Digital Watermarking for Copyright Owners

CSF Group Number: 4

Diogo Monteiro, Joao Santos, Pedro Reganha

Instituto Superior Tecnico

diogo.p.monteiro, joao.nuno.santos, pedro.reganha

Abstract

With the advent of the internet, artists nowadays have even more difficulties in proving that their artwork is indeed from the creator. With this in mind, we developed a tool that is able to protect the copyright of these artworks, based on investigation made in the subject of digital watermarking.

1. Introduction

Nowadays, it is easy for anyone to claim their ownership on any photograph found online. This leads to the problem, that the real owners of photographs (possibly, artists), maybe unable to prove their ownership of the art. This lack of copyright protection is disturbing. With this in mind, we developed a tool that any artist can use to sign their artworks. In it's essence, it is a framework designed with two objectives in mind:

- To provide artists with the tool to protect their copyrights, being able to watermark their creations, and to recover their images in case of partial destruction of these;
- To provide developers with a simplified way to develop algorithms and to add them onto the framework, simply having to abide by a small set of rules.

With this solution in mind, we hope to be able to protect copyright owners against artwork tampering and edition, and identify the owner of a copyrighted artwork.

In this paper, we present to the reader, by order, the architecture of the tool itself (2, referencing how the framework is implemented in a white-box perspective), the algorithms we implemented (3), an evaluation of how the tool processes its input and the benchmark results after a batch of operations on the processed image 4, and finally, the closing ideas on the subject (5).

2. Architecture

In terms of the architecture of the tool, it can be decomposed in two main components:

- The framework itself, that provides developers an easy way to implement and test algorithms;
- The algorithms, that can be developed by independent developers and can be tested from within the framework. These need to abide by a small set of rules, and there are already three algorithms shipped by default.

2.1 Framework

The framework was implemented in Python (2.7), with open-source modules in order to process the images, with the addition of algorithms from other developers in mind. Before explaining how the framework is structures, the following term should be clarified: A **Digital Watermark**, as described by Cox et al. [3], is "an identification code that is permanently embedded in the data". This said identification code should be preferably invisible, and it remains even after decryption operations over the "signed" data. This allows any artist to "sign" their artwork, and to protect their copyrights.

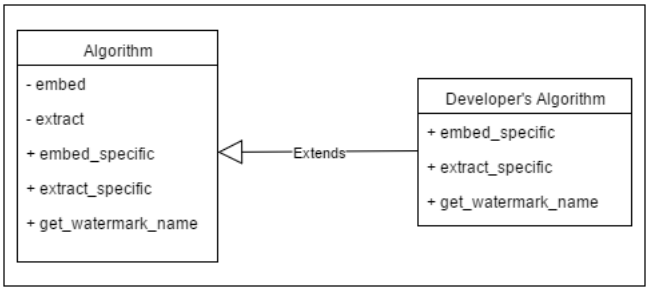


Figure 1. A graphical example of how the developer must implement his algorithm

In it's essence, all a developer needs to do in order to add a new algorithm is to extend the abstract class **Algorithm**, and to override the methods *embed_specific* , *extract_specific* and *get_watermark_name*. The developer does not have to worry himself with image opening or saving the image, having only to implement the algorithm, as the abstract class already implements this logic.

2.2 Testing the algorithms

The framework also provides the option to create unit tests in order to test the algorithms that the developer produces.
In order to do so, the developer simply has to import the algorithm and create the desired tests, as, if we recall from the previous section, it is already explained that all logic in terms of image treatment before applying the algorithm is already taken care off.

3. Algorithms

In this section we will present the three algorithms that we incorporated in our solution.

3.1 Cox algorithm

The Cox *et al.*[3] algorithm is one of the first approaches to transparent watermarking, requiring both original and watermarked image for recovering the embedded watermark, in our implementation we also use another file that is basically the watermark, later used for comparisons. In Cox[3] the watermark is embedded in the

largest magnitude DCT coefficients in order to provide a more robust solution to the compression algorithms(like JPEG compression) than the LSB-type methods like [10].

Our implementation of this watermarking algorithm consists of 5 steps:

- Calculate the DCT and identify the best regions for watermark embedding.
- Construct the watermark X from a normal (Gaussian) distribution.
- Insert the watermark in the DCT using equation 1.

$$v_i \leftarrow v_i + \alpha * x_i \tag{1}$$

- Store the watermark image in a file, so we could use it in extract method.
- Calculate the inverse DCT to generate the watermarked image.

To extract the watermark and see if/how much of the watermark still exists in the watermarked attacked we do this 3 steps:

- Compute the DCT of the watermarked image, and the original one.
- Compute the difference between the two and calculate the gamma(a measure that shows how much the image as been tampered with).

3.2 DWT Based Algorithm

Cox algorithm had some problems when it came to attacks like contrast and cropping piece of an image, so we committed ourselves to find a more resilient algorithm to those attacks. In our search we found many DWT Based Algorithms like Akter and Ullah [1], Al-Haj [2], Rahman et al. [7], Yahya et al. [9].

Based on those algorithms we created a simple version, for embedding we do the following steps:

- Compute the DWT of the image we want to watermark.
- Compute the DCT of the watermark we want to use.
- Compute the DCT of the LL quadrant of the DWT, and add the DCT of the watermark we calculated.
- Do the inverse DCT of the obtained calculus.
- Do the inverse DWT to obtain the watermark image.

To recover the watermark we original image and the watermarked one.

- We first calculate the DWT of the original image, O and of the watermarked one, W.
- We compute the DCT of O and W, resulting in DO and DW respectively.
- Using the previous values we obtain the DCT of the supposed watermark, DI, using equation 2.

$$DI_i = (DW_i - DO_i)/\alpha \tag{2}$$

- Do the inverse DWT of I and like so we recover the watermark.

3.3 Data Recovery and Tampering Detection

Based on the studies made by Lin et al. [4], we decided to implement a data recovery algorithm. More precisely, recovery of the image. The paper also presented a way to detect the tampering of the watermarked images, based on processing of pixel blocks. In its essence, the algorithm provides a way of storing information of a given sector of the input data in another sector, this way being able to recover part of the image based on the rest of it.

Figure 2. The test image without watermark.

Needless to say, if a great part of the input data is missing (for example, if the image is cropped), it won't be viable to recover completely.

4. Evaluation

In this section we describe the set of benchmarks (section 4.1) we developed to understand the pros and cons of each algorithm that we have implemented. In section 4.2 we describe the obtained results.

4.1 Benchmarks

We developed a tool to automate the testing on watermark attacks. The benchmarks we considered and implemented consist on the following attacks to a watermarked image:

- Add 20% of contrast.
- Unsharp mask (a technique to increase sharpness in images).
- Mode Filter (a type of low-pass filter).
- Median Filter (a type of low-pass filter).
- Rotate 3 degrees counterclockwise.
- Add noise.
- Add gaussian blur.
- Use JPEG compression with 10% quality
- Draw white squares at the middle of the image (sizes: 5%, 15% and 30%).
- Double the size of the image, then reduce to half again.
- Reduce the size of the image to half, then double the size.

To measure how imperceptible the watermark is on an image, we used the Peak Signal Noise Ratio between the original image and the watermarked image. This ratio, measured in db (decibel) is higher when the watermark is imperceptible and lower when the watermark is perceptible.

$$PSNR = 10 \times \log \frac{max(I_w, I)^2}{\sqrt{mean[(I_w - I)^2]}} \tag{3}$$

To measure how similar the original watermark and the extracted watermark from an attacked image are, we use the Pearson correlation coefficient. A value of gamma close to 1 states that watermarks are similar, and a value of gamma close to 0 states that the extracted watermark is completely destroyed.

4.2 Results

The image we used to perform all tests for this report is the Lena image (figure 2), the standard image to perform tests on image processing since 1973. Due to space constraints, we are not able to show the results of every benchmark. However, we are providing the image results as standalone files.

Figure 3. Watermarked image with Cox algorithm

Figure 4. Watermarked image with DWT algorithm

Figure 5. Inserted watermark image

Regarding the Cox algorithm, the image we obtained after the embedding process of the watermark is illustrated at figure 3 and has a value of Peak Signal Noise Ratio of 44.74, stating that the watermark is imperceptible.

According to table 1, the Cox algorithm shows poor results when attacked with slight changes in contrast, white cropping and rotation. Our focus goes specially on the fact that a simple rotation of three degrees completely destroys the watermark, not leaving the attacked image destroyed.

Next we embedded figure 5 as a watermark of the Lena image and the result can be found at figure 3. The value of Peak Signal Noise Ratio is 38.1, stating that the watermark acceptably imperceptible.

The table 2 shows the extracted watermark for each attack and the respective gamma value. The attacked images look very similar as the ones in table 1. Compared to Cox algorithm, the DWT algorithm is more resistant to contrast changes and white square cropping, but fails miserably when the image is compressed or blurred.

About the recovery algorithm, we watermarked the Lena image (again) an then we've made a small change on the watermarked image. This result can be seen in figure 6. Then, we tried to recover the original image using our algorithm, the result is present on figure 7.

5. Conclusion

Concluding, we developed a stable tool that is able to watermark images, protecting the copyright of the owners of said works of art. For future objectives, we intend to release the code base onto the

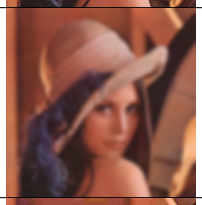
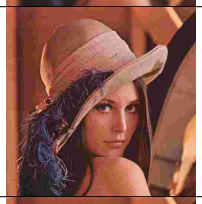
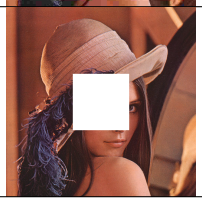
Attack	Attacked Image	Gamma
Contrast		0.34
Unsharp Mask		0.96
Rotation		0.38
Noise		0.97
Gaussian Blur		0.96
JPEG Compression		0.91
35% white square		0.35

Table 1. Cox algorithm benchmark results

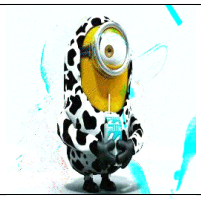
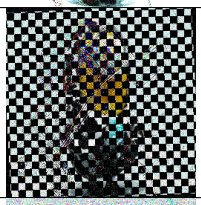
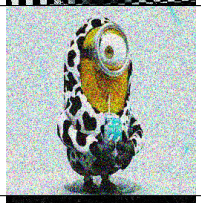
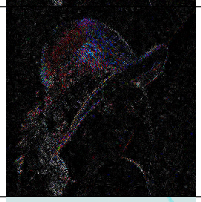
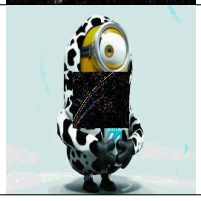
Attack	Extracted Watermark	Gamma
Contrast		0.95
Unsharp Mask		0.86
Rotation		0.20
Noise		0.79
Gaussian Blur		0.03
JPEG Compression		0.04
35% white square		0.87

Table 2. DWT Algorithm benchmark results

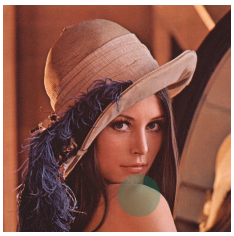


Figure 6. Tampered watermarked Lena image

Figure 7. Tampered watermarked Lena image after recovery

open source community, expecting that more developers are able to pick up the framework and develop their own algorithms. Also, we expect to develop a Graphical User Interface, so that it makes it easier for the end user to operate.

References

[1] A. Akter and M. A. Ullah. Digital watermarking with a new algorithm.

[2] A. Al-Haj. Combined dwt-dct digital image watermarking. *Journal of computer science*, 3(9):740–746, 2007.

[3] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *Image Processing, IEEE Transactions on*, 6(12):1673–1687, 1997.

[4] P. L. Lin, C.-K. Hsieh, and P.-W. Huang. A hierarchical digital watermarking method for image tamper detection and recovery. *Pattern recognition*, 38(12):2519–2529, 2005.

[5] A. Piva, M. Barni, F. Bartolini, and V. Cappellini. Dct-based watermark recovering without resorting to the uncorrupted original image. In *Image Processing, 1997. Proceedings., International Conference on*, volume 1, pages 520–523. IEEE, 1997.

[6] C. Podilchuk, E. J. Delp, et al. Digital watermarking: algorithms and applications. *Signal Processing Magazine, IEEE*, 18(4):33–46, 2001.

[7] M. Rahman et al. A dwt, dct and svd based watermarking technique to protect the image piracy. *arXiv preprint arXiv:1307.3294*, 2013.

[8] M. D. SWANSON, M. KOBAYASHI, and A. H. TEWFIK. Multimedia data-embedding and watermarking technologies. *PROCEEDINGS OF THE IEEE*, 86(6), 1998.

[9] A.-N. Yahya, H. A. Jalab, A. Wahid, and R. M. Noor. Robust watermarking algorithm for digital images using discrete wavelet and probabilistic neural network. *Journal of King Saud University-Computer and Information Sciences*, 27(4):393–401, 2015.

[10] M. M. Yeun and F. Mintzer. An invisible watermarking technique for image verification. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, pages 680–683. IEEE, 1997.