Forensic Science International 283 (2018) 47–57

Contents lists available at ScienceDirect

Forensic Science International

journal homepage: www.elsevier.com/locate/forsciint

Detection of copy-move image modification using JPEG compression model

Adam Novozámský*, Michal Šorel

The Czech Academy of Sciences, Institute of Information Theory and Automation, Pod Vodárenskou věží 4, 182 08 Prague 8, Czechia

ARTICLE INFO

Article history:
Received 14 July 2017
Received in revised form 15 November 2017
Accepted 20 November 2017
Available online 9 December 2017

Keywords:
Copy-move modification
Forgery
Image tampering
Quantization constraint set

ABSTRACT

The so-called copy-move forgery, based on copying an object and pasting in another location of the same image, is a common way to manipulate image content. In this paper, we address the problem of copy-move forgery detection in JPEG images. The main problem with JPEG compression is that the same pixels, after moving to a different position and storing in the JPEG format, have different values. The majority of existing algorithms is based on matching pairs of similar patches, which generates many false matches. In many cases they cannot be eliminated by postprocessing, causing the failure of detection. To overcome this problem, we derive a JPEG-based constraint that any pair of patches must satisfy to be considered a valid candidate and propose an efficient algorithm to verify the constraint. The constraint can be integrated into most existing methods. Experiments show significant improvement of detection, especially for difficult cases, such as small objects, objects covered by textureless areas and repeated patterns.

1. Introduction

The integrity of visual data is important for the credibility of news media and especially when used as an evidence in court or during criminal investigation. In the times of analog recording and classical photography any tampering was considered difficult and time consuming but the availability of easy-to-use image processing software made the manipulation of digital content extremely simple. Image forensics, a branch of forensic data analysis, evolved as a scientific means to verify the source of image data and detect potential modifications.

One of the most common image modifications, the so-called *copy-move forgery* [1], is based on copying an object and pasting in another location of the same image. Transition between the inserted object and original contents is often masked by various retouching tools. Copying from the same image keeps statistical properties of the image such as the noise, contrast and color, which complicates detection. On the other hand, reusing the same object in one image can be detected and is what is looked for by the majority of copy-move forgery detection (CMFD) techniques.

While the copy-move forgery is relatively easy to detect in images stored in lossless formats, such as PNG, GIF, and TIFF, detection in JPEG images is complicated by the fact that the same pixels, after moving to a different position and storing in the JPEG format, have different values. For this reason, when looking for copy-move candidates, existing methods usually consider all patches that are in a sense similar. Irrespective of how the similarity is measured, the problem arises that the criterion used is always a compromise between detecting all true candidates and getting a reasonable number of false positives, i.e. patches that actually were not copied but must be considered valid candidates.

In this article, we analyze the problem whether it is possible to reduce the number of false positives using the exact mechanism of JPEG compression. In other words, whether it is possible to say something about how originally identical patches can differ under compression.

The solution we present in this article is based on the observation that all original images that could have resulted in the coefficients stored in the JPEG file we analyze, form a convex set, in the field of digital image restoration known as the quantization constraint set [2]. This set can be specified separately for each JPEG block, typically of size 8×8 pixels. For grayscale images, the set has a form of a multi-dimensional box (in general orientation) centered in the values we get by ordinary JPEG decompression. The dimensions of this box are given by the values from the quantization table stored in every JPEG file during

* Corresponding author.
E-mail addresses: novozamsky@utia.cas.cz (A. Novozámský), sorel@utia.cas.cz (M. Šorel).

https://doi.org/10.1016/j.forsciint.2017.11.031
0379-0738/

compression. Therefore, since the patch we are analyzing is present at two different places, the original question can be reformulated as verifying whether the intersection of two sets is empty or not. Is there an original patch that would be compressed to two different sets of coefficients stored in two different positions in the JPEG file? In this paper, we call this condition the *JPEG copy-move constraint* and show an efficient method how its validity can be verified. We show that the idea of using the JPEG copy-move constraint is compatible with most existing methods. It helps especially in images with repeated motives and objects retouched by an indistinct texture, which are common situations extremely difficult for all methods we can find in literature.

2. Related work

The detection of copy-move modification was first proposed by Fridrich et al. [1], which inspired a large amount of research in the same direction. For this type of image modification, they started using the term *copy-move forgery*, which became a standard in subsequent literature. The number of papers has been increasing every year and CMFD has become one of the most active topics in image forensics. In this section, we give a short overview of main approaches, for a more complete treatment, we refer readers to [3,4] comparing many popular algorithms. Readers interested in exposing digital forgeries and image forensics in general can start with [5] and tutorials available on the web pages of Hany Farid.

The typical processing pipeline used by most CMFD methods is shown in Fig. 1 (derived from Fig. 2 in [3]). The preprocessing phase includes decompression of image files if stored in a compressed format and for methods working on grayscale images also conversion to grayscale.

Based on the second step (the second block from left in Fig. 1), CMFD methods can be divided into two large groups, block-based and keypoint-based, depending on the mechanism used to find the candidates for matching. The vast majority of them is in the first group, including [1]. In this approach, algorithms look for a similar block obtained by partitioning the image into overlapping blocks. The reason, why it is usually impractical to look for exactly the same values (exact match in [1]) is that values can be damaged by JPEG compression or additional retouching operations. Methods either work directly with pixels or compute features transforming blocks into a representation of lower-dimension or having convenient invariant properties. For example, [1] used weighted DCT transform coefficients and Bashar et al. [6] the Daubechies four tap wavelet transform.

An important reason for using features is to achieve invariance with respect to some transforms. While the basic copy-move detection assumes that object is unchanged and in the same orientation, in practice it is possible that before being pasted, it was rotated, scaled or even blurred. To ensure the invariance to rotation, Wang et al. [7] used a circle block model, Bayram et al. [8] added a scale invariance by features obtained from the Fourier-Mellin transform. Several authors worked with moment invariants. Mahdian and Saic [9] suggested 24 blur invariants, Ryu et al. [10] used the Zernike moments.

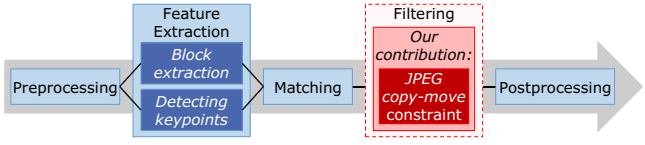


Fig. 1. Typical processing pipeline of copy-move forgery detection algorithms. The main contribution of this paper is filtering using the JPEG copy-move constraint, which increases sensitivity and reduces significantly the number of false matches.

In the matching phase (third phase in Fig. 1), algorithms take the blocks or their features stored in a feature matrix and look for similar entries. Since considering all pairs of blocks would be extremely time-consuming, methods save time in various ways. One possibility is decreasing the number of features, which reduces the dimension of space we are searching. Popescu et al. [11] reduced the feature matching space by the PCA applied on blocks. Bashar et al. [6] modified their algorithm to use the kernel PCA. Huang et al. [12] simply shortened the feature vector to its first quarter. A complementary way to reduce computational complexity is using an efficient data structure for the approximate nearest-neighbor search. In the original paper [1], similar blocks were found by lexicographically sorting the rows in the feature matrix and comparing the adjacent rows, which was used by many followers. A more reliable efficient data structure are k-d trees, mostly used in the best-bin-first variant [13], which is applicable in both block and keypoint-based algorithms. Comparison [3] uses multiple randomized k-d trees [14]. Another fast iterative randomized technique for computing the approximate nearest-neighbor search can be found in [15].

The purpose of the filtering step (4th in Fig. 1) is to remove false matches that inevitably arise in all methods. A common procedure is to remove spatially close matches that appear because of correlation between spatially close regions. Another heuristic is skipping low-variance areas, such as skies, building facades, water surfaces, etc. The main contribution of this paper belongs to this phase, filtering pairs of regions based on the compatibility with the JPEG compression process (JPEG copy-move constraint).

Finally, a postprocessing step verifies the spatial coherence of shift vectors (or in general transform parameters) obtained from the candidates generated by matching and filtering, as a rule by a combination of ideas known in image processing as the Hough transform [16] and RANSAC [17]. The seminal paper [1] simply considered only pairs with a shift vector identical to a sufficient number of other pairs. Postprocessing can also eliminate objects smaller than a threshold.

The block-based techniques are time-consuming due to the large number of compared blocks and lose accuracy when the tampered areas are blurred, scaled, rotated or otherwise geometrically transformed. To address these problems, some authors, instead of working with blocks, applied keypoint-based techniques used in computer vision, where the task is to find point correspondence between two images of one object or the same scene. Huang et al. [18] came up with a CMFD method based on the SIFT features [19], Bo et al. [20] used SURF [21]. In general, the keypoint-based techniques are fast and can be easily used when the patches were deformed by a geometric transform. On the down-side, they do not work well for small objects and are less stable than block-based methods. Especially, since the keypoints are usually detected in regions with high entropy, these methods lose accuracy in the areas retouched by an indistinct texture [22].

Hybrid schemes combine advantages of block-based and keypoint-based approaches [23–27]. Postprocessing can also include segmentation to estimate the contours of the copied object [28].

This paper deals with the basic but very frequent scenario of simple copy-move in JPEG images, i.e. without geometric distortions, added noise or blurring. We analyze the influence of JPEG compression and propose an algorithm that uses the principles of JPEG compression to increase the sensitivity of detection.

In the next section, we describe the principles of JPEG compression (Section 3). Section 4 describes the main idea of this paper, the JPEG copy-move constraint. Section 5 shows an efficient algorithm to verify the constraint. In the experimental section (Section 6), we show how the constraint can be used in a

practical detection scheme. The results are compared to several alternative algorithms.

3. JPEG compression

JPEG is undoubtedly the most widespread format for efficient storage of image data [29]. JPEG uses a lossy type of compression based on the quantization of discrete cosine transform (DCT) coefficients, where the two-dimensional DCT is applied to small blocks of usually 8×8 pixels. In this section we describe this process in detail and introduce the mathematical notation needed later.

For grayscale images, the lossy part of JPEG compression can be expressed [2,30] as

$$y = [QCx], \tag{1}$$

where x is a vectorized original image, y the vector of integer coefficients stored in the JPEG file, Q and C are matrices, and the square brackets denote the operator of rounding to the nearest integer. C is the block-diagonal matrix of the block DCT made up of the square matrices of the two-dimensional DCT (64×64 matrix for each 8×8 block). C is orthogonal, because all the DCT sub-matrices are orthogonal. Q is a diagonal matrix corresponding to the element-wise division by the coefficients from the quantization table stored in each JPEG file replicated for each block along diagonal (64 values for each 8×8 block). We will denote the vector of these coefficients as q, i.e. $Q = \mathrm{diag}(1/q)$. For color images, the image is first transformed into $Y'C_BC_R$ space and individual channels are stored separately. The brightness Y' is treated as described above but the chrominance channels are often stored at smaller resolution, which complicates the degradation model. Formally,

$$y = [QCDx], \tag{2}$$

where D is a down-sampling matrix. As a rule, D returns the average value for every (non-overlapping) square of 2×2 pixels but other dimensions of down-sampling windows, such as 2×1 or 1×2 are possible. The grayscale case (1) can be considered a special case of (2) with $D = I$, i.e. identity. JPEG decompression can be written as

$$\tilde{x} = \frac{1}{k} D^T C^T Q^{-1} y = \frac{1}{k} D^T C^T \,\mathrm{diag}(q) y, \tag{3}$$

where $k = 1/4$ for the 2×2 down-sampling window, $k = 1/2$ for 2×1 and 1×2, and $k = 1$ for grayscale (no down-sampling).

Given coefficients y stored in a JPEG file, the quantization constraint set is the set of images satisfying condition (2) in all color channels. This set is local in the sense that it is defined independently for each JPEG block. This locality is reflected in the structure of matrix QCD, which is block-diagonal with blocks of size 64×64 for grayscale and 64×256 for the chrominance channels with 2×2 down-sampling.

4. JPEG copy-move constraint

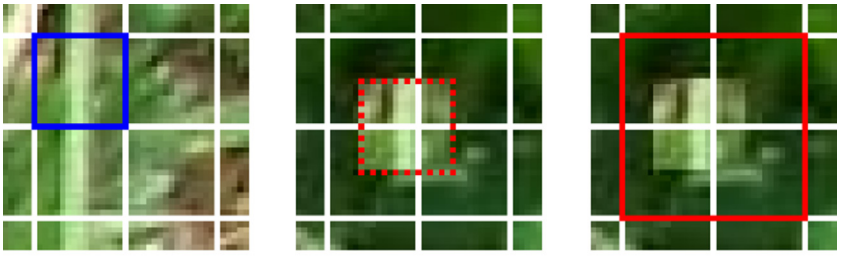
The main contribution of this paper is a procedure to verify the possibility of copy-move between two patches with known coordinates using the knowledge of JPEG compression process including the quantization table extracted from the input JPEG file. We assume that the object was only moved, i.e. there was no rotation or any other geometrical distortion, and that there was no noise added. The object could have been copied to an arbitrary part of the image and after moving to the target area, the object could have been retouched by blurring boundaries to mask the transition to the surrounding area. The main requirement is that the original object contains at least one JPEG block, i.e. a rectangular area of size at least 8×8 pixels aligned with the JPEG grid (see Fig. 2(a)). This is guaranteed for objects of size 15×15 pixels and larger. The corresponding target area (Fig. 2(b)) must not be retouched. If we use also the chromatic channels, the necessary patch size increases to 16×16 pixels.

For our purpose, we assume that the detection algorithm works with patches of size being a multiple of the size of JPEG blocks (typically 8×8 for grayscale and 16×16 if we consider also the chrominance channels). In addition, we assume that the source patch is aligned with the JPEG grid (in Fig. 2(a) the patch consists of only one JPEG block). In the target area, we consider a larger patch, the smallest patch containing the cloned area aligned with JPEG blocks, see Fig. 2(c). Denoting the vectorized pixel values before compression in the enlarged target area as x, the values of the original source patch (Fig. 2(a)) can be expressed as Mx, where M is the matrix selecting the pixels corresponding to the source area in Fig. 2(b) (selection matrix). Eq. (2) implies that the set of possible target areas before JPEG compression satisfy

$$QCDx \in \left\langle y_2 - \frac{1}{2}, y_2 + \frac{1}{2} \right), \tag{4}$$

where y_2 is the vector of integer coefficients stored in the JPEG file that corresponds to the target patch. The interval $\langle y_2 - \frac{1}{2}, y_2 + \frac{1}{2})$ is the multi-dimensional interval of numbers rounding to the nearest integers in y_2. The interval is left-closed (angle bracket) and right-open (round bracket) as usual when rounding. In the original area containing the same (source) patch Mx similarly

$$QCDMx \in \left\langle y_1 - \frac{1}{2}, y_1 + \frac{1}{2} \right). \tag{5}$$



(a) original patch stored as y_1 (b) copy in the target area (c) Enlarged target area stored as y_2

Fig. 2. Alignment of original and target patches as used in the verification of the JPEG copy-move constraint.

To summarize the dimensions of the above defined variables for grayscale blocks of 8×8 pixels in general position, x, y_1 and y_2 are vectors of sizes $256 = 16 \times 16$, $64 = 8 \times 8$ and $256 = 16 \times 16$, respectively. In the case of the target area aligned with the grid in one of dimensions, the size of x and y_2 is reduced to $128 = 16 \times 8$.

Sets (4) and (5) are convex. We can see that the necessary condition for the possibility of copy-move between source and target areas is equivalent to the existence of a point in the intersection of sets (4) and (5). We call this condition the *JPEG copy-move constraint* or shortly *JPEG constraint*. Unfortunately, it is probably impossible to verify the existence of this intersection directly by a closed-form formula. Nevertheless, in the following section, we derive an efficient algorithm that verifies the JPEG constraint iteratively.

5. Algorithm

We start this section by describing a general algorithm to find a point in the intersection of arbitrary two convex sets A and B. This algorithm will be finally applied to our sets (4) and (5). The simplest way to find a point in the intersection of sets A and B in a vector space is alternating projection (see Fig. 3). If the intersection is not empty, this procedure provably converges to a point in the intersection. Otherwise, the distance between two consecutive projections converges to the distance between A and B. This method is known as the projection on convex sets (POCS) [31]. Unfortunately, POCS converges relatively slowly (in our case hundreds of iterations), which makes it unsuitable for our purpose.

The convergence can be significantly speeded up by the Douglas–Rachford splitting [32], which is a special case of the alternating direction method of multipliers (ADMM) [33]. If exists, the intersection $A \cap B$ can be found by iterating the following three steps

$$x \leftarrow P_A(a + d), \tag{6}$$

$$a \leftarrow P_B(x - d), \tag{7}$$

$$d \leftarrow d - (x - a), \tag{8}$$

where P_A and P_B are projections on sets A and B, and a and d are auxiliary variables of the same size as x. By the projection on a set we mean the point in the set closest to the projected point in the sense of l_2 norm. Variable x is in general a point in set A, which will be in our algorithm the same x defined in the previous section. Variable a is initialized in a starting point x_0, d is initially zero. Iterations are stopped, when the l_2 norm of $x - a$ is smaller than a threshold ε or the number of iterations exceeds a limit. The latter implies that $A \cap B$ is empty. Note that both POCS and ADMM require sets A and B to be closed. For this reason, instead of intervals in (4) and (5), we work with closed intervals $\langle y - \frac{1}{2}, y + \frac{1}{2} - \delta \rangle$, where δ is a sufficiently small constant.

To make the algorithm efficient, we need also the projections in (6) and (7) to be fast. Since the set (4) is of the same form as (5), its is sufficient to show the efficient projection for the latter. Indeed, projection (5) can be expressed as

$$P_{QCDMx \in \langle y - \frac{1}{2}, y + \frac{1}{2} - \delta \rangle}(z) = z - \frac{1}{k} M^T D^T C^T \operatorname{diag}(q) \cdot \left(QCDMz - P_{\langle y - \frac{1}{2}, y + \frac{1}{2} - \delta \rangle}(QCDMz) \right), \tag{9}$$

which is straightforward to compute in time proportional to the number of pixels in the patch. Transpose C^T is the inverse DCT, D^T replicates each value in a down-sampled image to the corresponding rectangle in the full size image and M^T keeps the pixels selected by M (corresponding to the source patch) intact and fills the rest of the pixels of the target area with zeros. Constant $k = 1/mn$ is a down-sampling factor ($1/4$ for 2×2 down-sampling). The projection on set (4) is a special case with $M = I$, i.e. identity. Note that y in (9) stands for y_1 and y_2 for sets (4) and (5), respectively. Informal proof (using a lemma from [34]) is given in the appendix.

The algorithm is summarized as Algorithm 1. We can see that (9) and therefore also the algorithm basically consists of JPEG compression and decompression operations (compare with (2) and (3)). The compression operations *QCD* and *QCDM* are computed only once in each projection, since its result can be reused. In our experiments the threshold in the stopping criterion was set to $\varepsilon = 10^{-10}$ and the maximum number of iterations to 12. What is important, the number of iterations does not depend on the number of image pixels. In theory, it can slightly increase for larger patches, though. The initial estimate x_0 can be set to the values in the target area.

The time needed to compute one projection is approximately the same as the time of one JPEG compression (*QCD*) and decompression ($\frac{1}{k} D^T C^T \operatorname{diag} q$) of an image of the same size as the patch. As a result, since there are two projections in one iteration, one on a source patch and one on a target patch, and we use twelve iterations, the time of Algorithm 1 corresponds to 12

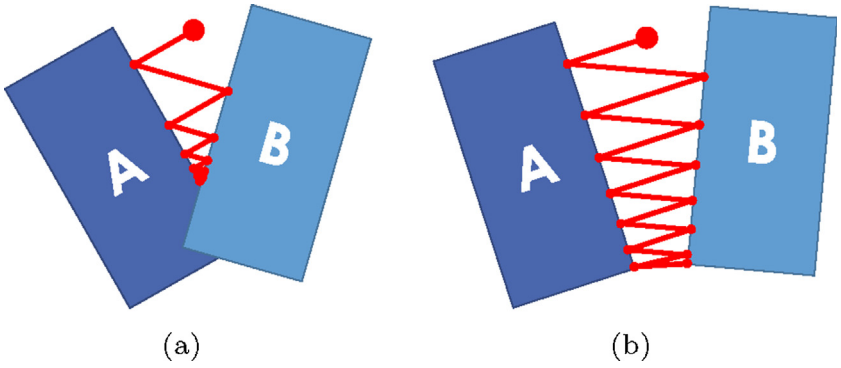


Fig. 3. Finding the intersection of two sets by alternating projection, where $A \cap B \neq \emptyset$ (a) or $A \cap B = \emptyset$ (b). We use a faster variant of this algorithm to verify the proposed JPEG copy-move constraint.

grayscale JPEG compressions and decompressions of an image of the same size as the aligned target patch, i.e. 16×16 pixels for the source patches of size 8×8, and 12 of an image of the size of the source patch.

Algorithm 1. Verification of the JPEG copy-move constraint, i.e. the possibility of copy-move between two image patches.

1. Initialize $a_0 = x_0$, $d_0 = 0$
2. repeat
3. $$x \leftarrow a + d - \frac{1}{k} D^T C^T \operatorname{diag}(q) \cdot \left(QCD(a+d) - P_{\langle y_2 - \frac{1}{2}, y_2 + \frac{1}{2} - \delta \rangle} (QCD(a+d)) \right)$$
4. $$a \leftarrow x - d - \frac{1}{k} M^T D^T C^T \operatorname{diag}(q) \cdot \left(QCDM(x-d) - P_{\langle y_1 - \frac{1}{2}, y_1 + \frac{1}{2} - \delta \rangle} (QCDM(x-d)) \right)$$
5. $d \leftarrow d - (x - a)$
6. until the stopping criterion $||x - a|| \leq \varepsilon$ is satisfied or the number of iterations exceeds a limit

6. Experiments

In this section, we present results of a series of experiments demonstrating the benefits of the proposed JPEG copy-move constraint. Experiments in Sections 6.1–6.3 statistically analyze the influence of the constraint on the number of falsely detected moves, irrespectivelly of other steps in the chain of operations in Fig. 1, i.e. independently of what particular method is used. Section 6.4 shows the performance of the constraint in a complete detection algorithm. For this purpose, we use the JPEG copy-move constraint with a simple patch-level matching, finally verifying the coherence of shift vectors. This approach is compared to two representatives of block-based techniques and four methods based on keypoints, using three different datasets. Two datasets [35,4] are standard benchmark datasets available online, the third was created by us to show performance on difficult cases with repeated patterns and indistinct texture.

6.1. Estimating the parameters of the algorithm

Purpose of the first experiment is to determine a detection threshold needed to select candidates for matching and two parameters of Algorithm 1, convergence criterion ε and the number of iterations sufficient to reliably verify the JPEG constraint. We use ten images of size 1024×1024 pixels (Fig. 4), covering several types of textures. We select randomly a position of source patch aligned with the JPEG grid, copy the data, paste the patch to a random off-grid position and save this new picture to JPEG. The patches pasted on the positions aligned with the grid are not considered, because they keep exactly the same values. After reloading the image, we take both patches, compute their l_2 distance per pixel and run our algorithm until convergence ($\varepsilon = 10^{-10}$). Since these patches come from the same original patch, the algorithm always converges. For each iteration, we remember the norm of the residual $||x - a||$. We repeat this experiment 1000 times for each image, with the quality of JPEG set to values in {10, 20, . . . , 90, 95, 98}.

This experiment gives us two valuable outputs. The maximum value of the l_2 distance gives us an estimate of the threshold necessary to select the pairs of patches that must be considered valid candidates for matching. However, for computational reasons, we select the threshold to detect only 99.5% of candidates. Otherwise, we would have to test an excessive number of candidate patches. The resulting threshold is shown in Fig. 5. The threshold decreases with increasing JPEG quality and stays surprisingly high (above 15) even for the best qualities.

The second output is the number of iterations the algorithm requires to tell whether the JPEG constraint is satisfied or not. Fig. 6 shows that the brightness component Y' never requires more than 12 iterations for any JPEG quality. This is in contrast to the chrominance components C_B and C_R that require up to 50 iterations for higher qualities. In practice, since the brightness channel carries more information than the chrominance channels, it is mostly sufficient to work only with brightness, which speeds up computation and does not considerably increase the number of false matches. In addition, we can use smaller patches of 8×8 pixels, instead of 16×16, which is useful for detecting small objects. For this reason, in the rest of this section, all experiments work only with the brightness channel, i.e. in grayscale.

Our tests on the other image sets showed that the setting $\varepsilon = 10^{-10}$ and 12 iterations is quite universal, which means that using the JPEG copy-move constraint does not require setting any additional parameters.

6.2. Reducing the number of false matches

The purpose of the second experiment is to demonstrate that the JPEG copy-move constraint decreases the number of false matches. The procedure is similar to what we did previously except

Fig. 4. Images used to estimate the parameters of the algorithm and analyze the number of false matches.

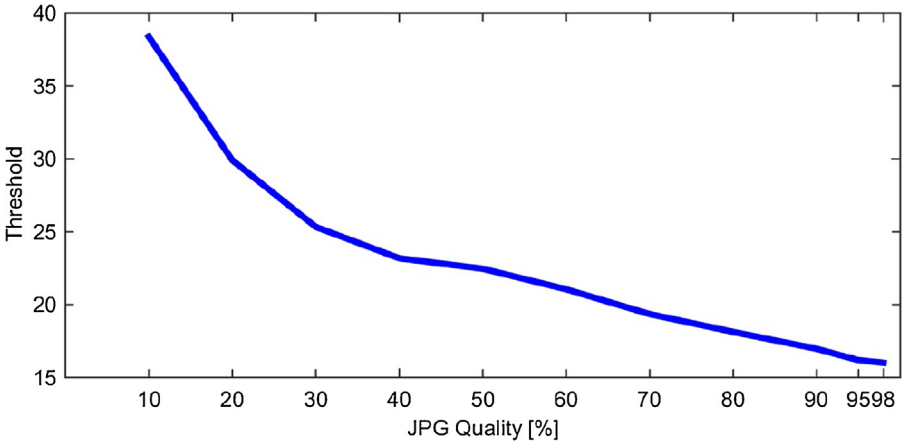


Fig. 5. Maximum l_2 distance per pixel of copy-moved patches at the sensitivity level 99.5%.

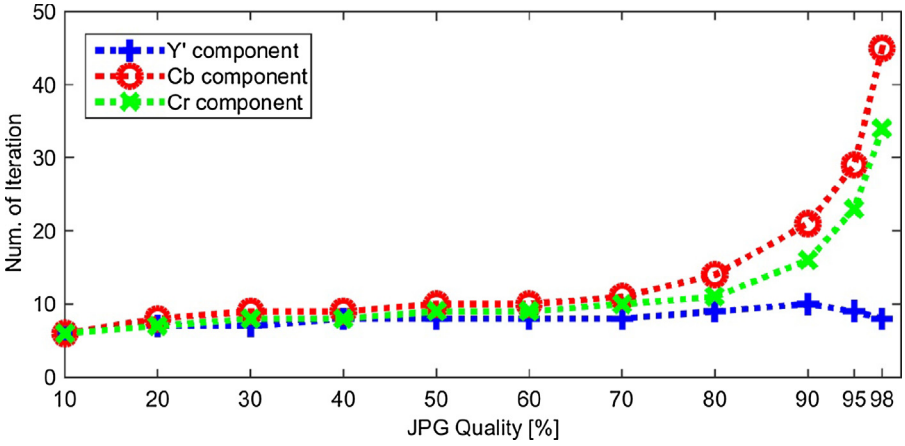


Fig. 6. Number of iterations needed for convergence ($\varepsilon = 10^{-10}$).

that we do not clone the patches and assume that randomly taken pairs of patches should always be negatives. We take images from Fig. 4 and save them using the same set of compression qualities {10, 20, . . . , 90, 95, 98}. After reloading each image, we go patch by patch from the upper-left corner to the lower-right corner. For each patch, we find the most similar patch from the same image (in the l_2 distance) and run our procedure for verifying the JPEG constraint.

If we use the thresholds computed in the first experiment using the l_2 distance, the number of false positives is huge, over 60% even for qualities above 90 (solid line in Fig. 7). We cannot set the threshold lower, otherwise we could miss some small objects because of decreased sensitivity. Note that using the DCT features as in [1] would keep the percentage of false matches high and therefore, for our comparison, it would not help. The dotted line shows the percentage of false matches when we apply also the

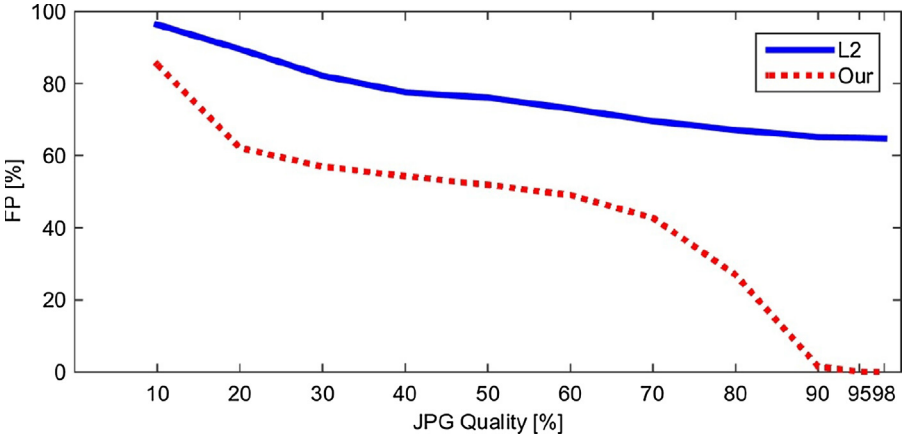


Fig. 7. Percentage of false matches (false positives) at sensitivity level 99.5% (thresholds taken from Fig. 5, number of iterations 12).

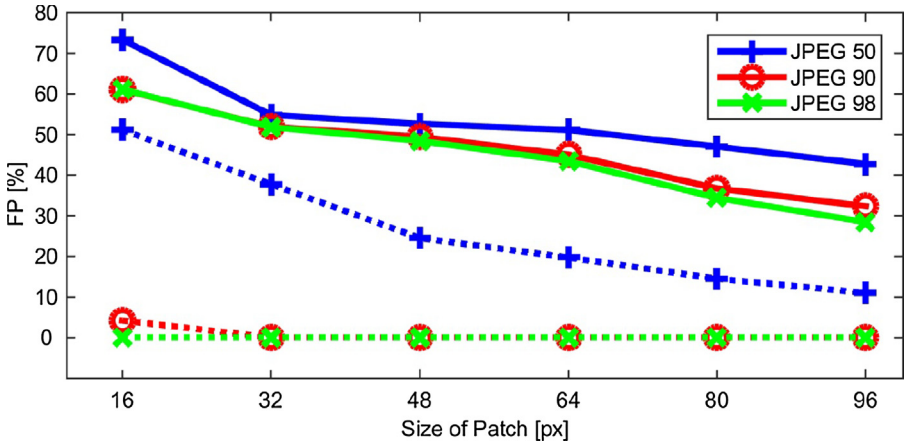


Fig. 8. Percentage of false matches for three quality levels (50, 90 and 98) using the l_2 distance (solid line) and the JPEG copy-move constraint (dotted) at sensitivity level 99.5%.

JPEG copy-move constraint. We can see that the number of false matches decreases. The effect is very strong at qualities higher than 90, where the percentage decreases to less than 1.5%. At quality 95, we get less than 0.04% and at 98 there are no false matches. Note that the default JPEG qualities in the majority of modern photo editing tools and cameras are above 90 and therefore these high qualities are very common.

6.3. Influence of patch size

The third experiment (Fig. 8) shows the influence of the patch size. We considered patches of sizes 16×16, 32×32, 48×48, 64×64, 80×80 and 96×96 pixels. As the patch size increases, the number of false matches slightly drops. For example, for quality 50 and the JPEG constraint, the percentage of false matches drops from 50% to 10%. Without the constraint, this number remains as high as 30%.

6.4. Comparison on benchmark datasets

Previous experiments showed that the JPEG copy-move constraint decreases the number of false matches significantly. In this section, we show how this procedure improves performance of a complete detection algorithm. We use simple patch-level matching, followed by the JPEG copy-move constraint and verification of the coherence of shift vectors. For each JPEG block, we find the nearest patch in the l_2 norm. Such pair of patches is considered a valid copy-move candidate if their l_2 distance is smaller than a threshold determined by the experiment described in Section 6.1. The coherence is achieved by considering only the object with the highest number of detected shift vectors. We will see that even this simple variant gives impressive results. This approach is compared to six alternatives.

As a representative of block-based methods, we chose the method based on the quantization of the DCT coefficients from the seminal paper [1]. When used with 8×8 patches needed to detect smaller objects, the basic version of the algorithm, as presented in the paper, gives relatively weak results. The main reason is that the algorithm uses lexicographic sorting and therefore misses many potential candidates. On the other hand, it is quite easy to tune up the method to give much better results by increasing the level of quantization and setting a suitable threshold to the maximum allowed distance of quantized DCT coefficients to be considered copy-move candidates. To distinguish these two versions, we denote them in our comparison as DCT-basic and DCT-tuned. To verify the coherence of shift vectors, both DCT-based variants we implemented use the same rule as the algorithm with the JPEG constraint. As a representative of methods based on keypoints, we use the SIFT-based algorithm [36], available online. For a wider comparison we also added the approach of Cozzolino et al. [15], namely their three types of features: the Zernike moments (ZM), the Polar Cosine Transform (PCT), and the Fourier–Mellin transform (FMT).

To simplify comparison to other methods in literature, we use two standard image datasets [35,4]. The performance of other methods on the same datasets is analyzed in [3,4]. We also created a third image set containing difficult cases with repeated patterns, objects masked by an indistinct texture (skies, building facades, etc.) and small objects. All three datasets (DS1, DS2 and DS3), contain in total 81 images:

- **DS1:** 23 images by Silva at al. [4],
- **DS2:** 40 images, CoMoFoD dataset [35],
- **DS3:** 18 images created by our research group, representing the difficult cases.

To evaluate the algorithms, we use two different metrics. In both cases, we compare with the ground truth represented, for each image, by a matrix of ones on the tampered pixels and zeroes elsewhere.

- *Image level:* An algorithm is successful if it finds at least a portion of the tampered area. The algorithm fails if it does not find any modified pixel.
- *Patch level:* We divide images to non-overlapping patches of 8 x 8 pixels. A patch is considered correctly marked as tampered if there is an overlap with a pixel in the ground truth. We count the number of correctly found tampered patches (true positivies,

Table 1
Comparison of methods on dataset DS1, 23 images, 13,645 patches in ground truth.

Method	Image level		Patch level TPs		Patch level FPs	
	#	%	#	sens. [%]	#	FDR [%]
DCT-basic	16	69.57	2697	19.77	1855	40.75
DCT-tuned	23	100.00	11,644	85.34	89	0.76
SIFT	21	91.30	9127	66.89	1641	15.24
ZM	23	100.00	12,988	95.19	776	5.64
PCT	23	100.00	12,956	94.95	777	5.66
FMT	23	100.00	12,788	93.72	630	4.70
OUR	23	100.00	12,432	91.11	56	0.45

Table 2
Comparison of methods on dataset DS2, 40 images, 9288 patches in ground truth.

Method	Image level		Patch level TPs		Patch level FPs	
	#	%	#	sens. [%]	#	FDR [%]
DCT-basic	16	40.00	532	5.73	1130	67.99
DCT-tuned	36	90.00	6601	71.07	595	8.27
SIFT	26	65.00	6057	65.21	9196	60.29
ZM	18	45.00	6560	70.63	855	11.53
PCT	20	50.00	6989	75.25	970	12.19
FMT	21	52.50	7035	75.74	1705	19.51
OUR	40	100.00	8055	86.72	127	1.55

Table 3
Comparison of methods on dataset DS3 (difficult cases), 18 images, 9850 patches in ground truth.

Method	Image level		Patch level TPs		Patch level FPs	
	#	%	#	sens. [%]	#	FDR [%]
DCT-basic	10	55.56	2353	23.89	862	26.81
DCT-tuned	16	88.89	8672	88.04	890	9.31
SIFT	6	33.33	4845	49.19	4793	49.73
ZM	10	55.56	6965	70.71	675	8.84
PCT	11	61.11	7060	71.68	1062	13.08
FMT	11	61.11	6983	70.89	2422	25.75
OUR	18	100.00	8987	91.24	119	1.31

Table 4
Comparison of methods on full dataset (DS1 + DS2 + DS3), 81 images, 32,783 patches in ground truth.

Method	Image level		Patch level TPs		Patch level FPs	
	#	%	#	sens. [%]	#	FDR [%]
DCT-basic	42	51.85	5582	17.03	3847	40.80
DCT-tuned	75	92.59	26,917	82.11	1574	5.52
SIFT	53	65.43	20,029	61.10	15,630	43.83
ZM	51	62.96	26,513	80.87	2306	8.00
PCT	54	66.67	27,005	82.38	2809	9.42
FMT	55	67.90	26,806	81.77	4757	15.07
OUR	81	100.00	31,301	89.91	302	1.01

TPs) and the number of patches labeled as tampered with no overlap with the ground true (false positives, FPs).

For space reasons, we show results on JPEG images compressed only with quality factor 95. As Fig. 7 suggests, we can expect similar behavior for qualities above around 90.

Quantitative results for all the datasets as well as for the union of all the datasets are given in Tables 1–4. The second column shows the number of images, where the algorithm found at least a part of the tampered area. The same number, as a percentage of all images in the dataset, is given in the third column (success rate). The forth and sixth columns show the numbers of patch level TPs and FPs. We also compute two standard statistical quantities, sensitivity (abbreviated as sens. in Tables 1–4) and false discovery rate (FDR, one minus precision). They are given in columns five and seven. The number of patches in ground truth is defined as the number of patches containing at least one modified pixel. Three examples in Figs. 9–11 illustrate typical outputs of the tested approaches.

At the image level, the SIFT-based method achieves a decent success rate of 91% on DB1 but often fails on DB2 and DB3 (65% and 33%). Moreover, this method produces a large number of false matches at the patch level, even for relatively easy examples. Fig. 10(d) illustrates how repeated motives can cause a complete failure of this algorithm, a problem mentioned already in the original paper [36]. We see similar results also for all three methods of Cozzolino [15]. Although the success rate on DS1 database is at the patch level over 95% for ZM, over 94% for PCT, and over 93% for FMT, the FDR is around 5%. And the success rates on the other two databases are between 70% and 75% only.

The results in Tables 1–4 and examples in Figs. 9–11(b) show that the basic version of the DCT-based algorithm [1] misses most true matches. Using the modified version of [1] described above (DCT-tuned), we achieved an overall success rate over 92% at the image level and sensitivity over 82% at the patch level, see Table 4. As the weakest point of this approach, we identified the pictures with large areas of weak texture. This is demonstrated on two examples in Figs. 12 and 13, where the algorithm produces a large number of false matches in the homogeneous background, which cannot be distinguished from the true patches containing a relatively small object. We observed analogous behavior in many

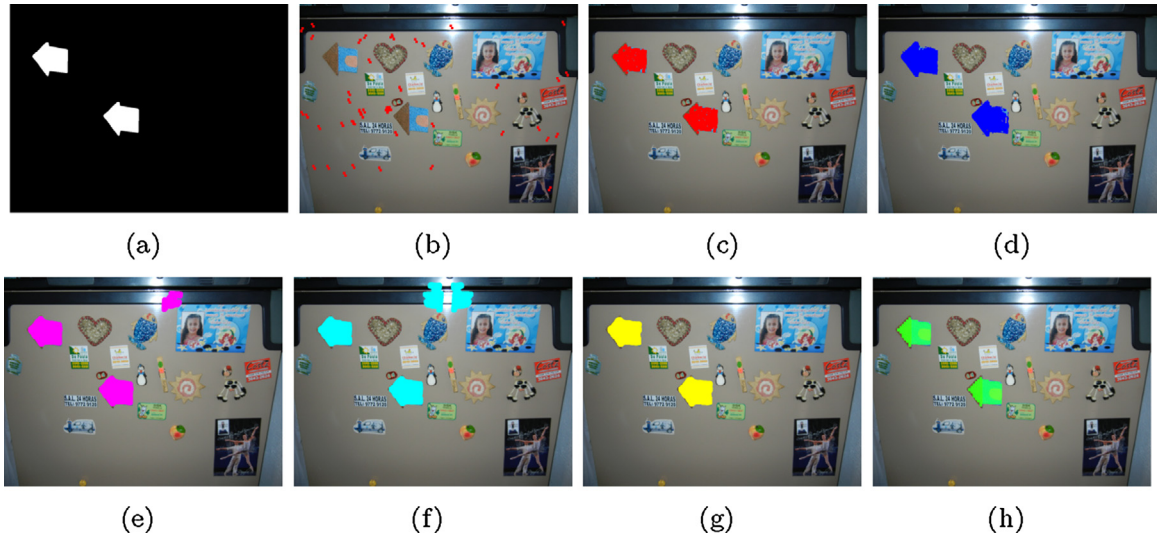


Fig. 9. Example of detection: ground truth from DS1 (a), DCT-basic (b), DCT-tuned (c), SIFT (d), ZM (e), PCT (f), FMT (g), and proposed algorithm with JPEG copy-move constraint (h).

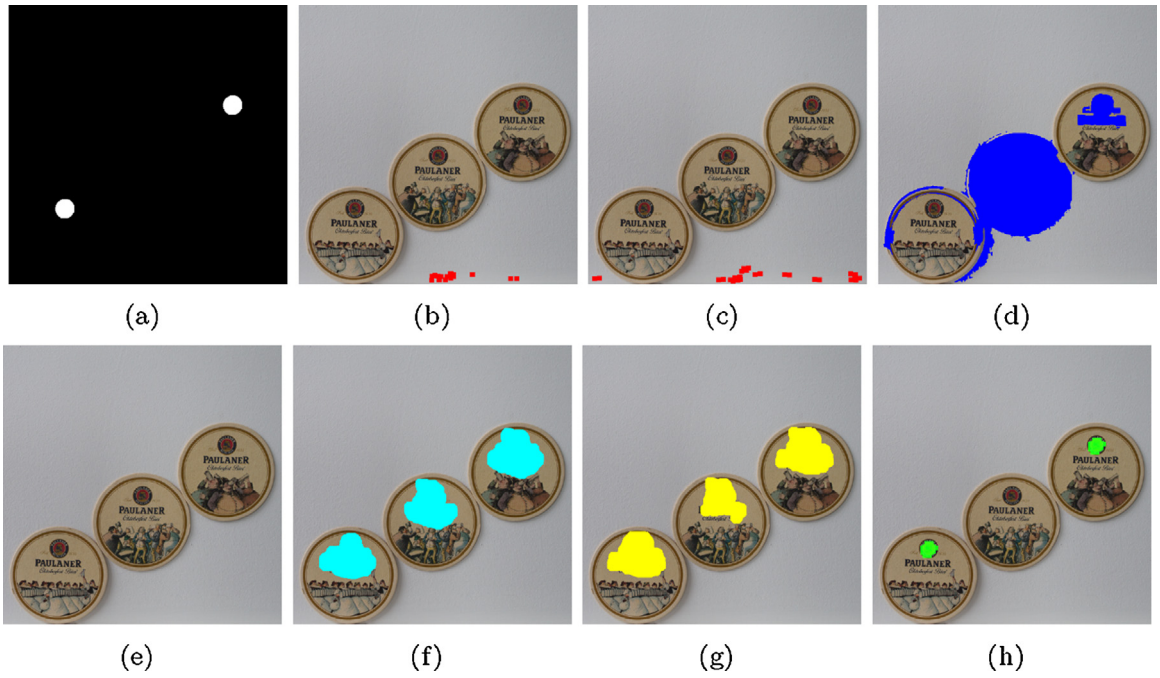


Fig. 10. Example of detection: ground truth from DS2 (a), DCT-basic (b), DCT-tuned (c), SIFT (d), ZM (e), PCT (f), FMT (g), and proposed algorithm with JPEG copy-move constraint (h).

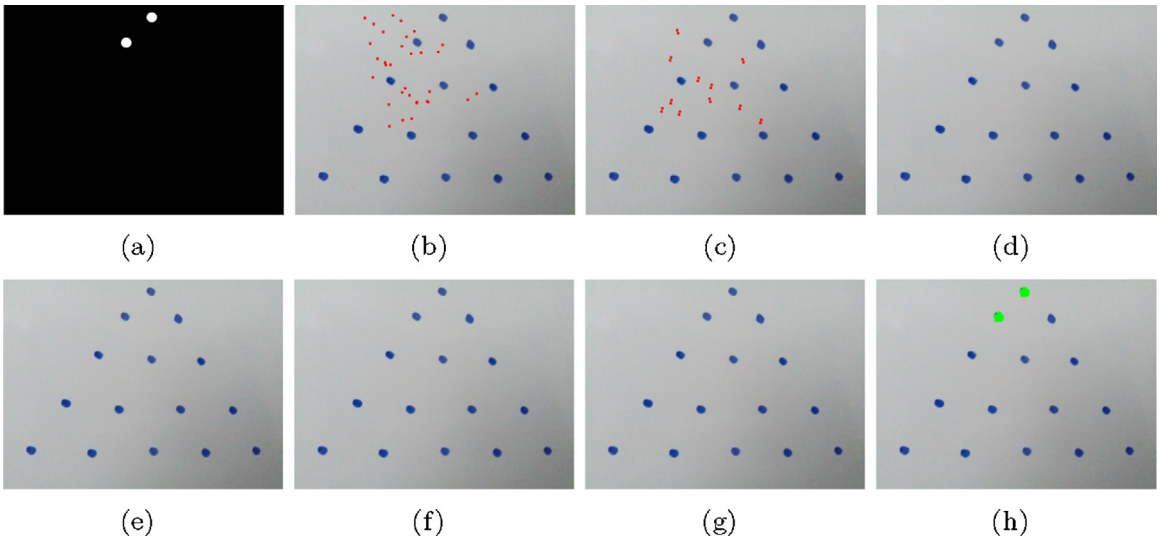


Fig. 11. Example of detection: ground truth from DS3 (a), DCT-basic (b), DCT-tuned (c), SIFT (d), ZM (e), PCT (f), FMT (g), and proposed algorithm with JPEG copy-move constraint (h).

pictures with for example the skies, building facades or road surfaces.

The proposed JPEG copy-move constraint improves results significantly even when used with the simple l_2 distance between patches. It achieves the overall success rate of 100% at the image level and sensitivity 89.91% at the patch level. The main advantage of the JPEG constraint is that it increases precision and sensitivity by reducing the number of false positives (1.55%) even in extremely homogeneous areas like in Figs. 12 and 13. As a result, for larger objects, we get very compact results even without any additional postprocessing, see Figs. 9–11(h). Note that several missing pixels in Figs. 9(h) and 10(h) are caused by considering, for speed reasons, only the closest patch. Using a more elaborate nearest neighbor search algorithm would easily remove them in these examples. They could be also quickly removed by verifying the JPEG constraint for all "holes" in the detected objects.

The computation of JPEG copy-move constraint comes at the cost of additional time, depending on how many candidate pairs we consider. In the simplified algorithm used in our experiments that work with only the closest patch to each 8x8 block on the JPEG grid, the estimate from the end of Section 5 gives a rough upper bound of $4 * 12 + 12 = 60$ JPEG compressions and decompressions of the analyzed image in the worst case, when all the pairs we found are close enough to be considered valid candidates, for example on an image of clear skies. For example, on our PC with Intel Core i5 CPU (3.4 GHz) one JPEG compression/decompression of an image

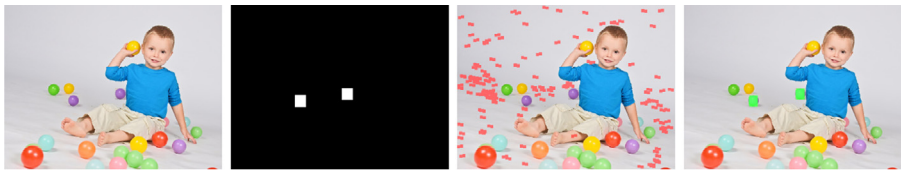


Fig. 12. Reducing false matches. Figure (a) is an original image from DS3, (b) ground truth, (c) result of DCT-tuned, (d) proposed algorithm with JPEG copy-move constraint. The SIFT, ZM, PCT, and FMT method failed completely (found nothing).

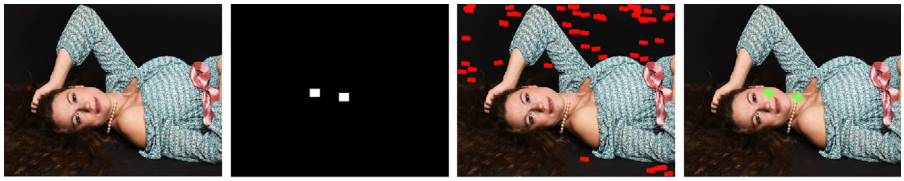


Fig. 13. Reducing false matches. Figure (a) is an original image from DS3, (b) ground truth, (c) result of DCT-tuned, (d) proposed algorithm with JPEG copy-move constraint. The SIFT, ZM, PCT, and FMT method failed completely (found nothing).

of size 512×512 (DS2) using the Matlab JPEG toolbox takes about $1/40s$, i.e. $2 * 60/40 \sim 3s$ in total.

7. Conclusion

In this paper, we propose a method to improve reliability of detecting copy-move forgery in JPEG images. For this purpose, we introduce the JPEG copy-move constraint that can be used to filter out false matches of candidate patches in almost any existing detection algorithm. We show an efficient algorithm to verify this constraint. Since the constraint is exact, i.e. gives no false negatives, results are always better than without this constraint. For color images, the algorithm is derived for both the brightness and chrominance channels. However, to speed up the computation, we work only with brightness, which allows for smaller patches (8×8 instead of 16×16).

Experiments show that the JPEG copy-move constraint is very strong for JPEG quality factors above 90, where it almost completely eliminates false matches. However, the reduction of false positives is considerable already from qualities around 70. The JPEG constraint is useful especially for small objects and areas with indistinct texture, where using the coherence of shift vectors in the postprocessing phase is not sufficient. This phenomenon is less pronounced for larger objects but the statistics in the experimental section show that even there the JPEG constraint helps significantly. The JPEG constraint can also distinguish the copy-move forgery from naturally repeating patterns, which is very hard for block-based methods and probably impossible for the methods based on keypoints.

To demonstrate the influence of the JPEG copy-move constraint, we used a relatively simple algorithm. We can expect that using the constraint with a more elaborated algorithm the results could be even better.

The proposed approach cannot be directly used with geometrical transforms like scale change or rotation. One could imagine an extension to these operations but results probably would not be worth additional complexity of the algorithm. In our opinion, improved detection justifies this restriction, though. In addition, the scenario of pure shift is probably frequent enough to be considered separately. Another limitation of the JPEG constraint is that it requires the moved object to contain at least one JPEG block. On the other hand, smaller objects are probably too hard also for all other methods.

To help both researchers and practitioners further investigate the potential of the JPEG copy-move constraint, we provide a Matlab implementation of the proposed algorithm. The code and our DS3 database are available at: https://github.com/michalsorel/jpegcopymove.

Acknowledgment

This research was partially funded by the Czech Science Foundation, grant GA16-13830S (Michal Šorel) and GA15-16928S (Adam Novozámský).

Proposition 1 Appendix

Proposition 1. *Projection*

$$P_{QCDMx \in \langle y-\frac{1}{2}, y+\frac{1}{2}-\varepsilon \rangle}(z) = z - \frac{1}{k} M^T D^T C^T \operatorname{diag}(q) \cdot \left(QCDM(z) - P_{\langle y-\frac{1}{2}, y+\frac{1}{2}-\varepsilon \rangle}(QCDMz) \right). \tag{10}$$

Proof. The formula for the projection is a consequence of Lemma 1. To be applicable, we must show that AA^T, where $A = QCDM$, is diagonal. First, M^T takes a source patch and expands it to the size of the target patch by filling with zeros and M selects back the pixels of the source patch, therefore $MM^T = I$. Second, let us assume that D is the averaging over non-overlapping windows of $m \times n$ pixels. Then D^T replicates each pixel of a down-sampled image to a $m \times n$ window multiplied by a factor $k = 1/mn$. Since D only averages the values in the window, we are getting $DD^T = I/mn = kI$. Finally, since C is orthogonal, i.e. $CC^T = I$, we are getting $AA^T = QCDMM^T D^T C^T Q^T = QCDD^T C^T Q^T = kQCC^T Q^T/mn = kQQ^T$, which is diagonal. Finally, $A^T (AA^T)^{-1} = M^T D^T C^T Q^T \frac{1}{k} \operatorname{diag}(q^2) = \frac{1}{k} M^T D^T C^T \operatorname{diag}(q)$. □.

Lemma 1. *Projection* $P_{Ax \in \langle b_1, b_2 \rangle}(z) = \arg\min_x \| x - z \|$, *s.t.* $Ax \in \langle b_1, b_2 \rangle$, $b_1 \le b_2$, $A \in R^{m \times n}$, $m \le n$ *full rank,* AA^T *diagonal, can*

be computed as $P_{Ax\in\langle b_1,b_2\rangle}(z) = z - A^T(AA^T)^{-1}(Az - P_{\langle b_1,b_2\rangle}(Az))$, *where* $P_{\langle b_1,b_2\rangle}(y) = \min(\max(b_1, y), b_2)$.

The proof of this lemma can be found in the appendix of [34].

References

[1] J. Fridrich, D. Soukal, J. Lukas, Detection of copy move forgery in digital images, Digital Forensic Research Workshop (2003).

[2] Y. Yang, N.P. Galatsanos, A.K. Katsaggelos, Projection-based spatially adaptive reconstruction of block-transform compressed images, IEEE Trans. Image Process. 4 (7) (1995) 896–908.

[3] V. Christlein, C. Riess, J. Jordan, C. Riess, E. Angelopoulou, An evaluation of popular copy-move forgery detection approaches, IEEE Trans. Inf. Forensics Secur. 7 (6) (2012) 1841–1854, doi:http://dx.doi.org/10.1109/TIFS.2012.2218597.

[4] E. Silva, T. Carvalho, A. Ferreira, A. Rocha, Going deeper into copy-move forgery detection: exploring image telltales via multi-scale analysis and voting processes, J. Vis. Commun. Image Represent. 29 (2015) 16–32, doi:http://dx.doi.org/10.1016/j.jvcir.2015.01.016. http://www.sciencedirect.com/science/article/pii/S1047320315000231.

[5] A. Piva, An overview on image forensics, ISRN Signal Process. 2013 (2013) 1–22.

[6] M.K. Bashar, K. Noda, N. Ohnishi, K. Mori, Exploring duplicated regions in natural images, IEEE Trans. Image Process. PP (99) (2010), doi:http://dx.doi.org/10.1109/TIP.2010.2046599 1–1.

[7] J. Wang, G. Liu, H. Li, Y. Dai, Z. Wang, Detection of image region duplication forgery using model with circle block, 2009 International Conference on Multimedia Information Networking and Security, vol. 1 (2009) 25–29, doi:http://dx.doi.org/10.1109/MINES.2009.142.

[8] S. Bayram, T.H. Sencar, N. Memon, An efficient and robust method for detecting copy-move forgery, 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (2009) 1053–1056, doi:http://dx.doi.org/10.1109/ICASSP.2009.4959768.

[9] B. Mahdian, S. Saic, Detection of copy-move forgery using a method based on blur moment invariants, Forensic Sci. Int. 171 (2–3) (2007) 180–189, doi:http://dx.doi.org/10.1016/j.forsciint.2006.11.002. http://www.sciencedirect.com/science/article/pii/S0379073806006748.

[10] S.-J. Ryu, M.-J. Lee, H.-K. Lee, Detection of copy-rotate-move forgery using Zernike moments, Information Hiding: 12th International Conference, IH 2010, Revised Selected Papers, Calgary, AB, Canada, June 28–30, 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 51–65, doi:http://dx.doi.org/10.1007/978-3-642-16435-4_5.

[11] A.C. Popescu, H. Farid, Exposing Digital Forgeries by Detecting Duplicated Image Regions, Tech. Rep. TR2004-515, Department of Computer Science, Dartmouth College, 2004. www.cs.dartmouth.edu/farid/publications/tr04.html.

[12] Y. Huang, W. Lu, W. Sun, D. Long, Improved DCT-based detection of copy-move forgery in images, Forensic Sci. Int. 206 (1-3) (2011) 178–184, doi:http://dx.doi.org/10.1016/j.forsciint.2010.08.001. http://www.sciencedirect.com/science/article/pii/S0379073810003890.

[13] J.S. Beis, D.G. Lowe, Shape indexing using approximate nearest-neighbour search in high-dimensional spaces, Computer Vision and Pattern Recognition, 1997, Proceedings, IEEE, 1997, pp. 1000–1006.

[14] M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, VISAPP (1) 2 (2009) 331–340.

[15] D. Cozzolino, G. Poggi, L. Verdoliva, Efficient dense-field copy-move forgery detection, IEEE Trans. Inf. Forensics Secur. 10 (11) (2015) 2284–2297.

[16] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.

[17] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395, doi:http://dx.doi.org/10.1145/358669.358692.

[18] H. Huang, W. Guo, Y. Zhang, Detection of copy-move forgery in digital images using sift algorithm, Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008, PACIIA'08, vol. 2 (2008) 272–276, doi:http://dx.doi.org/10.1109/PACIIA.2008.240.

[19] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110, doi:http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[20] X. Bo, W. Junwen, L. Guangjie, D. Yuewei, Image copy-move forgery detection based on surf, 2010 International Conference on Multimedia Information Networking and Security (2010) 889–892, doi:http://dx.doi.org/10.1109/MINES.2010.189.

[21] H. Bay, A. Ess, T. Tuytelaars, L.V. Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359, doi:http://dx.doi.org/10.1016/j.cviu.2007.09.014 similarity Matching in Computer Vision and Multimedia. http://www.sciencedirect.com/science/article/pii/S1077314207001555.

[22] L. Yu, Q. Han, X. Niu, Feature point-based copy-move forgery detection: covering the non-textured areas, Multimed. Tools Appl. 75 (2) (2016) 1159–1176.

[23] E. Ardizzone, A. Bruno, G. Mazzola, Copy-move forgery detection by matching triangles of keypoints, IEEE Trans. Inf. Forensics Secur. 10 (10) (2015) 2084–2094, doi:http://dx.doi.org/10.1109/TIFS.2015.2445742.

[24] C.-M. Pun, X.-C. Yuan, X.-L. Bi, Image forgery detection using adaptive oversegmentation and feature point matching, IEEE Trans. Inf. Forensics Secur. 10 (8) (2015) 1705–1716, doi:http://dx.doi.org/10.1109/TIFS.2015.2423261.

[25] J. Li, X. Li, B. Yang, X. Sun, Segmentation-based image copy-move forgery detection scheme, IEEE Trans. Inf. Forensics Secur. 10 (3) (2015) 507–518.

[26] M. Zandi, A. Mahmoudi-Aznaveh, A. Talebpour, Iterative copy-move forgery detection based on a new interest point detector, IEEE Trans. Inf. Forensics Secur. 11 (11) (2016) 2499–2512, doi:http://dx.doi.org/10.1109/TIFS.2016.2585118.

[27] H. Li, W. Luo, X. Qiu, J. Huang, Image forgery localization via integrating tampering possibility maps, IEEE Trans. Inf. Forensics Secur. 12 (5) (2017) 1240–1252, doi:http://dx.doi.org/10.1109/TIFS.2017.2656823.

[28] W. Wang, J. Dong, T. Tan, Exploring dct coefficient quantization effects for local tampering detection, IEEE Trans. Inf. Forensics Secur. 9 (10) (2014) 1653–1666.

[29] W.B. Pennebaker, J.L. Mitchell, JPEG Still Image Data Compression Standard, 1st edition, Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[30] M. Šorel, M. Bartoš, Fast bayesian jpeg decompression and denoising with tight frame priors, IEEE Trans. Image Process. 26 (1) (2017) 490–501, doi:http://dx.doi.org/10.1109/TIP.2016.2627802.

[31] H.H. Bauschke, J.M. Borwein, On projection algorithms for solving convex feasibility problems, SIAM Rev. 38 (3) (1996) 367–426.

[32] J. Douglas, H.H. Rachford, On the numerical solution of heat conduction problems in two and three space variables, Trans. Am. Math. Soc. (1956) 421–439.

[33] J. Eckstein, D.P. Bertsekas, On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators, Math. Program. 55 (1992) 293–318, doi:http://dx.doi.org/10.1007/BF01581204.

[34] M. Šorel, M. Bartoš, Efficient jpeg decompression by the alternating direction method of multipliers, International Conference on Pattern Recognition, ICPR'16 (2016).

[35] D. Tralic, I. Zupancic, S. Grgic, M. Grgic, CoMoFoD – new database for copy-move forgery detection, Proceedings ELMAR-2013 (2013) 49–54.

[36] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, G. Serra, A sift-based forensic method for copy-move attack detection and transformation recovery, IEEE Trans. Inf. Forensics Secur. 6 (3) (2011) 1099–1110, doi:http://dx.doi.org/10.1109/TIFS.2011.2129512.