Final Project: Watermark/Steganography

Anh Leu (anhleu2)
Dustin Mayfield-Jones (dustinm2)

1 Introduction

1.1 Motivation

Watermarks are identifying images or patterns that are embedded in images. They are often used on postage stamps, currency, and other documents to discourage counterfeiting. In the case of image copyright, watermarks are often trademarks or logos that artists and photographers embed in their images to discourage unwanted use. We are motivated to inconspicuously embed our "payload" (a logo) into our "carrier signal" (the Lena image) in order to protect our copyright without visibly damaging our image quality. We think this is useful in that we can share our digital images without the obnoxious intrusion of a noticeable watermark while maintaining a way to recognize the authorship of our images. As an added benefit, the secret image does not attract attention on its own like a visible watermark. We were motivated to discourage people from simply cropping out a visible watermark. Thus, by embedding a secret digital image within our images, we discourage a malicious person from trying to steal our images and remove the copyright evidence. Last, we wanted to test the boundary of how much information we could embed into our image without noticeably visually distorting it.

1.2 Background and literature review

We adopted techniques from the practice of steganography to embed an image within another image. When starting this project, we had the idea to imbed a secret watermark into our image of interest, but steganography was a new concept. Unlike watermarking and cryptography, steganography aims to embed large capacities of information that is undetectable and robust (resists image processing methods/compression) into a cover media. These aims separate it from related techniques such as watermarking and cryptography in which the embedded information is conspicuous or encoded but difficult to break. Steganography is intended to be undetectable so that only the sender and receiver even know of the hidden message's existence. See [1] for review. Specifically, the authors detail the connections between steganography, watermarking, and cryptography. They present objectives for steganography, watermarking, and cryptography as secret communication, copyright preservation, and data protection, respectively. Here, we accomplish both secret communication and copyright protection.

We develop two steganographic systems, or methods. In both cases, we leave behind detectable traces of our work in our resulting images with hidden content. While our secret images cannot be revealed without knowledge of our steganographic system, the existence of our changes are easily detectable. For example, we see significant MSE between our original image and our images embedded with secret payloads. The practice of finding the underlying distortions that contribute to these differences is called statistical steganalysis. For a review of statistical steganalysis algorithms, see [2]. In general, the information-hiding process begins by finding redundant bits in the cover media. In our project, we have done this by replacing redundant frequencies in the DFT of our cover image and by replacing the least significant bits of images directly with bilevel images. In addition to our methods, one could also replace the least significant bits of DCT coefficients with a payload [3]. Improved algorithms that elude less sophisticated statistical steganalysis include the pseudo-random selection of which DCT coefficients get modified [4] or decrements the least-significant bit of a DCT coefficient's value by matrix encoding [5].

1.3 Objective

Our group chose 3 objectives that build upon each other to explore watermarks in images in order to both include a conspicuous, and inconspicuous, watermark in an image.

1. To introduce a logo directly into our Lena image as a classic watermark.
2. To use the Discrete Fourier Transform (DFT), computed using the Fast Fourier Transform (FFT), to hide an extractable watermark in an image.
3. To replace bitplanes to hide an extractable watermark in an image. The last two objectives have the same aim: to hide an image within an image.

2 Proposed approach

2.1 Outline

We adopted techniques from the practice of steganography to embed an image within another image. We proposed to hide an extractable watermark (Figure 1b) in our image (Figure 1a) using two relevant image processing methods.

2.2 Watermark

As an initial step toward our goal, we first decided to add a visable watermark (Figure 1b) by thresholding it to a bi-level image and overlaying it on our 512 x 512, gray scale, lena.png (Figure 1a) image. We did this by averaging the pixel intensity between the two images, resulting in a watermarked image. Later on, we added an alpha factor ranging from 0 to 100 that determines the proportion of watermark to carrier image; the logo is more visible as the alpha value is decreased.

2.3 First Steganographic Method - DFT

In the first method we began by computing the discrete Fourier transform (DFT) of the gray scale Lena image using a fast Fourier transform (FFT) algorithm similar to [6]. Then we hid our watermark by applying binary thresholding followed by run length encoding to compress its information content. In the spatial domain of our Lena image, we replaced the magnitude of the DFT in the mid-range frequencies with our encoded watermark.

From ECE 418 Lab 2: Two-Dimensional Fourier Transform, we learned that much of the image's appearance is changed by altering the phase, and thus we didn't change any phase values. We also learned that low-frequencies contain much of the information content in the Fourier domain, and high-frequencies contribute to an image's sharpness. Therefore, we proposed to store our encoded image in the mid-range frequencies in a ring region centered around the zero frequency of the DFT magnitude image similar to [6]. In [6], the authors embed a secret text into an RGB image. We modified their code in several critical ways. First, we inject a run length encoded gray scale watermark image into the gray scale lena.png instead of text. Second, we create a difference matrix between the float value from the inverse DFT and the uint8 value from digital image quantization. This is in order to allow a way for the author to retrieve the watermark from the resulting steganographic image (with our hidden payload), and it is further described in our discussion. Third, we adjusted the ring region size (range) to allow us to control the number of mid-range frequencies that hold our payload. Fourth, we added computation to calculate the MSE between our resulting steganographic image (with our hidden payload) and the original lena.png.

2.4 Second Steganographic Method - Bitplane

In our second method, we replaced the least significant bits of our image with binary thresholded watermark images. First, we would bitmask the desired least significant bit and replace it with a 512x512 image that had been thresholded using our custom encoder.cc program. We then implemented lab10-bitplanes.cc from

ECE 418 Lab 10 to visualize the 8 bitplanes. All 8 of these images were composed of the pixel values 0 or 255, and the least significant bits storing our secret images were recovered.

In order to test the limit of how many images we could hide in our lena.png, we reiteratively added additional 512x512 thresholded images, one at a time, to test when the secret image would visibly distort our lena.png image. For example, in our test of replacing the least three significant bits, we replaced the last 3 bitplanes from lena.png (see f g and h from Figure 7) with our secret images (see f g and h from Figure 8).

2.5 Advantages/disadvantages

Our two steganographic methods share several advantages over applying a watermark (Figure 3). With our first method, we were able to run-length endcode our logo.png into middle frequencies of the frequency domain and thus accomplish our goal of hiding an image within an image.

With our second method, we were able to embed more information than a watermark in that we added two additional secret images into lena.png. Our secret images are visually undetectable, and thus we achieved our goal of hiding our payload (logo.png) in a recoverable way so that our cover image (lena.png) is not visually distorted and we can maintain our proof of copyright authorship in the image.

Of our two steganographic methods, the first method requires saving a difference image between the original lena.png and the lena.png with our hidden encoded image. This is because when adding the encoded image in the DFT of lena.png, rounding error is lost when applying the inverse DFT to recover our two images. Therefore, it can be seen as a disadvantage of method one compared to method two (bitplanes) in that the user wanting to hide information must keep a difference image in order to fully recover their payload. An additional advantage of method two is that we can embed multiple secret images. For our 8 bit lena.png, we were able to embed three images into the three least significant bits with minimal visual detection (see our results). For images with greater bits, such as 16 bit images, we can hide even more information in our images. This is beyond the scope of hiding a watermark in the image to preserve copyright, but it could be advantageous to hid more information in other applications where secret message capacity is a goal.

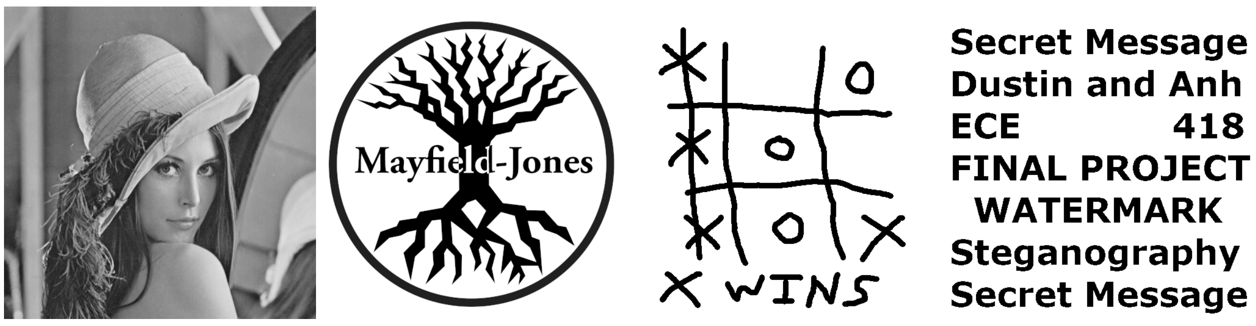
3 Experimental design and results

3.1 Experimental settings

We decided to use the images processing skeleton libraries provided to us for this course as a base to build our code for this project. However, because the provided libraries only work properly for gray scale images (with pixel values quantized to discreet values between 0 and 255), we chose to specifically work with images that also satisfy those conditions.

3.2 Dataset Used

See Figure 1 for images used as our input images for our methods. Each image is a 512x512 8 bit gray scale image.



(a) lena.png (b) logo.png (c) game.png (d) letter.png

Figure 1: Image dataset used

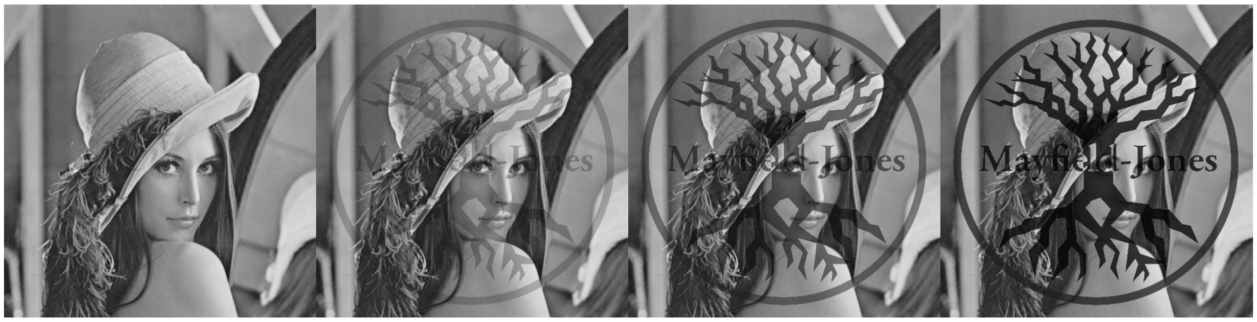
3.3 Evaluation criteria

Our evaluation criteria were a combination of subjective and quantitative measures. To best determine if our hidden messages were visibly detectable, we had human test subjects visually inspect the images and report the presence or absence of image distortion. We used MSE to quantify our image distortion.

3.4 Experimental results

3.4.1 Watermark

Qualitatively, we observed a useful visual watermark in our lena.png. The approximate MSE between lena.png and lena_basicwatermark.png with alpha value 75, 50, and 25 was 240.4473, 948.1944, and 2133.4821, respectively. The lower alpha increases the logo.png contribution.



(a) lena.png (b) Alpha 75 (c) Alpha 50 (d) Alpha 25

Figure 2: Basic Watermark

3.4.2 First Steganographic Method - DFT

1. In our first method, our hidden message was our first result. We half-sized our payload (Figure 1b) to (256×256) pixels and applied run length encoding to reduce our message size. Then we added the hidden message to the DFT magnitude image of the original lena.png (Figure 1a) around a ring region centered around the zero frequency. The inner radius of the ring was defined as $\frac{1}{4}$ of the image length. The ring then expanded from the inner radius to a user defined length. We called this parameter "range" as it describes the range of mid-frequencies to be replaced by our payload. Our first result used range = 50 pixels to define our ring region (Figure 3).

   The hidden message started as:

$$2, 256, 256, 3194, 11, 235, 33, 216, 45, 208, 54, 197, 61, 192, 69, 183, 33, 9, 33, 179, 25, 31, 25, 172, ...$$

The output image was visually undetectable from the original input image with MSE of 11.5883. By calculating the difference matrix between the float image and the uint8 saved image, we were able to retrieve the original message. This procedure is repeated for the following tests.

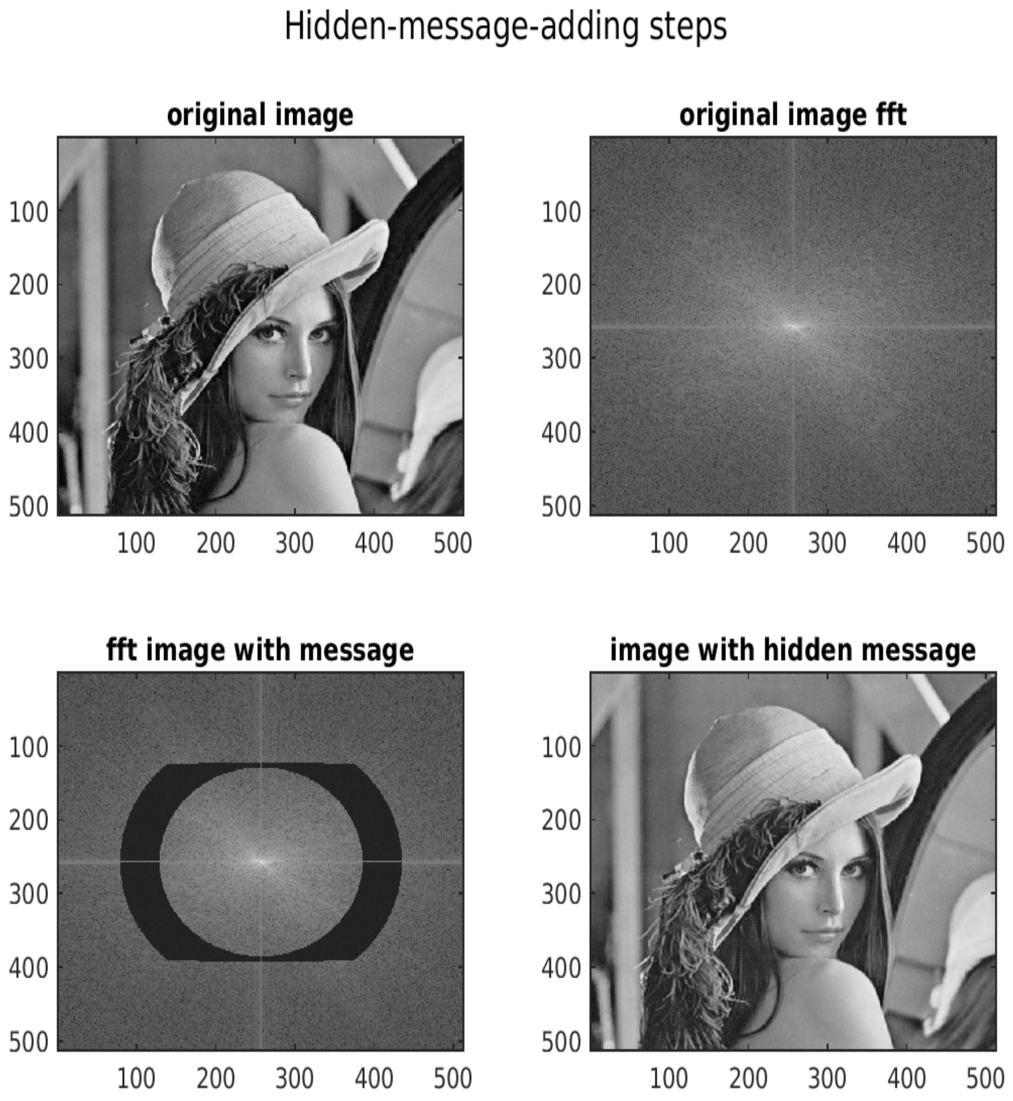


Figure 3: Method 1, Test 1

2. Next, we looked at injecting the logo.png image (512 × 512) without resizing. However, this resulted in a message that exceeded the available space for a range of 50 in lena.png (Figure 4). The hidden message started as:

$$2, 512, 512, 13038, 39, 461, 58, 449, 72, 432, 84, 425, 95, 410, 104, 406, 112, 394, 120, 390, 128, 379, ...$$

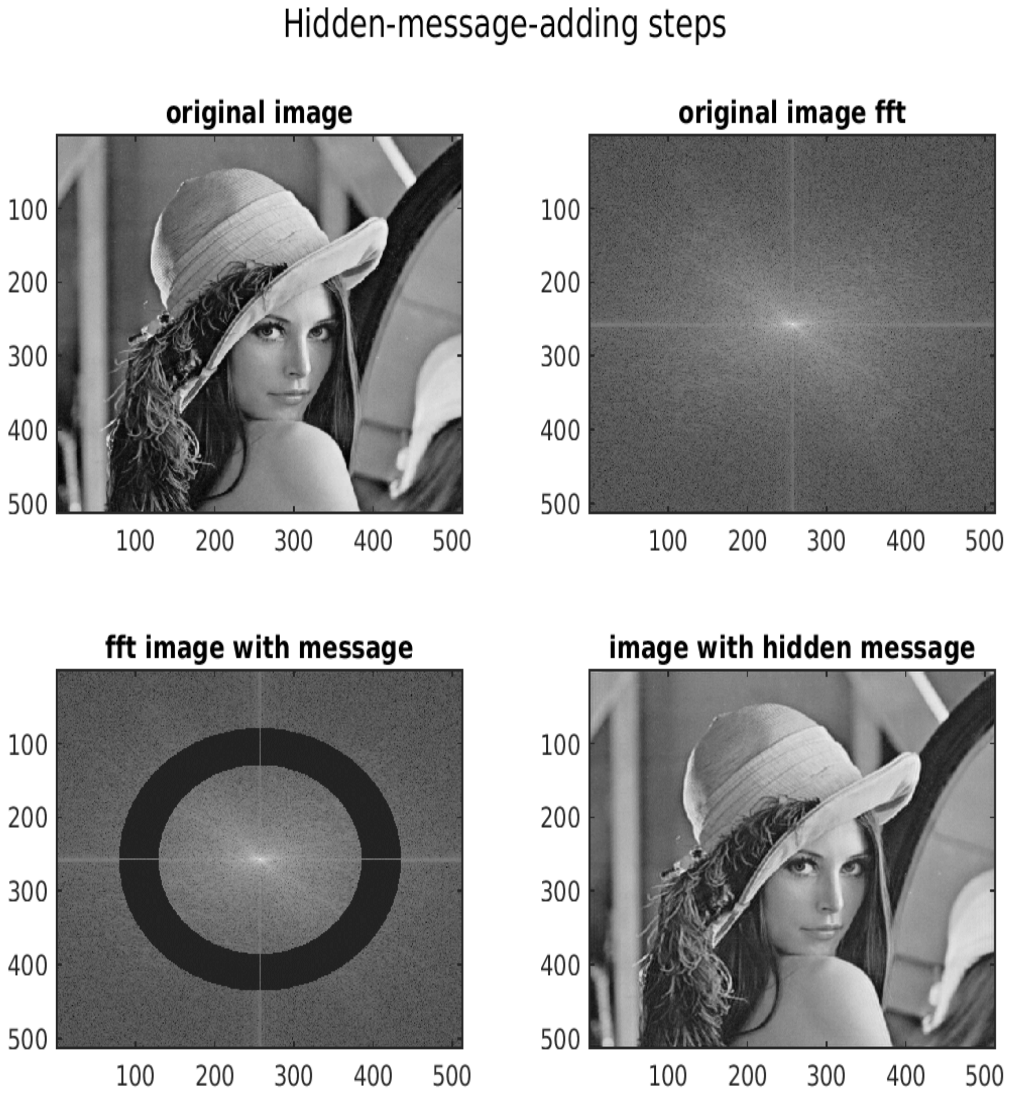


Figure 4: Method 1, Test 2

3. As we could not fit the full-size payload into our cover image with a range of 50, we expanded it to 100 (Figure 5). The output image is very similar to the original input image with MSE of 17.2810.

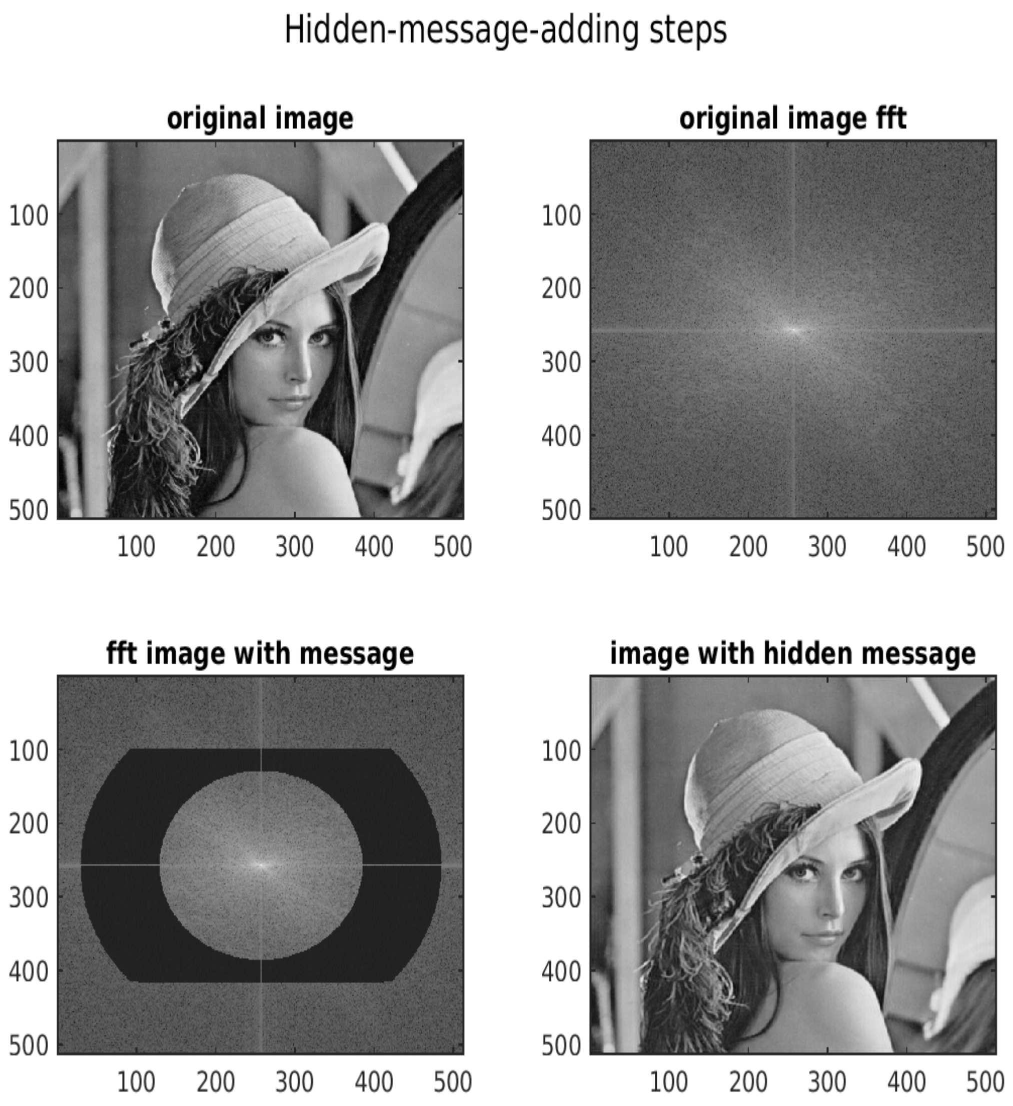


Figure 5: Method 1, Test 3

4. Finally, we tried to push the limit to see how much information we could encode into the cover image until the it became recognizably distorted. We encoded the half-sized (256 × 256) payload into the half-sized lena.png (256 × 256) with a range size of 60 (Figure 6). The output image distortion is barely visually noticeably from the original with a granular noisy effect and MSE of 59.6191.

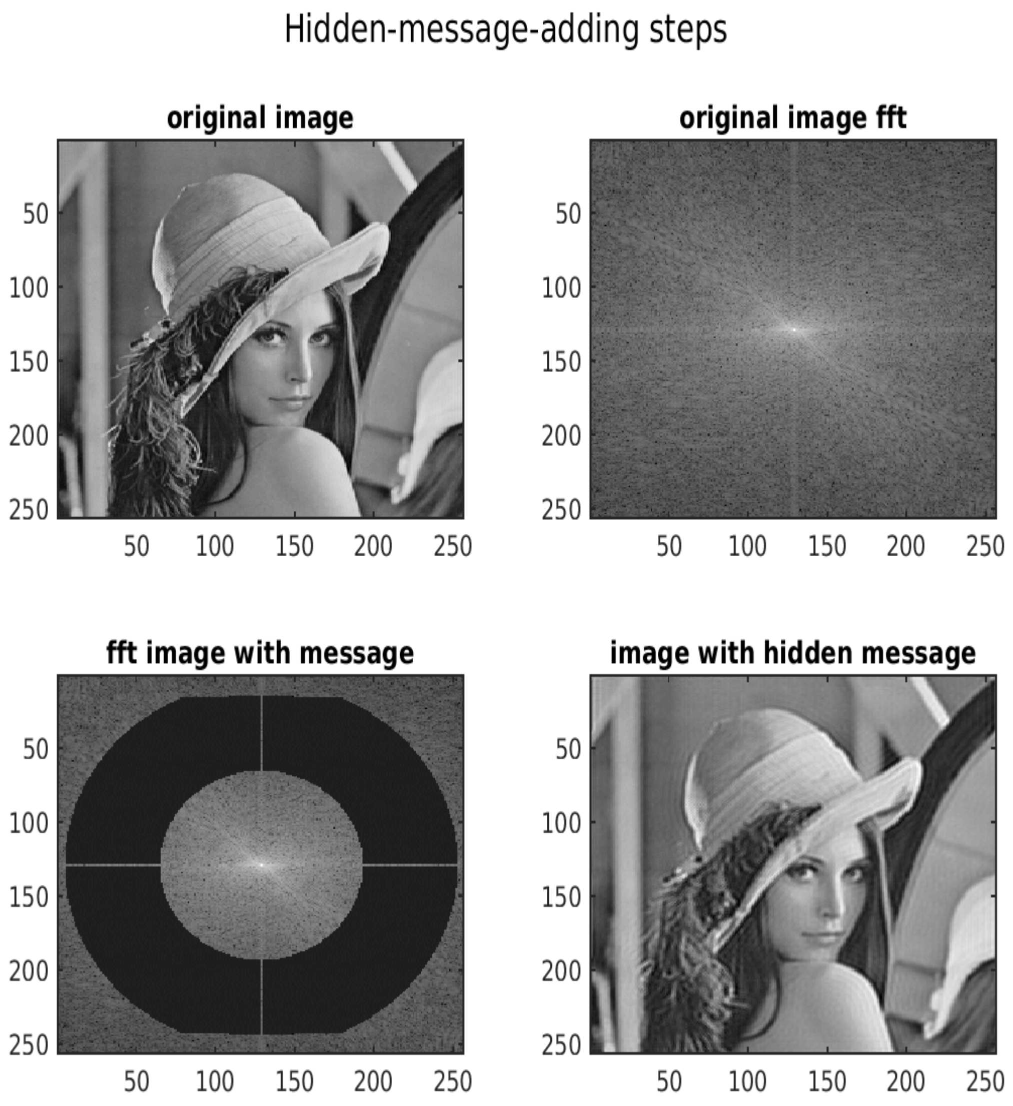
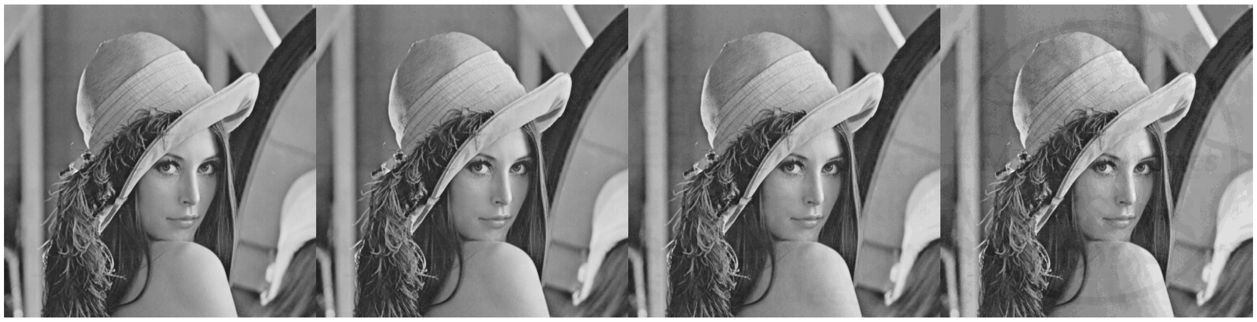


Figure 6: Method 1, Test 4

3.4.3 Second Steganographic Method - Bitplane

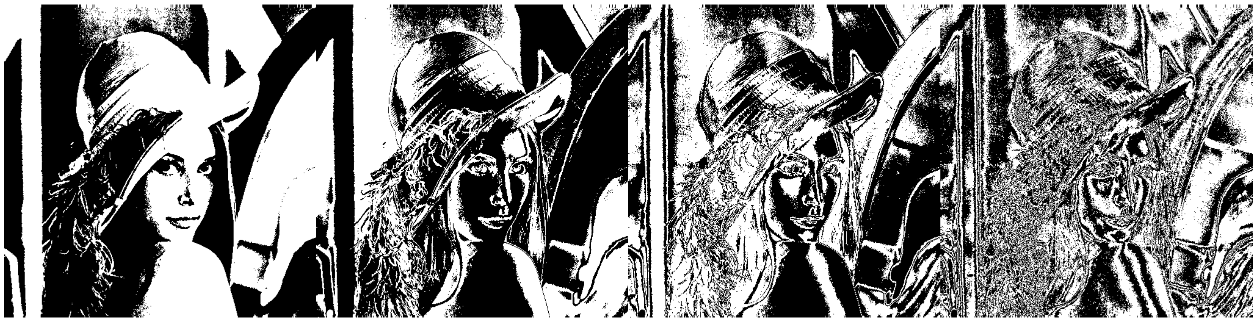
No students identified distortion in Lena with the least two significant bits replaced by our watermark (test 2 - Figure 7b), and 10/14 students identified distortion in Lena with the least two significant bits replaced. Therefore, we concluded that replacing the least two significant bits with hidden images was appropriate to achieve our goal of adding visually undetectable images. The approximate MSE for tests 1-4 were 0.4979, 2.9415, 13.3123, and 59.1286, respectively, reflecting the greater distortion due to the increasing capacity of our payload.

Second Steganographic Method		
Number of least significant bits replaced	Number of student identifying distortion	MSE
0	0/14	0.0
1	0/14	0.4979
2	0/14	2.9415
3	10/14	13.3123
4	14/14	59.1286

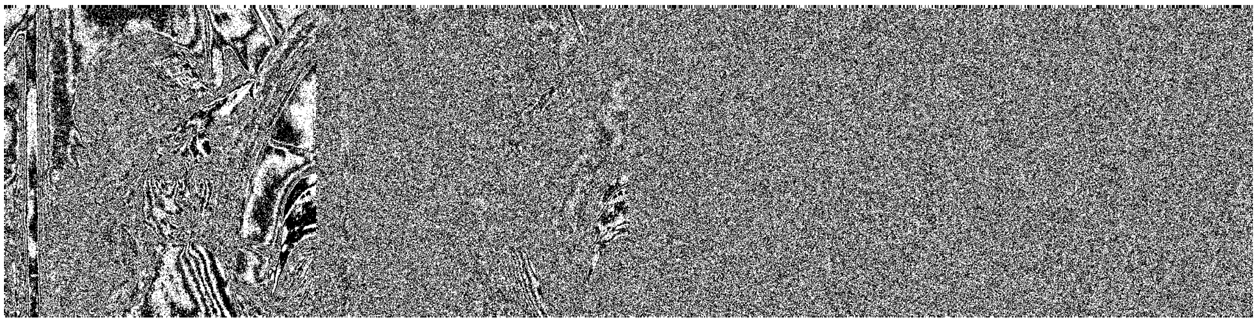


(a) Lena with least significant bit as watermark (b) Lena with least two significant bits as watermark (c) Lena with least three significant bits as watermark (d) Lena with least four significant bits as watermark

Figure 7: Method 2: Replacing Bitplanes

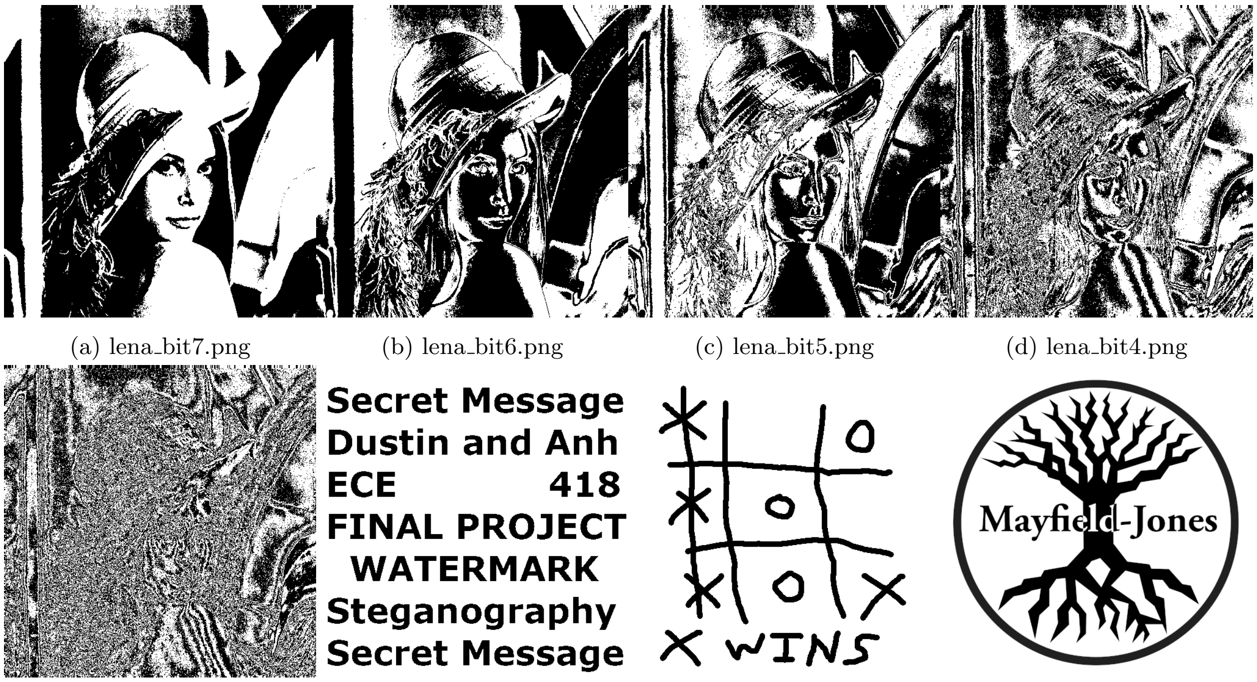


(a) lena_bit7.png (b) lena_bit6.png (c) lena_bit5.png (d) lena_bit4.png



(e) lena_bit3.png (f) lena_bit2.png (g) lena_bit1.png (h) lena_bit0.png

Figure 8: Original Lena Bitplanes



(a) lena_bit7.png (b) lena_bit6.png (c) lena_bit5.png (d) lena_bit4.png

(e) lena_bit3.png (f) letter.png (g) game.png (h) logo.png

Figure 9: Bitplanes of Lena with least three significant bits as watermark

4 Discussion and future work

Overall, our approaches performed unexpectedly toward our goals, and we suggest some improvements for future study. We started with the base libraries used in ECE 418: Introduction to Image Video Processing, especially the image and FFT library. We restricted our project to gray scale images with pixel values as integers ranging from 0 to 255. A future improvement would be to adapt our code for RGB images of various bit sizes and value ranges.

4.1 Tools and Basic Watermark

With the base code provided by the course labs on image decimation and interpolation, we were able to use our understanding of different methods for 2D Interpolation/Decimation to reduce our message payload size. Therefore, we successfully devised a tool to easily change image sizes and perform thresholding to create a logo image with pixels of only 0 or 255 used throughout our project. In the future, we could potentially add tools to work with other types of image (RBG or 16 bit images of various pixel sizes and shapes), deblur, increase/decrease contrast, etc.

Creating a basic watermark was our first step to examine how we could combine images. We wanted to start with a practical and easy to use way to add a watermark to an image. Therefore, it was rather straightforward to achieve this goal. In the future, once we include RGB images in our platform, we may be able to do more with this by changing the watermark to be more appealing by adding color effects to it (changing its color of adding a color gradient for a rainbow effect).

4.2 First Steganographic Method - DFT

For this method, we were inspired by Lab 10: Compression: Run-Length Encoding, and the paper [6]. Lab 10 taught us to use compression with run-length encoding while paper [6] informed us on how to encode a visually inconspicuous message into an image. At first, we thought about encoding the whole image as a payload, but we then discovered the restriction length of the message prevented this. Therefore, we decided to combine compression and our understanding of the DFT to include our secrete message into cover images. In addition to run-length encoding our images payload into the stored message, we included its starting pixel of either 0 or 255 and its width and height. This is necessary information to reconstruct the original logo images. The result was as expected. We can encode our payload into the DFT image and save the resulting new picture which does not look different than the original picture when our payload is small compared to the original image size. However, we discovered a problem in that when we save the image with the embedded payload, we need to round the pixel from float value to uint8 value. This resulted in a inaccurate DFT imagem and we could not retrieve the hidden message. We overcame this challenge by keeping a difference float matrix that stores the differences between the float image and uint8 image. Thus, the original image and the payload can be recovered by using the difference image to calculate the original float matrix.

We suggest this method could be improved in several ways. First, the encoding is currently in ASCII but we realize that the hidden messages are numbers. We suggest modifying the hidden message to be separated by 0 instead of ',' and to change the encoding from ASCII values to numbers to further reduce the message length. Secondly, a potential way to encode the message so that it can be retrieved directly from the uint8 image without having to keep the difference image by using some rounding number patterns would improve this methods utility. Research invested to secure a way to remove the need to store a difference image may be fruitful.

4.3 Second Steganographic Method - Bitplane

We also were inspired for this method from Lab 10: Compression: Run-Length Encoding. We used the provided code that split the image to 8 bit plane images and reverse engineered it to replace any bit plane of the image with a payload image. We then successfully used our provided code to split the original image to 8 bit plane images to recover our payload. We expected that we would be able to successfully hide more than one image in our 8 bit cover image. We experimented by injecting our payload into different bit planes and saw that adding payloads to the last two least significant bits resulted in images that are indistinguishable from the original cover image and showed low MSE. These results were as we expected.

We believe that the current logo encoding is naive. Instead of simply changing the least significant bit planes, we could do some further encoding to store more images in those last two bit planes. We propose for future studies to investigate ways to mask the payload image. We suggest using methods such as splitting a payload image to only occupy every 3 bits, and then further split the message over 3 bit planes, so when someone does bit slice our resulting hidden image, they do not directly know that there is a hidden payload. We could then easily retrieve the payload using our knowledge of the decoding pattern in our steganographic system.

5 Acknowledgements

We extend our thanks to the 2019 members of HKN Honors Fraternity at the University of Illinois Urbana-Champaign for their help as human subjects for our experiments.

References

[1] Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3):727–752, 2010.

[2] N. Provos and P. Honeyman. Hide and seek: an introduction to steganography. *IEEE Security Privacy*, 1(3):32–44, May 2003.

[3] Tao Zhang and Xijian Ping. A fast and effective steganalytic technique against jsteg-like algorithms. In *Proc. 8th ACM Symp. Applied Computing*, pages 307–311. ACM Press, 2003.

[4] Jessica Fridrich, Miroslav Goljan, and Dorin Hogea. Attacking the outguess. 10 2002.

[5] Jessica Fridrich, Miroslav Goljan, and Dorin Hogea. Steganalysis of jpeg images: Breaking the f5 algorithm. In Fabien A. P. Petitcolas, editor, *Information Hiding*, pages 310–323, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[6] Sudeep Sarkar. Physical problem for fast fourier transform computer engineering, 2009.