IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2020–05–21.
Submitted on 2019–10–16, accepted on 2020–04–23.
ISSN 2105–1232 © 2020 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2020.283

Local JPEG Grid Detector via Blocking Artifacts, a Forgery Detection Tool

Tina Nikoukhah, Miguel Colom, Jean-Michel Morel, Rafael Grompone von Gioi

Université Paris-Saclay, ENS Paris-Saclay, CNRS, Centre Borelli, F-94235, Cachan, France
{nikoukhah, colom, morel, grompone}@cmla.ens-cachan.fr

Communicated by Miguel Colom *Demo edited by* Tina Nikoukhah

Abstract

Image JPEG compression leaves blocking artifact traces. This paper describes an algorithm that exploits those traces to locally recover the grid embedded in the image by the JPEG compression. The algorithm returns a list of grids associated with different parts of the image. The method uses Chen and Hsu's cross-difference to reveal the artifacts. Then, an a contrario validation step according to Desolneux, Moisan and Morel's theory delivers for each detected grid a Number of False Alarms (NFA) which tells how unlikely it is that the detection is due to chance. The only parameter is the step size of the windows used, which represents the exhaustiveness of the method. The application to image forgery detection is twofold: first, the presence of discrepant JPEG grids with low NFA is a strong forgery cue; second, knowledge of the grid is anyway required for further JPEG forensic analysis.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. Compilation and usage instruction are included in the `README.txt` file of the archive.

Keywords: JPEG compression; blocking artifact analysis; a contrario method; forgery detection

1 Introduction

The JPEG format is currently the most common method for compression of digital photography. The encoding process consists of the following steps:

1. The RGB (red, green and blue) color channels are converted to YCbCr (luminance and two chroma components).

[1] https://doi.org/10.5201/ipol.2020.283

2. The chroma channels Cb and Cr are subsampled. The sampling ratios depend on the parameters used in the compression method.
3. Each of the three image channels is partitioned into 8×8 non-overlapping blocks.
4. The type II 2D Discrete Cosine Transform (DCT) is then applied to each block.
5. The DCT coefficients of each block are quantized according to a given table.
6. The resulting 8×8 blocks are losslessly encoded by using run length and Huffman coding.

The quantization of DCT coefficients (lossy compression) leaves traces at the boundaries of each 8×8 block, as shown in Figure 1. These traces, characteristic of JPEG compression, can be used to retrieve the grid, depicted in red in the figure. Since the blocks are of size 8×8, there are 64 possible grid origins. In the following, a grid will be characterized by its origin's coordinates g_x and g_y. If the JPEG image has not been further processed, the grid's origin should be $(0, 0)$.

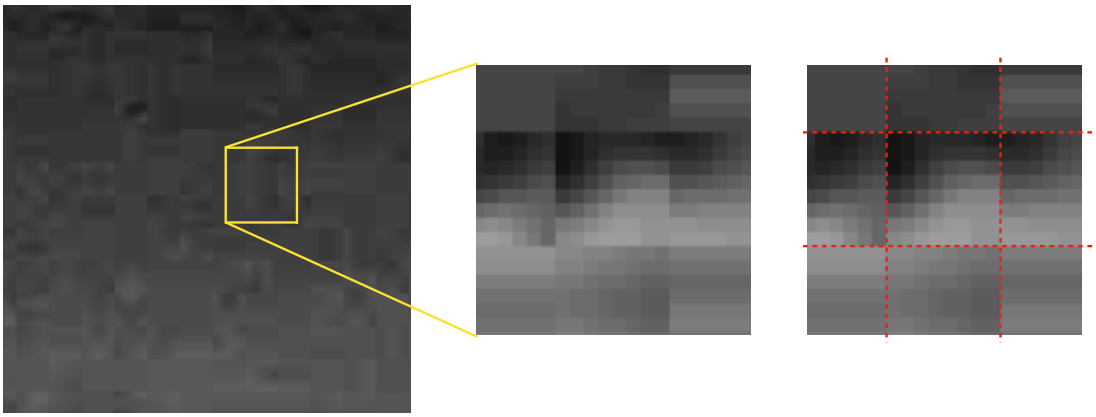


Figure 1: JPEG block artifacts. The red dotted lines highlight the boundaries of the 8×8 blocks used in the compression.

Our method analyses the blocking artifacts *locally* in several image windows, and aims to determine whether a JPEG grid is observed in each one. The algorithm is composed of three main steps: First, a cross-difference filter [1] is applied to the luminance channel of the image to emphasize the JPEG traces. Then, a family of overlapping windows is created as illustrated in Figure 2. In each window, the horizontal and vertical local maxima of the cross-difference vote for the JPEG grid origin that would imply that a block boundary passed through them. Finally, the votes go through a statistical validation step based on the *a contrario* theory [2]. The result of the algorithm is the list of all the windows and their vote: a meaningful grid or no detection. The presence of two or more different JPEG grid origins may be a cue for image forgery. On the other hand, when a single and coherent JPEG grid origin is found all over the image, further JPEG analysis can be performed to authenticate the image.

The only parameter of this method is the minimum window step ($W \times W$ pixels, W must be a multiple of 8 as we will see later). The smaller W, the more exhaustive the method; nevertheless, the exhaustiveness implies the cost of a longer computational time.

The algorithm described here derives from the one in [7] but includes some improvements.

2 Algorithm

Algorithm 1 provides a pseudo-code of the full method. Each step of this algorithm is described in the following subsections.

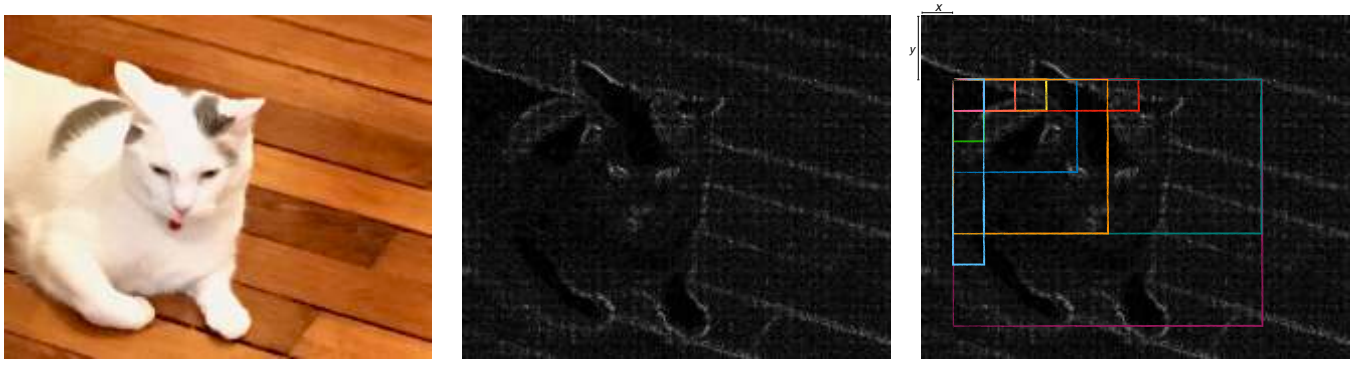


Image | Filtered image | Sample of windows starting at coordinate (x, y)

Figure 2: Sample of windows, with $x = W$ and $y = 2W$.

2.1 Luminance Component

The algorithm takes an RGB image and computes (Algorithm 1 step 1) its luminance according to the JPEG standard

$$I = 0.299\ R + 0.587\ G + 0.114\ B,$$

where R, G and B are the values of the red, green and blue channels at a given pixel. The chroma components, Cb and Cr, are not used in the proposed method. These components are usually sub-sampled in JPEG images. The sub-sampling ratios for rows and columns may be different and they vary from image to image. When these ratios are known, the proposed approach could be adapted to use, additionally, the chroma components.

2.2 Grid Extraction

The blocking artifacts appear as luminance changes along the block frontiers. Several filters were proposed in the literature to emphasize the blocking artifacts.

Let I be the $X \times Y$ luminance component of the input image and $I(x, y)$ the intensity value at pixel (x, y), with $0 \leq x \leq X - 1$ and $0 \leq y \leq Y - 1$. The simplest method [6] to reveal the presence of block artifacts computes the absolute value of the gradient magnitude image. This first order derivative is approximated by two difference filters,

- horizontally:
$$|I_x(x, y)| \approx |I(x, y) - I(x - 1, y)|; \tag{1}$$
- and vertically:
$$|I_y(x, y)| \approx |I(x, y) - I(x, y - 1)|. \tag{2}$$

Other authors [5] use the absolute value of second order derivatives approximated by

- horizontally:
$$|I_{xx}(x, y)| \approx |2I(x, y) - I(x + 1, y) - I(x - 1, y)|; \tag{3}$$

Algorithm 1: Local JPEG grid detector via blocking artifacts

input : A color image (R, G, B) of size $X \times Y$
input : Window step size W
output: A list L of windows with detected JPEG grid

$I \leftarrow 0.299\,R + 0.587\,G + 0.114\,B$ // compute luminance image
$C \leftarrow \left| I \star \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right|$ // compute cross-difference
foreach $\omega \in \Omega(W)$ **do** // loop on the family of local windows, Section 2.3
  $\text{vote}_x[\cdot] \leftarrow 0$ // initialize votes to zero
  $\text{vote}_y[\cdot] \leftarrow 0$
  foreach $(x, y) \in \omega$ **do**
    if $C(x, y) > C(x-1, y)$ **and** $C(x, y) > C(x+1, y)$ **then** // local horiz. maximum
      increment $\text{vote}_x[x \bmod 8]$
    if $C(x, y) > C(x, y-1)$ **and** $C(x, y) > C(x, y+1)$ **then** // local vert. maximum
      increment $\text{vote}_y[y \bmod 8]$
  $n_x \leftarrow \text{sum}(\text{vote}_x)$ // total number of horizontal votes
  $k_x \leftarrow \max(\text{vote}_x)$ // votes for best horizontal origin
  $\text{NFA}_x \leftarrow \frac{(XY)^2}{1024}\mathcal{B}\left(\frac{|\omega|}{16}, \frac{k_x}{2}, \frac{n_x}{|\omega|}\right)$ // compute horizontal NFA, Section 2.4
  $n_y \leftarrow \text{sum}(\text{vote}_y)$ // total number of vertical votes
  $k_y \leftarrow \max(\text{vote}_y)$ // votes for best vertical origin
  $\text{NFA}_y \leftarrow \frac{(XY)^2}{1024}\mathcal{B}\left(\frac{|\omega|}{16}, \frac{k_y}{2}, \frac{n_y}{|\omega|}\right)$ // compute vertical NFA, section 2.4
  if $\text{NFA}_x < 1$ **and** $\text{NFA}_y < 1$ **then** // meaningful JPEG grid found
    $g_x \leftarrow \arg\max(\text{vote}_x)$
    $g_y \leftarrow \arg\max(\text{vote}_y)$
    append $(\omega, g_x, g_y, \text{NFA}_x, \text{NFA}_y)$ **to** L

- and vertically:

$$|I_{yy}(x,y)| \approx |2I(x,y) - I(x,y+1) - I(x,y-1)|. \tag{4}$$

Yet, as can be seen in Figure 3, both filters have a strong response to the edges and textures present in the image and may induce aberrant grid detection. To reduce the interference of the background scene details, a cross-difference filter proposed in [1] is defined by

$$C(x,y) = |I(x,y) + I(x+1,y+1) - I(x+1,y) - I(x,y+1)|. \tag{5}$$

This filter amounts to the absolute value of a convolution of the image with a 2×2 kernel as given in step 2 of Algorithm 1. To avoid setting boundary conditions, the JPEG grid detection will work only in the region where the cross-difference is well defined: everywhere except the last row and column of the image.

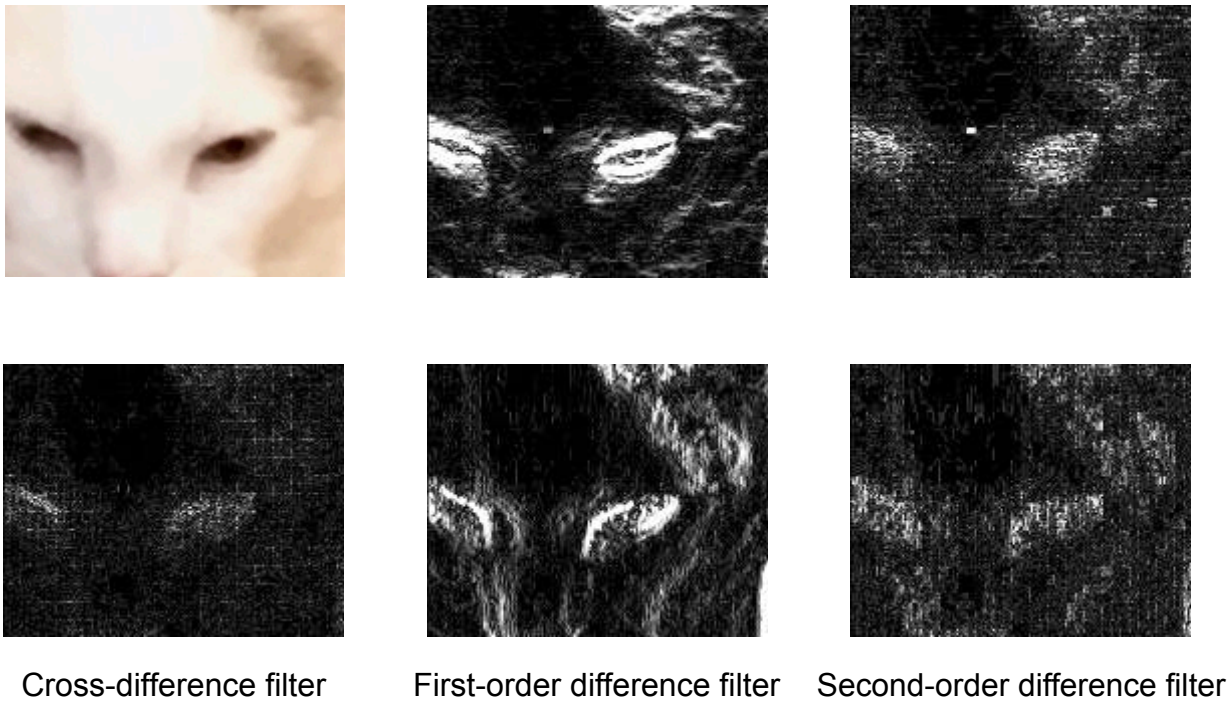


Figure 3: Close view of the cross-difference, first-order derivative and second-order derivative.

Our method uses this filter to reveal the compression artifacts. However, in cases where the image has been weakly compressed, even this filter can be inefficient. The JPEG format has a quality parameter Q measuring the compression quality in a scale ranging from 1 to 100. The higher the compression quality Q, the less the image is compressed and the dimmer the JPEG grid. The image in Figure 3 comes from a smartphone camera which compresses at quality 93 and the images of Figure 4 have been compressed with **imagemagick** with varying quality factors.

2.3 Voting Process

The JPEG grid is evaluated locally, so a counting process is performed independently in each window, see Algorithm 1 step 3. Figure 2 illustrates the family of windows $\Omega(W)$ we use. In principle, any pixel of the image can be used for the upper-left corner and any pixel can be used for the lower-right corner of a window. But to simplify the comparison between different grid origins, we will restrict the windows to sizes multiple of 8; in this way, any of the eight horizontal or vertical grid origins are equally represented in each window.

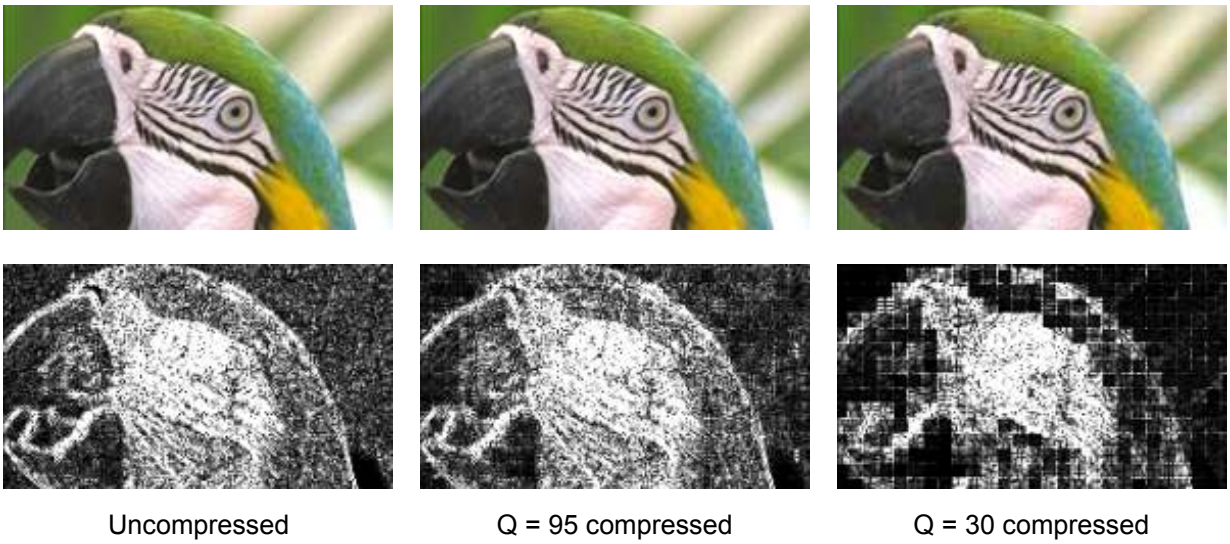


Figure 4: Comparison of cross-difference images for different JPEG compression quality factors.

However, to accelerate the computation, only a subset of these rectangular windows is computed: the number depends on how local we want the method to be, with W being the minimum window side size such that pixels of the image with coordinates multiple of $\left\lfloor \frac{X}{W} \right\rfloor$ and $\left\lfloor \frac{Y}{W} \right\rfloor$ are used as the corners of the family of rectangular windows.

Thus the total number of windows with sizes that are multiples of W is

$$|\Omega(W)| = \frac{1}{4} \left\lfloor \frac{X}{W} \right\rfloor \left(\left\lfloor \frac{X}{W} \right\rfloor + 1 \right) \left\lfloor \frac{Y}{W} \right\rfloor \left(\left\lfloor \frac{Y}{W} \right\rfloor + 1 \right). \tag{6}$$

When a JPEG grid is present, the local maxima of the cross-difference tend to concentrate on JPEG block frontiers as shown in Figure 3. Therefore each horizontal or vertical local maximum of the cross-difference $C(x, y)$ votes for the grid origins compatible with such block frontiers, namely $x \bmod 8$ and $y \bmod 8$, see Algorithm 1 steps 6 to 10. Thus, each local maximum votes for origin g_x or g_y with values from 0 to 7.

To work in an area where the cross-difference and the computation of the local maxima are both defined, the family of local windows $\Omega(W)$ is set with $1 \leq x \leq X - 2$ and $1 \leq y \leq Y - 2$.

Working with a reduced family of windows is just the result of a practical consideration, to obtain a faster algorithm. With an adequately chosen value for W, the algorithm will give a good balance between producing a result similar to the exhaustive search ($W = 8$) while significantly reducing the computational time.

2.4 Validation Step

The validation step is based on the *non-accidentalness* principle which prescribes to reject detections that could be the result of an accidental configuration. Accordingly, the *a contrario* approach introduced by Desolneux, Moisan, and Morel [2] proposes to control the expected number of false detections on a noise or *a contrario* model H_0 where the desired structure could only be present by chance.

Our *a contrario* assumption is the absence of a JPEG grid. Under that assumption the local maxima votes should be *ceteris partibus* uniformly distributed between 0 and 7. A detection will be considered when the number of votes for a particular position is too large to be the result of chance.

The mathematical setting corresponds to a multiple testing procedure to control the expected number of false detections under the null model H_0 [3]. The Number of False Alarms (NFA) of observing a value e is defined by

$$\text{NFA} = N_{test}\, P_{H_0}(E \geq e) \tag{7}$$

where N_{test} is the number of events tested and $P_{H_0}(E \geq e)$ is the probability of observing a value as large as e for a random variable E under the stochastic model H_0. The event is called ϵ-meaningful if and only if NFA $< \epsilon$. The question to be answered is whether a window's vote for a coordinate (g_x, g_y) is meaningful or not. Each window has two events to test: the horizontal g_x and the vertical g_y grid origin coordinate. A window will be called meaningful under the *a contrario* assumption when both of these events are ϵ-meaningful, i.e. $\text{NFA}_x < \epsilon$ and $\text{NFA}_y < \epsilon$.

Let us denote by n_x and n_y the total number of horizontal and vertical votes (see Algorithm 1 steps 11 and 14). We will denote by k_x and k_y the number of votes for the most voted grid, horizontal and vertical respectively (Algorithm 1 steps 12 and 15). Because it was imposed that the window size is a multiple of eight, each of the possible grid origins has the same number of potential votes, and directly comparing the number of votes is fair. We need now to determine whether k_x and k_y are too large to be the result of chance, which implies that too many of the positions compatible with a given grid voted for it. However, evaluating directly the probability of observing k_x or more votes for a given origin is difficult because the votes are not independent. Indeed, the computation of a local maximum requires comparing three consecutive values of the cross-difference, and each of the latter is computed using a 2×2 set of image pixels. Thus, a horizontal local maximum involves a 4×2 set of image pixels. As a consequence, the votes on a given column are not independent. Nevertheless, the votes would be independent if we counted only rows at distance two. Ideally, we should perform two tests, one with even rows and another one with odd rows. A simpler way is to count all the rows and then divide by two. If the votes were equally distributed on the rows, then this count would give the same value as any of the sub-counts. If not, necessarily one of the two sub-counts would have more votes. So $k_x/2$ is a conservative count of the number of independent votes.

Among the $|\omega|$ pixels in the window ω, only $|\omega|/8$ are potential maxima associated to each of the 8 horizontal grid origins. Using the same reasoning as before, only half of them can be considered independent. That means that among the $|\omega|/16$ positions that could have voted for a given horizontal grid position, only $k_x/2$ actually did. Our *a contrario* random model H_0 is that each of these votes are independent Bernoulli random variables. The probability of voting for these random variables is unknown *a priori*; the proposed algorithm makes an empirical estimation given by $\frac{n_x}{|\omega|}$, that is the total number of votes in the window over the total number of pixels in the window. Then, the probability of observing as many votes just by chance is thus

$$\mathcal{B}\left(\frac{|\omega|}{16}, \frac{k_x}{2}, \frac{n_x}{|\omega|}\right),$$

where $\mathcal{B}(\eta, \kappa, \rho)$ is the binomial tail given by

$$\mathcal{B}(\eta, \kappa, \rho) = \sum_{j=\kappa}^{\eta} \binom{\eta}{j} \rho^j (1-\rho)^{\eta - j}. \tag{8}$$

The number of tests N_{test} corresponds to the total number of windows in the image, times the number of different horizontal grid origins, times 2 to count the two tests theoretically performed, one for odd rows and one for even rows. Here, we will not take into consideration the reduction of the family of windows described in the last section; indeed, the only purpose of such reduction is reducing the computational time, but we want the result for a given window to be the same as if the exhaustive search were performed. Thus, $N_{test} = 2 \times 8 \times |\Omega(8)| \approx 16\, \frac{1}{4} \left(\frac{X}{8} \frac{Y}{8}\right)^2 = \frac{(XY)^2}{1024}$.

Following the *a contrario* theory, we define the Number of False Alarms (NFA) as

$$\mathrm{NFA}_x = \frac{(XY)^2}{1024}\mathcal{B}\left(\frac{|\omega|}{16}, \frac{k_x}{2}, \frac{n_x}{|\omega|}\right). \tag{9}$$

An analogous reasoning is performed for the vertical grid origin. The evaluation of the horizontal and vertical grid origins are different tests and even different families of test; therefore, there is no problem of independence between the horizontal and vertical test for the same window. Similarly to the horizontal case, the NFA for the second test is

$$\mathrm{NFA}_y = \frac{(XY)^2}{1024}\mathcal{B}\left(\frac{|\omega|}{16}, \frac{k_y}{2}, \frac{n_y}{|\omega|}\right). \tag{10}$$

All in all, a JPEG grid is detected when $\mathrm{NFA}_x < \epsilon$ and $\mathrm{NFA}_y < \epsilon$, see Algorithm 1 step 17.

Desolneux et al. [2] suggested using $\epsilon = 1$ which implies getting, on average, less than one false detection per image. This makes sense when many detections are expected per image. An example of this is the detection of line segments in an image [4], where hundreds or thousands of them are present in a typical image; accepting less than one false detection seems thus reasonable. In our current problem, however, it may seem contradictory to set $\epsilon = 1$ as in a normal JPEG image we expect to find just *one* grid; accepting one false grid detection in H_0 would imply getting, on average, a spurious grid detection on every image, even when no JPEG compression is present. Nevertheless, the proposed *a contrario* formulation treats the horizontal and vertical grid evaluations as different families of tests, and the number of false detections is controlled so as to get no more than ϵ false horizontal origin detections and no more than ϵ false vertical origin detections. The JPEG grid origin detection requires both tests to be satisfied, so we know the expected number of JPEG grid origin detections is also controlled by ϵ. Actually, its expected value is much lower. Indeed, when setting $\epsilon = 1$ one expects to obtain, under the null model H_0, one test among a total of $\frac{(XY)^2}{1024}$ *horizontal* tests to be positive; it is also expected that one test among a total of $\frac{(XY)^2}{1024}$ *vertical* tests to be positive. But to obtain a false JPEG grid origin detection, both tests must correspond to the same window. There is no reason why for spurious horizontal and vertical tests to be satisfied on the same window. As a result, we may expect to observe accidental horizontal and vertical detections on the same window in about one out of $\Omega(8) \approx \frac{(XY)^2}{16384}$ random images of size $X \times Y$; this is again reasonable. Even for images as small as 100×100, this corresponds to one false detection every 6103 images. (A tighter selection of ϵ would be possible, but it would require the user to select the acceptable false alarm rate.) All in all, we set $\epsilon = 1$ as this simple criterion results in an effective control of the number of false JPEG grid origin detection without the need for further user intervention.

Concerning the numerical implementation, two comments are relevant. First, in our implementation, the computation of the binomial tail is performed using the following relation to the Gamma function,

$$\binom{n}{k} = \frac{\Gamma(n+1)}{\Gamma(k+1)\cdot\Gamma(n-k+1)},$$

for which there are effective implementations readily available. To speed up the computations, the sum of the binomial tail is truncated when the error can be bounded to be less than 10%.

Second, the NFA may reach very small values, which may underflow the usual IEEE 754 number representation. Our implementation in the C programming language, which uses IEEE 754 number representation, computes $\log_{10}(\mathrm{NFA})$ instead of NFA, allowing for a larger numeric range. Any logarithm base is equally useful for this purpose; the 10 base makes it slightly easier to read the order of magnitude of the NFA values. Of course, the test must now compare $\log_{10}(\mathrm{NFA})$ to $\log_{10}(\epsilon)$, which for $\epsilon = 1$ is zero.

2.5 Parameter Choice and Computational Complexity

The algorithm's only parameter is the size of the smallest window $W \times W$. The smaller W, the more local the method, and the longer the computation. The code can be executed in parallel, the computation at each window is independent from the others. It is reasonable to use values $W \geq 64$ so that each window has at least 8 repetitions of the JPEG 8×8 blocking artifact. We observed that smaller values increase the computation time while rarely adding meaningful detections.

In a nutshell, the algorithm first computes the cross-difference of the whole image; then, for each window the votes are counted, and finally a NFA value is computed. The number of operations required for computing the cross-difference is proportional to the number of pixels in the image. The bottleneck is the second step, which is proportional to the product of the number of windows analyzed and the size of each window. From Equation (6) we know that the number of windows is bounded by

$$|\Omega(W)| \leq \frac{(XY)^2}{W^4}.$$

Two operations are performed per window: counting the number of votes and selecting the maxima per axis for the computation of the NFA values. Only the vote count is relevant for the present calculation as the other computations require a constant number of operations per window. The complexity, thus, is determined by the number of votes which requires as many operations as the size of the window. This value is bounded by the largest window, the one covering the full image. Thus, the complexity of the algorithm is

$$\text{computational complexity} = O\left(\frac{(XY)^3}{W^4}\right). \tag{11}$$

In other words, the complexity is proportional to the cube of the number of pixels in the image. Also, the larger the smallest window W, the faster the method. The speed-up comes at the cost of a reduced spatial resolution, reducing the capacity to detect small forgeries.

To give an idea of the usefulness of performing a non-exhaustive search, Equation (11) shows that doubling W reduces the computational time by a 16 factor. Then, relative to the exhaustive search ($W = 8$), the speed-up obtained are about 16 for $W = 16$, 256 for $W = 32$, 4096 for $W = 64$, and so on. Thus, the value of W determines in practice the analysis time and may result in an exhaustive but slow, or in a very fast process.

3 Experiments

The algorithm should detect the "strongest" grid, namely the one with the heaviest compression, in case of several successive compressions. Indeed, most post-processed or tampered images have been compressed at least twice, once when acquired and once after processing. In the following we analyze the results of our detection algorithm for several meaningful applications.

3.1 JPEG Compression Detection

A first simple application is to tell if an image has undergone JPEG compression or not. If the image has undergone a lossy compression, a global grid is detected and so the coordinates to its origin are returned, otherwise there is no detection. The method does not need to be exhaustive and can therefore look at big windows in the image, therefore performing few tests and being very fast.

Figure 5: Uncompressed image and JPEG compressed image at quality 90.

Let us consider the examples in Figure 5. It consists of two copies of the same 768×512 images, one uncompressed and the other compressed with JPEG at quality 90. The proposed algorithm, with $W = 64$ gives the following result for the uncompressed version:

```
image size: 768 x 512
window step size: 64
total number of evaluated windows: 1848
number of meaningful windows: 0 (0 %)
number of meaningful windows for each JPEG grid origin:
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
best log(NFA) for each JPEG grid origin:
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
number of meaningful JPEG grids found: 0
no meaningful grid found

No suspicious traces found in the image with the performed analysis.
```

Again, the proposed algorithm, with $W = 64$ gives the following result for the JPEG version of the image:

```
image size: 768 x 512
window step size: 64
total number of evaluated windows: 1848
number of meaningful windows: 1736 (93 %)
number of meaningful windows for each JPEG grid origin:
    1736        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
best log(NFA) for each JPEG grid origin:
  -637.8        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
number of meaningful JPEG grids found: 1
most meaningful JPEG grid origin (0,0) with NFA: 10^-637.838

No suspicious traces found in the image with the performed analysis.
```

This second table represents the number of votes per coordinate. Here, 1736 windows voted meaningfully for the origin $(0, 0)$ out of the 1848 windows which did vote. In the online demo, to each block's vote is associated a NFA. Here, the most meaningful NFA is printed.

We can illustrate with this example the impact of the parameter W on the processing time:

W	time (s)
64	1.54
32	21.07
16	391.98

These values are in general agreement with Equation (11).

3.2 Crop Detection

In Figure 6, we took an original JPEG image and cropped a square out of it. Algorithm 1 was tested on the cropped image with $W = 64$, which led to testing 3025 windows. Of these windows, 2883 detected the grid $(4, 4)$ with overwhelming significance. The origin of the global grid being different from $(0, 0)$, the (anticipated) conclusion is that the image has been cropped.

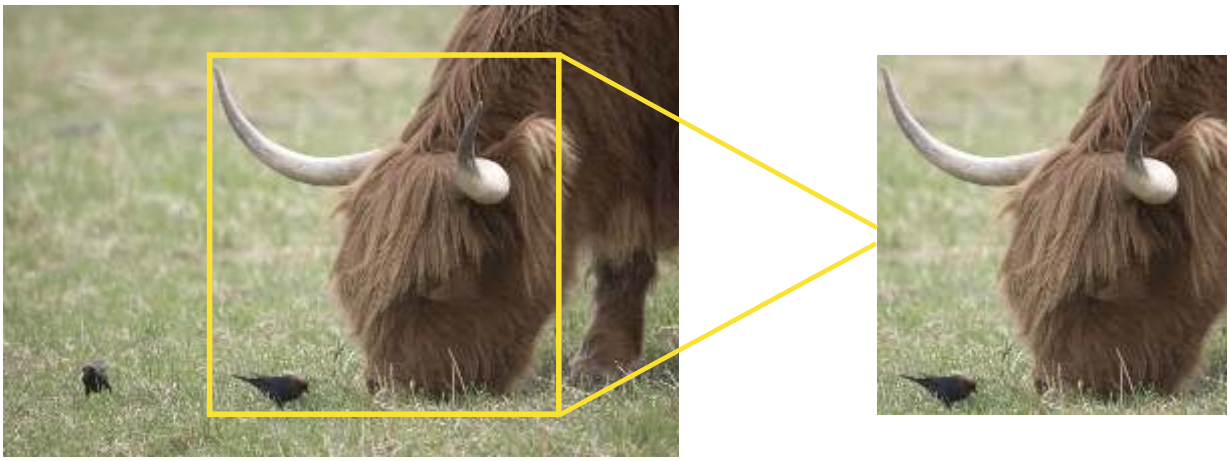


Figure 6: Original and cropped JPEG compressed images.

The output of the algorithm is the following:

```
image size: 668 x 687
window step size: 64
total number of evaluated windows: 3025
number of meaningful windows: 2883 (95 %)
number of meaningful windows for each JPEG grid origin:
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0    2883       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
best log(NFA) for each JPEG grid origin:
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -  -930.3       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
number of meaningful JPEG grids found: 1
most meaningful JPEG grid origin (4,4) with NFA: 10^-930.259

The most meaningful JPEG grid origin is not (0,0).  This may indicate that
the image have been cropped.
```

3.3 Forgery Detection

In the forged image of Figure 7, with $W = 256$, the list of votes returned two meaningful grids $(0, 0)$ and $(0, 5)$. In Figure 8, the red area represents the windows which voted for a foreign grid and the blue area the windows with a non-meaningful vote: the whole image voted for the coordinates $(0, 0)$, whereas the foreign area for another. We conclude that the area marked in red has a JPEG grid with an offset which is different from the rest of the image.

Figure 7: Original and forged image.

Figure 8: Result for the forged image. Meaningful windows for a foreign grid in red, non-meaningful windows in blue.

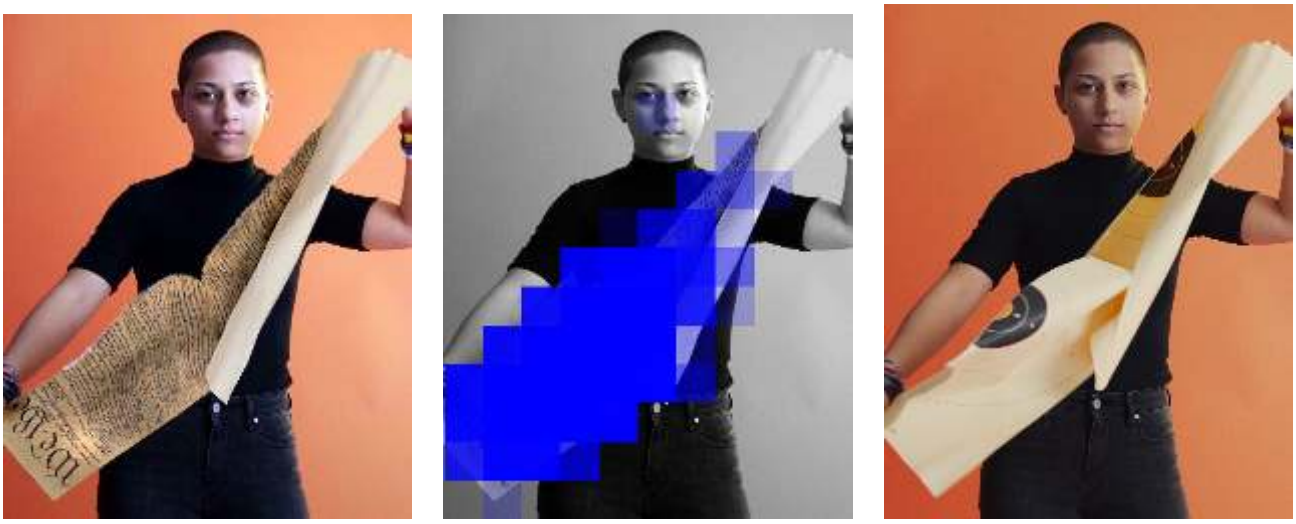
The output of the algorithm is the following:

```
image size: 3264 x 2448
window step size: 256
total number of evaluated windows: 3510
number of meaningful windows: 3458 (98 %)
number of meaningful windows for each JPEG grid origin:
    3455        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       3        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
       0        0        0        0        0        0        0        0
best log(NFA) for each JPEG grid origin:
 -1162.4        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
    -4.2        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
       -        -        -        -        -        -        -        -
number of meaningful JPEG grids found: 2
most meaningful JPEG grid origin (0,0) with NFA: 10^-1162.4
second most meaningful JPEG grid origin (0,5) with NFA: 10^-4.22603

This image shows more than one meaningful JPEG grid.  This may be caused
by image manipulations such as resampling, copy-paste, splicing, or some
particular periodic pattern in the scene.  Please examine the deviant
meaningful blocks to make your own opinion about a potential forgery.
```

In some cases, an area can be revealed where there are no meaningful grids at all. This may be caused by several reasons: if there has been an external copy-paste from an uncompressed image, or by operations such as erasing. Further work will look into areas of non-meaningful overlapping windows.

For example, the image of Figure 9 is an example of a real case image from the social network Twitter posted to propagate fake news. The method with $W = 64$ is applied to the forged image. The blue area is forged, as it can be seen thanks to the original image on the right. Indeed, there are no detectable JPEG blocks in this area probably caused by too much post processing.



forged image | detection | original image

Figure 9: Real case image from Twitter, detection and original image. The blue area represents an area without any detection.

4 Limitations

As the method relies on the ability of the cross-difference filter to reveal the blocking artifacts, in some cases which are detailed below, it may not detect the proper grid or any grid at all.

4.1 High Quality Images

When the image is only slightly compressed, with quality parameters 98, 99 or 100, the JPEG blocks are often imperceptible, even after the cross-difference enhancement. The algorithm may fail to detect the JPEG global grid for small images of high quality.

4.2 Interference with Resampling traces

The main limitation of the proposed method is its relation with the presence of periodic patterns in the image. Indeed, a JPEG grid is revealed by periodic structures on the cross-difference. To be detected, the structure needs to show, locally, a period of 8 pixels (or a multiple). In rare occasions, this may be observed in a natural image, and it is of course a violation of our *a contrario* hypothesis. Nevertheless, this phenomenon is arguably rare, as the method requires the presence in the image of a periodic structure with the right 8-period on both, the horizontal and vertical directions. While rare in natural images, such periodic traces, however, can arise as artifacts left by an image resampling operation.

Resampling an image creates a regular pattern [8] which can, when aligned horizontally or/and vertically interfere with the JPEG 8×8 grid. For example, a JPEG image loses (naturally) its JPEG blocking artefact when stretched. However, sometimes, it creates a new periodic pattern as it can be detected in the image of Figure 10. The image was JPEG compressed and of size 512×512, after being stretched vertically, became of size 512×520.


Figure 10: Compressed image Pelican and stretched version pelican (8 pixels in height).

The output of the algorithm for the stretched image is the following:

```
image size: 512 x 520
window step size: 32
total number of evaluated windows: 16320
number of meaningful windows: 56 (0 %)
number of meaningful windows for each JPEG grid origin:
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       9       0
       0       0       0       0       0       0      11       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       9       0
       0       0       0       0       0       0      23       0
       0       0       0       0       0       0       4       0
best log(NFA) for each JPEG grid origin:
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -    -2.3       -
       -       -       -       -       -       -    -2.5       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -       -       -
       -       -       -       -       -       -    -2.5       -
       -       -       -       -       -       -    -4.9       -
       -       -       -       -       -       -    -1.1       -
number of meaningful JPEG grids found: 5
most meaningful JPEG grid origin (6,6) with NFA: 10^-4.88746
second most meaningful JPEG grid origin (6,2) with NFA: 10^-2.48428

This image shows more than one meaningful JPEG grid.  This may be caused
by image manipulations such as resampling, copy-paste, splicing, or some
particular periodic pattern in the scene.  Please examine the deviant
meaningful blocks to make your own opinion about a potential forgery.
```

In our case illustrated in Figure 11, the resampling was applied to an uncompressed image, and it led to the detection of several grid origins. The image on the left was stretched horizontally and vertically to obtain an image twice as big.

```
image size: 1024 x 1024
window step size: 64
total number of evaluated windows: 14400
number of meaningful windows: 10473 (72 %)
number of meaningful windows for each JPEG grid origin:
      23        0       99        0      182        0      573        0
       0        0        0        0        0        0        0        0
     424        0     1132        0      939        0     3320        0
       0        0        0        0        0        0        0        0
      62        0       25        0       76        0      242        0
       0        0        0        0        0        0        0        0
     208        0      512        0      759        0     1897        0
       0        0        0        0        0        0        0        0
best log(NFA) for each JPEG grid origin:
   -21.2        -    -23.1        -    -29.7        -    -79.5        -
       -        -        -        -        -        -        -        -
   -43.9        -    -90.8        -   -108.2        -   -180.4        -
       -        -        -        -        -        -        -        -
   -18.5        -    -16.1        -    -30.3        -    -41.0        -
       -        -        -        -        -        -        -        -
   -43.1        -    -80.8        -    -89.2        -   -116.3        -
       -        -        -        -        -        -        -        -
number of meaningful JPEG grids found: 16
most meaningful JPEG grid origin (6,2) with NFA: 10^-180.387
second most meaningful JPEG grid origin (6,6) with NFA: 10^-116.288

This image shows more than one meaningful JPEG grid.  This may be caused
by image manipulations such as resampling, copy-paste, splicing, or some
particular periodic pattern in the scene.  Please examine the deviant
meaningful blocks to make your own opinion about a potential forgery.
```

Future work will focus on using similar techniques specially tailored for image resampling detection and on being able to tell the difference between resampling and JPEG compression. Our best guess so far is to look for several periodic patterns, not only of 8 pixels. We observed that if an image is upsampled before being also JPEG compressed, the resampling traces can remain detectable if the final compression is mild enough.

5 Conclusion

The proposed JPEG grid detection method involves Chen and Hsu's cross-difference filtering to emphasize blocking artifacts. The detection is made locally in a family of windows, where each local maximum votes for a JPEG grid origin, and the most voted grid position is taken as candidate. An *a contrario* validation step of this candidate is used to control the number of false detections. The resulting method is unsupervised and depends on a single parameter for selecting the balance between exhaustiveness and speed of the algorithm.

The algorithm can be used in image forensics to detect cropped or tampered images, and it can also be used to provide the grid localization for further JPEG analysis. The main limitation of the

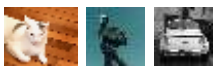
Figure 11: Uncompressed image and double stretched in both directions image.

proposed method is that image upsampling traces may lead to meaningful detections that are not JPEG related. Future work will concentrate on this decision problem.

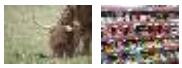
Acknowledgment

Work funded by the ANR-DGA challenge DEFALS (ANR-16-DEFA-0004).

Image Credits

 by the authors,

 Kodak Lossless True Color Image Suite[2],

 Image Manipulation Dataset from Friedrich-Alexander-Universitat Erlangen-Nurnberg[3],

 from Twitter.

[2] http://http://r0k.us/graphics/kodak/
[3] http://www5.cs.fau.de/research/data/image-manipulation/

References

[1] Y-L. Chen and C-T. Hsu, *Image tampering detection by blocking periodicity analysis in JPEG compressed images*, in IEEE 10th Workshop on Multimedia Signal Processing, Oct 2008, pp. 803–808. http://dx.doi.org/10.1109/MMSP.2008.4665184.

[2] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis*, Springer, 2008. ISBN 978-0-387-74378-3.

[3] A. Gordon, G. Glazko, X. Qiu, and A. Yakovlev, *Control of the mean number of false discoveries, Bonferroni and stability of multiple testing*, The Annals of Applied Statistics, 1 (2007), pp. 179–190.

[4] R. Grompone von Gioi, J. Jakubowicz, J-M. Morel, and G. Randall, *LSD: a Line Segment Detector*, Image Processing On Line, 2 (2012), pp. 35–55. https://doi.org/10.5201/ipol.2012.gjmr-lsd.

[5] W. Li, Y. Yuan, and N. Yu, *Passive detection of doctored JPEG image via block artifact grid extraction*, Signal Processing, 89 (2009), pp. 1821 – 1829. https://doi.org/10.1016/j.sigpro.2009.03.025.

[6] W.S. Lin, S.K. Tjoa, H.V. Zhao, and K.J. Ray Liu, *Digital image source coder forensics via intrinsic fingerprints*, IEEE Transactions on Information Forensics and Security, 4 (2009), pp. 460 – 475. https://doi.org/10.1109/TIFS.2009.2024715.

[7] T. Nikoukhah, R. Grompone von Gioi, M. Colom, and J-M. Morel, *Automatic JPEG grid detection with controlled false alarms, and its image forensic applications*, in Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2018, pp. 378–382. http://dx.doi.org/10.1109/MIPR.2018.00083.

[8] A. C. Popescu and H. Farid, *Exposing digital forgeries by detecting traces of resampling*, IEEE Transactions on Signal Processing, 53 (2005), pp. 758–767. https://doi.org/10.1109/TSP.2004.839932.