COPY-MOVE FORGERY DETECTION BASED ON PATCHMATCH

Davide Cozzolino, Giovanni Poggi, Luisa Verdoliva

Universitá Federico II di Napoli, DIETI, 80125 Naples Italy

ABSTRACT

In this work we propose a new algorithm for copy-move forgery detection and localization, based on the fast computation of a dense nearest-neighbor field. To this end, we use PatchMatch, an iterative randomized algorithm for nearest-neighbor search, which exploits the regularity of natural images to converge very rapidly to a near-optimal and smooth field. We modify the basic algorithm to gain robustness against rotations, while keeping the original computational efficiency. Experimental results show the proposed technique to outperform almost uniformly all tested reference techniques in terms of both accuracy and speed.

***Index Terms*—** Digital Image forensics, copy-move forgery detection, patchmatch.

1. INTRODUCTION

In recent years, research on image forensics has focused especially on passive techniques which detect manipulations based only on the analysis of the image content. Many different approaches have been proposed to detect and localize a forgery [1], and much attention has been devoted to the copy-move case, where portions of the image are cut and pasted elsewhere in the same image to duplicate or hide objects of interest.

In fact, this is probably the simpler and certainly the most common form of image manipulation. Identical or similar regions in the image are discovered based on *matching*. Virtually all such algorithms are based, although with many variations, on the following three steps: *featuring:* suitable features are associated with all pixels, or with a limited set of keypoints; *matching:* for each pixel of interest, the best matching pixel is located based on the associated features; *post-processing:* the displacement field is filtered and processed to detect actual copy-moved regions.

The techniques proposed in the literature can be grouped in two large classes, depending on whether the matching is performed for each pixel, generating a dense displacement field, or for just some selected keypoints, in which case the field is sparse. This preliminary choice impacts heavily on complexity. In the first case, a matching score must be computed, in principle, for all couples of pixels, with a complexity proportional to N^2, with N the image size. Reducing this complexity is probably the main issue in this class of algorithms. In the second case, instead, after locating a small number $M \ll N$ of image keypoints, the matching is restricted to these points, and the complexity, proportional to M^2 reduces hugely.

The first approach is typically performed through some matching procedure applied to each block to detect the most similar block in the image. Given the potentially huge complexity, featuring is required to be intrinsically simple, and to produce features as short as possible. Using the native RGB pixel values is a legitimate choice, which satisfies obviously the first requirement, but not the second. In the literature, features are typically extracted in some transform domain, like PCA, DCT, Wavelet [2], aimed at reducing their dimensionality, through decorrelation of coefficients, and/or gaining robustness with respect to noise, compression, and other common forms of distortion. A major challenge is to detect copy-moves in the presence of rescaling and rotation. In [3], a Fourier Mellin Transform (FMT) is applied on image blocks, and the magnitudes of Fourier coefficients are then re-sampled and transformed into log-polar coordinates, obtaining feature vectors invariant to rotation, scale and translation. In [4] the mapping into log-polar coordinates is carried out in the spatial domain producing a one-dimensional descriptor robust to geometric transformations. Another interesting approach [5] is based on the rotation-invariant Zernike moments, which turn out to guarantee increased robustness also w.r.t. additive white Gaussian noise and JPEG compression.

In keypoint-based methods, where the number of matching scores to be computed is orders of magnitude smaller, much more complex and longer features can be used, with intrinsic invariance properties, such as SIFT, employed for example in [6] and [7], or SURF, used in [8]. These techniques are usually much faster than those based on dense matching. However, they are also intrinsically less accurate. In particular, since keypoints are associated with high entropy regions, copy-moves involving only smooth regions remain mostly undetected. This is not a minor problem, considering that quite often these manipulations concern pieces of background, generally smooth or highly textured, which are copied to hide objects of possible interest. As a consequence, there is a clear performance gap w.r.t. dense-matching techniques as shown in the benchmarking paper [9].

Once established that dense-matching is the most promising approach for reliable copy-move detection, the major focus of research should be on speeding up the matching phase without impairing the quality of the displacement field. Exhaustive search is obviously not an option, barring the trivial case of very small images. Some early techniques [3, 4] relied on lexicographic sorting, which is fast but rather prone to errors when the image quality is not high. More recently, approximate nearest neighbor search based on kd-trees and locality sensitive hashing has gained some popularity for copy-move detection [10, 11, 9, 5], as it is more robust to image impairment and can be still pretty fast.

In this paper we propose a new fast and reliable technique for copy-move detection based on the Patchmatch algorithm. Patchmatch was proposed originally in [12] to compute the approximate nearest neighbor field (NNF) of an image, with application to a number of editing tasks. Leveraging on the intrinsic regularity of natural images, it performs an iterative randomized search procedure which converges very quickly to a smooth and reliable NNF. The authors themselves generalized the algorithm in [13] under several points of view, including the ability to find matches after resizing and rescaling of objects. Given its properties, PatchMatch looks as the perfect tool to carry out copy-move detection, and indeed this possible application, among many others, was pointed out already in [13].

We devised a first Patchmatch-based copy-move algorithm in the context of the First IEEE Forensics Challenge [14, 15, 16]. Here, we propose a much more sophisticated algorithm using state-of-the-art tools for all phases of the detection process. In particular, we modify the PatchMatch algorithm itself in order to deal effectively with rotations without significantly increasing the processing time and without impairing the performance in all other situations. The resulting technique is extremely fast, accurate and robust, providing a competitive performance under all major conditions of practical interest.

In the rest of the paper, after providing some more detail on the PatchMatch algorithm, we describe the proposed copy-move detection technique, and in particular the improved managing of rotations, and finally we present experiments results.

2. THE PATCHMATCH ALGORITHM

PatchMatch [12] is a fast randomized algorithm which finds dense approximate nearest neighbor matches between image patches.

A NNF is defined formally as a function $f : I \to R^2$ which associates an offset or displacement $f(x, y) \in R^2$ with each pixel (x, y) of the image I. Switching to a more compact vectorial notation, $\mathbf{z} = (x, y)$ is the reference pixel of patch $P(\mathbf{z})$, and therefore $\mathbf{z} + f(\mathbf{z})$ points to the nearest neighbor patch. The exact nearest neighbor patch minimizes a suitable distance measure $D(\cdot, \cdot)$, that is

$$f(\mathbf{z}) = \arg \min_{\phi:\mathbf{z}+\phi\in\Omega, f\neq 0} D(P(\mathbf{z}), P(\mathbf{z} + \phi)) \tag{1}$$

where Ω is the image support. Finding the *exact* NNF, however, is computationally infeasible so we look for an approximation of it, and keep using the acronym NNF for the sake of simplicity. It should be underlined that, for most applications, the exact NNF is not necessarily the best one, and regularity is an additional important property. This is well-known, for example, in the context of video coding, where a regular motion-compensation field is much more compressible, hence preferable, than a more chaotic one.

Patchmatch is an iterative algorithm. Initially, displacements are set by sampling at random and uniformly the image support: $\mathbf{z} + f(\mathbf{z}) \sim U(\Omega)$. Most of these displacements are of little use, but it is very likely that a certain number of them will be optimal or near-optimal. The main idea of PatchMatch is to propagate good displacements.

In the generic iteration, in fact, the image is raster scanned top-down and left-to-right, and for each pixel $\mathbf{z}$ the current offset is updated as

$$f(\mathbf{z}) = \arg \min_{\phi\in\{f(\mathbf{z}),f(\mathbf{z}^u),f(\mathbf{z}^l)\}} D(P(\mathbf{z}), P(\mathbf{z} + \phi)) \tag{2}$$

where $\mathbf{z}^u$ and $\mathbf{z}^l$ are the pixels above and to the left, respectively, of the current one. Therefore, if a good displacement is available for a given pixel of a region with constant displacement, this will very quickly propagate, filling the whole region below and to the right of it. To avoid biases, the scanning order is then reversed (bottom-up and right-to-left) at every other iteration, involving pixels below $\mathbf{z}^d$ and to the right $\mathbf{z}^r$ of the current one. Of course, this idea is based on the fundamental hypothesis that the true NNF is mostly regular, composed of a relatively small number of regions with the same displacement, and would be totally ineffective, for example, with a white noise field.

The above propagation procedure is obviously greedy, and as such suboptimal. Therefore, to minimize the risk of being trapped in local minima, after the updating of eq.(2), a similar one is tried based however on random sampling of the NNF. The new candidates $f_i, i = 1, \ldots, L$ are chosen as

$$f_i = f(\mathbf{z}) + 2^{i-1}R_i \tag{3}$$

where R_i is a bi-dimensional random variable uniform in $\{-1, 0, 1\} \times \{-1, 0, 1\}$ excluding (0,0). In practice, new candidates are taken, one for each grid, on square grids centered on $f(\mathbf{z}) = f_0$ and with exponentially increasing radii. The random-search updating reads therefore as

$$f(\mathbf{z}) = \arg \min_{\phi\in\{f_0,f_1,\ldots,f_L\}} D(P(\mathbf{z}), P(\mathbf{z} + \phi)) \tag{4}$$

For an image of, say, 1024×1024 pixels, $L \leq 10$. Considering that the procedure typically converges after a few iterations, the whole computational load is in the order of 10^2 patch distance computation as opposed to 10^6 for full-search, which fully explains the algorithm speed.

3. DEALING WITH ROTATION: A MODIFIED PATCHMATCH FOR PIECE-WISE LINEAR NNFS

The basic algorithm described above finds only a single nearest-neighbor, and does not deal with scale changes and rotations. In a subsequent paper [13], however, the same authors generalized and extended it under several respects. The Generalized PatchMatch algorithm is able to find k nearest neighbors, instead of just one, to search across scales and rotations, going beyond mere translations, and to match patches based on arbitrary descriptors and distances, not just sum-of-squared-differences on patch colors. It must be pointed out, however, that the three extensions are mutually exclusive, and their combination does not seem straightforward.

To deal with rescaling and rotation, generalized PatchMatch looks for nearest neighbors in a four-dimensional space (x, y, θ, s), with θ the rotation angle and s the scale factor. Of course, this extended search increases somewhat the complexity, but the resulting algorithm can be still considered relatively fast. However, the experimental analysis described in next Section shows this version of PatchMatch to cause a significant loss in the forgery detection performance, definitely not worth the increased robustness against rotations. Our conjecture is that with the enlarged optimization space, a large number of suboptimal matchings are available, which trap the algorithm into local minima. The random search itself becomes probably less effective in such a large space, and does not help escaping from such minima.

Given these dismaying results, we propose here a simple modification of the basic PatchMatch algorithm which deals pretty well with rotations without impairing the performance in their absence. To deal with resizing, instead, we will rely on the intrinsic robustness of the algorithm for scales close to 1 and, if necessary, on brute-force search, which makes sense for just one dimension.

The proposed modification concerns exclusively the propagation phase, namely, the displacement updating step of eq.(2). In the original algorithm, the current displacement for pixel $\mathbf{z}$ is compared with two other candidate displacements, those available for pixels z^u and z^l which precede z along rows and columns in the scanning order. In practice, the displacements $f(\mathbf{z}^u)$ and $f(\mathbf{z}^l)$ can be seen as the causal zero-order predictions of $f(\mathbf{z})$ along image rows and columns. Of course, zero-order predictors are effective only in constant regions, namely, in our context, in the presence of copy-moves with rigid translations, where the NNF is uniform. Since rotated copy-moves correspond instead to a linearly varying NNF, a first-order prediction should work correctly in this case. We therefore

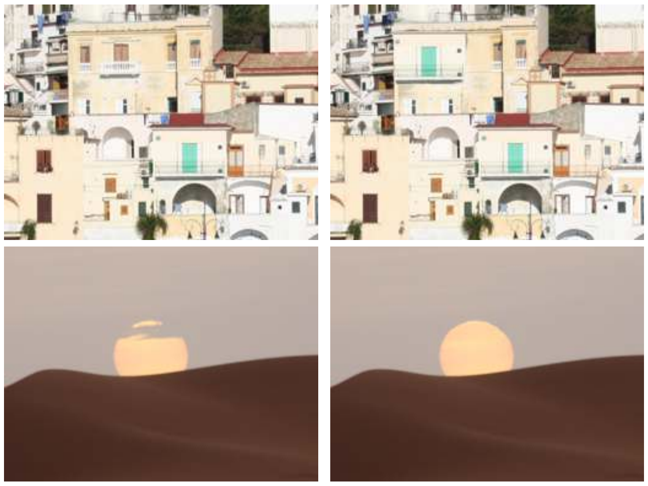
Fig. 1: Examples of genuine and forged images from our dataset.

enlarge the set of candidates to be used in equation (2) including also

$$f^1(\mathbf{z}^u) = f(\mathbf{z}^u) + \Delta f(\mathbf{z}^u) \quad (5)$$
$$f^1(\mathbf{z}^l) = f(\mathbf{z}^l) + \Delta f(\mathbf{z}^l) \quad (6)$$

where

$$\Delta f(\mathbf{z}^u) = f(\mathbf{z}^u) - f(\mathbf{z}^{uu}) \quad (7)$$

$\mathbf{z}^{uu}$ is the pixel above $\mathbf{z}^u$, and similar definitions hold for the other predictor. Therefore, to keep reasoning on the vertical first-order predictor, if $f(\mathbf{z}^u)$ and $f(\mathbf{z}^{uu})$ follow correctly the local affine transformation induced by the rotated copy-move, the correct NNF will quickly propagate to the rest of the interested region within two iterations. Moreover, thanks to the zero-order predictor, and the random sampling of eq.(3), it is not difficult to reach the required initial conditions which triggers the whole process.

It is worth underlining that this modified version of PatchMatch, dealing in general with NNFs that are piece-wise linear, as opposed to piece-wise constant, can be expected to deal effectively also with scale changes, provided a scale-invariant feature is used.

4. EXPERIMENTAL RESULTS

To assess the performance of the proposed PatchMatch-based methods, also in comparison with established reference techniques, we carried out a number of experiments on a dataset available online (www.grip.unina.it) of 80 images of dimension 768×1024-pixel. For each image we generated a realistic copy-move forgery by rigid translation and the corresponding forgery map. Then we considered, one at a time, the most typical deviations from this ideal condition, that is, noising, JPEG compression, rotation and rescaling, varying the main parameter of interest, e.g., the JPEG quality factor, and generating the corresponding forged images with the help of the software tool proposed in [9]. Two examples of genuine and forged images are shown in Fig.1.

As synthetic measures of performance we use the F-measure, which can be defined as

$$\text{FM} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FN} + \text{FP}} \quad (8)$$

where TP (true positive), FN (false negative), and FP (false positive) count, respectively, the number of detected forged pixels, undetected forged pixels, and wrongly detected genuine pixels. Results are then averaged on all 80 images. Complexity is measured in terms of average running time on a desktop PC with two 2 GHz Intel Xeon cores, used in single-thread modality.

4.1. Performance of the proposed technique

The proposed technique follows the general three-step structure outlined in Section 1, with feature extraction, matching, and post-processing. As features we use either the original RGB values or the Zernike moments (ZM), always with relatively large blocks of 16×16 pixels. Although the Zernike moments possess useful rotation invariance properties and have proven quite robust to various forms of processing, the generalized version of PatchMatch which deals with rotation and rescaling works only with RGB values, so we must necessarily include this combination. For the matching step, we use of course PatchMatch, in the basic (B-PM) and generalized (G-PM) versions, and in the modified (M-PM) version proposed here. For the post-processing we resort to standard state-of-the-art tools. In particular, spurious matches are filtered out, eliminating also matches with displacement shorter than a given threshold. Then, surviving matches are collected in groups that follow the same transformation pattern, by means of the Same Affine Transformation Selection (SATS) algorithm proposed in [11]. Overall, we consider 4 techniques, with all the meaningful matching+feature combinations of the set $\{$B-PM, G-PM, M-PM$\}\times\{$RGB, ZM$\}$.

Fig.2 shows, for all these combinations, the average F-measure computed after various types of image impairments. The basic algorithm (B-PM+RGB) provides a very good performance in general, with $F = 0.906$ in case of simple translation. Moreover, it is quite robust to noise and also to limited rotation and resizing. Performance is instead relatively poor in the presence of JPEG compression, and drops sharply for scale changes above 20% and rotation angles beyond 10 degrees. By replacing RGB values with Zernike moments (B-PM+ZM) we obtain a much higher robustness to JPEG compression, because of the intrinsic filtering associated with the more compact ZM representation. In addition, thanks to the rotation invariance of the moments, performance improves in the presence of rotation, with angles up to about 30 degrees. For larger angles, the F-measure drops again, due to the inability of the propagation phase to follow rotations. This problem is eventually solved by resorting to Generalized PatchMatch (G-PM-RGB), which provides a nearly constant F-measure over all rotation angles. Unfortunately, we also observe a significant performance loss in most other situations, including pure translation with an F-measure of 0.808 as opposed to the 0.906 obtained with the basic version. As said before, this is probably due to the higher probability of finding local minima in the higher-dimensional search space. On the contrary, the proposed technique with modified PatchMatch and Zernike moments (M-PM+ZM) shows a remarkable robustness to rotation at all angles, and provides the very same performance of the basic version (B-PM+ZM) in all other situations, with a very limited increase in complexity.

Detailed results on computational efficiency, measured by average CPU-time, are reported in Table 1. Computing ZM features has a small initial cost, which is largely compensated in the matching phase, since shorter feature vectors are used. The use of generalized PatchMatch causes, as expected, a significant increase in the matching time and also in the post-processing phase, probably because the NNF is less regular.

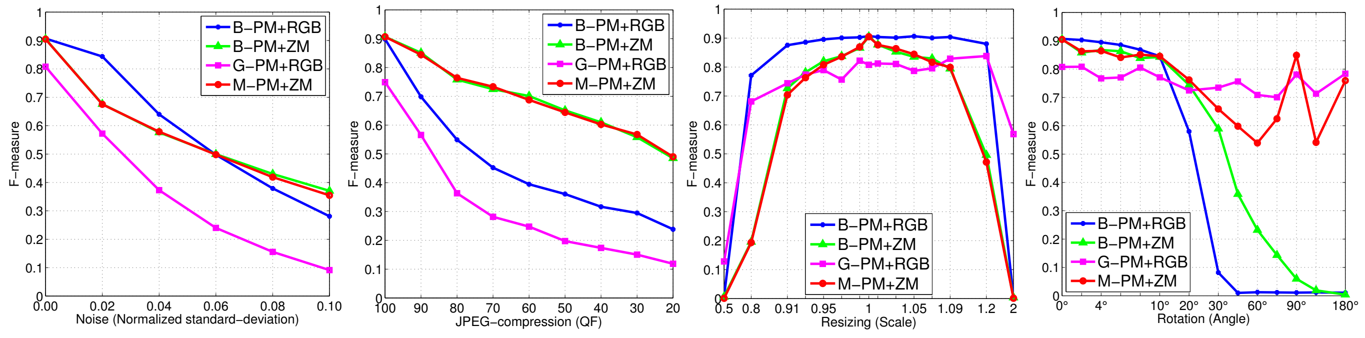

Fig. 2: F-measure of PatchMatch-based techniques computed on the test dataset after AWGN addition, JPEG compression, resizing, and rotation.

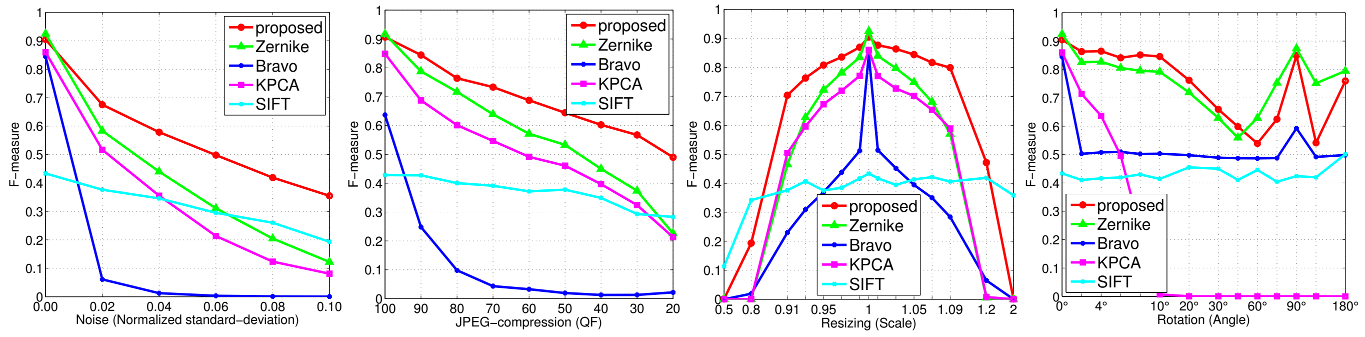

Fig. 3: F-measure of proposed and reference techniques computed on the test dataset after AWGN addition, JPEG compression, resizing, and rotation.

Technique	Featuring	Matching	SATS	total
B-PM+RGB	0.05	69.80	29.48	99.33
B-PM+ZM	2.13	16.50	28.80	47.43
G-PM+RGB	0.05	230.24	76.32	306.61
M-PM+ZM	2.13	19.54	33.06	54.73

Table 1: CPU-time (seconds) for PatchMatch-based techniques.

Technique	mean	st. dev.
proposed	54.74	50.47
Zernike [9]	53.91	11.71
Bravo [4]	102.78	10.40
K-PCA [18]	936.09	13.25
SIFT [17]	3.71	2.78

Table 2: CPU-time (seconds) for proposed and reference techniques.

4.2. Comparison with the state-of-the-art

To conclude this analysis, we repeated all experiments for a number of reference techniques recently proposed in the literature, whose code is available online or can be reproduced easily and without uncertainties. The techniques are shortnamed here as Zernike [9], BRAVO [4], K-PCA [18], and SIFT [17]. Results, reported in Fig.3, show that the proposed technique outperforms almost uniformly all the other. In particular, except for large scale changes, it is much better than the keypoint-based SIFT technique which, on the other hand, is the only one that can exhibit a significantly smaller average CPU-time, less than 4s (Table 2). Moreover, it is consistently better, although only slightly, and has about the same CPU-time of Zernike, which uses a randomized kd-trees search [19]. All other techniques, besides being less effective, exhibit much larger CPU times, up to 900s.

5. CONCLUSIONS

We proposed a new copy-move detector based on PatchMatch, an iterative randomized algorithm for fast nearest-neighbor search. Thanks to the efficient NN search engine, we could consider dense descriptors, generally more reliable than keypoint-based descriptors, and still keep complexity under control. We also proposed a modification of PatchMatch which deals effectively with rotated copy-moves without increasing complexity. Performance is almost uniformly superior to that of state-of-the-art reference techniques.

Currently, we are experimenting with new features, invariant to both rotation and resizing. In both cases, the NN field is characterized by a local linear behavior, hence the first-order predictor used in the modified version of PatchMatch to propagate displacements is expected to be equally effective. Furthermore, we aim to replace SATS, relatively slow with regular fields, with faster techniques.

6. REFERENCES

[1] J.A. Redi, W. Taktak and J. Dugelay, "Digital image forensics: a booklet for beginners," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 133–162, jan. 2011.

[2] S. Bayram and H.T. Sencar and N. Memon, "A survey of copy-move forgery detection techniques," *proc. of IEEE Western New York Image Processing Workshop*, 2008.

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

[4] S. Bravo-Solorio and A.K. Nandi, "Automated detection and localisation of duplicated regions affected by reflection, rotation and scaling in image forensics," *Signal Processing*, vol. 91, pp. 1759–1770, 2011.

[5] S.-J. Ryu, M. Kirchner, M.-J. Lee and H.-K. Lee, "Rotation invariant localization of duplicated image regions based on Zernike moments," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1355–1370, aug. 2013.

[6] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *IEEE Trans. on Information Forensics and Security*, vol. 5, no. 4, pp. 857–867, dec. 2010.

[7] 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 Trans. on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, 2011.

[8] B.L. Shivakumar and S. Baboo, "Detection of region duplication forgery in digital images using SURF," *International Journal of Computer Science*, vol. 8, no. 4, pp. 199–205, 2011.

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

[10] A. Langille, and M. Gong, "An efficient match-based duplication detection algorithm," *Canadian Conf. on Computer and Robot Vision*, pp. 1–8, 2006.

[11] V. Christlein, C. Riess and E. Angelopoulou, "On rotation invariance in copy-move forgery detection," *IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2010.

[12] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.

[13] C. Barnes, E. Shechtman, D.B. Goldman and A. Finkelstein, "The Generalized PatchMatch Correspondence Algorithm," *European Conference on Computer Vision*, vol. 6313, pp. 29–43, 2010.

[14] http://ifc.recod.ic.unicamp.br/fc.website/index.py?sec=0.

[15] D. Cozzolino, D. Gragnaniello and L. Verdoliva, "Image forgery detection through residual-based local descriptors and block-matching," *IEEE International Conference on Image Processing (ICIP)*, 2014.

[16] D. Cozzolino, D. Gragnaniello and L. Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," *IEEE International Conference on Image Processing (ICIP)*, 2014.

[17] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, L. Del Tongo and G. Serra, "Copy-move forgery detection and localization by means of robust clustering with J-Linkage," *Signal Processing: Image Communication*, vol. 28, no. 6, pp. 659–669, 2013.

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

[19] M. Muja and D.G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *International Conference on Computer Vision Theory and Applications*, pp. 331–340, 2009.