JPEG Grid Detection based on the Number of DCT Zeros and its Application to Automatic and Localized Forgery Detection

T. Nikoukhah J. Anger T. Ehret M. Colom J.M. Morel R. Grompone von Gioi
CMLA, CNRS, ENS Paris-Saclay, Université Paris-Saclay
{nikoukhah,anger,ehret,colom,morel,grompone}@cmla.ens-cachan.fr

Abstract

This work proposes a novel method for detecting JPEG compression, as well as its grid origin, based on counting the number of zeros in the DCT of 8×8 *blocks. When applied locally, the same method can be used to detect grid alignment abnormalities. It therefore detects local image forgeries such as copy-move. The algorithm includes a statistical validation step which gives theoretical guarantees on the number of false alarms and provides secure guarantees for tampering detection. The performance of the proposed method is illustrated with both quantitative and visual results from well-known image databases and comparisons with state of the art methods.*

1. Introduction

Image tampering is currently used massively on the web and continuously feeds fake news [12]. This issue has become important since digital image manipulation tools are available to the general public. Some social networks even allow to edit images and videos directly online. Since these platforms (Facebook, Instagram, Snapchat, etc.) delete all metadata for confidentiality reasons, there is a crucial need for journalists and mass-media producers to have access to tools for detecting forgeries from the image itself. Several such tools are readily available online, *Fotoforensics*, *Forensically*, *Ghiro* and *Reveal*[1] for instance. These tools provide a number of "tampering localization maps" [26] in the form of so-called image *heat maps* revealing suspicious alterations. However, those localization maps work only as trace enhancers and do not provide any solid proof of forgery. The result needs to be analyzed with care and interpreted by experts. We attempt here to fill in this gap by producing an automatic method using the tampering traces left in the image and taking an automatic decision about forgeries and their precise location. Indeed, even in absence of image metadata, much information about the history of the image is still present in the image itself. Many processes undergone by an image leave invisible but detectable traces all over the image. Following [2, 3, 10, 11, 12, 15, 20, 29], we address one of the most common processes, namely the JPEG compression. This compression leaves traces in the form of 8×8 pixels blocking artifacts that produce a grid over the image.

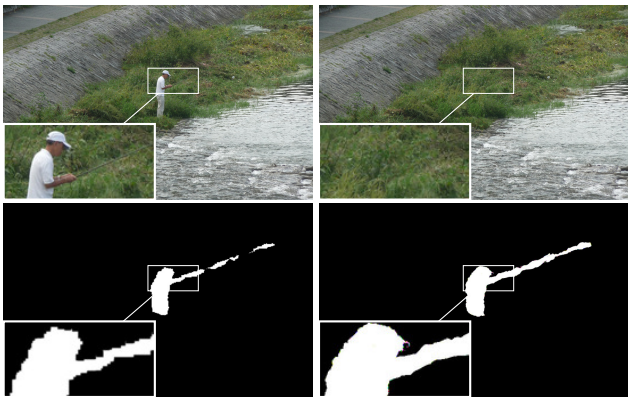

Figure 1: Result of the proposed method, from top left to bottom right: original image, forged image (input of the method), detected forgeries, ground-truth. The fisherman and the fishing rod have been removed. The resulting detected mask does not require visual interpretation.

Here we propose an automatic and statistically founded method for detecting the grid division of the image. The proposed method, hereafter referred to as ZERO, performs a global grid detection and then applies it more locally to detect forgeries. Indeed, image splicing generally breaks locally the original grid alignment. A locally detected grid therefore may contradict the global grid and become a reliable cue that a forgery took place. We evaluate ZERO against several state of the art algorithms on publicly available datasets. The detection does not require a human expert's decision as shown in Figure 1, and is associated with theoretical guarantees.

[1] https://fotoforensics.com, https://29a.ch/photo-forensics, https://www.getghiro.org, http://reveal-mklab.iti.gr

The paper is organized as follows. A brief description of the JPEG compression algorithm is presented in Section 2, followed by a review of the state of the art in Section 3. In Section 4, the proposed method is described and its application to forgery detection is detailed in Section 5. Finally, evaluations of the performances of the method are presented in Section 6. A discussion concludes this article.

2. JPEG Compression

The JPEG image format is widely used by most digital cameras (including smartphones) and by social networks to share images on the Internet [16, 27]. First, the image is converted from the RGB to the YC_bC_r colorspace. The Y channel is the luminance component while C_b and C_r correspond to the chroma components. For simplicity, we will focus on the Y channel[2] since it contains most of the image visual information, which we will refer to as the luminance image. The luminance image is then partitioned into non-overlapping 8×8 pixel blocks and the 2D discrete cosine transform (DCT) type II is applied to each of these blocks. Due to the independent encoding of the blocks, pixel discontinuities are introduced across the block boundaries of the decompressed image.

Each of the 8×8 blocks undergoes a quantization step performed in the spectral domain. A quantization table (related to the compression quality) provides a factor for each DCT component. At this step, some DCT coefficients are cancelled out when they have a small value relative to the quantization factor. After this step, all 8×8 blocks have a number of zeros that depends both on the compression quality and on the image content. Finally, the quantized DCT coefficients are losslessly compressed by exploiting, among other things, the presence of zero values.

3. State of the art

There are several tools for detecting forgeries based on JPEG compression traces. The most famous ones (used by mass-media online) are ELA (Error Level Analysis) [18] and GHOST [11], which are very similar. Both attempt to detect JPEG compression ratio differences throughout the image. In JPEG forensics, the main methods are either based on the histograms of DCT coefficients [2, 29] or based on the detection of a higher contrast at the block edges [3, 10].

Three methods [15, 20, 23] are closely related to ours as they detect forgeries by locating inconsistencies of JPEG blocking artifacts. In [20], the image is filtered based on local derivatives, weak edges are detected, and the coherence with an aligned 8×8 grid is measured. A feature corresponding to the local strength of the blocking pattern is extracted. Feature variations indicate local absence or misalignment of the grid and can be considered as a tampering cue. In [15], the authors use the artifact measure introduced by Fan and Queiroz [10]: their method evaluates multiple grid positions with respect to a fitting function. Areas with low contribution are identified as inconsistent with the main grid and therefore potentially tampered. An image segmentation step is introduced to differentiate between inconsistencies produced by tampering and those attributable to image content. In [23], the authors apply the filter proposed by Chan and Hsu [3] to reveal these blocking artifacts before using a statistical method to increase the reliability of the detection.

These methods make it possible to detect what is undoubtedly one of the most commonly used tampering schemes: the copy and paste of image parts which break the alignment of the original grid, either because of its location or because of transformations (scaling, rotation, etc.) of the manipulated area. Another way to alter an image is by simply cropping it to remove undesirable parts of the photographed scene. This method, frequent in photojournalism, can significantly alter the interpretation of a scene. To detect cropping, Li *et al*. [20] and Nikoukhah *et al*. [22, 23] detect the grid globally and exploit the fact that its origin may no longer be at $[0, 0]$. Our method, being based on the detection of the global JPEG grid, is also able to detect this type of manipulation.

Finally, besides the spectral and spatial methods, a third way has been introduced based on the principle that JPEG compression has the goal to minimize the file size [22]. Therefore, to detect the JPEG grid, the method compresses the image with all 64 possible grids and selects the one yielding the shortest file. Our method extends this idea as we decide to pick the likeliest JPEG blocks as those containing the largest number of zero DCT coefficients.

4. JPEG grid detection

As described before, the JPEG algorithm sets to zero some of the DCT coefficients of 8×8 blocks. Based on this fact, the core of the proposed method is thus to count the total number of zeros of each hypothesized DCT block. In the presence of JPEG compression, this number should be maximum when the 8×8 block is aligned with the JPEG grid. Indeed, non-aligned blocks include additional discontinuities due to blocking artifacts, leading to larger DCT coefficient values relative to the aligned block, see Figure 2 (right). A statistical test is used to confirm that a JPEG grid is actually present.

4.1. Voting process

The first thing to explain is how to determine which DCT coefficient are "zeros". During JPEG decompression, an inverse DCT is performed on each block, transforming the integer DCT coefficients to real pixel values. Then, those

[2] $Y = 0.299R + 0.587G + 0.114B$.

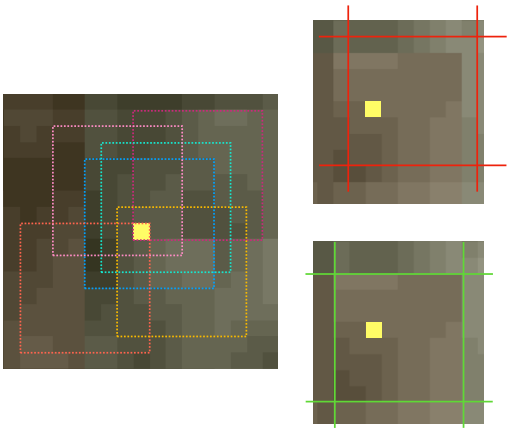

Figure 2: Each pixel (yellow) belongs to 64 different 8×8 blocks of the image. Six of them were drawn in different colors on the left. Top right shows (in red) the position of a patch not aligned with the grid. Bottom right shows (in green) the position of the patch containing the pixel matching the JPEG grid.

pixel values are quantized to produce an integer image. Notice that this quantization step is different from the one during the compression. This pixel quantization step, of course, also modifies the corresponding DCT values. Thus, a DCT coefficient that was put to zero during compression does not keep an exact zero value after decompression. Yet it remains close to zero. We propose to count the number of coefficients with absolute values smaller than 0.5. This allows to discriminate zeros even when DCT coefficient quantization is at its finest rate, namely to integer values.

Each pixel may belong to 64 different overlapping 8×8 blocks, as illustrated in Figure 2. In the first step of the method, those 64 blocks are evaluated for each pixel. The 64 DCTs are computed as well as the corresponding number of zeros. Each pixel votes for the grid origin of the block with most zeros. In the case of a tie, the pixel does not vote.[3]

Performing the count as described requires computing 64 DCTs per pixel, but every block is shared with 64 other pixels and this can be exploited to avoid recomputing the DCT. Algorithm 1 describes the procedure. A table is used to keep track of the largest number of zeros found for each pixel. The DCT of every 8×8 block in the image and its number of zeros are computed. Every pixel included in the block is checked and the table of zeros is updated when the current block has more zeros than in other blocks previously evaluated. The table of votes is also updated to the grid origin corresponding to the block with more zeros ($\text{GridAlignedWith}(b)$), or to NON VALID in case of a tie.

[3]There is relevant information when two or more blocks have the same number of zeros. However, exploiting this information would make the statistical evaluation more complex. Given that the method is already reasonably sensitive, we preferred to keep a simple formulation.

Algorithm 1: Compute grid votes

input : luminance channel L defined on Ω
output: grid vote map

$\text{votes}(\Omega) \leftarrow$ NON VALID — *initialize votes*
$\text{zeros}(\Omega) \leftarrow 0$ — *initialize number of zeros*
for $b \in \text{Blocks8x8}(\Omega)$ **do** — *loop on all 8x8 blocks*
  $d \leftarrow \text{DCT}(L(b))$ — *DCT of the block*
  $z \leftarrow \sum_{d_i \in d} \mathbb{1}_{|d_i|<0.5}$ — *number of zeros in the block*
  for $(x, y) \in b$ **do**
    if $z = \text{zeros}(x, y)$ **then** — *tie, do not vote*
      $\text{votes}(x, y) \leftarrow$ NON VALID
    else if $z > \text{zeros}(x, y)$ **then**
      $\text{zeros}(x, y) \leftarrow z$
      $\text{votes}(x, y) \leftarrow \text{GridAlignedWith}(b)$
return votes

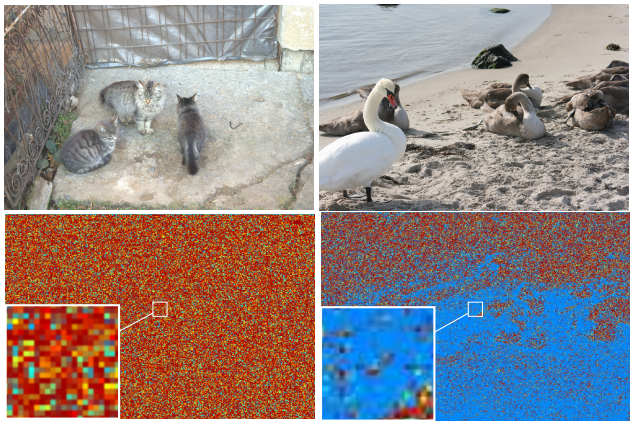

Figure 3: Left: Uncompressed image and its vote map. Right: JPEG compressed image and its vote map. In both cases, the pixels which return a NON VALID vote (a tie) are shown in dark red.

Figure 3 shows two examples of vote maps. On the left, the image is uncompressed and we observe a random vote map. On the right, the same is shown for a JPEG compressed image.

4.2. Statistical validation

When analyzing a JPEG image, the most voted grid probably corresponds to the right one. But, even in uncompressed images, one of the grids will get more votes than the others, probably by a small margin. A statistical criterion is therefore needed to decide whether this prominence is caused by JPEG compression or not.

The proposed validation procedure is based on the *a contrario* theory [8], which relies on the non-accidentalness principle [21, 28]. Informally, this principle states that there should be no detection in noise. In the words of D. Lowe,

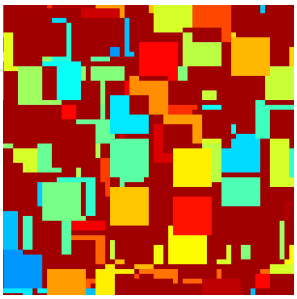

Figure 4: Zoom on a vote map for an image of Gaussian noise. Each color represents a vote for a given grid origin. The dark red color correspond to NON-VALID votes (the ties). One can observe entire blocks of 8×8 pixels voting for the same origin; this is the case when a block has a local maximum of number of zeros.

"*we need to determine the probability that each relation in the image could have arisen by accident, $P(a)$. Naturally, the smaller that this value is, the more likely the relation is to have a causal interpretation*" [21, p. 39]. This principle has shown its practical use for detection purposes such as line segment detection [14], vanishing points detection [19], anomaly detection [7], or forgery detection [1, 23].

In our context, we need to assess the probability that a given grid origin gets a large number of votes purely by chance. To that aim, a stochastic null model H_0 for the votes is required. It is here easily given by Laplace's principle of indifference: in absence of JPEG compression, each of the 64 blocks containing a given pixel would have the same chance of being the one with the largest number of zeros; that would depend on the image content and there is no reason to suppose that it is synchronized with a particular 8×8 grid origin. However, the votes of neighbor pixels are not independent, even in noise images. Indeed, there are always blocks that are local maxima of the number of zeros, and those blocks get the votes of every pixel belonging to it. Figure 4 shows a vote map obtained in an image of noise. Votes are correlated within a distance of 8 pixels; on the other hand, pixels at distance larger than eight are largely uncorrelated. Thus, we define a stochastic null model H_0 for votes at distance eight in which votes are independent and uniformly distributed among all the 64 grid origins.

Let us suppose that we are observing a square patch of an image where the number of votes for a given valid grid origin is counted at a distance of eight pixels. Let us say that k votes are counted for that valid grid among a total of n votes. Under the null hypothesis H_0, votes for the given grid origin become Bernoulli random variables with probability $\frac{1}{64}$. So under H_0, the number of votes becomes a random variable K and, given the independence of votes (at distance larger than eight), it follows a binomial distribution of parameter $p = \frac{1}{64}$. Thus,

$$\mathbb{P}(K \geq k) = \mathcal{B}(n, k, p) = \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j},$$

where $\mathcal{B}(n, k, p)$ is the tail of the binomial distribution. Given an observed number of votes k, $\mathbb{P}(K \geq k)$ is the probability of obtaining at least k votes under H_0. When this probability is small enough, there exists evidence to reject the null hypothesis and declare that a significant grid origin was found. However, the multiplicity of tests needs to be taken into account. Indeed, if 100 tests were performed, it would not be surprising to observe an event that appears with probability 0.01 under random conditions. The number of tests N_T needs to be included as a correction factor, as it is standard in statistical multiple hypothesis testing [13].

Following the *a contrario* methodology, we define the Number of False Alarms (NFA) of a candidate grid g on a given window w as

$$\mathrm{NFA}(g, w) = N_T \mathbb{P}(K \geq k). \tag{1}$$

One can show [8] that under the null hypothesis H_0 the expected number of false alarms with $\mathrm{NFA}(g, w) < \varepsilon$, is bounded by ε:

$$\mathbb{E}_{H_0}\left[\sum_{(g,w)\in\mathcal{N}_T} \mathbb{1}_{\mathrm{NFA}(g,w)<\varepsilon}\right] < \varepsilon, \tag{2}$$

where $\mathcal{N}_T$ is the set of N_T tests. As a result, ε corresponds to the mean number of false detections per image under H_0. In most practical applications, the typical value $\varepsilon = 1$ is suitable; we will set it once and for all in our application as well. With this choice, the expected number of false detections per image under H_0 is guaranteed to be upper-bounded by 1.

The same criterion will be used for the whole image as well as for all sub-images. We want the tests to be selective enough to discriminate a JPEG grid using only a local region of the image. As we will see later, this also allows to detect local forgeries. Thus, every square window of a $X \times Y$ pixels image is included in the family of tests and the 64 grid origins are tested for each one. Then, the number of tests can be approximated by

$$N_T = 64 \cdot XY \cdot \sqrt{XY}, \tag{3}$$

where $\sqrt{XY}$ gives a rough estimation of the possible window sides, and XY gives the number of possible positions for the square windows of a given size. All in all, given a window to be analyzed, the grid origin with the maximum of votes is selected and its number of votes at distance eight pixels is counted. Then, the NFA is given by

$$\mathrm{NFA} = 64 \cdot XY \cdot \sqrt{XY} \cdot \mathcal{B}(n, k, p). \tag{4}$$

Algorithm 2: JPEG grid detection

input : image I of size $X \times Y$
output: main grid G
output: NFA value of the main grid

$L \leftarrow \text{Luminance}(I)$
$\text{votes} \leftarrow \text{ComputeVotes}(L)$ *algorithm 1*
$G \leftarrow \arg\max_{g \text{ is VALID}} \sum_{x,y} \mathbb{1}_{\text{votes}(x,y)=g}$ *most voted valid grid*
$v \leftarrow \sum_{x,y} \mathbb{1}_{\text{votes}(x,y)=G}$ *number of votes for main grid*
$\text{NFA} \leftarrow 64XY\sqrt{XY} \cdot \text{BinTail}\left(\frac{XY}{64}, \frac{v}{64}, \frac{1}{64}\right)$
if $\text{NFA} < 1$ **then**
  return G, NFA *JPEG grid found*
else
  return $\varnothing$ *JPEG grid not found*

A JPEG grid is detected when $\text{NFA} < 1$.

According to the theory, to test a grid we must count one vote out of eight in both directions. These are the votes inside the window with coordinates $(x_0 + 8i, y_0 + 8j)$ for integers i and j. This test must be performed for all other 64 grids with x_0 and y_0 in $\{0, 1, \ldots, 7\}$. A simpler way to evaluate a lower bound for the number of votes in the most voted grid is as follows. Instead of counting votes at distance of eight pixels for those offsets, we can count every vote and divide the number by 64. Indeed, let v be the total of votes in the window for the given grid. If those votes were equally distributed on the sub-samplings, one would have $k = \frac{v}{64}$ for each of the sub-samplings. If not, necessarily one of the sub-samplings will have more votes. Hence we can deduce that there is at least one of those sub-samplings with k votes satisfying $k \geq \frac{v}{64}$. So by counting every vote and dividing the count by 64 we are considering the worst case and we are sure that a detected grid is meaningful. Naturally, the count of votes for every pixel in the window is also divided by 64. Algorithm 2 describes the ensuing JPEG grid detection method.

5. Forgery detection

Our JPEG grid detection can be performed globally but also at every square window of the image. When a local region has a meaningful grid that is different from the main one, it means that it is a foreign grid and so the result of a forgery. Indeed, when part of a JPEG image is copied and pasted, it retains its grid traces. In 63 over 64 times (assuming that the forger did not explicitly align the grid), the grid origin will not correspond to the main one, thus allowing its detection. This is true whether it is a case of copy-move from the same image or when the copied part is taken from a different JPEG image.

Algorithm 3: Forgery detection

input : grid vote map defined on Ω of size $X \times Y$
input : main grid G
input : neighborhood size $W = 12$
output: forgery mask

$\text{mask}(\Omega) \leftarrow \text{FALSE}$ *initialize forgery mask*
for $(x, y) \in \Omega$ **do**
  if $\text{votes}(x, y)$ *is* VALID **and** $\text{votes}(x, y) \neq G$ **then**
    $R \leftarrow \text{GrowRegion}(\text{votes}, x, y, W)$
    $B \leftarrow \text{BoundingBox}(R)$
    $N \leftarrow \max(B_x, B_y)$ *size of square bounding box*
    $\text{NFA} \leftarrow 64XY\sqrt{XY} \cdot \text{BinTail}\left(\frac{N^2}{64}, \frac{|R|}{64}, \frac{1}{64}\right)$
    if $\text{NFA} < 1$ **then** *forgery found*
      $\text{mask}(R) \leftarrow \text{TRUE}$ *mark tampered region*
    $\text{votes}(R) \leftarrow \text{NON VALID}$ *do not test again in R*
$\text{mask}(\Omega) \leftarrow \text{Closing}(\text{mask}, W)$ *fill holes in mask*
return mask

The same algorithm as described in Section 4 can be applied directly on every square window. But this would be computationally expensive. Instead, we propose a heuristic method using a greedy algorithm to accelerate the search for forged regions. Nevertheless, the final validation uses the same statistical test used for the global grid.

Algorithm 3 describes the method. The vote map is partitioned into connected regions sharing the same grid vote. A region growing algorithm is used for partitioning the vote map: starting from a seed pixel (x, y), the neighbor pixels are iteratively aggregated when voting for the same grid. As Figure 4 shows, votes for the same grid often have gaps, so a relaxed notion of neighborhood is needed. A window with a meaningful grid origin must have a vote density of at least $\frac{1}{64}$ for the right grid origin. Thus, the votes for the right grid should not be further away than eight pixels. To allow for some variation in the distribution, we set this neighborhood size a little larger and use $W = 12$.

Then, for each region with a valid grid origin different from the main one, a square bounding box is computed and the statistical test is performed with the NFA framework introduced before. When a meaningful region is found with a grid origin different from the main one, the pixels in the region (which all voted for the same grid) are marked in a forgery mask. Figure 5 shows an example. After a region is evaluated, its votes are marked as NON VALID to gain time by preventing the same pixels from being explored again. Due to variations in the number of votes, the raw forgery mask contains holes. To give a more useful forgery map, these holes are filled by a mathematical morphology closing operator [25] with a square structuring element of size W (the same as the neighborhood used in the region growing).

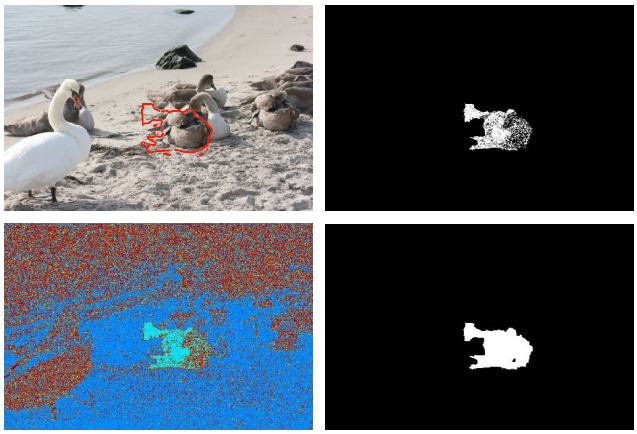

Figure 5: Up-left: an image with a tampered region indicated in red. Down-left: grid origin vote map. Up-right: raw forgery mask. Down-right: final forgery mask after holes filled by a mathematical morphology closing operation.

Figure 5 shows an example of the final forgery mask.

Figure 6 illustrates some limitations of the proposed method. Forgeries are detected as regions in which the local grid does not agree with the global grid. This means that when the grid of the forged regions aligns perfectly with the global grid, our method will fail to detect the forgery. Nevertheless, this happens only once for every 64 positions. In a saturated region, the DCT coefficients of the blocks are all equal to zero except for the DC coefficient. The number of zeros are tied and the votes are all non-valid. This means that it is impossible to distinguish the JPEG grid in saturated regions. Since no valid JPEG grid can be found, it will never disagree with the global grid and therefore saturated parts of a forgery cannot be found. However, as soon as a part of the forgery is not saturated it can be detected as it is shown in Figure 6. Another limitation is when the forged region is too small. Since the statistical test must be satisfied to detect a forgery, there is a minimal detectable region size that depends on the image size, the JPEG compression quality and the image contents.

6. Experimental results

In this section we evaluate the proposed method on two tasks: global grid detection and forgery detection. Comparisons with other available methods are performed which illustrate the superiority of our approach, both in terms of low false rate and accuracy on true detections.

6.1. Grid detection

Grid detection is our main application as it represents the first step of most forgery detection algorithms but this is not the only application. In image restoration, grid detection is also used to remove grid artifacts by a deblocking procedure [4]. To this aim, it is useful to detect the grid in every case, and the hardest cases are when the quality factor is high. Our first evaluation was performed with cases where no detection should be obtained. This experiment is important to illustrate the main strength of the proposed algorithm: it gives a principled method to decide whether the image has undergone JPEG compression or not. The first dataset is composed of 200 images of noise following a Gaussian distribution. Two image sizes were generated: 500×500 and 1000×1000. We also used the UCID [24] uncompressed image collections (886 images) and Kodak [17] (24 images). The method's performance is compared to three other methods in Table 1. ZERO is the sole to produce no false detection.

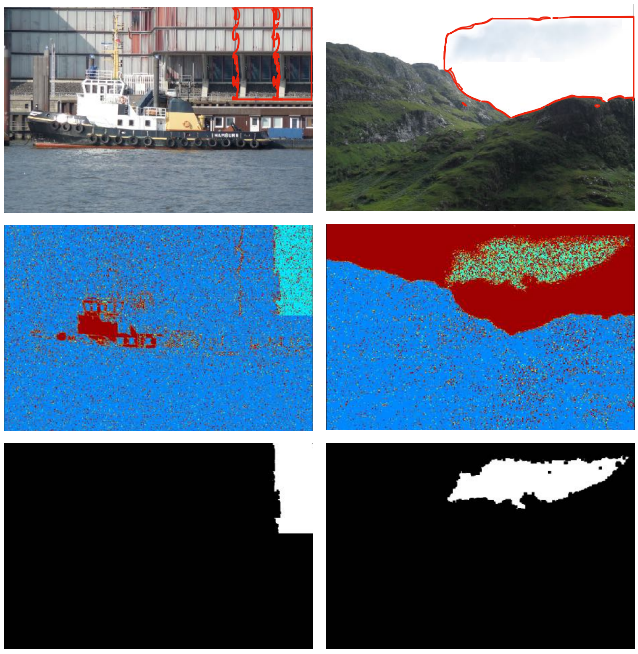

Figure 6: Up: images with tampered regions indicated in red. Middle: grid origin vote maps. Down: detected forgery masks. On the left, an example of a missed detection: one of the two forged regions was not detected because its local grid was correctly aligned with the global grid. On the right, an incomplete detection caused by saturation in the image.

To illustrate the validity of the proposed approach, the second experiment is applied to 12 288 images generated from the Kodak [17] uncompressed image database. The 24 images (of size 768×512) were compressed, using the `imagemagick` tool to increasing quality factors (50, 60, 70, 80, 90, 93, 95, 98 and 99), then cropped into the 64 different positions to test all possible grid positions. The results are shown in Table 2.

This application is useful by itself. Indeed, knowing if an image has undergone JPEG compression is important. Moreover, knowing its grid origin can tell whether an image

		Dataset		
		Noise	UCID [24]	Kodak [17]
BLK [20]	% true	—	—	—
	% false	100	100	100
GOD [23]	% true	—	—	—
	% false	0	0.6	0
SGOD [22]	% true	—	—	—
	% false	0	0.5	0
ZERO	% true	—	—	—
	% false	0	0	0

Table 1: Results of the proposed method compared to BLK, GOD and SGOD on uncompressed images. Percentage of images where a JPEG grid was detected. In this case, every detection is a false detection.

		JPEG quality factor				
		≤ 80	90	95	98	99
BLK [20]	% true	97	95	85	31	0
	% false	3	5	15	69	100
GOD [23]	% true	100	91	70	55	41
	% false	0	0.003	0.05	0.06	0.1
SGOD [22]	% true	100	100	100	50	0
	% false	0	0	0	0	0
ZERO	% true	100	100	100	100	100
	% false	0	0	0	0	0

Table 2: Results of the proposed method compared to BLK, GOD and SGOD on 12 288 compressed and cropped images generated from the Kodak [17] database. Percentages of true and false JPEG grid detections.

has been cropped. Table 2 shows a perfect detection of JPEG compression and cropping after a JPEG compression even with a very high quality (99%). JPEG compression at 100% is not detectable by the proposed method as it does not increase the number of zeros in the DCT. Even in those cases, the proposed method gives no false detection.

6.2. Forgery detection

To test how detections can be based on JPEG grid misplacement, we used a database of tampered images created by copy-paste [5]. The copied area is taken from the same image and its borders hidden in a smooth transition. However, the proposed method would work exactly the same if the copied area came from a different JPEG image. A detection based on a disparity in the grid position in some blocks may fail with probability $1/64$ when the copied area is placed so that its grid is aligned with the global grid.

The proposed algorithm gives back two types of important information: the main JPEG grid origin detection and the forgery detection when it is the case. The

		ZERO	GOD [23]	CMFD [6]
Original	true	—	—	—
	false	0	6	5
Forged	true	35	23	44
	missed	2(+11)	21(+4)	4

Table 3: Quantitative results on the dataset [5], containing 48 pairs of original and forged images. 11 of the images are correctly detected as not JPEG by ZERO (4 for GOD). Since these image are not JPEG the forgery cannot be detected therefore they are reported in parenthesis.

database [5] contains 48 pairs of original and forged images. On this dataset, we compare the results of GOD [23], CMFD [6] and ours. CMFD by Cozzolino *et al.* [6] proposes a method that allows a precise and accurate detection of a copy-move forgeries inside a single suspicious image. Therefore this method is particularly appropriate on the tested images and is used as reference for these types of forgeries. We used the implementation provided by [9]. Note that our method does not need the forged region to come from the same image. Both GOD [23] and ZERO are able to filter out the 11 pairs that are either not JPEG compressed, or with a quality factor of 100%. On the 37 remaining images, the proposed method is able to detect most forgeries with no false detection. The quantitative results are reported in Table 3. Regarding the two missed forgeries, one is caused by the copied area having the same grid origin as the global image, and the other one is due to its small size.

We also qualitatively compared our method to other forgery detection methods based on compression traces analysis in Figure 7. This allows us to compare with state-of-the-art methods that only produce heatmaps.

7. Conclusion

This paper presented a novel JPEG grid detection and tampering localization method based on the number of zeros in the DCT blocks. It has a high accuracy detecting JPEG compression up to quality factor of 99%. It performs reliable reverse engineering and detects forgeries by giving an automatic, localized, and reliable result without requiring any human interpretation. The proposed algorithm is efficient; the bottleneck is the computation of the vote map, which has about the same complexity of 64 JPEG compressions. Color information and handling ties in the number of zeros will be explored in future work.

Reproducibility. The source code of the proposed method as well as an online demo are available at the following link: https://github.com/tinankh/ZERO

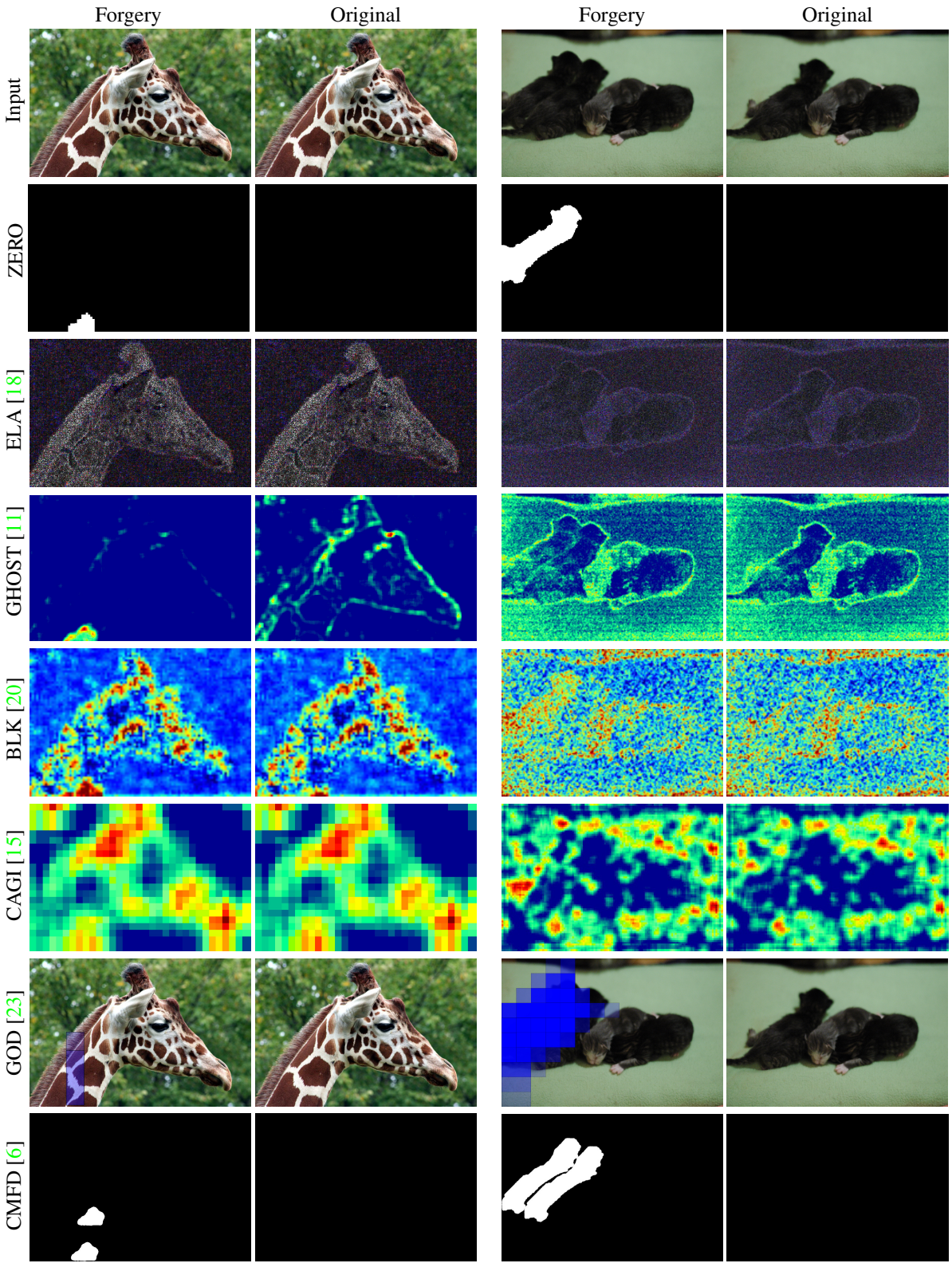

Figure 7: Results of the proposed method compared to the JPEG state-of-the-art methods. The first one produces a difference image, the three following heat maps and the last two masks. The methods are applied to the forged image and also its original source.

References

[1] Quentin Bammey, Rafael Grompone von Gioi, and Jean-Michel Morel. Automatic detection of demosaicing image artifacts and its use in tampering detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 424–429. IEEE, 2018. 4

[2] Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2444–2447. IEEE, 2011. 1, 2

[3] Yi-Lei Chen and Chiou-Ting Hsu. Image tampering detection by blocking periodicity analysis in jpeg compressed images. In *2008 IEEE 10th Workshop on Multimedia Signal Processing*, pages 803–808. IEEE, 2008. 1, 2

[4] Jim Chou, Matthew Crouse, and Kannan Ramchandran. A simple algorithm for removing blocking artifacts in block-transform coded images. In *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*, volume 1, pages 377–380. IEEE, 1998. 6

[5] Vincent Christlein, Christian Riess, Johannes Jordan, Corinna Riess, and Elli Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on information forensics and security*, 7(6):1841–1854, 2012. 7

[6] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Efficient dense-field copy–move forgery detection. *IEEE Transactions on Information Forensics and Security*, 10(11):2284–2297, 2015. 7, 8

[7] Axel Davy, Thibaud Ehret, Jean-Michel Morel, and Mauricio Delbracio. Reducing anomaly detection in images to detection in noise. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1058–1062. IEEE, 2018. 4

[8] A. Desolneux, L. Moisan, and J.-M. Morel. *From Gestalt Theory to Image Analysis*. Springer, 2008. 3, 4

[9] Thibaud Ehret. Automatic Detection of Internal Copy-Move Forgeries in Images. *Image Processing On Line*, 8:167–191, 2018. 7

[10] Zhigang Fan and Ricardo L De Queiroz. Identification of bitmap compression history: Jpeg detection and quantizer estimation. *IEEE Transactions on Image Processing*, 12(2):230–235, 2003. 1, 2

[11] Hany Farid. Exposing digital forgeries from jpeg ghosts. *IEEE transactions on information forensics and security*, 4(1):154–160, 2009. 1, 2, 8

[12] Hany Farid. *Photo Forensics*. The MIT Press, 2016. 1

[13] A. Gordon, G. Glazko, X. Qiu, and A. Yakovlev. Control of the mean number of false discoveries, bonferroni and stability of multiple testing. *Ann. Appl. Stat.*, 1:179–190, 2007. 4

[14] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2010. 4

[15] Chryssanthi Iakovidou, Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. Content-aware detection of jpeg grid inconsistencies for intuitive image forensics. *Journal of Visual Communication and Image Representation*, 54:155–170, 2018. 1, 2, 8

[16] Eric Kee, Micah K Johnson, and Hany Farid. Digital image authentication from jpeg headers. *IEEE Trans. Information Forensics and Security*, 6(3-2):1066–1075, 2011. 2

[17] Kodak. Kodak Lossless True Color Image Suite, Nov. 1999. 6, 7

[18] Neal Krawetz and Hacker Factor Solutions. A pictures worth... *Hacker Factor Solutions*, 6, 2007. 2, 8

[19] José Lezama, Rafael Grompone von Gioi, Gregory Randall, and Jean-Michel Morel. Finding vanishing points via point alignments in image primal and dual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 509–515, 2014. 4

[20] Weihai Li, Yuan Yuan, and Nenghai Yu. Passive detection of doctored jpeg image via block artifact grid extraction. *Signal Processing*, 89(9):1821–1829, 2009. 1, 2, 7, 8

[21] David G Lowe. Visual recognition from spatial correspondence and perceptual organization. In *IJCAI*, pages 953–959. Citeseer, 1985. 3, 4

[22] Tina Nikoukhah, Miguel Colom, Jean-Michel Morel, and Rafael Grompone von Gioi. Détection de grille JPEG par compression simulée. *preprint*, 2019. 2, 7

[23] Tina Nikoukhah, Rafael Grompone von Gioi, Miguel Colom, and Jean-Michel Morel. Automatic jpeg grid detection with controlled false alarms, and its image forensic applications. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 378–383. IEEE, 2018. 2, 4, 7, 8

[24] Gerald Schaefer and Michal Stich. Ucid: An uncompressed color image database. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, volume 5307, pages 472–481. International Society for Optics and Photonics, 2003. 6, 7

[25] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982. 5

[26] Denis Teyssou, Jean-Michel Leung, Evlampios Apostolidis, Konstantinos Apostolidis, Symeon Papadopoulos, Markos Zampoglou, Olga Papadopoulou, and Vasileios Mezaris. The invid plug-in: web video verification on the browser. In *Proceedings of the First International Workshop on Multimedia Verification*, pages 23–30. ACM, 2017. 1

[27] Gregory K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 1991. 2

[28] Andrew P Witkin and Jay M Tenenbaum. On the role of structure in vision. In *Human and machine vision*, pages 481–543. Elsevier, 1983. 3

[29] Shuiming Ye, Qibin Sun, and Ee-Chien Chang. Detecting digital image forgeries by measuring inconsistencies of blocking artifact. In *2007 IEEE International Conference on Multimedia and Expo*, pages 12–15. IEEE, 2007. 1, 2