Detection of Near-Duplicated Image Regions

Babak Mahdian[1] and Stanislav Saic[2]

[1] Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, Břehová 7, 115 19 Prague 1, Czech Republic
mahdian@utia.cas.cz

[2] Academy of Sciences of the Czech Republic, Institute of Information Theory and Automation, Pod Vodárenskou věží 4, 18208 Prague 8, Czech Republic
ssaic@utia.cas.cz

Summary. Modern, easy to use image processing software enables forgeries that are undetectable by the naked eye. In this work we propose a method to automatically detect and localize near-duplicated regions in digital images. The presence of near-duplicated regions in an image may signify a common type of forgery called copy—move forgery. The method is based on blur moment invariants, which allows successful detection of copy—move forgery, even when blur degradation, additional noise, or arbitrary contrast changes are present in the duplicated regions. These modifications are commonly used techniques to conceal traces of copy—move forgery. Our method works equally well for lossy format such as JPEG.

1 Introduction

In this work we focus on detecting a common type of digital image forgery, called copy—move forgery. In copy—move forgery, a part of the image is copied and pasted into another part of the same image, with the intention to hide an object or a region of the image. Figure 1 shows an example. We can determine whether an image contains this type of forgery by detection of duplicated regions. Duplicated regions may not always match exactly. For example, this could be caused by a lossy compression algorithm, such as JPEG, or by possible use of the retouch tool.

Existing copy–move forgery detection methods are mostly based on matching of overlapping image blocks. For example, Fridrich et al. [4] has proposed a method which is based on matching the quantizied lexicographically sorted discrete cosine transform (DCT) coefficients of overlapping image blocks. The lexicographically sorting of DCT coefficients is carried out mainly to reduce the computational complexity of the matching step. Another method has been proposed by Popescu and Farid [5] and is similar to [4]. This method differs from [4] mainly in the representation of overlapping image blocks. Here, the principal component transform (PCT) has been employed in place of DCT. The representation of blocks by this method has better discriminating features.

As pointed out in [4], ideal regions for using copy—move forgery are textured areas with irregular patterns, such as grass. Because the copied areas will likely blend with the background it will be very difficult for the human eye to detect

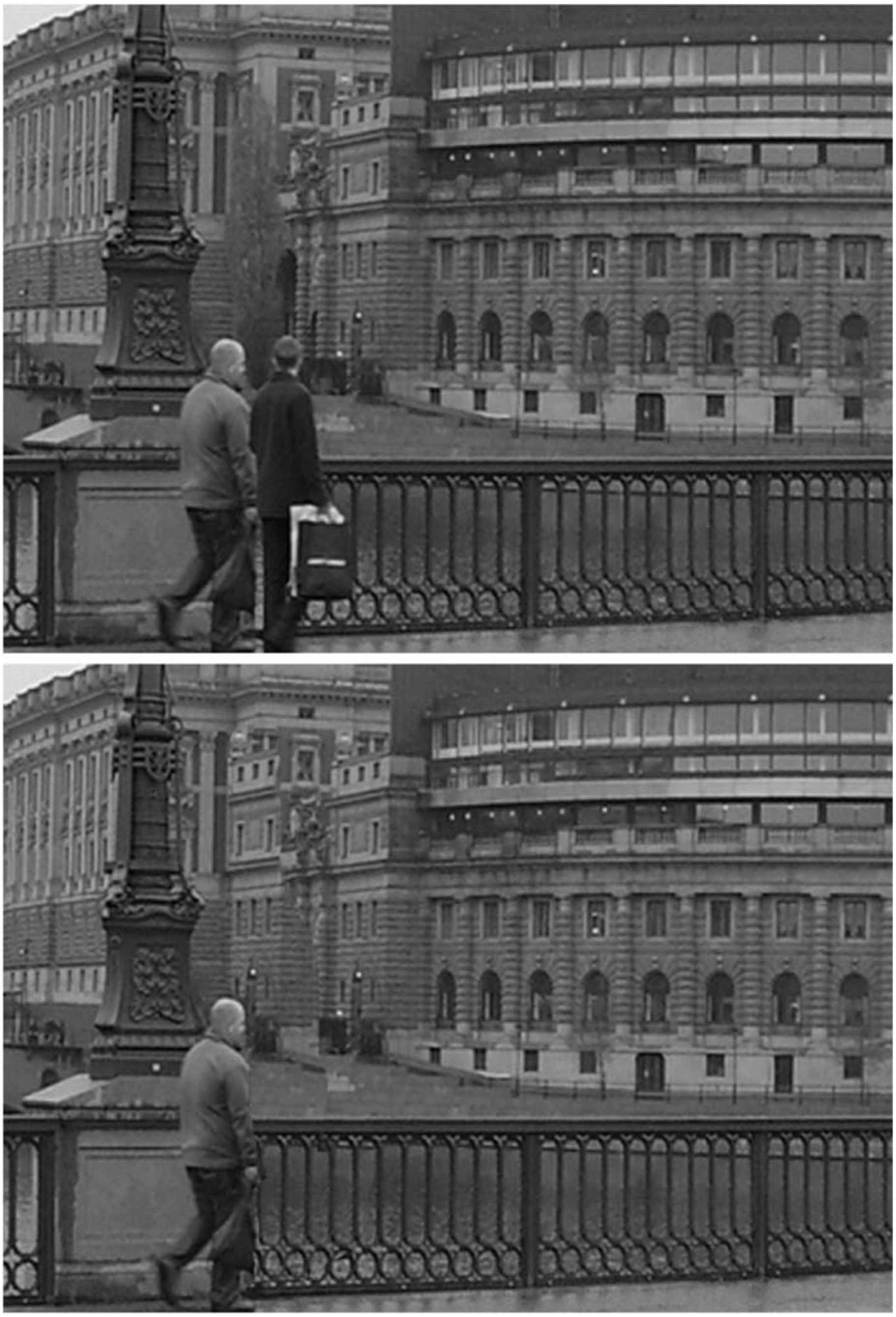

Fig. 1. An example of a copy—move forgery. The original (left) and forged version (right).

any suspicious artifacts. Another fact which complicates the detection of this type of tampering is that the copied regions come from the same image. They therefore have similar properties, such as the noise component or color palette. It makes the use of statistical measures to find irregularities in different parts of the image impossible.

2 Detection of Near-Duplicated Regions

To detect the copy—move forgery we focus our aim on detection of near-duplicated regions in the image. Existing copy—move forgery detection methods have limited abilities. In most cases of forgery investigated, they were able to detect duplicated regions in the tampered image despite of the presence of an acceptable amount of noise. This is mainly caused due to a quantization step or a similarity threshold. Additionally, it allows for analysis images compressed with a lossy algorithm, such as JPEG. However, a skilled falsifier will be able to produce a work undetectable by these methods.

This can be, for instance, achieved easily by blurring. An experienced falsifier can use a simple 2D convolution of the duplicated region with a blur filter mask to make detection of forgery even more difficult. Thus, to improve the detection abilities of the current available approaches, we can describe analyzed regions by some features invariant to the presence of unknown blur. From a mathematical point of view, we are looking for a functional B, which is invariant with respect to blur degradation. In other words, B satisfies the condition $B(f) = B(D(f))$, where operator D denotes the blur degradation. Furthermore, due to the fact that the falsifier can also use additive noise to make detection more difficult, these invariants should also work well with the presence of additive noise.

The aforementioned requirements are satisfied by blur moment invariants. They have been previously addressed by Flusser and Suk [1, 2] and have found successful applications in many areas of image processing such as: in face recognition on out-of-focused photographs, template-to-scene matching of satellite images, in focus/defocus quantitative measurement, etc. An advantage of moment invariants is that they are computed by a summation over the whole image, so they are not significantly affected by additive zero-mean noise.

We will define the problem of copy—move forgery detection in the following way. Given an image $I(x, y)$ containing an arbitrary number of near-duplicated regions of unknown location and shape, our task is to determine the presence of such regions in the image and to localize them. The aim of this investigation is create a method that can detect duplicated regions, even when some contain degradations caused by convolution with a shift-invariant symmetric energy-preserving point spread function (PSF) and additive random noise. Formally: let $f(x, y)$ be a function describing the original region and $g(x, y)$ the acquired region created by the falsifier via convolution of $f(x, y)$ with the PSF, then $g(x, y) = (f * h)(x, y) + n(x, y)$, where $h(x, y)$ is a shift invariant PSF, $n(x, y)$ an additive random noise and $*$ denotes a 2D convolution.We would like to find all $g(x, y)$ created from $f(x, y)$ and $h(x, y)$ via the above equation. Due to the fact that moment invariants are utilized as features, we will assume the following restrictions. Both $f(x,y) \in L_1$ and $g(x,y) \in L_1$ are real functions and have a bounded support and nonzero integral. Moreover, the PSF is assumed to be axial symmetric and energy-preserving, i.e.: $h(x,y) = h(-x,y) = h(y,x)$ and $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)dxdy = 1$. These assumptions do not cause a significant limitation. Most imaging systems that we are interested in perform some type of symmetry. By supposing other types of symmetries, like central, four-fold or circular symmetry, we can also construct blur invariants based on moments. However, generally, the higher degree of symmetry of the PSF is assumed, the more invariants can be obtained [1].

The proposed copy—move forgery detection method is based on a few main steps: tiling the image with overlapping blocks, blur moment invariants representation of the overlapping blocks, principal component transformation, k—d tree representation, blocks similarity analysis and near-duplicated regions map creation. Each step is explained separately in the following sections.

2.1 Overlapping Blocks

This method begins with the image being tiled by blocks of $R \times R$ pixels. Blocks are assumed to be smaller than the size of the duplicated regions, which have to be detected. Blocks are horizontally slid by one pixel rightwards starting with the upper left corner and ending with the bottom right corner. The total number of overlapping blocks for an image of $M \times N$ pixels is $(M-R+1) \times (N-R+1)$. For instance, an image with the size of 640×480 with blocks of size 20×20 yields 286281 overlapping blocks.

2.2 Blur Invariants Representation

Each block is represented by blur invariants, which are functions of central moments. The two-dimensional $(p+q)$th order moment m_{pq} of image function $f(x,y)$ is defined by the integral:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy$$

The two-dimensional $(p+q)$th order central moment μ_{pq} of $f(x,y)$ is defined as

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-x_t)^p (y-y_t)^q f(x,y)dxdy$$

where the coordinates (x_t, y_t) given by the relations $x_t = m_{10}/m_{00}$, $y_t = m_{01}/m_{00}$ denote the centroid or the center of gravity of $f(x,y)$. By supposing that $g(x,y) = (f * h)(x,y)$, we can simply derive that central moments of $g(x,y)$ are defined as $\mu_{pq} = \sum_{k=0}^{p}\sum_{j=0}^{q}\binom{p}{k}\binom{q}{j}\mu_{kj}^{(f)}\mu_{p-k,q-j}^{(h)}$.

We are looking for features invariant to blur. Feature B is called blur invariant if $B^f = B^{f*h} = B^g$.

As mentioned, we consider only symmetric $h(x, y)$. By applying the algorithm as derived and described in [1, 3], we can construct blur invariants based on central moments of any order by using the following recursive relation:

$$\begin{aligned} B(p,q) =& \mu_{pq} - \alpha \cdot \mu_{pq} - \frac{1}{\mu_{00}} \sum_{n=0}^{K} \sum_{i=m_1}^{m_2} \binom{p}{t-2i} \binom{q}{2i} \\ & \cdot B(p-t+2i, q-2i) \mu_{t-2i,2i} \end{aligned}$$

where

$$\begin{aligned} &K = [(p+q-4)/2], \quad t = 2(K-n+1), m_1 = max(0, [(t-p+1)/2]), \\ &m_2 = min(t/2, [q/2]), \quad \alpha = 1 \Leftrightarrow p \wedge q \text{ are even}, \quad \alpha = 0 \Leftrightarrow p \vee q \text{are odd}. \end{aligned}$$

The proposed algorithm uses 24 blur invariants up to the seventh order to create the feature vector $B = \{B_1, B_2, B_3, \ldots, B_{23}, B_{24}\}$ of each block. Some examples of utilized invariants in their explicit forms are listed below:

$$\begin{aligned} &B_1 = \mu_{11}, B_2 = \mu_{12}, B_3 = \mu_{21}, B_4 = \mu_{03}, B_5 = \mu_{30}, \\ &B_6 = \mu_{13} - \frac{3\mu_{02}\mu_{11}}{\mu_{00}}, B_7 = \mu_{31} - \frac{3\mu_{20}\mu_{11}}{\mu_{00}}, \\ &B_8 = \mu_{32} - \frac{3\mu_{12}\mu_{20} + \mu_{30}\mu_{02}}{\mu_{00}}, B_9 = \mu_{23} - \frac{3\mu_{21}\mu_{02} + \mu_{03}\mu_{20}}{\mu_{00}}. \end{aligned}$$

Because we will use an Euclidean metric space, the invariants should be normalized to have the same weight. To achieve this, the normalization described in [2, 3] is used $B_i^{'} = \frac{B_i}{(R/2)^r u_{00}}$, where R is the block size and r the order of B_i. Please note that in this manner normalized invariants are also invariant to contrast changes, which improves the duplication detection abilities of the algorithm.

As is obvious, each block is represented by a feature vector of length 24 in the case of gray-scale images. For RGB images, moment invariants of each block in each channel are computed separately, resulting in feature vector $B_{rgb} = \{B_{red}, B_{green}, B_{blue}\}$ of length 72.

2.3 Principal Component Transformation

In the case of an RGB image, the dimension of the feature vector is 72 (24 invariants per channel). Using the principal component transformation we reduce this dimension. Fraction of the ignored variance along the principal axes is set to 0.01. In PCT, the orthogonal basis set is given by the eigenvectors set of the covariance matrix of the original vectors. Thus, it can be easily computed on very large data sets. Note that PCT preserves the Euclidean distance among blocks.

2.4 k—d Tree Representation

In blocks similarity analysis (see next section) we will need to efficiently identify all blocks which are in a desired similarity relation with each analyzed block.

A simple exhaustive search computes the distance from the block to all others. This approach is very inefficient and its computational cost is $O(N)$. To improve the efficiency of finding neighboring blocks, some hierarchical structures have been proposed. The k—d tree is a commonly used structure for searching for nearest neighbors.

The k–d tree preprocesses data into a data structure allowing us to make efficient range queries. It is a binary tree that stores points of a k–dimensional space in the leaves. In each internal point, the tree divides the k–dimensional space into two parts with a $(k-1)$-dimensional hyperplane. If a k—d tree consists of N records, it requires $O(Nlog_2N)$ operations to be constructed and $O(log_2N)$ to be searched. Because of these reasons, the proposed method transforms blocks representation to a k–d tree for a more effective closest neighbors search.

2.5 Blocks Similarity Analysis

The main idea of this step is that a duplicated region consists of many neighboring duplicated blocks. If we find two similar blocks in the analyzed space and if their neighborhoods are also similar to each other, there is a high probability that they are duplicated and they must be labeled.

The similarity measure s employed here is defined by the formula:
$s(B_i, B_j) = 1/(1 + \rho(B_i, B_j))$, where ρ is a distance measure in the Euclidean space $\rho(B_i, B_j) = (\sum_{k=1}^{dim}(B_i[k] - B_j[k])^2)^{1/2}$. For each analyzed block represented by the feature vector B, we look for all blocks with an equal or larger similarity relation. It must be an equal or larger similarity to the threshold T. The method finds all similar blocks for each one (similar to the nearest neighbors search) and analysis their neighborhood. This is done efficiently using the k—d tree structure, which was created in the previous step.

If $s(B_i, B_j) > T$, where T is the minimum required similarity, we also analyze the neighborhood of B_i and B_j. Note that the threshold T plays a very important role. It expresses the degree of reliability with which blocks i and j correspond with each other. It is obvious that the choice of T directly affects the precision of results of the method. Due to the possibility of the presence of additive noise, a boundary effect, or JPEG compression, this threshold should not be set to 1.

After two blocks with the required similarity have been found, a verification step begins. In the verification step, similar blocks with different neighbors will be eliminated.

For analyzing the blocks neighborhood, we choose 16 neighboring blocks with a maximum distance of 4 pixels from the analyzed block (distance from their upper left corners). If 95% of these neighboring blocks satisfy the similarity condition, the analyzed block is labeled as duplicated. More formally, block 1 with coordinates (i, j) and block 2 with coordinates (k, l) are labeled as duplicated if $s(block(i + x_r, j + y_r), block(k + x_r, l + y_r)) \geq T$, where $x_r \in \langle -4, -3, \ldots 3, 4\rangle$ and $y_r \in \langle -4, -3, \ldots 3, 4\rangle$ and $r = 1 \ldots 16$. This part of the algorithm also determines the minimum size of the copied area, which can be detected by the algorithm.

To have more precise results, the verification step additionally uses information about the image distances of analyzed blocks.

If $s(block(i,j), block(k,l) \geq T)$, but $\sqrt{(i-k)^2 + (j-l)^2} \leq D$, these blocks will not be further analyzed and will not be assigned as duplicated. Threshold D is a user-defined parameter determining the minimum image distance between duplicated regions.

The output of this section is matrix Q with the same size as the input image. Elements of this matrix are either zero or one. An element of this matrix is set to one if the block at this position is duplicated.

2.6 Near-Duplicated Regions Map Creation

The output of the method is a near-duplicated regions map showing the image regions,which are likely duplicated. It is created by the multiplication of each element of $I(x,y)$ by its respective element in $Q(x,y)$. Matrix $Q(x,y)$ is created in the previous section.

3 Experimental Results

An experimental version of the proposed method was implemented in Matlab. The output of the method is a duplication map, in which likely duplicated regions are shown. Parameters of the method were set to $R = 24$ (block size), $T = 0.98$ (similarity threshold), $D = 24$ (blocks image distance threshold). These parameters can be changed (for instance, $R = 20, T = 0.97$), but it has a strong influence on obtained results. In the PCT step, the fraction of the ignored variance along the principal axes, ϵ, was set to 0.01. Please note that the computational time of the method is highly dependent on these parameters (specially on T and ϵ). In the presented example, the tampering was realized by copying and pasting a region in the image with intent to conceal a person or object. Additionally, in order to make the detection of forgery more difficult and interesting, the second example contains further manipulations of the pasted region.

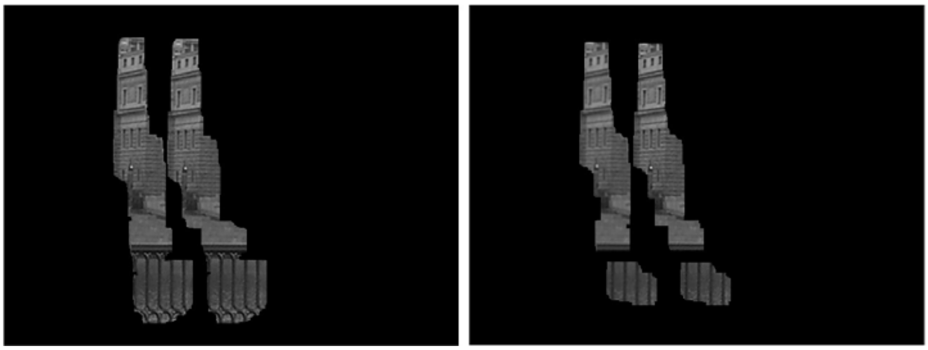

Fig. 2. Two duplication maps constructed by the proposed method

The first example is presented in the top image of Figure 2. It shows the output of the method applied to the tampered image shown in Figure 1(b). In this example, no further manipulations were carried out with the tampered regions. The tampered image was saved in TIFF format. The output shows that the proposed method correctly detected the near-duplicated regions.

The bottom image of Figure 2 shows the duplication map created by applying our method to Figure 1(b). In this example, the tampered region was blurred with a Gaussian blur filter with radius 0.3 pixels. The tampered image in this case was saved in JPEG format quality 80.

4 Discussion

Our results show that the use of blur moment invariants can improve the detection abilities of the copy—move forgery detection methods. By using blur moment invariants we are able to additionally detect duplicated regions with presence of acceptable blur and additive Gaussian noise. By normalizing moment invariants they also become invariant against contrast changes. The proposed method also works with lossy JPEG format images.

It can be interesting to mention briefly the stability of moment invariants under additive random noise. As aforementioned, moment invariants are computed by a summation over the whole image, so they are not significantly affected by additive zero-mean noise. For a more detailed discussion about this topic see [3].

Our method is based on truncated versions of the filtered blocks which are additionally corrupted by the neighboring pixels. Therefore we have to mention the stability of moment invariants with respect to boundary effect. If our blocks have $R \times R$ pixels and the size of PSF support is $H \times H$ pixels, the correct size of the resulting block must be $(R + H - 1) \times (R - H - 1)$. In our case, the value of H is unknown. If $H \ll R$ the errors of invariant calculation caused by the boundary effect is negligible. If H is relatively large as in the case of heavy blur, the boundary effect will cause significant miscalculations of the invariant values. For an experiment on this topic, see [3].

A disadvantage of the proposed method is its computational time. The average run time of the implemented experimental version with parameters $R = 24$ (block size) and $T = 0.98$ (similarity threshold) for 640×480 RGB images on a 2.1 GHz processor and 512 MB RAM is 30 min. The computational time is not the same for images with the same size. It is dependent on each images characteristics (the dimension of space created after the PCT) and especially on the similarity threshold parameter of the algorithm. It is also important to note that the implemented experimental version was not optimized and there exist possibilities to improve the computational time.

A way to considerably improve the computational time is to eliminate large uniform areas in a preprocessing step and apply the proposed method to the rest of the analyzed image. Then the output of the method could consist of a duplication map and a uniform areas map.

References

1. J. Flusser and T. Suk. Degraded image analysis: An invariant approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(6):590–603, 1998
2. J. Flusser, T. Suk, and S. Saic. Image features invariant with respect to blur. Pattern Recognition, 28(11):1723–1732, 1995
3. J. Flusser, T. Suk, and S. Saic. Recognition of blurred images by the method of moments. IEEE Transactions on Image Processing, 5(3):533–538, 1996
4. J. Fridrich, D. Soukal, and J. Lukas. Detection of copy-move forgery in digital images. In Proceedings of Digital Forensic Research Workshop, Cleveland, OH, USA, August 2003, IEEE Computer Society
5. A. Popescu and H. Farid. Exposing digital forgeries by detecting duplicated image regions. Technical Report TR2004-515, Department of Computer Science, Dartmouth College, 2004