ECES T680 - Winter 2018 Drexel University
Instructor: Matthew Stamm

Assignment 3

This assignment is due by 11:59 p.m. on Monday, Feb 19. All reports should be submitted as PDFs.

Part 1: Least Significant Bit Data Hiding

In class, we discussed a method to embed a hidden signal in a digital image known as least significant bit (LSB) data hiding. This method operates by replacing a bit plane (typically the least significant) in an image with a hidden binary message. Often, this hidden message is a binary representation of another image or contains text.

- Write a Matlab function that extracts and displays a user specified bit plane from an image. Use this function to examine each of the bit planes in the **peppers.tif** and **baboon.tif** images. As you examine progressively lower bit planes, you should notice that each bit plane increasingly resembles noise.

  What is the highest bit plane for each image that no longer resembles image content and instead appears to be noise? Are these bit planes the same or different for these two images? If they are different, why would this be the case?

- Use your Matlab function from the previous part to examine the different bit planes of the following images: **LSBwmk1.tiff**, **LSBwmk2.tiff**, **LSBwmk3.tiff**. Do any of these images contain hidden content? If any of these images contain hidden information, please make note of the image and include the hidden content in your report.

- Write a Matlab function that replaces the N least significant bit planes from one image with the N most significant bit planes from another image. This function should accept both images along with the parameter N as inputs and return the modified image containing hidden data in unsigned 8-bit integer format as an output. Please comment your code and append it to your report.

  Use this function to hide the top N layers of the **Barbara.bmp** image in both the **peppers.tif** and **baboon.tif** images. How many bit planes of **Barbara.bmp** can you embed in each image before you start to notice distortion in the host image? How many bit planes of **Barbara.bmp** can you embed in each image before you can begin to see the hidden content? Are the number of bit planes that you can embed the same or different for each image? If they are different, why would this be the case?

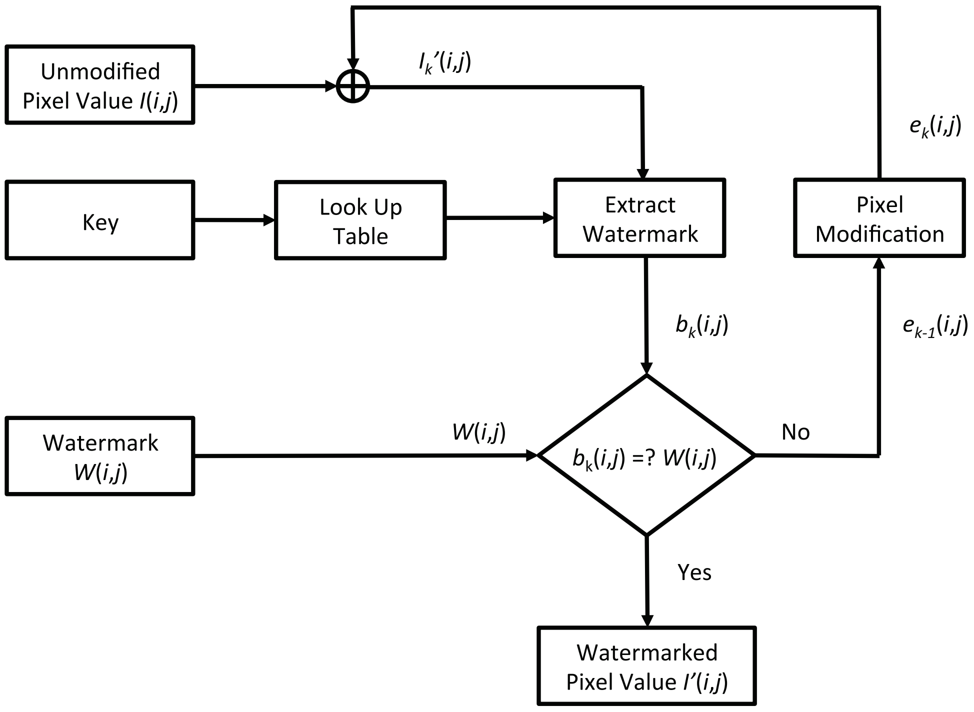
Part 2: Yeung–Mintzer Watermarking

While the LSB data hiding technique can be used to embed a fragile watermark in a digital image, it has several weaknesses. For example, data hidden using LSB can often be detected through visual inspection of each bit plane.

To address these weaknesses, Yeung and Mintzer proposed an improved fragile watermarking scheme that uses a look up table to embed a watermark in a digital image. A flowchart of the Yeung-Mintzer watermarking algorithm is shown in the figure below.

- Write a Matlab function that uses the Yeung–Mintzer altorithm to embed a binary watermark in an image. This function should accept both the image to be watermarked, a binary watermark (with the same number of rows and columns as the image), along with a key to be used to generate the look up table as inputs and return the modified image containing the watermark in unsigned 8-bit integer format as an output. Please comment your code and append it to your report.

  To generate the look up table, use the *rand* command in Matlab to generate an array of 256 uniformly distributed random numbers, then compare them to a threshold of 0.5. This will create an array of 256



uniformly distributed 1's and 0's. The first entry in this array will be the extracted watermark bit corresponding to the pixel value '0', the second entry will be the extracted watermark bit corresponding to the pixel value '1', and so on.

You should seed Matlab's pseudorandom number generator using the watermark key using the `rng` command. This will ensure that the same look up table is generated every time the same key is used.

Your code to generate the look up table should resemble the code below:

```
% seed the random number generator with the user specified key
rng(key)
% generate look up table values
LUTvals= rand(1,256) > .5;
```

Note: You do not need to generate the error diffusion portion of the Yeung–Mintzer algorithm described in their papers.

- Use your Yeung–Mintzer embedding function to hide the most significant bit plane of the **Barbara.bmp** image in both the **peppers.tif** and **baboon.tif** images (i.e. use the most significant bit plane of **Barbara.bmp** as your watermark).

  Use the function that you wrote in Part 1 to examine the lower bit planes of each of these images. Is the hidden watermark visually detectable? Include an image of the least significant bit plane of each watermarked image in your report. What is the PSNR between the original version of each image and their Yeung-Mintzer watermarked versions.

  Now use the LSB embedding function that you wrote in Part 1 to hide the most significant bit plane of the **Barbara.bmp** image in both the **peppers.tif** and **baboon.tif** images. What is the PSNR between the original version of each image and their LSB watermarked versions. Are these PSNR values higher or lower than the PSNR values obtained for the versions watermarked using the Yeung-Mintzer algorithm? Why (please explain which algorithm you would expect to introduce greater distortion into an image)?

- Write a Matlab function that extracts a watermark from an image that has been watermarked using the Yeung–Mintzer algorithm to embed a binary watermark in an image. This function should accept both the watermarked image to be watermarked and the key used to generate the look up table as inputs. It should return the watermark as an output. Please comment your code and append it to your report.

  Use this function to extract the watermark embedded in the image **YMwmkedKey435.tiff**. When extracting the watermark, use the number '435' as the key to generate the look up table. Include a picture of the extracted watermark in your report.

- One of the weaknesses of fragile watermarking using the LSB embedding technique is its susceptibility to attacks. Replace the top half of the watermarked versions (both LSB and Yeung–Mintzer) of the **baboon.tif** image with the bottom half of the **peppers.tif** image. Extract the watermark from each of these images and include them in your report. Now try to design an attack where you replace the top half of the **baboon.tif** image with the bottom half of the peppers image without destroying the watermark. How can you do this for the LSB watermarked image? Is this possible for the Yeung-Mintzer watermarked image (assuming the attacker does not know the key or look up table)?