UNIVERSITÀ DEGLI STUDI DI TRENTO

DIPARTIMENTO DI INGEGNERIA e SCIENZE DELL'INFORMAZIONE
Master in Artificial Intelligence Systems

An image-statistics-based approach to detect recaptured images from screens
Project report

Signal, Image and Video
Academic Year 2021/2022

Corso Laura, Miotto Sara, Zaghen Olga

Contents

Overview of the adopted approach

1.1 Introduction

In our project we followed the approach proposed by Kai Wang in the article *A Simple and Effective Image-Statistics-Based Approach to Detecting Recaptured Images from LCD Screens* published by *Digital Investigation* number 23 of 2017 [1].

The aim of the paper is to detect whether an image is an original one or if it is a recapture of an already existing image showed on a LCD screen. In fact, one particular problem addressed by image forensics is that nowadays, with the increasing popularity and quality of digital cameras, everyone can take a picture of an already existing one and use this recaptured image for bad and illegal purposes. In order to deal with such problem, this paper describes a mathematical tool that can be used to distinguish between *single captured* images (*i.e.*, real-world scenes captured by a digital camera) and *recaptured* images from LCD screens (*i.e.*, single captured images displayed on a LCD screen and then recaptured by a digital camera).

Lots of image recapture detection methods have been proposed in literature and most of them use strong assumptions about the recapture process. In fact, the recapturing introduces aliasing-like distortions, changes in image's color and edge sharpness and so on. In the followed paper [1], instead, very loose assumptions about the consequence of recapturing images are made and the proposed method relies on a simple image statistics feature: the pixel-wise correlation coefficients (CC) in image differential domains.

The proposed method is validated by the author on two large data-sets comprising high-resolution and high-quality recaptured images: ROSE data-set [3], which was constructed by Cao and colleagues at the Nanyang Technological University in 2010, and ICL data-set [4], created by Thongkamwitoon et al. at the Imperial College London in 2015.

1.2 Proposed Method

As stated before, differently from other approaches, the method proposed by Kai Wang for recapture forensics uses image statistics derived directly in the pixel domain. Indeed, it extracts a feature vector consisting of simple pixel-wise correlation coefficients in image differential domains which conveys discriminative information that can be used by a SVM classifier for the task of recaptured image detection.

The whole procedure designed by the author is the following:

1. **Pre-processing**

   Each given image X is first converted into grey-scale and then it is rescaled while keeping the ratio between height and width unchanged so that the resized version has a width of 2048 pixels.

2. **Computation of two residue images**

   In order to extract the feature vector for a given image X of size $M \times N$, initially, two residue images $\mathbf{R}^{(\mathrm{i})}, i \in \{1, 2\}$, are calculated as follows:

$$\mathbf{R}^{(\mathrm{i})} = trim(X - X * f_{\mathrm{i}})$$

where $*$ means mathematical convolution and the function *trim* removes the first row and column, as well as the last row and column, from the input image X.

Therefore, to the input image X is first applied a low-pass filtering using two simple filters with the following kernels:

$$f_1 = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix}, f_2 = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{bmatrix}.$$

As a result, for a given pixel x_{j} in X, the corresponding pixel value in the residue image $\mathbf{R}^{(1)}$ (respectively $\mathbf{R}^{(2)}$) measures the difference between the value of x_{j} and the average value of its two horizontal (respectively vertical) neighboring pixels in X.

Secondly, a trimming is applied so that the two residue images are a trimmed version of the input image X. This pruning is done in order to apply the filters only to pixels that have neighbours included by the kernel that are within the image and not outside of it.

After these two operations it can be easily observed that the residue images describe the edges and the noises in the image, two characteristics strictly correlated to the blurriness and aliasing-like alterations introduced by the process of recapturing. For this reason, from the residue images it would be possible to extract a meaningful statistical feature that can expose the differences between single captured and recaptured images.

3. **Pixel-wise correlation coefficient**

   From the two residue images $\mathbf{R}^{(\mathrm{i})}, i \in \{1, 2\}$, the statistics feature is computed. In particular, the author has chosen to compute the feature vector as a vector of correlation coefficients (CC) between the residues.

   More precisely, from a residue image $\mathbf{R}^{(\mathrm{i})}$, first of all K overlapping patches of 5×5 pixels defined as $\mathbf{P}^{(1)}$, $\mathbf{P}^{(2)}$, ..., $\mathbf{P}^{(\mathrm{K})}$ are extracted. Each pixel in the patch $\mathbf{P}^{(\mathrm{k})}$ is denoted by ${p^{(\mathrm{k})}}_{\mathrm{i,\ j}}$ with $i, j \in \{1, 2, 3, 4, 5\}$ and $k \in \{1, 2, ..., K\}$.

   Then, 25 vectors of length K of the form $\mathbf{v}_{\mathrm{i,j}} = [{p^{(1)}}_{\mathrm{i,\ j}}, {p^{(2)}}_{\mathrm{i,\ j}}, ..., {p^{(\mathrm{K})}}_{\mathrm{i,\ j}}]$ are created concatenating the residue value with the same index i, j from each patch.

   Afterwards, the correlation coefficients $c_{\mathrm{i,j}}$ between the vector corresponding to the central pixel, *i.e.* $\mathbf{v}_{3,3}$, and all the 25 vectors is computed using the following formula:

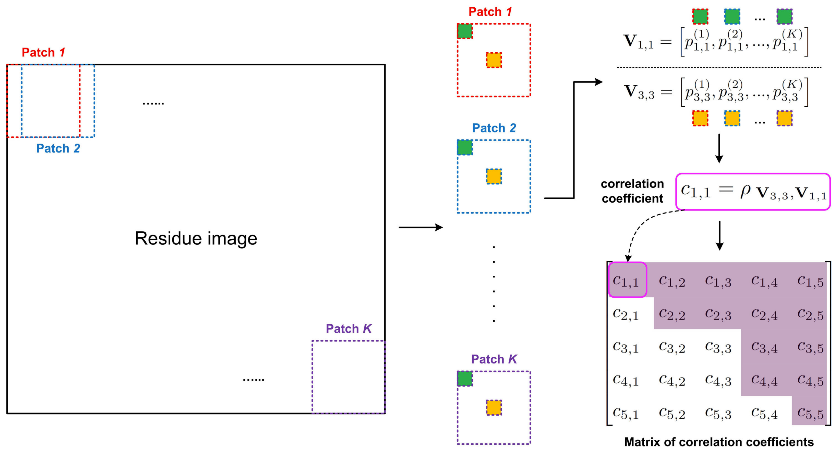
$$c_{\mathrm{i,j}} = \frac{\sum_{k=1}^{K} ({p^{(\mathrm{k})}}_{3,\ 3} - \overline{p}_{3,\ 3})({p^{(\mathrm{k})}}_{\mathrm{i,\ j}} - \overline{p}_{\mathrm{i,\ j}})}{\sqrt{\sum_{k=1}^{K} ({p^{(\mathrm{k})}}_{3,\ 3} - \overline{p}_{3,\ 3})^2} \sqrt{\sum_{k=1}^{K} ({p^{(\mathrm{k})}}_{\mathrm{i,\ j}} - \overline{p}_{\mathrm{i,\ j}})^2}}$$

   where $\overline{p}_{\mathrm{i,\ j}}$ is the mean of the elements in $\mathbf{v}_{\mathrm{i,j}}$. As a result, for each residue image $\mathbf{R}^{(\mathrm{i})}$ 25 correlation coefficient values are computed. These values can be grouped in a 5×5 matrix.

   These CC values measure how the horizontal or vertical residues, at different locations within a small local neighborhood of 5×5 pixels, are related to each other through second-order mixed statistical moments. The extracted CC values are sensitive to the alterations due to image recapture and exhibit different characteristics for single captured and recaptured images. In particular, the mean CC values between the central pixel and most neighbors in the local 5×5 patch are higher for recaptured images due to the introduction of blurriness and the mean CC values between the central pixel and distant neighbors in recaptured images are lower due to the aliasing-like distortion.

4. **Feature vector**

   The extracted matrix of CC values is redundant. For this reason, in order to avoid redundancy, only the upper triangle of the matrix is retained and the central element of the matrix is removed. As a consequence, a 14-dimensional feature vector for each residue image $\mathbf{R}^{(\mathrm{i})}$ is obtained. Then, the two feature vectors are concatenated to form the final feature vector of 28 correlation coefficients.



Patch 1
Patch 2
Residue image
Patch K
Patch 1
Patch 2
Patch K
$\mathbf{V}_{1,1} = \left[p_{1,1}^{(1)}, p_{1,1}^{(2)}, ..., p_{1,1}^{(K)}\right]$
$\mathbf{V}_{3,3} = \left[p_{3,3}^{(1)}, p_{3,3}^{(2)}, ..., p_{3,3}^{(K)}\right]$
correlation coefficient
$c_{1,1} = \rho_{\mathbf{V}_{3,3},\mathbf{V}_{1,1}}$
Matrix of correlation coefficients

Figure 1.1: Illustration of the patch extraction and the computation of pixel-wise correlation coefficients, taking $c_{1,1}$ as example. The shaded elements in the matrix of correlation coefficients are retained elements comprised in the proposed feature vector.

5. **Classification**

The discrimination between single captured and recaptured images is performed by a SVM classifier. In particular, in the off-line phase the aforementioned image statistics feature is extracted from a group of single captured and recaptured images. After that, the extracted feature vectors, along with the associated image labels, are fed into a Support Vector Machine (SVM) for training a recapture forensic detector.

Subsequently, during the on-line phase, for each new image a feature vector is extracted from it and given as input to the trained SVM classifier in order to get the forensic result of whether the given image is a single captured or a recaptured one.

In the SVM classifier a RBF (Radial Basis Function) kernel is used and, in particular, the values of the hyperparameters of the classifier are determined with a 5-fold cross validation on the training set.

In the testing phase, the forensic performance is evaluated using the classification accuracy of single captured images, recaptured images and all the images in the testing set. In order to enhance the statistical significance of the results, for each test of the proposed method 50 runs are performed, and for each run the partition between training and testing sets is re-computed randomly always keeping the ratio between the number of training and testing images unchanged (15:100). At the end, the classification accuracies are reported as the mean of the values obtained from the 50 runs.

Implementation choices

2.1 The Python libraries used for each phase

In order to perform some complex specific operations required by the algorithm we followed, we imported some useful modules from already existing Python libraries. For what concerns the handling of the images at the initial stages, the libraries that suited our needs the most were `PIL` and `cv2`. After converting the images into matrices, we performed the processing through the `numpy` and `scipy` libraries. For what concerns the machine learning phase, that is the training of the SVM, the classification and the computation of the accuracies, we chose to utilize the methods already implemented in the `sklearn` libraries.

2.2 Handling of the ICL data-set

We tested our software on the ICL data-set[4], which is composed of 900 single captured and 1440 recaptured images. In order to make the information contained in the data-set exploitable by our code, we created a folder, named `AllImages`, containing the whole set of original captured and recaptured images, and a `.txt` file, `labels_final.txt`. Each line of the file refers to a particular image in `AllImages` and reports its name and its label (`Original Captured` or `Recaptured`), with a tab space between them.

These are the only two elements that our program needs in order to function properly and we chose this input format because it seemed to us the clearest and most general way to set out the necessary information contained in the data-set. The software exploits these inputs in order to define the vector `Y` and the matrix `X`. `Y` is the vector of labels and it is created first; it only contains `0` and `1` values (where `0 = Original Captured`, `1 = Recaptured`), each one corresponding to a particular image in the same order followed in `labels_final.txt`. `Y` is defined by creating a `.csv` file with two columns out of `labels_final.txt` (one for the images' names and one for the labels, or types), taking into account the `"Type"` column and converting the values to `0` and `1`.

For what concerns `X`, instead, it is a matrix of dimension $[n\_images \times 28]$, where n_images stands for the total number of images (2340) and 28 is the size of the feature vectors corresponding to each image. Indeed, each row of `X` represents a feature vector of an image in `AllImages`, and these vectors are computed through a *for* cycle in which all the images contained in such folder, in the same order in which their name appears in `labels_final.txt`, are processed (see section 2.3). At the end of the *for* cycle the first two rows of `X` are removed, because they contain only 0 values and were added just for implementation purposes. After this operation the definitive configuration of the matrix is achieved: row i contains the feature vector corresponding to the image whose label is the i-th element of `Y`.

2.3 Image processing: creation of the feature vectors

Following the procedure illustrated in the paper [1] we converted each image contained in `AllImages` to grey-scale and then we resized the image so that the resulting width was 2048 pixels and the resulting height being compliant with the original ratio between the two dimensions.

Afterwards, we created the two low-pass filters and we used them to compute the result of the convolution with the input image.

Consequently, in order to obtain the two residue images, as described by the paper [1], we applied a trim function to each residue image resulting from the pixel-pixel difference between the original image and each of the two filtered ones. In order to evaluate the differences between the original image and each residue image, a commented section was left in the code that can be used to visualize and compare the three images, through which we obtained the following results:

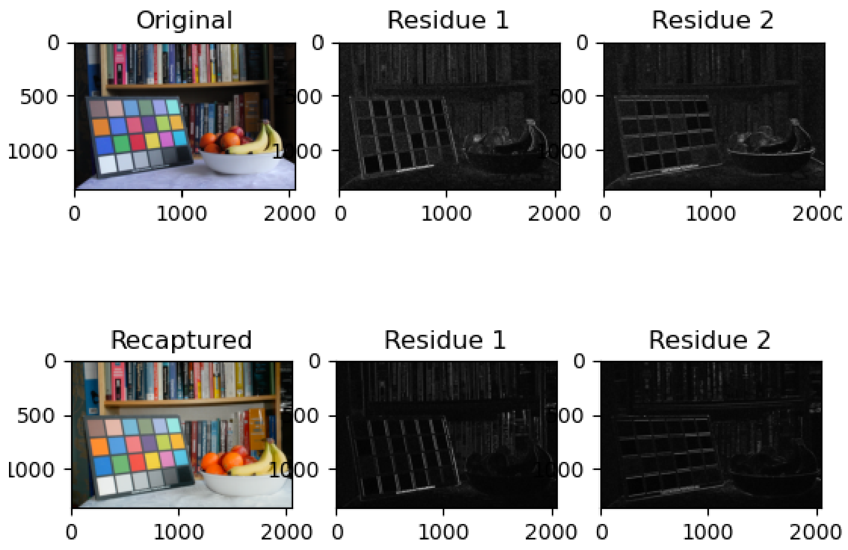


Figure 2.1: In the top row we show a single captured image and its residue images, while in the second row the residues are computed for the corresponding recaptured image. The residues in the two cases have very different characteristics due to alterations introduced by image recapture. For a better visualization, we have taken pixel-wise absolute values of the residue images and performed thresholding at 15 (the range of pixel values in unfiltered images is from 0 to 255).

After this pre-processing phase we proceeded with the extraction of the correlation coefficients from the two residue images. In particular, the 25 vectors $\mathbf{v}_{\mathrm{i,j}}$ containing the concatenated residue values with the same index i, j from each 5×5 pixels patch were determined selecting a sub-matrix of the original image corresponding to the area covered by the sliding of the patch along each row and each column of the image. In addition, each correlation coefficient $c_{\mathrm{i,j}}$ is computed using the `scipy` command `pearsonr`. In fact, the direct implementation of the suggested formula on the paper [1] (which we initially tried and which is still present as a commented section in the code) takes too long to be computed. Thus, for efficiency reasons, we decided to use the already implemented calculation available in the `scipy` library of Python.

At the end, we concatenated the two feature vectors extracted from the two residue images in a final feature vector.

2.4 Training of the SVM, classification and accuracy evaluation

In order to evaluate the performance of the classifier, we computed the classification accuracy (that is the percentage of correctly classified images) of single captured images, recaptured images and all the images in the folder (the so-called overall accuracy). These values were obtained as the mean of the results of 50 runs of the 5-fold cross validation; the reason for this high number of runs is to enhance the statistical significance of the results. In each run `train_test_split` performs a different and random (thanks to the `random_state` value that always changes) partition between training set and testing set, maintaining the ratio of their cardinality unchanged. Indeed, as expressed in the reference paper, the ratio between the training and testing sets is 15:100, that is $train\_size/allimages\_size = \frac{15}{115} = 0.13$.

After both `X` and `Y` are partitioned, the training of the classifier is performed. We followed the guidelines of our reference paper [1], which suggests to choose a RBF kernel for the Support Vector Machine and to determine the values of the hyperparameters γ and C using a 5-fold cross validation on the training set. In order to satisfy all the requirements, we defined the classifier in the following way:

```
classifier = GridSearchCV(SVC(), tuned_parameters, scoring="accuracy")
```

`GridSearchCV` stands for grid search cross validation: it performs an exhaustive search over specified parameter values of an estimator, which are optimized by cross-validated grid-search over a parameter grid. In this case the estimator is a Support Vector Classifier (`SVC`), the optimization aims to maximize the classification accuracy (`scoring="accuracy"`) and the parameter grid over which the grid search is performed at each CV step is defined in `tuned_parameters`. It is essential to point out that `GridSearchCV` by default performs a 5-fold cross validation: this is why the number 5 was not specified as an input parameter.

The reference paper [1] only referred to the 5-fold cross validation approach for determining the values of the hyperparameters, making no reference on the sets of possible values to test for choosing them and on the method to follow for the choice. As stated in an article that analyzes the properties of a well-performing SVM, that is *A Practical Guide to Support Vector Classification* [2], the most effective way to determine the hyperparameter values for a Support Vector Classifier with a RBF kernel is through a grid search using cross validation, in which various pairs of (C, γ) values are tried and the one that grants the best accuracy is picked. The same article also suggests a range of values on which the grid search should be performed, that is between 2^{-5} and 2^{15} for C and between 2^{-15} and 2^{3} for γ. This justifies our choice for the definition of the parameter grids (`tuned_parameters`), in which we considered 100 values among the minimal and maximal ones specified in the article. Taking into account more than 100 values would have been too expensive from a computational point of view.

After its definition, the classifier is trained (and the hyperparameters determined) with the `fit` command, and after that the testing phase is performed through `predict`, that returns the predicted labels corresponding to the feature vectors in `X_test`.

The last step performed in each of the 50 runs is the computation of the classification accuracies related to the original captured images, the recaptured images and all images (the overall accuracy). The values obtained for the specific run are then summed up with the ones resulting from the previous runs, and the total values obtained at the end of the cycle are used to compute the mean values for each of the three accuracies, which are useful in order to evaluate the performance of the classifier. The overall accuracy is easily obtained with the ad-hoc command `accuracy_score`. For what concerns the accuracies related to the original captured and recaptured images separately, instead, we can also interpret these values as the recall and specificity of the classifier. The method `recall_score` with input parameter `average = None` returns the desired recall scores for each class, in the order specified by the `labels` parameter, which we set equal to `[0, 1]`: the first value returned is the one corresponding to the original captured images while the second one refers to the recaptured images.

2.5 Further discussion on technical choices

- **Correlation coefficients**

  Following the guidelines of the author of the paper [1], we computed the correlation coefficient instead of, for example, the straightforward correlation. The reason is that this coefficient is independent of the proper content of the image, while the other depends on the overall brightness. Furthermore, the computation of the correlation coefficient incorporates a divisive normalization procedure, which makes the CC value independent of the amplitude of the input vectors.

- **Rescaling width**

  Regarding the width used for the rescaling phase, the paper [1] reports the results of applying the described method to different image widths. We chose to implement the 2048 pixels case, as it seemed the most significant one: the author highlights that this size is the best in order to compare his work with the method of *Thongkamwitoon et al* [4] on the ICL data-set.

- **Patches dimension**

  About the dimension of the patches, we opted for 5×5 pixels as the size of the overlapping patches for feature extraction because the paper shows that it gives the best results compared to 7×7 and 3×3 patches within the chosen width of 2048 pixels.

- **Data-set choice**

  To our knowledge, there exist two publicly available, large-scale data-sets of high-resolution and high-quality recaptured images for testing and comparing methods of image recapture detection. The first data-set is the one constructed by *Cao* and colleagues in 2010, and referred to as the ROSE data-set. It is composed by 2776 recaptured images and 2710 single captured images. The second and more recent data-set was constructed by *Thongkamwitoon et al.* in 2015, and it is called the ICL data-set: it contains 900 single captured images and 1440 recaptured images.

  We have decided to test our method on the ICL data-set because the author reports that it seems more challenging than the ROSE data-set, even if smaller. Another more practical reason is that we could not utilize the ROSE data-set due to its restricted access policy, while instead the ICL data-set is freely accessible.

  Both data-sets include a large number of high-resolution and high-quality single captured and recaptured images for which a variety of digital cameras from different makers were used as image acquiring devices. The authors of the ROSE data-set also used multiple LCD screens from different manufacturers for image recapturing.

  So, while single captured images have more or less similar statistical properties within both the data-sets, there is a noticeable statistical difference between recaptured images from the ICL and ROSE data-set since the adopted recapturing technique differs, which leads to different properties of the induced distortion.

  This can be proved observing that the recaptured images in the ICL data-set are of higher quality than those in the ROSE data-set. The reason is that the recapturing parameters were carefully determined with convincing theoretical justification, so that aliasing distortions have been largely reduced to a nearly invisible level.

  So, all in all, despite being more recent, ICL data-set comprises higher-quality recaptured images with almost invisible aliasing distortions, that is why the author prefers it for testing the algorithm.

Results and final considerations

We now analyze the results obtained by implementing the method proposed in the paper [1].

The forensic performance has been evaluated by using the classification accuracy of single captured images, recaptured images and all the images in the testing set. To enhance the statistical significance of the obtained results, we performed 50 runs of 5-fold cross validation, and for each run we had different and randomly partitioned training and testing sets. At the end, we reported the classification accuracy as the mean of the results obtained from the 50 runs.

We first of all tested our code on a simplified data-set, composed by only 100 images. This data-set is available at the same link of the ICL data-set and it is composed by 50 single captured and 50 recaptured images. Such data-set also provides a .txt file that lists the type of each image in the same format of the .txt file that our code accepts as input. After the 50 runs, we obtained:

- overall accuracy: 94.8506%;
- original captured accuracy: 93.6087%;
- recaptured accuracy: 96.2679%.

Afterwards, we extended the computation on the whole ICL data-set. The achieved results are the following:

- overall accuracy: 97.3379%;
- original captured accuracy: 96.2266%;
- recaptured accuracy: 98.0360%.

We can notice that the accuracies of the reduced data-set are quite lower than the ones of the complete data-set. Additionally, for what concerns the accuracies obtained in the single iterations of the cycle, in the complete case these values are steady around their final mean value, instead for the reduced case they vary in a wide range. This is due to the fact that in the reduced data-set the ratio 15:100 between the quantity of the training images and the testing images is very unbalanced and problematic, so the training phase is less reliable.

In order to compare our accuracies with the ones reported in the paper, these are the values to be considered:

Method	Overall accuracy	Original captured accuracy	Recaptured accuracy
Paper	97.71%	96.27%	98.62%
Ours	97.34%	96.23%	98.04%

Table 3.1: Results compared

It can be observed that our accuracy values are slightly lower than the ones reported in the paper. One of the reasons for this small difference may be that since the author has not specified an exhaustive method to determine the hyperparameters in the cross validation, he might have adopted a different approach from ours. Indeed, we calculated the values of C and γ through a grid search on predefined sets of values.

Additionally, generally speaking, we can notice that the accuracies of recaptured images for the complete ICL data-set are quite higher than the ones of single captured images. This is probably due to the higher number of recaptured images contained in the data-set with respect to the number of single captured images. This holds both for our results and for the ones reported in the paper.

Although it is quite simple as it is image-statistics-based, the method proposed by the author appears to be discriminative enough to well detect the differences between single captured and recaptured images. It achieves slightly higher classification accuracy when compared with other methods from which it is different due to the fact that the current trend is the one of developing more sophisticated recapture forensic methods by either combining multiple features or using complex machine learning tools.

Bibliography

[1] Kai Wang (2017) *A simple and effective image-statistics-based approach to detecting recaptured images from LCD screens*, Univ. Grenoble Alpes.

[2] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin (2016) *A Practical Guide to Support Vector Classification*, Department of Computer Science, National Taiwan University.

[3] Cao, H., Kot, A.C. (2010) *Identifcation of recaptured photographs on LCD screens*, in: Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 1790-1793.

[4] Thongkamwitoon, T., Muammar, H., Dragotti, P.L. (2015) *An image recapture detection algorithm based on learning dictionaries of edge profiles*, IEEE Transactions on Information Forensics and Security 10, 953-968.