Digital Image Watermarking

ECE 533 Image Processing, University of Wisconsin-Madison

Matthew Elliott and Brian Schuette

December 21, 2006

Introduction

Watermarking is a technique used to hide data or identifying information within digital multimedia. Our discussion will focus primarily on the watermarking of digital images, though digital video, audio, and documents are also routinely watermarked. Digital watermarking is becoming popular, especially for adding undetectable identifying marks, such as author or copyright information. Because of this use, watermarking techniques are often evaluated based on their invisibility, recoverability, and robustness. Our goal was to implement two different watermarking methods and evaluate their susceptibility to attack by various image processing techniques. Additionally, we wanted to create a GUI that would allow users unfamiliar with Matlab to add and extract watermarks, as well as evaluate their respective robustness based on a few morphological image attacks.

After learning about watermarking by bit-plane slicing in class, we were very interested to investigate the process by which one watermarks an image, as well as the degree to which the original image is changed by the watermarking process. To help us learn how images can be watermarked, we decided to implement two watermarking techniques, watermarking by bit-plane slicing and watermarking using the Cox method. It was extremely difficult to decide which watermarking methods to implement, because there are a multitude of different methods by

which to watermark an image. The Cox method and the bit-plane method allowed us to take two very different approaches to watermarking. We got to work in both the special and frequency domain, as well as having different goals for each method. Our bit-plane slicing approach is designed to work primarily as a fragile watermark. A fragile watermark shows the degree to which changes are made to an image. The Cox method, on the other hand, is designed to be robust. It works in the frequency domain, allowing it to resist many common attacks to the image.

In implementing these methods, we had to learn and create the processes to add a watermark and extract a watermark from digital images. To evaluate the degree to which watermarking affects the original image, the GUI was designed to display image difference graphically as well as numerically in a relative error format. This helps the user evaluate the invisibility of the watermark, as they can compare the changes watermarking makes to the original image. When the user extracts a watermark from an image, the difference between the watermarks is also shown both graphically and numerically. This will help the user decide if a watermark can be consistently recovered with the given method.

Approach

In applying watermarks, our focus was on invisibility, recoverability, and robustness. All of these are intricately linked. The less the image is affected, the easier it is to remove the watermark; recoverability is heavily reliant on robustness, for the watermark must still be present even after morphological attacks. Attacks may be accidental or intentional, but all images that

are digitally watermarked may be subject to attack. Most attacks are attempts to alter the image in order to destroy the watermark while preserving the image. Since watermarks may be hidden copyrights, this is extremely undesirable.

In order to address the issue of robustness, we decided to allow the user to use seven different morphological attacks to see how the extracted watermark is affected. The morphological attacks that are provided in the GUI are image scaling and cropping, as well as Gaussian low-pass (blur) filtering, unsharp contrast-enhancement filtering, averaging filtering, and circular averaging filtering. These attacks can be used to alter a watermarked image. The watermark can then be extracted and compared to the original watermark, allowing the user to evaluate the method's performance with respect to alterations. This allows the user to consider robustness in terms of recoverability, and how each of the methods stand up to various changes in the image.

To address the issue of invisibility, the GUI allows users to compare images before and after watermarking. It displays the difference visually and numerically. This same system is used to allow users to compare images before and after morphological attacks, allowing a wide spectrum of uses. With time, this GUI would aid in systematically identifying the strengths and weaknesses of various methods, allowing one to prepare counterattacks against the widest array of attacks possible.

In creating the project, we first began by implementing bit-plane slicing watermarking. Matlab functions to insert and extract a watermark were created. These functions work with the original image, as well as a binary watermark, which is inserted into the least-significant bit-plane as shown in Figure 1. The GUI and assisting functions were designed to accept color, grayscale, and binary images, which will be converted as needed when used as the original

image or the watermark to be inserted. The functions were tested for correct operation and then built into the GUI. Unfortunately, bit-plane slicing is computationally intensive and takes a significant amount of time to complete. The user is forced to wait until the computations finish before they may continue.

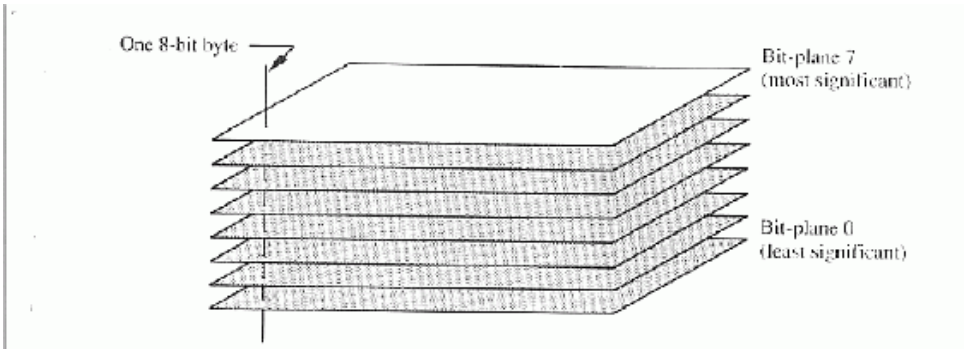

Figure 1. Bit-plane representation of an 8-bit digital image.

Next, the Cox method was implemented. The watermark insertion and extraction functions were created. The insertion function requires an original image. It will create a watermark if not provided and will return the watermarked image as well as the watermark that was inserted. The watermark generated is an independent, identically distributed zero-mean unit variance Gaussian, which is essentially random noise. The insert function takes the discrete cosine transform of the original image, adds the noise to the 1,000 largest coefficients, and then uses the inverse discrete cosine transform to produce the watermarked image, as seen in Figure 2. The extraction function requires the watermarked image and the original, from which it will extract and return the embedded watermark. After the functions were tested, they were built into the GUI. Fortunately, they are significantly faster than the bit-plane slicing approach and involve only minimal delay.

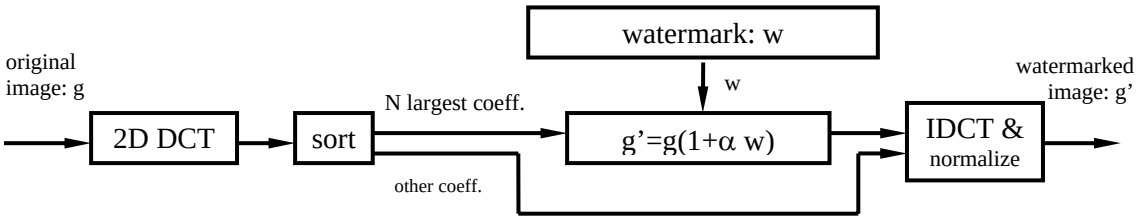

Figure 2. The Cox method of inserting a watermark.

After the two watermarking methods were correctly implemented, the morphological image attacks were created and built into the GUI. They were fairly straightforward to implement, as Matlab provides functions that do much of the work. Once these were created, the GUI was structured such that a watermarked image can be attacked and then evaluated. The GUI was then tested and fine-tuned to ensure it was working correctly.

Work Performed

We divided our program up to ease in code review and reuse. The GUI allowed user interaction and ensured variable stability. It also did many of the basic operations. For special functions, we designed and implemented the following files:

bitslice.m: Separates grayscale images into bit-planes.

cbitslice.m: Separates color images into bit-planes.

bitslice2gray.m: Combine bit-planes into grayscale image.

bitslice2rgb.m: Combine bit-planes into color image.

makewm.m: Converts any image to appropriate binary watermark.

cox_e.m: Extracts watermark using Cox method.

cox_extract.m: Cox method extract for grayscale and color images.

cox_i.m: Adds watermark using Cox method.

cox_insert.m: Cox method insert for grayscale and color images.

edit_image.m: Morphological image attacks.

watermark.m/watermark.fig: GUI files.

perror.m: Calculates relative error between images.

The GUI was designed to provide an easy to use interface that allowed easy comparisons of images. The user first loads the image to be modified, then chooses a watermarking method. When applying bit-plane slicing to an image, the user is allowed to perform each step in turn by following the progression on the left, slice, insert watermark, and reform. The watermark can be an image of their choice, as the program will convert it to the necessary binary image. Each bit-plane is displayed, and the comparison can be seen in the upper-right corner of the GUI (see Figure 3). When using the Cox method, the user only needs to load the image before pressing the insert watermark button. This displays the pseudo-random watermark that is automatically generated, and again displays a comparison. The user can then extract the watermark with the button labeled extract watermark, showing both the watermark retrieved and the result of the comparison (see Figure 4). The generated and extracted watermarks may be saved for future use.

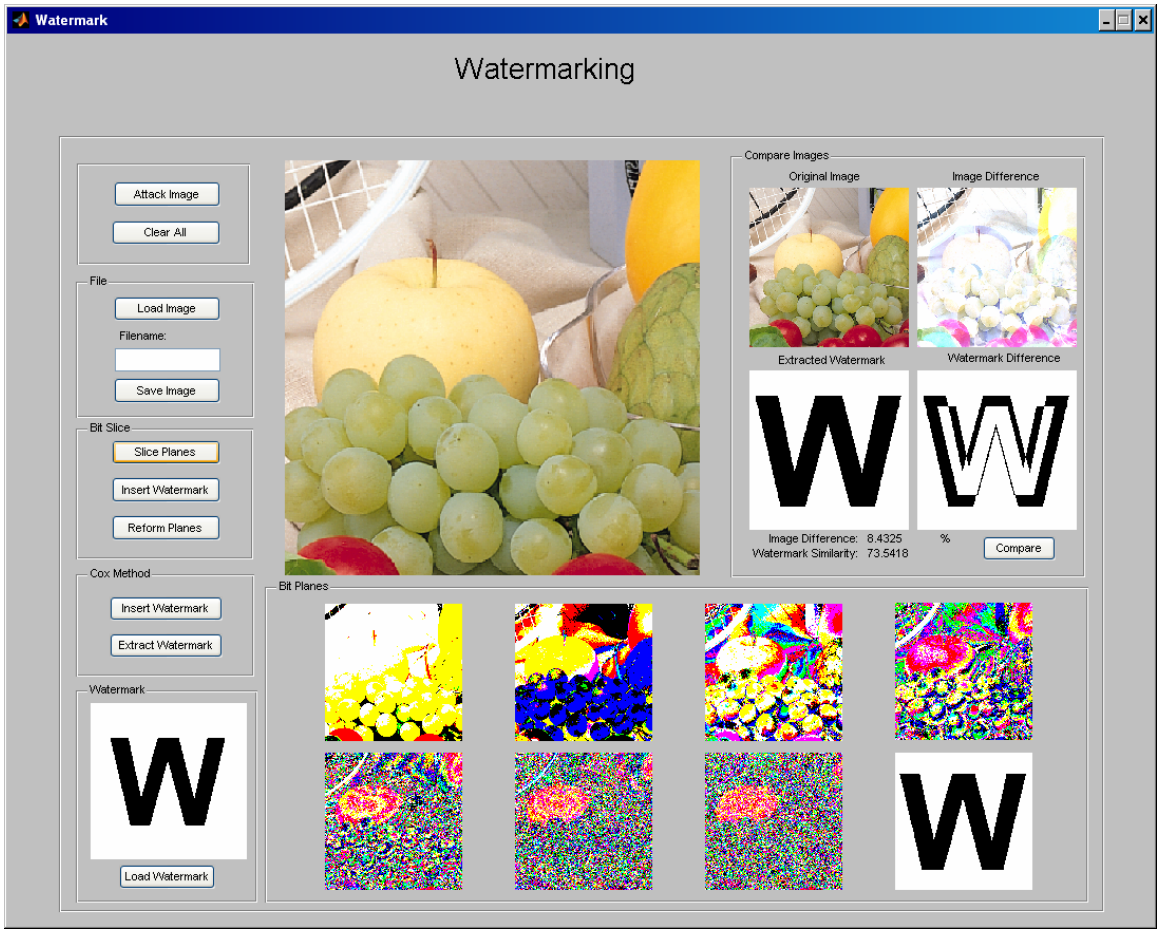

Figure 3: GUI example of bit-plane slicing.

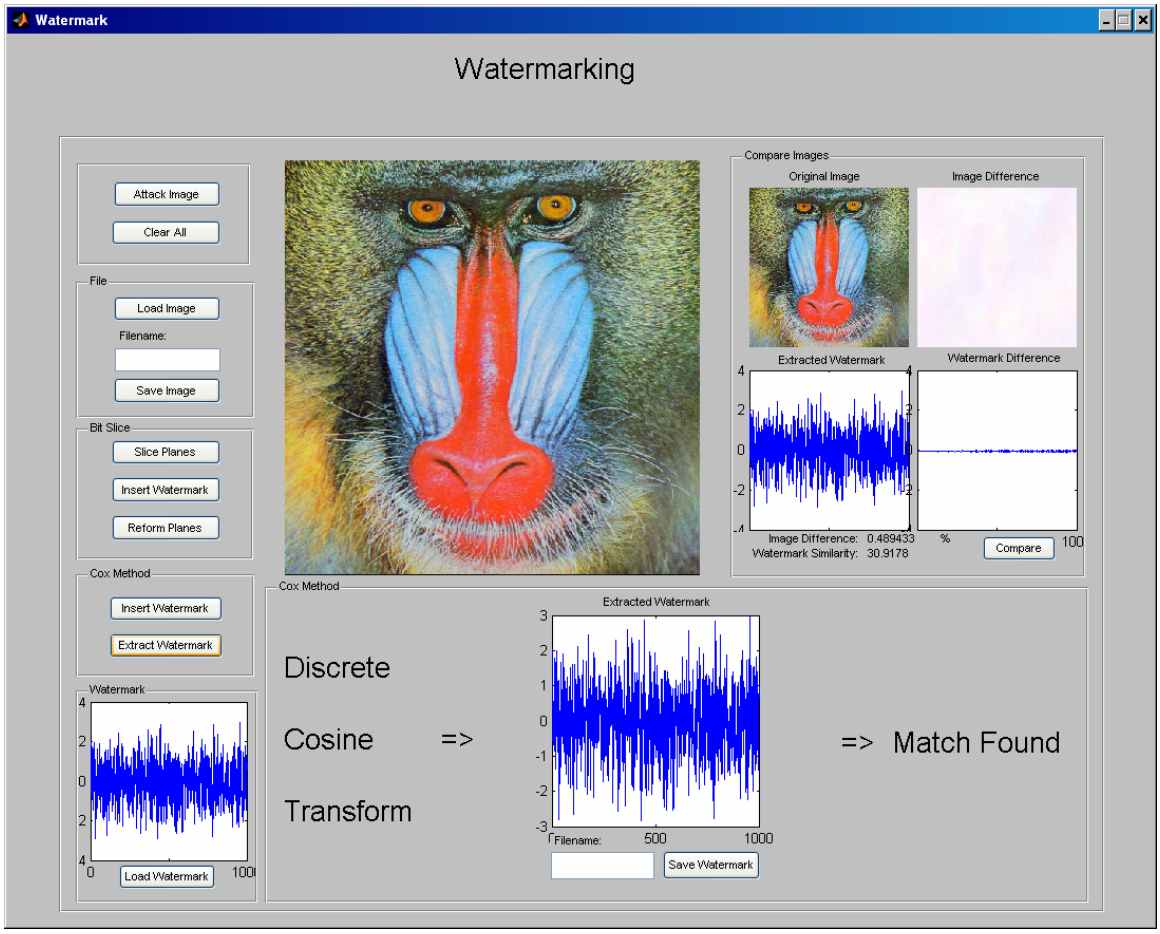

Figure 4: GUI example of Cox watermarking.

At any time the user may switch to the attack screen to modify the image. Whichever image is in the large main display when the user clicks the attack button is the one that is provided for attack. The image is shown both before and after modification, and the user may reset the image to the state it was at when the attacks were begun. Attacks may be selected from the buttons along the top, as seen in Figure 5. The difference of the attacked image from the original is shown, allowing the user to attempt various combinations of attacks to minimize visual difference while hopefully obtaining the greatest change to the watermark. The user may then return to the watermarking screen and the main image is replaced by the newly modified image. This allows users to use previously obtained data about the original image together with

the newly modified image. Users may save the image in the main slot at any time using the button on the left.

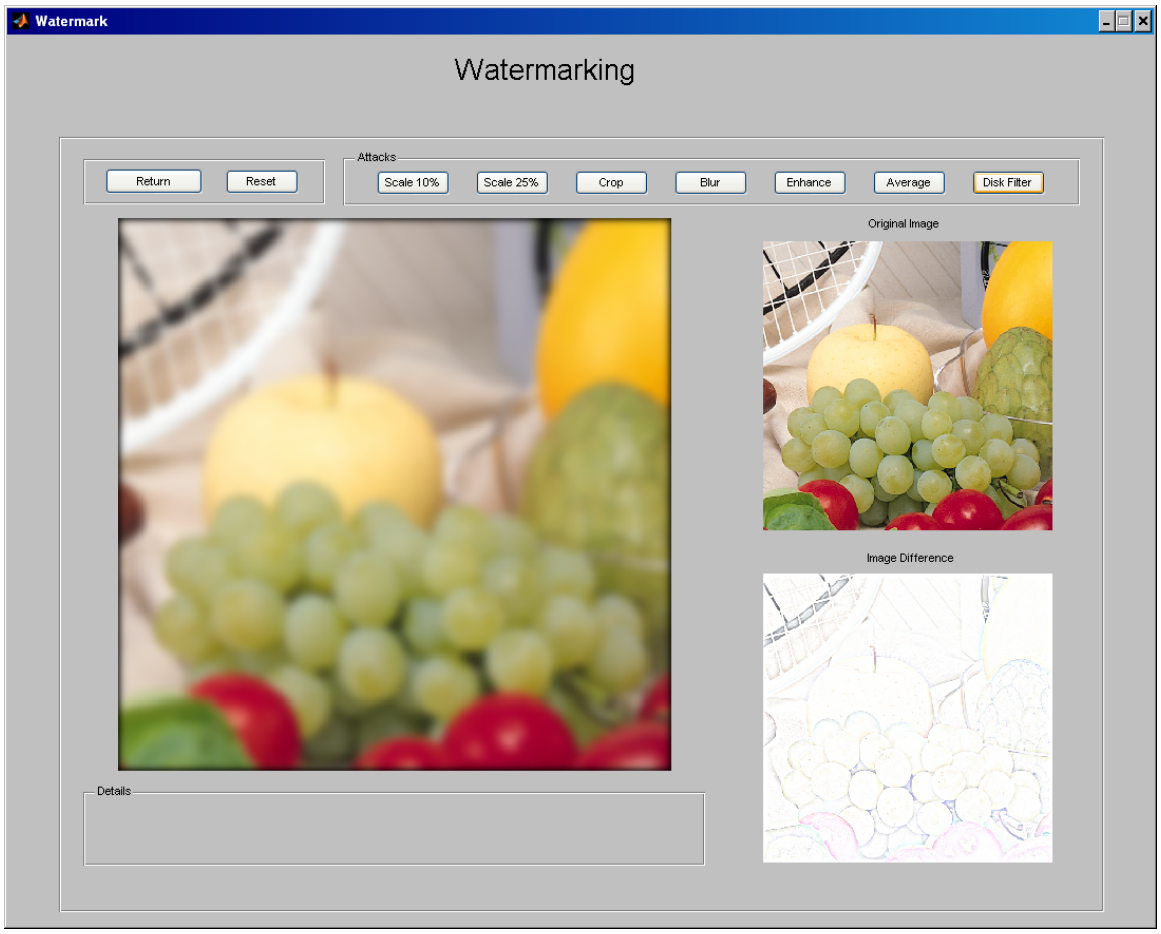

Figure 5: GUI example of image attacks.

Once the Matlab GUI and functions were working correctly, we started by testing the invisibility of watermarks when applied to an image. We tested the images with a variety of watermarks, but for consistency, chose the image that was most identifiable and clearly different from the original pictures (see Figure 6). We inserted the watermarks, but did not attack the watermarked images. Our methods produced extremely small changes, and so invisibility was not an issue for us with the watermarking methods we are using. We then ran various images

through the watermarking process and used a series of morphological image attacks to assist us in evaluating their relative recoverability and robustness. Two color images and one grayscale image were watermarked using both the bit-plane slicing and the Cox method. The results are given in Table 1.

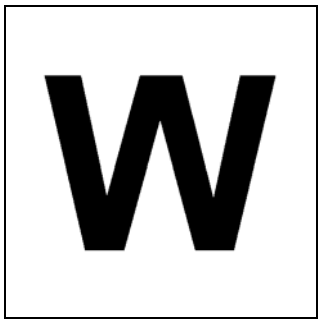

Figure 6: Example watermark image.

We applied each attack individually, as well as all of them in series. Differing the order and amount of attacks produced a wide range of results, but all of these combinations were too numerous to relate. Instead, we performed all seven attacks in the same order and recorded the changes. After each attack, the watermark was extracted and compared to the watermark inserted earlier. The change to the image, together with the proximity of the watermarks, shows the response of the extracted watermark to the attack. These came together to paint a picture of the watermarking method's robustness.

To evaluate robustness in terms of the Cox watermarking method, we compared the extracted watermark to the original watermark using the following similarity equation:

$$sim(w_{new}, w_{old}) = \frac{< w_{new}, w_{old} >}{\sqrt{< w_{new}, w_{new} >}}$$

This allowed us to determine the similarity despite the normalization of the image and whatever subsequent attacks it may have underwent. For the bit-plane slicing, we compared the retrieved watermark to that of the original using relative pixel values, and determined a threshold of 50% similarity. This value was chosen so that it could be seen if changes had been made to the image, while still being able to tell that the watermark was derived from the original.

Results

As can be seen in Table 1, the watermarking performed with the Cox method is more robust than the bit-plane slicing, as it withstands nearly all attacks. Cropping was the most detrimental to the extracted watermark, while image enhancement (unsharp filter) provided the best extracted watermark. Bit-plane slicing was extremely vulnerable to any sort of smoothing or averaging filters, as the watermark failed to be appropriately discernable after these attacks in all cases. Bit-plane slicing responded the best to scalar changes and performed acceptably when attacked with cropping. While not as robust as the Cox watermarking method, bit-plane slicing performs acceptably as a fragile watermark as long as smoothing or averaging is not involved. It does allow one to discern how much an image has been tampered with to some degree, but only in approximately half of the attacks.

Cox watermarking

Attack	Fruit (color) Percent difference (image)	Fruit (color) Watermark similarity**	Fruit (color) Attack passed	Fruit (grayscale) Percent difference (image)	Fruit (grayscale) Watermark similarity**	Fruit (grayscale) Attack passed	Baboon (color) Percent difference (image)	Baboon (color) Watermark similarity**	Baboon (color) Attack passed
None	2.3900	30.3942	Y	3.0096	30.0334	Y	0.9593	29.7591	Y
Downscale 10%	3.0832	28.4710	Y	2.9075	30.9950	Y	4.0944	31.7463	Y
Downscale 25%	2.6242	30.2646	N	2.3071	30.8746	Y	3.0320	29.8650	Y
Crop	9.1646	0.3878	Y	7.7568	9.2905	N	15.6439	5.6057	N
Blur filter	2.9755	29.4742	Y	0.5129	28.1221	Y	0.5375	25.9272	Y
Unsharp filter	4.4386	34.3781	Y	4.5493	34.0107	Y	12.9012	33.3689	Y
Averaging filter	2.6764	29.0849	Y	2.9755	27.8780	Y	3.6496	26.6065	Y
Circular averaging filter	4.9088	19.7060	Y	2.8214	19.1057	Y	5.6077	18.2659	Y
All attacks	9.1808	25.2709	Y	5.9913	2.1723	Y	13.0809	12.7705	Y

Bit-plane watermarking

Attack	Fruit (color) Percent difference (image)	Fruit (color) Watermark percent similarity	Fruit (color) Attack passed	Fruit (grayscale) Percent difference (image)	Fruit (grayscale) Watermark percent similarity	Fruit (grayscale) Attack passed	Baboon (color) Percent difference (image)	Baboon (color) Watermark percent similarity	Baboon (color) Attack passed
None	0.0747	100.0000	Y	0.0062	100.0000	Y	0.0868	100.0000	Y
Downscale 10%	1.1542	99.4479	Y	1.0160	99.4479	Y	4.4664	99.4479	Y
Downscale 25%	0.8031	99.6508	Y	0.7006	99.6508	Y	3.3037	99.6508	Y
Crop	8.4325	73.5418	Y	6.4222	73.5416	Y	16.7151	73.5418	Y
Blur filter	0.4393	41.7713	N	0.3903	45.8209	N	1.9268	35.7571	N
Unsharp filter	3.2094	38.3598	N	2.8818	37.2542	N	13.7557	35.4702	N
Averaging filter	1.2732	44.4001	N	1.1415	43.2154	N	5.1360	42.7482	N
Circular averaging filter	2.9370	37.1522	N	2.6839	35.8303	N	7.8927	36.0159	N
All attacks	8.4711	36.4893	N	6.4765	36.2665	N	15.0598	35.8293	N

** Calculated based on the original and extracted watermarks. A value greater than 6 is considered good.

Table 1. Results of various morphological attacks on the images in Figure 7.

Figure 7. Images that were watermarked and attacked. Experimental results are in Table 1.

The robustness of the Cox method can be seen in Figure 8. This example is in color, though both the Cox method and the bit-plane slicing method can handle grayscale and color images. A watermark was added to the original image, changing the image by only 0.60%. This watermarked image was then attacked using the circular averaging filter, which then makes the image 6.79% different from the original. Finally the watermark is extracted. Despite the change

in the image, the similarity rating is 23.73, well above the threshold value of 6. Similarity values tend to range from -15 to 35, with higher numbers indicating greater similarity.

Figure 8. Cox method example from left to right: Watermarked image (0.60% different from original), Attacked image (6.79% different from original), Extracted watermark (23.73 similarity)

The bit-plane slicing seen in Figure 9 below demonstrates its grayscale application. This method produces negligible changes when the watermark is applied as seen in the image on the left. The cropping attack is demonstrated in the center image to show the successful use of the bit-plane slicing method. It can be seen that the fruit is larger in appearance, thought the difference between the image and the original is only 6.42%. When the watermark is extracted from this altered image, the watermark is still clearly identifiable with 73.54% similarity to the original watermark. This result successfully indicates that a change has been made to the image, the goal of the fragile watermarking process. With further testing, this method could provide information regarding the amount of alteration and possibly the attacks used.

Figure 9. Bit-plane slicing example from left to right: Watermarked image (0.066% different from original), Attacked image (6.42% different from original), Extracted watermark (73.54% similar to original watermark)

Discussion

It is clear from the results of the experiment that the Cox method is far more robust than the bit-plane slicing method. The only cases that the Cox method failed to produce an extractable watermark were cropping attacks. With more time, the Cox method could be modified to apply the watermark to blocks of pixels throughout the image, rather than the whole thing. Due to this change, the method would begin to resemble that of the Podilchuk method, generally considered to be an improvement upon the Cox method. The program we created could easily be expanded to include many more watermark attacks, and provides an easy way to isolate the strengths and weaknesses of each method. By applying that knowledge, watermarking methods can be created that are more resilient and contain the best combination of tamper-proof techniques.

The bit-slicing method works acceptably as a fragile watermark, but is still too susceptible to smoothing and averaging attacks. It would need to be strengthened to be ready for application as a fragile watermark, but it provides an excellent basis for spatial domain

techniques. This method is also ideal for private images where tampering is not an issue because personalized images can be used as watermarks, as opposed to the pseudo-random identifiers relied on by the Cox method.

The GUI created allows users to apply and compare watermarking without prior knowledge, and can also be used to enhance ones understanding of special techniques. When dealing with images, it is useful to be able to approach the matter from a visual standpoint, and the GUI makes visual comparison and application much easier. We hope that this could be a learning tool for anyone interested in watermarking methods, or even expanded to act as a research aid.

Tasks Completed

Both team members put equal work into the project. We worked side-by-side in the lab for nearly the entire project. The following is a listing of tasks completed.

Task	Matt Elliott	Brian Schuette
Research	50%	50%
Proposal	50%	50%
Initial Report	50%	50%
GUI	75%	25%
Bit-Plane Methods	20%	80%
Cox Methods	40%	60%
Expirements	65%	35%
Final Report	50%	50%
Overall	**50%**	**50%**