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

Digital watermarking is the process of embedding information in an image to establish ownership and prevent unauthorized distribution. Traditional image watermarks that place translucent text or images over a source image must balance a degraded viewing experience with ease of removal. Watermarks that are placed in the corner of an image rarely cover up important image features but can be removed trivially through cropping. Prominent watermarks can discourage unauthorized image use (see Figure 1) but are impractical for use cases where a rights-holder wants to prevent dissemination of a high fidelity, minimally distorted image.

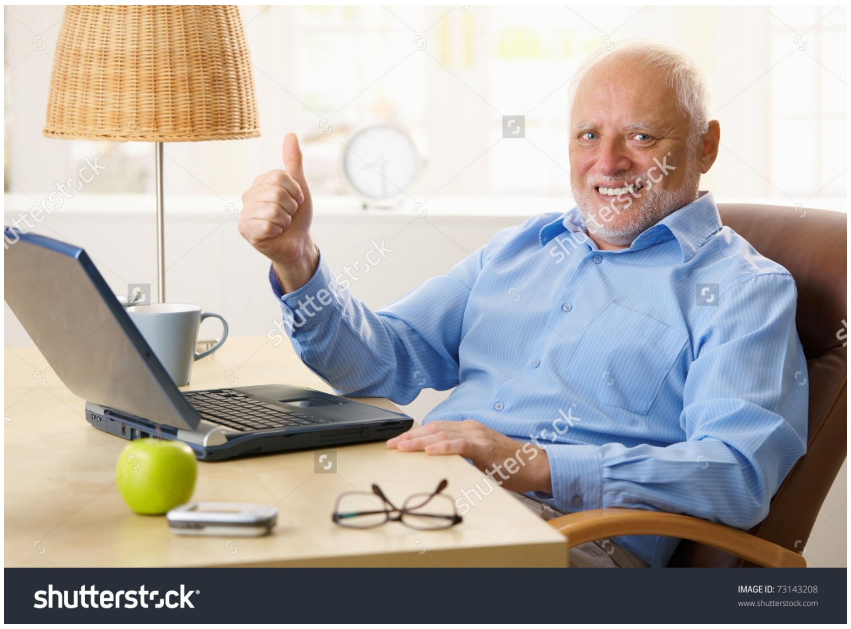

Figure 1: Stock image sites use overlay and footer watermarks to discourage unauthorized image use [1]

Image metadata (e.g. EXIF, IIPC, PLUS, Dublin Core) exists in the digital file header and leaves the source image unaltered. However, file metadata offers no protection against removal. cite needed An ideal digital watermark should be as inseparable from the source image as its namesake.

In this paper we demonstrate a procedure for watermarking digital files with almost no degradation of source image quality. We take a plain language phrase and encode it in the image

DCT blocks to make it an intrinsic part of the image that cannot be removed. The watermark phrase is recovered by comparison with an original, unwatermarked image.

2 Theory and analysis

We do the same thing as those guys (spooky terrorists?) who passed messages by offsetting rgb values

Unless specifically designed to be otherwise, the largest frequency component of an image is its DC component. The DCT process typically yields a DC value over 1000. Altering this value by ± 1 results in almost no change to image quality. This is the core principle at the heart of many compression algorithms. cite needed

Relative image quality is determined by the Peak Signal-to-Noise Ratio (PSNR) value, representing the effect of noise from the $DCT \rightarrow IDCT \rightarrow DCT$ process and distortion from the watermark.

BCH codes?

3 Implementation

The proof of concept was implemented in MATLAB with separate encode and decode stages. The encode stage takes an image file and a watermark phrase and outputs a watermark embedded image. The decode stage takes the watermarked and original images and outputs the extracted watermark message.

3.1 Encoding

Figure 2 shows the encoding process. The encode file prompts the user to supply a source image and message to embed. The message must be in basic alphanumeric format and the length is limited to 33 characters as a requirement for the BCH code.

The letters of the watermark message are associated with a number corresponding to its frequency in the English language.[1] This ensures that letters the most frequent letters will have

[1]See: https://en.wikipedia.org/wiki/Letter_frequency

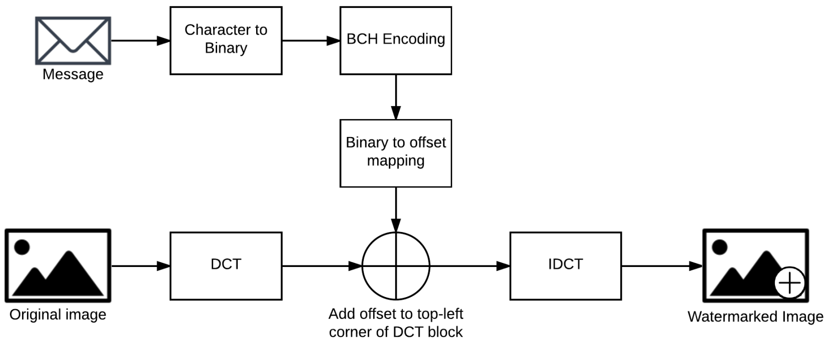

Figure 2: Watermark encoding process

more 0s, and thus a smaller distortion, when mapped onto the source image. The numbers are converted into 6-bit binary values[2] and form a binary sequence.

The *DCT* $\rightarrow$ *IDCT* $\rightarrow$ *cast to* `uint` process is *lossy*, primarily because of the downcast to `uint`. To minimize ambiguous downcasting we use an offset generation mapping $x \mapsto y$ and recovery mapping $y' \mapsto x'$ defined as:

$$y_i = \begin{cases} 0, & x_i = 0 \\ 2, & x_i = 1 \end{cases} \qquad x'_i = \begin{cases} 0, & y'_i \leq 0 \\ 1, & y'_i > 0 \end{cases}$$

Despite this precaution, recovery errors occur with a rate $\approx 1\%$. In order to correct this, we apply a BCH code to the binary sequence to allow for error detection and correction. We use a BCH code with $n = 255$, $k = 199$ capable of correcting 7 errors (2.7451% error rate). To apply the BCH code, we pad the binary watermark sequence with 0s so it has 199 bits. The BCH encoder returns a sequence of 255 bits that can now be embedded into the image.

Next, we obtain the 8×8 block DCT of the image. Each binary value is mapped to the appropriate offset and added to the DC component of the DCT block, starting with the top left block. Once all 255 bits have been added to the DCT blocks, the image is reconstructed via Inverse DCT and written to an output file.

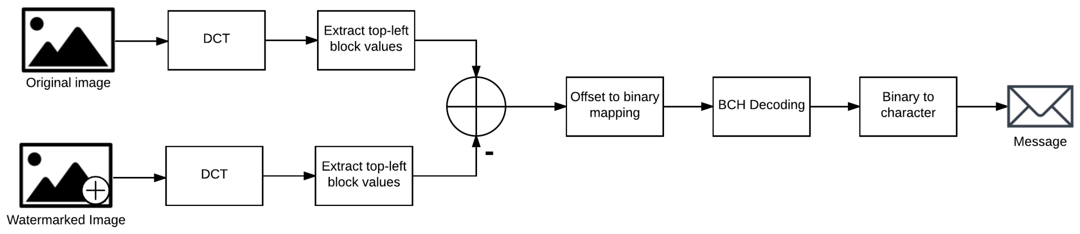

Figure 3: Watermark decoding process

3.2 Decoding

Figure 3 shows the process for retrieving the watermark message from the image. This process is not "blind" since it requires the original image to extract the offset values in DCT DC coefficients. Moreover, the values of n, k for the BCH code and the structure of the character-to-binary map *must* be shared by the encode and decode processes.

After recovering the binary sequence, 6-bit words are mapped to their original characters with padding characters mapping to `"  "` empty space. The user is presented with the extracted watermark.

4 Data

Working versions of encoder and decoder scripts are available at https://github.com/trstephen/elec483. The scripts will embed and extract a text watermark from a provided grayscale or color image.

To evaluate the effects on image quality with spatial and frequency alterations, we compared the method described in Section 3 to one that embedds the binary sequence without the BCH code directly in the image grayscale values. This spatial embedding method produced a PSNR of 49.3924, significantly less than the PSNR of 58.7447 with our frequency embedding method. why?

We also attempted to determined the best "layer" to embed the watermark in a color image. We used a couple color images, each with RGB and YCbCr encoded variants. The results in Table 1 show that embedding a watermark in any RGB layer causes less of a decrease in PSNR than embedding it in any YCbCR layer. why?

[2] The inclusion of letters *and* numbers in the watermark character space pushes the total number of characters over 32, thus requiring 6 bits.

Table 1: Effects on PSNR after embedding a watermark in a single layer

Source Image	Layer	PSNR
peppersYCbCr	Y	58.91533
	Cb	58.12800
	Cr	58.95000
peppersRGB	R	63.60164
	G	63.74127
	B	63.70763
fruitsYCbCr	Y	58.88691
	Cb	57.95389
	Cr	59.07317
fruitsRGB	R	63.88637
	G	63.69708
	B	63.59464

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

How to defeat!

Okay for applications that have the time to do DCT/IDCT

5.1 Further work

Embed / recover the watermark in such a way that the original image isn't needed for comparison.

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

References

[1] StockLite. (2017). Happy senior man giving thumb up, sitting at desk using laptop computer at home, Shutterstock, [Online]. Available: https://www.shutterstock.com/image-photo/happy-senior-man-giving-thumb-up-73143208 (visited on 04/15/2017).