Erasing watermarks on images using blind sources separation

BATY Léo, BRUNOD-INDRIOGO Luca, EDDINE NAJJAR Chiheb, LAURET Jérémy

Introduction

BLABLA INTRODUCTIF

We chose to use our theoretical results on blind source separation in order to erase watermarks. In each case, we assumed that we had a set of three images composed of a target image with a watermark to erase and two copies of another image, one with watermark and the other without.

Fig. 1: Examples of watermarked images[1]

I - Theory and Mathematical tools

The observed information is contained in a random vector $\vec{x}$, it's the image of an unknown random vector $\vec{s}$ by an unknown linear transformation A, i.e. $\vec{x} = A\vec{s}$. The question is, can we find $\vec{s}$ using only $\vec{x}$?

According to CAMON's therom, if the problem satisfies the following conditions :

- The components of $\vec{s}$ are independent
- At most, one of $\vec{s}$ components is gaussian
- A is an invertible matrix, and time independent

Then, if we find a matrix B such that the vector $B\vec{x}$ has independent components, we must have $B\vec{x} \simeq \vec{s}$ to a permutation and a homothetie of the components.
To characterize the independence of the components of a vector, we can use **Mutual penalized Information :**

$$J\left(\vec{y}\right) = \int_{\vec{t}} p_{\vec{y}}\left(\vec{t}\right) \ln\left(\frac{p_{\vec{y}}\left(\vec{t}\right)}{\Pi_i p_{y_i}(t_i)}\right) \mathrm{d}\vec{t} + \lambda \sum_i \left(\mathrm{Var}(y_i) - 1\right)^2 \tag{1}$$

where $\vec{y} = (y_i)_i$ is a random vector, and λ is the penalty setting

J satisfies the following properties :

- $J \geq 0$
- $J\left(\vec{y}\right) = 0 \Leftrightarrow \vec{y}$ has independent components, and $\mathrm{Var}(y_i) = 1 \forall i$

Now we can reformulate the problem as follows : find B such that $J(B\vec{x}) = 0$ which is equivalent to find $\min_B J(B\vec{x})$.

J does not only help us find a vector with independent components from $\vec{x}$ but also a bounded one.

To minimize J over B we use the gradient decent algorithm : $B_{n+1} = B_n - \mu \frac{\partial J}{\partial B}(B_n)$. Where μ is the step parameter.
We are going to impose the fact that $\mathbb{E}[y_i] = 0$ in our algorithm.

II - Case of a linear watermark

At first, we used our algorithm in a case that matched the theoretical framework, that is to say a watermark obtained by linear combination of a text on plain background with the original image:

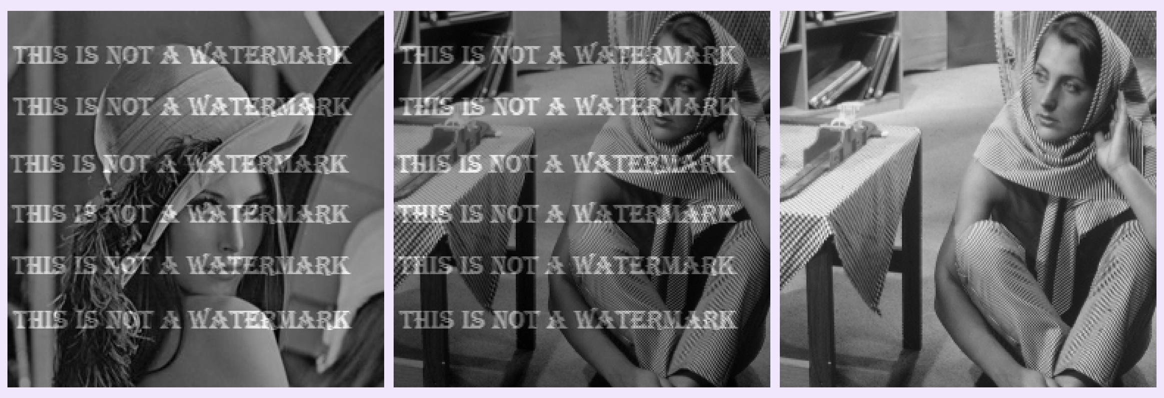

Fig. 2: Input of two images with a linear watermark and one without

As expected, the algorithm returns both original images along with the image of the watermark:

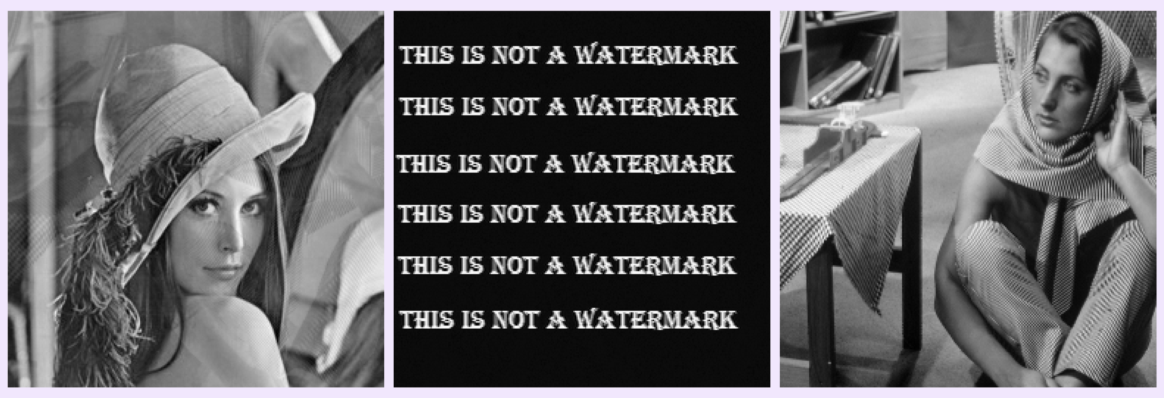

Fig. 3: Output of the program

III - Case of a common (non-linear) watermark

We generalized the code to colored images working separately on each of the three RGB colors. The case of a linear watermark is very specific and most of the commonly used watermarks are not that simple. That is why we also tried to remove a watermark made by a specialized website[2].

Fig. 4: Input of two images with a non-linear watermark and one without

As the modification of the images we obtained that way were not necessarily linear, we couldn't expect to erase the watermark only by using blind source separation. The results it gives are interesting, though.

Fig. 5: Result of the program

The watermark barely changed and is still visible, however, the algorithm manages to isolate the letters of the watermark on a plain background.

Thus, it becomes rather easy to identify the pixels of the images that correspond to the letters. This allows to determine the kind of transformation that is applied by comparing the pixels of the image we have with and without watermark. Obviously, an important drawback of this method is that it is not systematic and is possibly inconclusive when the transformation is too complicated. Nevertheless, the results were very satisfying in our case where transformations turned out to be affine.
BLABLA DE CONCLUSION

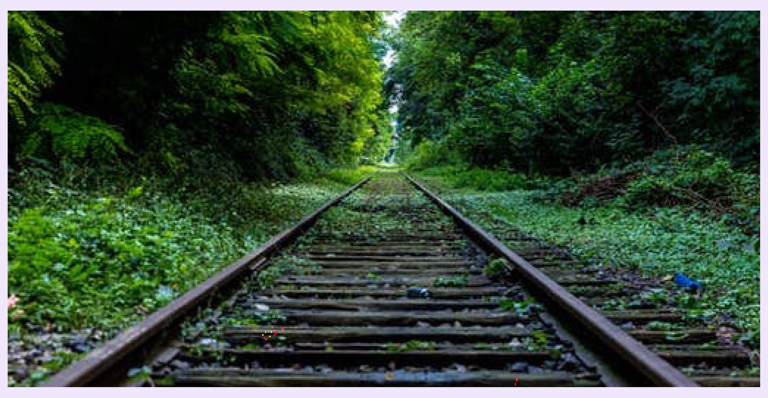

Fig. 6: Final result

[1] site des watermarks
[2] site des watermarks
[3] Code Python: https://github.com/JeremyLauret/watermark-remover