IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2022–11–07.
Submitted on 2022–09–29, accepted on 2022–09–29.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.432

Image Forgery Detection via Forensic Similarity Graphs

Marina Gardella[1], Pablo Musé[2]

[1]Université Paris-Saclay, Université Paris Cité, ENS Paris-Saclay, CNRS, SSA, INSERM, Centre Borelli, Gif-sur-Yvette, France
[2]IIE, Facultad de Ingeniería, Universidad de la República, Uruguay
marina.gardella@ens-paris-saclay.fr, pmuse@fing.edu.uy

Communicated by Jean-Michel Morel *Demo edited by* Marina Gardella

Abstract

In the article "Exposing Fake Images with Forensic Similarity Graphs", O. Mayer and M. C. Stamm introduce a novel image forgery detection method. The proposed method is built on a graph-based representation of images, where image patches are represented as the vertices of the graph, and the edge weights are assigned in order to reflect the forensic similarity between the connected patches. In this representation, forged regions form highly connected subgraphs. Therefore, forgery detection and localization can be cast as a cluster analysis problem on the similarity graph. The authors present two graph clustering methods to detect and localize image forgeries. In this paper, we present briefly the method and offer an online executable version allowing everyone to test it on their own suspicious images.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.txt` file of the archive. The original implementation of the method is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: image forensics; forgery detection; graph clustering

1 Introduction

Seeing is no longer believing when it comes to digital contents. Indeed, during the last years, we have observed a huge raise in the quantity of manipulated material being shared in internet. Visual evidence, such as the one used in court, has been questioned given the ease with which it can be tampered. To address these concerns, image forensics techniques are being developed. These tools provide scientific evidence to help determine the authenticity of the images under question.

[1]https://doi.org/10.5201/ipol.2022.432
[2]https://omayer.gitlab.io/forensicgraph/

Classical image forensics methods aim at reconstructing the image formation process by detecting visually imperceptible traces that are left in the image at each step of the camera processing pipeline [21]. Indeed, several methods have been proposed to tackle specific traces such as demosaicing inconsistencies [1, 3], JPEG artifacts [24, 20], noise incoherence [5, 10] and more [11].

Recently, deep learning methods for forgery detection have been developed. Some of these methods are directly trained on forged images and aim at identifying specific manipulations [23, 22, 15]. However, training a network to learn all possible manipulations is not feasible.

Following this observation, other deep learning approaches have been proposed [2, 6, 12, 16]. These methods mimic the hand crafted methodology adopted by classical methods. Namely, they aim at extracting features related to the camera processing chain, and to detect local inconsistencies in these features. The main difference is that in this case, these features are learnt.

In [17], the forensic potential of features learnt for source camera classification is shown. Indeed, the authors develop the "forensic similarity score" that aims at distinguishing if two image patches share the same forensic traces or not. In this article, we explore the approach in [16] which exploits this forensic similarity measure for forgery detection.

2 Method

2.1 Forensic Similarity Score

In their previous work [17], Mayer and Stamm introduce a new deep learning technique to identify if two image patches carry the same or different forensic traces. Forensic traces are the visually imperceptible artifacts left in an image during the image formation process and subsequent post-processing. This approach takes two image patches as input and outputs a single score between 0 and 1 that measures the forensic similarity of the two image patches. Scores near 0 indicate low coincidence of forensic traces while scores near 1 indicate that similar forensic traces were found in both patches.

To do so, they propose a system comprising two CNN feature extractors in a siamese configuration, followed by a comparison network that evaluates how similar both features are. The feature extractor is trained in a first phase to learn camera model features. A second training phase is then carried out to update the weights to target a specific task. A brief explanation of this method together with an online executable version of it can be found in [9].

The weights used in the rest of this work are those corresponding to the source camera model comparison task. Two pre-trained versions of the proposed network are available, one trained on 128×128 patches and another one trained on 256×256 patches.

2.2 Forensic Similarity Graph

A similarity graph is a graph where the edge weights represent the similarity between the connected nodes. In [16], Mayer and Stamm propose to construct a similarity graph from the image using a similarity measure that reflects the similarity of the processing pipeline. To do so, they represent the image patches as nodes and assign weights to the edges according to the forensic similarity score between them, that was previously reported by the same authors in [17]. The resulting graph is a fully connected graph, where edges connecting patches having similar forensic traces will have weights near 1, and edges connecting patches with low forensic similarity will have weights near 0.

In this representation, patches that have undergone the same processing operations are expected to form communities. Communities are characterized by strong connections within the members and

weak connections to non-members. This way, forgery detection can be formulated as a community detection problem, and forgery localization as a community partition problem.

2.3 Forgery Detection and Localization

As mentioned in the previous section, the problem of forgery detection and localization can be regarded as a clustering problem, where one needs to establish the number of clusters (one for non-tampered images, bigger than the one for forged images) and the partition in case there is more than one cluster. Several community detections techniques have been reported in the literature [8]. In their work, Mayer and Stamm focus on two particular techniques: spectral clustering and modularity optimization. However, it is important to notice that it is straightforward to extend the proposed approach to other clustering methods.

2.3.1 Spectral Clustering

A weighted graph $G = (V, W)$ is a pair where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and W is a symmetric matrix satisfying $W_{ij} \geq 0$ for all $i, j = 1, \ldots, n$ and $W_{ii} = 0$ for all $i = 1, \ldots, n$. The weight matrix W can be considered as an extension of the adjacency matrix for non-weighted graphs. Indeed, if $W_{ij} \in \{0, 1\}$, both definitions become equivalent.

Spectral clustering methods aim at partitioning the graph G based on the spectrum of W itself or any other matrices built on it. In this work, the authors focus on the study of the spectrum of the graph Laplacian matrix, defined as

$$L = D - W, \tag{1}$$

where D is the degree matrix defined as $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ if $i \neq j$. The matrix D generalizes the degree matrix for non-weighted graphs. Indeed, if $W_{ij} \in \{0, 1\}$, for all i, j, both definitions are equivalent.

Since, by construction, all the rows of L sum up to 0, $\lambda_1 = 0$ is always an eigenvalue for L, corresponding to the eigenvector $(1, 1, \ldots, 1)$. Furthermore, its multiplicity corresponds to the number of connected components in the graph. Indeed, the membership vectors (i.e. vectors having ones for the connected nodes and zeros for the other nodes) will be eigenvectors associated to the 0 eigenvalue. This property of the graph Laplacian can be applied to community structure detection, by observing that if there are weakly linked sub-graphs, the smallest non-zero eigenvalue will still be close to zero [7].

The authors use the *eigengap heuristic* [14] to detect if more than one community exists by computing the second smallest eigenvalue λ_2, and comparing it to a pre-defined threshold $\tau = 100$, derived empirically. If λ_2 is smaller than τ, the image is classified as forged and if it is bigger, as non-forged.

In case the image is considered as tampered, forgery localization can be performed by finding the community partition. The authors only consider the case when two communities exist. In this case, graph bipartition can be achieved from the eigenvector of the second smallest eigenvalue λ_2 of the graph Laplacian matrix [7]. To do so, it is enough to compute an eigenvector u_2 associated to λ_2 and partition the graph according to the sign of each component of u_2. Indeed, for each vertex v_i, labels are assigned according to

$$c_i = \begin{cases} 1, & \text{if } u_2^i \geq 0 \\ 0, & \text{if } u_2^i < 0, \end{cases} \tag{2}$$

where c_i denotes the predicted community for the node v_i.

2.3.2 Modularity Optimization

Modularity was introduced as a measure of the quality of a particular graph partition [19]. It is built on the observation that random graphs are not expected to have communities structures. Therefore, by comparing the observed edge density within a community to the expected edge density given by the background model, one can assess the meaningfulness of the given community structure. Modularity can be expressed as

$$Q = \frac{1}{2m} \sum_{i,j} \left(W_{ij} - E(W_{ij})\right) \delta(c_i, c_j), \tag{3}$$

where the sum is taken over all the pairs of vertices, m is the weighted total number of edges $m = \frac{\sum_{i,j} W_{ij}}{2}$, W is the weights matrix, $E(W_{ij})$ is the expected weight for an edge connecting vertices i and j given by the background model, δ is the Kronecker δ-function and c_i is the community to which vertex i belongs.

The choice of the null model is not entirely unconstrained according to Newman [18]. Firstly, $E(W_{ij})$ should be equal to $E(W_{ji})$ since we are considering undirected graphs. Secondly, Q should be equal to 0 when all the vertices belong to a single community. Setting $c_i = c_j$ for all $i, j = 1, \ldots, n$ it follows that

$$\sum_{i,j} W_{ij} = \sum_{i,j} E(W_{ij}) = 2m. \tag{4}$$

Despite these constraints, there are several possible background models satisfying these conditions. Mayer and Stamm adopt the same null model as Newman [18]. Firstly, this model imposes that the expected degree of each vertex is equal to the actual degree in the graph,

$$\sum_{i} E(W_{ij}) = D_{jj}. \tag{5}$$

This condition implies automatically the constraint $\sum_{i,j} E(W_{ij}) = 2m$. Secondly, the null model suggested by Newman imposes that the expected weight of an edge connecting vertices i and j is the product of a function f of their degrees

$$E(W_{ij}) = f(D_{ii}) f(D_{jj}). \tag{6}$$

Equation (5) and Equation (6) imply

$$E(W_{ij}) = \frac{D_{ii} D_{jj}}{2m}. \tag{7}$$

Therefore, under this background model, modularity can be written as

$$Q = \frac{1}{2m} \sum_{i,j} \left(W_{ij} - \frac{D_{ii} D_{jj}}{2m} \right) \delta(c_i, c_j). \tag{8}$$

Modularity optimization aims at finding the community partition for which the modularity is maximized, i.e.

$$Q^{\text{opt}} = \max_{c_1, \ldots, c_n} \frac{1}{2m} \sum_{i,j} \left(W_{ij} - \frac{D_{ii} D_{jj}}{2m} \right) \delta(c_i, c_j). \tag{9}$$

Different strategies to optimize the modularity have been addressed in the literature [8]. Mayer and Stamm use the fast greedy technique introduced in [4], where vertices are successively joined to form larger communities in a way such that modularity increases.

If the optimal value for the modularity is close to 0, there is no evidence of community structure [19]. Therefore, to detect if an image is tampered the authors compare Q^{opt} to a pre-defined threshold $\tau = 0.025$. Then, if the observed Q^{opt} is larger than τ, the image is classified as forged; otherwise, the image is classified as pristine.

In case the image is classified as forged, the same optimization problem can be used to partition the image into clusters. Indeed, in that case the optimization problem can be expressed as

$$c_1^{\text{opt}}, \ldots, c_n^{\text{opt}} = \underset{c_1,\ldots,c_n}{\text{argmax}} \frac{1}{2m} \sum_{i,j} \left(W_{ij} - \frac{D_{ii} D_{jj}}{2m} \right) \delta(c_i, c_j).$$

The authors impose two clusters, and set an edge weighting threshold equal to 0.9, meaning that all the edges with lower weights are set to 0. Figure 1 shows an example of the results obtained using the modularity optimization algorithm.

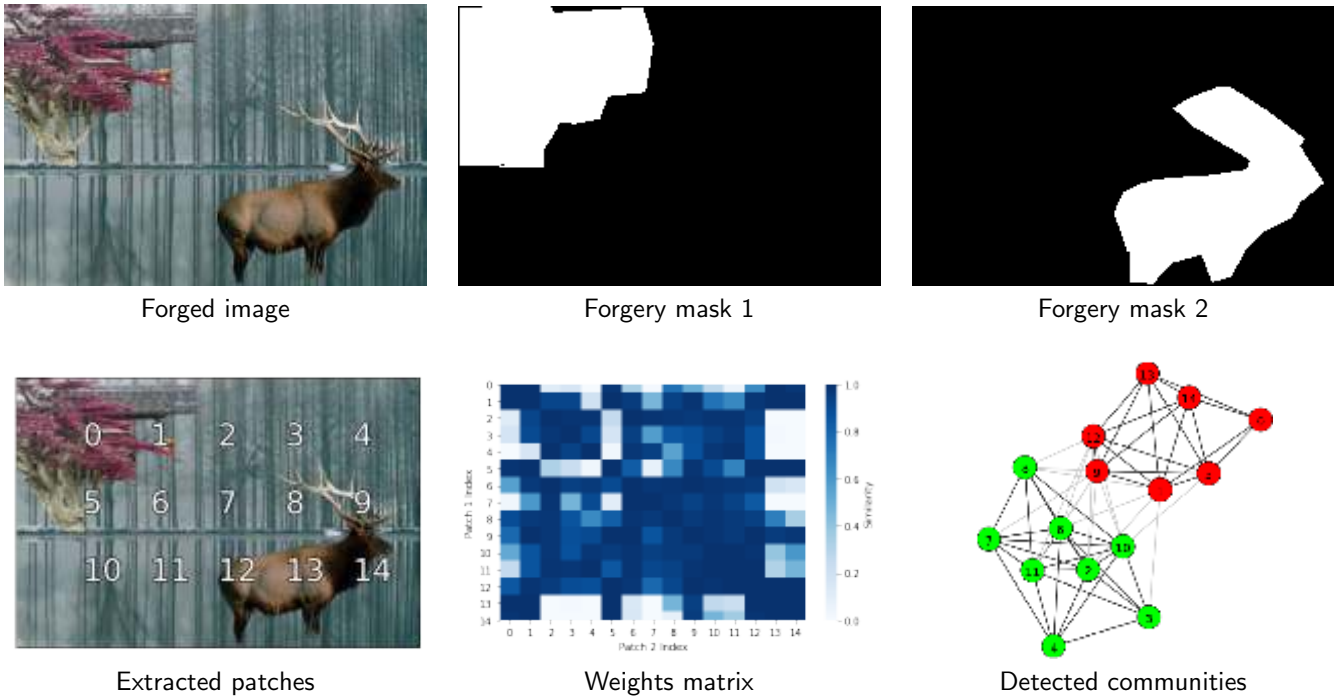


Figure 1: Examples of weight matrices and graph partitions on a forged image from the MISD Dataset [13], having two spliced regions. The community detection algorithm used for these examples is the modularity optimization, with patches of size 128×128 and 50% of overlap. The edge weight threshold used is equal to 0.9. We observe that the community partition found by the algorithm points to the forgery masks (in red).

2.4 Pixel-level Masks

Once an image is classified as forged and a community partition has been performed, labels are assigned to the nodes. In our setting, nodes are patches, which are usually taken with overlap. Therefore, it is necessary to aggregate this information to produce pixel-wise masks. To do so, Mayer and Stamm first construct a pixel-map. For the case in which the number of clusters is equal to 2, as it is in our setting, we can assume that there are only two labels: 0 and 1. The pixel-map can be then expressed as

$$P(x, y) = \frac{1}{\#\{R : (x, y) \in R\}} \sum_{R:(x,y)\in R} c(R), \tag{10}$$

where (x, y) is a pixel, R is a patch and $c(R)$ is the predicted label for the patch. The factor $\frac{1}{\#\{R:(x,y)\in R\}}$ normalizes the pixel-map to take into account the fact that pixels near the borders are covered by fewer patches.

This pixel-map is further smoothed using a 32×32 Gaussian kernel and compared to a threshold in order to produce a binary output. The threshold used by the authors is equal to 0.5. Finally, the smallest region is marked as forged and the largest one as pristine.

3 Demo

The online demo takes as input a suspicious image. The goal is to firstly classify the image as forged or pristine and, in the former case, to localize the forgery. The user is required to select the size of the patches, being 128×128 and 256×256 the two available options. The user can also decide the overlap with which these patches will be taken (50% or 75% of the patch size). Finally, the user can choose whether to use spectral clustering (Section 2.3.1) or modularity optimization (Section 2.3.2).

The demo prints the classification as tampered or non-tampered, according to the eigengap if spectral clustering is the chosen clustering algorithm, or according to the optimal modularity in case modularity optimization is chosen. The thresholds used to decide between tampered and non-tampered are those used by Mayer and Stamm, as explained in Section 2.3. If the image is classified as non-forged, a black mask is shown as output. On the other hand, if the image is classified as tampered, forgery localization is performed according to the chosen clustering technique, and the output mask shows the obtained results.

4 Experiments

4.1 Forged Images

We first test the proposed approach on forged images. In these cases, the method should manage to identify the image as forged, and then perform forgery localization. Figure 2 shows the results obtained on some images from [11], for both clustering techniques and both patch sizes. In all these experiments, the patch overlap was set to 75%.

We observe that in most cases the method is able to detect the images as forged, except for one of the examples when using the spectral clustering technique and blocks of size 128×128. Interestingly, the *eigengap* seems to be very sensitive to the patch size, with significant lower values for larger patches. On the other hand, the optimal modularity seems to be more stable.

Regarding forgery localization, both techniques show significant agreement between the ground truth mask and the estimated mask. However, modularity optimization shows some falsely detected regions that are not present when using spectral clustering.

4.2 Authentic Images

In these experiments we assess the performance of the studied approach when the input image is not tampered. Figure 3 shows examples of the obtained results for several authentic images from [11].

The method is able to correctly identify the images as non-tampered in most cases. However, spectral clustering, when patches are of size 256×256, delivers some false detections, as well as modularity optimization when used with 128×128 patches. These false detections seem to correspond to image regions having a distinctive texture. However, not all textured zones generate false alarms, as can be seen in the second image where none of the clustering techniques have false detections, despite the fact that the image has textured and flat zones.

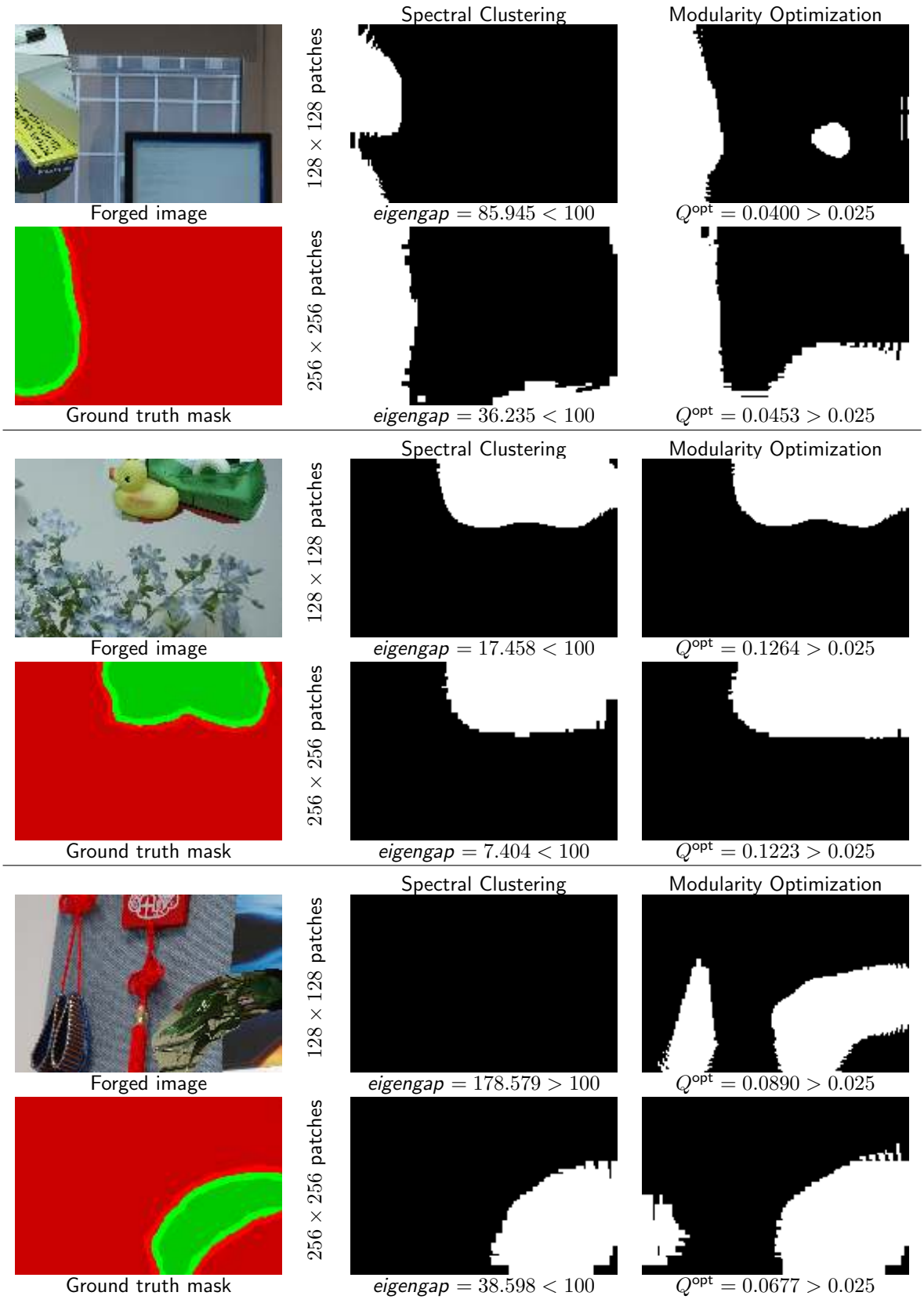


Figure 2: Results obtained using the demo for some spliced images from the Columbia dataset [11]. The overlap is set to 75% of the patch size in all cases.

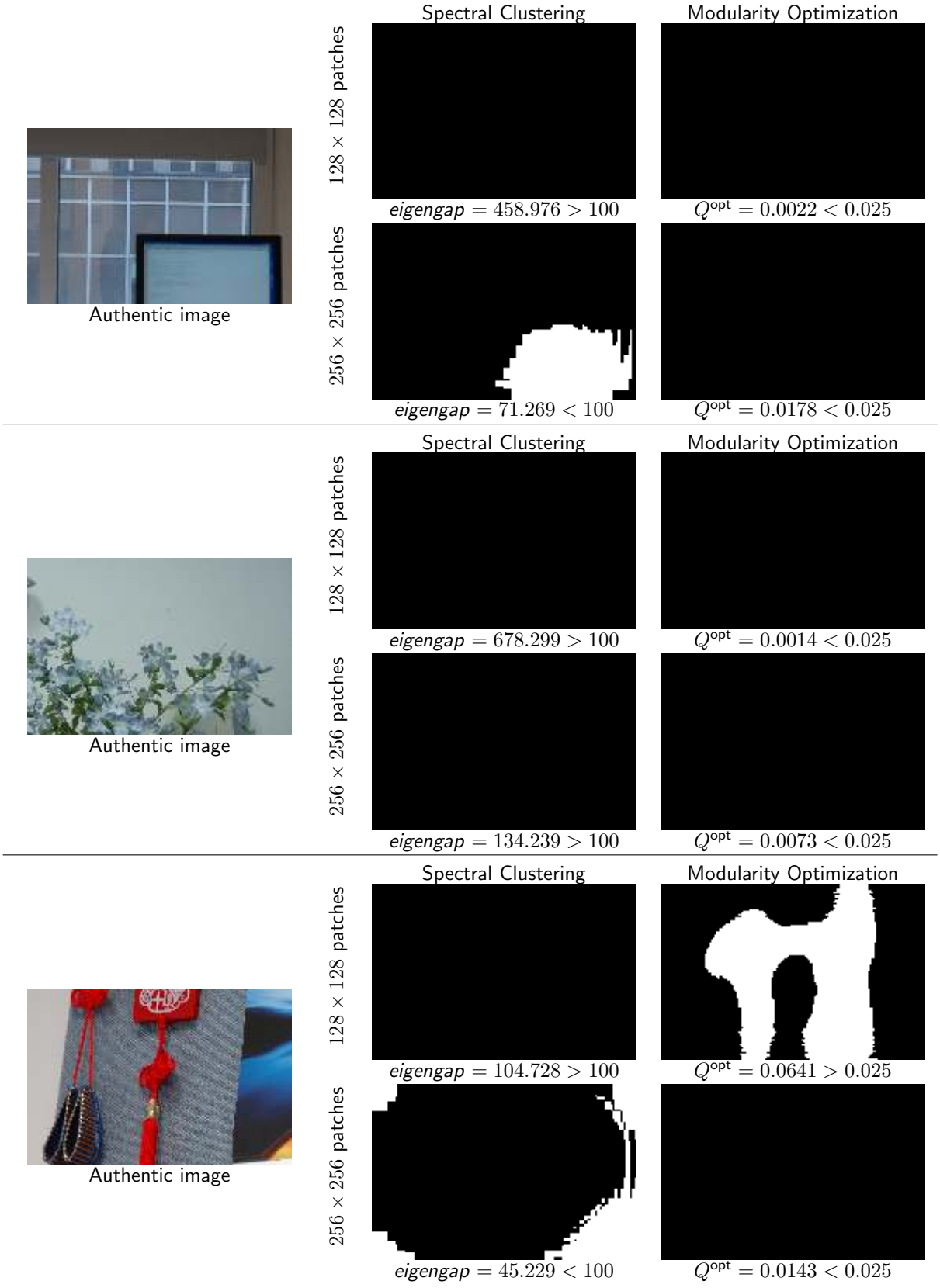
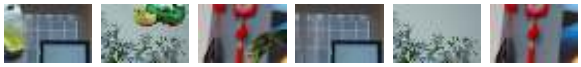


Figure 3: Results obtained using the demo for some authentic images from the Columbia dataset [11]. The overlap is set to 75% of the patch size in all cases.

Acknowledgment

This work has received funding by the Paris Region Ph.D. grant from Région Île-de-France, the ANR-DGA challenge DEFALS (ANR-16-DEFA-0004) and the European Union under the Horizon Europe vera.ai project, Grant Agreement number 101070093.

Image Credits

 Columbia Dataset [11]

 MISD Dataset [13]

References

[1] Q. Bammey, R. Grompone von Gioi, and J-M. Morel, *An adaptive neural network for unsupervised mosaic consistency analysis in image forensics*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020. https://doi.org/10.1109/CVPR42600.2020.01420.

[2] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E.J. Delp, and S. Tubaro, *Tampering Detection and Localization Through Clustering of Camera-Based CNN Features*, in IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 1855–1864. https://doi.org/10.1109/CVPRW.2017.232.

[3] C-H. Choi, J-H. Choi, and H-K. Lee, *CFA Pattern Identification of Digital Cameras Using Intermediate Value Counting*, in ACM Multimedia Workshop on Multimedia and Security, Association for Computing Machinery, 2011, p. 2126. https://doi.org/10.1145/2037252.2037258.

[4] A. Clauset, M.E.J. Newman, and C. Moore, *Finding community structure in very large networks*, Physical Review E, 70 (2004), p. 066111. https://doi.org/10.1103/PhysRevE.70.066111.

[5] D. Cozzolino, G. Poggi, and L. Verdoliva, *Splicebuster: A new blind image splicing detector*, in IEEE International Workshop on Information Forensics and Security (WIFS), 2015. https://doi.org/10.1109/WIFS.2015.7368565.

[6] D. Cozzolino and L. Verdoliva, *Noiseprint: A CNN-Based Camera Model Fingerprint*, IEEE Transactions on Information Forensics and Security, 15 (2020), pp. 144–159. https://doi.org/10.1109/TIFS.2019.2916364.

[7] M. Fiedler, *Algebraic connectivity of graphs*, Czechoslovak Mathematical Journal, 23 (1973), pp. 298–305. http://eudml.org/doc/12723.

[8] S. Fortunato, *Community detection in graphs*, Physics Reports, 486 (2010), pp. 75–174. https://doi.org/10.1016/j.physrep.2009.11.002.

[9] M. Gardella and P. Musé, *Forensic similarity for source camera model comparison*, Image Processing On Line (IPOL), (2022). https://doi.org/10.5201/ipol.2022.424.

[10] M. Gardella, P. Musé, J-M. Morel, and M. Colom, *Noisesniffer: a fully automatic image forgery detector based on noise analysis*, in IEEE International Workshop on Biometrics and Forensics (IWBF), 2021, pp. 1–6. https://doi.org/10.1109/IWBF50991.2021.9465095.

[11] Y.-F. Hsu and S.-F. Chang, *Detecting image splicing using geometry invariants and camera characteristics consistency*, in IEEE International Conference on Multimedia and Expo (ICME), 2006. https://doi.org/10.1109/ICME.2006.262447.

[12] M. Huh, A. Liu, A. Owens, and A.A. Efros, *Fighting fake news: Image splice detection via learned self-consistency*, in European Conference on Computer Vision (ECCV), 2018. https://doi.org/10.1007/978-3-030-01252-6_7.

[13] K.D. Kadam, S. Ahirrao, and K. Kotecha, *Multiple Image Splicing Dataset (MISD): A Dataset for Multiple Splicing*, Data, 6 (2021). https://doi.org/10.3390/data6100102.

[14] U. Luxburg, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), p. 395416. https://doi.org/10.1007/s11222-007-9033-z.

[15] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, *A Full-Image Full-Resolution End-to-End-Trainable CNN Framework for Image Forgery Detection*, IEEE Access, (2020). https://doi.org/10.1109/ACCESS.2020.3009877.

[16] O. Mayer and M.C. Stamm, *Exposing fake images with forensic similarity graphs*, IEEE Journal of Selected Topics in Signal Processing, 14 (2020), pp. 1049–1064. https://doi.org/10.1109/JSTSP.2020.3001516.

[17] O. Mayer and M. C. Stamm, *Forensic similarity for digital images*, IEEE Transactions on Information Forensics and Security, (2019). https://doi.org/10.1109/TIFS.2019.2924552.

[18] M.E.J. Newman, *Finding community structure in networks using the eigenvectors of matrices*, Physical Review E, 74 (2006), p. 036104. https://doi.org/10.1103/PhysRevE.74.036104.

[19] M. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Physical Review E, 69 (2004), p. 026113. https://doi.org/10.1103/PhysRevE.69.026113.

[20] T. Nikoukhah, J. Anger, T. Ehret, M. Colom, J-M. Morel, and R. Grompone von Gioi, *JPEG grid detection based on the number of dct zeros and its application to automatic and localized forgery detection*, in IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 110–118.

[21] A.C. Popescu and H. Farid, *Statistical tools for digital forensics*, in International Workshop on Information Hiding, 2004. https://doi.org/10.1007/978-3-540-30114-1_10.

[22] R. Salloum, Y. Ren, and C.-C.Jay Kuo, *Image Splicing Localization Using A Multi-Task Fully Convolutional Network (MFCN)*, CoRR, abs/1709.02016 (2017). https://doi.org/10.48550/arXiv.1709.02016.

[23] Y. Wu, W. AbdAlmageed, and P. Natarajan, *ManTra-Net: Manipulation Tracing Network for Detection and Localization of Image Forgeries With Anomalous Features*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. https://doi.org/10.1109/CVPR.2019.00977.

[24] S. Ye, Q. Sun, and E-C. Chang, *Detecting digital image forgeries by measuring inconsistencies of blocking artifact*, in IEEE International Conference on Multimedia and Expo (ICME), 2007, pp. 12–15. https://doi.org/10.1109/ICME.2007.4284574.