Combining Cellular Automata and Local Binary Patterns for Copy-Move Forgery Detection

Dijana Tralic[1], Sonja Grgic[1], Xianfang Sun[2], Paul L. Rosin[2]

[1]*University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Wireless Communications, Unska 3/XII, HR-10000 Zagreb, Croatia*

[2]*School of Computer Science and Informatics, Cardiff University, Cardiff CF24 3AA, UK*

Phone/ Fax: + 385 1 6129 567 / + 385 1 6129 717

E-mail: dijana.tralic@fer.hr

Abstract - Detection of duplicated regions in digital images has been a highly investigated field in recent years since the editing of digital images has been notably simplified by the development of advanced image processing tools. In this paper, we present a new method that combines Cellular Automata (CA) and Local Binary Patterns (LBP) to extract feature vectors for the purpose of detection of duplicated regions. The combination of CA and LBP allows a simple and reduced description of texture in the form of CA rules that represents local changes in pixel luminance values. The importance of CA lies in the fact that a very simple set of rules can be used to describe complex textures, while LBP, applied locally, allows efficient binary representation. CA rules are formed on a circular neighborhood, resulting in insensitivity to rotation of duplicated regions. Additionally, a new search method is applied to select the nearest neighbors and determine duplicated blocks. In comparison with similar methods, the proposed method showed good performance in the case of plain/multiple copy-move forgeries and rotation/scaling of duplicated regions, as well as robustness to post-processing methods such as blurring, addition of noise and JPEG compression. An important advantage of the proposed method is its low computational complexity and simplicity of its feature vector representation.

Keywords - Copy-move forgery, Duplicated regions, Cellular Automata, Local Binary Pattern

1. Introduction

Digital images are nowadays commonly used thanks to the simplicity of their acquisition, sharing, storing and editing. Many advanced processing tools allow editing of digital image without any visible traces, leading to the fact that digital images cannot be trusted any more [1]. Copy-move forgery (CMF) is one type of digital image forgery methods in which part of an image is selected, copied and moved to a new location in the same image [2] with the aim of adding or hiding an object. Digital image forensic [3], whose goal is to distinguish edited images from original images and discover any changes of image content, has two common approaches in the field of CMF detection. Active methods require the embedding of some information in the digital image in the process of its creation, such as digital signatures or watermarks [4]. Passive methods, on the other hand, do not require any additional data since they are based on analyzing properties of the image [5] such as sensor noise, illumination, statistical properties, etc. Many different passive approaches were proposed for CMF detection based on defining feature sets of small, overlapping blocks of the image [6-16] or key points [17-19].

A detection method based on Local Binary Patterns (LBP) [20] was introduced by Li et al. [15]. This approach was extended with Multi-resolution LBP (MLBP) [16]. The method showed good performance in the case of rotation, scaling, JPEG compression, blurring and noise addition. However, use of MLBP for feature set generation leads to large feature sets. To reduce the complexity of feature sets, 1D Cellular Automata (CA) [21] are used to generate feature sets as a binary array representing CA rules [22, 23]. 1D CA implies using pixels from one row of an image to learn the pixel's value in the next row of the image. However, the method was sensitive to rotation of duplicated regions due to the use of 1D neighborhoods and showed weak robustness to post-processing methods due to the applied search method based on lexicographic sorting.

This paper presents a new idea of combining LBP with CA for the purpose of CMF detection based on circular neighborhoods. Prior to the detection process, an image is divided into small, overlapping blocks. A circular neighborhood is formed for each block by defining circles of different radii around the block's central pixel and by sampling points on those circles using bilinear interpolation. The sampled points are used to form small neighborhoods in such a way that points from one angle are used to learn point value on the next angle. However, use of point values as an input to CA leads to a combinatorial explosion in the number of possible rules. To cope with that, a reduced description of point values based on LBP is applied locally on every neighborhood in each block. The feature vector is defined as a binary array where each element describes the use of a specific binary pattern. Furthermore, the Fast Library for Approximate Nearest Neighbors (FLANN) [24] is used to select nearest neighbors for every feature vector allowing better performance of the post-processing methods. A new search method based on analysis of spatial relationships between detected blocks is applied to identify true duplicated block pairs.

The rest of the paper is organized as follows. In Section 2, the brief background about CA and LBP is given. Section 3 presents a new approach with explanation of feature vector and detection method. Testing setup is given in Section 4 and experimental results are presented in Section 5. The conclusion is provided in Section 6.

2. Background

In this section, a brief introduction to cellular automata and local binary patterns is given to illustrate their role in texture description for the purpose of CMF detection.

2.1 Cellular Automata

A cellular automaton [21] is a discrete system containing a regular grid of cells whose states can be in only one finite-state set. The use of CA for image processing is interesting because of its property that very simple CA rules can result in very complex behavior, so it can be used for texture description.

A CA can be presented by a quadruple $<C, S, N, f>$ where C is a d-dimensional cellular space that consists of c cells, S is an s-value state space, N is an n-cell CA neighborhood, and $f: S^n \rightarrow S$ is a cell-state transition function [25]. Each observed cell c_o is in one finite state s_o determined by the states of a surrounding neighborhood of cells $N(c_o)$, usually spatially close to the cell c_o. The state of each cell in the next time step is defined using a transition function f, which can be represented by a set of rules. Each rule defines the next state of the observed

cell c_o corresponding to a specific combination of the neighborhood cells' values, called neighborhood pattern, and is represented as

$$\text{if} \quad N(c_o) = \{s_1, s_2, ..., s_n\} \qquad \text{then} \qquad c_o = s_o \tag{1}$$

Application of CA to digital images is possible using the following hypothesis:

- Set C is a 2-dimensional image that consists of $X \times Y$ pixels (each pixel p_i is one cell c_i so cell c will be marked with p),
- Set S contains all possible pixel values ($s = 256$ for 8-bit images or $s = 2$ for binary images);
- Set $N(p_o)$ is an arbitrarily selected group of n pixels spatially close to the observed pixel p_o,
- Function f is defined by a set of rules in a way that it represents the connection of a pattern in the selected neighborhood $N(p_o)$ and the value of the observed pixel p_o. For example, rule 0 means that all combinations of neighboring elements (s^n possible patterns) result in value of the observed pixel p_o equal to 0 (more details in Subsection 3.1).

By analyzing an image or region, it is possible to select a subset of s^n possible patterns (combinations of neighboring elements) that describes local changes of pixels' luminance values, e.g. it is possible to define a rule that can be used to generate a specific texture. Detection of duplicated regions is based on the fact that similar areas in an image should produce similar rules.

2.2 Local Binary Pattern

Local Binary Pattern (LBP) [20] is a simple texture descriptor that transforms an image into a set of labels that describe the appearance of the image luminance values. A very important property of LBP, which is based on a local pattern and texture, is gray-scale invariance. The LBP of a neighborhood P with radius r is obtained by using the value of the central pixel p_c of P as the threshold to define the values of the m neighborhood pixels located within radius r around the central pixel p_c. The binary values of the neighborhood pixels are then weighted by powers of two and summed to form a decimal number stored on the location of the central pixel p_c, which is denoted by LBP(P, r).

$$LBP(P, r) = \sum_{i=0}^{m-1} f(p_c - p_i) 2^i \text{, where } f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

Applying a CA on a gray-scale image implies using a whole range of image intensities as cell states ($s = 2^8 = 256$) leading to a large number of possible patterns ($s^n = 256^n$) and even larger number of possible rules. For illustration, use of a CA neighborhood of $n = 7$ pixels results in 256^7 possible patterns. A proper binary representation ensures that only two values (0 and 1) are used as cell states ($s = 2$), leading to a reduction of the possible patterns to 2^7 for a neighborhood of $n = 7$ pixels.

Although LBP treats regions locally, the representation still has 2^P LBP values so it is inappropriate as an image representation for CA. However, the main idea of local thresholding of image values according to a central pixel is mapped to the task of dealing with values of CA's neighborhood pixels as described in Subsection 3.1.

3. Proposed Method

The proposed method combines CA and LBP for description of image texture, i.e. it focuses on local changes in pixel luminance values. The CA rules (feature vectors) are formed by analyzing binary patterns on a circular neighborhood which are used as a reduced texture description. Binary values are generated using LBP locally on every CA neighborhood (note that the CA neighborhood differs from the circular neighborhood as described in more detail in Subsection 3.1).

Detection of duplicated image regions using the proposed method is done through the following steps:

- Pre-processing - Conversion of the image to gray-scale space is performed to adjust the image for the detection process. As an alternative to that, detection can be done for every color channel separately.
- Image subdivision - The image is divided into overlapping blocks of size $b \times b$ pixels. Due to the fact that sliding by one pixel is used, dividing an $X \times Y$ image with a $b \times b$ block gives Z blocks in total.

$$Z = (X - b + 1)(Y - b + 1) \tag{3}$$

- Feature vector formation - A description of every block, based on local changes of luminance, is accomplished by combining CA and LBP (see Subsection 3.1). A feature vector fv_i of size s^n for each block $i \in \{1, ..., Z\}$ is calculated and stored in a matrix **F** defined by

$$\begin{aligned} fv_i &= \{fv_i(1), fv_i(2), ..., fv_i(s^n)\} \\ \mathbf{F} &= \{fv_1; fv_2; ...; fv_Z\} \end{aligned} \tag{4}$$

- Duplication Detection - Feature vectors fv_i from matrix **F** are analyzed to detect duplicated blocks (see Subsection 3.2).
- Result generation - Detected pairs of blocks are marked as duplicated regions and mathematical morphology is applied to remove small regions using opening and smooth detected areas using closing.

3.1 Forming Feature Vector

After image pre-processing and subdivision, every defined block is analyzed separately with the goal to describe local changes in pixel luminance values using a reduced set of patterns. Those patterns actually represent the relationship between spatially close pixels that form its neighborhood. The main idea is to determine the frequencies of particular pattern occurrences and represent them in as simple a manner as possible (in the form of a short feature vector). Automation of this process, i.e. generating the feature vector, is possible using pixel values as input to the CA, which results in a description of generated patterns by CA rules. CA rules are formed for each overlapping block and used as feature vectors in the proposed method as described below.

3.1.1. 1D CA rules

We first use a 1D CA to illustrate the main idea of CA rules as feature vectors and highlight the main disadvantage. In the case of 1D CA, CA rules and neighborhood patterns are formed using pixels from two neighboring rows of an image. The 1D neighborhood of n pixels from one row of the image is used to define the pattern, which together with a single pixel from the next row of the image (at half the length of the neighborhood) is used to form a CA rule. The detailed rule learning process for 1D CA is presented in [22, 23] and is omitted from this paper because it is not relevant for understanding the main concept.

Table 1 represents a few 1D CA rules R for all possible combinations of CA neighborhood patterns $N(p_o)$ on a binary image. In the presented example, a neighborhood of $n = 3$ pixels above the observed pixel p_o are selected and $s = 2$ for a binary image so there are $s^n = 2^3 = 8$ binary neighborhood patterns $N(p_o)$. Moreover, there are $s^8 = 2^8 = 256$ possible rules R with values from 0 to 255. A rule's value defines the value of the observed pixel p_o for all possible neighborhood patters $N(p_o)$ (Table 1) and is used as a reduced feature vector fv_i. Note that a higher number of neighborhood pixels (n) results in more neighborhood patterns and more rules. However, the main issue in describing a block's texture by 1D CA is its sensitivity to rotation of a duplicated region due to its one dimensional neighborhood.

Table 1. Examples of a few CA rules R for all possible neighborhood patterns $N(p_o)$ on a binary image with neighborhood of $n = 3$ pixels. The value of rule R transformed into binary indicates the value of the observed pixel p_o for all possible binary combination of neighborhood patterns. For example, rule $R = 0$ means that the value of observed pixel p_o is equal to 0 for all possible combinations of neighborhood patterns $N(p_o)$.

Rule $R = 0$ (binary: 00000000)	**Neighborhood pattern $N(p_o)$**	111	110	101	100	011	010	001	000
	Observed pixel p_o	0	0	0	0	0	0	0	0
Rule $R = 60$ (binary: 00111100)	**Neighborhood pattern $N(p_o)$**	111	110	101	100	011	010	001	000
	Observed pixel p_o	0	0	1	1	1	1	0	0
Rule $R = 124$ (binary: 01111100)	**Neighborhood pattern $N(p_o)$**	111	110	101	100	011	010	001	000
	Observed pixel p_o	0	1	1	1	1	1	0	0
Rule $R = 255$ (binary: 11111111)	**Neighborhood pattern $N(p_o)$**	111	110	101	100	011	010	001	000
	Observed pixel p_o	1	1	1	1	1	1	1	1

3.1.2. Introducing a circular neighborhood

To ensure robustness to rotation, we introduced a circular neighborhood on each overlapping block using the following steps:

- j circles of radius r_j with the origin at the block's central pixel p_c are formed.
- A set of m_{rj} points is selected on each circle, located on radius r_j with angles $\alpha_i = i \cdot 360°/m_{rj}$, $i \in \{1,2,...,m_{rj}\}$. The number of selected points differs for each circle, i.e. on circles with smaller radius a smaller number of points is defined. The set of points sampled on all the defined circles in one overlapping block is called a circular neighborhood (note that CA cells c are represented by points in this case so term m will be used instead of c).
- Bilinear interpolation is used to sample each of the points using values of the closest four pixels surrounding the sampled point. The value of a sampled point is equal to a weighted average of those 4 pixels.
- Linear interpolation is used to extend a set of points sampled from each circle to get the same number of points on every circle. The number of points at each circle is expanded to equal the number of points at the largest circle.
- The selected set of points from each of the circles is transformed in such a way that points sampled at one circle form one column. Therefore, we can define that each row of a transformed set of points contains points located on different radii r_j with the same angle α_i.

The concept of circular neighborhood forming and learning process is illustrated in Fig. 1.

3.1.3. Forming CA neighborhood and CA rules

Points sampled on circular neighborhoods (and transformed into columns) are used to define CA rules (feature vector in the proposed method) that describe luminance changes in a block. To determine CA rules, the CA neighborhood is defined for each observed point separately:

- CA neighborhood $N(m_o)$ is defined for observed point m_o as a set of n points from the row above the point m_o. One point straight above m_o and an equal number of neighboring points from both sides of that point is selected as the neighborhood according to equation (5), where we use $m_{x,y}$ to represent the point m_o at position (x, y) in the transformed set of points.

$$N(m_o) = N(m_{x,y}) = \left\{ m_{x+i,y-1} \,\middle|\, i = \left(-\left\lfloor \frac{n-1}{2} \right\rfloor, \ldots, \left\lfloor \frac{n-1}{2} \right\rfloor \right) \right\} \tag{5}$$

  Therefore, the value for observed point m_o in row i, i.e. on radius with the angle α_i, is defined using neighborhood $N(m_o)$ from row i-1, i.e. on radius with the angle α_{i-1}. In the example at Fig. 1, n = 5 points from the first row, on radii with the angle α_1, are used as neighborhood $N(m_o)$ for the observed point m_o (at half the length of the neighborhood) from the second row, at radii with the angle α_2.

  Note that a circular neighborhood contains a set of points sampled on j circles in one block of an image, while a CA neighborhood is defined for each sampled point separately and include n points from all possible points. Therefore, each CA neighborhood can be observed as a subset of a circular neighborhood (more precisely, a CA neighborhood is a subset of extended set of point from a circular neighborhood).

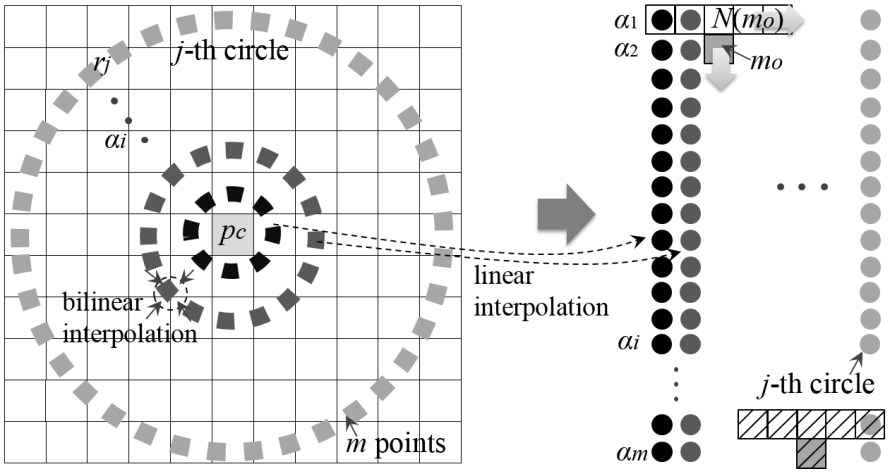


Fig. 1. Texture description using circular Cellular Automata for one block in an image: j circles are formed with radius r_j; m_{rj} points are sampled on every circle by bilinearly interpolating the closest four pixels; the set of points at each circle is extended using linear interpolation and transformed into columns; rule learning is done using points from one row (at angle α_{i-1} in the circular neighborhood) to form neighborhood $N(m_o)$ for an observed point m_o in the next row (at angle α_i in the circular neighborhood). For example, 5 points from the first row (at angle α_1) are used as neighborhood $N(m_o)$ for the observed point m_o (at half the length of the neighborhood) from the second row (at angle α_2).

- For every CA neighborhood, the mean value is calculated using the value of the point m_o and the values of all pixels from the CA neighborhood $N(m_o)$. Hereafter, we will use m_o/m_i to represent both the value of the

point m_o/m_i and the point itself without confusion.

- Thresholding based on the LBP method is applied locally to each CA neighborhood to obtain a binary representation by assigning every point m_i a binary value b_i .

$$b_i = \begin{cases} 1, & m_i \geq \text{mean}\{N(m_o) \setminus \{m_o\}\}, m_i \in \{N(m_o) \cup \{m_o\}\} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

After defining binary values, the fast rule identification method proposed by Sun et al. [25] is applied to generate a pattern that describes the relation between each point b_o and its binary CA neighborhood $N(b_o)$. The main idea proposed in [25] is to determine frequencies of specific (binary neighborhood) pattern occurrence in cases when the observed point $b_o = 1$ and $b_o = 0$, as presented below.

As described in Subsection 2.1, selection of n neighboring points for binary CA neighborhood results in 2^n possible patterns, so two vectors of size 2^n are generated: v_1 to store the number of patterns for $b_o = 1$ and v_0 to store the number of patterns for $b_o = 0$. The number of specific pattern is stored at location $l=\{0, 1, \ldots, 2^n\text{-}1\}$ in vectors v_1 and v_0 that is determined as the decimal value of binary CA neighborhood $N(b_o)$ of the observed point b_o.

$$\begin{aligned} v_1(l) &= |N(b_o)|_{b_0=1} \\ v_0(l) &= |N(b_o)|_{b_0=0} \\ l &= \sum_{k=1}^{n} 2^{k-1} N_{b_o}(k) \quad \text{where} \quad N_{b_o} = N(b_o) \end{aligned} \quad (7)$$

For CA neighborhood of $n = 5$ points, vectors v_1 and v_0 have total of $2^5 = 32$ elements, where locations $l = \{0, 1, \ldots, 2^n\text{-}1\}$ represent number of patterns $N(b_o) = \{$‘00000’, ‘00001’, …, ‘11111’$\}$. For example, vector $v_1(0)$ contains number of patterns $N(b_o)$ = ‘00000’ when $b_o = 1$, and vector $v_o(0)$ contains number of patterns $N(b_o)$ = ‘00000’ when $b_o = 0$.

Finally, a feature vector fv_i is formed by comparing the number of specific pattern $N(b_o)$ in cases when the observed point $b_o = 1$ and $b_o = 0$, i.e. by comparing values from vectors v_1 and v_0 at each location l:

$$fv_i(l) = \begin{cases} 1, & v_1(l) \geq v_0(l) \\ 0, & \text{otherwise} \end{cases} . \quad (8)$$

Figure 2 illustrates the process of forming a feature vector fv_i with a neighborhood of $n = 5$ elements. The feature vector fv_i contains $2^n = 2^5 = 32$ values defined as described above. For example, for binary pattern $N(b_o)$ = ‘00000’, the values of vectors v_1 and v_0 are determined and stored at location $l = 0$ that corresponds to the selected binary pattern. If $v_1(0) < v_0(0)$, then feature vector $fv_i(l) = fv_i(0) = 0$. In contrast to that, values of vectors v_1 and v_0 for binary pattern $N(b_o)$ = ‘10100’ are stored at location $l = 20$. In this case, $v_1(20) > v_0(20)$ so feature vector $fv_i(l) = fv_i(20) = 1$. Each value in the feature vector is defined in the same way and is set to:

- 1 if most times when the pattern $N(b_o)$ appears, the value of point b_o is 1,
- 0 if most times when the pattern $N(b_o)$ appears, the value of point b_o is 0.

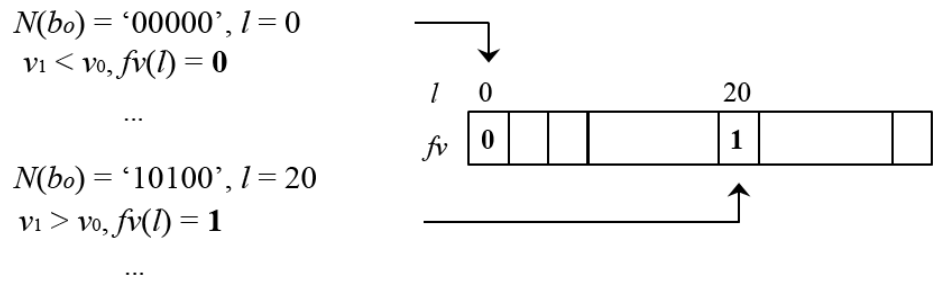


Fig. 2. Example of forming a feature vector (for one block in an image): number of each binary pattern $N(b_o)$ is determined in the case when $b_o = 1$ and $b_o = 0$; determined values are stored in vectors v_1 and v_0 at location l; location l is determined by conversion of binary pattern $N(b_o)$ to its decimal number; feature vector fv at location l is set to 1 if $v_1(0) > v_0(0)$ or to 0 is $v_1(0) < v_0(0)$. In this example, binary pattern $b_o = 00000$ at location $l = 0$ appears more often in the case when $b_o = 0$ ($v_1 < v_0$), therefore $fv(0) = 0$. Elements at other locations in feature vector fv are determined in the same way.

All generated feature vectors are stored in matrix **F** in such a way that the i-th row contains 2^n values of feature vector fv_i. Therefore, matrix **F** contains Z feature vectors for each overlapping block defined in the analyzed image.

3.2 Detection Process

3.2.1 Identifying k nearest neighbors

Having in mind the large number of overlapping blocks (for an image of size 512×512 pixels, and block of size $b = 13$, the number of overlapping blocks is $Z = 250\,000$), brute force searching for duplicated blocks would be highly time consuming, and therefore inefficient. Brute force searching means that each overlapping block of an image is selected in turn, and all the other blocks are compared with it. In case of a CMF, brute force is not effective because the search area is the whole matrix **F** of Z elements, e.g. every block has to be compared with Z-1 blocks. To improve the efficiency of the detection process, a sorting algorithm can be applied to matrix **F** to group similar feature vectors and search can then be done on a subset of the data.

The described feature vectors have an important property - each element shows use of a particular binary pattern and therefore has the same level of importance for the description of texture. Use of a sorting algorithm which gives higher importance to some elements in a feature vector (e.g. lexicographic sorting which gives higher importance to first elements in feature vectors) is not suitable for this problem, so selection of similar feature vectors is done using a *kd*-tree. Due to the fact that feature vectors contain only binary values, FLANN [24] is used to find the k most similar feature vectors for all the feature vectors (i.e. k most similar blocks for every defined block in an image).

FLANN hierarchically decomposes the search space (matrix of all feature vectors **F**) by clustering the input set and constructing trees. The tree building process divides feature vectors into K clusters by selecting K feature vectors at random as the cluster centers and assigning the rest of the feature vectors to the closest center. The

algorithm is recursively repeated until the number of feature vectors in each leaf is below a certain threshold (called the maximum leaf size ls_{max}). At the end, every cluster contains one non-leaf node (cluster center) and leaf nodes with input feature vectors to be matched. The process of searching multiple clustering trees in parallel starts with a single traverse of each of the trees. The algorithm selects the feature vector closest to the query feature vector and recursively explores it, while adding the unexplored feature vectors to a priority queue. After each of the trees has been explored once, the search is continued by extracting from the priority queue the closest node to the query feature vector and resuming the tree traversal from there. The search ends when the number of examined feature vectors exceeds a maximum limit, i.e. the degree of approximation d_{aprox}. A higher value of d_{aprox} results in the more exact neighbors, but it also leads to more expensive search. Thanks to the hierarchical clustering trees, FLANN accomplishes significant speedups over linear search ranging between one and two orders of magnitude for search precisions in the range 50-99% [24].

Application of FLANN to the matrix **F** gives k nearest feature vectors for each feature vector fv_i, i.e. it results in a matrix of $Z \times k$ possible duplicated feature vectors **P**, where row i defines k nearest feature vectors for i-th feature vector fv_i.

3.2.2 Identifying duplicated blocks

To identify true duplicated blocks, matrix **P** is further analyzed using following steps:

- For each row i Euclidean distance v_{ij} between values of feature vector fv_i and its k nearest feature vectors $fv_j = \mathbf{P}(i, j), j = \{1, 2, \ldots, k\}$ from i-th row is calculated using (9), and stored in matrix **V**.

$$v_{ij} = \sqrt{\sum_{l=1}^{2^n} (fv_i(l) - fv_j(l))^2} \tag{9}$$

  Value v_{ij} represents similarity of two feature vectors, i.e. similarity of two generated CA rules, so its analysis assures removal of blocks that differ in more binary patterns than allowed (defined by similarity threshold T_s).

- Then, matrix **P** is analyzed by calculating the Euclidean distance d_{ij} between blocks' coordinates (x_i, y_i) of feature vector fv_i (corresponding to i-th row of matrix **P**) and coordinates (x_j, y_j) of its k nearest feature vectors $fv_j = \mathbf{P}(i, j), j = \{1, 2, \ldots, k\}$ from i-th row using (10), and stored in matrix **D**.

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{10}$$

  Value d_{ij} represents the spatial distance between two blocks so its analysis allows removal of spatially close blocks (especially partly overlapping blocks) which usually have very similar texture, and produce very similar feature vectors. Its analysis assures removal of blocks that are spatially closer then allowed (defined by distance threshold T_d).

- Elements $\mathbf{P}(i, j)$ from matrix **P** are removed from further analysis if distance v_{ij} from matrix **V** is higher than threshold T_s or distance d_{ij} from matrix **D** is smaller than distance threshold T_d. After this step every row in matrix **P** contains between 0 and k elements.

- Matrix **P** is used to form matrix **I** that contains identified pairs of feature vectors in a way that every remaining element in matrix **P** from i-th row is assigned to corresponding feature vector fv_i. Therefore, a two-column matrix **I** is created using indexes $i, j \in \{1, 2, \ldots, Z\}$ of identified pairs of feature vectors fv_i, fv_j.

- Finally, set **I** is analyzed using the shift vector between blocks so that block pair fv_i, fv_j is marked as truly

duplicated if at least n_s spatial neighbors of block fv_i are copied to any n_s spatial neighbors of block fv_j. Spatial neighborhood $N_s(fv_i)$ for feature vector fv_i is determined using feature vector index i from matrix **I**:

$$N_s(fv_i) = \left\{ fv_{i-Z_X-2}, fv_{i-Z_X-1}, fv_{i-Z_X}, fv_{i-1}, fv_{i+1}, fv_{i+Z_X}, fv_{i+Z_X+1}, fv_{i+Z_X+2} \right\}, \quad Z_X = X - b \tag{11}$$

where X is the size of an image in the horizontal direction, and b is the size of overlapping blocks. Note that spatial neighborhood $N_s(fv_i)$ represents Moore neighborhood. All pairs of blocks fv_i, fv_j that do not meet this condition are removed from matrix **I**.

After this process, the detection result is generated by marking true duplicated blocks using coordinates of remaining feature vectors in matrix **I** and applying post-processing.

4. Testing Setup

To analyze the performance of the proposed method 80 plain CMF examples and 80 rotation CMF examples from a recent benchmark database called CoMoFoD is used [26]. All images are post-processed by applying JPEG compression, noise and blurring, making total of 1360 images. Every forgery example consists of an original image (without any forgery), a forged image (with a forgery) and two masks (colored and black/white) that indicate the forgery. Images in the CoMoFoD database are 512 × 512 pixels, and all forgeries are generated using a Photoshop tool. The advantage of this dataset is that it contains examples of forgeries with different sizes of duplicated regions, homogenous/heterogeneous areas and cases of multiple forgeries. Thanks to its small image size, the database is adequate for fast and efficient testing of different detection approaches.

Parameters used for implementation of the proposed method are given in Table 2. A larger block size b makes it impossible to detect duplicated regions smaller than $(b+1) \times b$ pixels. However, a smaller block size results in a larger number of overlapping blocks. Therefore block size is selected having in mind different sizes of duplicated regions and the computational complexity of a proposed method.

Table 2. Parameters used for implementation of the proposed method

Parameter	Symbol	Value
Block size	b	13
Similarity threshold	T_s	7
Distance threshold	$T_d = floor(sqrt(2b^2))$	18
Number of nearest neighbors	k	4
Number of spatial neighbors	n_s	2
Number of circles	$j = (b\text{-}1)$	12
Radii of circles	r_j=1/2, 1, 3/2, …, $(b\text{-}1)/2$	1/2, 1, 3/2, …, 6
Number of points on each circle	$m_{r1} = 8$, $m_{rj} = 2m_{rj\text{-}1}$	8, 16, …, 16384
Radius of disk element for morphology	r_{morp}	3
Number of clusters in FLANN	K	8
Maximum leaf size in FLANN	ls_{max}	64
Degree of approximation in FLANN	d_{aprox}	0.9

Values of threshold T_d, number of circles j, radii r_j, number of points m_{rj} and size of a disk element for morphological operations r_{morp} are selected according to block size (Table 2). Note that circles are formed with

radii r_j to increase the number of circles in block $b \times b$ and to ensure a sufficient number of samples for a better description of the block pattern.

Similarity threshold T_s is experimentally determined by selecting 10000 blocks of size $b \times b$ from 10 images. On each block, transformations (scaling with different factors and rotation with different angles) and post-processing operations (blurring, addition of noise and JPEG compression) are applied. CA rules are obtained for all variations of each block by the process described in Subsection 3.1. Results showed that most rules for the same block differ in less than 7 elements so threshold T_s is set to 7. The same experiment showed that in the case of post-processing and transformations, spatially close blocks have more similar rules than true duplicated blocks. Therefore we set $k = 4$ to select four blocks with similar rules for further analysis. The number of spatial neighbors n_s defines that at least two neighbor blocks have to be grouped to form duplicated areas.

Values for FLANN (K, $ls_{\max}$, d_{aprox}) are used as proposed in [24] for binary set of data.

To evaluate the performance of the proposed method, precision P, recall R and F-measure F are calculated at the pixel level for every image:

$$P = \frac{T_p}{T_p + F_p}, \; R = \frac{T_p}{T_p + F_n}, \; F = \frac{2PR}{P + R} \tag{11}$$

where F_p is the number of false positive pixels (mistakenly detected as copied), F_n is the number of the false negative pixels (mistakenly detected as not copied), and T_p is the number of the true positive pixels (correctly detected as copied).

5. Detection Results

The accuracy of CMF detection is tested for plain CMF, multiple CMF, rotation of duplicated regions and three common types of post-processing: blurring, addition of noise and JPEG compression. Plain CMF refers to forgery where a part of an image is copied and translated to a new location in the same image without changing any properties. Multiple CMF refers to a case when one or more regions are copied to different locations in the same image. The authors in [2] evaluated 13 block-based and two keypoint-based methods for CMFD. Among all tested methods, the DCT [6], PCA [14], Zernike [12] and SURF [17, 18] methods gained the best results in most tested scenarios so the proposed method is compared with those four methods.

5.1 Plain CMF

Figure 3 shows the detection results for plain CMF. The accuracy is quite high for all presented cases ($F >$ 0.92) showing that the method is capable of detecting duplicated regions of different sizes and shapes. Figure 3a and 3c illustrate that the proposed method can deal with repetitive image content. Figure 4 contains examples of multiple CMFD when one region is copied to different locations (Fig. 4a and 4b) and different regions are copied to different locations (Fig. 4c and 4d). Detection is satisfactorily accurate for all images leading to the conclusion that the proposed method can deal with multiple CMF. Additionally, Fig. 4a presents successful detection of duplicated homogeneous regions ($F = 0.9847$). The lowest F-measure (among all tested images) is achieved in the case when the copied region is very small with respect to the whole image (e.g. the signs' stands

in Fig. 4c).

(a) 360 pixels, $F = 0.9946$ (b) 1651 pixels, $F = 0.9005$ (c) 2122 pixels, $F = 0.9206$ (d) 15478 pixels, $F = 0.9631$

Fig. 3. Detection of plain CMF for different sizes of copied regions

(a) $F = 0.9847$ (b) $F = 0.9392$ (c) $F = 0.7105$ (d) $F = 0.9131$

Fig. 4. Detection of multiple CMF: one region on two locations (a, b), two regions on different locations (c, d)

Results for 40 examples of plain CMF are shown in Fig. 5. Results for original images are omitted because all methods correctly detect all 40 images ($F \approx 1$ in all cases). For almost all images, the proposed method showed better performance than the other tested methods (the highest average F-measure). The F-measure is higher than 0.7 for all images showing that detection is very accurate. Furthermore, detection is successful for different sizes of copied areas – the smallest successful detected area is around 0.13% of the image size (360 pixels, Fig. 3a), and the largest is around 14% of the image size. The Zernike method produces many false positives in homogeneous regions, while the PCA method failed to detect some small copied regions. The SURF method is unable to discriminate repetitive image content from true duplicated regions and falsely detects homogeneous duplicated regions.

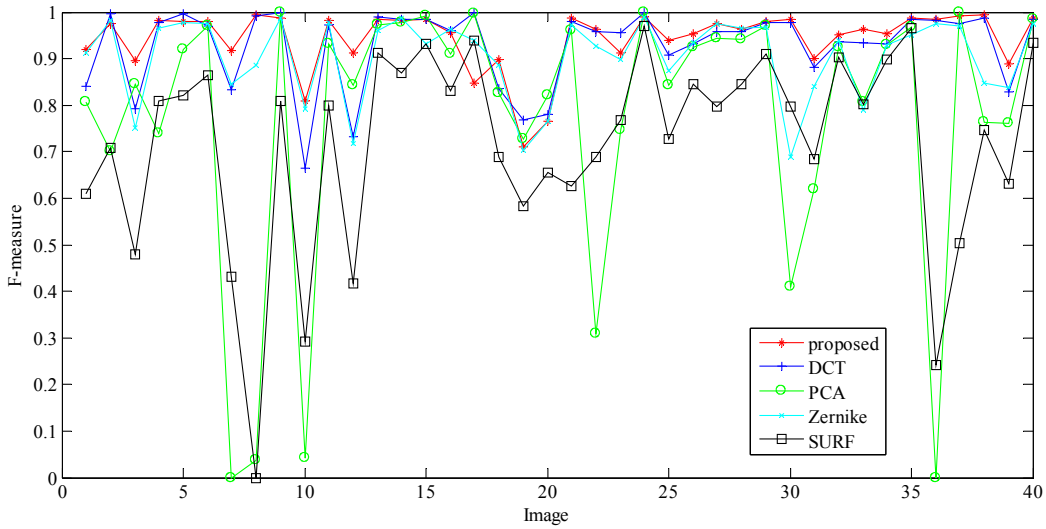


Fig. 5. Value of F-measure for 40 plain CMF images from CoMoFoD dataset with average F-measure: $F_{proposed} = 0.9428$, $F_{DCT} = 0.9272$, $F_{PCA} = 0.7715$, $F_{Zernike} = 0.9036$, $F_{SURF} = 0.7181$

5.2 Rotation of Duplicated Regions

Figure 6 shows detection result for cases when a copied region is rotated by 90° (Fig. 6a), 7° (Fig. 6b) and 180° (Fig. 6c), and one case of multiple CMF where the region is rotated by 2° and 3° (Fig. 6d). In all cases the copied regions are correctly detected demonstrating that the proposed circular neighborhood can deal with different angles of rotation. Note that there are no falsely detected areas in any of the presented cases. Also, detection is quite successful for complex textures (Fig. 6a, F =0.7384), as well as for homogeneous regions (Fig. 6d, F = 0.8097).

(a) α = 90°, F = 0.7384 (b) α = 180°, F = 0.6956 (c) α = 7°, F = 0.5688 (d) α_1 = 2°, α_2 = 3°, F = 0.8097

Fig. 6. Examples of detection results for rotation of the copied region for different angle α

Results for 80 images and all tested methods are shown in Fig. 7. Testing is done on images with rotation angles α = {1°, 2°, 5°, 7°, 10°, 40°, 90°, 180°}. The DCT method successfully detects rotation by small angles (α < 5°). The Zernike method showed good performance for almost all angles, while the PCA method was completely unable to handle higher rotation angles (α > 3°). The SURF method exhibits the most stable F-measure but it is significantly lower than the Zernike and the proposed method. The proposed method showed high F-measures for α < 10° as well as for α = 90° and α = 180°. For other tested cases detection was less accurate but it was still possible to partly detect copied regions for most rotation angles.

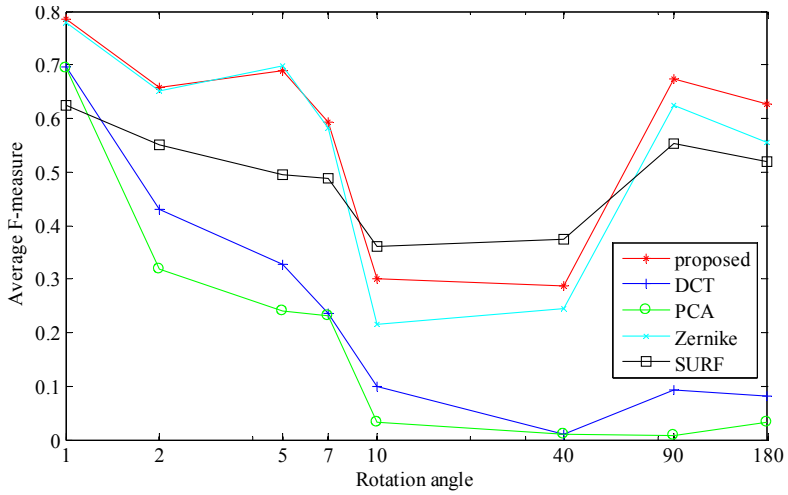


Fig. 7. Average F-measure for 80 images and different rotation angles (note that testing is done only for the highlighted angle values)

5.3 Scaling

Figure 8 shows detection results for cases when a copied region is scaled by the scaling factor f. Note that factor f has the same value in all directions, e.g. scaling is uniform. Scaling with factor f = 109 % (the copied

region is 9 % larger than the original region) is illustrated in Fig. 8a. Detection is only partly possible ($F \approx 0.5$), but no falsely detected regions are introduced, leading to easy identification of duplicated objects. An example of scaling a large region with scaling factor $f = 91$ % (the copied region is 9 % smaller than the original region) is presented in Fig. 8b. Detection is also only partly possible, but it clearly indicates duplicated regions. Figure 8d contains an example of multiple CMF when the duplicated region is scaled with factor $f_1 = 76$ % and with factor $f_2 = 111$ %. The F-measure is satisfactory high ($F = 0.6131$) indicating that detection is quite successful even in a case of higher scaling factors. Also, both copied regions are successfully detected with no falsely detected areas.

(a) $f = 109$ %, $F = 0.5309$ (b) $f = 91$ %, $F = 0.5549$ (c) $f = 95$ %, $F = 0.5681$ (d) $f_1 = 76$ %, $f_2 = 111$ %, $F = 0.6131$

Fig. 8. Examples of detection results for scaling of the copied region for different factor f

Average F-measures for 40 images and different scaling factors are given in Fig. 9. Testing is done for scaling factors $f = \{91\ \% - 109\ \%\}$ with a step of 2 and for $f = \{50\ \%, 80\ \%, 120\ \%, 200\ \%\}$. The proposed method demonstrated good capabilities to handle a moderate amount of scaling with high F-measure for $91\ \% < f < 109\ \%$. For the higher amounts of scaling, detection accuracy rapidly decreases. Other block-based methods, namely Zernike, PCA and DCT, showed similar behavior but for most scaling factors they gained a lower average F-measure. Opposite to that, the SURF method remained stable across the whole scaling range leading to the conclusion that keypoint-based methods perform better for scaling factors higher than 9 %.

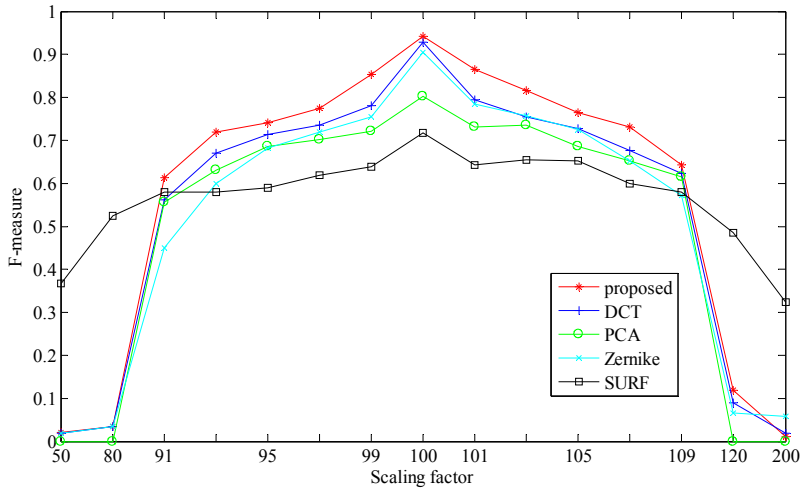


Fig. 9. Average F-measure for 80 images and different scaling factors (note that testing is done only for the highlighted values of scaling factor)

5.4 Blurring

CMFD on blurred images is done for two cases: an averaging filter of size 3 × 3 and 5 × 5, as shown in Fig. 10. In the first case (Fig. 10a and 10c), detection is almost perfect even after blurring. However, in the second

case, filtering with an averaging filter of size 3 × 3 results in successful detection of forged regions but also introduces an additional falsely detected area (Fig. 10b). Note that the falsely detected area is very similar to the duplicated areas (letter ”o”). Applying a 5 × 5 filter to the same image reduces the size of detected regions and introduces some other falsely detected blocks (Fig. 10d). Those blocks can be removed by increasing the threshold T_d.

(a) 3 × 3, $F = 0.9686$ (b) 3 × 3, $F = 0.8820$ (c) 5 × 5, $F = 0.9629$ (d) 5 × 5, $F = 0.5603$

Fig. 10. Examples of detection results on blurred images with different averaging filter

Detection results for all the methods are shown in Fig. 11 which illustrates the average F-measure for 80 original and forged images and both testing cases. The proposed method gained the highest F-measure for both cases, meaning that it can correctly detect most blurred images. Other tested methods were also successful in detection of blurred images but they introduced more falsely detected areas than the proposed method. Also, the Zernike method over-detected homogeneous regions, while the SURF method showed over sensitivity on low contrast regions after image blurring.

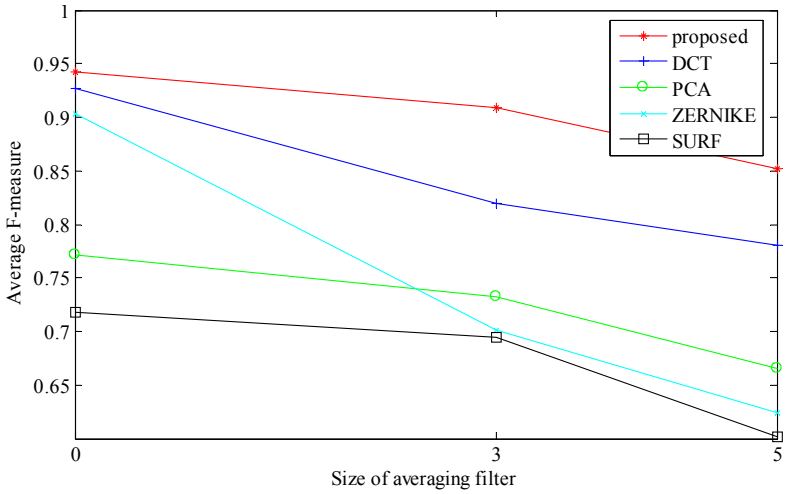
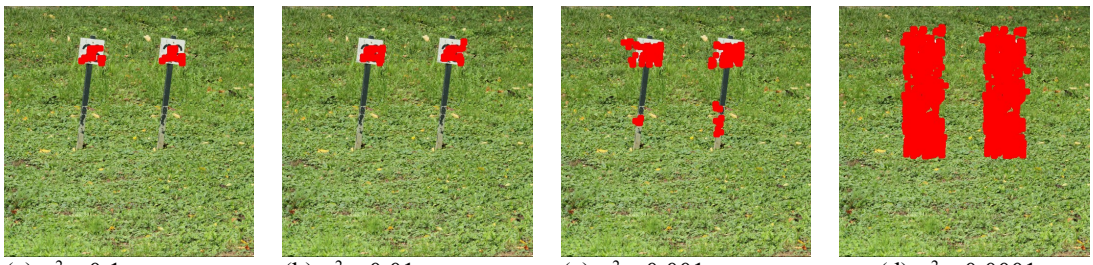


Fig. 11. Average F-measure for 80 images and different size of averaging filter

5.5 Addition of Noise

Addition of noise randomly changes properties of duplicated image regions so in that case it is not sufficient to search for two blocks with the same properties. Therefore, filtering the image with an averaging filter of size 3 × 3 is applied prior to the detection. Figure 12 contains an example of detection on a noisy image which contains added Gaussian noise with zero mean and different values of variance. Note that image intensities were normalized to the range [0, 1] prior to the addition of noise. Detection is partly possible even when a large amount of noise is added (Fig. 12a). However, note that even a large amount of noise does not introduce any falsely detected areas, which is a very important feature for digital image forensics.

Figure 13 contains results for 80 images when Gaussian noise of zero mean and different values of variance is added (0.0001 - 0.1, with multiplier step equal to 10). For a smaller amount of noise, the Zernike method was slightly more successful in detection of duplicated areas, but for larger amounts of noise the proposed method achieved the best performance in comparison to the other tested methods.



(a) $\sigma^2 = 0.1$, $F = 0.1189$ (b) $\sigma^2 = 0.01$, $F = 0.1195$ (c) $\sigma^2 = 0.001$, $F = 0.4014$ (d) $\sigma^2 = 0.0001$, $F = 0.9570$

Fig. 12. Example of detection results on a noisy image for different values of variance σ^2

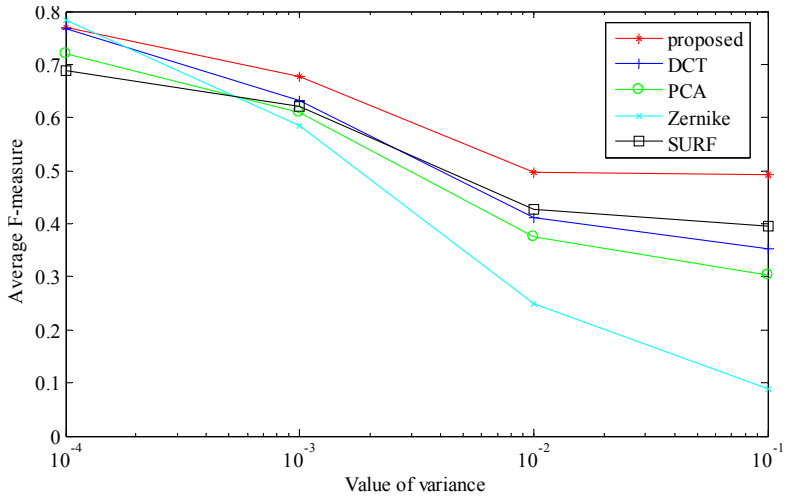


Fig. 13. Average F-measure for 80 images and different values of variance

5.6 JPEG compression

Detection of duplicated images after JPEG compression presents a serious problem for most detection algorithms because of its specific compression process where every 8×8 block of the image is treated separately. The consequence is that the same blocks have completely different binary representations, and so different sets of rules are generated. Also, JPEG blocks are smaller than the overlapping blocks defined in the proposed method (b = 13). However, as in the case of added noise, dealing with JPEG compression is possible by pre-processing the image using an averaging filter. Figure 14 presents detection results for a forgery example with different JPEG quality factors. Accuracy of detection of duplicated areas rapidly decreases for higher levels of JPEG compression. However, even for images with high JPEG compression, detection is partly possible, and there are no falsely detected areas (Fig. 14a).

Results for all tested methods are presented in Fig. 15, where the average F-measure for 80 test images is given. The proposed method was equally successful in detection of JPEG images as the DCT method for higher quality factors, while for lower JPEG quality the DCT method showed slightly better performance. The SURF method was most stable but it also gained a lower F-measure for most JPEG compression factors in comparison with the

proposed method. Additionally, the proposed method outperformed other tested methods for all amounts of JPEG compression.

(a) q = 30, F = 0.2381 (b) q = 50, F = 0.4009 (c) q = 70, F = 0.6010 (d) q = 80, F = 0.8104

Fig. 14. Examples of detection results on a JPEG image with different values of quality factor q

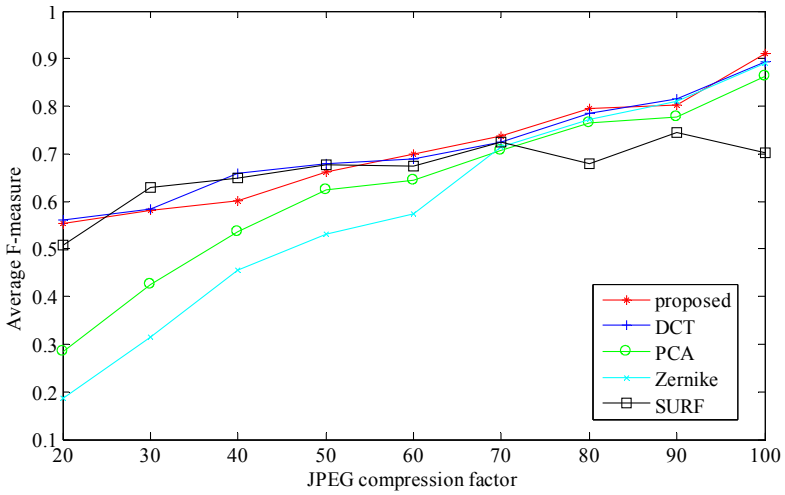


Fig. 15. Average F-measure for 80 images and different values of JPEG compression factor

5.7 Feature Vector Complexity

An important part of every CMF detection algorithm is the description of features using reduced and robust feature vectors. Table 3 shows the size of the calculated feature vectors for all tested methods. The size of feature vectors for the DCT and PCA methods (as well as computational complexity) depends on block size and image content, respectively. The size of the Zernike feature vector is defined by Zernike moment order o, and it is equal to 12 for o = 5 [12], however it is computationally demanding. The size of SURF feature vectors (64 values) does not depend on any method specific parameters. Although SURF produces large feature vectors, it contains a smaller number of feature vectors than any block-based method. In the proposed method, the size of feature vector depends on the size of the CA neighborhood (2^n). However, each feature vector contains only binary values so it can be represented as a single decimal number. In comparison with other methods, this property allows a significantly simplified description of texture. Computational complexity depends on the number of sampling points and the size of neighborhood but it is not more demanding than other block-based methods.

Table 3. Properties of generated feature vectors for all tested methods

Group	Method	Size of feature vector	Dependence of parameter
Block-based	Proposed	128 $_{(b)}$ = 1 $_{(d)}$	Neighborhood size *n**
	DCT [6]	256 $_{(d)}$	Blocks size *b*
	PCA [14]	/	Image content
	Zernike [12]	12 $_{(d)}$	Zernike moment order *o*
Keypoint-based	SURF [17, 18]	64 $_{(d)}$	/

* influence only on size of the feature vector in binary form

6. Conclusion

Detection of duplicated image regions has been widely researched in the past few years due to the fact that digital image content can be easily manipulated. Therefore we presented a new block-based method for detection of duplicated image regions that combines LBP with CA to accomplish a powerful pattern description. Identification of duplicated regions is accomplished by analyzing local changes of pixel luminance values in a circular neighborhood. Pixel values are transformed to binary values using LBP to form a reduced representation of a block and the binary values are used as an input to CA. The produced feature vector indicates the use of a specific set of patterns in the block texture, so similar image areas should produce similar feature vectors. FLANN is applied to the feature vectors set to find the *k* nearest neighbors for every element and a new search method is applied to select the duplicated blocks.

Testing results showed excellent performance in the case of plain CMF and multiple CMF, where the proposed method outperformed the DCT, PCA, Zernike and SURF methods. While the DCT method gained similar F-measures as the proposed method, the PCA method was unable to detect small duplicated regions. Furthermore, the Zernike method generated a lot of falsely detected areas on homogeneous regions (e.g. sky) and the SURF method falsely marked a large amount of repetitive image content due to background similarities. Detection of rotated regions was quite successful for most rotation angles in which the proposed method gained similar accuracy as the Zernike method. The SURF method showed most stable detection but with lower average F-measure, and the PCA and DCT methods successfully detected only at small rotation angles. In detection of scaling, the proposed method showed similar behavior to other block-based methods, but with a slightly higher average F-measure thanks to better robustness to homogeneous regions and repetitive image content. The keypoint-based method showed the lowest change in detection accuracy, but it also achieve the lowest average F-measure for almost all scaling factors. Furthermore, the proposed method showed good robustness to blurring, addition of noise and JPEG compression. It gained the best average F-measure for blurred images and images with added Gaussian noise, and it showed similar accuracy as the DCT method for JPEG compression.

An important advantage of the proposed method is its binary coded feature vectors that can be represented as a

single number, in contrast to all previous proposed methods. A simple descriptor of local luminance changes is applicable to classification tasks thanks to its low computational complexity and possibility for fast and efficient analysis. The strength of the proposed method lies in the description of local changes of pixel luminance values so it is not significantly affected by image post-processing. Additionally, the circular neighborhood assures insensitivity to rotation of the duplicated region. Use of a new search method allows better analysis of similarities between calculated feature vectors. The proposed method has low computational complexity and low memory requirements, and it produces a significantly more reduced description of image texture than all previously proposed methods.

References

[1] Farid H, Image Forgery Detection: A Survey, IEEE Signal Processing Magazine, vol. 26, no. 2, pp 16–35, 2009

[2] Christlein V, Riess C, Jordan J, Riess C, Angelopoulou E, An Evaluation of Popular Copy-Move Forgery Detection Approaches, IEEE Transactions on Information Forensics and Security, vol. 7, no. 6, pp 1841–1854, 2012

[3] Redi JA, Taktak W, Dugelay J-L, Digital image forensics: a booklet for beginners, Multimedia Tools and Applications, vol. 51, iss. 1, pp 133-162, 2011

[4] Bhatnagar G, Wu QMJ, Raman B, A new aspect in robust digital watermarking, Multimedia Tools and Applications, vol. 66, iss. 2, pp 179-200, 2011

[5] Hou X, Zhang T, Xiong G, Zhang Y, Ping X, Image resampling detection based on texture classification, Multimedia Tools and Applications, vol. 72, iss. 2, pp 1681-1708, 2014

[6] Fridrich J, Soukal D, Lukas J, Detection of Copy Move Forgery in Digital Images, Proc. Digital Forensic Research Workshop, 2003

[7] Bashar M, Noda K, Ohnishi N, Mori K, Exploring Duplicated Regions in Natural Images, IEEE Transactions on Image Processing, 2010 accepted for publication.

[8] Bayram S, Sencar H, Memon N, An Efficient and Robust Method for Detecting Copy-Move Forgery, pp 1053–1056, IEEE International Conference on Acoustics, Speech, and Signal Processing, 2009

[9] Luo W, Huang J, Qiu G, Robust Detection of Region-Duplication Forgery in Digital Images, IEEE Information Forensics and Security, vol. 4, pp 746–749, 2006

[10] Bravo-Solorio S, Nandi AK, Exposing Duplicated Regions Affected by Reflection, Rotation and Scaling, in International Conference on Acoustics, Speech and Signal Processing, pp 1880–1883, 2011

[11] Lin H, Wang C, Kao Y, Fast Copy-Move Forgery Detection, WSEAS Transactions on Signal Processing, vol. 5, no. 5, pp 188–197, 2009

[12] Ryu S, Lee M, Lee H, Detection of Copy-Rotate-Move Forgery using Zernike Moments, in Information Hiding Conference, pp 51–65, 2010

[13] Wang J, Liu G, Zhang Z, Dai Y, Wang Z, Fast and Robust Forensics for Image Region-Duplication Forgery, Acta Automatica Sinica, vol. 35, no. 12, pp 1488–1495, 2009

[14] Popescu A, Farid H, Exposing Digital Forgeries by Detecting Duplicated Image Regions, tech. rep. tr2004-515, Dartmouth College, 2004

[15] Li L, Li S, Zhu H, Chu S, Roddick J, Pan J, An Efficient Scheme for Detecting Copy-move Forged Images by Local Binary Patterns, Journal of Information Hiding and Multimedia Signal Processing, vol. 4, no. 1, pp 46–56, 2013

[16] Swaminathan A, Min W, Liu K, Copy-move forgery detection using multiresolution local binary patterns, Forensic Science International, vol. 231, no. 1 - 3, pp 61 – 72, 2013

[17] Shivakumar BL and Baboo S, Detection of Region Duplication Forgery in Digital Images Using SURF, International Journal of Computer Science Issues, vol. 8, no. 4, pp 199–205, 2011

[18] Bo X, Junwen W, Guangjie L, Yuewei D, Image Copy-Move Forgery Detection Based on SURF, in Multimedia Information Networking and Security, pp 889–892, 2010

[19] Amerini, I, Ballan, L, Caldelli, R, Del Bimbo, A, Giuseppe, S, "A SIFT-Based Forensic Method for Copy-Move Attack Detection and Transformation Recovery," IEEE Transactions on Information Forensics and Security, vol. 6, no. 3, pp. 1099–1110, 2011.

[20] 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, vol. 24, no. 7, pp 971–987, 2002

[21] Rosin PL, Training Cellular Automata for Image Processing, IEEE Transaction on Image Processing, vol.

15, no. 7, pp 2076–2087, 2007
[22] Tralic D, Rosin PL, Sun X, Grgic S, Detection of Duplicated Image Regions using Cellular Automata, in Proceeding of International Conference on Systems, Signals and Image Processing, pp 167–170, 2014
[23] Tralic D, Rosin PL, Sun X, Grgic S, Copy-Move Forgery Detection Using Cellular Automata. In: Rosin P L, Adamatzky A, and Sun X (eds) Cellular Automata in Image Processing and Geometry, pp. 105 – 125, Springer, 2014
[24] Muja, M, Lowe, DG, Fast Matching of Binary Features, in Computer and Robot Vision (CRV), pp 404–410, 2012
[25] Sun X, Rosin PL, Martin RR, Fast Rule Identification and Neighborhood Selection for Cellular automata, IEEE Transactions on Systems, Man, and Cybernetics - Part B, vol. 41, no. 3, pp 749– 760, 2011
[26] 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