Chapter 6
Copy-Move Forgery Detection Using Cellular Automata

Dijana Tralic, Paul L. Rosin, Xianfang Sun, and Sonja Grgic

Abstract. Thanks to the availability of many sophisticated image processing tools, digital image forgery is prevalent nowadays. One of the common methods is copy-move forgery (CMF), where part of an image is copied to another location in the same image. Detection of copy-move forgery has been widely researched recently, and many different solutions have been proposed. This chapter introduces a different approach, in which cellular automata (CA) are applied to the task of copy-move forgery detection (CMFD). The basic idea is to learn, for each overlapping block in the image, a set of CA rules that represents the intensity changes within that block. These rules are then used as features for the detection of copied blocks.

A problem arises when applying CA to image processing. If pixel intensities are used as cell states, then the large range of image intensities leads to a combinatorial explosion in the number of possible rules, making it difficult to both learn and represent rules efficiently. We describe a solution in which a reduced description of a neighbourhood is accomplished by a proper binary representation of the image based on local binary patterns (LBPs).

In the case of plain copy-move forgery, a simple 1D CA are sufficient for detection purposes, but any transformation of the copied area (for example, rotation and scaling) introduces large changes into the binary representation of the image, resulting in the need for more complicated forms of the CA's neighbourhood. However, the main issue of CMFD using CA rules is its sensitivity to processing after the copy-move operation applied to hide traces of the forgery, for example, addition of noise. Nevertheless, in some cases the CA can effectively cope with such forgeries if image pre-processing (for example, simple image filtering) is applied before forgery detection.

Dijana Tralic · Sonja Grgic
Faculty of Electrical Engineering and Computing, University of Zagreb,
Unska 3, 10000 Zagreb
e-mail: {Dijana.Tralic,Sonja.Grgic}@fer.hr

Paul L. Rosin · Xianfang Sun
School of Computer Science and Informatics, Cardiff University, Cardiff CF24 3AA
e-mail: {Paul.Rosin,Xianfang.Sun}@cs.cardiff.ac.uk

6.1 Introduction

Nowadays analogue images are completely replaced by digital images thanks to the simplicity of their acquisition, processing and storage. Many sophisticated digital image processing tools allow modification (tampering/manipulation) of digital image content to be done without any visible traces, leading to many forged images used in everyday life. Different approaches for revealing the manipulation of the content of digital image have been developed recently, with the aim to ensure the credibility of digital images. However, there is still no methodology to verify the integrity of digital images in an automatic manner. Therefore, digital image forensics [7] is an emerging research field that includes methods for determining the authenticity and the origin of digital images.

Image authentication methods can generally be classified into two main categories [6]: active and passive methods. Active methods involve embedding of some information into an image in the archiving process, such as digital watermarks [13] and signatures. Tampering of images usually destroys or modifies that embedded information, so any kind of content manipulation can be easily detected by analyzing information extracted from image. One active approach for digital image forensics is described in Chapter 7, where cellular automata are used as a tool for generating digital signatures.

Passive methods on the other hand involve checking the integrity of an image by analysis of image statistics and properties (for example, sensor noise [8], illumination conditions [10], etc.). Detection methods in this category are usually aimed at solving some special kind of tampering attempts so there is no unique technique for detection of all types of forgeries.

One common type of forgery attack is a copy-move forgery (CMF) [20] in which part of an image is copied and moved to a new location. CMF is often used because of the simplicity of performing that type of forgery. Detection of CMF can be done using different techniques, but the most basic method is based on dividing the image into overlapping blocks and calculating some set of features from every block. Those features are then used to determine similarity between blocks in image. Different feature sets were proposed for this purpose, such as average or intensity [14] or frequency coefficients [9], etc.

In this chapter, a new approach for copy-move forgery detection based on cellular automata (CA) is presented [23]. The basic idea is to use cellular automata to learn a set of rules for each sub-image block in an image. Those rules can serve as features for determining the similarity of blocks. A problem that arises when using CA with grayscale images is the large number of possible rules, which leads to an even larger number of possible combinations of those rules. A reduced representation can be accomplished by binary representation of images in a proper manner using techniques based on local binary patterns (LBP's) [16]. The problem of plain copy-move detection can be solved by applying simple 1D CA, but detection of different types of CMFs (for example, rotation or scaling of copied area) requires defining more complicated types of neighbours for CA. Also, post-processing methods such as JPEG compression or the addition of noise can reduce the accuracy of the detection algorithm.

6.2 Copy-Move Forgery (CMF)

One of the most common used digital image forgery methods is a copy-move forgery (CMF) [20], where part of an image is copied and moved to another location in the same image. The purpose of this kind of forgery is usually to hide or to add some content or object in the image. Since the forged region came from the same image, it is impossible to use some statistical properties (for example, camera noise or illumination conditions) for forgery detection because they are very well matched within the image. Taking the forged region from the same image also simplifies the forgery process because it is easier to fit the forged region into the image due to the similarity of properties of the copied region and the rest of image.

Plain copy-move forgery is a type of forgery where the copied area is translated to a new location in the same image without changing any properties of the copied area. Therefore, in that kind of forgery, there are two completely identical areas in the image which makes plain copy-move forgery detection quite easy to implement.

More complicated types of forgery can be done by transformation of a copied region before translation to a new location. Some possible transformations of copied regions are:

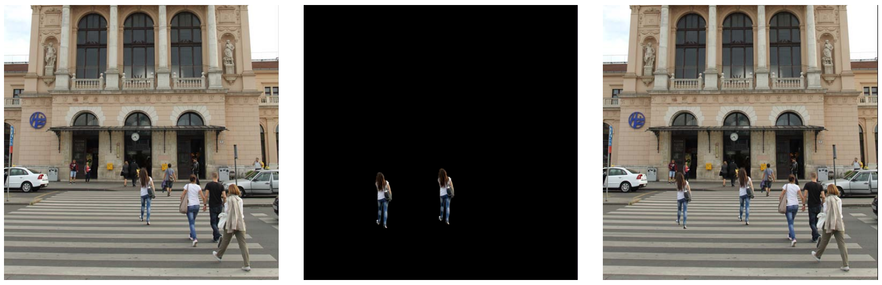
1. scaling – enlarging or shrinking of a copied area by an equal scale factor in all directions,
2. rotation – circular moving of a copied area around a centre of rotation by a arbitrary angle,
3. distortion – enlarging or shrinking of a copied area by a scale factor that is not the same in all directions,
4. combination – application of more than one transformation of a copied area.

The result of applying any of these transformations is a change in the properties of the copied area. Therefore, searching for forgeries is not as simple as in the case of plain CMF. Figure 6.1 shows some examples of CMFs from the CoMoFoD database [24].

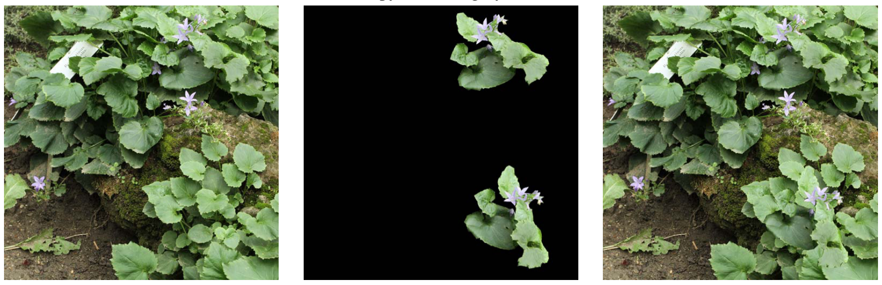
Hiding of forgery traces can be done by applying some post-processing methods. It is possible to apply a post-processing method on the whole image after forgery, but sometimes post-processing is applied only on copied region borders to assure better fitting with the new background. The most common post-processing methods used in digital image forgery are JPEG compression, addition of noise and image blurring (Fig. 6.2).

6.3 Copy-Move Forgery Detection (CMFD)

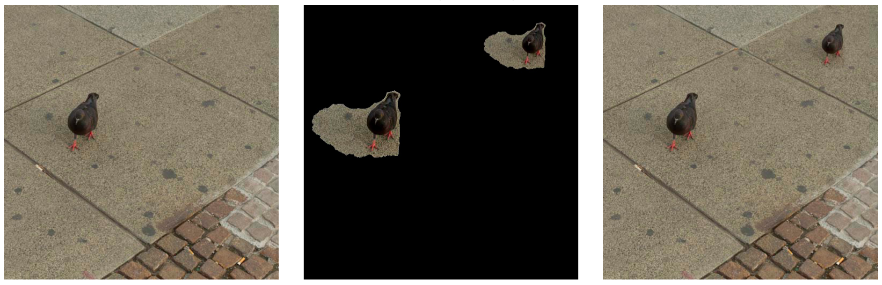
Detection of copy-move forgery has been widely researched [9]. Developed methods for copy-move forgery detection can be categorized as keypoint-based and



a Plain copy-move forgery (001_F)



b Rotation - rotation angle: 90 degrees (061_F)



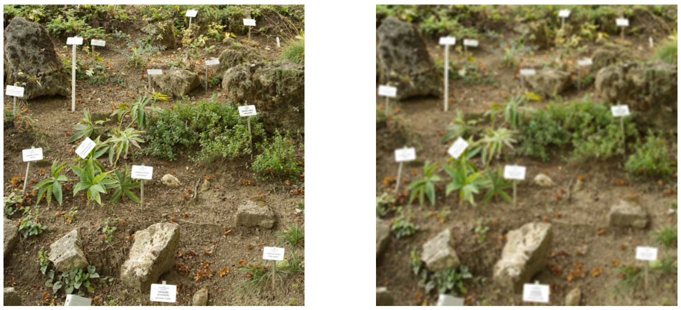
c Scaling - scaling factor: 0.7 (098_F)

Fig. 6.1 Examples of CMF (left: original image, middle: regions with copy-move version, right: forged image). The image identifiers in brackets are provided for comparison with results in Tab. 6.1.

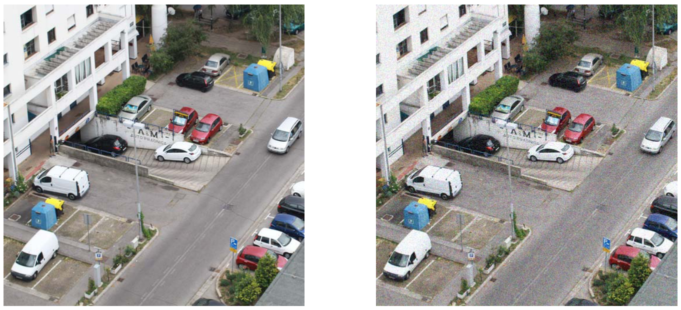
block-based methods. Keypoint-based methods include scanning of the whole image with the aim of finding points of interest (for example, point with high entropy). Those points are then analyzed to select only point with the same properties and detect similar areas in the image. Some popular examples of keypoint-based methods are SIFT (Scale-invariant feature transform) [1] and SURF (Speeded Up Robust Features) [21].

Block-based methods involve dividing an image into small overlapping blocks as a first step of the algorithm. A set of features is then calculated for every defined block, and those features are used for detection of similar blocks in the image. Different sets of features, such as DCT (Discrete Cosine Transform) [9] / DWT (Discrete Wavelet Transform) [2] coefficients, PCA (Principal Component Analysis)

a JPEG compression - quality factor: 30 (005_F)



b Image blurring - 3×3 averaging filter (018_F)



c Noise adding - zero mean, variance: 0.001 (032_F)

Fig. 6.2 Examples of postprocessing methods applied on whole image (left: original image, right: post-processed image)

[17] or Zernike moments [19], have been proposed for use in block-based methods, but the use of cellular automata for this purpose is a completely new approach.

6.3.1 Block-Based Method for CMFD

Generally all block-based copy move forgery detection approaches follow similar steps (Fig. 6.3 illustrates each step):

1. First the image is pre-processed because most algorithms require only the luminance component information, and so it is necessary to convert images to grayscale space. Sometimes a Gaussian pyramid decomposition is also applied (for example, in [25]).
2. After pre-processing, an image is divided into overlapping blocks by sliding a predefined window by one pixel through the whole image. The size of the window is usually small (for example, 8×8, 16×16, 24×24 pixels) to assure detection of areas of all sizes. Dividing an $N\times M$ image into overlapping blocks of size $b\times b$ leads to a very large number of different blocks according to equation (6.1) (for illustration: dividing a 512×512 image using a 8×8 window produces 255,025 different blocks).

$$N_b = (N-b+1)\times(M-b+1) \tag{6.1}$$

3. For every defined block a feature vector f is calculated by same method. The feature vector is used as a reduced description of a block because it contains information about shape, texture, orientation or some other properties of a block. The size of the feature vector depends on the selected method for its calculation.
4. Applying brute-force search to find similar blocks by mutual comparison of all pairs of blocks requires a lot of computational time and resources. Therefore, all feature vectors are stored in one matrix that is sorted by some algorithm (for example, lexicography sorting) to accomplish grouping of similar blocks. Beside sorting, some other methods for finding similar blocks can be applied, for example, kd-tree.
5. Neighbor feature vectors in the sorted matrix are than compared by analyzing the similarity between them, using the Euclidean distances between feature vector elements according to equation (6.2). All pairs of blocks with distance v higher than some predefined threshold T_s are removed from the set of possible results. Selection of threshold T_s depends on the type of forgery, for example, it can be set to zero for plain CMF, or it has to be adjusted to some higher values if any transformations/post-processing methods are applied. After this step only similar pairs of blocks are kept as possible results.

$$v = \sqrt{\sum_{i=1}^{size(f)} (f_1(i)-f_2(i))^2} \tag{6.2}$$

6. The set of possible results is analyzed again and Euclidean distance d is calculated between coordinates of blocks of every pair according to equation (6.3). All pairs with distance d smaller than predefined threshold T_d are removed from the set of possible results. Threshold T_d is usually defined according to selected block size (for example, $k\times b$, where k is some small positive constant) to remove all close blocks (it can be assumed that a block is moved more than T_d pixels). After this step only similar pairs of blocks that are not close to each other are kept as possible matches.

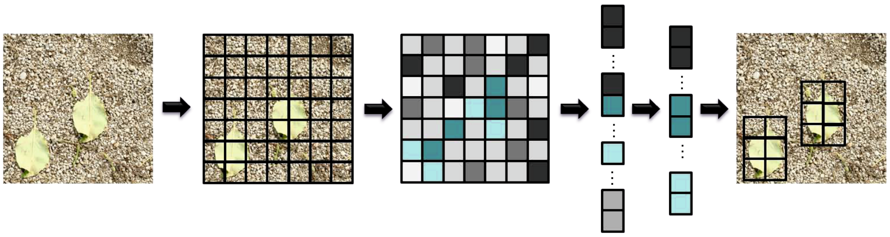


Fig. 6.3 Block-based copy-move forgery detection: 1. preprocessing; 2. overlapping blocks; 3. feature vectors; 4. sorting; 5. detection of similar blocks, 6. detection of CMF

$$d = \sqrt{(x_{f1} - x_{f2})^2 + (y_{f1} - y_{f2})^2} \tag{6.3}$$

7. The detection image is generated by marking all remaining pairs of blocks. Some simple post-processing can be applied to remove small, falsely detected areas in the image (for example, morphological opening).

6.3.2 Possible Feature Vectors

Defining an appropriate feature set is a common problem in block-based methods, because features have to yield similar results for duplicated blocks despite the transformation of the copied area or applied post-processing methods. Different sets of feature vectors for block-based CMFD have been proposed [5].

One of the first approaches used quantized frequency coefficients of the Discrete Cosine Transform (DCT) [9] as features. Thanks to the properties of DCT, it gives good results in cases of added noise, compression, and retouching. A similar approach is presented by Bashar et al. [2], where the coefficients of a Discrete Wavelet Transform (DWT) using Haar-Wavelets was introduced. Bayram et al. [3] recommended using the Fourier-Mellin Transform (FMT) for generating feature vectors.

Popescu and Farid [17] computed Principal Component Analysis (PCA) to reduce the feature set size. This representation is robust to compression and adding of noise, but any transformation of the copied area (for example, scaling, rotation) would affect the eigenvalues. Later this approach is expanded by dividing every block into 4 sub-blocks using the Discrete Wavelet Transform (DWT) [12]. A similar approach to [17] was proposed in [11], where Singular Value Decomposition (SVD) was used.

Luo et al. [15] introduced features based on the intensity of pixels in blocks. The first three values of the feature vectors included the average of the red, blue and green color components. The rest of the feature vector was defined by dividing of the block into 2 equal parts in 4 directions and calculating the ratio of each part's intensity with respect to the intensity of the whole block. Bravo-Solorio et al. [4] used the same three components as in the previous method with the addition of the entropy of a block. A similar approach is presented in [14] where every block was divided into 4 sub-blocks and the feature vector is defined as a ratio of intensities of those sub-blocks. In the circle approach, proposed in [25], the image is first reduced in dimension by Gaussian pyramid decomposition and every block is divided into four concentric circles. The feature vector is calculated as a mean of the image pixel value in each circular region of every block.

Wang et al. [26] introduced the first four Hu moments as features. The image is first reduced in dimension by Gaussian pyramid decomposition, and the Hu moments are computed from the overlapping blocks of the low-frequency image. The use of Zernike moments of degree 5 as features was proposed by Ryu et al. [19].

6.4 CA for CMF Detection

Cellular automata can be described as a discrete system that contains a regular grid of cells. Each cell can be in only one finite-state determined by the previous states of a surrounding neighbourhood of cells (reference on CA). The use of cellular automata for image processing is interesting because of the property that very simple rules can result in very complex behaviour. Detection of CMF by CA is based on the fact that similar areas in an image should produce similar rules.

The basic approach can be described as a variation of block-based methods for CMFD with a new set of feature vectors. In this approach, the feature vector for each block is obtained using a CA to generate a set of rules that describe the texture of that block [23]. This process can be described as a selection of a subset of rules from all possible rules.

Applying a CA on a greyscale image results in the combinatorial explosion in the number of possible rules, because a whole range of image intensities (256 levels) is used as cell states. For example, using a neighbourhood of 8 pixels for learning rules, leads to 256^8 possible rules. Moreover, the large number of possible rules leads to an even larger number of possible subsets of those rules, which makes it difficult to learn and represent rules efficiently. A reduced description of a neighbourhood can be accomplished by a proper binary representation of image. In that case only two values (0 and 1) are used as cells states leading to a compact description. For comparison, a neighbourhood of 8 pixels on binary image generates only 2^8 possible rules. It is clear that a binary representation of image is more suitable for this purpose, but the problem that arises is to find a proper binary representation. In the following section, we describe several possible solutions.

Another important part of CMFD with CA is the proper selection of neighbourhood. Some tasks can be solved by using simple 1D neighbourhood, but in cases of some transformations, complicated versions of neighbourhood have to be applied.

6.4.1 Representation of Image in Binary

Applying a CA to a grayscale image requires taking the whole range of intensities as a cell states which leads to a large number of possible rules, and a large number of possible subsets of rules. One way to avoid that is to represent the grayscale image as a binary image, but this must be done in a proper manner that ensures that enough information is kept for further analysis.

6.4.1.1 Image Thresholding

A grayscale image can be represented in binary by simple thresholding of pixel intensity values g_i using a predefined threshold T_b according to equation (6.4).

$$g_i = \begin{cases} 1, & g_i \geq t_b \\ 0, & g_i < t_b \end{cases} \tag{6.4}$$

Threshold T_b has to be selected according to image content with the aim of preserving sufficient information about image textures so enough descriptive information is kept for further analysis. Figure 6.4 shows a few examples of grayscale image thresholding using different thresholds. The use of smaller values (for example, 50) results in not enough details in the image, as can be seen in Fig. 6.4b. Larger values for thresholding (Fig. 6.4d) leads to loss of some details in the image. For this example, a good threshold value would be around 70, as can be seen in Fig. 6.4c. Even if using mean or median values for image thresholding appears to be an intuitive solution, global use of those values can lead to unreliable results (see Fig. 6.4e and 6.4f) where most information of image content and texture is lost.

The use of binary images obtained by simple thresholding of grayscale images as an input into CA is not the best solution for CMFD. A problem arises since applying the same (global) threshold to all pixels values results in many homogeneous regions in the binary image, especially in areas with smooth textures. A possible solution for this problem is to generate more binary images with different thresholds, so-called threshold decomposition [18], and use all of them as an input into the algorithm. The final result is then accomplished by combining (summing) the results from all binary images. Even if this approach can lead to good results, better results can be accomplished by different binary representation.

6.4.1.2 Image Binary Planes

Another way of representing a greyscale image in a binary way is to convert all pixel values into binary codes and use each plane of bits as one binary image. The result of this process is 8 different binary images, as can be seen in Fig. 6.5). Naturally, the use of the least significant bits will produce an image similar to noise (Fig. 6.5b), but that effect drops with moving to more significant bits (Fig. 6.5f– 6.5i). Binary images obtained by using more significant bit planes are more similar to images accomplished by thresholding of the greyscale image, especially the image for the most significant bit (Fig. 6.5i). According to that, the use of those images would lead to the same problem as in the case when thresholding is used. To avoid that, images obtained by using lower bits should be used. Even if they look like noise (especially Fig. 6.5b– 6.5d), they contain a detailed description of texture. Also, the lack of uniform white or black areas on those images leads to fewer falsely detected areas.

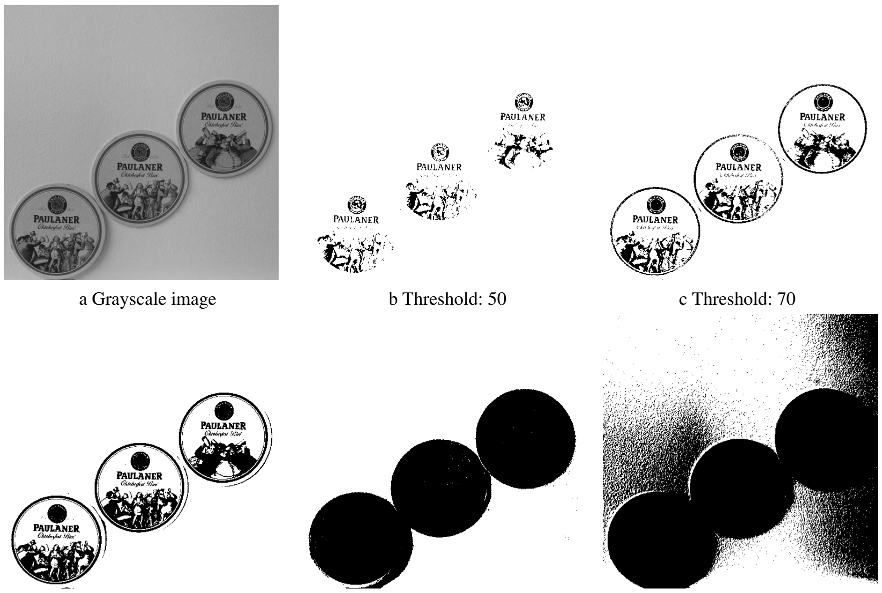


Fig. 6.4 Thresholding of a grayscale image

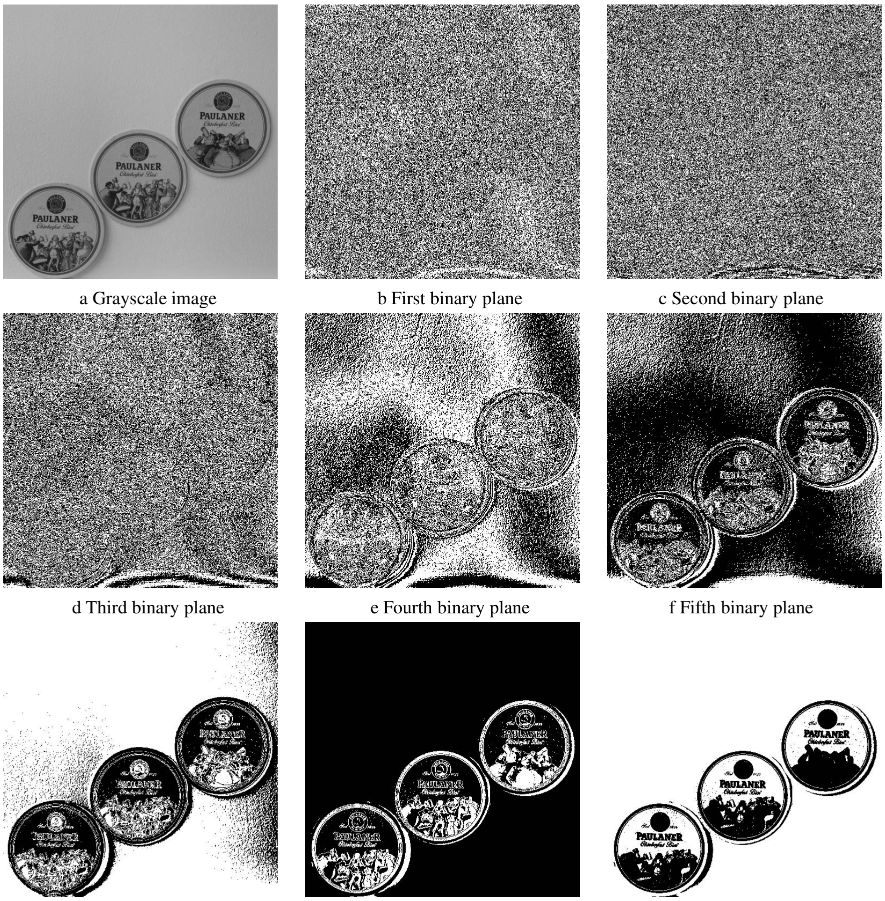
The main problem with this approach is noise sensitivity because adding even a small amount of noise significantly affects the lower bits. In order to reduce noise sensitivity, it is possible to use a binary image obtained by some higher bits. For example, in Fig. 6.5 proper results can be accomplished by using the image produced by the 4th significant bit of pixels values (Fig. 6.5e). That binary image is less sensitive to noise, but still contains enough information about image texture. However, selection of the proper binary level strongly depends on image content. For images with less homogeneous areas, higher levels could also be used as a proper representation.

6.4.1.3 Local Binary Pattern

Local Binary Pattern (LBP) is a simple and efficient texture operator obtained by thresholding the neighbourhood of central pixels and representing the result as a binary number [16]. It is simple to calculate but also has the important property of robustness to illumination change.

The LBP of neighbourhood P and radius R is obtained by using the value of central pixel g_c as the threshold for defining values of neighbourhood pixels g_p according to equation (6.5).

$$LPB(P,R) = \sum_{p=0}^{P-1} s(g_p - g_c) \times 2^p, \qquad s(x) = \begin{cases} 1, & x \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6.5}$$

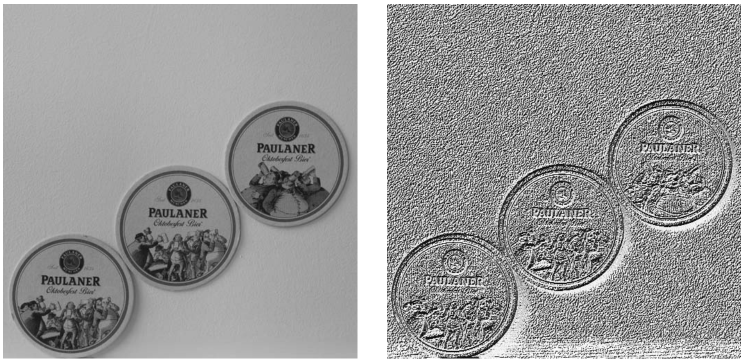


a Grayscale image; b First binary plane; c Second binary plane; d Third binary plane; e Fourth binary plane; f Fifth binary plane; g Sixth binary plane; h Seventh binary plane; i Eight binary plane

Fig. 6.5 Image binary planes

All values of neighbourhood pixels are represented by one bit according to the value of central pixel, making a binary code that is assigned to the central pixel. An example of a LBP image is shown in Fig. 6.6. It can be seen that LBP combines good properties of thresholding and binary planes, because it preserves global shapes and texture, but it also treats homogeneous region locally leaving enough information for analysis.

Although the LBP image properly presents texture, it still has the same number of intensity levels as a greyscale image (256) so it is not appropriate as image representation for cellular automata. Consequently, representation is defined using LBP but only for calculating binary values of neighbourhood pixels. Every neighbourhood is treated separately, and binary values of pixels in that neighbourhood are defined according to the difference between current pixel and the mean value of all pixels in the neighbourhood and current pixel. Those values are then used as a binary



a Grayscale image b Local binary pattern

Fig. 6.6 Local binary pattern example

representation of texture in every neighbourhood [23]. It is important to notice that representation for every neighbourhood in all blocks is based on the same technique, but it still produce different results thanks to its dependence on mean values.

6.4.2 Plain CMF

Detection of a plain copy-move forgery is the easiest task for any detection algorithm because the goal is simply to find two equal areas in the image. This refers to the fact that properties are not changed during the translation of the copied area to a new location because no transformation (for example, scaling or rotation) is used. Also, in this case we assume that no post-processing is used (handling with post-processing is described in Subsect. 6.4.3). Detection task for plain CMF can be easily solved with the simple 1D CA applied on every block of image [23]:

1. For every pixel p_c in the block define neighbourhood N for CA as a group of k pixels from the row of the image above pixel p_c. Pixels are chosen so that one pixel straight above pixel p_c and an equal number of neighbouring pixels from both sides of p_c is selected according to equation (6.6).

$$N(p_c) = N(p_{x,y}) = \{p_{x+i,y-1}, \quad i = (-\lfloor k/2 \rfloor, ..., \lfloor k/2 \rfloor)\} \tag{6.6}$$

2. For every neighbourhood calculate the mean value using the pixel p_c and its neighbour pixels' intensities. Use a mean value to threshold all pixels' values p_i to binary values b_i according to equation (6.7).

$$b_i = \begin{cases} 1, & p_i \geq mean(N(p_c) \cup p_c) \\ 0, & \text{otherwise} \end{cases} \tag{6.7}$$

3. Use the fast rule identification method proposed by Sun et al. [22] to generate a rule in the block that describes the relation between each pixel p_c and its

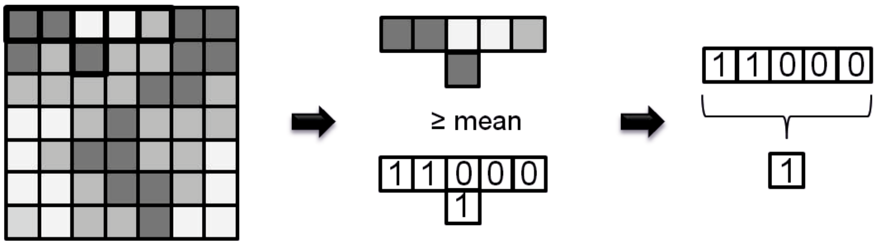


Fig. 6.7 Rule generation using 1D CA

neighbourhood. Note that Sun et al.'s method contains a step of neighbourhood selection, but we can simply ignore it because we have a fixed neighbourhood for each pixel.

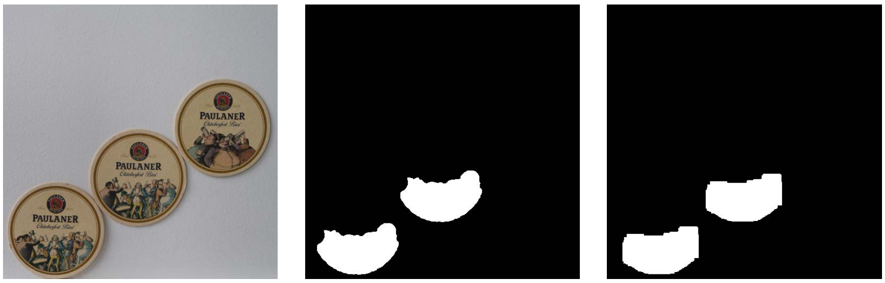
The result of this process is a subset of rules for every block in the image that describes the block's texture and is used as a feature vector to distinguish copied areas in the image. Figure 6.7 shows generation of one rule for one neighbourhood in 7×7 block of the image. Neighbourhood of 5 pixels from one row of the image is used for learning pixel p_c. Intensity values of all pixels from the neighbourhood and pixel p_c are used for calculation of mean value and thresholding of pixels intensities to binary values. After thresholding binary values of neighbourhood pixels are used for learning binary value of pixel p_c.

The use of neighbourhood of size k gives total of N_s possible rules, which leads to even higher number of possible subsets of those rules N_{ss}, according to equation (6.8). Thus, number of neighbourhood pixels should be selected considering the computational time and the accuracy of texture description.

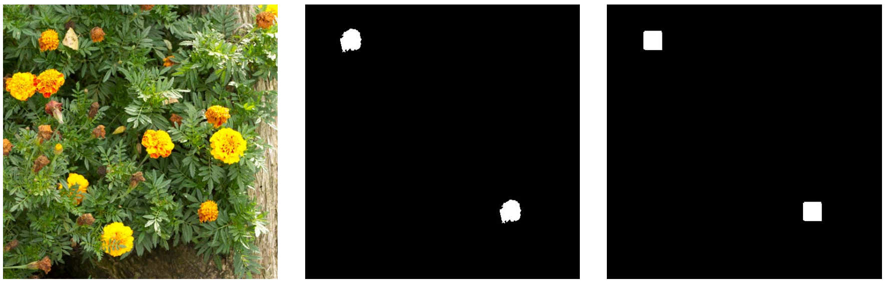
$$N_{ss} = 2^{N_s} = 2^{2^k} \tag{6.8}$$

Figure 6.8 shows a few examples of plain CMF detection on images from the CoMoFoD database [24]. A block size b of 32×32 pixels is used, and neighbourhood size k is set to 7. Values of thresholds are defined as follows: T_s = 0 and T_d = $1.5\times b$ = 48 pixels. The only pre-processing is just for conversion of RGB images into greyscale images, and no post-processing is used (for example, removal of falsely detected areas). Detection is very accurate under different conditions, such as presence of homogeneous regions (Fig. 6.8a) or complex textures (Fig. 6.8b).

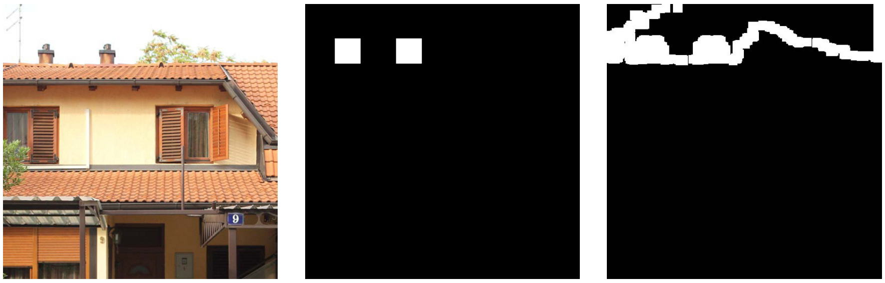
Detection is less accurate in conditions where many areas have very similar properties, for example, areas with very small differences of pixels values. Figures 6.8c and 6.8d show two examples of images where part of the area includes sky and homogeneous regions with very similar values. Beside detection of copied regions, much false detected area is also present, so detection in those cases is not satisfactory. The problem is caused by very small differences between pixels values so that in the thresholding process all blocks have very similar binary representations resulting in similar sets of rules.



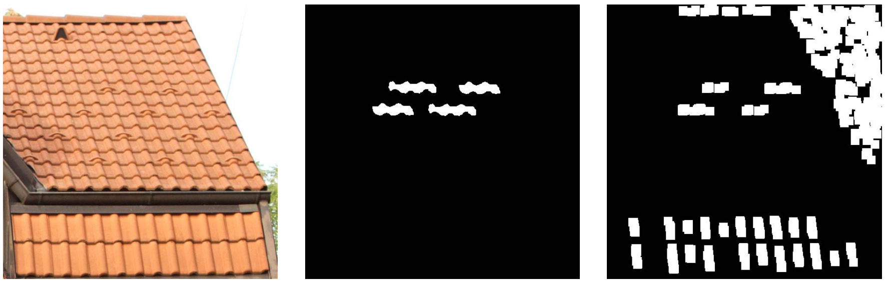
a Successful CMFD on image with homogeneous regions (005_F)



b Successful CMFD on image with complex textures (011_F)



c Problem with detection of sky (003_F)



d Problem with detection of homogeneous regions and sky (007_F)

Fig. 6.8 Plain CMFD (left: forged image, middle: expected result, right: actual result)

Table 6.1 shows detection results for 40 plain CMF images from the CoMoFoD database [24]. Block size and threshold T_d are chosen according to the image content in order to maximise the F-measure, threshold T_s is set to zero, and no post-processing is applied. Results show that 82.5% of the images have a F-measure score higher than 0.6, indicating that detection is very accurate. The main problem for detection represents homogeneous regions because in that case many blocks are falsely detected.

Furthermore, results achieved with this method are comparable with results of other CMFD methods. Lower accuracy and weaker performances are noticeable only in the case of large homogeneous regions with many pixels having similar values. In that case, better results are achieved with methods that do not require representation of the image in binary, such as DCT [9] or Zernike moments [19]. Better performance in this case can be accomplished by changing the binary representation to produce a more discriminative description of such similar areas. However, the advantage of the CA method is the very small number of false detected regions in all other tested cases, for example images with more complex textures.

6.4.3 Application on Post-processed Images

Post-processing of forged images introduces differences in pixel values resulting in differences in the binary representation and generated set of rules. One of the common problems is detection of forgery in the image after the addition of noise.

Figure 6.9 shows degradation of performance with adding of Gaussian noise on plain CMF image (simple 1D CA is used for detection). The result is accomplished using image with added noise of zero mean and variance equal to 0.00001 (image intensities were normalized to range [0,1] prior to addition of noise). Block size is set to 32×32 pixels and neighbourhood size of 7 pixels is used. Values of thresholds are $T_s = 6$ and $T_d = 1.5\times b = 48$ pixels. Conversion of the RGB image into a greyscale image is used before noise adding, and no post-processing is applied (for example, removing of false detected areas). Even when the amount of noise is very small, only part of the copied area is successfully detected (Fig. 6.9c). A larger part of the copied area can be detected by adjusting the similarity threshold for detection of similar blocks, but that would also introduce much false detected area.

Coping with noise can be done by using simple pre-processing of image in the form of filtering [23]. Figure 6.10 shows example of CMF detection on image with Gaussian zero mean noise with variance equal to 0.0001. In the case when no pre-processing is applied, detection is not possible at all (Fig. 6.10c). After application of an averaging filter of size 3×3 on noise image, detection becomes very accurate (Fig. 6.10d).

Another very common post-processing method is image blurring. Figure 6.11 shows one example of detection of blurred image accomplished by applying of 3×3 filter. It is possible to notice that detection is quite accurate because blurring does not affect pixels values in a way that changes properties of copied regions.

Table 6.1 Plain CMF detection for images from the CoMoFoD database

Image	Block size	Threshold T_d	Precision	Recall	F-measure
001_F	16	40	0.6759	0.5778	0.6230
002_F	16	35	0.7031	1.0000	0.8257
003_F	16	40	0.2062	1.0000	0.3419
004_F	16	25	0.5643	1.0000	0.7214
005_F	32	40	0.9529	0.9522	0.9526
006_F	32	40	0.9133	1.0000	0.9546
007_F	16	25	0.0766	0.7401	0.1388
008_F	any	any	0.0000	0.0000	0.0000
009_F	32	40	1.0000	1.0000	1.0000
010_F	16	25	0.0384	0.9305	0.0738
011_F	16	40	0.9133	1.0000	0.9546
012_F	16	40	0.1760	0.9930	0.2989
013_F	16	40	0.9756	0.9807	0.9782
014_F	16	40	0.8907	0.9804	0.9334
015_F	16	40	0.9822	0.9932	0.9877
016_F	32	40	0.9316	0.8868	0.9086
017_F	32	40	0.9024	0.9969	0.9473
018_F	16	40	0.8482	0.7370	0.7887
019_F	16	40	0.8392	0.5980	0.6984
020_F	16	40	0.7711	0.7607	0.7659
021_F	16	40	0.9436	0.9558	0.9497
022_F	16	40	0.9207	0.8929	0.9066
023_F	32	40	0.9201	0.7278	0.8128
024_F	32	40	0.9649	1.0000	0.9822
025_F	16	40	0.8081	0.9048	0.8537
026_F	16	40	0.9671	0.9461	0.9565
027_F	16	40	0.9295	0.9271	0.9283
028_F	16	40	0.9767	0.9444	0.9603
029_F	16	40	0.9911	0.9734	0.9822
030_F	16	40	0.9970	1.0000	0.9985
031_F	16	40	0.8873	0.8849	0.8861
032_F	16	40	0.9016	0.9394	0.9201
033_F	16	40	0.8856	0.9728	0.9272
034_F	32	40	0.9709	0.8668	0.9159
035_F	32	40	0.9609	0.9902	0.9754
036_F	any	any	0.000	0.0000	0.0000
037_F	16	40	0.0043	0.2066	0.0084
038_F	16	40	0.9114	1.0000	0.9536
039_F	16	40	0.7508	0.6961	0.7224
040_F	32	40	1.0000	0.9823	0.9911

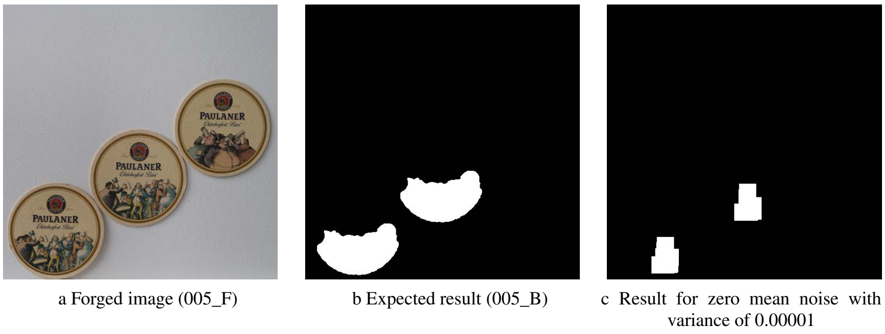

a Forged image (005_F) b Expected result (005_B) c Result for zero mean noise with variance of 0.00001

Fig. 6.9 Detection of images with noise

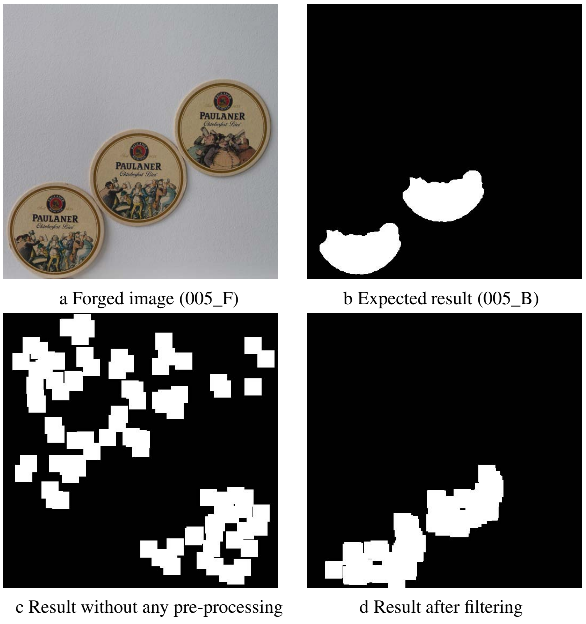
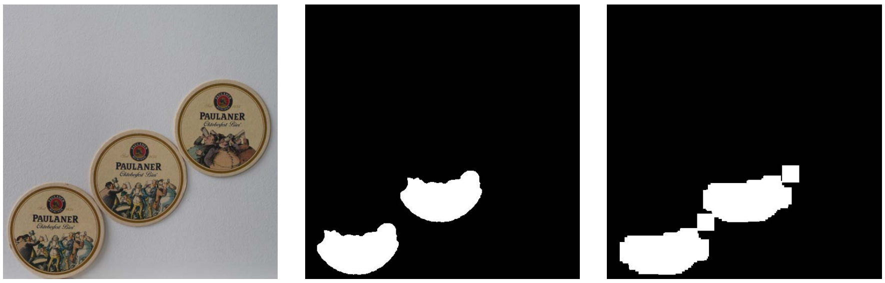

a Forged image (005_F) b Expected result (005_B) c Result without any pre-processing d Result after filtering

Fig. 6.10 Detection of images with Gaussian zero mean noise (variance=0.0001) after filtering



a Forged image (005_F) b Expected result (005_B) c Actual result

Fig. 6.11 Detection of blurred image

6.5 Future Work

The current method is developed with the aim to cope with detection of plain CMF. However, beside simple plain CMF, there are many different variations of forgery where some additional transformation of the copied region is applied. Transformation can include some linear change of size (scaling) or orientation (rotation) of the copied region. Furthermore, transformation can be done as a nonlinear function where the transformation factor is not the same in all directions. Finally, more than one transformation can be applied to the copied region.

The problem of detection of transformed copied area requires the definition of more complicated neighbourhoods. Detection of rotated regions can be solved by using a circular neighbourhood for every block of image. The learning process would remain the same as in the case when 1D CA is used. On the other hand, scaling of the copied region would require using more than one neighbourhood for each block. Those neighbourhoods should have different size to allow detection of changes in sizes of the copied region.

Furthermore, transformation of the copied region introduces small changes in pixels values due to intensity interpolation effects. Those changes do not impact the global texture of the copied region, but they affect the local thresholding process when neighbourhood binary values are defined. Coping with this problem is possible by application of more advanced thresholding methods for representation of images in a binary way.

6.6 Conclusion

Digital image forgery has become a huge problem due to the simplicity of processing of digital images. However, techniques for detection of forged images were introduced in parallel with the development of forgery attacks. Detection methods have some kinds of advantages and disadvantages, but there is still no unique method for detection of all kinds of forgeries. One very common forgery type is copy-move forgery (CMF) where part of the image is selected, copied and moved to a new location in the same image. Even if this kind of forgery is easy to carry out, successful detection is still a problem of great interest.

Thanks to its property that very simple rules can result in very complex behaviour, cellular automata (CA) are commonly used for various image processing tasks. The basic idea for using a CA for CMF detection is based on an assumption that similar areas on the image should produce similar rules. The main approach is one variation of block-based methods, where the image is divided into overlapping blocks and each block's properties are analysed. Thereby, CA are applied on every overlapping block of forged image with the aim to generate a set of rules. This process can be present as a selection of a subset of rules that describe the texture of block from all possible rules.

Application of a CA on a greyscale image leads to a large number of possible rules and an even larger number of possible subsets of those rules. Reduction of number of rules can be accomplished by a proper binary representation of image, resulting in only two possible values of cells states (as opposed to 256 in case when a greyscale image is used). Thresholding of a greyscale image by global threshold leads to binary image where much information about texture is lost, and usage of binary planes is highly noise sensitive. In order to solve these issues, a new representation of the image based on local binary pattern (LBP) is introduced. LBP defines binary values of neighbourhood pixels based on a difference between central and neighbourhood pixels. Consequently, LBP preserves local information but also keeps enough global images' information.

Detection of copy-move forgery is accomplished using simple 1D CA where the neighbourhood for every pixel is defined as a group of pixels from the row above the pixel under consideration. In our experiments, a neighbourhood size of 7 pixels showed best results according to computational time and detection accuracy. Thresholding to binary values is based on the mean value of all pixels in the neighbourhood and the pixel under consideration. The method is very reliable in different testing conditions, but there are some cases when detection is difficult, such as in the presence of homogeneous areas with small difference between pixels values. In those situations many blocks are false detected as forged. The problem is caused by the presence of similar values of pixels, leading to the same binary representation and the same set of rules for different blocks.

The presented method shows robustness to blurring of the image with averaging filters, but adding noise strongly affects the detection results. However, coping with noise is possible by a simple pre-processing using an averaging filter to smooth the image prior to applying the CA.

References

1. Amerini, I., Ballan, L., Caldelli, R., Bimbo, A.D., Serra, G.: A SIFT-based forensic method for copy-move attack detection and transformation recovery. IEEE Transactions on Information Forensics and Security 6(3), 1099–1110 (2011)
2. Bashar, M., Noda, K., Ohnishi, N., Mori, K.: Exploring duplicated regions in natural images. IEEE Transactions on Image Processing (2010) (accepted for publication)

3. Bayram, S., Sencar, H., Memon, N.: An efficient and robust method for detecting copy-move forgery. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 1053–1056 (2009)
4. Bravo-Solorio, S., Nandi, A.K.: Exposing duplicated regions affected by reflection, rotation and scaling. In: International Conference on Acoustics, Speech and Signal Processing, pp. 1880–1883 (2011)
5. Christlein, V., Riess, C., Jordan, J., Riess, C., Angelopoulou, E.: An evaluation of popular copy-move forgery detection approaches. IEEE Information Forensics and Security 7(6), 1841–1854 (2012)
6. Farid, H.: Image forgery detection: A survey. IEEE Signal Processing Magazine 26(2), 16–35 (2009)
7. Fridrich, J.: Digital image forensics. IEEE Signal Processing Magazine 26(2), 26–37 (2009)
8. Chen, M., Fridrich, J., Lukáš, J., Goljan, M.: Imaging sensor noise as digital X-ray for revealing forgeries. In: Furon, T., Cayre, F., Doërr, G., Bas, P. (eds.) IH 2007. LNCS, vol. 4567, pp. 342–358. Springer, Heidelberg (2008)
9. Fridrich, J., Soukal, D., Lukás, J.: Detection of copy move forgery in digital images. In: Proc. Digital Forensic Research Workshop (2003)
10. Johnson, M.K., Farid, H.: Exposing digital forgeries by detecting inconsistencies in lighting. In: Proc. ACM Multimedia and Security Workshop, pp. 1–10 (2005)
11. Kang, X., Wei, S.: Identifying tampered regions using singular value decomposition in digital image forensics. In: International Conference on Computer Science and Software Engineering, New York, vol. 3, pp. 926–930 (2008)
12. Li, G., Wu, Q., Tu, D., Sun, S.: A sorted neighborhood approach for detecting duplicated regions in image forgeries based on dwt and svd. In: IEEE International Conference on Multimedia and Expo, pp. 1750–1753 (2007)
13. Lin, C.Y., Wu, M., Bloom, J., Cox, I., Miller, M., Lui, Y.: Rotation, scale, and translation resilient watermarking for images. IEEE Transactions on Image Processing 10(5), 767–782 (2001)
14. Lin, H., Wang, C., Kao, Y.: Fast copy-move forgery detection. WSEAS Transactions on Signal Processing 5(5), 188–197 (2009)
15. Luo, W., Huang, J., Qiu, G.: Robust detection of region-duplication forgery in digital images. IEEE Information Forensics and Security 4, 746–749 (2006)
16. Ojala, T., Pietikainen, M., Maeenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
17. Popescu, A., Farid, H.: Exposing digital forgeries by detecting duplicated image regions. Tech. rep. tr2004-515, Dartmouth College (2004)
18. Rosin, P.L.: Training cellular automata for image processing. IEEE Transaction on Image Processing 15(7), 2076–2087 (2007)
19. Ryu, S.-J., Lee, M.-J., Lee, H.-K.: Detection of copy-rotate-move forgery using zernike moments. In: Böhme, R., Fong, P.W.L., Safavi-Naini, R. (eds.) IH 2010. LNCS, vol. 6387, pp. 51–65. Springer, Heidelberg (2010)
20. Shivakumar, B.L., Baboo, S.: Detecting copy-move forgery in digital images: A survey and analysis of current methods. Global Journal of Computer Science and Technology 10(7), 61–65 (2010)
21. Shivakumar, B.L., Baboo, S.: Detection of region duplication forgery in digital images using surf. International Journal of Computer Science Issues 8(4), 199–205 (2011)
22. Sun, X., Rosin, P.L., Martin, R.R.: Fast rule identification and neighborhood selection for cellular automata. IEEE Transactions on Systems, Man, and Cybernetics - Part B 41(3), 749–760 (2011)

23. Tralic, D., Rosin, P.L., Sun, X., Grgic, S.: Detection of duplicated image regions using cellular automata. In: International Conference on Systems, Signals and Image Processing (2014) (accepted for publishing)
24. Tralic, D., Zupancic, I., Grgic, S., Grgic, M.: Comofod-new database for copy-move forgery detection. In: Proc. 55th International Symposium ELMAR 2013, pp. 49–54 (2013)
25. Wang, J., Liu, G., Li, H., Dai, Y., Wang, Z.: Detection of image region duplication forgery using model with circle blocks, pp. 25–29 (2009)
26. Wang, J., Liu, G., Zhang, Z., Dai, Y., Wang, Z.: Fast and robust forensics for image region-duplication forgery. Acta Automatica Sinica 35(12), 1488–1495 (2009)