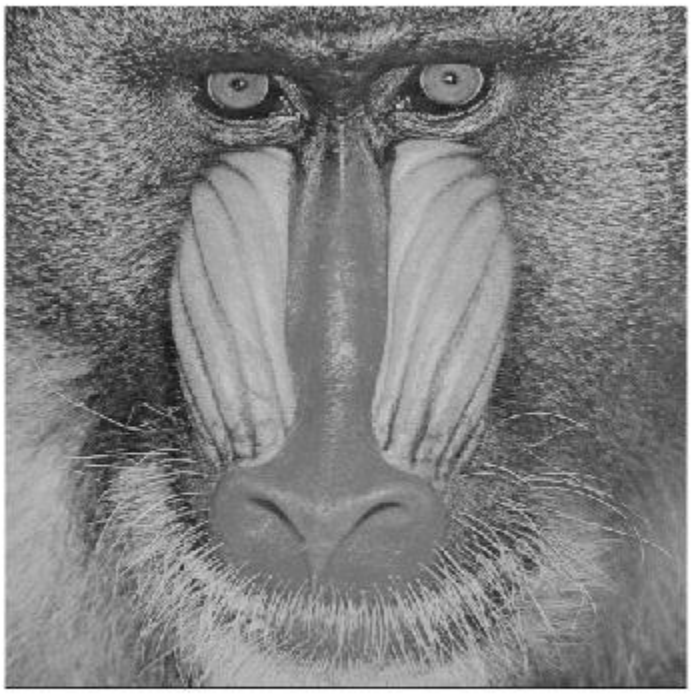
peppers.tif + Barbara.bmp

```
figure(9);
Im = ReplaceNLSB("peppers.tif", "Barbara.bmp", 1);
imshow(Im);
```

For peppers.tif, we start seeing distortion after N = 4. At N = 6, you can start seeing the watermark.

baboon.tif + Barbara.bmp

```
figure(10);
Im = ReplaceNLSB("baboon.tif", "Barbara.bmp", 1);
imshow(Im);
```



For baboon.tif, you start seeing distortion at N = 5. At N = 7, you are able to identify the watermark.

The reason that we see more or less distortion at different bitplanes of the host images, is because the are constructed differently. Our best guess is that the baboon.tif image has more information in the center of the image, and thus makes it harder for us to identify the watermark, which also has more information at the center of the image.

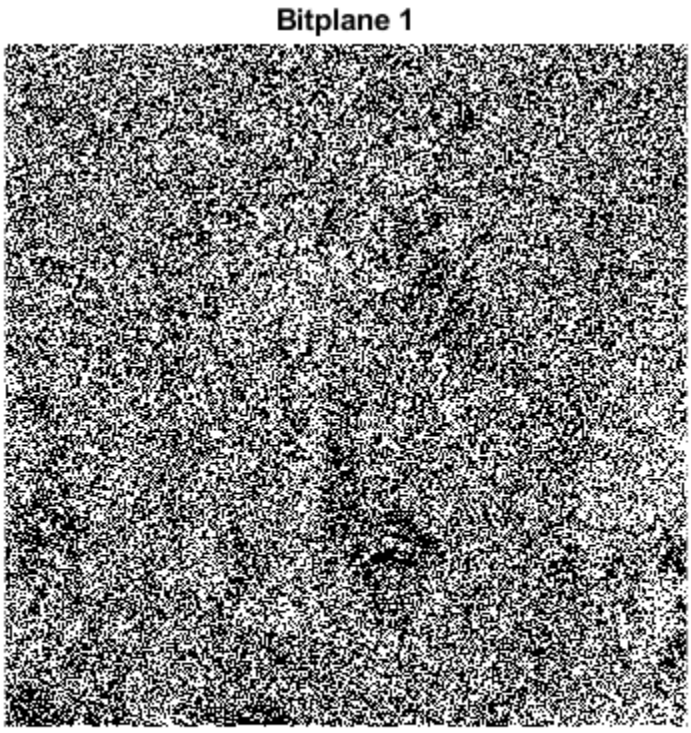
Part 2 Yeung-Mintzer Watermarking

Embed Barbara.bmp in peppers.tif and baboon.tif

```
% Get MSBP of watermark, convert to binary to embed

B = imread("Barbara.bmp");
B = double(B);
B = bitget(B, 8);
B = imbinarize(B);

peps = YMalgorithm('peppers.tif', B, 1);
babs = YMalgorithm('baboon.tif', B, 1);

imwrite(peps, "peps.tif");
imwrite(babs, "babs.tif");
```

Examine bit planes of peps.tif and babs.tif respectively

```
figure(11)
PepperWM = GetBitPlane(1, "peps.tif");
xlabel("Peppers");
figure(12)
BaboonWM = GetBitPlane(1, "babs.tif");
xlabel("Baboon");
```



Baboon

The hidden watermark is not detected in either image, even in the lower bit planes.

PSNR (Yeung-Mintzer algorithm) Between peppers.tif and peps.tif

```
A = imread("peppers.tif");
psnrA = psnr(A, peps);

% Between baboon.tif and babs.tif

B = imread("baboon.tif");
psnrB = psnr(B, babs);
```

PSNR between watermarked and original peppers using YM: 47.8592 PSNR between watermarked and original baboon using YM: 47.8092

PSNR (LSB algorithm) For peppers

```
pepsLSB = ReplaceNLSB("peppers.tif", "Barbara.bmp", 1);
psnrALSB = psnr(A, pepsLSB);

babsLSB = ReplaceNLSB("baboon.tif", "Barbara.bmp", 1);
psnrBLSB = psnr(B, babsLSB);
```

PSNR between watermarked and original peppers using LSB: 51.14.22 PSNR between watermarked and original baboon using LSB: 51.1391

The PSNR is higher when using the LSB method compared to the Yeaung-Mintzer algorithm. This agrees with our initial intuition, since the LSB method in this case only replaces the least significant bit plane of an image, and in contrast the Yeung-Mintzer method replaces many pixel values.

Extraction

```
key = 435;
W = Extraction("YMwmkedKey435.tiff", key);
figure(13);
imshow(W);
```

Last part

```
LSBbabs = ReplaceNLSB("baboon.tif", "Barbara.bmp", 1);
LSBpeps = ReplaceNLSB("peppers.tif", "Barbara.bmp", 1);

B = imread("Barbara.bmp");
B = double(B);
```

```
B = bitget(B, 8);
B = imbinarize(B);

YMbabs = YMalgorithm("baboon.tif", B, 10);
YMpeps = YMalgorithm("peppers.tif", B, 10);
```

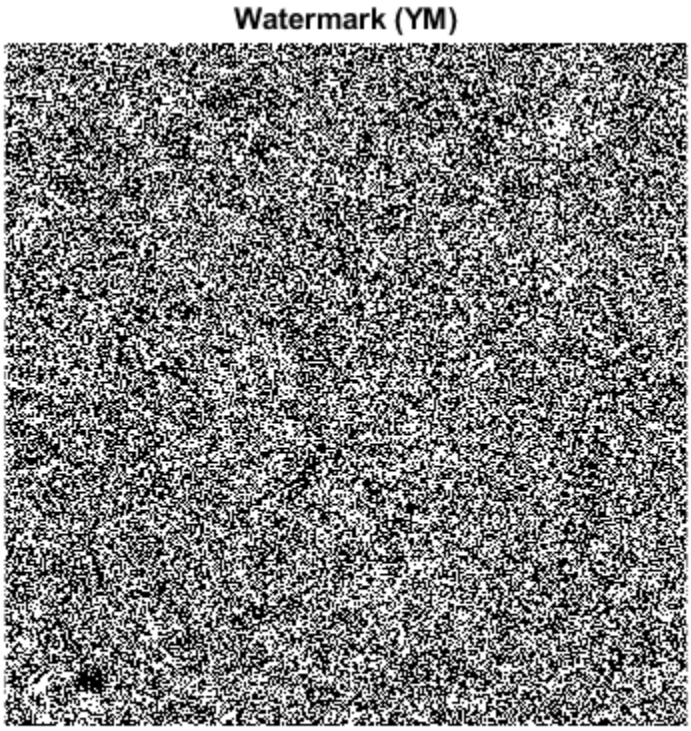
LSB method

```
LSBpeps = double(LSBpeps);
LSBbabs = double(LSBbabs);

% get lower planes of LSBpeps

low = {};
for i=1:4
    low{i} = bitget(LSBpeps, i);
end

% get lower LSBbabs bit planes
low2 = {};
for j=1:4
    low2{j} = bitget(LSBbabs,j);
end

LSBim = cell(1,8);
for i=1:4
    LSBim{i} = low2{i};
    LSBim{i+4} = low{i};
end

for i = 0:7
    LSBim{i+1} = LSBim{i+1} * 2^i;
end

Im = imadd(LSBim{1}, LSBim{2});
Im = imadd(Im, LSBim{3});
Im = imadd(Im, LSBim{4});
Im = imadd(Im, LSBim{5});
Im = imadd(Im, LSBim{6});
Im = imadd(Im, LSBim{7});
Im = imadd(Im, LSBim{8});

Im = uint8(Im);

Im = double(Im);
x = bitget(Im, 1);
figure(14);
imshow(x);
title("Watermark (LSB)")
```

YM method

```
YMpeps = double(YMpeps);
YMbabs = double(YMbabs);

% get lower planes of YMpeps

low = {};
for i=1:4
    low{i} = bitget(YMpeps, i);
end

% get lower YMbabs bit planes
low2 = {};
for j=1:4
    low2{j} = bitget(YMbabs,j);
end

% construct new image
YMim = cell(1,8);
for i=1:4
    YMim{i} = low2{i};
    YMim{i+4} = low{i};
end
```

```
for i = 0:7
    YMim{i+1} = YMim{i+1} * 2^i;
end

Im = imadd(YMim{1}, YMim{2});
Im = imadd(Im, YMim{3});
Im = imadd(Im, YMim{4});
Im = imadd(Im, YMim{5});
Im = imadd(Im, YMim{6});
Im = imadd(Im, YMim{7});
Im = imadd(Im, YMim{8});

Im = uint8(Im);

imwrite(Im, "YMmix.tif");
watermark = Extraction("YMmix.tif", 10);
figure(15);
imshow(watermark);
title("Watermark (YM)")
```



When using the LSB method, the group was able to extract the watermark correctly. When using the YM algorithm, the group was not able to recover the original watermark embedded using the algorithm. The error is attributed to either the YM algorithm we wrote, or the extraction function we created.

```
function bp = GetBitPlane(bp, img)

    % read input image
    I = imread(img);
    % convert to double
    ID = double(I);

    % get specified bitplane
    B = bitget(ID, bp);

    % display specified bitplane
    imshow(B);

    title(sprintf('Bitplane %i', bp));

end
```

```
function Im = ReplaceNLSB(A, B, N)

    % read contents of image A
    A = imread(A);
    A = double(A);

    % read contents of image B
    B = imread(B);
    B = double(B);

    C = {};

    % save top 8-N layers of image A
    for i = N+1:8
        C{i} = bitget(A , i);
    end

    % save N most significant bit planes of image B
    for j = 8-(N-1):8
        D{j} = bitget(B,j);
    end

    % construct the new image array
    ImArray = cell(1,8);
    for i = 1:N
        ImArray{i} = D{8-(N-i)};
    end

    for j = N+1:8
        ImArray{j} = C{j};
    end

    for i = 0:7
```

```
        ImArray{i+1} = ImArray{i+1} * 2^i;
    end

    % add each bitplane to form the new image
    Im = imadd(ImArray{1}, ImArray{2});
    Im = imadd(Im, ImArray{3});
    Im = imadd(Im, ImArray{4});
    Im = imadd(Im, ImArray{5});
    Im = imadd(Im, ImArray{6});
    Im = imadd(Im, ImArray{7});
    Im = imadd(Im, ImArray{8});

    Im = uint8(Im);

end
```

```
function NewIM = YMalgorithm(OI, B, key)

    % develop LUT based on specific key
    rng(key);
    LUTvals = rand(1,256)>0.5;

    % original image
    A = imread(OI);

    % get rows and columns of original image
    [rowsA, colsA] = size(A);

    % get rows and columns of binary watermark image
    [rowsB, colsB] = size(B);

    for i = 1:rowsA
        for j = 1:rowsB

            % identify each pixel value on original image
            pixel = A(i,j);

            % change 0 to 1 so errors are avoided
            if pixel==0
                pixel = 1;
            end

            % flag that helps identify if binary value matches LUT value
            found = 0;
            inc = 1;

            while found == 0
                % if values match, then move on to next pixel
                if B(i,j) == LUTvals(1, pixel)
                    found = 1;
                else
                   % otherwise look for near pixels
                   if B(i,j) == LUTvals(1, pixel+inc)
                       A(i,j) = pixel+inc;
```

```
                        found = 1;
                    elseif B(1,j) == LUTvals(1, pixel-inc)
                        A(i,j) = pixel-inc;
                        found = 1;
                    else
                        inc = inc + 1;
                    end
                end
            end
        end
    end

    NewIM = uint8(A);


end
```

```
function watermark = Extraction(A, key)
    A = imread(A);
    rng(key);
    LUT = rand(1,256)>0.5;

    [rows, cols] = size(A);

    for i=1:rows
        for j=1:cols
            pixel = A(i,j);
            watermark(i,j) = LUT(pixel+1);
        end
    end

end
```

Published with MATLAB® R2019b