AN IMPROVED AND IMPROVISED WAVELET WATERMARKING SCHEME

Boyd McKinnon
Utah State University, Logan, UT

ABSTRACT

This paper will discuss the modification and improvisation of two different wavelet watermarking techniques: Tree Quantization and Pixel-Wise Masking. Both techniques use Discrete Wavelet Transforms (DWT) of four levels to embed their watermarks in completely different methods. These methods have been modified from their original descriptions to assist in our use and testing. The watermark itself is made of a pseudo-random sequence of two numbers. We will combine the strength of both techniques along with the Human Visual System (HVS) to create an invisible watermark that is robust to common image processing attacks. The extraction process is blind for improving its practical use.

1. INTRODUCTION

Claims to ownership within the digital world have always been a hot topic. The need to prove ownership of intellectual knowledge or digital media is becoming more necessary. To solve this problem, individuals have begun to create watermarks similar to those that are located in a fine quality paper or on U.S. currency. Unlike digital watermarks, paper watermarks are visible to everyone who knows how to look for them. Today’s digital watermarks, however, have the aim to be invisible so that (1) the various media they are stored in are not visibly altered and (2) that others may be unaware that the mark is even present. Yet if needed there is a method to indicate a watermark embedded in the media in order to prove ownership.

As these invisible watermarks have improved over time, it has been found that there are key issues to keep in mind when trying to create an invisible watermark. Some of those are (1) the limitation of visual distortion, (2) the ability to retrieve the original item, (3) accurate detection, and (4) robustness against others who may ruin the watermark either knowingly or unintentionally [1] [2]. Another key factor for an effective invisible watermark is that its decoding process should be blind [3]. In other words, one should not need to have an unmarked original with them for the decoding process in order to prove the ownership of the image or watermarked item.

Many techniques have been used and tested in order to obtain these goals. Some past attempts included working in the spatial domain, adding noise to the busy portion of an image, transforming an image into its frequency domain and then changing specific frequencies to obtain a desired correlation value, and so forth. One major factor in being able to produce a better watermark has been the study of the Human Visual System (HVS) [4] which helps determine the maximum amount of change we can implement before the human eye can detect a difference. Knowing what we can add or remove without changing what a person sees is an immense help in creating an invisible watermark. Another technique that is being used more today is embedding the watermark in the Discrete Wavelet Domain (DWT) [2]; one such method is described in [5].

2. DYNAMIC TREE QUANTIZATION

In attempt to meet the qualifications of an effective invisible watermark, we plan to implement a new Dynamic Tree Quantization (DTQ) technique for embedding an invisible watermark into grayscale images and embed this watermark in the wavelet domain as described in [6]. In addition, we hope to improve robustness by using HVS [7] to allow the maximum change within the image without causing visible distortion. By combining and modifying these two processes, we will show improvements in both invisibility and robustness.

2.1. Preparation steps for using the DTQ method

There are two steps in using DTQ. The first is to perform a four level DWT decomposition upon the original image. Once this is accomplished, the family trees are placed into an array. Each pixel in the higher
has four corresponding children in the next decomposition level (See Figure 1). Then each of those pixels have their four corresponding children. In other words, each Parent pixel (the pixel in the highest level) is put into an array followed by its four children and then by its 16 grandchildren (the four children’s kids), totaling a length of 21 family members. Once all family trees are created they are combined randomly with another family tree to create a “Super tree” [6]. These Super trees are likewise randomly paired together in preparation for embedding the watermark through Super tree quantization.

Simultaneously, as each family tree is created the HVS values for each of the great-grandchildren are created using the formula found in [7]. Those HVS values are then averaged and stored to be averaged again with the average HVS value of whichever family tree it is paired with. The averaging of the two family trees’ HVS values creates an HVS value for the Super tree. We use these Super tree HVS values to determine our dynamic allowable error value which we will discuss and use later in the embedding process.

The second and final step before embedding the watermark into the image is creating the watermark sequence. This watermark sequence is a pseudo-random sequence of zeros and ones that are generated from a Seed value given by the individual doing the embedding. Using a watermark generated in this way allows for easier detection later; the watermark in question can be recreated in the same way that it was created to be embedded.

2.2. Embedding using the DTQ Method

Once the Super trees are created, their average HVS value calculated, and the watermark generated, the embedding process can begin. The first step in the actual embedding process is taking the first pair of Super trees and storing the sign bits of the Super tree that will have the watermark value embedded into it. (To determine which Super tree is to be embedded, check the watermark value. If it is a zero we embed into the first Super tree of the pair; if it is a one we embed into the second Super tree of the pair.) Once the sign bits are stored, we convert each Super tree to its binary equivalent. Then we use one third the average of the two Super trees’ HVS values to create our new maximum allowable error. This Error is used to determine the value of Qn for Each Super tree. (Qn is a position within each binary Super tree where, if all values below it are set to zero, the error caused is less than the maximum allowable error.) This is done for both Super trees. The larger of the two Qn’s is used for the embedding purpose to improve robustness.

When the value of Qn is determined, the value of the watermark is used to determine which Super tree will be quantized. With the appropriate Super tree selected the values from the position of Qn to the end of the matrix are set to zero (See Figure 2). Once that is accomplished the modified Super tree is converted back into decimal values and multiplied by the stored sign bits. Each family member of both family trees are then reinserted into the decomposition, overwriting their original values.

After all the values of the watermark have been embedded, the decomposition of the image is then sent through an inverse DWT process to create and return the final watermarked image.

2.3. Decoding the Watermark from the DTQ

To see if one’s watermark has been embedded into the image, you will need the Seed value that was used to embed the watermark in the first place. Take the watermarked image and perform a four level DWT decomposition as before, and then create the family and Super trees in the same method as used in embedding.
Table 1 Averaged Compression Test Results

<table>
<thead>
<tr>
<th></th>
<th>DTQ</th>
<th>QT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monkey</td>
<td>PSNR</td>
<td>47.52</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>1.00</td>
</tr>
<tr>
<td>Peppers</td>
<td>PSNR</td>
<td>46.77</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>0.99</td>
</tr>
<tr>
<td>Lena</td>
<td>PSNR</td>
<td>44.40</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>0.96</td>
</tr>
<tr>
<td>Elaine</td>
<td>PSNR</td>
<td>47.13</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>1.00</td>
</tr>
<tr>
<td>Boat</td>
<td>PSNR</td>
<td>46.93</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>1.00</td>
</tr>
<tr>
<td>Gold Hill</td>
<td>PSNR</td>
<td>48.14</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Also create the HVS value as before and the watermark using the Seed value.

After the Super trees are created, paired, and converted to binary, find the positions of Qn. If the position of Qn is higher for the first Super tree, assign the detected watermark the value of ‘0’; this shows that the first Super tree was quantized. If the second Super tree’s Qn is higher, assign a value of ‘1’ to the detected watermark for that pairing.

After all pairings have been checked, detected, and watermark values assigned, check the detected watermark values against the Seed-generated values. If they match add one to the probability total; if they do not match, subtract one from the total. Finally, after each value in the detected watermark is compared with the seed-generated version, divide the total sum by the length of the watermark; this will result in a probability of one or less.

As described in [6], there is still a possibility of receiving a false positive. Because of this, we chose to use the threshold value of 0.15 which, according to [6], will give a .0000161 chance of a false positive when the watermark length is 768 bits long.

3. EXPERIMENTAL RESULTS

Initially we tested our implementation of [6] and our new DTQ methods for relative numbers in order to compare and find how our results faired against the original Quantization tree method. Our encoding of the Quantization tree method [6] was less robust than what its paper stated, but we attribute these differences to the use of different versions of the same images and other trade secrets that might not have been discussed. Our new Dynamic Tree method was tested with over 12 different images (ranging from images of people, landscape, animals, etc) and 12 different seed values for robustness and rotation, pixel-shifting, compression, scaling, noise, and multi-marking attacks. For brevity, not all of our test results will be shown but can be made available upon request. For a majority of our tests we chose the Seed value of 37 (a randomly chosen number; numbers 1:10 and 55 were also tested). Our error value for the original Quantization tree was set at the value of 100.

The image “Lena” seemed to cause the most problems in the invisibility aspects of our watermarking; we will discuss our theory on why this was later. In general, most images accepted the embedding of a watermark without visible distortion equally well. As with all testing, some subjects responded better than others depending on the image itself and the Seed value used.

3.1. Undisturbed Watermark Results

Table 1 shows the average PSNR (Peak Signal to Noise Ratio) values and probability of watermark detection (P) from the use of 12 different seed values. Both the original Quantization tree (QT) and the new Dynamic Tree Quantization (DTQ) methods have high detection probabilities and PSNRs well above 35 dB. All images, regardless of content, were successfully able to retrieve the watermark that was embedded. We used the probability threshold of .15 to determine if the watermark we were searching for was present [6]. This value was discussed by the authors of the original method wherein they state that by selecting a threshold at a given level, one can reduce the chance of a false
### Table 3 Difference in Multi-Watermarking Results

<table>
<thead>
<tr>
<th></th>
<th>(P of DTQ) - (P of QT)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Monkey P</td>
<td>0.40</td>
</tr>
<tr>
<td>Peppers P</td>
<td>0.51</td>
</tr>
<tr>
<td>Lena P</td>
<td>0.48</td>
</tr>
<tr>
<td>Elaine P</td>
<td>0.43</td>
</tr>
<tr>
<td>Boat P</td>
<td>0.47</td>
</tr>
<tr>
<td>Gold Hill P</td>
<td>0.43</td>
</tr>
</tbody>
</table>

According to their paper, a threshold level of .15 would allow for a probability of falsely finding the watermark to .0000161 percent of the time when the watermark length is 768 bits long [6].

#### 3.2. Detection of Attacked Watermarks

The attacks we tested our implementation with were: Scaling and Histogram equalization, noise addition, over-watermarking, bitplane removal, rotation, JPEG, compression, filtering, and circular pixel shifting. All of our attacks were performed using default Matlab 7.0.1 settings whenever possible.

Both methods were still able to detect their watermarks after our images were rescaled, but, as shown in Table 2, the original method returned a higher probability of detection than the DTQ method. When using the default noise settings during our noise testing, we found that this type of watermark could only handle “Salt & Pepper” noise. The possibility exists that these watermarks can handle other noise if less amounts of that noise are added. (The amount of noise added by the default values caused major visual distortion within the image). This notion of adding a smaller amount of noise in order to not distort the image and test for watermark recovery has not yet been tested.

When testing for robustness against over-watermarking or multi-marking an image, we found a clear difference between the two methods. We use the claim that even if an attacker knows the type of watermarking technique that you used they would not be able to destroy your watermark by embedding multiple watermarks of the same type into your watermarked image [6]. Our first watermark was embedded into our test images with a Seed of 37. Then four additional watermarks were applied to the same image, Seed values beginning at one and continuing through four. The results in Table 3 shows the difference between the two methods probability of detection (DTQ-QT) which clearly indicated that DTQ method was able to more easily find the original watermark.

After testing for robustness against over-watermarking, the testing for bitplane removal was performed. We found that both methods have a resistance against bitplane removal through the three lower (least significant) bitplanes. When additional bitplanes were removed the watermarks were no longer able to be detected.

Rotation, like removing bitplanes, caused difficulty for both methods. Neither were able to handle any significant change; the most that they could reliably handle was that of .25° change in either direction. The detector was no longer able to consistently find its watermark at any angle larger that .25°.

The tests results shown in Figure 3 convey that these embedding and detection methods can handle being JPEG compressed with a quality factor of 40 percent and higher. The original method did well for most images through the 30 percent quality compression. For the filter attacks, both processes were about equal; neither stood out as more robust than the other. In the pixel-shifting test it was clear that the Tree method performed better than Pixel-Wise Masking [7]. Even with a one pixel shift, the Pixel-Wise method lost track of the embedded watermark, while the Tree methods seemed to handle pixel shifts fairly easily and were even more robust when the pixel shifts were a multiple of four.

### 4. DISCUSSION

After running these tests, we temporarily tried a few different varieties to further our understanding of the abilities of these watermarks. These resulting changes...
Figure 4 Distortion on Lena (Close-up)

were insignificant and only assisted in our understanding of the faults and strengths of our watermark embedding and decoding schemes. It is true that both Original Quantization Tree and DTQ techniques produce fairly good results. In attempting to incorporate the strengths from the Pixel-Wise method (namely their HVS equations) to assist in determining our dynamic error value, we were able to decrease some of the visible glitches that were occasionally caused during the watermarking process.

In trying to improve the original Quantization Tree method [6] with the HVS used in the Pixel-Wise method [7], we found that even though the results for our new dynamic version are acceptable and blindly detectable, the resulting watermark that is embedded seems to be more fragile than the original version. The dynamic determination of our error value in areas in which less error was allowed made it less feasible for the detector to find the correct watermark value because of the smaller differences between the two Super trees.

The ability to withstand an attack of multiple watermarks of a similar creation is the one clear advantage of the new DTQ method. Yet this reason alone does not justify the implementation of the HVS into the Quantization Tree method; the benefit might not be worth the additional time required since the original Quantization Tree method is still able to detect the watermark.

The benefits that might make this dynamic determination of our error value worthwhile is not represented in the results from testing but only by actual visual inspection of the resulting watermarked images. The left side of Figure 4 shows a portion of the Original method’s watermarked version of “Lena” using a 55 for the Seed value, while the right side shows the same portion using the DTQ method. Upon closer inspection, one can see that some of the visible flaws (blocked distortions) within this image have decreased by using the new dynamic error detection process. This shows that it clearly has benefit in reducing distortions caused by ill-advised pairing of family trees (which we will discuss in the future improvement section).

4.1. Possible Future Improvements

After working with the Quantization Tree method for watermarking and implementing the dynamic version of the Quantization Tree, we have come to the conclusion that there is an inherent flaw in the tree method as described [6]. The block distortion in some of the watermarked images is due to the pseudo-random pairing of the trees and the Super trees. Due to the fact that this is a random process, an error can occur in the quantization process. If the Qn [7] of the two Super trees differ greatly, it can cause a problem in the final image because one Super tree might reach its maximum error by Qn of 80 while the other might achieve the maximum allowable Qn of 336 [6]. If the first Super tree that achieved its max Qn by 80 is quantized to the Qn of 336, a visual distortion will be created within the final watermarked image.

These distortions can be seen in the “Lena” image that we have used for our testing and implementation of the two methods. Due to the smoothness of her skin tones and the contrast between different parts of the image, visual block distortions were created, but if additional noise was added to the image (specifically tested Matlab’s imnoise poisson function) before watermarking “Lena,” no visible distortions were created. This might be why the version of “Lena” that was selected in the Original Quantization Tree paper was selected (See [6] for image of their Lena image).

A possible future improvement to the Quantization Tree method is to use a different version of the HVS equation to pair the created trees and Super trees of an image. This should reduce the opportunity of larger mismatches between the trees when they are determining the value of Qn, thus reducing possible visual distortion upon the final watermarked image. Additional testing would then be needed to determine what error to allow within these new pairings.

Another possible improvement is to incorporate methods to assist finding a reference point within a watermark in case the image was rotated or cropped. Doing so would increase this method’s ability to handle rotation attack which is one of its main weaknesses.
5. REFERENCES


