IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2024–04–04.
Submitted on 2023–01–23, accepted on 2023–11–10.
ISSN 2105–1232 © 2024 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.2024.462

Image Forgery Detection Based on Noise Inspection: Analysis and Refinement of the Noisesniffer Method

Marina Gardella[1], Pablo Musé[2], Miguel Colom[1], Jean-Michel Morel[3]

[1]Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, F-91190 Gif-sur-Yvette, France
[2]IIE, Facultad de Ingeniería, Universidad de la República, Uruguay
[3]City University of Hong Kong
{marina.gardella, miguel.colom-barco}@ens-paris-saclay.fr,
pmuse@fing.edu.uy, jeamorel@cityu.edu.hk

Communicated by Rafael Grompone and Yanhao Li *Demo edited by* Marina Gardella and Pablo Musé

Abstract

Images undergo a complex processing chain from the moment light reaches the camera's sensor until the final digital image is delivered. Each of its operations leaves traces on the noise model which enable forgery detection through noise analysis. In this article, we describe the Noisesniffer method [Gardella et al., Noisesniffer: a Fully Automatic Image Forgery Detector Based on Noise Analysis, IEEE International Workshop on Biometrics and Forensics, 2021]. This method estimates for each image a background stochastic model which makes it possible to detect local noise anomalies characterized by their number of false alarms. We improve on the original formulation of the method by introducing a region-growing algorithm to detect local deviations from the background model. Results show that the proposed method outperforms the previous version as well as the state of the art.

Source Code

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

Keywords: image forensics; automatic forgery detection; noise residual

1 Introduction

In recent times, great concern has been raised about the authenticity of the images shared on social networks and used as evidence to support news. Although fake images are not a new issue, the ease with which it is possible to manipulate images today, as well as the quality of these fakes, have made this a highly relevant social problem. Visual inspection is not sufficient to detect tampering [12]: the average human accuracy when determining if an image is fake or not is only slightly better than

[1]https://doi.org/10.5201/ipol.2024.462

random chance [34]. The erosion of trust in images combined with this human inability to detect forgeries entails a growing need for quantitative and subjectivity-free detection algorithms.

Starting from the seminal work of H. Farid [13], digital forensics tools have been developed to provide scientific evidence to help determine the authenticity of the images under question [22]. Most classic forgery detection methods aim at detecting local inconsistencies of the traces left in the image by the camera processing chain. Such local disruptions hint at a local forgery. To do so, these methods strive to suppress image content and highlight intrinsic artifacts left by the forger. In the past few years, multi-purpose tools were proposed to detect inconsistencies from multiple traces simultaneously [8, 29, 37, 41]. However, results are not self-explanatory and the results of these methods are hard to justify. Furthermore, learning-based methods can be limited by the training data and may fail to generalize well in uncontrolled scenarios.

In this article, we analyze and improve the Noisesniffer method [14], an automatic forgery detector that chases traces of the image noise model disruptions. This method is based on the observation that, even though the camera processing chain modifies the initial raw Poisson-Gaussian noise, the final noise model should be coherent across the image. Local forgeries generally alter this coherence, creating an anomalous noise model in the tampered area [31]. Local anomalies of the noise model, though generally imperceptible to the human eye, can become informative cues for forensic analysis.

2 Related Work

The residual noise present in images depends on the in-camera processing chain. It can therefore reveal the presence of forged regions by detecting local inconsistencies in the noise statistics that are incompatible with a unique camera processing pipeline. Such inconsistencies can be produced by the forgery or its post-processing.

Blind noise-based detection methods usually estimate noise variance locally to detect suspicious regions and then apply a classification criterion to locate forgeries. Mahdian and Saic [28] propose a block-wise noise variance estimation using a median absolute deviation (MAD) estimator in the wavelet domain. Classification is performed using an homogeneous noise standard deviation criterion. In turn, Ke et al. [23] estimate the noise level using principal component analysis (PCA) [33]. K-means is then applied to group image blocks into two clusters. A similar approach was presented by Zeng et al. [40]. Lyu et al. [27] proposed a different method in which block-wise noise estimation is based on the observation that the kurtosis values across different band-passed filter channels are constant [43]. The method concludes by segmenting the image into regions with significantly different noise variances by k-means. Liu and Pun [25] use a different segmentation technique: the simple linear iterative clustering (SLIC) algorithm. Then, for each region, five filters are used to extract noise. The computed noise features are then used for classification, which is performed by energy-based graph cut.

The aforementioned methods share the same drawback: they estimate a single and constant noise level, namely an additive white Gaussian noise (AWGN) model. Yet this hypothesis does not hold in realistic scenarios where noise levels depend on the image intensity [26]. More recent methods consider this fact and estimate a noise level function (NLF) rather than a single noise level. Yao et al. [38], proposed to jointly estimate the NLF and the camera response function (CRF) by segmenting the image into edge and non-edge regions. Noise level functions are then compared and an empirical threshold is fixed to detect salient curves. The methods introduced by Julliand et al. [21, 20] instead analyze a histogram based on the noise density function at the local level in order to reveal suspicious areas. Pun et al. [32] compute an NLF-based on Wiener filtering. Local noise levels in regions with a certain brightness are assumed to follow a Poisson distribution, according to which, the larger the distance to the NLF, the higher the probability of forgery. On the other hand, the approach introduced by Zhu and Zhao [42] consists of estimating a noise level function that depends on the

local sharpness rather than on the intensity. Gardella et al. [16] suggest that, for JPEG-compressed images, not only the noise is intensity dependent but also scale-dependent. Thus, they propose to estimate an NLF at different scales to capture noise inconsistencies at multiple scales.

Recently, forgery detection methods based on deep learning and feature modeling have been developed. Splicebuster [7] proposes to use noise residuals to extract local features and compute their co-occurrence histograms, which are then classified in two classes using the Expectation–Maximization algorithm. More recently, the same authors presented a novel CNN-based method for noise residual extraction [8]. A similar approach was introduced by Mayer and Stamm [29, 15], where a pair of CNNs in a Siamese configuration is used to extract camera-related artifacts from pairs of patches which are then used to construct a similarity graph. Forgeries are detected as communities in the aforementioned graph. Zhou et al [41] proposed a two-stream CNN, one for the detection of tampering artifacts and the other to leverage noise features. Deep learning-based methods are more general than previously described ones. ManTraNet [37] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localize forgeries in the image. A major limitation of these methods is that they require large training datasets, which are not always available. Furthermore, performance generally remains dataset-dependent thus putting in doubt their generalization ability.

3 Method

Most non-parametric noise estimation methods share the same principles: they start by selecting the homogeneous regions of the image where noise dominates over the signal, and then they estimate noise in the frequency or the spatial domain using just a small portion of the blocks in the previously selected region [24]. In this work, we follow a similar procedure to identify the blocks within the homogeneous regions that are good candidates for noise estimation. However, instead of using these blocks to estimate the noise level, we are interested in analyzing the spatial distribution of such blocks.

Indeed, if the variance in the homogeneous regions identified in the first step is only explained by noise, the small proportion of blocks that are then used to estimate noise should be a random uniform selection over the homogeneous region. However, if we observe that the blocks used for noise estimation concentrate in a particular part of the homogeneous region, we can deduce that this zone exhibits a suspicious noise deficit.

The proposed method is developed in two steps. Firstly, we compute the set of blocks corresponding to the homogeneous regions and then the subset of those blocks having the lowest standard deviation. Secondly, the spatial distribution of these two sets is compared to detect if there is a statistically significant deviation from one distribution to the other. To do so, we adopt an a contrario approach [10] and assign a number of false alarms (NFA) to each detection.

3.1 Distributions Computation

The method takes as input an image I of size $X \times Y$, with C color channels. Given w, we first consider all its $w \times w$ overlapping blocks.

When the potential well in the camera sensor is full, the excess of photons does not contribute to the output voltage values and the pixel becomes saturated. Since noise is clipped at saturated pixels, saturated pixels may cause unreliable noise estimations. To avoid this situation, blocks having at least one saturated pixel are discarded. By doing so, we get a list of valid blocks.

When the light interacts with a detector, the photons arrive randomly and independently, resulting in fluctuations in the measured signal. These fluctuations are responsible for the noise at the first step of the image formation process to be signal-dependent. Therefore, to compare the noise

levels amongst different blocks, their intensity needs to be taken into account. For each channel, we group the valid blocks in bins with fixed size B according to their mean intensity.

In order to select the suitable regions for noise estimation, for each color channel c and bin b, we first compute the orthogonal Discrete Cosine Transform (DCT II) of each block in the bin, and for each block, we compute its variance in low and medium frequencies. A pair of frequencies (i, j) is said to correspond to a low or medium frequency if $0 < i + j < T$, where T is a fixed threshold that depends on the block size. Note that the pair $(i, j) = (0, 0)$ – the mean of the block term – is not considered [6].

Blocks are then sorted in ascending order according to their variance in low and medium frequencies, and only a small percentile given by the parameter n is kept. These blocks constitute the set L_b^c, where c denotes the corresponding color channel and b the corresponding bin. Since most of the energy corresponding to the image geometry is located at the low and medium frequencies, the selected blocks we keep correspond to the most homogeneous ones.

The intensity variations in these blocks are likely to be explained only by noise. The next step is to compute their variance (variance of the pixel values in the block) and order them in ascending order. The subset $V_b^c \subset L_b^c$ corresponds to the $m\%$ of the blocks in L_b^c having the lowest variance, including completely flat blocks (i.e. blocks whose variance is equal to zero). However, if more than the $m\%$ of the blocks in L_b^c are completely flat, the bin is declared invalid. This last precaution aims at preventing false detections due to strong image alterations introduced by the camera image processing pipeline. For instance, when applying strong JPEG compression, artificial flat zones may appear. Indeed, homogeneous regions usually have small DCT coefficients. Therefore, when the quantization factors are big, the JPEG encoder will set them to zero, resulting in flat zones.

Lastly, by aggregating the sets obtained for each color channel c and each valid bin b, we define

$$V = \bigcup_{c} \bigcup_{\text{valid } b} V_b^c \text{ and } L = \bigcup_{c} \bigcup_{\text{valid } b} L_b^c. \tag{1}$$

By construction, $V \subset L$.

3.2 Statistical Validation

Our null hypothesis (H_0) is the absence of any forgery. Under H_0, blocks in V and in L should have the same spatial distribution in the image. Indeed, the blocks in V should correspond to a random and uniform selection over the blocks in L. Nevertheless, when analyzing the distributions of V and L, small deviations from one to the other are likely to happen due to randomness. The fundamental question arises as to whether the observed spatial distribution is likely to happen by chance or not.

A criterion is needed in order to spot deviations that are statistically significant from those that could happen just by chance. Here, we propose to use an a contrario approach [11] to statistically validate these deviations. This theory is based on the Non-Accidentalness Principle, which states that we perceive a structure whenever a large deviation from randomness occurs [1]. However, computing the probability of these events might be hard. This problem is solved by the introduction of the Number of False Alarms (NFA) of an event, which is an upper bound on the expectation of occurrences of such event under the null model [11].

Given a region R to be tested for forgery, our null hypothesis H_0 implies that the blocks[2] in V correspond to a random uniform Poisson point process among the blocks in L. Suppose that for region R there are N blocks $o_1, \dots o_N$ in L, K out of which are also in V. We define the random

[2]In practice, we consider a block o to be in a certain region R if its origin is in R.

variables Z_i for $i = 1, \ldots, N$ as

$$Z_i = \begin{cases} 1 & \text{if } o_i \in V, \\ 0 & \text{if } o_i \notin V. \end{cases} \tag{2}$$

Since our null model is that the blocks in V correspond to a random and uniform selection over the blocks in L, under this hypothesis the variables Z_i follow a Bernoulli distribution with parameter p, for all $i = 1, \ldots N$. Since in each bin and in each channel exactly $m\%$ of the blocks in L_b^c are kept in V_b^c; globally exactly $m\%$ of the blocks of L are in V. Thus, we fix $p = m$.

Following the a contrario methodology [11], the number of false alarms (NFA) of the region R is defined as

$$\text{NFA}(R) = N_{\text{tests}} P_{H_0}(Z \geq K), \text{ where } Z = \sum_{i=1}^{N} Z_i. \tag{3}$$

The probability $P_{H_0}(Z \geq K)$ is difficult to compute directly because the random variables Z_i with $i = 1, \ldots, N$ are not independent. This is because the $w \times w$ blocks used to construct V and L are taken with overlap. We solve this problem by considering that we are making w^2 separate tests: one for each $w \times w$ grid without overlap and assuming that for each of these tests, we observe N/w^2 blocks in L and K/w^2 blocks in V. Then, the NFA of the region R is defined

$$\text{NFA}(R) = w^2 N_{\text{tests}} \mathcal{B}\left(\frac{K}{w^2}, \frac{N}{w^2}, m\right), \tag{4}$$

where N_{tests} is the number of tests to be detailed below and $\mathcal{B}$ denotes the tail of the binomial law

$$\mathcal{B}(k, n, p) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{5}$$

The expression in Equation (4) is, in fact, an upper bound of the actual NFA, since at least one of the grids will have more favorable parameters.

A region R is said to be ε-meaningful if $\text{NFA}(R) < \varepsilon$. Once ε is fixed, a region R is detected if it is ε-meaningful. This means that the expected number of regions to be declared ε-meaningful under H_0 is smaller than ε. Therefore, ε gives an *a priori* estimate of the mean number of false detections under H_0. For the rest of the article, the threshold ε is set to 1. Although we would normally require a mean number of false detections smaller than 1, due to the discrete nature of the binomial law, the average number of false detections is actually much smaller than the upper bound ε [18].

To complete the formulation we still need to specify the family of regions to be tested. Instead of using rectangular macro-blocks as in the original Noisesniffer formulation [14], we consider more general connected regions, as in [17]. With this aim, we consider a square tessellation of the image, with squares of size $l_\beta \times l_\beta$. Given a block $\beta(i, j)$ of this tilling, where (i, j) denotes the origin of the block, $\beta(i, j)$ is connected with its horizontal and vertical neighbor blocks, namely, to $\beta(i \pm l_\beta, j)$ and $\beta(i, j \pm l_\beta)$. The regions to be tested correspond to those that can be built under this 4-connectivity notion, using the squares of the tessellation as cells. These figures are called polyominoes. The exact number of polyominoes p_n of a given size n is – in general – not known. However, it can be approximated as [19]

$$p_n \approx 0.316915 \times \frac{4.062570^n}{n}. \tag{6}$$

Still, we need to consider that each polyomino can be placed at any position in the constructed square tiling. To consider all the possible placements, p_n needs to be multiplied by the number of

cell squares, namely $\frac{X}{l_\beta} \times \frac{Y}{l_\beta}$. This is not an exact calculation since it also considers some polyominoes that extend outside the image domain.

Finally, for a given region R of size $|R|$, where the size is the number of cells it contains, the number of tests can be written as:

$$N_{\text{tests}}^{|R|} = \left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right) p_{|R|}. \tag{7}$$

Note that we are interested in testing arbitrary-sized regions rather than fixed-sized regions. Hence, we need to distribute the weight of the NFA computed for each region size in such a way that, when computing the final NFA, it truly represents the NFA of the whole image where we have tested several region sizes. To do so, the number of tests is multiplied by the total number of possible region sizes. The region sizes we are interested in are those ranging from one only cell β to half of the image size $\frac{1}{2}\frac{X}{l_\beta} \times \frac{Y}{l_\beta}$, since we only consider forgeries up to this size. Therefore, we have $\frac{1}{2}\frac{X}{l_\beta} \times \frac{Y}{l_\beta}$ possible region sizes.

Then, the number of tests, when considering all the polyominoes sizes we are testing in the whole image is given by

$$N_{\text{tests}} = \frac{1}{2}\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right)\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right) p_{|R|}. \tag{8}$$

With this number of tested regions, the NFA of a candidate region is

$$\text{NFA}(R) = \frac{w^2}{2}\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right)^2 0.316915 \times \frac{4.062570^{|R|}}{|R|}\mathcal{B}\left(\frac{K}{w^2}, \frac{N}{w^2}, m\right). \tag{9}$$

3.3 Region Growing Algorithm

Testing all possible 4-connected regions is computationally intractable. Instead, we propose a heuristic approach to reduce the number of regions that will be evaluated using the a contrario approach described in the previous section. The construction of such candidate regions is based on the greedy algorithm proposed by Grompone et al. [17], and the modifications introduced in [35].

The construction can be summarized as follows: a first criterion is used to decide which cells are suspicious of being meaningful. Then, these cells are used as seeds to construct larger regions by iteratively adding connected cells satisfying the region growing criteria. Once the region stops growing, the NFA is computed. If a detection is made, the cells in the region are masked and will not start a new region in further iterations.

For a cell β to be used as a seed, we impose the following criteria. Let N_β be the number of observed blocks in L, of which K_β are also in V. If

$$\frac{K_\beta}{N_\beta} > m, \tag{10}$$

then β is a possible seed pixel for a new region. This criterion is based on the fact that the expected proportion of a Binomial law is equal to the probability of success on each Bernoulli experiment, which in our case is equal to m.

Starting from a seed cell, the region-growing algorithm iteratively adds neighbor cells that satisfy the region-growing criterion. To define this criterion, we follow the approach introduced by Tailanian et al. [35]. Namely, in order to decide if a cell is to be added or not to the region, we evaluate the NFA value of the region with and without this cell. If adding the cell makes the region more meaningful (i.e. if it lowers the NFA value), the cell is added.

The region-growing criterion can be stated as follows. Let R be a region and β a neighbor cell. Let N_R and N_β denote the number of observed blocks in L for R and β respectively, of which K_R and K_β are also in V. For the NFA of $R \cup \{\beta\}$ to be smaller than the NFA of R, the following condition must be met

$$\frac{4.062570}{|R|+1}\mathcal{B}\left(\frac{K_R+K_\beta}{w^2}, \frac{N_R+N_\beta}{w^2}, m\right) < \frac{1}{|R|}\mathcal{B}\left(\frac{K_R}{w^2}, \frac{N_R}{w^2}, m\right). \tag{11}$$

4 Detailed Implementation

As described in Section 3.1, the first step of the method consists in extracting the list of $w \times w$ overlapping valid blocks from image I. Valid blocks are those that do not have any saturated pixels. Here we adopt a relaxed notion of saturation: saturated pixels are those presenting the maximum value or the minimum value in at least one of the image channels. This notion is more robust to dynamic range changes than just checking pixels with intensities equal to 0 or 255. The implementation of this step is described in Algorithm 1.

Algorithm 1 Computes the indices of the valid blocks (i.e. not containing saturated pixels). (`ComputeValidBlocksIndices`)

Input I**:** image of size $X \times Y$ with 3 color-channels
Input w**:** block side
Output validBlocks**:** list of the indices of the valid blocks

$I_{\text{notsat}} \leftarrow (1)_{X \times Y}$ # initialize all pixels to 1 (not saturated)
for ch $\leftarrow 0$ **to** 2 **do**
  $I^{\max}_{\text{sat}} \leftarrow \left[I^{\text{ch}} < \max(I^{\text{ch}})\right]$ # mark as saturated pixels where the max in ch is achieved
  $I^{\min}_{\text{sat}} \leftarrow \left[I^{\text{ch}} > \min(I^{\text{ch}})\right]$ # mark as saturated pixels where the min in ch is achieved
  convert $I^{\max}_{\text{sat}}$ and $I^{\min}_{\text{sat}}$ to integers
  # update saturated pixels
  $I_{\text{notsat}} \leftarrow I_{\text{notsat}} \cdot I^{\max}_{\text{sat}} \cdot I^{\min}_{\text{sat}}$ # where $\cdot$ stands for the Hadamard product
K $\leftarrow (1)_{w \times w}$
$M \leftarrow [I_{\text{notsat}} * \text{K} > w^2 - 0.5]$ # where $*$ stands for convolution
  # the result is True only if all the pixels in the $w \times w$ block are not saturated
validBlocks $\leftarrow$ indices where M = True
return validBlocks

Secondly, the mean intensity of all the $w \times w$ overlapping blocks is computed channel-wise. Though the method only requires to compute the mean intensity of the valid blocks, computing the mean intensities of all the blocks can be efficiently done using a convolution, as described in Algorithm 2.

Algorithm 2 Computes a three dimensional array containing the means of all the $w \times w$ blocks in the image, in each color channel. (`AllImageMeans`)

Input I**:** image of size $X \times Y$ with 3 color-channels
Input w**:** block side
Output I_{means}**:** three dimensional array containing the means of all the $w \times w$ blocks in the image, in each color channel

K $\leftarrow \frac{1}{w^2}(1)_{w \times w}$ # define the averaging kernel of size $w \times w$
$I_{\text{means}} \leftarrow I * K$ # convolution between I and K
return I_{means}

After these two steps, the channel-wise processing starts. Given a color-channel ch, in order to define the bins, we first sort the valid blocks according to their mean intensity values, as shown in Algorithm 3.

Algorithm 3 Sorts the valid blocks according to their mean intensity in a given color channel ch. (`SortBlocksByMean`)

Input ch: color-channel
Input I_{means}: three-dimensional array containing the means of all the w × w blocks in the image
Input validBlocks: list of the indices of the valid blocks
Output sortedValidBlocks: list of valid blocks sorted in ascending order according to their mean intensity in channel ch.

```
sortedArgs ← argsort(I_means[validBlocks, ch])          # sort the blocks
sortedValidBlocks ← validBlocks[sortedArgs]
return sortedValidBlocks
```

Since it is unlikely that the number of valid blocks is a multiple of the number of samples in each bin, given by parameter B, we update this parameter in order to distribute the blocks in bins. This is done according to Algorithm 4.

Algorithm 4 Updates the number of samples per bin. (`UpdateSamplesPerBin`)

Input numBlocks: number of valid blocks.
Input B: number of samples per bin
Output $\tilde{B}$: updated number of samples per bin
Output numBins: number of bins

```
numBins ← round(numBlocks/B)
if numBins = 0 then
    numBins ← 1                                  # force to have at least one bin
B̃ ← ⌊numBlocks/numBins⌋
return B̃, numBins
```

Then for each bin in channel ch, the first step is to compute the list of blocks that corresponds to the bin b_k, for $k = 0, \ldots,$ `numBins` $- 1$. All the bins will have $\tilde{B}$ samples except for the last one.

Algorithm 5 Computes the list of blocks corresponding to the k-th bin. (`BinBlockList`)

Input k: bin.
Input $\tilde{B}$: updated number of samples per bin.
Input numBins: number of bins.
Input numBlocks: number of valid blocks.
Input sortedValidBlocks: list of valid blocks sorted according to their mean.
Output b_k: list of the blocks corresponding to the k-th bin.

```
if k = numBins − 1 then
    b_k ← sortedValidBlocks[(numBins − 1) × B̃, ..., numBlocks]
else
    b_k ← sortedValidBlocks[k × B̃, ..., (k + 1) × B̃]
return b_k
```

Once the list of blocks corresponding to a certain bin k is computed, we then compute their DCT II. Next, a mask of size $w \times w$ corresponding to the low and medium frequencies is computed, according to Algorithm 6. The thresholds used correspond to those in [6].

Algorithm 6 Computes a mask of size $w \times w$ that corresponds to low-medium frequencies. (`GetTMask`)

Input w: block side, $w \in \{3, 5, 8\}$.
Output mask: mask of size $w \times w$ that corresponds to low-medium frequencies.

```
if w = 3 then
    T ← 3
if w = 5 then
    T ← 5
if w = 8 then
    T ← 9
mask ← (0)_{w×w}                          # define as a zero matrix of size w × w
for i ← 0 to w − 1 do
    for j ← 0 to w − 1 do
        if i + j ≠ 0 and i + j < T then
            mask(i, j) ← 1
return mask
```

Then, the low-medium frequency energy of each DCT block is computed according to Algorithm 7. For commodity, we use here and afterward the term "variance" to refer to this energy. Indeed, since the expectation is zero, this sum is in fact proportional to the empirical variance.

Algorithm 7 Computes the variance in low-medium frequencies of the DCT II of a $w \times w$ block. (`ComputeLowFreqVar`)

Input D: a DCT block of size $w \times w$.
Input mask: a mask corresponding to low and medium frequencies. `# see Algorithm 6`
Output $\sigma^2_{\text{low-med}}$: variance in low-medium frequencies of the DCT II block.

```
D_low-med ← D · mask                      # keep only the low-med frequencies of the block D
σ²_low-med ← Σ_{i,j=0}^{w−1} D_low-med(i, j)²
return σ²_low-med
```

Afterwards, a small percentile[3], given by parameter n, of the blocks having the lowest low-medium frequency variance is selected. The selection procedure is summarized in Algorithm 8.

[3]Throughout this article we refer to percentiles and percentages indistinctly.

Algorithm 8 Selects the percentile n of blocks in the bin having the lowest variances in low-medium frequencies. (`SelectBlocksVL`)

Input $\tilde{B}$**:** updated number of samples per bin. `# see Algorithm 4`
Input b_k**:** list of the blocks in the bin. `# see Algorithm 5`
Input $\sigma^2_{\text{low-med}}$**:** list of the variances in low-medium frequencies of the blocks. `# see Algorithm 7`
Input n**:** a percentile.
Output $b_k^{\text{low-med}}$**:** list of the $n \times \tilde{B}$ blocks in the bin having the lowest variances in low-medium frequencies.

1 $N \leftarrow n \times \tilde{B}$ `# number of blocks that correspond to the` n `percentile`
2 sortedArgs $\leftarrow$ `argsort`$(\sigma^2_{\text{low-med}})$
3 $b_k^{\text{sorted}} \leftarrow b_k[\text{sortedArgs}]$
4 $b_k^{\text{low-med}} \leftarrow b_k^{\text{sorted}}[0, \ldots, N]$
5 **return** $b_k^{\text{low-med}}$

Then, the standard deviation of the selected blocks in the bin is computed. These standard deviations are analyzed as described in Algorithm 9 to check if the bin is valid or not.

Algorithm 9 Determines if a bin is valid or not. (`BinIsValid`)

Input $\tilde{B}$**:** updated number of samples per bin. `# see Algorithm 4`
Input σ**:** list of the standard deviations of the blocks in the bin.
Input n**:** a percentile.
Input m**:** a percentile.
Output binValid**:** boolean variable, `True` if bin is valid, `False` if not.

1 $M \leftarrow \tilde{B} \times n \times m$
2 binValid $\leftarrow [\texttt{count}(\sigma = 0) < M]$
3 **return** binValid

If the bin is valid, then the m percentile of the blocks having the lowest low-medium frequency variance (selected in Algorithm 8) is kept. The selection procedure is summarized in Algorithm 10.

Algorithm 10 Selects the percentile m of blocks having the lowest standard deviations amongst those having the lowest low-medium frequency variance. (`SelectBlocksStds`)

Input $\tilde{B}$**:** updated number of samples per bin. `# see Algorithm 4`
Input $b^{\text{low-med}}$**:** list of the blocks with the lowest low-med frequency variance. `# see Algorithm 8`
Input σ**:** list of the standard deviation of the blocks having the lowest low-med frequency variance.
Input n**:** percentile of blocks with the lowest energy in low frequencies
Input m**:** percentile of blocks with the lowest standard deviation
Output b^{σ}**:** list of the $m \times n \times \tilde{B}$ blocks in the bin having the lowest variances in low-medium frequencies.

1 $M \leftarrow m \times n \times \tilde{B}$ `# number of blocks that correspond to the` m `percentile`
2 sortedArgs $\leftarrow$ `argsort`(σ)
3 $b_{\text{sorted}}^{\text{low-med}} \leftarrow b^{\text{low-med}}[\text{sortedArgs}]$
4 $b^{\sigma} \leftarrow b_{\text{sorted}}^{\text{low-med}}[0, \ldots, M]$
5 **return** b^{σ}

Finally, if the bin is valid, the n percentile of blocks selected in Algorithm 8 is added to the set L

and the m percentile amongst them having the lowest standard deviation selected in Algorithm 10 is added to the set V.

Algorithm 11 Overview of the distributions computation (Section 3.1)

Input I**:** image of size $X \times Y$ with 3 color-channels
Input n**:** percentile of blocks with the lowest energy in low frequencies
Input m**:** percentile of blocks with the lowest standard deviation
Param w**:** block side
Param B**:** number of samples per bin
Output V**:** set of blocks in the n-th percentile having the lowest energy in low frequencies
Output L**:** subset of V of blocks in the m-th percentile having the lowest standard deviation

validBlocks $\leftarrow$ `ComputeValidBlocksIndices`(I, w)
$I_{\text{means}} \leftarrow$ `AllImageMeans`(I, w)
$V \leftarrow \emptyset$
$L \leftarrow \emptyset$
for ch $= 0$ to 2 **do**
    $V_{\text{ch}} \leftarrow \emptyset$
    $L_{\text{ch}} \leftarrow \emptyset$
    sortedValidBlocks $\leftarrow$ `SortBlocksByMean`(ch, I_{means}, validBlocks)
    $\tilde{B} \leftarrow$ `UpdateSamplesPerBin`(`len`(validBlocks), B)
    numBlocks $\leftarrow$ `len`(validBlocks)
    numBins $\leftarrow$ `round`(numBlocks/$\tilde{B}$)
    for $k = 0$ to numBins- 1 **do**
        $b_k \leftarrow$ `BinBlockList`$(b, \tilde{B}$, numBins, sortedValidBlocks$)$
        mask $\leftarrow$ `GetTMask`(w)
        $\sigma^2_{\text{low-med}} \leftarrow$ `ComputeLowFreqVar`(DCT(block), mask) for all block in b_k
        $b_k^{\text{low-med}} \leftarrow$ `SelectBlocksVL`$(\tilde{B}, b_k, \sigma^2_{\text{low-med}}, n)$
        $\sigma \leftarrow$ [`std`(block) for all block in b_k]
        if BinIsValid$(\tilde{B}, \sigma, n, m)$ **then**
            $b_k^{\sigma} \leftarrow$ `SelectBlocksStds`$(\tilde{B}, b_k^{\text{low-med}}, \sigma, n, m)$
            append $b_k^{\text{low-med}}$ to L_{ch}
            append b_k^{σ} to V_{ch}
        append L_{ch} to L
        append V_{ch} to V
return L, V

An overview of the distributions computation is provided in Algorithm 11. Once these distributions are computed, they go through the statistical validation process (Section 3.2) to spot significant deviations from one another. The NFA computation is described in Algorithm 12.

Algorithm 12 NFA computation

Input I: image of size $X \times Y$ with 3 color-channels
Input w: block side
Input l_β: block side
Input m: percentile of blocks having the lowest standard deviation
Output V: set of blocks in the n-th percentile having the lowest energy in low frequencies
Output L: subset of V of blocks in the m-th percentile having the lowest standard deviation
Output Mask: forgery detection mask

mask $\leftarrow (0)_{X \times Y}$
$\varepsilon \leftarrow 1$
cells $\leftarrow$ list of all the $l_\beta \times l_\beta$ non-overlapping blocks in I
for *each* cell $\in$ cells **do**
  if cell satisfies seed criteria (Equation(10)) **then**
    $R \leftarrow$ [cell]
    $R_{\text{init}} \leftarrow 1$
    $R_{\text{fin}} \leftarrow 0$
    $N_R \leftarrow$ number of $L-$blocks in R
    $K_R \leftarrow$ number of $V-$blocks in R
    while $R_{init} \neq R_{fin}$ **do**
      $R_{\text{init}} \leftarrow \text{len}(R)$
      $R_{\text{fin}} = R_{\text{init}}$
      neighbour cells $\leftarrow$ list of cells having a common edge to any element in R
      for *each* neighbour $\in$ neighbour cells **do**
        if neighbour $\notin R$ **and** neighbour satisfies growing criteria (Equation(11)) **then**
          append neighbour to R
          $N_{\text{neighbour}} \leftarrow$ number of $L-$blocks in neighbour
          $K_{\text{neighbour}} \leftarrow$ number of $V-$blocks in neighbour
          $N_R \leftarrow N_R + N_{\text{neighbour}}$
          $K_R \leftarrow K_R + K_{\text{neighbour}}$
          $R_{\text{fin}} \leftarrow R_{\text{fin}} + 1$
    $\text{NFA}_R \leftarrow \frac{w^2}{2}\left(\frac{X}{l_\beta} \times \frac{Y}{l_\beta}\right)^2 0.316915 \times \frac{4.062570^{R_{\text{fin}}}}{R_{\text{fin}}} \mathcal{B}\left(\lfloor \frac{K_R}{w^2} \rfloor, \lceil \frac{N_R}{w^2} \rceil, m\right)$
    if $NFA_R < \varepsilon$ **then**
      mask$[R] \leftarrow 1$
return mask

5 Experiments

We used the Trace database [3] to evaluate the method. The Trace database is made of 1000 images taken from the Raise dataset [9]. For each image, two forgery masks are made: the *endomask*, obtained by taking a random object from the image's automatic segmentation, and the *exomask*, which is simply the endomask of another image of the set and thus does not correlate to the contents of the image. The concept of the database is to process each raw image with two different pipelines, and splice both processed images according to the forgery masks. Of the six datasets that are proposed, only one is of interest to us: the raw noise level dataset. In this dataset, the two pipelines are the same except for the initial raw noise level model.

In this dataset, noise is added to each raw image before processing it. The noise variance is modeled as $\sigma^2 = A + Bu$, where A and B are constants and u is the noiseless image. Given a base image, the authors sort two different pairs of constants (A_0, B_0) and (A_1, B_1) that determine the two different noise models. Both images - which are in fact the same but having a different noise model - are then processed with the same subsequent pipeline. **Since the proposed method is only able to detect forgeries having lower noise levels, we restrict the analysis to those images**. This sums to 499 images in each, the endomasks and the exomasks datasets. The endomasks dataset is used to find the optimal parameter configuration and to analyze the influence of each parameter on the performance (Section 5.1). The comparison with other state-of-the-art methods (Section 5.2) was performed on the exomasks dataset, to avoid any kind of overfitting.

To assess the quantitative performance of the different parameters configurations and methods, we used three metrics: the Matthews Correlation Coefficient (MCC), the Intersection over Union score (IoU), and the F1 score, defined by

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}, \tag{12}$$

$$IoU = \frac{TP}{TP + FN + FP}, \tag{13}$$

$$F1 = \frac{2TP}{2TP + FN + FP}, \tag{14}$$

where TP, FP, TN and FN denote the pixel-wise number of true positives, false positives, true negatives, and false negatives, respectively. The IoU and F1 scores vary between 0 and 1. The result is 0 when no true positives are detected and 1 for perfect detection. The MCC score varies from -1 for a detection that is complementary to the ground truth, to 1 for a perfect detection. A score of 0 represents an uninformative result and is the expected performance of any random classifier. The MCC is more representative than the F1 and IoU scores [4, 5], particularly because it is less dependent on the proportion of positives in the ground truth.

The scores were computed for each image and then averaged over each dataset. As most surveyed methods do not provide a binary output but a heat map, to adapt the metrics to the continuous setting, we used their weighted version. In this approach, the value of a heat map H at each pixel x is regarded as the probability of forgery of the pixel. Therefore, given the ground truth mask M, we define the *weighted true positives*, *weighted false positives*, *weighted true negatives* and *weighted false negatives* as

$$TP_w = \sum_x H(x) \cdot M(x), \tag{15}$$

$$FP_w = \sum_x (1 - H(x)) \cdot M(x), \tag{16}$$

$$TN_w = \sum_x (1 - H(x)) \cdot (1 - M(x)), \tag{17}$$

$$FN_w = \sum_x H(x) \cdot (1 - M(x)), \tag{18}$$

where $\cdot$ stands for the Hadamard product.

5.1 Impact of the Parameters in the Detection Performance

The method has the following parameters:

w: the side length of the blocks used for noise analysis,

B: the number of samples per bin,

n: the percentile of blocks having the lowest variance in low-medium frequencies that are stored in L,

m: the percentile of blocks with the lowest standard deviation amongst those having the lowest variance in low-medium frequencies that are stored in V,

l_β: the side of the cells used in the NFA computation.

We tested several values for each parameter. For w we tested the values 3, 5 and 8, for B, 20000 and 40000, for n, 0.1, 0.05 and 0.01, for m, 0.5, 0.4, 0.3, 0.2 and 0.1, and for l_β, 60, 80 and 100. The best parameter configurations are given in Table 1.

Parameters	MCC	F1	IoU
$w = 3$, $B = 20000$, $n = 0.1$, $m = 0.5$, $l_\beta = 100$	0.3572 (2)	0.3716 (2)	**0.2723** (1)
$w = 3$, $B = 40000$, $n = 0.1$, $m = 0.5$, $l_\beta = 100$	0.3570 (3)	**0.3717** (1)	0.2719 (2)
$w = 3$, $B = 40000$, $n = 0.1$, $m = 0.5$, $l_\beta = 60$	**0.3640** (1)	0.3618 (3)	0.2647 (3)

Table 1: Best parameter configurations obtained for the tested values.

We set the optimal parameter configuration to $w = 3$, $b = 20000$, $n = 0.1$, $m = 0.5$ and $l_\beta = 100$, as it delivers the best results when giving the same weight to each of the three metrics. To analyze the individual performance of each parameter, we set the rest of them to the optimal value and only vary the parameter under investigation.

5.1.1 Impact of the Block Size of the Blocks Used for Noise Analysis w

While keeping the rest of the parameters to their optimal value, we tested three possible values for w: 3, 5, and 8. Setting w smaller than 3 would imply too few samples are available to estimate the variance in low and medium frequencies. The results in terms of performance metrics are shown in Table 2. We observe that the size of the blocks w highly impacts the performance of the method, being 3 the optimal value among the tested ones.

w	MCC	F1	IoU
3	0.3572	0.3716	0.2723
5	0.2837	0.2879	0.2013
8	0.2039	0.2004	0.1356

Table 2: Scores obtained on the endomasks noise level dataset from the Trace database [3] when setting w to 3, 5 and 8, while keeping the rest of the parameters to their optimal value.

Figure 1 shows an example of the results obtained when setting w to 3, 5 and 8. Since the forgery is a textured zone, finding homogeneous blocks inside the forgery is more difficult when using bigger blocks, as the distributions show. Therefore, the detection featured when setting w to 3 achieves better results. Indeed, when setting w to 3 the MCC score is 0.9322, the $F1$ score 0.9444, and the IoU score 0.8947 while the corresponding scores when setting w to 5 and 8 are 0.9275, 0.9404, 0.8875 and 0.8730, 0.8965, 0.81239, respectively.

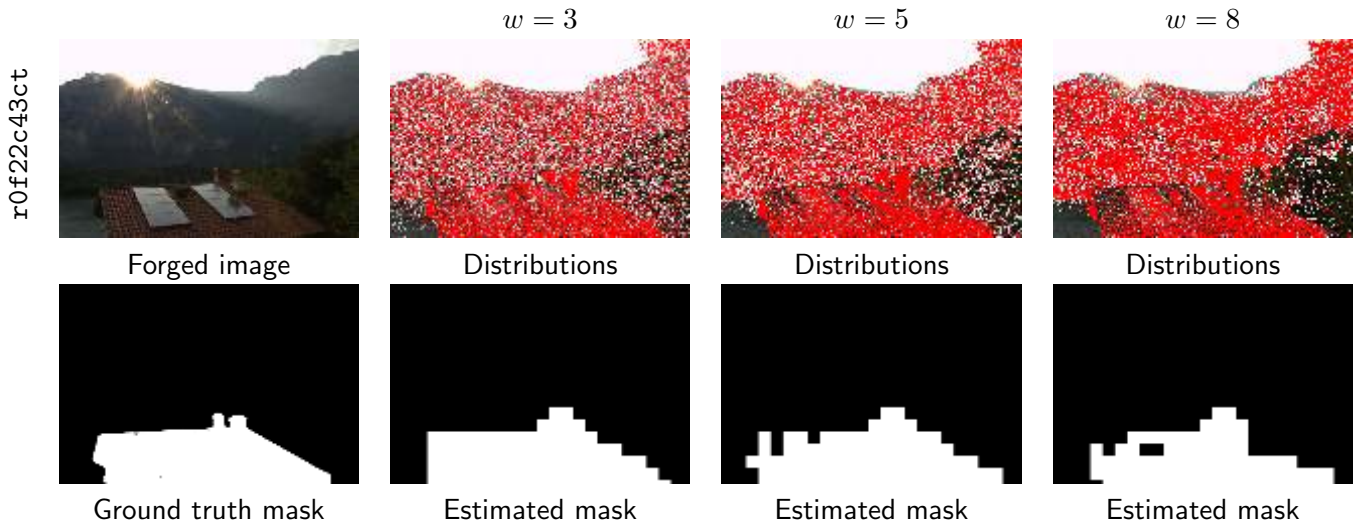


Figure 1: Results obtained on image `r0f22c43ct` from the Trace dataset [3] when setting w to 3, 5, and 8, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red.

5.1.2 Impact of the Number of Blocks per Bin B

While keeping the rest of the parameters to their optimal value, we tested two possible values for B: 20000 and 40000. The results in terms of performance metrics are shown in Table 3. We observe that the number of the blocks B has very little influence on the performance.

Indeed, as Figure 2 shows, the difference between the results obtained when using bins having 40000 or 20000 samples is very subtle, if any. In this example case, when setting $B = 20000$ the method achieves the scores $MCC = 0.8546$, $F1 = 0.8577$, and $IoU = 0.7509$, while when setting $B = 40000$ the method presents the following scores $MCC = 0.8440$, $F1 = 0.8493$, and $IoU = 0.7381$.

B	MCC	F1	IoU
20000	0.3572	0.3716	0.2723
40000	0.350	0.3717	0.2719

Table 3: Scores obtained on endomasks noise level dataset from the Trace database [3] when setting B to 40000 and 20000, while keeping the rest of the parameters to their optimal value.

5.1.3 Impact of the Percentile n

While keeping the rest of the parameters to their optimal value, we tested three possible values for n: 0.1, 0.05, and 0.01. The results in terms of performance metrics are shown in Table 4. We observe that the value of the parameter n has a strong influence on the performance of the method. We set 0.1 as an upper bound for n since bigger values could lead to false positives in textured images. Indeed, our method assumes that the n percentile of the total number of $w \times w$ blocks extracted from the image are homogeneous.

Figure 3 depicts an example of the different outputs that can be obtained with the method when varying n. When setting $n = 0.1$ the achieved scores are $MCC = 0.7064$, $F1 = 0.7154$ and $IoU = 0.5569$, when setting $n = 0.05$, the scores are $MCC = 0.6564$, $F1 = 0.6675$, $IoU = 0.5009$, finally, when setting $n = 0.01$, the scores are $MCC = 0.5070$, $F1 = 0.5240$, $IoU = 0.3550$.

Regarding the distributions in Figure 3, we observe that when setting n to 0.1, the concentration of the blocks in V (painted in red) in the forged zone becomes more evident than in the rest of the

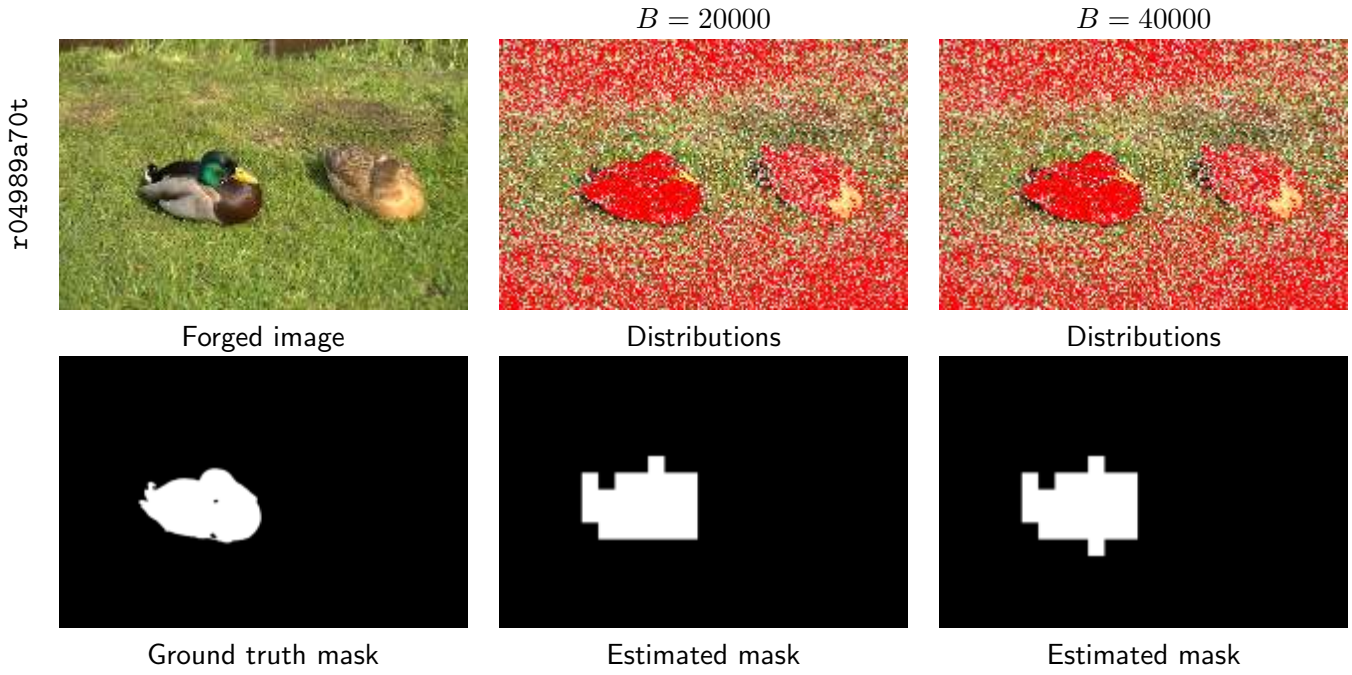


Figure 2: Results obtained on image `r04989a70t` from the Trace dataset [3] when setting B to 40000 and 20000, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. The detection featured when setting $B = 20000$ achieves the scores $MCC = 0.8546$, $F1 = 0.8577$, and $IoU = 0.7509$, while the one featured when setting $B = 40000$ presents the following scores $MCC = 0.8440$, $F1 = 0.8493$, and $IoU = 0.7381$.

cases. As a consequence, the statistical validation step detects more accurately the deviant statistics of the region.

n	MCC	F1	IoU
0.1	0.3572	0.3716	0.2723
0.05	0.3372	0.3382	0.2437
0.01	0.2020	0.1864	0.1295

Table 4: Scores obtained on endomasks noise level dataset from the Trace database [3] when setting n to 0.1, 0.05, and 0.01, while keeping the rest of the parameters to their optimal value.

5.1.4 Impact of the Percentile m

While keeping the rest of the parameters to their optimal value, we tested five possible values for m: 0.1, 0.2, 0.3, 0.4, and 0.5. The results in terms of performance metrics are shown in Table 5. The percentile m has a mild but non-negligible influence on the performance of the method.

Figure 4 shows an example of the different outputs that can be obtained with the method when varying m. For simplicity, we only display the results for m equal to 0.5, 0.3 and 0.1. When setting $m = 0.5$ the achieved scores are $MCC = 0.7791$, $F1 = 0.8040$ and $IoU = 0.6722$, when setting $m = 0.3$, the scores are $MCC = 0.7344$, $F1 = 0.7643$ and $IoU = 0.6185$, finally, when setting $m = 0.1$, the scores are $MCC = 0.7732$, $F1 = 0.7987$ and $IoU = 0.6648$.

Regarding the distributions depicted in Figure 4, we observe that the concentration of blocks in V (painted in red) becomes more evident when setting $m = 0.5$. This makes the detection of the deviant statistics of the region more accurate, as can be seen from the estimated masks. Still, the optimal value of m depends on the size of the forgery: small forgeries are more easily spotted with

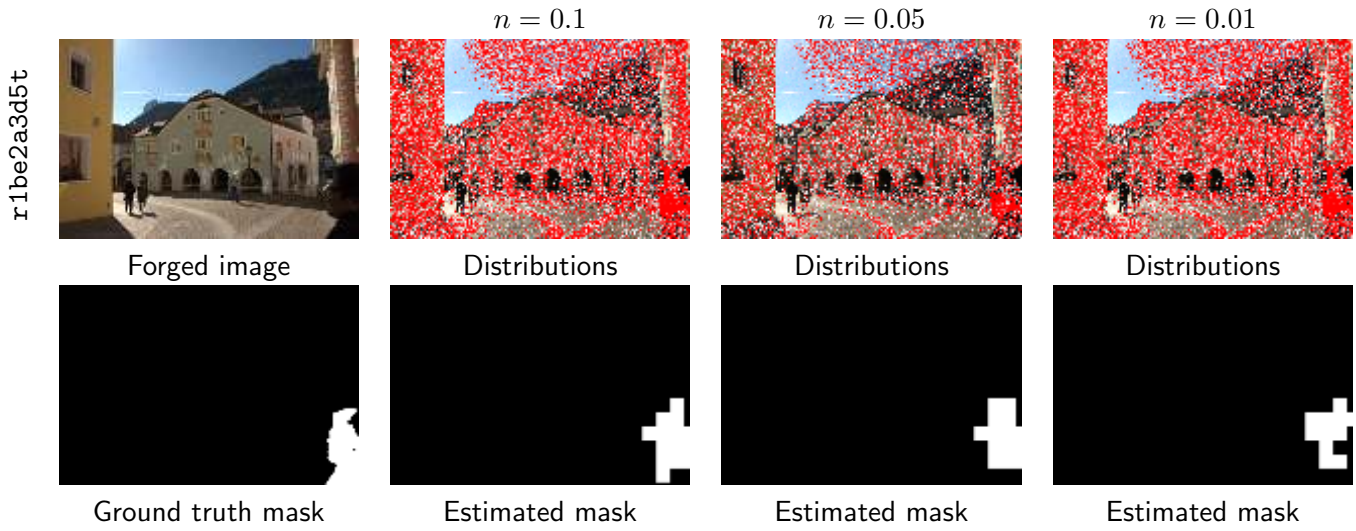


Figure 3: Results obtained on image `r1be2a3d5t` from the Trace dataset [3] when setting n to 0.1, 0.05 and 0.01, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. When setting $n = 0.1$ the achieved scores are $MCC = 0.7064$, $F1 = 0.7154$ and $IoU = 0.5569$, when setting $n = 0.05$, the scores are $MCC = 0.6564$, $F1 = 0.6675$, $IoU = 0.5009$, finally, when setting $n = 0.01$, the scores are $MCC = 0.5070$, $F1 = 0.5240$, $IoU = 0.3550$.

small m values while large forgeries, such as the one in the example, are better spotted with big values of m.

m	MCC	F1	IoU
0.5	0.3572	0.3716	0.2723
0.4	0.3540	0.3716	0.2723
0.3	0.3498	0.3600	0.2607
0.2	0.3388	0.3437	0.2466
0.1	0.3166	0.3155	0.2234

Table 5: Scores obtained on the endomasks noise level dataset from the Trace database [3] when setting m to 0.5, 0.4, 0.3, 0.2 and 0.1, while keeping the rest of the parameters to their optimal value.

5.1.5 Impact of the Size of the Cells l_β

While keeping the rest of the parameters to their optimal value, we tested three possible values for l_β: 60, 80, 100. The results in terms of performance metrics are shown in Table 6.

The parameter l_β has no influence in the distributions computation but it influences the statistical validation step. As can be seen in Table 6, the influence of this parameter on the overall performance is not critical. Furthermore, the evidence is not conclusive regarding the optimal value for l_β. Indeed, bigger values deliver higher $F1$ and IoU scores while smaller values deliver better MCC scores.

Figure 5 shows an example of the different estimated masks that can be obtained when varying the parameter l_β. Larger zones are detected when setting larger values for l_β. On the other hand, smaller values for l_β seem to capture more accurately the shapes. In terms of scores, the detection featured when setting $l_\beta = 100$ are $MCC = 0.6460$, $F1 = 0.7116$ and $IoU = 0.5523$, when setting $l_\beta = 80$, $MCC = 0.6553$, $F1 = 0.7107$ and $IoU = 0.5512$ and when setting $l_\beta = 60$, $MCC = 0.6473$, $F1 = 0.6945$, $IoU = 0.5320$.

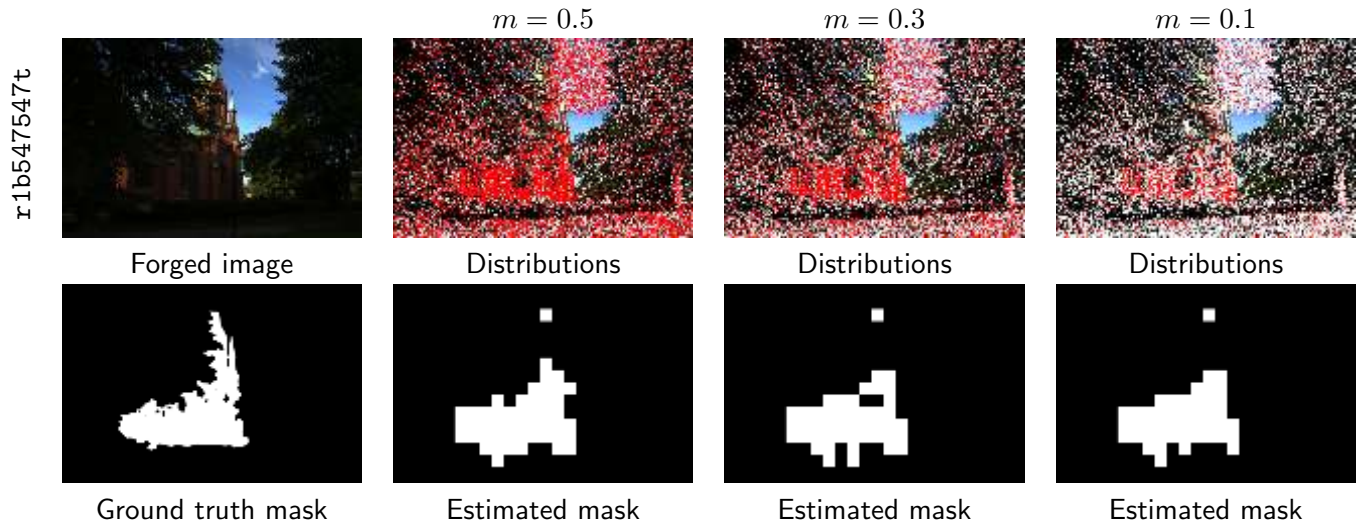


Figure 4: Results obtained on image `r1b547547t` from the Trace dataset [3] when setting m to 0.3, 0.2, and 0.1, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. When setting $m = 0.5$ the achieved scores are $MCC = 0.7791$, $F1 = 0.8040$ and $IoU = 0.6722$, when setting $m = 0.3$, the scores are $MCC = 0.7344$, $F1 = 0.7643$ and $IoU = 0.6185$, finally, when setting $m = 0.1$, the scores are $MCC = 0.7732$, $F1 = 0.7987$ and $IoU = 0.6648$.

l_β	MCC	F1	IoU
100	0.3572	0.3716	0.2723
80	0.3596	0.3679	0.2705
60	0.3623	0.3604	0.2635

Table 6: Scores obtained on the endomasks noise level dataset from the Trace database [3] when setting l_β to 100, 80, and 60, while keeping the rest of the parameters to their optimal value.

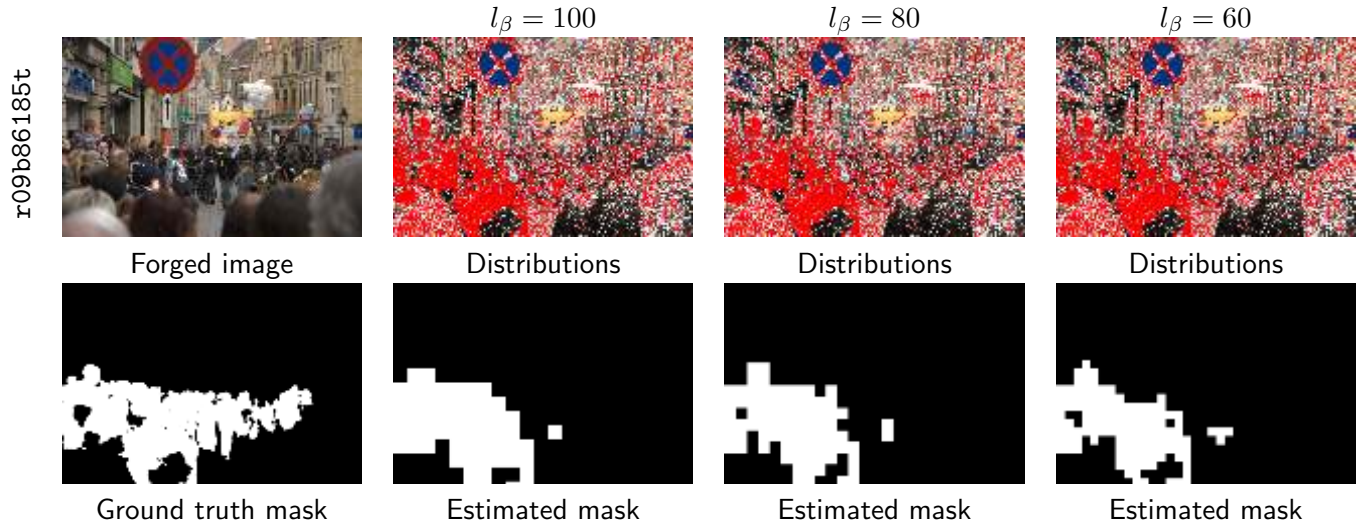


Figure 5: Results obtained on image `r09b86185t` from the Trace dataset [3] when setting l_β to 100, 80, and 60, while keeping the rest of the parameters to their optimal value. The distribution images show the blocks in L painted in white and, on top, those in V painted in red. In the three cases, the distributions are the same since the parameter l_β does not affect this step. In terms of scores, the detection featured when setting $l_\beta = 100$ are $MCC = 0.6460$, $F1 = 0.7116$ and $IoU = 0.5523$, when setting $l_\beta = 80$, $MCC = 0.6553$, $F1 = 0.7107$ and $IoU = 0.5512$ and when setting $l_\beta = 60$, $MCC = 0.6473$, $F1 = 0.6945$, $IoU = 0.5320$.

5.2 Comparison with the State of the Art

We used the exomasks dataset to conduct an evaluation comparing the performance of the proposed method (which we shall refer to as Noisesniffer+) to other relevant ones. Among them we compared to classic methods detecting noise traces: the previous formulation of Noisesniffer [14], Lyu [27, 39] and Mahdian [28, 39]; as well as to generic methods Splicebuster [7], Noiseprint [8] and ManTraNet [37, 2]. For each of these methods, we list the source code used for evaluation in Table 7.

Method	Ref.	Source Code
Noisesniffer	[14]	https://github.com/marinagardella/Noisesniffer
Mahdian	[28, 39]	https://github.com/MKLab-ITI/image-forensics
Pan	[30, 39]	https://github.com/MKLab-ITI/image-forensics
Splicebuster	[7]	http://www.grip.unina.it/research/83-multimedia_forensics
Noiseprint	[8]	http://www.grip.unina.it/research/83-multimedia_forensics
ManTraNet	[37, 2]	https://www.ipol.im/pub/art/2022/431/

Table 7: State-of-the-art methods used for the comparison as well as their reference and link to source code.

In the case of the Lyu and Mahdian algorithms, a specific transformation is needed to score their results. These methods do not act directly as detectors, but rather locally estimate and output the noise level. To turn their outputs into a heat map detection, we computed and normalized the distance of the output to its median. By doing so, we allow these methods to detect regions having noise levels that are far from the median noise level of the image. These regions can have noise deficit or noise excess. The MCC, $F1$ and IoU scores obtained by each of the analyzed methods are presented in Table 8.

Method	MCC	F1	IoU
Noisesniffer+	0.502	0.512	0.395
Noisesniffer [14]	0.285	0.274	0.181
Lyu [27, 39]	0.011	0.114	0.066
Mahdian [28, 39]	0.0331	0.114	0.067
Splicebuster [7]	0.178	0.188	0.116
Noiseprint [8]	0.133	0.174	0.106
ManTraNet [37, 2]	0.062	0.110	0.063

Table 8: Results of different state-of-the-art forensics tools on the exomasks noise level dataset from the Trace database [3].

Firstly, when comparing the performance of Noisesniffer+ in the exomasks dataset (Table 8) to the one in the endomasks dataset (Section 5.1), we observe that the performance is better when using exomasks. This can be explained by considering that, when using semantically generated masks, the forged zone usually has a homogeneous intensity while when using exogeneous masks, forgeries have more heterogeneous intensities. Since the method performs intra-bin comparisons, the wider range of intensities benefits the detection of the manipulated zone since more bins are involved. This phenomenon has been already reported by Bammey et al. [3] for Noisesniffer.

Noisesniffer+ outperforms all the evaluated methods. The region growing algorithm presented here, as well as the parameter optimization, doubles the scores obtained by the previous Noisesniffer method [14]. After Noisesniffer, the best-performing methods are Splicebuster and Noiseprint. Splicebuster presents better scores than Noiseprint when using the three proposed metrics. However, the difference is minor for the $F1$ and IoU scores. ManTraNet presents a higher MCC score than the Mahdian and Lyu algorithms. However, their order is inverted for the $F1$ or IoU scores, where the Mahdian and Lyu methods perform equivalently and outperform ManTraNet.

The observed difference in the ranking when considering the MCC score or the $F1$ or IoU scores can be explained by the definition of the scores. The $F1$ and IoU scores neglect the effectiveness of the methods in classifying negative samples. Indeed, their definition excludes the true negatives. On the other hand, the MCC score treats the positive and the negative classes equivalently. Hence, true negatives are as important as true positives. From this analysis, we should expect Splicebuster to have similar detections as Noiseprint but more true negatives. The same applies to ManTraNet, Lyu and Mahdian.

Figure 6 shows examples of images where Noisesniffer+ outperforms the other methods. Firstly, we observe that Noisesniffer+ provides more accurate masks than its previous version. For image `r17ad56act` we observe that only a few of the methods timidly highlight the true forgery, except for Noisesniffer+. In most cases, methods highlight the near-saturated regions of the image. On image `r1c5ec853t`, Lyu and Mantranet output very noisy heatmaps, due to the textures of the image. Mahdian, Noiseprint and Splicebuster provide very partial detections of the forgery. As for image `r1bf00696t`, besides Noisesniffer+, only Lyu seems to spot a different behavior on the forged region. Still, their output is tainted with the textures of the image. A similar phenomenon happens for image `r1ea8ccbbt`.

Figure 7 shows examples of images where Noisesniffer+ is outperformed by the other methods. For these images, we observe that Noisesniffer+ outputs partial detections (`r10cf67d1t`, `r0fd69c12t` and `r141665bdt`) or less accurate detections (`r059cae86t`). For a detailed analysis of the causes of such limitations, see Section 5.3.

For image `r10cf67d1t`, Noisesniffer+ and Noisesniffer are able to detect just a portion of the forgery while Splicebuster, Noiseprint and ManTraNet correctly identify the forged region. Similarly, on image `r0fd69c12t`, our method is outperformed by Mahdian, Splicebuster and Noiseprint. Image `r059cae86t` is correctly detected by Noisesniffer+. However, the provided mask is not as accurate as the ones provided by Noiseprint and Splicebuster. Finally, the partial detection of our method in image `r141665bdt` is outperformed by the detections made by Splicebuster. Other methods such as ManTraNet, Noiseprint and Lyu also spot the forgery. However, they provide noisy heatmaps.

5.3 Limitations

Besides the cases that are out of the scope of the method, namely when the forgery does not modify the background noise model or has higher noise levels, there are several scenarios under which the method may fail to detect the manipulation. Here, we analyze such cases: in Section 5.3.1 we inspect the missed detections, in Section 5.3.2 we examine the false detections and finally, in Section 5.3.3 we study wrong attributions.

5.3.1 Missed Detections

The method may fail to detect forgeries even when the forged area has lower noise levels than the image. The main reasons for this are related to the size of the forgery, the texture of the manipulated area, or even the saturation of the region.

Figure 8 shows examples of such cases. Indeed, image `r1b1c1019t` has a very small forgery. Therefore, even if blocks in V concentrate in this region, in the overall spatial distribution this deviation will not be significant. In image `r05db7e7ft`, the forgery is on a textured zone. Since blocks in L -and therefore in V- are chosen from homogeneous areas, only a few blocks are picked in the forged area. Consequently, the method fails to detect it. Finally, in image `r0667a51ft` the forgery is in a saturated zone. Since the method discards blocks having at least one saturated pixel, the method excludes saturated regions from the analysis. Thus, the forgery is undetectable to the method.

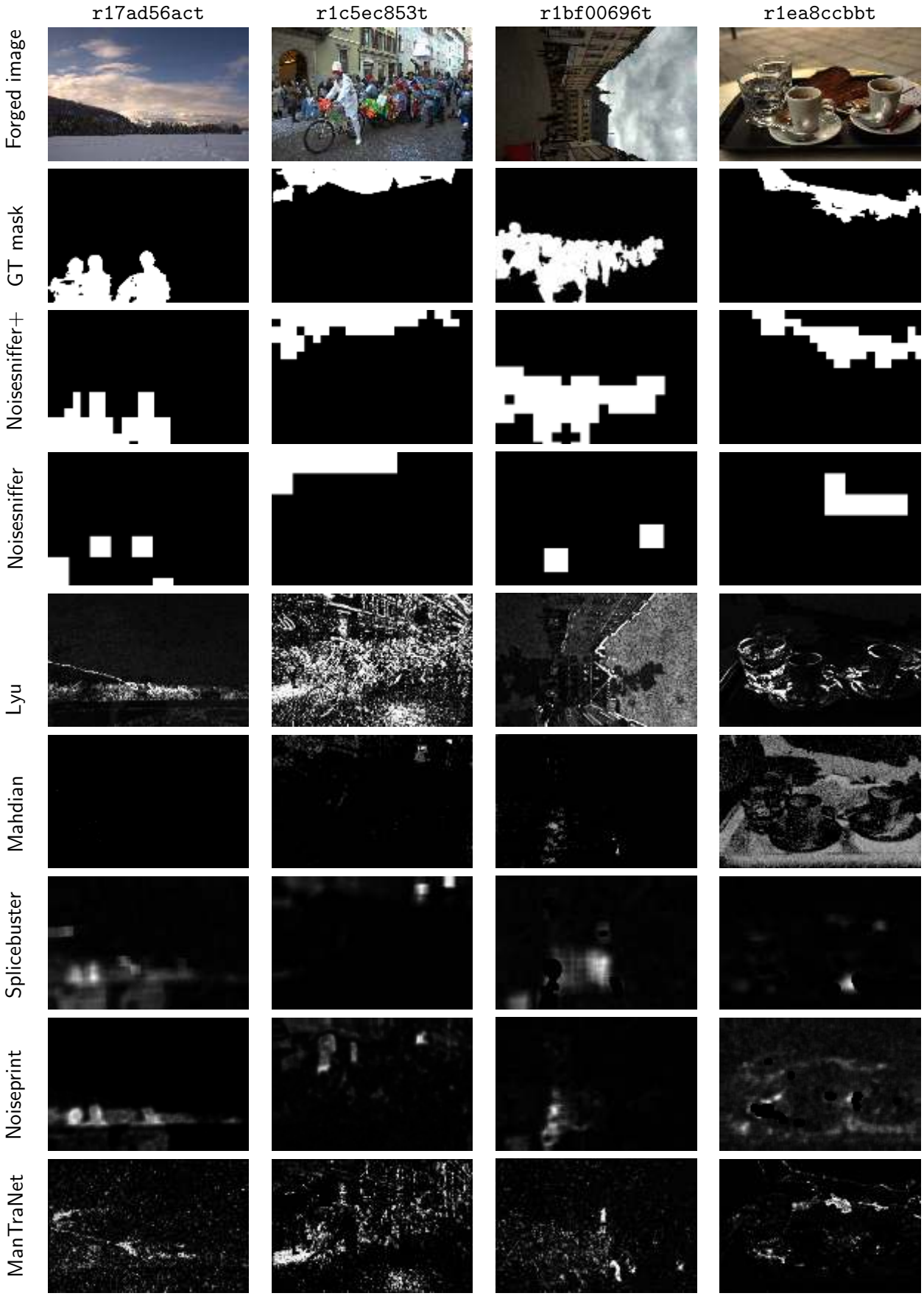


Figure 6: Examples from the Trace dataset [3] where the proposed method outperforms the state of the art. For visualization purposes, Splicebuster's and Mahdian's outputs were inverted.

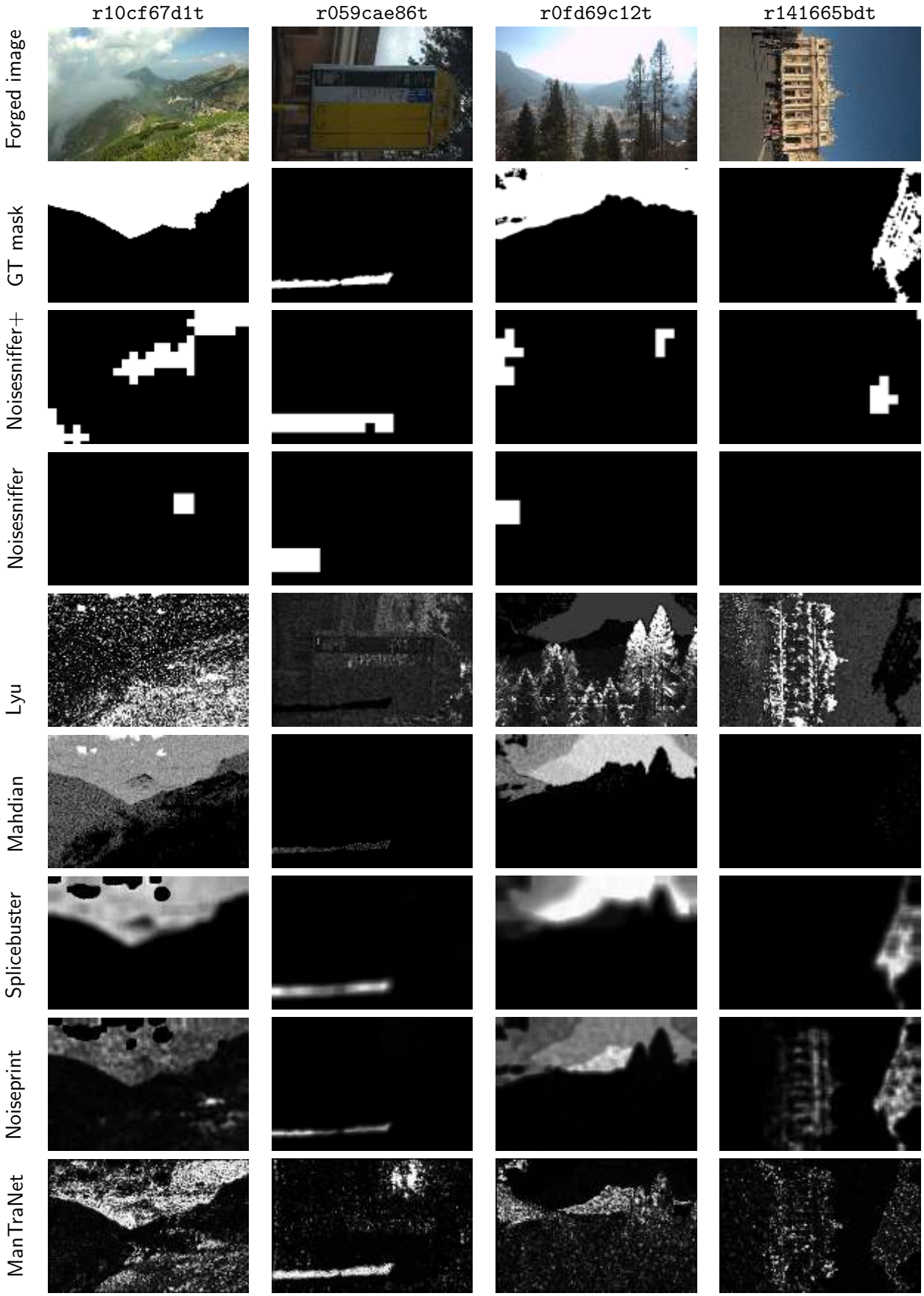


Figure 7: Examples from the Trace dataset [3] where the proposed method is outperformed by other the state-of-the-art methods. For visualization purposes, Splicebuster's and Mahdian's outputs were inverted.

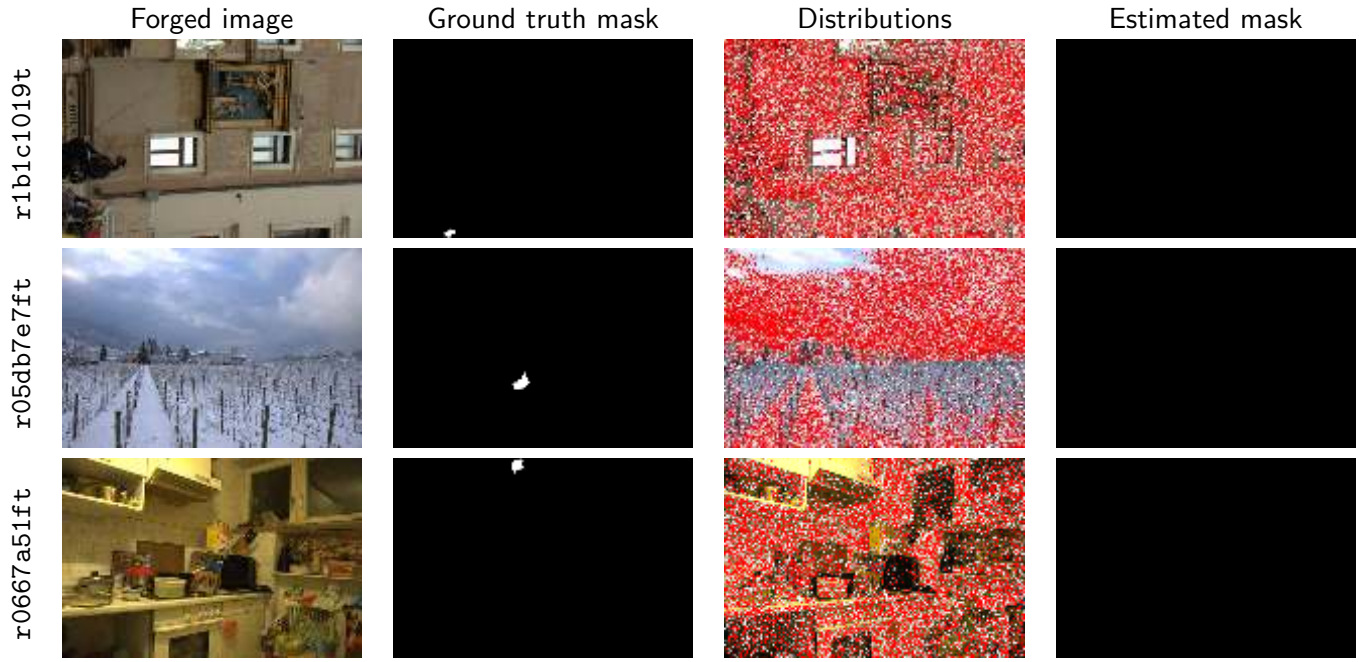


Figure 8: Examples of missed detections from the Trace dataset [3]. Even if the forged area has lower noise levels than the background, the method fails to detect it. The main reasons for this to happen are related to the size of the forgery (`r1b1c1019t`), the texture of the manipulated area (`r05db7e7ft`), or even the saturation of the forged region (`r0667a51ft`).

5.3.2 False Detections

The main cause of false detections is the presence of flat regions in highly textured images. Since the method assumes that there is a fixed proportion of the image blocks that are homogeneous, if the image is highly textured, some blocks in L will contain texture. Therefore, in the next step, when selecting those blocks having the lowest standard deviation, the method will select the homogeneous ones, since when compared to textured blocks, these blocks present a lower variance.

Figure 9 shows an example of false detections in highly textured images. In this case, the method detects the flat zones in the image as forgeries.

Figure 9: Example of false detection from the Trace dataset [3]. The image is highly textured. The method detects the flat zones in the image as forgeries.

5.3.3 Wrong Attribution

Although forgeries having higher noise levels than the background are not supposed to be detected by the method nor generate false detections, in some few and very specific cases, this can happen. When the forgery is big enough and increases the noise in most of the blocks having similar intensities, the method identifies as forged the non-forged regions. The wrong attribution phenomenon happens when the method confuses the background model with the one resulting from tampering.

Figure 10 shows an example of wrong attribution. In this case, the parameters used are $w = 5$, $B = 40000$, $n = 0.05$, $m = 0.3$ and $l_\beta = 100$. The method detects several forgeries. The one in the rocks is due to the textures in the image, as already explained in Subsection 5.3.2. However, the ones in the sky correspond to the wrong attribution phenomenon described above. Indeed, the forgery covers most of the sky. Therefore, the small pristine zones -that have lower noise levels than the forgery- are regarded as local anomalies and thus detected by the method.

Figure 10: Example of wrong attribution from the Trace dataset [3]. The false detection in the rocks is due to the textures in the image, as already explained in Subsection 5.3.2. However, the ones in the sky correspond to the wrong attribution phenomenon. Since the forgery covers most of the sky, the small pristine zones -that have lower noise levels than the forgery- are regarded as local anomalies and thus detected by the method.

6 Robustness

We focus on two common operations performed on social networks that limit the performance of the method: JPEG compression and downsampling. Since most fake images are shared through these networks, assessing the performance of the method under these manipulations is relevant for real-world usage.

6.1 JPEG Compression

JPEG compression is the most popular method for lossy compression of digital images [36]. The first step consists in a color space transformation after which the image is tessellated in 8×8 blocks. Each block is then processed by first applying the DCT II and then quantizing the resulting coefficients. The quantization factors depend on the desired final quality, being bigger for lower-quality factors and smaller for higher ones. Finally, the coefficients are lossless compressed.

Since the quantization step mainly acts on the high-frequency coefficients, the distribution computations, in particular the construction of the set L, become strongly affected by this compression scheme. This effect depends, of course, on the quantization factors used. Table 9 presents the scores obtained by the method at different compression qualities[4]: $QF = 90$, $QF = 70$ and $QF = 50$. The scores degrade as the compression factor decreases. Even for a quality factor of 90, the method already shows degraded results. Figure 11 depicts the effects of JPEG compression at different quality factors (QF) on the performance of the method. The results degrade as the quality factor decreases, and the detected zone is reduced as the compression becomes more aggressive. Besides, strong JPEG compression causes artificial flat zones in the image due to which the method delivers false detections for $QF = 70$ and $QF = 50$.

[4]Images where compressed using the 'ImageMagick' (`https://imagemagick.org`) `convert` command line and specifying the corresponding quality factor with the flag `-quality`.

	MCC	F1	IoU
Uncompressed	0.502	0.512	0.395
$QF = 90$	0.432	0.452	0.337
$QF = 70$	0.318	0.353	0.248
$QF = 50$	0.199	0.249	0.166

Table 9: Scores obtained by the proposed method at different JPEG compression levels of the exomasks noise level dataset from the Trace database [3]. The scores degrade as the quality factor decreases.

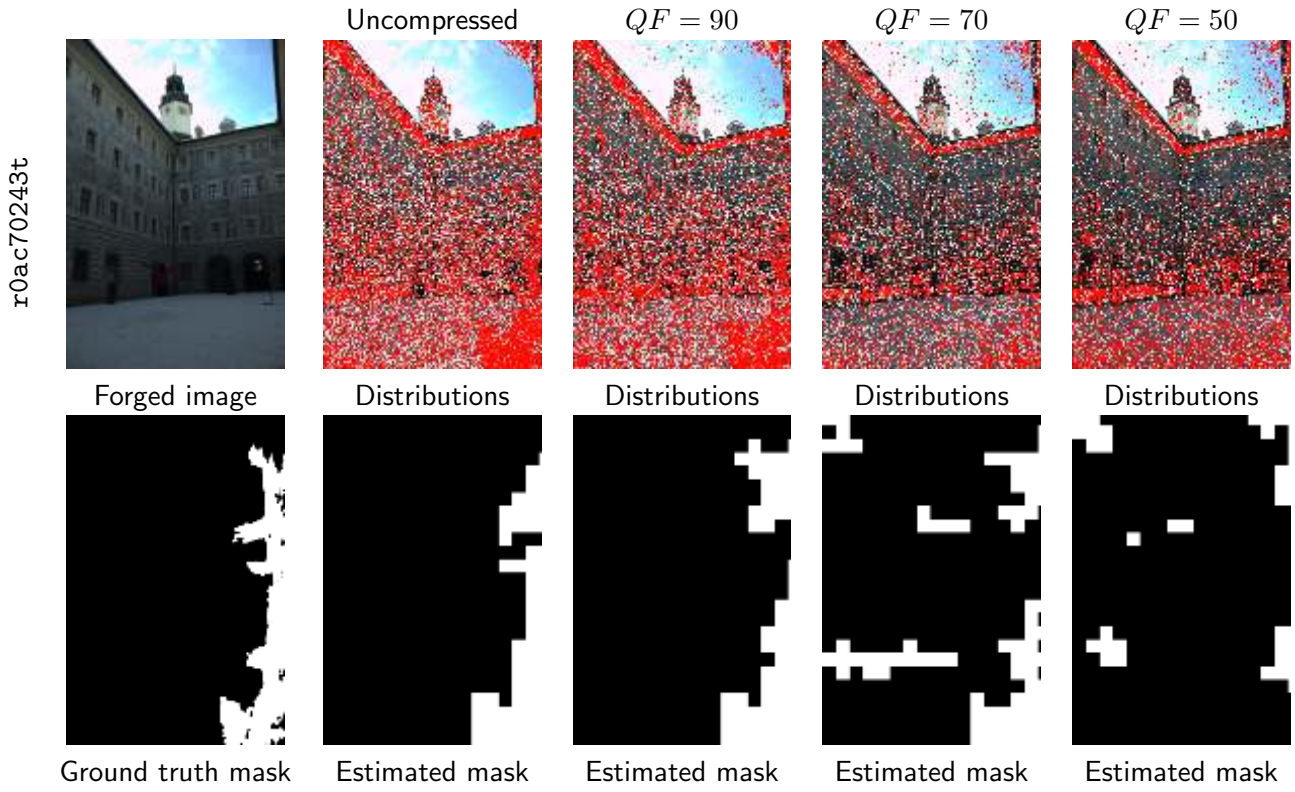


Figure 11: Results obtained on image `r0ac70243t` from the Trace dataset [3] when compressed at different quality factors. The results degrade as the quality factor decreases.

6.2 Image Downsampling

According to the Nyquist sampling theorem, resampling an image with a frequency lower than twice the maximum frequency produces aliasing artifacts. These artifacts can be avoided by applying a low-pass filter to the image before resampling. Although there is no standard way of downscaling an image, here we cover two possible approaches: first, a naive resampling where no pre-filtering is applied, and a more sophisticated case where a Gaussian blur is applied before resampling.

Table 10 presents the scores obtained by the method when scaling the image to half its original resolution with and without pre-filtering the image before downsampling. The downsampling applied consists simply at keeping one pixel out of two in each direction. The Gaussian pre-filtering consists in convolving the image with a Gaussian kernel of size 5×5 and standard deviation $(1.2, 1.2)$.

Firstly, we observe that the scores degrade when the image is downsampled. Downsampling an image with or without pre-filtering reduces the power of the method since fewer samples are available to estimate the distributions. Therefore, detecting statistically significant deviations becomes more difficult. Furthermore, since the size of the cells used for the region-growing algorithm is fixed, this entails a mask accuracy loss; the masks do not adapt correctly to irregular borders.

Gaussian pre-filtering is equivalent to applying a low-pass filter to the image in the frequency domain. Similarly to JPEG compression, the cancellation of high-frequency coefficients strongly affects the construction of the sets L and V. Therefore, when Gaussian pre-filtering is applied before downscaling, the results are worse than when downscaling the image itself.

Figure 12 presents an example of the effects of downscaling with and without pre-filtering for a scaling factor $s = 0.5$. We can observe the reduction in the number of samples in the sets L and V for both downsampling scenarios. When downscaling is applied without any prefiltering, the method is able to detect most of the forgery. However, the mask accuracy is affected by the smaller image resolution. When pre-filtering the image before downscaling, the method is also able to detect most of the forgery but looses some small regions due to the influence of the Gaussian blur in the high frequencies.

	MCC	F1	IoU
Original resolution	0.502	0.512	0.395
Downsampling ($s = 0.5$) without prefiltering	0.423	0.450	0.340
Downsampling ($s = 0.5$) with Gaussian blur prefiltering	0.399	0.430	0.3238

Table 10: Scores obtained by the proposed method at a scaling factor $s = 0.5$ with and without pre-filtering on the exomasks noise level dataset from the Trace database [3]. The performance of the method degrades when the image is downscaled. This degradation is worsen when Gaussian pre-filtering is applied.

7 Conclusion

This paper describes and improves Noisesniffer, a forgery detection method based on noise analysis. As a relevant addition to other existing noise-based methods, Noisesniffer incorporates a statistical validation step detecting only the inconsistencies that could not happen by chance. Each detection is associated with a number of false alarms (NFA). In this article, we build on the previous statistical detection step to provide refined detections. The method also provides, together with the statistically validated detection mask, a visual exploration of the relative flat patch distributions which can also aid the interpretation of the results. Results show that the proposed modifications to the original Noisesniffer method improve its performance. Additionally, the method outperforms the state of the art on forgeries having noise deficit.

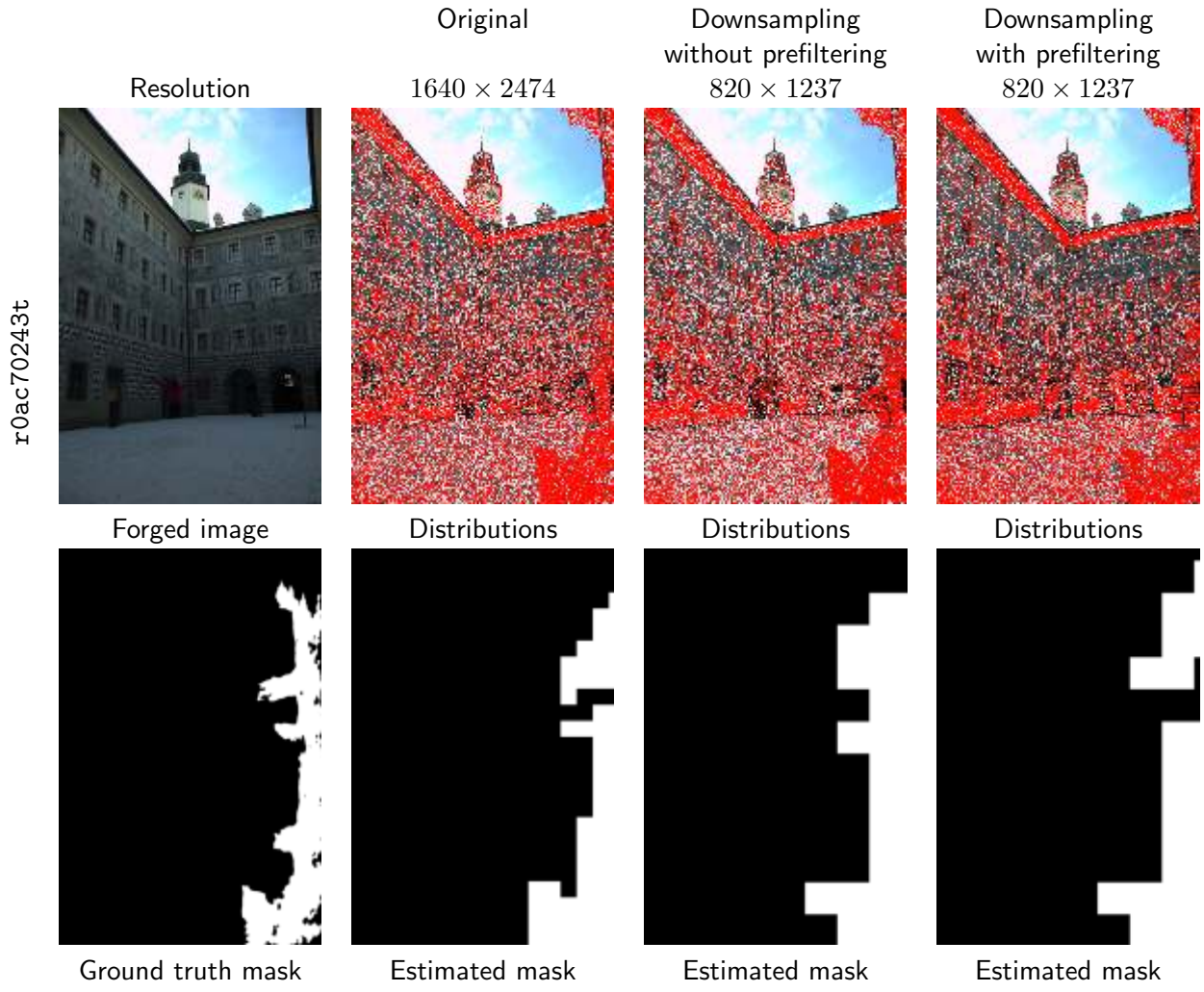
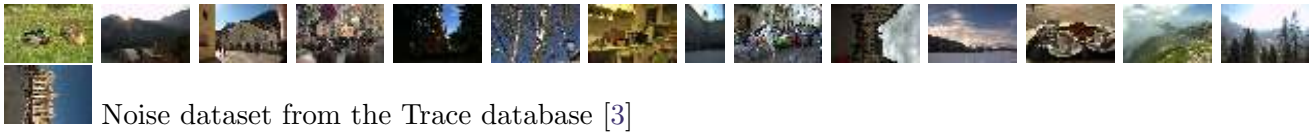


Figure 12: Results obtained on image `r0ac70243t` from the Trace dataset [3] when downscaled at a scaling factor $s = 0.5$ with and without pre-filtering.

Acknowledgment

This work has received funding by the Paris Region Ph.D. grant from Région Île-de-France, the ANR project APATE (ANR-22-CE39-0016) and the European Union under the Horizon Europe VERA.AI project, Grant Agreement number 101070093.

Image Credits



Noise dataset from the Trace database [3]

References

[1] F. Attneave, *Some Informational Aspects of Visual Perception.*, Psychological Review, 61 3 (1954), pp. 183–93, https://doi.org/10.1037/h0054663.

[2] Q. Bammey, *Analysis and Experimentation on the ManTraNet Image Forgery Detector*, Image Processing On Line, 12 (2022), pp. 457–468, https://doi.org/10.5201/ipol.2022.431.

[3] Q. Bammey, T. Nikoukhah, M. Gardella, R. Grompone von Gioi, M. Colom, and J.-M. Morel, *Non-Semantic Evaluation of Image Forensics Tools: Methodology and Database*, in IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022, pp. 2383–2392, https://doi.org/10.1109/WACV51458.2022.00244.

[4] D. Chicco, *Ten Quick Tips for Machine Learning in Computational Biology*, BioData Mining, 10 (2017), p. 35, https://doi.org/10.1186/s13040-017-0155-3.

[5] D. Chicco and G. Jurman, *The Advantages of the Matthews Correlation Coefficient (MCC) Over F1 Score and Accuracy in Binary Classification Evaluation*, BMC Genomics, 21 (2020), https://doi.org/10.1186/s12864-019-6413-7.

[6] M. Colom and A. Buades, *Analysis and Extension of the Ponomarenko et Al. Method, Estimating a Noise Curve from a Single Image*, Image Processing On Line, 3 (2013), pp. 173–197, https://doi.org/10.5201/ipol.2013.45.

[7] D. Cozzolino, G. Poggi, and L. Verdoliva, *Splicebuster: A New Blind Image Splicing Detector*, in IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, 11 2015, pp. 1–6, https://doi.org/10.1109/WIFS.2015.7368565.

[8] D. Cozzolino and L. Verdoliva, *Noiseprint: A CNN-Based Camera Model Fingerprint*, IEEE Transactions on Information Forensics and Security, 15 (2020), pp. 144–159, https://doi.org/10.1109/TIFS.2019.2916364.

[9] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, *Raise: A Raw Images Dataset for Digital Image Forensics*, in ACM Multimedia Systems Conference, 2015, pp. 219–224.

[10] A. Desolneux, L. Moisan, and J.-M. Morel, *Gestalt Theory and Computer Vision*, 2004, pp. 71–101, https://doi.org/10.1007/1-4020-2081-3_4.

[11] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, Springer Publishing Company, Incorporated, 1st ed., 2007. ISBN 0387726357.

[12] H. Farid, *Digital Doctoring: How to Tell the Real from the Fake*, Significance, 3 (2006), pp. 162 – 166, https://doi.org/10.1111/j.1740-9713.2006.00197.x.

[13] H. Farid, *Photo Forensics*, The MIT Press, 2016, https://doi.org/10.7551/mitpress/10451.001.0001. ISBN 9780262355452.

[14] M. Gardella, P. Musé, J.-M. Morel, and M. Colom, *Noisesniffer: a Fully Automatic Image Forgery Detector Based on Noise Analysis*, in IEEE International Workshop on Biometrics and Forensics (IWBF), IEEE, 2021, pp. 1–6, https://doi.org/10.1109/IWBF50991.2021.9465095.

[15] M. Gardella and P. Musé, *Image Forgery Detection Via Forensic Similarity Graphs*, Image Processing On Line, 12 (2022), pp. 490–500, https://doi.org/10.5201/ipol.2022.432.

[16] M. Gardella, P. Musé, J.-M. Morel, and M. Colom, *Forgery Detection in Digital Images by Multi-Scale Noise Estimation*, Journal of Imaging, 7 (2021), https://doi.org/10.3390/jimaging7070119.

[17] R. Grompone von Gioi, C. Hessel, T. Dagobert, J.-M. Morel, and C. de Franchis, *Ground Visibility in Satellite Optical Time Series Based on A Contrario Local Image Matching*, Image Processing On Line, 11 (2021), pp. 212–233, https://doi.org/10.5201/ipol.2021.342.

[18] R. Grompone von Gioi and J. Jakubowicz, *On Computational Gestalt Detection Thresholds*, Journal of Physiology-Paris, 103 (2009), pp. 4–17, https://doi.org/10.1016/j.jphysparis.2009.05.002.

[19] I. Jensen and A. J. Guttmann, *Statistics of Lattice Animals (Polyominoes) and Polygons*, Journal of Physics A: Mathematical and General, 33 (2000), pp. L257–L263, https://doi.org/10.1088/0305-4470/33/29/102.

[20] T. Julliand, V. Nozick, I. Echizen, and H. Talbot, *Using The Noise Density Down Projection To Expose Splicing in JPEG Images*, (2017). https://hal.science/hal-01589761.

[21] T. Julliand, V. Nozick, and H. Talbot, *Automatic Image Splicing Detection Based on Noise Density Analysis in Raw Images*, in International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2016, pp. 126–134.

[22] A. Kashyap, R. S. Parmar, M. Agrawal, and H. Gupta, *An Evaluation of Digital Image Forgery Detection Approaches*, International Journal of Applied Engineering Research, 12 (2017), pp. 4747–4758.

[23] Y. Ke, Q. Zhang, W. Min, and S. Zhang, *Detecting Image Forgery Based on Noise Estimation*, International Journal of Multimedia and Ubiquitous Engineering, 9 (2014), pp. 325–336, https://doi.org/10.14257/ijmue.2014.9.1.30.

[24] M. Lebrun, M. Colom, A. Buades, and J. M. Morel, *Secrets of Image Denoising Cuisine*, Acta Numerica, 21 (2012), p. 475–576, https://doi.org/10.1017/S0962492912000062.

[25] B. Liu and C.-M. Pun, *Splicing Forgery Exposure in Digital Image by Detecting Noise Discrepancies*, International Journal of Computer and Communication Engineering, 4 (2015), p. 33, https://doi.org/10.7763/IJCCE.2015.V4.378.

[26] C. Liu, R. Szeliski, S. B. Kang, C. Zitnick, and W. Freeman, *Automatic Estimation and Removal of Noise from a Single Image*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (2008), pp. 299–314, https://doi.org/10.1109/TPAMI.2007.1176.

[27] S. Lyu, X. Pan, and X. Zhang, *Exposing Region Splicing Forgeries with Blind Local Noise Estimation*, International Journal of Computer Vision, 110 (2013), pp. 202–221, https://doi.org/10.1007/s11263-013-0688-y.

[28] B. Mahdian and S. Saic, *Using Noise Inconsistencies for Blind Image Forensics*, Image and Vision Computing, 27 (2009), pp. 1497–1503, https://doi.org/10.1016/j.imavis.2009.02.001.

[29] O. Mayer, B. Bayar, and M. C. Stamm, *Learning Unified Deep-Features for Multiple Forensic Tasks*, in ACM Workshop on Information Hiding and Multimedia Security, ACM, 2018, pp. 79–84, https://doi.org/10.1145/3206004.3206022.

[30] X. Pan, X. Zhang, and S. Lyu, *Exposing Image Forgery with Blind Noise Estimation*, in ACM Multimedia Workshop on Multimedia and Security, New York, NY, USA, 2011, Association for Computing Machinery, p. 15–20, https://doi.org/10.1145/2037252.2037256.

[31] A. C. Popescu and H. Farid, *Statistical Tools for Digital Forensics*, in Information Hiding, 2004, https://doi.org/10.1007/978-3-540-30114-1_10.

[32] C.-M. Pun, B. Liu, and X. Yuan, *Multi-Scale Noise Estimation for Image Splicing Forgery Detection*, Journal of Visual Communication and Image Representation, 38 (2016), https://doi.org/10.1016/j.jvcir.2016.03.005.

[33] S. Pyatykh, J. Hesser, and L. Zheng, *Image Noise Level Estimation by Principal Component Analysis*, IEEE Transactions on Image Processing, 22 (2012), pp. 687–699, https://doi.org/10.1109/TIP.2012.2221728.

[34] V. Schetinger, M. M. Oliveira, R. da Silva, and T. J. Carvalho, *Humans Are Easily Fooled by Digital Images*, Computers & Graphics, 68 (2017), pp. 142–151, https://doi.org/10.1016/j.cag.2017.08.010.

[35] M. a. Tailanian, P. Musé, and A. Pardo, *A Contrario Multi-Scale Anomaly Detection Method for Industrial Quality Inspection*, 2022, https://doi.org/10.48550/ARXIV.2205.11611.

[36] G. K. Wallace, *The JPEG Still Picture Compression Standard*, IEEE Transactions on Consumer Electronics, 38 (1992), pp. xviii–xxxiv, https://doi.org/10.1109/30.125072.

[37] Y. Wu, W. AbdAlmageed, and P. Natarajan, *ManTra-Net: Manipulation Tracing Network for Detection and Localization of Image Forgeries with Anomalous Features*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019, https://doi.org/10.1109/CVPR.2019.00977.

[38] H. Yao, S. Wang, X. Zhang, C. Qin, and J. Wang, *Detecting Image Splicing Based on Noise Level Inconsistency*, Multimedia Tools and Applications, 76 (2017), pp. 12457–12479, https://doi.org/10.1007/s11042-016-3660-3.

[39] M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris, *Large-Scale Evaluation of Splicing Localization Algorithms for Web Images*, Multimedia Tools and Applications, 76 (2017), pp. 4801–4834, https://doi.org/10.1007/s11042-016-3795-2.

[40] H. Zeng, Y. Zhan, X. Kang, and X. Lin, *Image Splicing Localization Using PCA-Based Noise Level Estimation*, Multimedia Tools and Applications, 76 (2017), pp. 4783–4799, https://doi.org/10.1007/s11042-016-3712-8.

[41] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, *Learning Rich Features for Image Manipulation Detection*, in IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1053–1061, https://doi.org/10.1109/CVPR.2018.00116.

[42] N. Zhu and Z. Li, *Blind Image Splicing Detection Via Noise Level Function*, Signal Processing: Image Communication, 68 (2018), pp. 181–192, https://doi.org/10.1016/j.image.2018.07.012.

[43] D. Zoran and Y. Weiss, *Scale Invariance and Noise in Natural Images*, in IEEE International Conference on Computer Vision (ICCV), 2009, pp. 2209–2216, https://doi.org/10.1109/ICCV.2009.5459476.