Accepted in Digital Investigation, https://doi.org/10.1016/j.diin.2017.08.005

Clustering Image Noise Patterns by Embedding and Visualization for Common Source Camera Detection

Sonja Georgievska[a,*], Rena Bakhshi[a], Anand Gavai[a], Alessio Sclocco[a], Ben van Werkhoven[a]

[a]*Netherlands eScience Center, Science Park 140, Amsterdam, The Netherlands*

Abstract

We consider the problem of *clustering* a large set of images based on similarities of their noise patterns. Such clustering is necessary in forensic cases in which detection of common source of images is required, when the cameras are not physically available. We propose a novel method for clustering combining low dimensional embedding, visualization, and classical clustering of the dataset based on the similarity scores. We evaluate our method on the Dresden images database showing that the methodology is highly effective.

Keywords: Clustering; digital camera identification; common image source detection; Photo-response non-uniformity; Digital forensics

1. Introduction

Common source identification of digital photographs can play an important role in digital investigations. The identification problem exists because the meta-data accompanying the image can be easily altered by the creators to remove traces of its origin. Nevertheless, it has been found that small deficiencies in the imaging sensor of a camera leads to detectable noise in the image, so-called Photo-Response Non-Uniformity (PRNU) patterns (Lukas et al., 2006), which provides a signature that can be used to identify the source of an image in a robust manner. When a suspect camera is present, its PRNU fingerprint can be estimated from the set of images taken by it. Then, the fingerprint can be used on images to determine whether they originated from the corresponding camera.

However, many forensic investigations deal with large collections of images, without a suspect camera available. In absence of the fingerprint information, clustering the images based on similarities of their PRNU patterns becomes important, because it can lead to a suspect by indicating which images originate from the same source camera. Correct clustering with respect to the source cameras is the scope of our work.

In fact, extensive research on clustering PRNU patterns has already been done (Bloy, 2008; Li, 2010; Caldelli et al., 2010; Gisolf et al., 2014; Amerini et al., 2014; Fahmy, 2015; Lin & Li, 2016). The initial approaches (Li,

*Corresponding author

Email addresses: s.georgievska@esciencecenter.nl (Sonja Georgievska), r.bakhshi@esciencecenter.nl (Rena Bakhshi), a.gavai@esciencecenter.nl (Anand Gavai), a.sclocco@esciencecenter.nl (Alessio Sclocco), b.vanwerkhoven@esciencecenter.nl (Ben van Werkhoven)

2010; Caldelli et al., 2010) exploit a subset of the entire set of images as a *training set*, which is first clustered, and then the authors perform classification on the rest of the dataset. However, as also elaborated by Lin & Li (2016), if the number of source cameras is larger than the average number of images per camera, the entire dataset might be underrepresented by the training set, thereby limiting the scalability of this approach. Many proposals, usually for hierarchical clustering, require a pre-defined similarity threshold parameter, that has been tuned on a benchmarking dataset, or that needs prior information about the dataset. More concretely, the method by Li (2010) requires a pre-defined similarity threshold, that determines when a new cluster is formed – a drawback also pointed by the same author in Lin & Li (2016). A similar parameter is also required in Bloy (2008), which varies for different camera models, thus limiting its applicability in new environments. To that end, Lin & Li (2016) propose a methodology for clustering requiring a preset threshold (i.e., minimal cluster size) which is ideally based on the average cluster size – information which is not always available in practice. (In addition, pre-elimination of the saturated images is required.) Amerini et al. (2014) tune the threshold parameter on the benchmarking dataset, while Gisolf et al. (2014) provide for way to set their threshold based on the dataset at hand. However, it is unclear how the performance obtained with thresholds tuned on the experimental datasets in Gisolf et al. (2014) would generalize for completely new datasets with varying numbers and sizes of clusters or with new camera models. Therefore, there is a need to remove dependence on thresholds that have been optimized on a benchmarking dataset or that require prior information.

There is also significant room for improving the performance of clustering PRNU patterns itself. Namely, Lin & Li (2016) report precision rate of at least 98%, however the recall is between 60% and 74%. On a real case, Gisolf et al. (2014) report a true positive rate (TPR) of 89%, at a zero false positive rate (FPR); Amerini et al. (2014) reports FPR of at most 4%, but the TPR is between 79% and 92%. Fahmy (2015), the only later method that does not require a pre-defined threshold, reports TPR values of 95% (and FPR of 1%) on (relatively small) datasets with 5 cameras and 100 images per dataset.

Clearly, correct clustering of images to reveal common source, in absence of any information about the cameras, is a challenging task. Firstly, validation of the clustering is an issue: in a lab setting it can be evaluated on a benchmarking dataset that includes the cameras information. However, in reality this information is not present, and the evaluation is based on metrics comparing the inter-cluster and intra-cluster distances. Secondly, existing clustering algorithms often need parameters that require prior information or that are dataset specific.

In this paper, we propose a novel method for clustering image PRNU noise patterns which reduces the complexity and increases the confidence of the investigator in the resulting clustering. The main idea is to enable the user to *see and explore* the (potential) clusters, and to utilize their domain expertise in the clustering process. The method that we propose does not require a parameter that has been optimized on a benchmarking dataset or that requires prior information. Finally, we show that it is superior to the existing methods with respect to the clustering performance.

The rest of this paper is structured as follows. Section 2.1 provides background information on the technologies that we use in our method. More concretely, it briefly explains how PRNU patterns and PCE similarity scores are obtained, and provides a brief introduction to the techniques for embedding and visualization. The method that we

propose is presented in Section 3. In Section 4, we evaluate the proposed method on the Dresden Images database and compare its approach and performances to those reported in previous work. Finally, section 6 concludes the paper.

2. Underpinning technologies, methods and tools

2.1. PRNU patterns and PCE similarity scores computations on GPU

The Peak-to-Correlation Energy (PCE) similarity scores of PRNU patterns that we use in this paper, are pre-computed by an application developed in earlier work by van Werkhoven (Unpublished results). In particular, the application uses Graphics Processing Units (GPUs) to extract the PRNU patterns from large sets of images as well as to compute the all-to-all PCE scores within a reasonable timeframe.

The implementation of the PRNU extraction largely follows the procedure presented by Gisolf et al. (2014). Important differences with Gisolf et al. are that the digestion and quantization steps are left out, as these trade accuracy for performance. The image is first converted to grayscale. The initial estimate of the PRNU pattern is obtained using the First Step Total Variation (FSTV) algorithm (Gisolf et al., 2013). After that Zero Mean and Wiener filtering steps are performed to filter out any artifacts produced by color interpolation, on-sensor signal transfer, imaging sensor design, and JPEG compression (as proposed by Chen et al. (2008)).

Peak to Correlation Energy (PCE) ratio is a frequently used algorithm for comparing PRNU patterns (Chuang et al., 2011; Fridrich, 2009; Bayram et al., 2015). PCE is computed as the ratio between the height of the peak and the energy of the cross correlation between two PRNU patterns. Goljan (2008) have shown that PCE is a much more suitable detection measure, compared to the another frequently used Normalized Cross Correlation (NCC), for the related problem of camera identification. This is because periodic signals such as linear patterns in two PRNU patterns will increase their correlation, while reducing the PCE score.

2.2. Visualization by embedding

The added value of data visualization in statistical analysis has been first shown in Anscombe (1973). In order to demonstrate both the importance of visualizing data before analyzing it and the effect of outliers on statistical properties, Anscombe constructed four two-dimensional scatter plots that had identical descriptive statistics, whereas their graphs (and, thus, the corresponding datasets) were completely different. (The scatter plots are nowadays known as the Anscombe's quartet (Saville & Wood, 1991; web, a).)

Visualization of data represented by pairwise similarities, such as image noise patterns with their PCE scores, is more challenging. However, a (sparse) similarity matrix can be used not only as input to a clustering algorithm but also as input to *embedding* algorithms (Tenenbaum et al., 2000; Roweis & Saul, 2000). The embedding algorithms try to "embed" the dataset in a low dimensional vector space, such that the original distance between each pair of data points (e.g. noise patterns) is preserved as much as possible in the low dimensional space. Since not all distances can be preserved[1], the question of which distances to prioritize has arisen. In the recent proposals (van der Maaten, 2014;

[1]For example, the corner points of an equilateral triangle cannot be embedded in one dimensional space such that all three distances remain the same.

Tang et al., 2016b) higher similarities (and thus smaller distances) are assigned higher priorities, thereby preserving the intrinsic cluster structure of the dataset, if it exists, as much as possible on many levels. This means that close (or far away) points in the low dimensional space are also close (or far apart) in the original space. For distant points, however, the distance does not need to be exactly preserved, as long as it is relatively large enough (van der Maaten, 2014; Tang et al., 2016b).

In this paper we employ LargeVis (Tang et al., 2016b), a promising algorithm for embedding large datasets (in the order of millions of elements). The time complexity of the optimization with respect to the size of the dataset (i.e. number of images) is linear, and the input matrix can be sparse, e.g. discarding all similarity scores that are low and thus not interesting. The algorithm first constructs an accurately approximated k-nearest neighbors (KNN) graph from the data, using the similarity matrix, and then layouts the graph in the low-dimensional space through an optimization procedure: To construct the KNN graph, the KNN of a point are found by partitioning the data space in a tree-like manner. Once the KNN graph is built, an objective optimization function for low-dimensional mapping of the graph is created (having the edges as constraints), that assigns weight to every edge proportional to the similarity between the edge nodes. Unobserved edges have negative weights. In this way similar points in the original data space stay close to each other in the low dimensional space, and dissimilar points tend to be far away from each other. The optimization is performed using asynchronous stochastic gradient descent with a linear time complexity. Tang et al. (2016b) demonstrate that the hyper-parameters of LargeVis are stable over different datasets, i.e. that in practice parameter tuning is not required in order to obtain accurate KNN graph construction and embedding. (Accuracy of a KNN graph is defined as the percentage of data points that are truly KNN of a node, while accuracy of an embedding is evaluated by using a KNN classifier to classify the benchmarking datasets – provided with labels – based on their low-dimensional representations. The intuition of this evaluation methodology is that a good embedding should be able to preserve the structure of the original data as much as possible, and hence yield a high classification accuracy with the low-dimensional representations.)

2.3. Interactive visualisation of the Dresden image database

We have embedded the most interesting datasets of the Dresden image database (Gloe & Böhme, 2010) using the PCE similarity scores (Goljan, 2008) as input and using the default parameters of LargeVis (Tang et al., 2016a). Here, by a *dataset* we mean a set of images of the same size, and we consider a dataset to be interesting if the images originate from many cameras (devices). Table 1 gives overview of the datasets, their names, and the corresponding image sizes. For example, the Pentax dataset consists of all images of size (resolution) 4000×3000 pixels, the Canon dataset consists of all images of size 3072×2304, and so on. For the Fuji dataset, however, we are rather interested if the embedding confirms the artifacts observed by Gloe et al. (2012), namely, the images originating from the FujiFilm J50 and Casio EX-Z150 cameras may undergo additional post-processing to suppress the image noise. These artifacts influence camera detection based on PRNU fingerprints.

Table 1: Images from the Dresden database used for embedding and clustering.

Properties \ Name of dataset	Pentax	Canon	Praktica	Olympus	Fuji
Resolution(pixels)	4000 × 3000	3072 × 2304	2560 × 1920	3648 × 2736	3264 × 2448
Number of models	1	4	1	7	3
Number of devices	4	10	5	24	9
Number of images	638	2123	1019	4980	1918

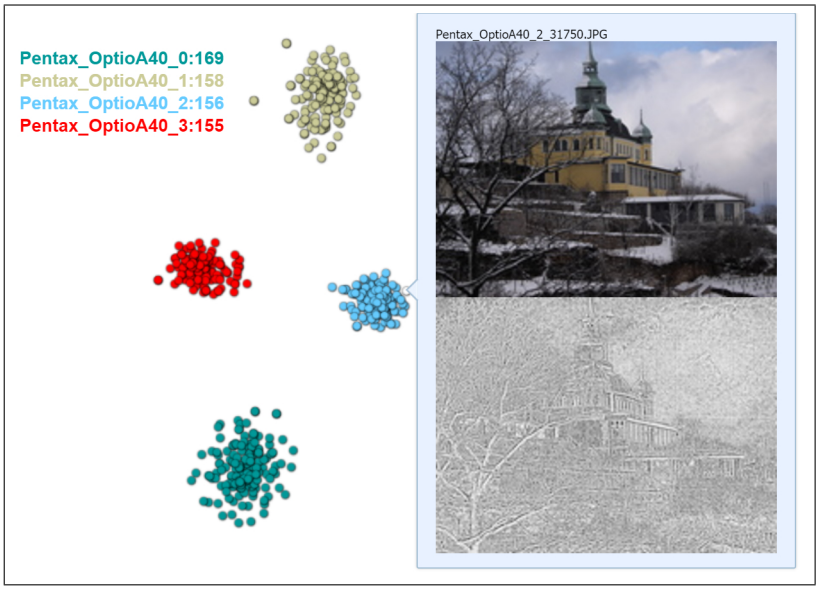

Figure 1: The Pentax dataset as embedded by LargeVis in 3D with default parameters and PCE similarity scores. Points are colored by their source camera. The list of source cameras along with number of images per camera is in the upperleft corner. The image shows an original photo and its PRNU pattern, the latter being post-processed to highlight the image leaking into the pattern.

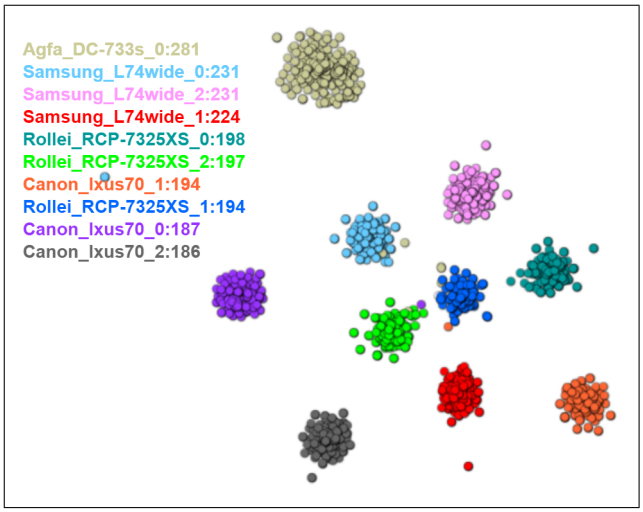

Figure 2: The Canon dataset as embedded by LargeVis in 2D with default parameters and PCE similarity scores. Points are colored by their source camera.

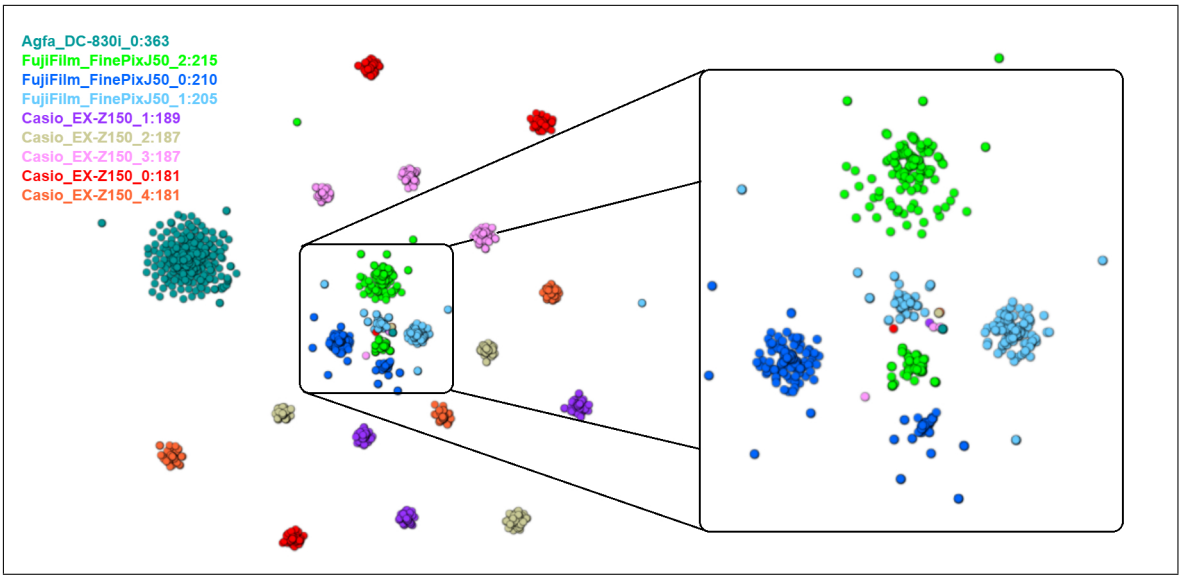

Figure 3: The Fuji dataset as embedded by LargeVis with default parameters and PCE similarity scores. Points are colored by their source camera.

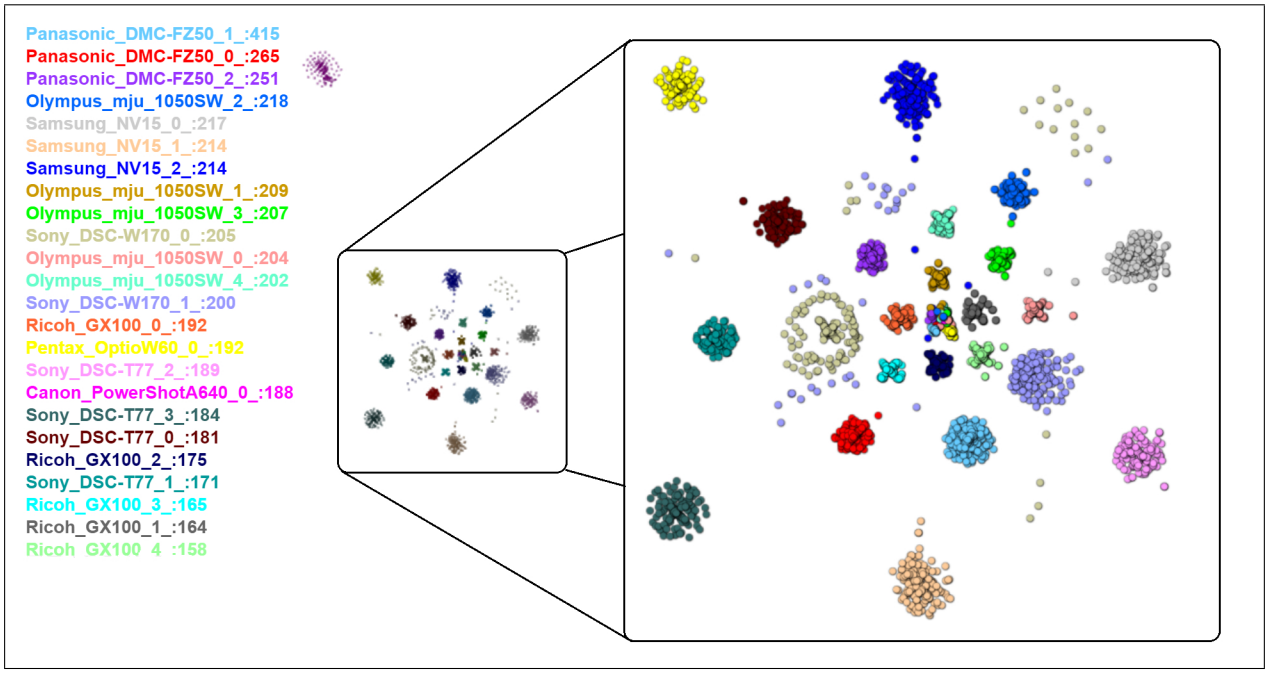

Figure 4: The Olympus dataset as embedded by LargeVis with default parameters and PCE similarity scores. Points are colored by their source camera.

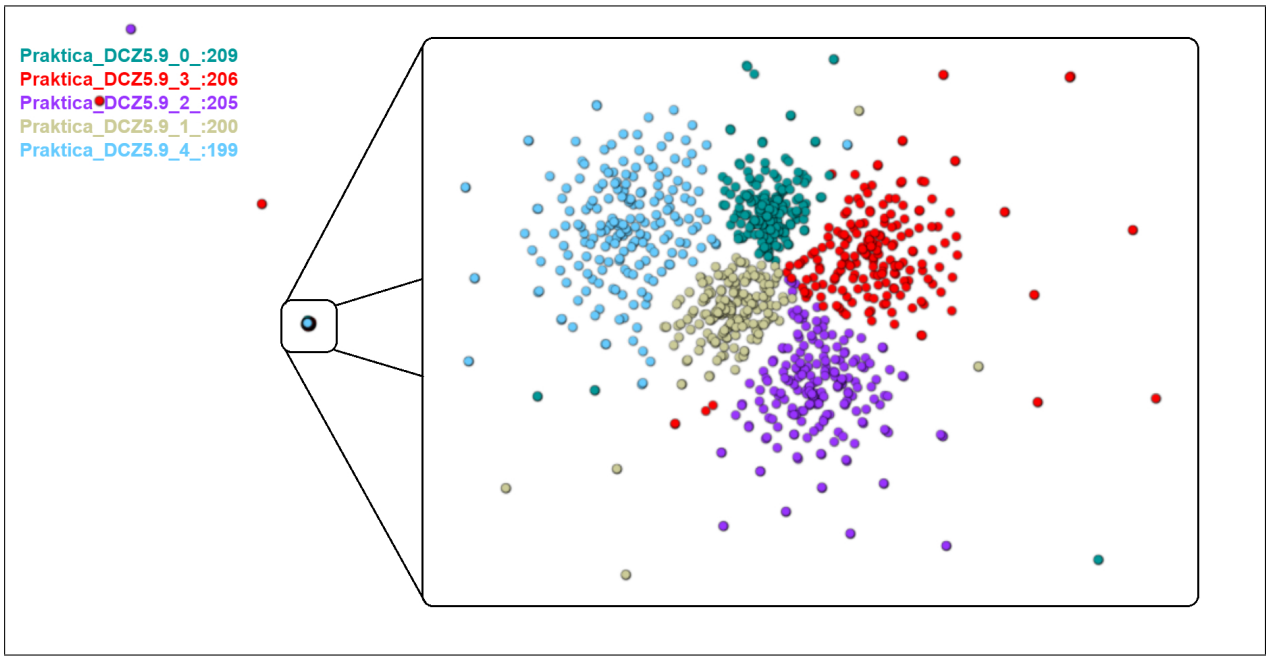

Figure 5: The Praktica dataset as embedded by LargeVis with default parameters and PCE similarity scores. Points are colored by their source camera.

The results of the embedding are presented in Figures 1– 5. In the figures, each point (image) is colored by its ground truth, namely, the camera it was taken with. (The left part of each figure contains a table of (colored) camera names and number of images per camera.) Note that the ground truth was not used in the embedding process, only the similarity matrix per dataset. We use the 3D interactive viewer SherlockDive (Georgievska, 2017b), an adaptation of DiVE (Georgievska, 2017a), to visualize a large number of points on a screen (with good interactivity for up to a million points). The Pentax dataset as in Fig.1 can be directly inspected at web (b), and a user interaction manual for SherlockDive is provided at web (c). Interactive visualization where one can inspect an individual point by mouse hovering, is important for embedded data, because there is no easy way to identify a point by its coordinates (the latter is just a low dimensional mapping of the original high dimensional point). When a user (investigator) is able to inspect the images with their noise patterns individually, not only can she detect the outliers, but also quickly determine the reason an image is far away from any cluster, e.g., because it is saturated. In addition, the interactive viewer allows a user to color and search the data space by any provided meta-data. Later we will see how the interactivity also aids the clustering.

3. The proposed methodology for clustering

We now provide the motivation and description of our method for clustering images based on PRNU noise pattern similarity. Our motivation for developing a method that combines embedding, visualization, and clustering, is as follows:

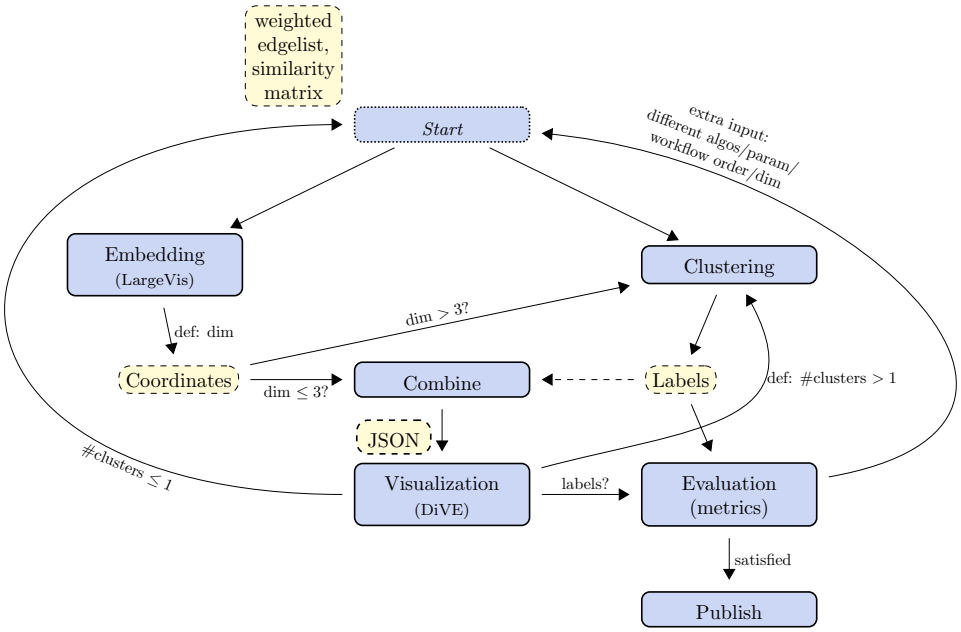

Figure 6: The proposed workflow

- Visualization, as shown by Anscombe's example in Sec. 2.2, can provide insights that are beyond simple clustering statistics of the dataset such as metrics based on inter-cluster versus intra-cluster distances, especially in presence of outliers. Interactive visualization allows one to spot potential outliers easily and to filter the dataset before the clustering process. This prevents that the outliers affect the clustering analysis.

- In the original non-Euclidean vector space (of the noise patterns) it is not straightforward to compute the mean (average) vector of a set of vectors, in order to apply clustering algorithms that require real-time computation of mean (like *k*-means, mean shift, Gaussian mixture model). Definition of a *mean* pattern is a matter of research itself (Lukas et al., 2006; Bloy, 2008; Bayram et al., 2015), but even if the definition is given a priori, it is computationally expensive to re-compute the means on-the-fly during clustering. Embedding the noise patterns in a low dimensional space transforms the original data space into an Euclidean space, and clustering can be applied on the embedded data[2].

- Embedding in combination with visualization provides also a way to visually estimate the number of clusters to use as input for the previously mentioned clustering algorithms. In fact, embedding can be seen as a non-linear feature extraction technique similar to auto-encoders (Hinton & Salakhutdinov, 2006), or as a non-linear counterpart of Principal Component Analysis. Thus, it can be used as a pre-processing step of any clustering algorithm.

[2]Note that embedding can be done in more than three dimensions; however, it is not recommended to use Euclidean metric in high (greater than 8) dimensional space due to the curse of dimensionality (Aggarwal et al., 2001; Keogh & Mueen, 2010).

Table 2: Performance results of clustering. The asterics (*) denotes the confirmed observations made by Gloe et al. (2012).

Metric \ Dataset	Pentax	Canon	Praktica	Olympus	Fuji
Adjusted Rand index	1.0	0.99	0.96	0.88	0.72*
Completeness	1.0	0.99	0.94	0.93	0.73*
Homogeneity	1.0	0.99	0.97	0.95	0.98
Mutual information	1.0	0.99	0.94	0.93	0.73*

- Visualization and clustering can be performed independently in order to cross-validate each other, by coloring points based on cluster labels.

The main steps of our method are summarized in Fig. 6. The classical way of clustering is incorporated in the right branch of *Start* → *Clustering* → *Labels* → *Evaluation* → *Publish performance metrics (or repeat from start)*.

Normally the clustering would stop when the performance show a good separation of the obtained clusters (by, e.g., comparing the inter-cluster and intra-cluster distances). The ability to embed data in a low dimensional space and visualize it (if the number of dimensions is at most three), however, adds new possibilities. One can start by first embedding the data in two or three dimensions. Then, if the visualization clearly shows (Gaussian) clusters, as in Figures 1 – 4 (disregarding the colors), the number of clusters can be visually determined and clustering with, e.g. k-means clustering, can be directly applied on the embedded data. Such a scenario corresponds to the unfolded branch *Start* → *Embedding* $\xrightarrow{def:dim}$ *Coordinates* $\xrightarrow{dim\leq 3}$ *Combine* $\xrightarrow{JSON}$ *Visualization* $\xrightarrow{def:\ \#clusters > 1}$ *Clustering* → *Labels* → *Combine* → *Visualization* → *Evaluation* → *Publish performance metrics*. However, if the preliminary visualization shows clusters that are visually separated, but not so clearly separated for k-means clustering (as shown in Fig. 5, disregarding the colors), one can embed the original similarity matrix in a higher dimensional space that allows for data to "spread" better, and then cluster the data in that space with k-means (where k is determined by the visualization). This scenario corresponds to diverging to *Evaluation* $\xrightarrow{extra\ input:\ param/dim}$ *Start* → *Embedding* $\xrightarrow{def:dim}$ *Coordinates* $\xrightarrow{dim>3}$ *Clustering* → *Labels* → *Combine* $\xrightarrow{JSON}$ *Visualization* → *Evaluation* after the initial 2D visualization. If the initial visualization showed clusters that are not clearly separated (#clusters ≤ 1), one can still embed the dataset in a higher dimensional space and apply, e.g., mean shift clustering (or conclude that there is only one cluster if it is a well formed blob).

Regardless in which space clustering is performed, some artifacts of the clustering can be detected from the visualization itself. For example, Fig. 9 shows that a (visually) big cluster is artificially split in five clusters. So, the visualization can further navigate the clustering process, and suggest to the user that those five clusters are essentially one cluster. Note that we present only a few interesting examples of data visualization to illustrate how the data clustering can be guided. In the following section we will discuss the clustering of the datasets presented in Table 1 in more detail.

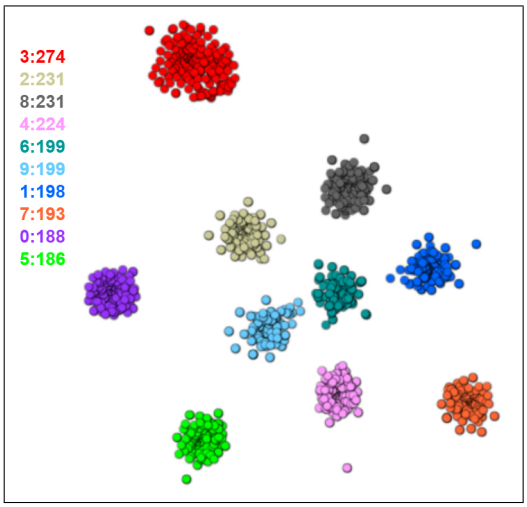

Figure 7: The Canon dataset, embedded in 2D. Colored by k-means clusters ($k = 10$), clustering performed after embedding.

4. Evaluation

We have applied our method on the datasets from Table 1. We choose to use the images without cropping to ensure easy reproducibility and comparability of the results. For clustering and evaluating with ground truth we employ Clustit (van Werkhoven et al., 2017), a Python tool that incorporates the algorithms implemented in scikit-learn (Pedregosa et al., 2011) and SciPy.

As performance metrics we utilize the scikit-learn implementations of adjusted Rand score (Hubert & Arabie, 1985), mutual information (Strehl & Ghosh, 2003), and homogeneity and completeness (Rosenberg & Hirschberg, 2007), using the ground-truth labels for comparison (we refer the reader to Rosenberg & Hirschberg (2007) and Amigó et al. (2009) for more information on various metrics). Intuitively, a clustering is homogeneous when every cluster contains only images from the same camera. It is complete if all images from a single camera are in the same cluster. The Adjusted Rand index computes how similar the clusters are to the ground truth, i.e. it is a measure of the percentage of correct decisions made by the algorithm. The mutual information is an information theoretic measure of how much information is shared between a clustering and a ground-truth classification. Table 2 summarizes the performances of our clustering for the five datasets of the Dresden images database. As mentioned earlier in Section 2.2, for the Fiju dataset we are interested in confirming the observations made by Gloe et al. (2012) rather than in the performance of clustering itself.

As explained before, Figures 1 – 5 show the embedding and corresponding visualization of those datasets, where the coloring corresponds to the ground truth. The embeddings were performed with the default parameters of the LargeVis implementation in Tang et al. (2016a), which targets datasets of millions of points. [3] (Of course, this is done

[3] In this respect, we note that for datasets of size a couple of thousands the input parameter $-samples$ can be set to a much smaller value, say 100, to obtain same quality embedding at much lower computational costs.

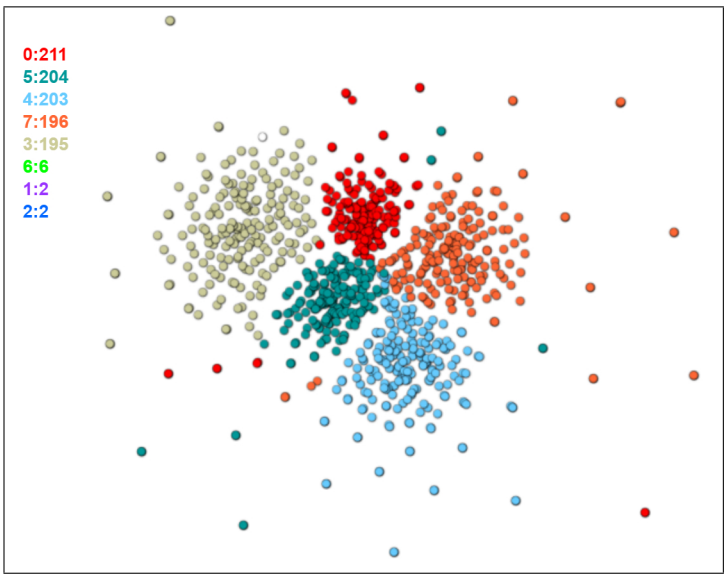

Figure 8: The most of the Praktica dataset, embedded in 2D. Coloring based on clustering with k-means ($k = 8$) after embedding in 8 dimensions. Points shown here correspond to the bottom-most cluster in Fig. 5

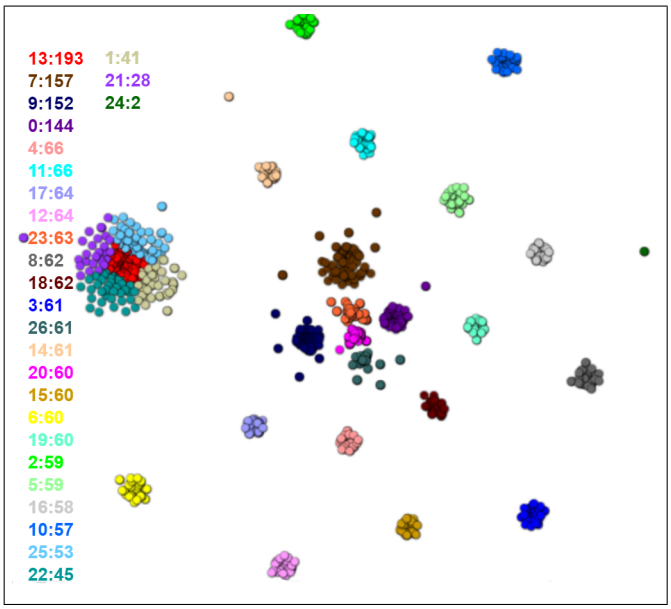

Figure 9: The Fuji dataset embedded in 2D. Coloring based on clustering with k-means ($k = 27$); after clustering clusters 1, 22, 13, 25, and 21 from the large blob have been merged to take into account the insight from visualization.

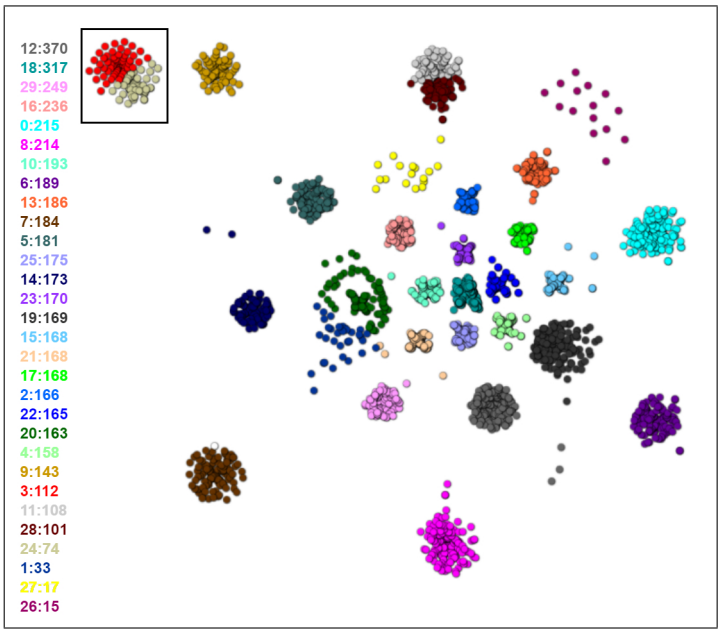

Figure 10: The Olympus dataset embedded in 2D, zoomed-in. (The original relative position of the upperlefmost blob in a box is as in Fig. 4) Coloring based on clustering with k-means ($k = 30$); after clustering, to take into account the insight from visualization, the following couples of clusters have been merged: clusters 24 and 3 (upperleft blob in a box), 11 and 28 (the blob with grey and dark red points) and 1 and 20 (the green-blue rings-shaped blob).

by specifying the number of dimensions to embed in and using a similarities graph, i.e. network, as an input type rather than high dimensional vectors.)

We can see that the embedding provides clear split into four clusters for the Pentax dataset (see Fig. 1), and indeed, the k-means algorithm[4] with four clusters applied afterwards gives a perfect score (see Table 2). Furthermore, the embedding results in almost clear split of the Canon dataset into 10 clusters (see Fig. 2), and thus, the k-means algorithm with 10 clusters applied afterwards, gives almost-perfect scores (see Fig. 7 and Table 2).

The Praktica dataset is more challenging. From the embedding of the entire dataset we notice four clusters initially. After careful inspection and zooming-in, we detect that most of the images are hidden (embedded) in the bottom-most cluster in Fig. 5, and that there are only six images in the other three clusters. By zooming-in enough, the points hidden in the square reveal an interesting structure, as shown in Fig. 5. Here, we notice five sub-clusters in the zoomed-in region. Together with the four clusters in the upper level of the hierarchy, we conclude that there are $(4 - 1) + 5 = 8$ clusters in total (note that we are not using the ground truth color in the reasoning). Thus, we use k-means with $k = 8$. However, because the clusters in the middle of Fig. 5 are not well separated for k-means, we embed the original dataset in eight dimensions and apply k-means with $k = 8$ (even though in reality five cameras were used to produce the dataset). We choose eight dimensions because in more than 8 dimensions the Euclidean distance acts counter-intuitively (Keogh & Mueen, 2010).

As for the Fuji dataset, from the embedding in Fig. 3 we spot 22 clusters (disregarding colors). Applying k-means

[4]We use the scikit-learn implementation.

with $k = 22$ splits the Agfa subset (see Fig. 3) into two clusters as well as merges the three small clusters in the center of the figure into one cluster. To make use of the visual insights, we apply k-means with $k = 27$. This results in the clustering presented in Fig. 9, and we merge clusters 1, 22, 13, 25, and 21 into one cluster (if the colors are not clearly distinguishable, the detection of the cluster labels that need to be unified can be easily done by hovering with the mouse over the points in the interactive viewer). Similar reasoning is applied for the Olympus dataset (see Fig. 10). It is likely that some algorithm other than k-means (for example, Gaussian mixtures) would have directly led us to clustering where merging is unnecessary. Finding an algorithm that fits best with this dataset is, however, out of scope of this paper. We use k-means presently because it only requires the parameter k which can be inferred from the visualization.

As we see from the results in Table 2, clustering on all datasets other than Fuji is highly effective with respect to the ground truth. For the Fuji dataset, we confirmed the artifacts discovered by Gloe et al. (2012). However, clustering even in presence of the dataset artifacts, is still highly homogeneous (images from different cameras are rarely assigned to the same cluster), which is important in forensics to avoid wrong suspects.

5. Performance comparison

In Sec. 1 we mentioned briefly the performance measures reported by related work on clustering PRNU patterns. Here we will make a more detailed comparison to the performance measures of our method. First, let us note that all approaches use different performance measures – for example, the definitions of TPR and FPR in Gisolf et al. (2014) and Amerini et al. (2014) are different. Nevertheless, all performance measures can be aggregated into those that indicate likelihood that the clusters do not contain images from different cameras (homogeneity, 100% – FPR, precision) and those that indicate likelihood that all images from a camera are contained within a single cluster (completeness, TPR, recall). The other measures – F-score, adjusted Rand, mutual information – are combinations of the above two types. To start with Lin & Li (2016), their precision is high (98% -100%), but the recall rate is 71%-74% for the so-called easy datasets and 60%-72% for the difficult datasets, where the number of clusters is bigger than the cluster size (50 clusters and 10-30 images, respectively). On datasets with several thousand images, 100 images per camera, the recall rate is 55%-76%. In comparison, our Olympus dataset with 4980 images and 24 cameras had completeness of 93% at homogeneity of 95%. On three datasets with 4–6 cameras and 24–63 images per camera, Amerini et al. (2014) report TPR 80%-92% at FPR 0%-4%. Such datasets are comparable to our Pentax and Praktica datasets in terms of number of cameras (although the number of images is much higher presently), with completeness 94%-100% and homogeneity 97%-100%. Similarly, Fahmy (2015) reports TPR of 95% at FPR around 1%, but on datasets with 5 cameras and 100 images per dataset. In our opinion, the best previous performance has been reported by Gisolf et al. (2014) – on a real dataset with the threshold parameter that was not tuned, the TPR is 89% at 0% FPR. The real dataset had 8760 images and 42 found clusters; such a magnitude is comparable to that of our Olympus dataset with the similar performance. However, the challenge in the method of Gisolf et al. (2014) is to correctly set the similarity threshold, which depends on the underlying camera models, that is, on information not available

in practice. As demonstrated in Gisolf et al. (2014), for Canon IXUS 220 HS a twofold increase of the threshold is needed with respect to other cameras in order to split well, and it is suspected that this type has model-specific post-processing that causes a higher correlation between cameras of this model. This phenomenon is not surprising. For example, the distance between cameras in our Praktica set is much smaller than the distance between the cameras in, e.g., Pentax dataset. Thus, the crux of our method is the number of visually separated clusters in the low-dimensional embedding, regardless of the actual inter-cluster (or intra-cluster) distance. This is in contrast with other approaches that rely on a pre-defined threshold to split the images into clusters.

With respect to the above point, the performance of our methodology may seem to be influenced by how the user interprets the visualization – for example, how many clusters she/he can recognize in the low-dimensional embedding. However, going back again to Anscombe's quartet (Anscombe, 1973), let us note that visualization can actually add value to data analysis where statistical analysis cannot make a difference. Moreover, as we discussed above, relying on a threshold similarity parameter that has been tuned on a benchmarking dataset may cause bias. This can happen when the real dataset contains cameras that apply image post-processing and thus disturb the expected correlation (similarity) among their images. In contrast, our method does not require pre-tuned parameters and the user always starts from a "scratch" when considering new datasets that may contain (so far) unseen camera models. In other words, clustering based on visualization allows for a wide range of intra-cluster versus inter-cluster similarities.

In conclusion, there has been a significant corpus of previous work on clustering images for common source camera detection. However, to the best of our knowledge, no previous approach uses data visualization in the clustering process. More concretely, the added values of our approach are as follows:

1. Our method does not require "blind" parameter tuning – the number of clusters that is required to feed the classical clustering algorithms is visually recognizable from the embedding;
2. Our method does not require a priori filtering of under- or over saturated images to avoid their influence on the clustering process, because outliers are recognizable from the visualization;
3. The relationship between the cluster sizes and the total number of clusters does not affect the embedding and clustering process;
4. The interactive visualization allows the user to take an active role and to use his domain expertise to gain insights that would aid the investigation by e.g. removing outliers;
5. Finally, the quantitative evaluation of our method also shows its effectiveness over previous approaches.

6. Conclusions

We have presented a new method for clustering images based on the PCE similarity scores of their PRNU patterns. It uses interchangeably low-dimensional embedding, interactive visualization and classical clustering. We evaluated the proposed method on the Dresden image database, and the results (presented in Table 2) show its effectiveness. The method does not require parameter tuning in the embedding phase. On the other hand, in the clustering phase

parameters like the number of clusters are easily deducible from the visualization. To the best of our knowledge, this is the first method for clustering images for common source camera detection that uses both embedding and (interactive) visualization.

Possibilities for future work include rigorous performance analysis of the image noise patterns embedding, other similarity scores computation algorithms, and testing the proposed method on other datasets.

Acknowledgements

This work was supported by the Netherlands eScience Center (grant number 027.000.001, 2016-2017). We would like to thank the Netherlands Forensics Institute for their close collaborations.

Appendix A. Data and software

The pairwise PCE similarity scores for the datasets used in this paper (Table 1) can be found in van Werkhoven (2017). These similarity scores were used as input to LargeVis Tang et al. (2016a) to generate vector representations of the image PRNU patterns. The vector representations can be found in Georgievska & van Werkhoven (2017) as .txt files. These vector representations, combined with the source camera labels, can also be found as .json files in Georgievska & van Werkhoven (2017). These files can be uploaded into SherlockDive, directly from web (b), to explore the quality of the embedding.

The textual (.txt) vector representations were used as input to Clustit (van Werkhoven et al., 2017) to obtain the cluster labels and the performance metrics presented in Table 2. The vector representations, enriched with cluster labels, can also be found as .json files in Georgievska & van Werkhoven (2017). These files can be uploaded into SherlockDive, directly from web (b), to explore the quality of the clustering.

The repositories with the latest versions of Clustit and SherlockDive can be found in web (c).

References

(a). Anscombe's quartet. `https://en.wikipedia.org/wiki/Anscombe's_quartet`. Accessed: 13 May 2017.

(b). The embedded pentax dataset. `https://sherlock-clustering.github.io/Sherlock_DiVE/`. Accessed: 19 May 2017.

(c). Sherlock clustering github organization. `https://github.com/sherlock-clustering`. Accessed: 19 May 2017.

Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Database Theory — ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings* (pp. 420–434). Berlin, Heidelberg: Springer Berlin Heidelberg.

Amerini, I., Caldelli, R., Crescenzi, P., Del Mastio, A., & Marino, A. (2014). Blind image clustering based on the normalized cuts criterion for camera identification. *Signal Processing: Image Communication*, *29*, 831–843.

Amigó, E., Gonzalo, J., Artiles, J., & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, *12*, 461–486.

Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician*, *27*, 17–21. URL: http://www.jstor.org/stable/2682899.

Bayram, S., Sencar, H. T., & Memon, N. (2015). Sensor fingerprint identification through composite fingerprints and group testing. *IEEE Transactions on Information Forensics and Security*, *10*, 597–612.

Bloy, G. J. (2008). Blind camera fingerprinting and image clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*, 532–534.

Caldelli, R., Amerini, I., Picchioni, F., & Innocenti, M. (2010). Fast image clustering of unknown source images. In *2010 IEEE International Workshop on Information Forensics and Security* (pp. 1–5). IEEE.

Chen, M., Fridrich, J., Goljan, M., & Lukáš, J. (2008). Determining image origin and integrity using sensor noise. *IEEE Transactions on Information Forensics and Security*, *3*, 74–90.

Chuang, W.-H., Su, H., & Wu, M. (2011). Exploring compression effects for improved source camera identification using strongly compressed video. In *2011 18th IEEE International Conference on Image Processing* (pp. 1953–1956). IEEE.

Fahmy, O. (2015). An efficient clustering technique for cameras identification using sensor pattern noise. In *Systems, Signals and Image Processing (IWSSIP), 2015 International Conference on* (pp. 249–252). IEEE.

Fridrich, J. (2009). Digital image forensics. *IEEE Signal Processing Magazine*, *26*, 26–37.

Georgievska, S. (2017a). Nlesc/dive: Dive 1.1. URL: https://doi.org/10.5281/zenodo.581179. doi:10.5281/zenodo.581179 funded by the Netherlands eScience Center.

Georgievska, S. (2017b). Sherlockdive 1.0. URL: https://doi.org/10.5281/zenodo.581175. doi:10.5281/zenodo.581175 funded by the Netherlands eScience Center.

Georgievska, S., & van Werkhoven, B. (2017). Embedded PRNU patterns of images from the Dresden Database. https://figshare.com/articles/data_zip/5016965. doi:10.6084/m9.figshare.5016965.v2 [dataset].

Gisolf, F., Barens, P., Snel, E., Malgoezar, A., Vos, M., Mieremet, A., & Geradts, Z. (2014). Common source identification of images in large databases. *Forensic science international*, *244*, 222–230.

Gisolf, F., Malgoezar, A., Baar, T., & Geradts, Z. (2013). Improving source camera identification using a simplified total variation based noise removal algorithm. *Digital Investigation*, *10*, 207–214.

Gloe, T., & Böhme, R. (2010). The Dresden Image Database' for Benchmarking Digital Image Forensics. In *Proceedings of the 25th Symposium On Applied Computing (ACM SAC 2010)* (pp. 1585–1591). volume 2.

Gloe, T., Pfennig, S., & Kirchner, M. (2012). Unexpected artefacts in prnu-based camera identification: A 'dresden image database' case-study. In *Proceedings of the on Multimedia and Security* '12 (pp. 109–114). New York, NY, USA: ACM.

Goljan, M. (2008). Digital camera identification from images–estimating false acceptance probability. In *International Workshop on Digital Watermarking* (pp. 454–468). Springer.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507. doi:10.1126/science.1127647.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*, 193–218.

Keogh, E., & Mueen, A. (2010). Curse of dimensionality. In C. Sammut, & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 257–258). Boston, MA: Springer US. URL: https://doi.org/10.1007/978-0-387-30164-8_192.

Li, C.-T. (2010). Unsupervised classification of digital images using enhanced sensor pattern noise. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* (pp. 3429–3432). IEEE.

Lin, X., & Li, C.-T. (2016). Large-scale image clustering based on camera fingerprints. *IEEE Transactions on Information Forensics and Security*, .

Lukas, J., Fridrich, J., & Goljan, M. (2006). Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, *1*, 205–214.

van der Maaten, L. (2014). Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, *15*, 3221–3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. *Columbia University*, .

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Saville, D. J., & Wood, G. R. (1991). *Statistical methods: the geometric approach*. Springer.

Strehl, A., & Ghosh, J. (2003). Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, *3*, 583–617.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016a). Largevis c++ implementation. `https://github.com/lferry007/LargeVis`. Accessed: 13 May 2017.

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016b). Visualizing large-scale and high-dimensional data. In *Proc. Conf. on World Wide Web* WWW '16 (pp. 287–297). ACM.

Tenenbaum, J. B., Silva, V. d., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

van Werkhoven, B. (2017). PCE similarities of the Dresden Image Database. `https://figshare.com/articles/PCE_similarities_dresden_zip/5017058`. doi:`10.6084/m9.figshare.5017058.v1` [dataset].

van Werkhoven, B. (Unpublished results). A jungle computing approach to common image source identification in huge collections of images, article in preparation. (pp. 1–12).

van Werkhoven, B., Kuzniar, A., Gavai, A., Sclocco, A., Bakhshi, R., Spreeuw, H., & Georgievska, S. (2017). Clustit 1.0. URL: `https://doi.org/10.5281/zenodo.581169`. doi:`10.5281/zenodo.581169` funded by Netherlands eScience Center.