A multi-method approach for tampering localization on digital forensic

Matteo Proch, Zeno Sambugaro, Giuseppe Spallitta, Matteo Zanoni

***Abstract*—A great deal of different tampering methods were developed in the past decades by scientists, providing a wide knowledge about the topic. In this report we will briefly explain the main concepts behind the most powerful state-of-the-art algorithms and then we will explain how we combined them in a unique pipeline which tries to take the best from each algorithm.**

KEYWORDS: Digital forensic, tampering localization, JPEG traces, Color Filter Array, Photo-Response Non-Uniformity

I. Exploited state-of-the-art algorithms

A. ADQ2

The ADQ2 algorithm [1] is a test to find forged areas in images. The algorithm works for JPEG images. It is based on the idea that a JPEG compressed image is forged and then saved applying a second JPEG compression without resizing. In this assumption, the forged areas would have been compressed only once whereas the original areas would have been compressed twice.

When an image is compressed into JPEG format its DCT coefficients are quantized with QF_1 factor; if the same image is then compressed again a second quantization occurs. This means that the DCT coefficients are quantized again with QF_2 factor. If QF_2 is not a multiple of QF_1, then the second quantization creates artifacts in the image histograms. Those artifacts, if detected, can be used to identify doubly quantized images.

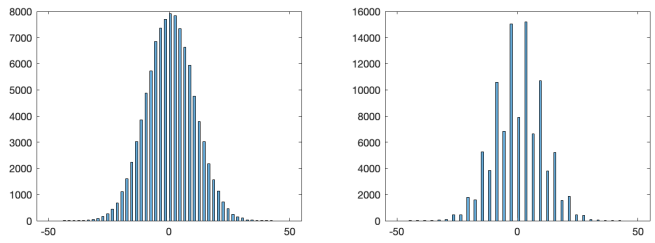

Fig. 1. On the left: histogram after first quantization with $QF_1 = 2$; on the right: histogram after second compression with $QF_2 = 3$

The main idea is to use Bayesian interference to assign to each DCT coefficient a value in the range $[0, 1]$ representing the probability of that coefficient being doubly quantized. Those probabilities are then accumulated in 8x8 blocks and provide a double quantization probability map allowing to differentiate between singularly and doubly compressed areas.

B. CAGI

The CAGI method [2] is based on a technique looking for JPEG blocking artifact discontinuities as a sign of possible manumission. It was originally developed to solve one of the most common tampering schemes: image splicing that breaks the original grid alignment either due to its placement or due to resampling transformations (scaling, rotation, etc.) of the spliced area. The method extends a JPEG grid detection algorithm from the literature by introducing two updates:

- a grid alignment confidence measure designed to identify whether an image block violates the global grid pattern, either due to misalignment, distortion or complete absence of encoding artifacts;
- a content-aware filtering step designed to account for grid discontinuities caused by the image content, strengthening the method's localization ability and overall output interpretability.

As a first step, this approach improves upon the grid position estimation approach presented in the past literature by adding a secondary level of analysis which allow to estimate the grid position more reliably. A measure K" drawn from the interrelationships between values of K at different positions it used in following steps. To estimate K", it was first estimated two intermediate measures: K', which calculates the value differences between values of K at different positions, and S which analyzes the sign patterns of K. K" is then calculated as a combination of K' and S. Consecutively, the blocks that do not conform to the detected grid were indicated by K" as tampering candidates. To moderate the impact of a variety of factors (such as strong edges, visual texture patterns, over/under exposure, etc) all information gathered during the previous procedure were exploited. Besides the heat map of local block contributions to K", it is also produced a series of auxiliary maps aimed to isolate and remove the artifacts produced by such phenomena, and only keep the regions that can be confidently violating the JPEG grid due to tampering. To produce these masks it was used:

- The heat map of the local K" block scores for the best-fitting grid;
- The heat maps of the local K" block scores for all other candidate grids;
- An edge detection map used both to remove strong edges from the results (as they tend to disrupt false positives) and to locate soft, widespread edges, (which we use as indicators that the block is suitable for accurate grid estimation);
- A map identifying over- and under-exposed areas, where

grid detection would be impossible anyway, and thus any inconsistencies found there are unreliable.

The final step consists in producing a readable output with clear contrast between tempered and untampered regions. To achieve this aim all intermediate information are merged together, providing our final map.

C. CFA

The CFA method [5] is used to detect the presence of digital fingerprints and traces of tampering. Using this tool can provide also a fine-grained localization of forgery within digital image, in particular for double JPEG compression.
Among the numerous fingerprints considered in image forensic literature we consider the traces left by the interpolation process, which is applied in two phases:

- Acquisition processing: the missing pixels values for the three color layers are obtained by applying the interpolation process, also referred as demosaicing.
- Geometric transformations, to obtain transformed image. When scaling, rotation, translation, shearing, are applied to an image, pixels within the to-be-transformed image are relocated to a new lattice, and new intensity values have to be assigned to such positions by means of interpolation of the known values, also referred to as resampling operation.

Ideally, an image coming from a digital camera, in the absence of any successive processing, will show demosaicing artifacts on every group of pixels corresponding to a CFA element. On the contrary, demosaicing inconsistencies between different parts of the image, as well as resampling artifacts in all or part of the analyzed image, will put image integrity in doubt. The effort is focused on the study of demosaicing artifacts at a local level: by means of a local analysis of such traces we aimed at localizing image forgeries whenever the presence of CFA interpolation is not present.

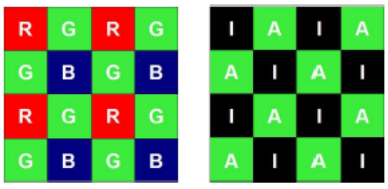

Fig. 1. (a) the Bayer's filter mosaic; (b) the quincunx lattice $\mathcal{A}$ for the acquired green channels and the complementary quincunx lattice $\mathcal{I}$ for the interpolated green channels.

The main idea was to consider as specific CFA the most frequently used Bayers filter mosaic, a 2-2 array having red and green filters for one row and green and blue filters for the other. Furthermore, only the green channel is considered; since the green channel is upsampled by a factor 2, for a generic square block we have the same number of samples (and the same estimation reliability) for both classes of pixels (either acquired or interpolated).
By focusing on the green channel, the even/odd positions (i.e. acquired/interpolated samples) of the one-dimensional case turn into the quincunx lattice A for the acquired green values and the complementary quincunx lattice I for the interpolated green values. Similar to the onedimensional case, it is assumed that in the presence of CFA interpolation the variance of the prediction error on lattice A is higher than the variance of the prediction error on lattice I, and in both cases it is content dependent. On the contrary, when no demosaicing has been applied, the variance of the prediction error assumes similar values on the two lattices.
Hence, in order to detect the presence/absence of demosaicing artifacts, it is possible to evaluate the imbalance between the variance of the prediction error in the two different lattices.

D. NOI4

The basic idea behind NOI4 [3] is very simple. Natural images are full of noise. When they are modified this often leaves visible traces in the noise in an image. The biggest problem is how to see this noise.

Fig. 2. The figure shows an image taken from the development dataset and its tampering map provided by NOI4. We can clearly see how the method was able to detect the tampering present in the image.

The implementation consists in using a noise reduction filter (a separable Median Filter) and reverse the results. In this way noise is still present and it removes the rest of the image. This tool is particular effective with recognition of airbrushing, warps, deforms transformed clones.

E. MULTI-PRNU

Multi-Scale Photo-Response Non-Uniformity algorithm [4] is an evolution on the classical PRNU based algorithms. Each photograph has different traces left during the acquisition process due to manufacturing imperfections in the camera sensor and CFA or due to the use of different pixel-interpolation algorithms and post-processing by the Digital Image Processor. Estimation of this noise yields a unique fingerprint for each camera and enabling one link a specific photograph to a particular camera.

The image acquisition pipeline can be modeled as follows:

$$I(x,y) = I_0(x,y) + K(x,y) * I_0(x,y) + N(x,y)$$

Where $I(x, y)$ represent the final image, $I_0(x, y)$ the idealized noise-free version of the image, $K(x, y)$ the PRNU fingerprint and $N(x, y)$ the remaining addictive noise. Given a new image to analyze we can use one of various denoising filters to extract the PRNU fingerprint of that image and then compare it with PRNUs of known cameras to find the most likely source.

For forgery detection the most used method is to use sliding-windows and then compute the normalized correlation between the residual image (denoised image) and the PRNU estimate (multiplied by the image content). In order to achieve reliable results the window dimensions should not be to small, however larger windows can lead to smaller forgeries being missed. Hence in almost all literature it is accepted that $128px * 128px$ is the best possible compromise and this size is the only one used.

Multi-Scale PRNU improves on classical PRNU by using multiple dimensions for the sliding-window and then combining the results into a single map. This method allows the algorithm to use the benefits from both small windows and larger ones. In fact large windows guarantee that the correlation yields reliable statistics but tend to miss small forgeries; smaller windows on the other hand can easily detect small forgeries but tend to give big areas of uncertainty and noise. Each window size is used to extract a *candidate probability map*, those maps are then fused into a single probability map used as the final result. Each pixel value of the final map is calculated by a weighted combination of all the candidate maps taking various factors into account:

- **Neighbouring values:** in order to penalize highly noisy areas of the smaller masks.
- **Probability values:** weighting is done so that the final decision is biased towards either of the hypothesis (forged or original) avoiding values in the middle.
- **Region reliability:** areas of the original image that are classified as unreliable (e.g. dark, highly textured or saturated areas) have their weights lowered in order to increase the effect of neighbouring (more reliable) values on such areas.

This process allows for better results especially for smaller forgeries if compared to other PRNU based algorithms.

II. The pipeline

The development dataset was composed by a variety of different manipulations so, in order to accomplish our task, we tried to extract our localization tampering maps refining and combining the state-of-the-art algorithms described above. In particular, we relied on a **"waterfall" approach**: starting from the most reliable algorithm according to our manual tests, we applied the algorithms to our images. Once the tampering map was obtained, we calculated some statistics; the goal to this step was to define a mechanism which could help us in deciding if the resulting map had to be considered our final result or if it was necessary to discard it and proceed to the next algorithm.
Going into detail, given as input an image:

- First of all we verify its format: if the extension is *tif* then we apply a JPEG compression with quality equal to 100, saving the obtained result in a new image file with extension *jpg*. Some algorithms require as input a JPEG compressed image, so this is why we need this step.
- Some tif images have an additional image layer, usually referred as *transparency* in literature. The presence of this layer inhibits the execution of most of the algorithms, so we provided some lines of code to remove it from images, saving the resulting new image in a new file. We will use this file from now on if transparency is present.
- Once we performed the preprocessing phase, we start our tampering localization using the **ADQ2 algorithm**, giving as input the JPEG associated image. The resulting matrix values swing from 0 (not tampered pixel) to 1 (definitely tampered pixel), so we decided to use 0.75 as threshold value to distinguish tampered and not-tampered pixels. If the percentage of retrieved tampered pixels was not lower than 5% and not greater than 95%, the approach was considered successful and the result was used as output.
- If the previous approach fails, we apply the **CAGI algorithm**. The CAGI algorithm returns a matrix with values in an unbounded interval; we know that the standard CAGI map reveals tampered pixels only if they have high values. In order to solve this issue, taking the CAGI map, we calculated the difference between the highest and the lowest value. Since in our empirical tests we have seen that good results were significative only when this difference was higher than 0.5, we set this as additional condition together with the 5-95 percentage of tampering area extension. All pixels whose value was between the interval $[highestValue*0.5, highestValue]$ was considered tampered. The approach used for CAGI INVERSE was in a mirror pattern to the one described above. In particular the threshold value for the difference between highest and lowest value was set to 0.48 and the tampered pixels were considered the ones in the interval $[lowestValue, lowestValue + (difference * 0.5)]$
- If also CAGI was unsuccessful, we tried the **CFA algorithm**. We maintained all the parameters and threshold values used by the developers of the algorithm, which guaranteed good performances. As in the CAGI algorithm the values inside the matrix are unbounded. Empirically we provided additional rules to overcome this issue.
  If the difference between highest and lowest values was higher than 100, results were mostly correct and all pixels in the interval $[lowestValue, lowestValue + (difference * 0.05)]$ were principally tampered.
  If the difference was higher than 7 but lower than 100, only negative pixels were considered tampered.
  If the difference was lower than 7, the approach was considered ineffective and the map discarded. The 5-95 percentage of tampering area extension condition is always used.
- The penultimate algorithm chosen was **NOI4**. The procedure is similar to the steps above; we apply the algorithm, then we set as tampered pixels only a fraction of them based on an interval similar to the CAGI algorithm. The 5-95 percentage of tampering area extension condition is

always used.
- Last but not least, we applied the **PRNU** approach. Thanks to the package found on the web, we were able to fuse different maps obtained from different adaptive windows (96, 128, 192, 256). The result is already usable, so we did not need to set a threshold value by ourselves.

Before saving the decision map in a bitmap file, we performed a cleaning function, removing scattered pixels and isolated from all the rest. The MULTI-PRNU provided a function, *mapCleanup*, which performed this task so we used it.
There are some rare cases where every algorithm does not provide a good results or it does not match our empirical conditions. In this case instead of returning no decision map, we chose to add a "dummy decision map". We tried different patterns and evaluated them comparing the following maps with some ground truth maps:

- Upper/lower section of an image
- Left/right section of an image
- Upper/lower diagonal of an image
- Central square of the image

After evaluating them with a thirty of images, we saw that the choice of upper/lower section was the best in order to obtain a good mean of F_1-score. This leads us to use this map as our final try.

III. Results on development set

We run the Matlab function giving as input each of the 800 images, returning the maps and saving in a log file the tampering algorithm applied. To understand the effectiveness of each algorithm we analyzed their frequency of application. Table 1 shows how often each algorithm produced a map. As we can see, ADQ2 produced most of the results; this is not surprising since it is the first method exploited by our pipeline. It is also normal that the sum of percentage doesn't reach 100% since there are some images where every algorithm failed so we used the dummy decision map.

TABLE I
Simulation Parameters

ADQ2	54%
CAGI	10.375%
CFA	15.875%
NOI4	9.875%
MULTIPRNU	6.125%

An essential parameter useful to estimate the performance of our pipeline is the F_1-score. It was calculated using the code provided in the rules of this competition. The analysis, enforced on the whole dataset, gave us a mean F_1-score of 0.81606; removing from the results the 5% worse images we achieved a F_1-score of 0.84836.

A. Worst 3 results

Based on the development dataset, our pipeline performed very bad on *dev-0225*, *dev-0259* and *dev-0289*. Figure 3 shows the results. All the three images were tampered using the NOI4 method. This highlights two different aspects:

- ADQ2, CAGI and CFA algorithms were not able to provide a useful result. This is not a big shock; it is impossible to provide a universal tampering localization method, able to detect every type of existing manipulation.
- NOI4 tampering maps quality can vary from image to image. The main reason behind the inconstancy of the results is probably connected to a not-so-good skim of pixel in the map. Actually NOI4 was the last algorithm to be implemented in our pipeline, so it is quite possible we have missed the choice of a better threshold value.

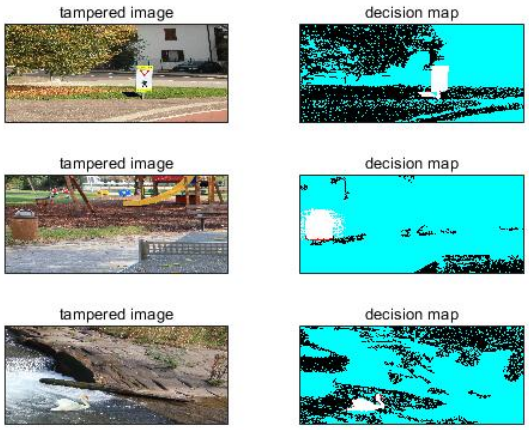

Fig. 3. The image shows the 3 worst images and their relative decision maps. White pixels are tampered pixels correctly tampered, blue pixel are not tampered pixels wrongly considered tampered, red pixels are tampered pixels wrongly considered not tampered.

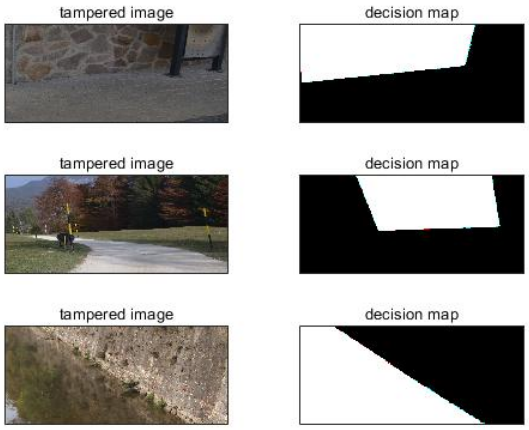

Fig. 4. The image shows the 3 best images and their relative decision maps. White pixels are tampered pixels correctly tampered, blue pixel are not tampered pixels wrongly considered tampered, red pixels are tampered pixels wrongly considered not tampered.

B. Best 3 results

Based on the development dataset, our pipeline performed very well on *dev-0027*, *dev-0357*, *dev-0561*.Figure 4 shows the results. All the three images were tampered using the ADQ2 method. This is a proof of the effectiveness of the method in the presence of JPEG traces and it is the main reason it was chosen as the first algorithm to apply in our pipeline.

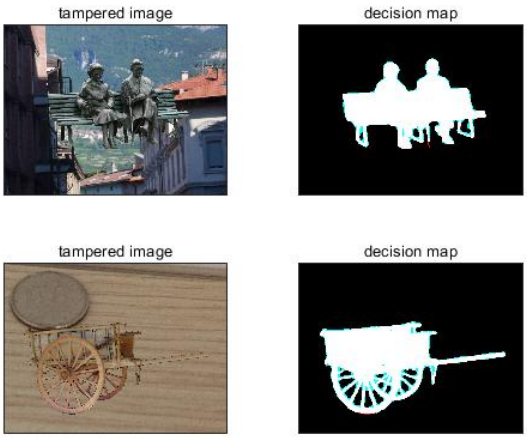

Fig. 5. The image shows the power of ADQ2 in managing edges and empty spaces. White pixels are tampered pixels correctly tampered, blue pixel are not tampered pixels wrongly considered tampered, red pixels are tampered pixels wrongly considered not tampered.

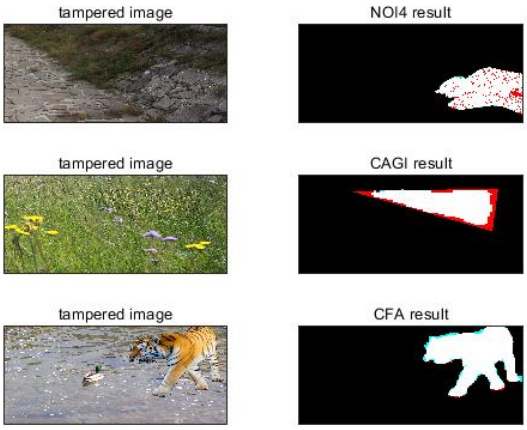

Fig. 6. The image shows the performance of NOI4, CAGI and CFA algorithms. White pixels are tampered pixels correctly tampered, blue pixel are not tampered pixels wrongly considered tampered, red pixels are tampered pixels wrongly considered not tampered.

C. Honorable mentions

Besides the best 3 tampering maps based on F_1-score, we wanted to show some other maps whose results are excellent and which offer some insights.
In figure 5 we decided to show how ADQ2 performed while trying to detect tampered section whose pattern was not an uniform area. ADQ2 is able to detect precisely the tampered area in presence of JPEG traces, delineating the edge with high details and also not covering small empty areas (for instance area in the middle of the spokes of the wheels). Even though none of the best 3 tampering maps was produced by CAGI, NOI4 and CFA algorithms, their results are usually impressive, so we decided to show a sample for each method highlighting their effectiveness. Figure 6 shows some relevant examples.

IV. Results on test set

Using the *get_map* function, only 3 images used the dummy map, demonstrating how the methods implemented return something almost every time.
Based on the ground truth of the development set, we think that at least the 75% of all the tampering maps have a good result, since the shape of the tampered area is well defined on these images. Most of the other 25% of images are usually produced by NOI4 and PRNU; it is a clear sign that future improvements can be done in using these method, probably refining the parameters and the camera models.
As a final consideration to this project, studying all these different approaches to solve different manipulations clearly highlighted the difficulty of this field of study. Digital forensic has to deal with an incredibly large number of different attacks, so it is not surprising that it is subject of study for decades. The search of JPEG traces in images provides excellent results in defining both edges and area of the tampered area. Also studying inconsistencies based on the Color Filter Array and the PRNU provides good results, despite some ”dirty results” (i.e. too much bits considered tampered around the correct area).
We decided not to rewrite from scratches the codes implemented by the papers discussed above: they are the fruit of years of studies and, as a consequence we decided to focus on building the best pipeline to achieve our goal, reducing the number of false positives, and slightly modify them to improve their performances if necessary.
Obviously other types of tampering localization methods could be introduced (for instance geometric studies, analyzing guidelines and shadows of object, easily revealing superimposed external objects), but given the deadlines and the overall quality of manipulations we decided to discard it.

References

[1] Tiziano Bianchi, Alessia De Rosa and Alessandro Piva. Improved DCT coefficient analysis for forgery localization in JPEG images. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference*, Istanbul, Turkey, pp. 2444–2447, May 2011.
[2] Iakovidou, Chryssanthi and Zampoglou, Markos and Papadopoulos, Symeon and Kompatsiaris, Ioannis. (2018). Content-aware Detection of JPEG Grid Inconsistencies for Intuitive Image Forensics. Journal of Visual Communication and Image Representation.
[3] https://29a.ch/2015/08/21/noise-analysis-for-image-forensics, Noise Analysis for Image Forensics by Jonas Wagner.
[4] P. Korus and J. Huang, ”Multi-Scale Analysis Strategies in PRNU-Based Tampering Localization,” in IEEE Transactions on Information Forensics and Security, vol. 12, no. 4, pp. 809-824, April 2017.
[5] P. Ferrara, T. Bianchi, A. De Rosa and A. Piva, ”Image Forgery Localization via Fine-Grained Analysis of CFA Artifacts,” in IEEE Transactions on Information Forensics and Security, vol. 7, no. 5, pp. 1566-1577, Oct. 2012.