JPEG Compression Model in Copy-move Forgery Detection

Adam Novozámský and Michal Šorel
The Czech Academy of Sciences, Institute of Information Theory and Automation
Pod Vodárenskou věží 4, 182 08, Prague 8, Czechia
e-mail: novozamsky@utia.cas.cz, sorel@utia.cas.cz

***Abstract*—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. The common way to manipulate image content is copying an object and pasting in another location of the same image. In this paper, we describe a new idea for the detection of this type of forgery in JPEG images, where the compression significantly degrades detection by popular algorithms. We derive a JPEG-based constraint that any pair of patches must satisfy to be considered a valid candidate for tampered area. We propose also efficient algorithm to verify the constraint that 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.**

***Index Terms*—Image forensics, Copy-move forgery detection, Image tampering, JPEG, Quantization constraint set.**

I. Introduction

One of the most common image manipulations is the copy-move forgery, 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 makes detection more difficult. 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.

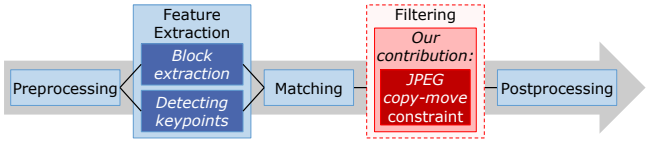


Fig. 1. Typical processing pipeline of copy-move forgery detection algorithms. The main contribution of this paper is filtering using the JPEG copy-move constraint, which increases sensitivity and reduces significantly the number of false matches.

II. Related work

Copy-move forgery detection was first proposed by Fridrich et al. [1]. 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 [2], [3] comparing many popular algorithms.

The typical processing pipeline used by most CMFD methods is shown in Fig. 1 (derived from Fig. 2 in [2]). 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 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. [4] 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. [5] used a circle block model, Bayram et al. [6] added a scale invariance by

features obtained from the Fourier-Mellin transform. Several authors worked with moment invariants. Mahdian and Saic [7] suggested 24 blur invariants, Ryu et al. [8] used the Zernike moments.

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. [9] reduced the feature matching space by the PCA applied on blocks. Bashar et al. [4] modified their algorithm to use the kernel PCA. Huang et al. [10] 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 kd-trees, mostly used in the best-bin-first variant [11], which is applicable in both block and keypoint-based algorithms. Comparison [2] uses multiple randomized kd-trees [12]. Another fast iterative randomized technique for computing the approximate nearest-neighbor search can be found in [13].

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 sky, 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 [14] and RANSAC [15]. 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. [16] came up with a CMFD method based on the SIFT features [17], Bo et al. [18] used SURF [19]. 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 [20].

III. JPEG COMPRESSION

JPEG is undoubtedly the most widespread format for efficient storage of image data [21]. 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 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 [22], [23]. 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. $Q = \operatorname{diag}(1/q)$. 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,

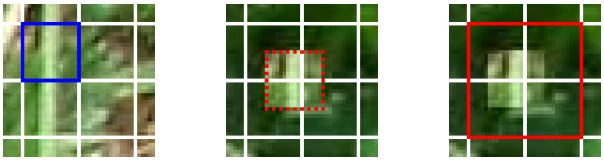
$$y = [QCDx]\,, \tag{2}$$

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

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

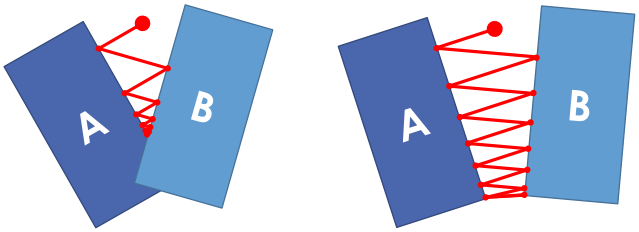
where $k = 1/4$ for the 2×2 down-sampling window, $k = 1/2$ for 2×1 and 1×2, and $k = 1$ for grayscale (no down-sampling).

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.



(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.



(a) (b)

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.

IV. 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). Equation (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 $\left\langle y_2 - \frac{1}{2}, y_2 + \frac{1}{2} \right)$ 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

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

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. Nevertheless, in the following section, we derive an efficient algorithm that verifies the JPEG constraint iteratively.

V. ALGORITHM

The simplest way to find a point in the intersection of arbitrary two convex 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) [24]. 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 [25], which is a special case of the alternating direction method of multipliers (ADMM) [26]. If exists, the intersection $A \cap B$ can be found by iterating the following three steps

$$x \leftarrow P_A\left(a + d\right), \tag{6}$$
$$a \leftarrow P_B\left(x - d\right), \tag{7}$$
$$d \leftarrow d - \left(x - a\right), \tag{8}$$

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

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

Fig. 4. The algorithm for verification of the JPEG copy-move constraint, i. e. the possibility of copy-move between two image patches.

(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

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

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. The proof is described in [27]).

The algorithm is summarized in Fig. 4. 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.

VI. Experiments

In this section, we show how our 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.

TABLE I
Comparison of methods on dataset DS1, 23 images, 13645 patches in ground-truth.

	Image level		Patch level TPs		Patch level FPs	
Method	**#**	**%**	**#**	**sens. [%]**	**#**	**FDR [%]**
DCT - basic	16	69.57	2697	19.77	1855	40.75
DCT - tuned	23	100.00	11644	85.34	89	0.76
SIFT	21	91.30	9127	66.89	1641	15.24
ZM	23	100.00	12988	95.19	776	5.64
PCT	23	100.00	12956	94.95	777	5.66
FMT	23	100.00	12788	93.72	630	4.70
OUR	**23**	**100.00**	**12432**	**91.11**	**56**	**0.45**

TABLE II
Comparison of methods on dataset DS2, 40 images, 9288 patches in ground-truth.

	Image level		Patch level TPs		Patch level FPs	
Method	**#**	**%**	**#**	**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**

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. As a representative of methods based on keypoints, we use the SIFT-based algorithm [28], available online. For a wider comparison we also added the approach of Cozzolino et al. [13], 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

TABLE III
Comparison of methods on dataset DS3 (difficult cases), 18 images, 9850 patches in ground-truth.

	Image level		Patch level TPs		Patch level FPs	
Method	**#**	**%**	**#**	**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**

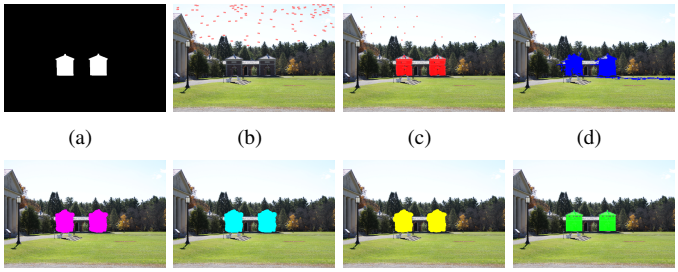


Fig. 5. Example of detection: ground truth from DS1 (a), DCT-basic (b), DCT-tuned (c), SIFT (d), ZM (e), PCT (f), FMT (g), and proposed algorithm with JPEG copy-move constraint (h).

use two standard image datasets [29], [3]. The performance of other methods on the same datasets is analyzed in [2], [3]. 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. [3],
- **DS2:** 40 images, CoMoFoD dataset [29],
- **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, 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. In our experiments, we see the similar behavior for qualities above around 90.

Quantitative results for all the datasets are given in Tables I-III. 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 I-III) 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. 5, 6, and 7 illustrate typical outputs of the tested approaches.

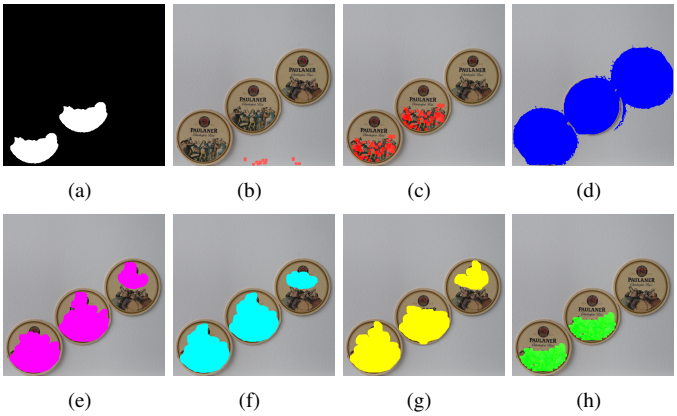


Fig. 6. Example of detection: ground truth from DS2 (a), DCT-basic (b), DCT-tuned (c), SIFT (d), ZM (e), PCT (f), FMT (g), and proposed algorithm with JPEG copy-move constraint (h).

At the image level, the SIFT-based method achieves a decent success rate of 91% on DS1 but often fails on DS2 and DS3 (65% and 33%). Moreover, this method produces a large number of false matches at the patch level, even for relatively easy examples like in Fig. 5(d). In addition, Fig. 6(d) and Fig. 7(d) illustrate how repeated motives can cause a complete failure of this algorithm, a problem mentioned already in the original paper [28]. We see similar results also for all three methods of Cozzolino [13]. 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 percent only.

Using the DCT-tuned, we achieved the overall success rate over 92% at the image level and sensitivity over 82% at the patch level. As the weakest point of this approach, we identified the pictures with large areas of weak texture. The proposed JPEG copy-move constraint improves results significantly. It achieves the overall success rate of 100% at the image level and sensitivity 89.91 % at the patch level, by reducing the number of false positives (1.01%) .

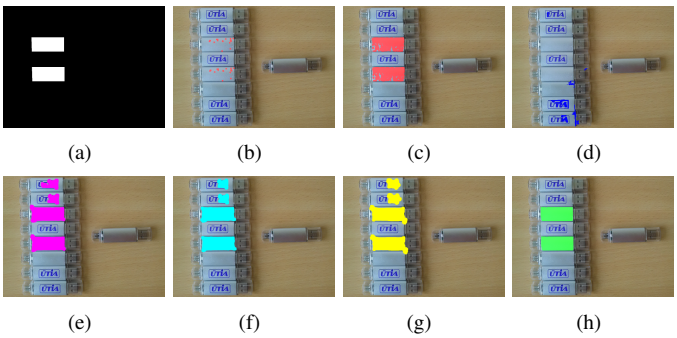


Fig. 7. Example of detection: ground truth from DS3 (a), DCT-basic (b), DCT-tuned (c), SIFT (d), ZM (e), PCT (f), FMT (g), and proposed algorithm with JPEG copy-move constraint (h).

VII. 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. Since the constraint is exact, i. e. gives almost no false negatives, results are always better than without this constraint.

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.

Acknowledgment

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

References

[1] J. Fridrich, D. Soukal, and J. Lukas, "Detection of copy move forgery in digital images," in *Digital Forensic Research Workshop*, Aug. 2003.

[2] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, Dec 2012.

[3] E. Silva, T. Carvalho, A. Ferreira, and A. Rocha, "Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting processes," *Journal of Visual Communication and Image Representation*, vol. 29, pp. 16 – 32, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047320315000231

[4] M. K. Bashar, K. Noda, N. Ohnishi, and K. Mori, "Exploring duplicated regions in natural images," *IEEE Transactions on Image Processing*, vol. PP, no. 99, pp. 1–1, 2010.

[5] J. Wang, G. Liu, H. Li, Y. Dai, and Z. Wang, "Detection of image region duplication forgery using model with circle block," in *2009 International Conference on Multimedia Information Networking and Security*, vol. 1, 2009, pp. 25–29.

[6] S. Bayram, T. H. Sencar, and N. Memon, "An efficient and robust method for detecting copy-move forgery," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1053–1056.

[7] B. Mahdian and S. Saic, "Detection of copy-move forgery using a method based on blur moment invariants," *Forensic Science International*, vol. 171, no. 2-3, pp. 180–189, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0379073806006748

[8] S.-J. Ryu, M.-J. Lee, and H.-K. Lee, *Information Hiding: 12th International Conference, IH 2010, Calgary, AB, Canada, June 28-30, 2010, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. Detection of Copy-Rotate-Move Forgery Using Zernike Moments, pp. 51–65.

[9] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting duplicated image regions," Department of Computer Science, Dartmouth College, Tech. Rep. TR2004-515, 2004. [Online]. Available: www.cs.dartmouth.edu/farid/publications/tr04.html

[10] Y. Huang, W. Lu, W. Sun, and D. Long, "Improved dct-based detection of copy-move forgery in images," *Forensic Science International*, vol. 206, no. 1-3, pp. 178–184, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0379073810003890

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

[12] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP (1)*, vol. 2, pp. 331–340, 2009.

[13] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy–move forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2284–2297, 2015.

[14] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[16] H. Huang, W. Guo, and Y. Zhang, "Detection of copy-move forgery in digital images using sift algorithm," in *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*, vol. 2, Dec 2008, pp. 272–276.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[18] X. Bo, W. Junwen, L. Guangjie, and D. Yuewei, "Image copy-move forgery detection based on surf," in *2010 International Conference on Multimedia Information Networking and Security*, 2010, pp. 889–892.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008, similarity Matching in Computer Vision and Multimedia. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314207001555

[20] L. Yu, Q. Han, and X. Niu, "Feature point-based copy-move forgery detection: covering the non-textured areas," *Multimedia Tools and Applications*, vol. 75, no. 2, pp. 1159–1176, 2016.

[21] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[22] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Projection-based spatially adaptive reconstruction of block-transform compressed images," *Image Processing, IEEE Transactions on*, vol. 4, no. 7, pp. 896–908, 1995.

[23] M. Šorel and M. Bartoš, "Fast bayesian jpeg decompression and denoising with tight frame priors," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 490–501, Jan 2017.

[24] H. H. Bauschke and J. M. Borwein, "On projection algorithms for solving convex feasibility problems," *SIAM review*, vol. 38, no. 3, pp. 367–426, 1996.

[25] J. Douglas and H. H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables," *Transactions of the American mathematical Society*, pp. 421–439, 1956.

[26] J. Eckstein and D. P. Bertsekas, "On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Program.*, vol. 55, pp. 293–318, June 1992.

[27] M. Šorel and M. Bartoš, "Efficient jpeg decompression by the alternating direction method of multipliers," in *International Conference on Pattern Recognition, ICPR'16*, 2016.

[28] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "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, Sept 2011.

[29] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic, "CoMoFoD-New database for copy-move forgery detection," in *Proceedings ELMAR-2013*, Sept 2013, pp. 49–54.