Math 414 Final Project – Steganography

Tyler Harmon

**Introduction**

Steganography, roughly meaning “covered writing” is a method of hiding a piece of information inside of another (Lo, Topiwala, & Wang, 1998). Take, for example, a letter with a secret message hidden inside. This may simply be that the first letter in every line can be taken and turned into a comprehensible sentence, or it may follow a more complex pattern. The letter itself may be innocuous and have a coherent message without knowing the key, but the interpreted message can be completely different than the plaintext one. Steganography is not limited to a particular medium or file type, and can vary in complexity.

In this project, the goal is to hide files of various types and sizes in images using their Haar and Baubechies decompositions. Then, the reconstructed images will be compared to their original counterparts to see how the technique affects the quality of the image.

**Mathematical Background**

Steganography shares many similarities with its cousin cryptography, but there are a few differences. Primarily, the purpose of the different techniques is usually different. While steganography tries to conceal a message, cryptography focuses on security. Cryptography assumes a bad-actor can intercept whatever encrypted message is being sent and tries to keep them from recovering the actual data. From a functional perspective, cryptography typically secures data by altering the message (image, text, or other data) sent using processes that are difficult to reverse without the proper key (typically some sort of backdoor that is usable only when one has the proper signature). In contrast to this, steganography tends to leave the message relatively unchanged, but instead breaks it up and embeds the chunks of message into the medium. Both of these techniques could be used in tandem to both obscure and secure a secret message being sent.

Some steganographic techniques for hiding data within images apply data directly to the individual pixel values of the image. However, this can affect the image quite drastically. For example, if two adjacent pixels are the same color in the original image but the adjacent bits of data in the hidden file are sufficiently different, the pixels in the reconstructed image will vary. This can make a passerby or recipient who is looking at the image suspicious, especially if the original image is available for quick visual comparison.

In contrast to this pixel-by-pixel technique, the wavelet-based technique doesn’t affect pixels on an individual basis. Instead, the coefficients generated by the decomposition are altered to contain the data. Since the decomposition “smooths” the value of adjacent units, the changes caused by reconstruction after a file is hidden are dispersed over a wider region of the image.

The MRAs used in this project are the Haar and Daubechie II. The Haar decomposition has compact support and is discrete, which is fine since the digital images are pixelated anyways. This wavelet is a simple baseline to test this processing on. The Daubechie MRAs are used often in image compression and processing for their ability to approximate the image well when reconstructing. The db2 wavelet is a continuous wavelet with compact support, which makes it one of the ideal wavelets for our processing (Boggess & Narcowich, 2009).

To extend the wavelet decomposition from 1-D to 2-D, additional steps are taken. Transforms are taken horizontally (row-wise), vertically (column-wise), and diagonally across the image to create three separate transformed images, which are synthesized to get the combined coefficients for the decomposed image.

**Application**

This application runs through the typical MRA processing steps: sampling and decomposition, processing, and reconstruction. The processing step, however, is based almost entirely on the file to be hidden as opposed to the coefficients calculated by the MRA. The Update\_Coefs.m program can load a .mat file created in the 2-D wavelet analysis tool, embed another file inside the coefficients, and save the .mat file to be reconstructed by the wavelet analyzer. This can also extract a file from coefficients that have a file embedded in them.

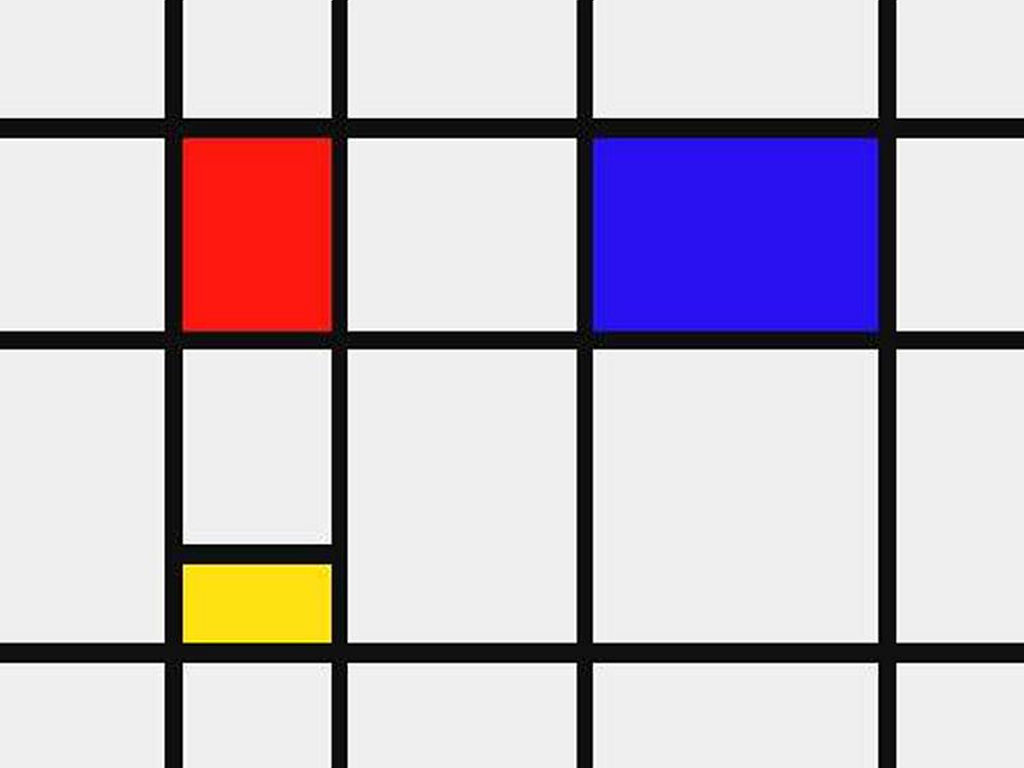
The program leverages the binary encoding of files to embed the files. Many common types of digital media use 8-bit values (one Btye) per pixel/audio sample/character (eg. .png, .m4a, and .txt files, respectively). The least significant bits of coefficient values are replaced with substrings of the binary file to reduce the impact the swapped bits have on the decomposed images. For example, the bitstring “11111111” represents the value 255 in most file representations. If we take replace the last 4 bits with “0000” the outcome (“11110000”) is 240, but if we instead replace the most significant 4 bits, the outcome (“00001111”) is 15. A 4-bit substring was chosen as the size to replace to easily subdivide the Bytes in the hidden file; concatenating the last 4 bits of a two adjacent coefficients will yield one Byte of data from the hidden file. Reducing the number of bits replaced per coefficient reduces the effect of the change on the resulting image, but requires many more coefficients to be changed to store the entire file. No compression or modification is done to the hidden file other than separating the bits, so no loss of quality is accrued through this processing.

To recover the file from the coefficients, a 32-bit value is extracted from the first 8 coefficients of the decomposition. This was written in during the embedding process to determine how many values must be read to reconstruct the entire file. Then each proceeding pair of coefficients are taken together: the odd-indexed coefficient contains the most significant 4 bits of the file and the even-indexed coefficient contains the least significant 4 bits. These sets of 4 bits are concatenated to recreate the Byte of data from the file. All of these values are saved together into one binary file and, by opening the file using the proper file extension, the file can be viewed as if it was the original.

**Test Images**

To test the effect of steganography on the images reconstructed, different images with very different styles were deconstructed and files of varying sizes were hidden inside them. The following images were selected as targets to embed other files within:

Piet Mondrian-esque Modernist Painting



*Figure 1: Painting in the style of Piet Modrian; Simple shapes and blocks of color*

This image was selected to be the simplest of those tested. The lines are distinct and the colors are constant across the shapes. Thus, changes to a small region of pixels (as would occur with pixel-by-pixel steganography) should stand out quite easily.

Multi-Colored Gradient



*Figure 2: Smooth gradient across multiple colors*

This image was generated in Microsoft PowerPoint to directly contrast Figure 1. While the former has clear-cut regions of color, the latter changes gradually with no hard lines.

A Sunday Afternoon on the Island of La Grande Jatte, Georges Seurat



*Figure 3: A Sunday Afternoon on the Island of La Grande Jatte. From The Art Institute of Chicago*

*(https://www.artic.edu/artworks/27992/a-sunday-on-la-grande-jatte-1884)*

Besides being a phenomenal painting, this image was selected because of the style in which it was created. It uses a technique called “pointillism”, where the entire picture is created using small dots of paint that, when seen together, create the illusion of continuous shapes (not unlike a wavelet approximation). These individual dots make for plenty of inherent noise within the image, meaning it should be all the more difficult to distinguish differences between the original and altered versions.

**Test Files**

Files of varying size and type were embedded within the test images. Using different file types was done to show the generality of this process, and the increasing amount of data hidden illustrates the visual effect it causes.

Text File



*Figure 4: Sample text file embedded using steganography*

This file is the smallest and simplest. Comprised of only 107 bytes of data, this file can be hidden quickly and surreptitiously.

Image File



*Figure 5: Texas A&M University logo in .png format*

While still a relatively simple image, the 400x400 size means there are 160,000 values in in an uncompressed state, exponentially larger than the text file.

Audio File

Audio files such as songs run through data at a very high rate, so just a few seconds of music can translate into a much larger file than that of a document or image. To test this, a 25-second clip of Saint-Saens’s “Carnival of the Animals – Le Cygne”, performed by Yo-Yo Ma, Eugene Ormandy, and Yan Pascal Tortelier was used. This was close to the largest file that could be stored in the test images without increasing the number of bits stored in each coefficient.

**Procedure**

For each test image, eight decompositions were created. First, the first-level decomposition coefficients were generated using both the Haar and db2 MRAs. Then for each of these decompositions, each of the test files was embedded in the coefficients and reconstructed to observe changes to the approximations of the images.

**Results**

The steganography processing algorithm was able to successfully embed the test files into each of the images’ decompositions. Figure 6 shows the test image after being extracted from the gradient’s coefficients. This file appears exactly the same as the original.



*Figure 6: Texas A&M logo extracted*

**References**

“Chapter 6: The Daubechies Wavelets.” A First Course in Wavelets with Fourier Analysis, by Albert Boggess and Francis J. Narcowich, Wiley-Blackwell, 2009, pp. 234–249.

Lo, Han-Yang, et al. “Wavelet Based Steganography and Watermarking.”, Cornell University, 1998, www.cs.cornell.edu/topiwala/wavelets/report.html.

Seurat, Georges. *A Sunday Afternoon on the Island of La Grande Jatte*. The Art Institute of Chicago, https://www.artic.edu/Artworks/27992/a-Sunday-on-La-Grande-Jatte-1884, 1884, The Art Institute of Chicago, Chicago, IL.

TAMU, ESPN. “Texas A&M University Logo.” Twitter.com, 2015, pbs.twimg.com/profile\_images/378800000413334813/8a3dbf0c6d9a35f5efcf4c26afa45fff.png.