CoMoFoD - New Database for Copy-Move Forgery Detection

Dijana Tralic, Ivan Zupancic, Sonja Grgic, Mislav Grgic
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, HR-10000 Zagreb, Croatia
E-mail: *dijana.tralic@fer.hr*

***Abstract* - Due to the availability of many sophisticated image processing tools, a digital image forgery is nowadays very often used. One of the common forgery method is a copy-move forgery, where part of an image is copied to another location in the same image with the aim of hiding or adding some image content. Numerous algorithms have been proposed for a copy-move forgery detection (CMFD), but there exist only few benchmarking databases for algorithms evaluation. We developed new database for a CMFD that consist of 260 forged image sets. Every image set includes forged image, two masks and original image. Images are grouped in 5 categories according to applied manipulation: translation, rotation, scaling, combination and distortion. Also, postprocessing methods, such as JPEG compression, blurring, noise adding, color reduction etc., are applied at all forged and original images. In this paper we present database organization and content, creation of forged images, postprocessing methods, and database testing. CoMoFoD database is available at http://www.vcl.fer.hr/comofod.**

***Keywords* - Digital Image Forensics; Copy-paste Forgery; Benchmark Database; Postprocessing Methods; CoMoFoD**

I. Introduction

The aim of a digital image forensics is to determine the authenticity and the origin of digital images. One of the most common used forgery methods is a copy-move forgery, where part of an image is copied to another location in the same image. A copied region can be transformed before translation by applying scaling, rotation, distortion or combination of those transformations. The purpose of this kind of forgery is usually to hide or to add some content or object in the image. Due to the fact that forged region come from the same image, it is impossible to use some statistical properties (such as camera noise, illumination conditions etc.) for a forgery detection because they are well matched. Taking the forged region from the same image also simplifies the forgery process because it is easier to fit the forged region into the image. Although a digital image forensics is relatively new field, many research has been done in a copy-move forgery detection (CMFD) recently [1].

The goal of this paper is to present a new database for a CMFD. Some databases for CMFD already exist, but they are not suited for evaluation of postprocessing methods. Ng and Chang [2] created a database of automatically sliced images by copying a part of an image and pasting it randomly in a different image. Due to the process of image creation and the fact that images are not postprocessed, most images are not semantically meaningful because copied regions have sharp edges. The CASIA database [3] consists of more realistic images with postprocessed boundaries of spliced regions. However, only postprocessing methods used in this database are JPEG compression and blurring. Battiato and Messina [4] made a database of tampered JPEG images. Dresden Image Database [5] was developed for purpose of camera identification, and Goljan et al. [6] created a database for the identification of sensor fingerprints. Databases MICC F220 and MICC F2000 [7] consist of 220 and 2000 images respectively. Those databases include only two types of tampering methods (rotation and scaling). Also, applying the postprocessing methods is difficult. Cristlein et al. [1] recently published a new database for a CMFD by selecting 48 source images and extracting 87 regions called snippets. To create tampered images, they developed a software framework that uses original images and snippets. Also, JPEG artifacts, noise, scaling and rotation of snippets can be automatically applied. Although all of those databases can be used for a CMFD, they address some disadvantages in the case of in-depth evaluation of CMFD methods, such as size of images or applied transformation/postprocessing methods.

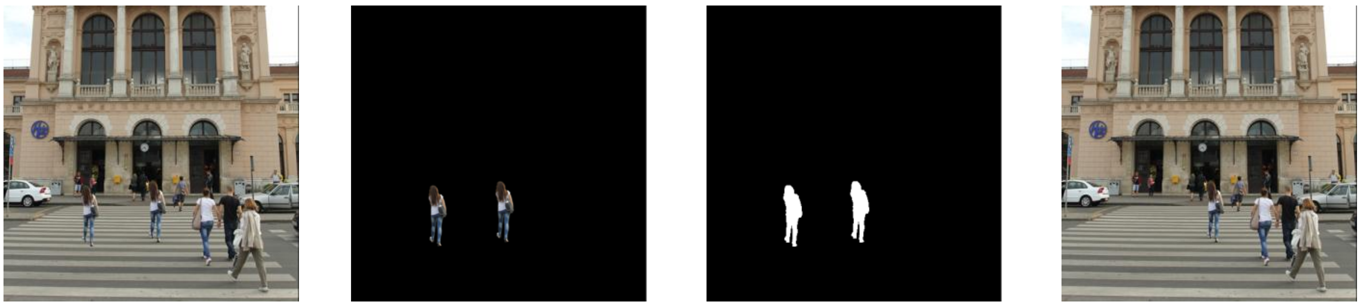

a) Forged image ("001_F.png") b) Mask ("001_M.png") c) Binary mask ("001_B.png") d) Original image ("001_O.png")

Figure 1. Example of image forgery and naming of first image in small image category

We developed a new database, called CoMoFoD [8], that consist of 260 tampered examples. For every tampered image, we stored original image, two types of masks that mark forgery (Fig. 1), and additional information such as size of tampered region. Images are categorized in 5 categories according to applied transformation. Also, six different postprocessing methods are applied to images in all categories.

This paper is organized as follows. In Section II we present database content starting with image acquisition process and database organization. Section II depicts forgery methods. Section III consists of description of all postprocessing methods applied in our database. In Section IV, database testing by using a simple CMFD algorithm is presented. Conclusion is depicted in Section V.

II. DATABASE DESCRIPTION

A. Image acquisition process

All images are recorded by Canon EOS 7D camera, and stored in CR2 (Canon RAW version 2) format as minimally processed data. Camera settings were equal for all images, and size of recorded images was 5202×3465 pixels. The acquisition was performed in different outside conditions (we captured nature, buildings, city views, etc.). We also included different backgrounds, such as grass, roof, wall, sky, etc.

B. Forgery method

Forged images are made using Photoshop CS3 and CS5. Images are first converted in PNG (Portable Network Graphics) format. Regions of 512×512 pixels and 3000×2000 pixels were cropped or resized from captured images to make original images.

Images are forged by copying a part of an original image and pasting it on a different location in the same image. The main goal was to embed the copied region into the original image content without leaving any visible traces of tampering. In some cases, copied part was transformed before changing its location. We applied several types of transformations, and grouped images in 5 categories according to applied transformation:

1. translation - a copied region is only translated to a new location without performing any transformation;
2. rotation - a copied region is rotated and translated to a new location;
3. scaling - a copied region is scaled and translated to a new location;
4. distortion - a copied region is distorted and translated to a new location;
5. combination - two or more transformation are applied on a copied region before moving it to a new location.

Size of copied region differ from image to image. Smallest copied part in 512×512 images is ~0.14 % of image size, and ~0.11% of image size in 3000×2000 images. Biggest copied region in 512×512 images is ~14.32 % of image size, and ~17.34% of image size in 3000×2000 images. Size of copied area per image categories can be seen in Table I. Some images are multiple forged images that contain more than one tampered region which are located without any mutual overlapping. Multiple forged images can be made by copying the same region on different locations or by copying different regions on different locations (Table I).

C. Database content and organization

Database is organized in 2 main categories according to the image size: small (512×512) and large (3000×2000) image category. In every category, images are grouped by forgery methods described in Section II.B. There are 40 examples of all types of forgery in small image category, giving total of 200 forgery examples. In large image category, there are 10 examples for all types of forgery beside distortion which has 20 examples of forgery, giving total of 60 examples of forgery in large image category.

Every example consists of an original image (without any forgery), a forged image (with some forgery) and two masks (colored and black/white), as can be seen in Fig. 1. Masks indicate the forgery by showing the non-tampered region as black background. Copied and pasted regions (tampered part of image) are presented as colored regions in the first mask, and as white regions in the second mask. Those masks are made for CMFD evaluation. Additionally, a text file called "readme.txt" is stored in all categories. The file contains information about tampered region, such as size of region before and after transformation, and type of transformation.

TABLE I. IMAGE FEATURES IN COMOFOD DATABASE

Category	**Image category**	**Total number of images per category**	**Size of smallest copied area [pixels]**	**Size of biggest copied area [pixels]**	**Number of images with more than one copied area**	**Number of images with more than one different copied area**
Translation	512×512	40	360	28405	12	7
	3000×2000	10	7180	1130658	4	1
Rotation	512×512	40	403	37542	6	1
	3000×2000	10	11851	1040232	3	1
Scaling	512×512	40	403	37542	7	3
	3000×2000	10	11851	549188	4	0
Distortion	512×512	40	1037	37542	2	0
	3000×2000	20	24057	238083	4	1
Combination	512×512	40	403	37542	5	3
	3000×2000	10	6519	2611416	4	1

Every category has also 6 subcategories that consist of the postprocessed images, as described in Section III. Postprocessing was done on all original and forged images in both categories, giving a total of 10,400 images in small, and 3,120 images in large image category.

D. Image naming

In both image categories, images are named as follows: $N_1\_M_1$, where N_1 is three (small image category) or two (large image category) digits which mark the image number in category, and M_1 is one of four different marks:

1. "F" for forged image;
2. "B" for binary mask (black and white mask);
3. "M" for mask (colored mask);
4. "O" for original image.

Example of image naming can be seen in Fig. 1 for the first image in small image category. Additional information about transformations applied on forged images, such as rotation angle, scaling value, etc., can be seen only in "readme.txt" files in corresponding image category. However, images with the same transformation are grouped together by reserving consecutive numbers for them. For example, first 40 images in small image category are examples of image forgery by only translating forged region, next 40 images are made by rotating forged region, etc.

Postprocessed images are named as follows: $N_1\_M_1\_M_2$, where additional mark M_2 points one of six postprocessing methods applied on image:

1. "JC" for JPEG compression;
2. "NA" for noise adding;
3. "IB" for image blurring;
4. "BC" for brightness change;
5. "CR" for color reduction;
6. "CA" for contrast adjustments.

Postprocessing was done only for original and forged images, so mark M_1 can be only "O" or "F". Usage of different parameters in postprocessing methods was marked with numbers at the end of image in such a way that lower number correspond to lower used parameter value. For example, first original and forged images in small image category that are saved with JPEG quality of 20 have the following names: "001_O_JC1.jpg" and "001_F_JC1.jpg". Examples of image naming for postprocessed images can be seen in Fig. 2.

III. POSTPROCESSING METHODS

There are many different types of postprocessing methods that can be applied to forged images with the aim of hiding tampering traces. Most common postprocessing methods are JPEG compression, noise adding and image blurring, but there are many others such as brightness change, color reduction and contrast adjustments.

TABLE II. PARAMETERS OF POSTPROCESSING METHODS

Method	Parameters
JPEG compression	factor = [20, 30, 40, 50, 60, 70, 80, 90, 100]
Noise adding	$\mu = 0$, $\sigma^2 = [0.009, 0.005, 0.0005]$
Image blurring	averaging filter = [3×3, 5×5, 7×7]
Brightness change	(lower bound, upper bound) = [(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]
Color reduction	intensity levels per each color channel = [32, 64, 128]
Contrast adjustments	(lower bound, upper bound) = [(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]

In this paper we present 6 different types of postprocessing methods that were applied on images in CoMoFoD database. Information and parameters for all postprocessing methods are given in Table II. It is important to mention that the same parameter set was applied for postprocessing of images from all categories. Most of the methods were implemented in MATLAB®, beside JPEG compression which was applied by PIXresizer software [9]. Examples of postprocessed forged images can be seen in Fig. 2.

A. JPEG compression

Both forged and original images were saved as JPEG images with different quality factors. JPEG quality factors were ranged from 20 to 100, with the step of 10. During image saving, resolution has not been changed. For JPEG pictures whose JPEG quality factor was below 70, image visual degradation and blocking artifact were noticeably visible. Examples of forged images postprocessed with JPEG compression with quality factors of 20 and 100 are shown in Fig. 2.a.

B. Noise adding

Gaussian white noise with zero mean and different values of variance was added to images. Parameters that were used are listed in Table II, and PSNR for every postprocessed image was calculated and stored in 'PSNR.txt' file. Image with variance of 0.009 have quite visible noise artifacts, and average PSNR $\approx$ 20.6. Level of noise was reduced by reducing the value of variance to 0.005 (average PSNR $\approx$ 23.2), and 0.0005 (average PSNR $\approx$ 33.1). Two images with added noise with variance equal to 0.009 and 0.0005 are shown in Fig. 2.b.

C. Image blurring

Image blurring was obtained by convolving the image with 3×3, 5×5 or 7×7 averaging filters. Blurred images are the same dimensions as original images due to selected option that input image values outside the bounds of the image are assumed to be equal to the nearest image border value. Blurred images (obtained by any averaging filters) are noticeably visually blurred comparing to the original images, especially images obtained by filtering with 7×7 averaging mask. Examples of forged blurred images with 3×3 and 7×7 averaging filters are shown in Fig. 2.c.

D. Brightness change

Changing the brightness of the image was obtained by mapping the intensity values of the original image that were between lower and upper bound (Table II) to interval [0, 1]. Intensity values below lower bound and above upper bound were saturated to minimal and maximal value, respectively. Using brightness change in the range (0.01, 0.95) had visually almost imperceptible impact on appearance of postprocessed image, while usage of brightness change in the range (0.01, 0.8) resulted in visually significantly brighter images. Images obtained by using brightness change in the range (0.01, 0.95) and (0.01, 0.8) are shown in Fig. 2.d.

E. Color reduction

Color reduction of the original image was obtained by uniform quantization of the original image intensity values. For each color channel of the original image the number of different intensity levels was reduced from 256 to 32, 64 or 128 levels. Images obtained by reducing the number of intensity levels have visually almost imperceptible degradations comparing to the original image with 256 intensity levels per channel. Two examples of images obtained by color reduction to 32 and 128 levels are shown in Fig. 2.e.

F. Contrast adjustments

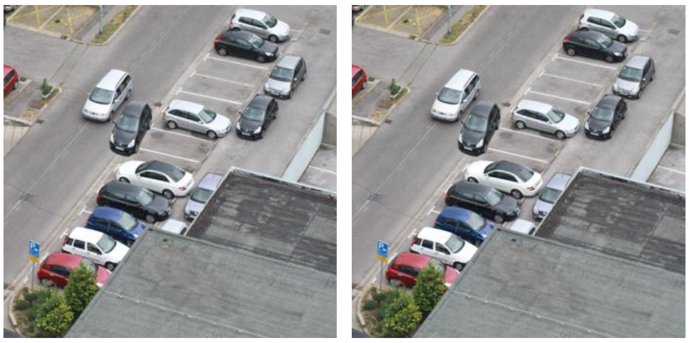
Image contrast was modified by mapping the whole range of input image intensity values to a new interval bounded with lower and upper bound (Table II). Hence, the output image intensity range has been reduced. Contrast adjustments in the range (0.01, 0.95) had visually almost imperceptible impact on postprocessed image, while images obtained by usage of contrast adjustments in the range (0.01, 0.8) resulted in visually significantly darker images. Examples of postprocessed images with contrast adjustments in the range (0.01, 0.95) and (0.01, 0.8) are shown in Fig. 2.f.

a) JPEG compression: left - quality factor: 20, name: "015_F_JC1.jpg", right - quality factor:80, name: "015_F_JC7.jpg"

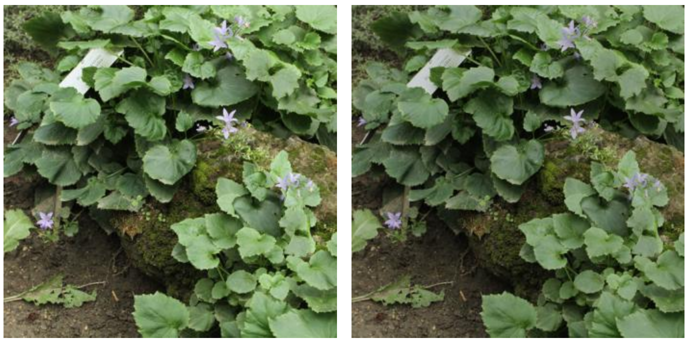
b) Noise adding, left- variance: 0.009, name: "006_F_NA1.png", right - variance: 0.005, name: "006_F_NA2.png"

c) Image blurring: left - averaging filter: 5×5, name: "120_F_IB2.png", right- averaging filter: 7×7, name "120_F_IB3.png"

d) Brightness change: left - range: (0.01, 0.9), name: "054_F_BC2.png", right - range: (0.01, 0.8), name: "054_F_BC3.png"



e) Color reduction, left - color levels: 32, name: "128_F_CR1.png", right - color levels: 128, name: "128_F_CR3.png"



f) Contrast adjustments: left - range: (0.01, 0.95), name: "174_F_CA1.png", right - range: (0.01, 0.8), name: "174_F_CA3.png"

Figure 2. Examples of postprocessed forged images and nameing format

IV. Database testing

For demonstration purpose, CMFD was done by using method developed by Fridrich et al. [10]. They proposed blocks-based method that uses 256 coefficients of a discrete cosine transform (DCT) as feature set. Method was implemented through following steps:

1. image preprocessing - image transformation into grayscale space;
2. blocking - image division into overlapping blocks of size *b*x*b* pixels; for image of size *N*x*M*, total $(N\text{-}b+1)\times(M\text{-}b+1)$ blocks were defined;
3. feature extraction - calculation of feature vectors of every block by DCT transformation;
4. sorting - grouping feature vectors with similar values by using lexicographical sorting;
5. matches detection - comparison of feature vectors values by calculating the Euclidean distance of every vector and its neighbor vectors; detection of matched blocks by selecting pairs of blocks where Euclidean distance was smaller than similarity threshold T_s;
6. postprocessing - removing false detected that were too close in images (neighbor blocks with distance smaller than the distance threshold T_d) and regions smaller than the predefined region threshold T_r.

A. Parameters selection and error measurements

Since selection of parameters depends on image size, testing was performed on 512×512 images (small image category). According to that, we defined block size of 8×8 pixels, which gives total of 255,025 blocks. Thresholds values were experimentally determined as follows: T_r=128. Values of other thresholds depend on transformation and postprocessing methods as can be seen in Table III.

For presentation purposes, testing was performed only on forged images in small category, and following postprocessing scenarios: images without any postprocessing, JPEG compression with the quality factor of 40, noise adding with variance of 0.005, image blurring with 5×5 averaging filter, brightness change in the range (0.01, 0.8), color reduction to 32 intensity levels, and contrast adjustments in the range (0.01, 0.8).

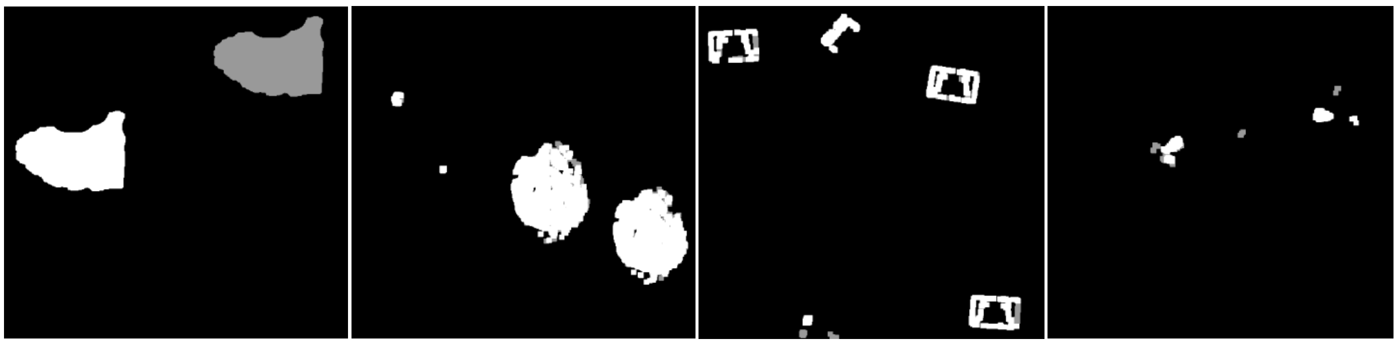
To evaluate the performance, the percentage of false positive pixels F_P, the false negative rate F_N, and the true positive rate T_P are used to calculate following error measurements:

- Precision *P* or exactness of method denotes the probability that a detected region is truly a forgery;
- Recall *R* or completeness of the method is the probability that a forged region is detected;
- F1-measure combines precision and recall, and it is a test's accuracy that reaches best performance at 1 and worst at 0.

B. Testing results

Testing results showed that proposed method can be used for detection of copy-move forgeries in digital images. Examples of detection results for different postprocessing methods are presented in Fig. 3. Detection is more reliable in the case when only translation of the forged region is used (Fig. 3.a) than in the case when other transformations are applied (Fig. 3.b-3.d). Reason for that is the fact that DCT transformation, used for matches detection, is not robust on rotation, scaling or other distortions. Detection of those transformations is possible only for small rotation angles or scaling/distortion values (Fig. 3.b).

Testing results with more details can be found in Table III. Results for forgery detection on images without any postprocessing show highest f1-measure in case when only translation is applied, while in other cases depend on amount of transformation. For most postprocessing methods. beside JPEG compression and noise adding, detection of translation is very accurate. Detection in case of using JPEG compression, and noise adding is possible only for some images. Also, number of images with f1-measure higher of 0.5 (we marks it as correctly detected images) is under 5% for most cases beside translation. Detection depend on selected thresholds which have been experimentally defined for every set of images (Table III).



a) No postprocessing (translation), f1-measure = 0.9969 (forged image - Fig. 2.a)

b) Noise adding (scaling), f1-measure = 0.7428 (forged image - Fig. 2.b)

c) Image bluring (rotation), f1-measure = 0.7154 (forged image - Fig. 2.c)

d) Brightnes change (combination), f1-measure = 0.0815 (forged image - Fig. 2.d)

Figure 3. Examples of detection results for different types of postprocessing methods

TABLE III. TESTING RESULTS FOR FORGED IMAGES IN SMALL IMAGE CATEGORY (1400 IMAGES)

Postprocessing method	**Transformation (number of images)**	T_s	T_d	**Images with f1-measure > 0.5**	**Average precision, *P***	**Average recall, *R***	**Average F1-measure**
No postprocessing	***Translation (40)***	4	130	40	0.9978	0.9484	0.9725
	Rotation (40)	50	50	3	0.9589	0.4858	0.6449
	Scaling (40)	20	50	2	0.9968	0.8594	0.9230
	Distortion (40)	20	60	5	0.8025	0.4319	0.5616
	Combination (40)	30	50	3	0.8315	0.4008	0.5411
JPEG compression, quality factor = 40	***Translation (40)***	3	130	1	0.6878	0.5578	0.6160
	Rotation (40)	40	60	3	0.9367	0.4549	0.6123
	Scaling (40)	40	50	1	0.8367	0.5078	0.6320
	Distortion (40)	any	any	0	/	/	/
	Combination (40)	any	any	0	/	/	/
Noise adding, variance = 0.005	***Translation (40)***	40	130	1	0.8859	0.4325	0.5812
	Rotation (40)	30	130	1	0.8924	0.5641	0.6913
	Scaling (40)	30	130	1	0.9439	0.6124	0.7428
	Distortion (40)	any	any	0	/	/	/
	Combination (40)	any	any	0	/	/	/
Image blurring, 5×5 averaging filter	***Translation (40)***	5	130	40	0.9959	0.8637	0.9251
	Rotation (40)	50	90	3	0.9610	0.5340	0.6865
	Scaling (40)	50	90	1	0.6346	0.5068	0.5635
	Distortion (40)	20	50	3	0.9858	0.5064	0.6691
	Combination (40)	any	any	0	/	/	/
Brightness change, range (0.01, 0.8)	***Translation (40)***	4	130	40	0.9701	0.9418	0.955
	Rotation (40)	60	90	1	1	0.6619	0.7966
	Scaling (40)	60	80	1	0.9842	0.7256	0.87468
	Distortion (40)	any	any	0	/	/	/
	Combination (40)	any	any	0	/	/	/
Color reduction, 32 intensity levels	***Translation (40)***	3	130	40	0.9934	0.9490	0.9707
	Rotation (40)	20	50	2	0.9986	0.6544	0.7907
	Scaling (40)	20	50	1	0.9378	0.8520	0.8928
	Distortion (40)	50	50	1	1	0.4098	0.5814
	Combination (40)	any	any	0	/	/	/
Contrast adjustments, range (0.01, 0.8)	***Translation (40)***	2	130	40	0.9979	0.9486	0.9726
	Rotation (40)	50	60	3	0.8916	0.4715	0.6168
	Scaling (40)	50	50	1	0.8186	0.8643	0.8408
	Distortion (40)	50	50	1	1	0.4477	0.6185
	Combination (40)	any	any	0	/	/	/

V. CONCLUSION

A copy-move forgery is common way of image tampering thanks to the simplicity of its conduction, so accurate and robust detection is the aim of every CMFD algorithm. To allow testing of algorithms in various conditions, it is important to develop different test cases. Database CoMoFoD consists of five different types of tampered images according to the transformation of a copied region. Every forged image is accompanied with an original image and two masks that indicate the forgery. Since the main purpose of the database is to allow testing of the postprocessing impact on a detection, six different postprocessing methods were applied on all forged and original images, giving total of 13,520 images. By testing the new database on a simple CMFD algorithm, we showed that database CoMoFoD is suitable for in-depth evaluation of CMFD algorithms and postprocessing methods. Future work is oriented to evaluation of postporcessing impact on detection for different CMFD algorithms.

ACKNOWLEDGMENT

The work described in this paper was conducted under the research projects 036-0361630-1635 and 036-0361630-1643 supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

REFERENCES

[1] 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, December 2012

[2] T. Ng and S. Chang, “A Data Set of Authentic and Spliced Image Blocks,” Columbia University, Tech. Rep. 203-2004-3, January 2004

[3] CASIA Image Tampering Detection Evaluation Database, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, available at: forensics.idealtest.org

[4] S. Battiato and G. Messina, “Digital Forgery Estimation into DCT Domain - A Critical Analysis,” in ACM Multimedia Workshop Multimedia in Forensics, pp. 37-42, October 2009

[5] T. Gloe and R. Böhme, “The ‘Dresden Image Database’ for benchmarking digital image forensics,” in 25th Symposium on Applied Computing, vol. 2, pp. 1585-1591, March 2010

[6] M. Goljan, J. Fridrich, and T. Filler, “Large Scale Test of Sensor Fingerprint Camera Identification,” in SPIE Media Forensics and Security, vol. 7254, pp. 0I 01-12, January 2009

[7] I. Amerini, L. Ballan, R. Caldelli, A. D. 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, September 2011

[8] CoMoFoD database, available at: http://www.vcl.fer.hr/comofod

[9] PIXresizer software, available on: pixresizer.en.softonic.com

[10] J. Fridrich, D. Soukal, and J. Lukas, “Detection of copy-move forgery in digital images,” in Proceedings of Digital Forensic Research Workshop, pp. 55-61, August 2003