Department of Electrical and Computer Engineering
University of Victoria
ELEC 483 - Digital Video Processing

Invisible watermarks

Report submitted on:	21 April, 2017
To:	Prof. P. Agathoklis
Names:	H. Emad (V00757795)
	T. Stephen (V00812021)

Abstract

The purpose of this... Well come back and write this part after the rest of the report is finished.

1 Introduction

The act of watermarking is synonymous with digital documents. In fact, one would be hard pressed to find any sort of document, photo, or video on the internet, without some sort of watermarking or digital rights statement. The problem is that traditional watermarking are the relics of a bygone era. Most visual watermarking today relies on the brute-force technique of physically overlaying a secondary text or image branding over top of the content. While this method makes the watermarking obvious, it does not employ any clever techniques, or even attempt to minimize the loss of visual quality of the original content.

This lack of an elegant digital watermarking solution led us to attempt to create a digital, invisible, embedded watermarking technique, for use on digital image files.

2 Theory and analysis

Invisible digital watermarking means the ability to embed data in an image, in a manner that is recoverable, without causing significant alteration to the image or lowering its visual quality. The key components of this solution involve taking the DCT coefficients of an image, quantizing those coefficients, and embedding a numerical data string containing the watermark information, as numerical offsets, into the DCT coefficients of the image.

The extraction of the watermark is done through comparing the DCT coefficients of the original quantized image, with the DCT coefficients of the watermarked image. The offsets are mapped back to the numerical data string containing the watermark information.

Opportunity to talk in detail about the meaning of DCT coefficients, quantization

3 Implementation

This project was implemented through MATLAB and carried out in two stages outlined below. The encode stage takes in an image file and a watermark phrase and outputs a watermark embedded image. The decode stage takes in a watermarked image and an original image and outputs the extracted watermark in text format.

3.1 Encoding

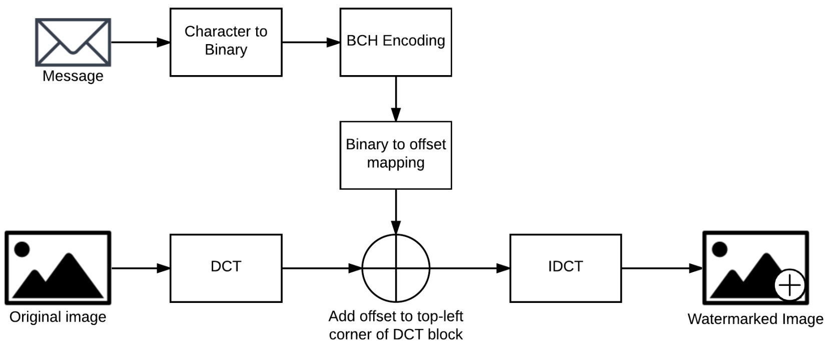

Figure 1: Watermark encoding process

The watermark encoding function of this project is implemented through a MATLAB m file. The file reads an image as input and also reads a string from the user as the watermark message. The message must be in basic alphanumeric format and the length is limited to less than 33 characters. This character limit is due to the requirements of the error correction methods, which will be discussed in the following sections.

The blockproc function is used to obtain the 8x8 DCT coefficients of the input image. The watermark string is mapped letter by letter as a digit from 0 to 26, corresponding to its frequency in the english alphabet. By employing this mapping method, we ensure that the most frequently used letters are mapped to the smallest numerical values. The decimal values are converted to a base 6 binary format, and are then converted to a single long binary string.

In order to facilitate error correction, BCH coding was employed. BCH coding takes a binary sequence and appends a binary correction code sequence to it. This long binary string can be fed back into the BCH function, in the decoding stage, to produce an error-corrected binary sequence for our use in mapping back to the correct watermark message.

The binary sequence with the BCH correction code is then inserted into the top left corner of each of the DCT blocks of the image, bit by bit, as an offset. By targeting those corners, the impact of the offsets on the quality of the overall image is minimized. Once the watermark sequence is embedded, the image is inverse DCT and outputted to the user, ready for decoding.

3.2 Decoding

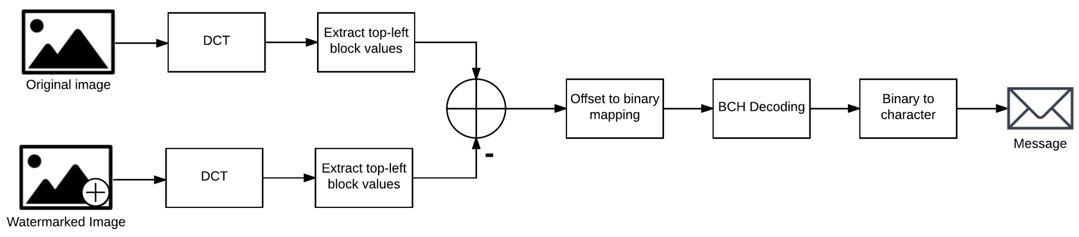

Figure 2: Watermark decoding process

Similar to encoding, the decoding function of this project was also implemented through a MATLAB m file. The file reads the watermarked image as an input, as well as the original, unwatermarked image. The DCT coefficients of both images are acquired and the top left corner values of each block is compared to the corresponding value from the other image. The difference is the offsets string.

This offsets string is then forced to a binary format, truncated to the correct length, and sent to through MATLABs BCH decoding function, where the correction code is employed in extracting the binary version of the original watermark message. This binary code is run through an inverse mapping process and the final alphanumeric message is presented to the user as the extracted watermark.

4 Examples

[Before and after images of watermark embedding

phrases embedded vs extracted

Matlab code verbatim

5 Discussion

All the real world things that went wrong in our implementation and the steps we took to fix them

the constraints we ended up imposing on the watermark phrase

6 Conclusion

Resounding success!

if we wanted to sink more time into this, we could improve/extend it in this way

heres how we see our project being useful in the real world