Image Forgery Classification : Tampering Detection

Mid Term Report

Team ID - 17

AVISH SANTHOSH
ee20btech11007@iith.ac.in
DEVULAPALLI SAI PRACHODHAN
ee20btech11013@iith.ac.in
FASAL MOHAMED T
ee20btech11016@iith.ac.in
NANDYALA VISHWARAM REDDY
ee20btech11059@iith.ac.in

Abstract

Images represent an effective and natural communication medium for humans due to their immediacy and the ease with which image content can be understood. The widespread availability of image editing software tools makes it easier to alter image content or create new images. As a result, the possibility of tampering and counterfeiting visual content is no longer restricted to experts. This situation underscores the need for methods to verify the truthfulness of images and assess their quality. Answering these queries is relatively easy when the original image is known. However, in practical cases, almost no information can be assumed to be known a priori about the original image, making our task difficult. This paper explores all possible methods that can be applied to detect tampering forgery and its subtypes, such as Copy-move, splicing, and part-removal.

Index Terms – Passive methods for forgery detection, Gabor filter, SIFT, Feature matrix, Clustering, Traditional methods

0. Aim

For this report, our objective is to address the challenge of detecting copy-move forgery using traditional methods.

1. Introduction

Copy-move forgery can be defined as the act of duplicating one or more regions of an image and pasting them in another location within the same image to create a new image that misrepresents the original context or intent. Copy-move forgery is commonly used in digital image tampering to conceal or add objects in an image.

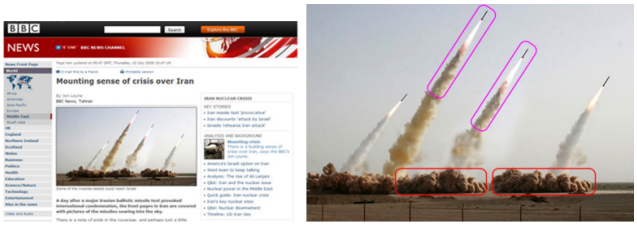

Figure 1. Example of image tampering that appeared in press in July, 2008. The image on the right is a forged one, – it shows four Iranian missiles, but only three of them are real. Two different sections (encircled in red and purple, respectively) have been replicated by applying a copy-move attack.

We have classified the image passive forgery detection techniques into **Traditional** and **Deep learning** based methods. Our general outline to detect copy-move forgery as discussed in preliminary project report is as shown in the below figure.

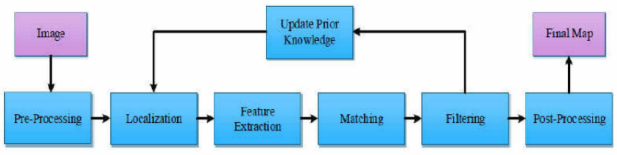

Figure 2. Common Framework Of the Copy-move Forgery Detection Technique [9]

According to [1], we have divided copy-move forgery detection techniques divided into two:

- **Block-based techniques**: In block-based methods, an image is split into overlapping blocks and then feature vectors are computed corresponding to each block. These vectors are matched to identify tampered images taking into account some empirical parameters.
- **Key-point based techniques**: In key-point-based methods, key-points are extracted and matched to detect image forgeries.

This report will cover two traditional methods to detect copy-move forgeries. One of them is block-based, and the other is key-point based.

2. Build up on previous report

2.1. Our plan

After the submission of our PPR, we have decided to focus on traditional methods up till the mid-term report submission and after that explore DL-based methods. The four of us divided the work between ourselves, such that we focused on the key-point based and block-based methods in groups of two. As part of that we have read some additional papers that are focused on the above methods and finally ended with [6] and [2] to reproduce results.

2.2. Dataset

As said in the previous report we have used this [3] to obtain our datasets. For the block-based method, the dataset used for results can be downloaded here [8] (Small-image category database). For the key-point based method, dataset used is available here [2] (MICC-F2000).

2.3. Metrics

We have used Accuracy, Precision, Recall and F1-score as our metrics to test the working of our algorithm.

$$\text{Accuracy} = 1 - \frac{\text{N(e)}}{\text{L}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{3}$$

$$\text{F1-score} = \frac{\text{2*precision*recall}}{\text{precision+recall}} \tag{4}$$

2.3.1 Notation followed

L = Number of Total Images
N(e) = Number of errors = Number of images classified as forged given they are original + Number of images classified as original given they are forged
TP = Number of images detected as forged being forged
FP = Number of images detected as forged being original
FN = Number of images detected as original being forged
TN = Number of images detected as original being original

3. Block-based method using Gabor filter

3.1. General Overview

As part of this class of methods, we have followed this [6] to construct our algorithm and results. This method uses Gabor filters to detect the copy-move forgery.

3.1.1 Idea

Since copy-move forgery is done by copying and then duplicating the required area, there must be at least two similar regions in the image. A natural image, on the contrary, is very unlikely to have two similar regions (except for the images depicting scenery, or items of such a scale, having large, smooth variations). Thus we can say the image is copy-move forged if two similar regions are detected.

3.1.2 Gabor filters

Gabor filters are based on the idea of analyzing local frequency and orientation information in an image. They can detect edges and texture features in an image and are often used in tasks such as image segmentation, object recognition, and face detection. Gabor filters have been widely adopted in various applications due to their ability to capture both spatial and frequency information in an image. Its impulse response is given below,

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{[2\pi Wix - \frac{1}{2}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2})]} \tag{5}$$

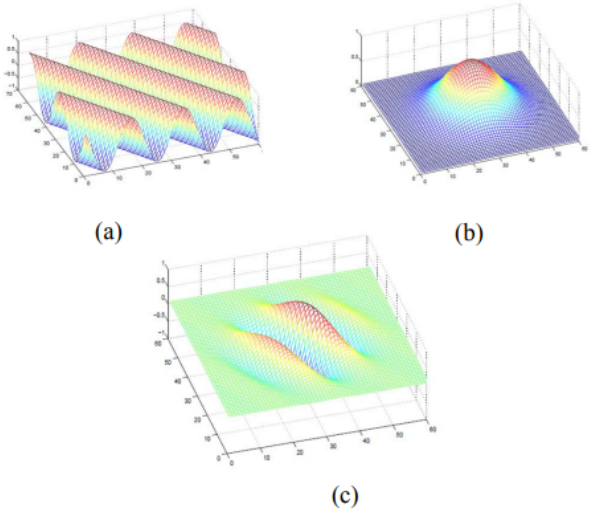

Figure 3. Gabor filter composition: (a) 2D sinusoid ori-

ented at 30° with the x-axis, (b) a Gaussian kernel, (c) the corresponding Gabor filter. [6]

3.2. Algorithm and Workflow

3.2.1 Briefing

This algorithm is based on rotation-invariant Gabor filters which are further used to extract the feature matrix. This feature matrix is sorted lexicographically to group similar regions (if detected) and classify the image as a forgery or not.

3.2.2 Workflow

- **Block generation** : For every pixel at (i, j) we have to extract square block of size $B \times B$. Doing this for whole image of size $M \times N$, we get $(M - B + 1) \times (N - B + 1)$ blocks.

- **Gabor filters generation**: We have to pre-compute the Gabor filters for different scales and rotations using the impulse response as above. After computing Gabor filters for fixed scale but different rotations, we should average the filters to get rotation-invariant filter for given fixed scale. We have to repeat the same for all considered scales. In this project, we are taking scale range in between $[0.5, 1.5]$ (which is reasonable).

  Reason for pre-computing : As we generally use a bunch of images to get our metrics and filters are fixed so we need not to compute the filters for all images so that we can reduce time taken to run the code.

- **Filtering and Feature matrix construction**: After pre-computing the Gabor filters, filter the images using linear convolution for each of the above mentioned rotation-invariant filters for different scales. For a image and its block, compute the mean of obtained convolution **(dimensionality reduction)** to represent as its feature entry at a given scale.

  Reason of taking mean : As block size decreases then for given small block we will not have much variations in pixels of image so we can approximate the block as the mean of all pixels obtained by Gabor filtering.

  Now the feature vectors are constructed for each block using the mean for each scale. All these feature vectors for all blocks are aggregated in a matrix by forming feature vector matrix. Now we sort the feature vector matrix lexicographically to do further processing. In sorting process, we also make a note of the top corner indices of the block considered.

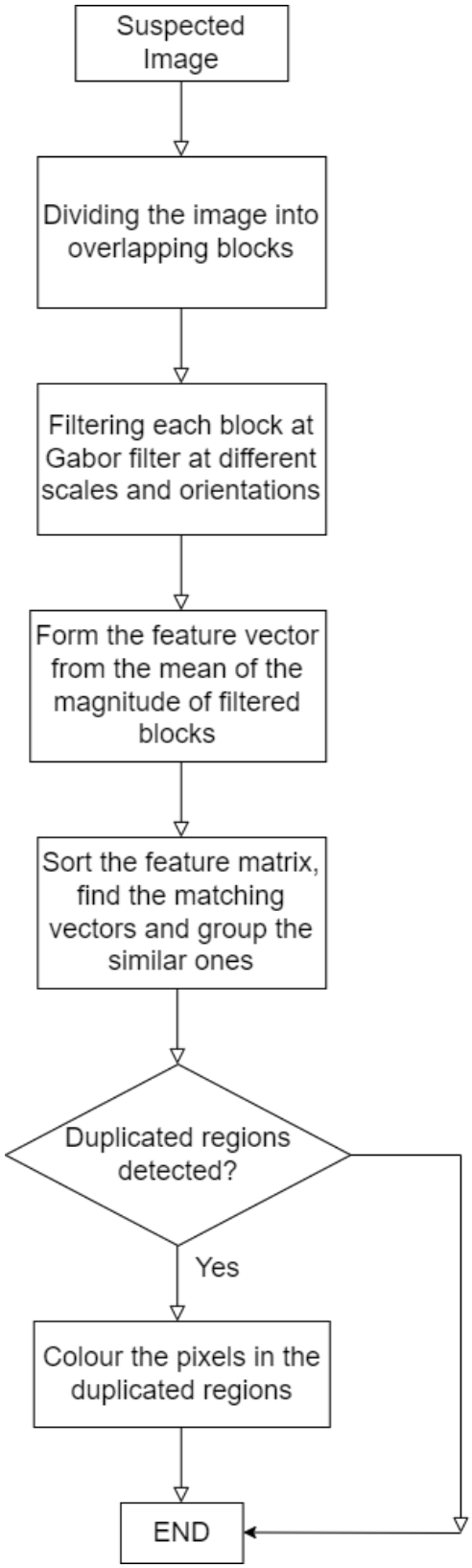

Figure 4. Showing workflow of the algorithm [6]

- **Matching** : After getting feature matrix, consider the i^{th} and j^{th} rows of the matrix let say a_i and a_j (here rows in that matrix represent feature vectors of each block) and then if j-i¡N_f then calculate q = $||a_i - a_j||_2$ and if q < $D_{similar}$ along with distance between the corresponding blocks of i^{th} and j^{th} row is > N_d then we say those regions are matched and forgery is detected.

3.2.3 Mathematics involved

- **For Rotation invariant vector calculation**

$$g_{mn}(x, y) = s_i g(x', y') \tag{6}$$

s_i = m^{th} scale in our scale range

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta_n) & sin(\theta_n) \\ -sin(\theta_n) & cos(\theta_n) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where θ_n = $\frac{n\pi}{K}$, n = 0,1,....,K-1. K denotes number of angles taken = 15 (for this project we took).

$$g_m(x, y) = \frac{\sum_{n=0}^{K-1} g_{mn}(x, y)}{K} \tag{7}$$

The above $g_m(x, y)$ is the required rotation invariant filter for given m^{th} fixed scale.

- **Feature vector calculation**

$$J_m(x, y) = I_b(x, y) * g_m(x, y) \tag{8}$$

* indicates linear convolution and I_b = image block b

$$\mu_m = \frac{\sum_x \sum_y J_m(x, y)}{N} \tag{9}$$

$$\text{feature vector, } f_i = \begin{bmatrix} \mu_0 & \mu_1 & .....\mu_{s-1} \end{bmatrix} \tag{10}$$

s denotes number of scales in scale range.

$$\text{feature matrix, } A = \begin{bmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_{b-1} \end{bmatrix} \tag{11}$$

b denotes number of generated blocks of image I.

The above equations are used but for project I am using OpenCV to generate different scale and rotated Gabor filters along with different luminance levels, and thus Gabor filter kernel is more useful than others.

3.3. Results

The below are the results of the metrics calculated for first 40 images from the dataset (excluding the bit mask images) and parameters that I have considered are $N_f = 3$, $D_{similar} = 3$, $N_d = 16$, $B = 8$, ilter window size = 5×5, scales range = $[0.5, 1.5]$ with change of 0.2 i.e scales = $[0.5, 0.7, 0.9, 1.1, 1.3, 1.5]$ and $K = 15$. The dataset that I have considered for this method is [8] which has images with different distortions like gamma compression, jpeg compression and image feature rotations with different rotations and different compression levels along with AWGN noise added images with different standard deviation of noise to test the robustness of algorithm with respect to these attacks also.

Metric	Score
Accuracy	0.6
Precision	0.6
Recall	1
F1-score	0.75

Table 1. Results of the above block-based method

3.4. Advantages of using this method

- Intuitive and easily understandable.
- In general, robust to rotation, gaussian noise addition, jpeg and gamma compression in compared to other algorithms.

3.5. Disadvantages of using this method

- Computationally expensive for large images. Lot of memory and time taken to run code because we are will be having 8×8 sized $(M - B + 1)(N - B + 1)$ blocks and in memory prospective we are storing $(M-B+1)(N-B+1)S$ bytes of memory for feature matrix. (S is number of scales)
- Less accuracy and will detect forged for original images having large similar flat portions.
- It is more of like a brute force method and it is possible to optimize the code using some other technique.

4. Key-point based method

4.1. General Overview

The forgery detection technique detailed in [2] proposes a method for detecting copy-move attacks in digital images and additionally, recovering the transformations applied to the copied regions. This method employs the Scale-Invariant Features Transform (SIFT) to extract the keypoints along with the respective feature descriptors.

4.1.1 Idea

The method proposed, based on the SIFT algorithm to extract key-points, takes advantage of the fact that the copied part has basically the same appearance of the original patch. Thus, key-points extracted in the forged region will be quite similar to the original ones, and hence, we can rely on key-point matching techniques, and further clustering, to verify whether there are regions which are identical.

4.1.2 SIFT

SIFT, or Scale Invariant Feature Transform, is a feature detection algorithm, that helps locate the key-points of the image; these features are invariant to image scaling and rotation. The SIFT features are well-localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise.

The algorithm can be broken down into four steps [4]:

1) **Scale-space extrema detection:**
The algorithm searches over all scales and image locations, to look for potential location for finding features. It uses the Difference-of-Gaussian function (Figure 5) to efficiently identify potential interest points that are invariant to scale and orientation.

2) **Key-point localization:**
We accurately locate the feature keypoints, by fitting each candidate with a detailed model to determine location and scale. Key-points are selected based on measures of their stability.

3) **Orientation assignment:**
One or more orientations are assigned to each key-point to achieve invariance to image rotation; this is because, in this step, the image data has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing rotational invariance to any future transformations.

4) **Key-point descriptor:**
The local image gradients are measured at the selected scale in the region around each key-point; as per the text, a 16×16 neighbourhood around the key-point is taken, which is divided into 16 sub-blocks of 4×4 size. For each sub-block, an 8-bin orientation histogram is created, this creating a total of 128 bin values. Thus the key-points are represented as a high dimensional vector.

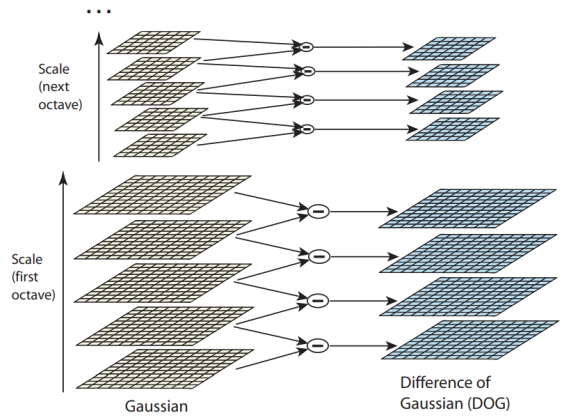

Figure 5. For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the Difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process is repeated. [6]

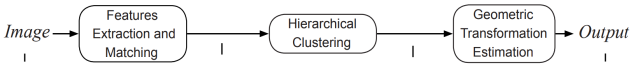

Figure 6. Overview of the proposed system [2]

4.2. Algorithm and Workflow

4.2.1 SIFT features extraction and multiple key-point matching

Given a test image, a set of key-points $\mathbf{X} = \{x_1, \ldots, x_n\}$ with their corresponding SIFT descriptors $\mathbf{F} = \{f_1, \ldots, f_n\}$ are extracted. The best candidate match for each key-point x_i is found by identifying its nearest neighbor from all the other $(n-1)$ key-points of the image, which is the key-point with the minimum Euclidean distance in the SIFT space i.e, the distance between their descriptors.

We have used `cv2.SIFT_create()` function from OpenCV, as opposed to the original SIFT implementation in `cv2.xfeatures2d.SIFT_create()` as the latter is a patented algorithm [5]. Other alternatives for feature extraction are SURF, ORB, BRISK etc.

Image-space refers to the spatial coordinates of an image(s) denoted as I with $m \times n$ elements, where m and n are respectively the number of rows and columns in the image. The elements in image space, $I(i,j);\ (i = \{1, \ldots m\}; j = \{1, \ldots, n\})$ are image pixels. They represent spatial sampling units from which electromagnetic energy or other phenomena are recorded. All possible image pixel values constitute the feature space

V. One image band constitutes a one-dimensional feature space.

If we choose to decide that two key-points match by thresholding the Euclidean distances (using a global threshold), we obtain incorrect results (due to the high dimensionality of the feature space), in which some descriptors are much more discriminative than others.

A more effective procedure is to use the ratio between the distance of the closest neighbor to that of the second-closest one, and comparing it with a threshold T (often fixed to 0.6). To elaborate, given a key-point we define a similarity vector $D = \{d_1, \ldots, d_{n+1}$ that represents the sorted euclidean distances, with all possible pairs of key-points taken into account. Following this idea, the key-point is said to be matched only if $\frac{d1}{d2} < T$, where $T \in (0, 1)$. This procedure is referred to as the 2NN test.

The key drawback of the 2NN test is that it is incapable of detecting multiple matching key-points. The authors propose a modification; this is referred to as the generalized 2NN test, or the g2NN test. We start from the observation that in a high-dimensional feature space such as that of SIFT features, choosing pairs of key-points that differ a lot, share the property that their Euclidean distances are very high and very similar in value. Moreover, key-points corresponding to similar features show low Euclidean distances.

The g2NN test proposes iterating the 2NN test between d_i and d_{i+1} until $\frac{d_i}{d_{i+1}} > T$ (in their findings they have empirically set this value to 0.5; we have however set it to 0.6 since it produces better results). If k is the value of i for which the procedure stops, each key-point occurring as one among the Euclidean distances $\{d_1, \ldots, d_k\}$, (where $1 \leq k \leq n$) is considered as a match for the inspected key-point. Finally, by iterating over all such key-points in $\mathbf{X}$, we can obtain the set of matched points. All the matched key-points are retained, but isolated ones are no longer considered in subsequent processing steps.

Already at this stage a preliminary idea of the authenticity of the image can be gleaned. But it can happen that images that legitimately contain areas with very similar texture, like a picture of the sky or of sand in the desert, may yield matched key-points which is a false alarm. The following two steps of the proposed methodology reduce this possibility.

4.2.2 Clustering and forgery detection

To identify possible cloned areas, an agglomerative hierarchical clustering is performed on spatial locations (i.e. (x, y) coordinates) of the matched points.

Agglomerative (bottom-up) clustering creates a hierarchy of clusters that may be represented by a tree structure. The algorithm starts by assigning each key-point to a cluster; then it computes all the reciprocal spatial distances among clusters, finds the closest pair of clusters, and finally merges them into a single cluster. This computation is iteratively repeated until a final merging situation is achieved. The way this final merging can be accomplished is conditioned both by the linkage method adopted and by the threshold used to stop cluster grouping; the authors have mentioned 3 linkage methods (Single, Centroid and Ward's linkage), but in the view of time we have used only Ward's linkage method, considering that it has the highest True Positive Rate experimentally. Further, a cut-off threshold of $T_h = 1.6$ is advised by the paper for best results.

The agglomerative hierarchical clustering algorithm implemented in scikit-learn's `AgglomerativeClustering` class [7] does not merge clusters explicitly. Instead, it computes a hierarchy of clusters using a specified linkage criterion and a distance threshold, which determines when clusters should stop being merged. The `AgglomerativeClustering` class is used to perform the clustering, and the `distance_threshold` parameter is set to the value of T_h, which is the threshold used to stop cluster grouping.

The resulting clustering is represented as a hierarchy of nested clusters, which can be visualized as a dendrogram. The dendrogram can be cut at a certain height to obtain a clustering with a specified number of clusters. In the code, the `n_clusters = None` parameter is used to obtain the full hierarchy, but the number of clusters can be set to a specific value by specifying `n_clusters = k`, where k is the desired number of clusters.

At the end of the clustering procedure, clusters that do not contain a significant number (more than three) of matched key-points are eliminated. Finally, they considered that an image has been altered by a copy-move attack if the method detects two (or more) clusters with at least three pairs of matched points linking a cluster to another one.

4.2.3 Geometric transformation estimation

The authors have further devised a technique to identify the geometric transformation that is present in the copy-move forgery. We plan to implement it by the next report.

4.3. Results

The following results were obtained using the MICC-F2000 dataset [2], which consists of copy-move and splicing images. We chose 40 images, out of which 8 images are original and 32 images are tampered. Using the same metrics that the authors used, we obtained the following results. We first define the following metrics.

4.3.1 Metrics

1. True Positive Rate (TPR)

$$\text{TPR} = \frac{\text{No. of images detected as forged, being forged}}{\text{No. of forged images}}$$

2. False Positive Rate (FPR)

$$\text{FPR} = \frac{\text{No. of images detected as forged being original}}{\text{No. of original images}}$$

4.3.2 Specifications

1. `cv2.SIFT_create()`

   In this function, we have set `n_features = 9000` (n can be increased to increase the accuracy).
   Each of these 9000 key-points have a descriptor array consisting of 128 points and we considered the default nOctave = 3.

2. `AgglomerativeClustering()`

   Here, we are using the following parameters:

```
n_clusters = None
metric = Euclidean
linkage = Ward
distance_threshold = 1.6
```

Metric	Score
TPR	0.68
FPR	0.14

Table 2. Results of the key-point based method

We plan on improving on these methods in the further reports.

References

[1] Soumen Ba Anuja Dixit. A fast technique to detect copy-move image forgery with reflection and non-affine transformation attacks. 1

[2] R. Caldelli A. Del Bimbo-G. Serra I. Amerini, L. Ballan. “a sift-based forensic method for copy-move attack detection and transformation recovery”, ieee transactions on information forensics and security, vol. 6, issue 3, pp. 1099-1110, 2011. 2, 4, 5, 7

[3] Chao Zhang Jingjing Chen Yu-Gang Jiang Larry S. Davis Junke Wang, Zhenxin Li. Fighting malicious media data: A survey on tampering detection and deepfake detection. 2

[4] David G. Lowe. Distinctive image features from scale-invariant key-points. 5

[5] OpenCV. cv::sift class reference. 5

[6] Manju Manuel Raichel Philip Yohannan. Detection of copy-move forgery based on gabor filter. 2, 3, 5

[7] scikit learn. sklearn.cluster.agglomerativeclustering. 6

[8] Grgic S. Grgic M. Tralic D., Zupancic I. ”comofod - new database for copy-move forgery detection”, in proc. 55th international symposium elmar-2013, pp. 49-54, september 2013. 2, 4

[9] A. Mahmoudi-Aznaveh Zandi, M. Iterative copy move forgery detection based on a new interest point detector. 1