Department of Information Engineering and Computer Science (DISI)

University of Trento, Italy

MULTIMEDIA SECURITY - AY 2018/2019
2nd Assignment Report

Image tampering detection and localization

ThreeOfAveMaria

De Pellegrini Martin [207051]
Rzechowski Kamil [205082]
Favia Federico [207350]

Table of contents

Method

Our starting point was the Matlab laboratory class [4] on PRNU: this parameter is computed as a noise fingerprint to identify from which camera the picture was shot (using two dimensions-correlation between residual test image and PRNU). In the formula below, I(x,y) is the original image, **K** is the sensor noise pattern, aka PRNU, γ is a multiplicative factor and N the additive noise.

$$I(x,y) = I_0(x,y) + I_0\gamma(x,y)\boldsymbol{K}(x,y)+N(x,y)$$

The steps to compute **K** are:

1) Get noisy test set of images (keeping only the green channel because it contains the main information);
2) Denoise images with Wiener Filter;
3) Compute for each image the residual $W_k(x,y) = I_k(x,y) - I_k^{denoised}(x,y)$;
4) Compute PRNU of the denoised set

$$\boldsymbol{K}(x,y) = (\sum^n_{k=1} W_k(x,y)I_k(x,y)) / (\sum^n_{k=1} I_k^2(x,y)) ;$$

Initially, to try something new, we have thought to work with C++ and the useful library of OpenCV, but then we started to get some issues: therefore we have decided to work in parallel both on Matlab and C++ to keep at the end only the best results.

1st Approach: PRNU-based and 2D-correlation

According on what the above-mentioned lecture has taught us and knowing that the 800 images of the development set were taken by the same four cameras (of whom each we have 100 noisy images) used for the test set, we agreed to start computing the PRNU for each camera and using it as a fundamental fixed parameter in our algorithm. Since at the laboratory we were provided by different size images, we have learned how to compute the useful parameter in littler patches (windows). Therefore we computed different PRNU matrices using different size patches, beginning from the entire image (1500x2000) and then we decreased over and over the patch size such as 150x200, 100x100, 50x50 and at the end 15x20. Except for the PRNU computed on the whole image which obviously has the same dimension of the test images (1500x2000), for the smaller ones we iteratively has repeated the step patch by patch to obtain a PRNU matrix of the same size of the image, for each camera. Our impression was that computing a PRNU in smaller merged blocks was more precise that a PRNU computed on the whole image, even if after some considerations we had to face up to reality.
In short, we have computed two dimension-correlation block by block (e.g. 150x200, or 100x100, or 50x50, until 15x20) between the PRNU matrix of each camera and the test image, then shifting the moving the window with a step equal to the block size. So for every images we have gained four correlation maps (corresponding to each camera). Summing the values of each correlation map and finding the maximum one we have got according to our reasoning the higher correlation map corresponding to the maximum probable camera that shot that picture. After populating each relating block of pixels with the same value of

correlation we have got an image of same size of the original one (1500x2000). After converting it into grey-scale and thresholded with a threshold computed automatically by Matlab with Otsu's method, since in our opinion the blocks below a certain value were the less correlated ones compared to the authentic camera sensor pattern, hence the tampering ones. Our algorithm inverts the tampering maps because originally white pixels were the higher values (higher correlated). The biggest problems of this approach are:

- Time-consumption of calculation the PRNU block by block (even if it has be done one time for all), that increases when the block size decreases. For instance it took 32 hours for 15x20 size blocks.
- Time-consumption of the correlation algorithm itself (not very much but around one minute per image).
- Some results are nice like the two examples below, but many others are not. Indeed some of them are noisy or sense-less. Furthermore, a significant number of images are inverted, maybe because the tampered spot is higher correlated than the pristine part of image.

100x100	50x50	15x20 (after morphological filters)	Ground truth tampering map
Image: dev_0004.tif			

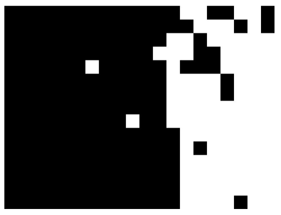
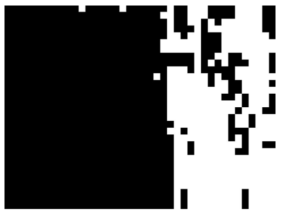
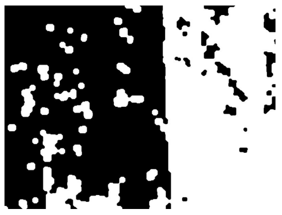
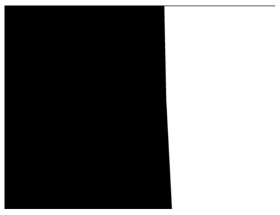
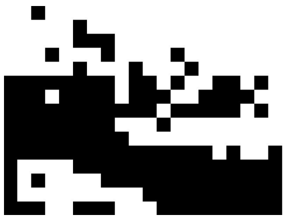
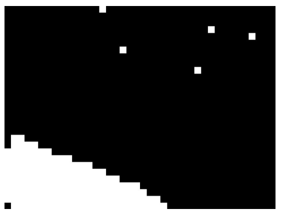
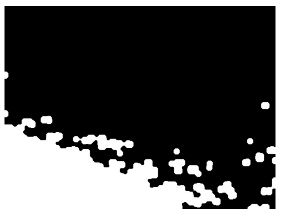
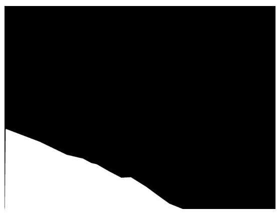

			

Image: dev_0016.tif			
			

Initially we have thought that computing the correlation based on small size block should had increased the quality of the result (as we can see going from 100x100 to 50x50), but going further in details we the noise increased: thereby, still not satisfied by the results, we have decided to find another way to compute the correlation.

2nd Approach: Phase-correlation and then overlapping blocks

In fact at a certain point we have realized that our raw PRNU-based approach was correct in principle but needed some adjustments to be more precise. Therefore, reading a suggested paper on PRNU-based forgery detection [1], we have tried to implement the phase correlation as follows:

$$\mathbf{R}_b(i,j) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{W}^b)\mathcal{F}(\mathbf{IK})^*}{|\mathcal{F}(\mathbf{W}^b)\mathcal{F}(\mathbf{IK})^*|}\right)$$

We had good expectation with regard this method, but in the end we achieved worse results, as we can see on the side. We tried in C++/OpenCv because there is a fast appropriate function (cv::phaseCorrelate()) as well in Matlab by steps, but we probably made some mistake observing the performances. In Matlab the errors were provided by a *NotANumber* issue (zeros in the denominator matrix). Anyway, we sadly came back to the old two dimensions-correlation method, changing strategy and keeping some useful tips read on the paper [1].
For instance instead of analysing separated blocks we have followed the “overlapping blocks” manner with small size blocks like 3x3. In the same way we have achieved useless results. Then after reading the paper [3] we have discovered the missing link: according to literature good size for windows are of three types: 64x64 for small tampering details, 256x256 for big ones, and 128x128 for both. Then we have chosen 128x128 as block size trade-off. In addition to the only use of the PRNU matrix computed on whole image (and not the ones made patches by patches, which had little sense), another strategic change has been the highest correlated PRNU matrix selection as first step. Then the main operation is computing two dimensions-correlation block by block (128x128) and shifting these windows pixel-wise. However overlapping blocks shifted pixel by pixel means compute a impressive number of matrix correlations, resulting in a huge amount of time consumptions (around 15 minutes for each image). So to reduce the computational load we have opted for a shift of 16 pixels step. After filling the correlation map, we have refined it with a Gaussian filter to smooth contours, binarized and inverted. We have also applied morphological filters to reduce outliers, opening and then eroding with a structured element shaped as a disk of size 16. The best results on development set obtained with this final algorithm will be shown later.

3rd Approach: JPEG-based

Since PRNU-base method gave better results for *tif* images and worse for *jpeg* images, we have decided to look for a jpeg-based analysis method. We have chosen to follow a method based on Blocking Artifact Measure, which is describe in a paper “Detecting Digital Image Forgeries by Measuring Inconsistencies of Blocking Artifact”, written by Shuiming Ye, Qibin Sun ,and Ee-Chien Chang from Singapore Institute for Infocomm Research [2]. This method turned out to be much faster on jpeg images than the one based on PRNU and gave easy on the eye results. However, average f-measure results computed for jpeg tampering maps, where only slightly better, compared to those obtained with PRNU-based approach on jpeg pictures, but it is better than nothing. In the end, our final algorithm performs Blocking Artifact Measure method for *.jpeg images while applies PRNU-based method for *.tif images. In particular, jpeg image forensic analysis is based on Blocking Artifact Measure (BAM). BAM is calculated according to the formula:

$$B(i) = \sum_{k=1}^{64} | D(k) - Q(k) round(\frac{D(k)}{Q(k)}) |$$

$$BAM = \frac{1}{N} \sum_{i} B(i)$$

where B(i) is the estimated blocking artifact for one block of testing image, D(k) is the DCT coefficient at the position k, Q(k) is the estimated quantization table and N is the total number of blocks in the image.

Image analysis starts with image segmentation into blocks, then B(i) is calculated for each block. Modified area of the image will have inconsistent B(i) according to others, which means that BAM will have significant different values in forensic areas. By detecting inconsistency of segments, it is possible to locate tampered areas.

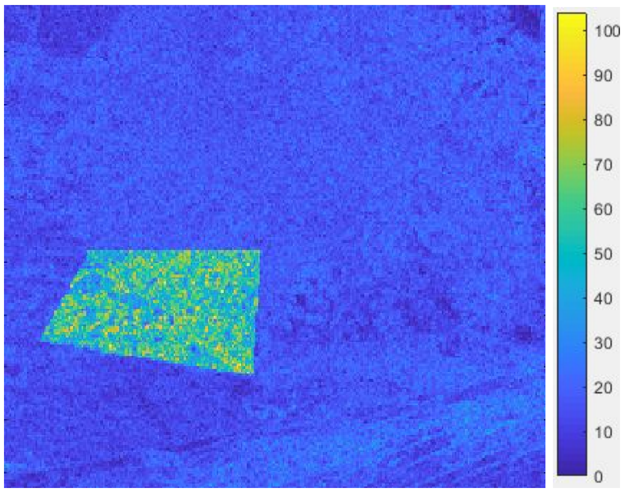


Image 1. BAM image. Yellow blocks have significantly different values, than blue ones, what indicate high probability of tampering in this location.

After histogram based thresholding and morphological operations on BAM image it is possible to separate tampered areas from original picture, as we can see below.

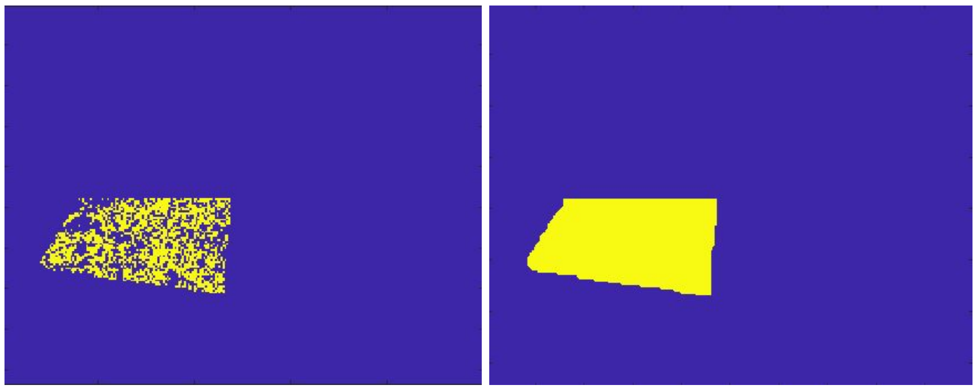
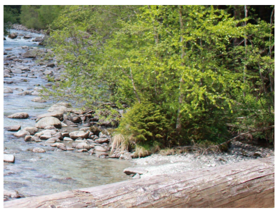
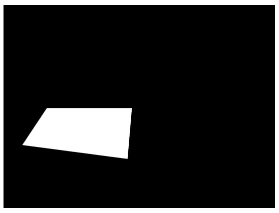
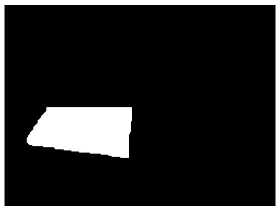
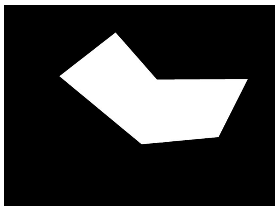
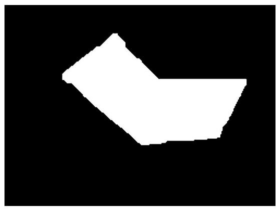
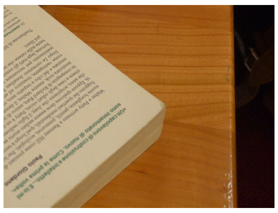
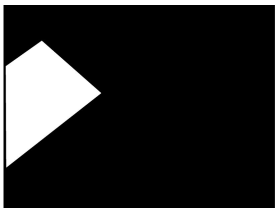
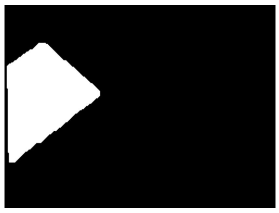


Image 2. On the left image after histogram based thresholding. On the right image after morphology operation.

Results on development set

Regarding the analysis of the obtained results during the training phase, the average F1-score computed on the development set of 800 images is 0.5176, a little above 50% accuracy. Now on we will discuss about the best three and worse three results.

Three **best** results:	Original image	Ground truth map	Calculated map
Image: dev_0001.jpg F-measure: 0.98819			
Image: dev_0025.jpg F-measure: 0.9881			
Image: dev_0050.jpg F-measure: 0.9857			

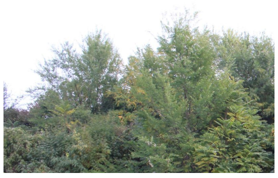
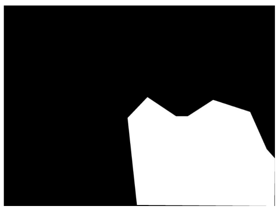
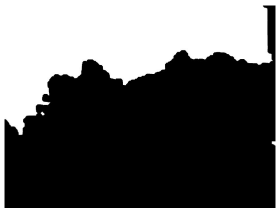
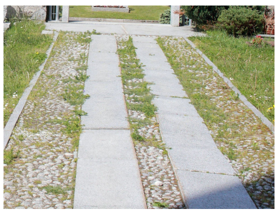
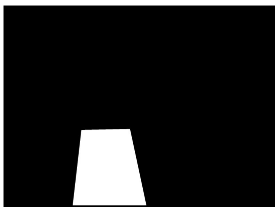
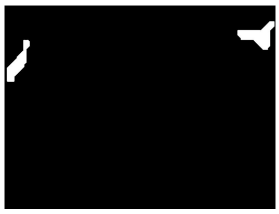
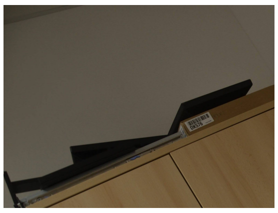
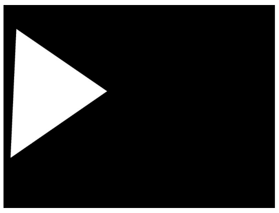
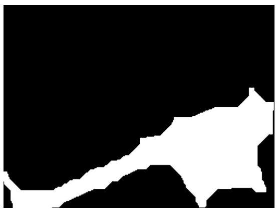
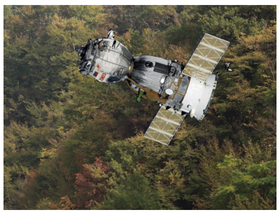
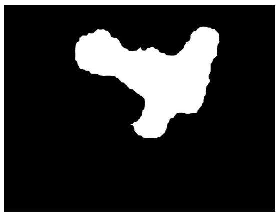
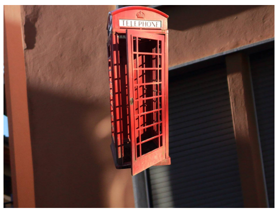
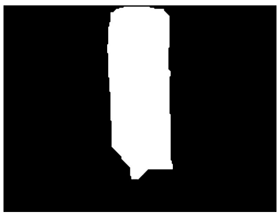
Three **worst** results:	Original image	Ground truth map	Calculated map
Image: dev_0048.tif F-measure: 0			
Image: dev_0055.jpg F-measure: 0			

Image: dev_0116.jpg F-measure: 0			

As said before, we manage to achieve average f-measure at level ~ 0.52. But this is underrepresented, by 30 images with f-measure = 0 and almost 100 images below 0.1. However more than 400 images have f-measure above 0.5. It is also interesting that our three best results are on *.jpeg images. Anyway, having almost 100 images below 0.1, may be caused by condition dominant during acquisition. Some images acquired in bad light condition are noisier than ones taken during sunny day. It makes more difficult to indicate tampering localization in the image: what is on the image has also influence on the results of tampering localization. High frequencies makes it much harder to correctly indicate tampering spots. On the other side flat areas contains a lot of noise introduced by camera sensor, what makes localization of modified areas much easier.

Results on test set and discussion

Although we could not compute f-measure on the test set provided, we could draw similar conclusions on the performance compared to the development set. In fact our algorithm performs very well in certain cases, at least on those identifiable at first sight (as we can observe in *.tif and *.jpeg examples below), whilst in others makes confusion and noise, probably because of double compression, one of the aspects related to image tampering not deeply faced by us also because of time.

Original image	Calculated map	Original image	Calculated map
Image: test_0113.tif		Image: test_0033.jpg	
			

In conclusion, working on the competition allowed us to learn different approaches in image analysis. We learnt how the *jpeg* compression works and we got knowledge about using cameras sensor noise in tampering spot localization. We have improved our English language and our team working skills, because everyone worked really hard during the competition and had to collaborate with each other, leaving no one behind. We understood our strengths and our weaknesses, trying to overcome them with commitment. We tested and discussed lot of possible solutions, before coming to the final result, and finally we gained more skills in Matlab coding.

Bibliography

[1] L. Gaborini, P. Bestagini, S. Milani, M. Tagliasacchi, S. Tubaro, Multi-clue image tampering localization, IEEE Workshop on Information Forensics and Security (WIFS), 2014.

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

[3] P. Korus, J. Huang, Multi-scale analysis strategies in PRNU-based tampering localization, IEEE Transactions on Information Forensics and Security (TIFS), 2017.

[4] Giulia Boato, Multimedia Data Security [145614], Lectures materials, University of Trento