Wrocław University
of Science and Technology

Faculty Of Electronics

Computer Science

Internet Engineering

Multimedia and Computer Visualization

Digital Watermarking

Authors:
Rafał Grądkowski
Bartosz Gardziejewski
Paweł Zaborowski
Łukasz Obrębski

Lecturer: dr inż. Marek Woda
Group: E05-17b

Spis treści

1 Subject

The project's aim is to implement and compare two digital watermarking methods when applied to images. Chosen technologies are Least Significant Bit [LSB] Technique and Patchwork Algorithm. Analysis will be focused on main areas of watermarking which are: transformation, distortion, compression...

//TODO: dopisac jakie jeszcze.

2 State of the art

Digital watermarking in general is a kind of marker embedded in a noise-tolerant signal. The most popular kinds of the signals which allow watermarking are images, videos and audio signals. Digital watermarking techniques are typically used for identifying ownership and mark copyrights. Most recognizable form of digital watermarking in images is a simple overlay of a text or logo image over the original image. An example of such watermark can be seen below:

However, in professional applications, digital watermarks should be perceptible under certain conditions only, e.g. after using some specific algorithm. If the carrier signal is distorted by a digital watermark in such a way

that it becomes easily noticable or the size of the carrier signal is changed, it could be considered not effective (depending on its purpose.

There are currently three types of digital watermarking methods:

1. Spatial domain techniques - directly add the watermark to pixel values - exploit Human Visual System for hiding the data
2. Transformed domain techniques - add the watermark to the coefficients of a full-frame transform (DFT, DCT, Mellin, Radon, Fresnell)
3. Hybrid techniques

3 Project's scope

A concise project's description was done in the following subchapters, consisting of information about project's goal, requirements and implemented features.

3.1 Goal of the project

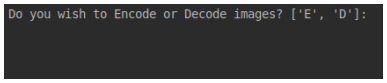
The main goal of the project is to prepare simple application with both of the algorithms implemented. Our tasks were focused on proper algorithm implementation rather than on user interface. We were trying to implement both algorithms properly and to compare parameters between them. Also our aim was to compare how do they work in practice.

3.2 Requirements

The user need to have possibility to encode their picture by watermark, optionally, chosen by them. The user need to have possibility to choose which algorithm they would like to use.

3.3 Features

Application is Command Line Interface based. Firstly it allows user to choose between encoding picture, or with usage of patchwork, decoding, shown on the figure a). After picking encoding we have two algorithms to choose (figure b)).



(a) figure a)

(b) figure b)

3.3.1 Encoding

- LSB Application requests either for picture to be watermarked and the one to be used as watermark.

```
Please specify algorithm to use to watermark the image ['LSB', 'PAT']: LSB
Please specify image to be used as watermark for ../resources/bright.png (press Enter to use random image
Encoded file: ../resources/bright.png
```

- Patchwork

  In opposition to previous option there application needs to provide encoding key.

```
Please specify algorithm to use to watermark the image ['LSB', 'PAT']: PAT
Please specify the secret key [0 .. 4,294,967,295] to watermark the ../resources/high_res.png with:
```

3.3.2 Decoding

```
Please specify the secret key [0 .. 4,294,967,295] to watermark the ../results/bright.png with: 3
There's 2.272204840928105% probability that the image is encoded using key: 3
```

4 Techniques and Technologies

4.1 Tools

- Python 3.7 with packages:
  1. NumPy
  2. scikit-image
- Algorithms:
  1. LSB – Least Significant Bit
  2. Patchwork techniqe

4.2 Communication between team members

Communication of team members took place remotely using Trello (an internet application for project management) and Facebook Messenger (messenger on a social networking site). After the division of tasks through Trello, each member could take on a specific task and present the progress of the work and the final effects of the project stages on an ongoing basis. Important point of communication were team meetings, where we could discuss problems related to the project implementation and solve difficult tasks more efficient. The source code of the application has been managed remotely using the GitHub repository.

5 Milestone and project plan

5.1 Team members

- Bartosz Gardziejewski
- Rafał Gradkowski
- Łukasz Obrebski
- Paweł Zaborowski

5.2 Project plan

Out team have splitted into topic groups:

1. LSB implementation
2. Patchwork implementation
3. Documentation work

5.3 Gantt Chart

- 11.10.19 – Setup of Trello & GitHub
- 17.10.19 – Project Kick OFF (first presentation)
- 14.11.19 – Literature analysis, working Python environment, beginnings of a documentation

- 28.11.19 – Implementation of embedding digital watermark in images using both algorithms
- 12.12.19 – Implementation of decoding digital watermark in images using both algorithms
- 09.01.20 – Juxtaposition and comparison of results
- 16.01.20 – Project closure (Final presentation, project demo, documentation)

6 Results

6.1 LSB

6.1.1 Implementation

In our program the LSB algorithm is going through the image colour by colour and pixel by pixel and checks the most significant bit of the watermark pixel, if it is 1 then it sets 1 in lest significant bit of the pixel in image, if it is 0 then it is set to 0. If the water mark is smaller then image it is looped.

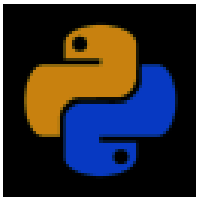
6.1.2 Testing

We performed test using our LSB algorithm on two pictures. The results was as follows. Firs picture was python logo with resolution 225x225 it take approximately 1 second to watermark that image and the second was photo with resolution 393x206 it take approximately 1.5 second:

(a)

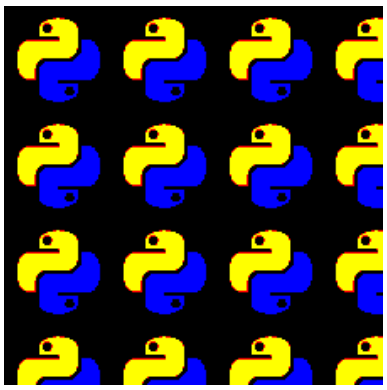
(b)

As watermark we used small python logo with resolution 64x64 and color inversion:



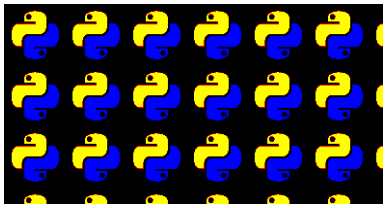
Watermarked image and the water mark extracted from it looked as expected as on figure:

(a)



(b)

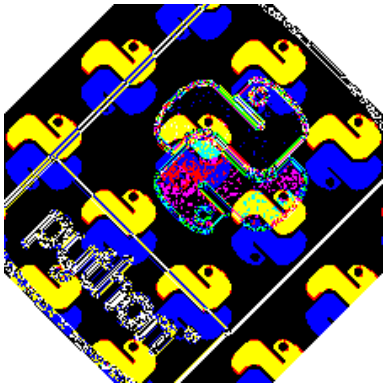
(a)



(b)

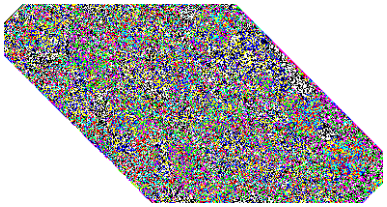
Next step was to transform the watermarked image and check if watermark is extractable from it. To perform transformation we used program GIMP2.0. First we try rotating image by 45 degrees, figure

(a)



(b)

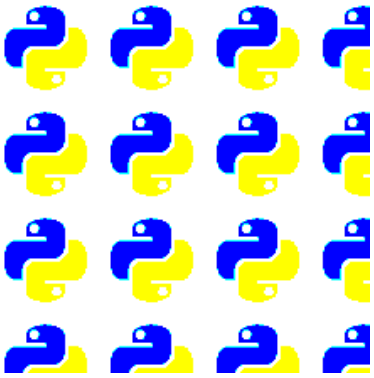
(a)



(b)

Expected outcome of the rotation would be rotated wtermark, but in this case we could see distortion around edges in *python* logo and absolutely unreadable watermark in *photo*. It turns out that some programs have more complex rotation algorithm to maker rotated image look better. Next we try color inversion:

(a)



(b)

In this case as expected watermark was color inverted only. Next we try manipulate with color, and change contrast:

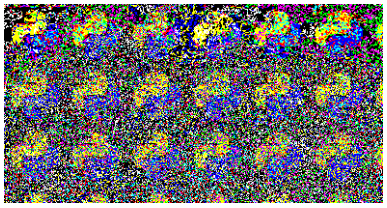
(a)

(b)

It Turns out that even slightly changed contrast makes watermark almost unreadable. Last transformation was more advanced one, performed on *photo*.

Noise reduction:

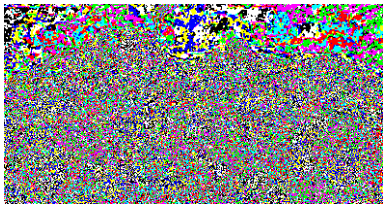
(a)



(b)

Gaussian blur:

(a)



(b)

They both almost completely destroyed watermark.

6.2 Patchwork algorithm

6.2.1 Implementation

Patchwork algorithm variation used in the project, was implemented as follows:

- Convert image color model to YUV.
- Randomly (based on certain key) select N pairs of pixels.
- Increase first pixel's Y channel value by a constant value δ. Decrease second pixel's Y channel value analogically.
- Convert image color model back to original.

Now, to detect such watermark, key must be known to obtain the same pairs of pixels during image decoding. Then, pixels pairs are 'visited' in the same order, to calculate control sum as follows:

$$\sum_{i=0}^{N} A_i - B_i \tag{1}$$

where A_i and B_i are first and second pixel's Y channel values respectively.

It is expected that for unmodified image control sum value will be close to 0, as the number of times A_i is greater than B_i should be offset by the number of times the reverse is true.

Unfortunately, this assumption is not always true, which is the main disadvantage of currect implementation, as can be seen in the next section.

6.2.2 Testing

Patchwork algorithm should be resistant primarily to color changes. There are also implementations that can be resistant to cropping or more advanced tones adjustments but our implementation does not support that.

Algorithm was tested on various images, two of which are presented below:

Both images were watermarked and subjected to distinct changes: brightness, contrast, color temperature and saturation increase. Key used to generate pixel pairs was: 12345.

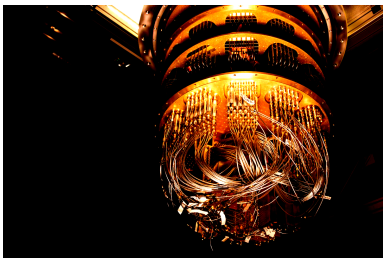
(a)

(b)

(a)

(b)

(a)



(b)

(a)

(b)

(a)

(b)

7 Conclusions

7.1 LSB

LSB is fairly simple and quite fast algorithm, unfortunately it is not very robust - simple transformation can make it "disappear". We could use for example not the least significant bit but middle one that should increase robustness, although it makes watermark much more visible:

LSB pros and cons:

- Resistance to geometric transformations, like cropping, stretching or rotating, in condition that the operation is not using more sophisticated algorithm
- High capacity of watermark
- fast and easy to implement
- Can be easily destroyed by distortions or gamma changes

7.2 Patchwork

Patchwork pros and cons

- Resistance to cropping and to gamma and tone scale corrections

- The detector doesn't require the original cover image to determine whether the image has been watermarked.
- Can be destroyed by any affine transformation, like translation, rotation or scaling