Large-Scale Image Clustering Based on Camera Fingerprints

Xufeng Lin and Chang-Tsun Li, *Senior Member, IEEE*

***Abstract*—Practical applications of digital forensics are often faced with the challenge of grouping large-scale suspicious images into a vast number of clusters, each containing images taken by the same camera. This task can be approached by resorting to the use of *sensor pattern noise* (SPN), which serves as the fingerprint of the camera. The challenges of large-scale image clustering come from the sheer volume of the image set and the high dimensionality of each image. The difficulties can be further aggravated when the Number of Classes (i.e., the number of cameras) is much higher than the average Size of Class (i.e., the number of images acquired by each camera). We refer to this as the $NC \gg SC$ problem, which is not uncommon in many practical scenarios. In this work, we propose a novel clustering framework that is capable of addressing the $NC \gg SC$ problem without a training process. The proposed clustering framework was evaluated on the Dresden image database and compared with the state-of-the-art SPN-based image clustering algorithms. Experimental results show that the proposed clustering framework is much faster than the state-of-the-art algorithms while maintaining a high level of clustering quality.**

***Index Terms*—Image clustering, digital forensics, sensor pattern noise, large-scale data mining, divide-and-conquer, graph partitioning.**

I. Introduction

BEING capable of determining the origin of a digital image, *sensor pattern noise* (SPN), which arises from the imperfections in the manufacturing of sensors, has drawn much attention from the digital image forensics community in recent years. Its uniqueness to individual cameras and stability against environmental conditions make it a feasible fingerprint for identifying and linking source cameras [1–7]. To determine the origin of a given image among candidate cameras, a reference SPN is constructed for each candidate camera by averaging the noise residues extracted from a set of images taken by the camera. A given image is deemed to be taken by the camera associated with the detected SPN in the image. In this case, a set of images taken by the same camera is required for constructing the reference SPN. This requirement can be easily fulfilled when the camera is available to the investigators. However, in many real-world scenarios, only a set of images are available, without any information about the source cameras. In such circumstances, sometimes it is still desired to cluster the images into a number of groups, each including the images acquired by the same camera, so that the forensic investigators will be able to associate different crime scenes and thus be in a better position to link the evidence to the seized hardware owned by the suspects. This task can be accomplished by clustering the images based on the SPNs extracted from the images. Because the noise residue of an image (i.e., the difference between the original image and its denoised version) can be considered as the "noisy" SPN, we will refer to the noise residue as the "camera fingerprint", while the true SPN component in the noise residue is referred to as the "true SPN" in this paper. Additionally, to differentiate the ground truth and the clustering results, we will refer to the fingerprints of the same camera as a *class*, and refer to those clustered into the same group by the clustering algorithm as a *cluster*.

Clustering camera fingerprints is not an easy task, especially when the number of images has explosively increased across the Internet over the past decades. The size of image databases can be in the order of hundreds of thousands, which makes exhaustive pairwise comparison computationally prohibitive. Moreover, the extracted fingerprint from an image can be severely contaminated by other interferences. To guarantee the reliability and the accuracy of clustering, the dimension of camera fingerprint has to be very large, for example, 512×512 pixels or above. The high dimension of a camera fingerprint will impose a heavy burden on data storage and computation. Another challenge comes from the scenario where the Number of Classes is much higher than the average Size of Class, which will be referred to as the $NC \gg SC$ problem in this paper. This usually happens when we are dealing with massive Internet data. For example, the number of users on photo-sharing websites is normally much higher than that of photos uploaded by each user. Therefore, it is necessary to develop an effective algorithm to overcome the above-mentioned challenges.

In this work, we propose a novel clustering framework capable of dealing with large-scale camera fingerprint databases. It takes advantage of the dimension reduction and the inherent sparseness of the similarity matrix, to reduce the computational cost. Based on the analysis of correlation distribution, we have also derived an adaptive threshold, with regard to the size and the quality of clusters. It is the adaptive threshold that allows the clustering algorithm to work in a divide-and-conquer manner and makes the clustering on large-scale databases much more efficient.

The remainder of this work is organized as follows. In Section II, we will revisit the previous works on image clustering based on camera fingerprints. In Section III, the

X. Lin and C.-T. Li are with the Department of Computer Science, University of Warwick, Coventry, CV4 7AL, U.K. (e-mail: xufeng.lin@warwick.ac.uk; c-t.li@warwick.ac.uk).

details of the proposed clustering framework will be presented. The reasons why the proposed framework can cope with large-scale databases as well as the time complexity and I/O cost will be discussed in Section IV. Parameter settings and comprehensive experimental results will be given in Section V. Finally, Section VI concludes the work.

II. Previous Works

One of the first works dedicated to clustering camera fingerprints was reported in [8], where each enhanced fingerprint is treated as a random variable and Markov random field (MRF) is used to iteratively update the class label of each fingerprint. A subset of images are randomly chosen from the entire dataset to set up a training set. Based on the pairwise similarity matrix of the training set, a reference similarity and a membership committee are determined for each fingerprint. The similarity values and class labels within the membership committee are used to estimate the likelihood probability of assigning each class label to the corresponding fingerprint. Then the class label of a fingerprint is updated as the one with the highest likelihood probability in its membership committee. The clustering process stops when there are no label changes after two consecutive iterations. Finally, the fingerprints not in the training set will be assigned to their closest clusters identified in the training set. This algorithm performs well on small databases, but it is very slow due to the calculation of the likelihood probability, which involves all the members in the membership committee and needs to be calculated for every class label and every fingerprint. The time complexity is nearly $\mathcal{O}(n^3)$ in the first iteration, where n is the number of fingerprints, so it becomes obviously computationally prohibitive for large-scale databases. Another limitation is that when the $NC \gg SC$ problem presents, the size of the training set has to be very large to guarantee most of the classes are present in the training set. These two limitations make it computationally infeasible for large databases.

In [9], camera fingerprints clustering is formulated as a weighted undirected graph partitioning problem. Each fingerprint is considered as a vertex in a graph, while the weight of an edge is the similarity between the two fingerprints linked by the edge. To avoid the time-consuming pairwise similarity calculation, a κ-nearest graph is constructed as follows: A vertex is randomly selected as the first center, and its edge weights with all the other vertices are calculated. The $(\kappa+1)$th closest vertex to the initial center is then selected as the second center and its edge weights with all the other vertices, except the first center, are calculated, where κ is a parameter controlling the sparsity of the graph. This procedure is repeated until the number of vertices that have not been considered as a center is not larger than κ. A multi-class spectral clustering algorithm [10] is then employed on the constructed κ-nearest graph to partition the vertices (fingerprints). For every vertex being investigated, its similarities with all the other vertices have to be calculated when constructing the κ-nearest graph, which incurs high I/O cost for large databases because one fingerprint needs to be read from the disk many times for calculating its similarities with the centers due to the limited size of RAM. The time complexity of the algorithm in [10] is $\mathcal{O}(n^{\frac{3}{2}}m + nm^2)$, where m is the number of partitions. The optimal number of partitions is determined by specifying an increasing m and repeating the spectral clustering until the size of the smallest cluster equals 1. However, the feasibility of such manner for determining the optimal partitions number is still an issue for large-scale camera fingerprint databases.

Another algorithm proposed in [11] is based on the hierarchical clustering. Similar to [8], the fingerprint is enhanced beforehand and only a random subset (training set) of the whole dataset is used for clustering, followed by a classification stage for the remaining fingerprints. Initially considering each fingerprint as one cluster, the algorithm first calculates the pairwise similarity matrix of the training set. The two most similar clusters are merged into one and the similarity matrix is updated by replacing the corresponding two rows and columns with the similarities between the merged cluster and all other clusters. After the update, a silhouette coefficient, which measures the separation among clusters and the cohesion within each cluster, is calculated for each fingerprint. The silhouette coefficients are averaged to give a global measure of the aptness of the current partition. When all fingerprints have been merged into one cluster, the partition corresponding to the highest aptness is deemed to be the optimal partition. Another hierarchical clustering based algorithm was proposed in [12], where the only difference is that the calculation of the silhouette coefficient is performed for each cluster rather than for each fingerprint and only the separation to the nearest neighboring cluster is measured. As reported in [11], with comparable accuracy, the hierarchical clustering based algorithm is faster than [8]. But the computational cost of the hierarchical clustering is still very high and requires at least $\mathcal{O}(n^2 \log n)$ operations. The high computational cost, therefore, limits its applicability to large databases.

It can be observed that existing methods either cluster on a training set randomly sampled from the original dataset [8, 11, 12] or calculate a small portion of the pairwise similarities [9] to generate a collection of representative clusters, the centroids of which will be used to classify the remaining fingerprints by assigning each of them to the most similar centroid. However, the successful classification requires that the whole dataset is well represented by the representative clusters. However, sometimes we are confronted with the $NC \gg SC$ problem, where the number of classes within the training set is probably far less than that of the original dataset. The $NC \gg SC$ problem makes it difficult, if not impossible, to form a training set at random that can sufficiently represent the entire population. In consequence, misclassifications happen when some of the remaining fingerprints do not belong to any of the representative clusters. Li proposed in [8] that if the similarity between one fingerprint and the most similar centroid is less than a predefined threshold, the fingerprint is considered as a new representative cluster and used to classify the remaining fingerprints. But not only the reliability of the new singleton representative cluster is doubtful, but also the selection of the threshold can be annoying. If the threshold is not appropriately set, it is very likely that one fingerprint does not belong to any of the representative clusters, but

its similarity with the closest representative cluster is higher than the preset threshold. This usually happens when some fingerprints within one representative cluster are not purely from the same camera. What is worse, such misclassification can be propagated in the succeeding classification process. Therefore, an effective way of determining an appropriate threshold is urgently needed.

The characteristics of the camera fingerprints clustering problem also make it difficult to employ most of the classic clustering algorithms. For example, the partition clustering algorithms, as typified by K-means [13] and CLARANS [14], require users to input the desired partition number K, the determination of which can be tricky in practical situations. Moreover, they may require several passes over the database and thus do not scale well to large-scale camera fingerprint databases. The density based approaches, such as DBSCAN [15], are directly performed on the entire database. As a result, for large databases that cannot fit in the main memory, it could incur substantial I/O cost [16]. Furthermore, its sensitiveness to parameters and its inability to handle clusters with various densities make it hard to produce satisfactory results on camera fingerprints, whose noise-like characteristics can easily result in clusters with various densities. Some hierarchical clustering algorithms using random sampling to reduce the input size for large databases, such as [16] and [17], will suffer from the $NC \gg SC$ problem. Other hierarchical clustering algorithms designed for large-scale databases, as typified by BIRCH [18] and CHAMELEON [19], do not perform well on camera fingerprint databases because of either the sensitivity to outliers or the high I/O cost when constructing the κ-nearest neighbor graph.

III. Proposed Clustering Framework

Given the limitations of the existing methods reviewed in the previous section, we propose a new clustering framework as shown in Fig. 1. The proposed framework mainly consists of preparation, coarse clustering, fine clustering, attraction and post-processing. In what follows, we will look into each of the five steps to provide a rough picture of the proposed framework.

Step 1: Preparation

- **Image preprocessing:** Eliminate the images with saturation or low intensities. Rotate the remaining images to ensure they are all horizontally oriented.
- **Fingerprint extraction and standardization:** Suppose there are n images left in the database after removing the dark or saturated images. A d-dimensional fingerprint is extracted from the center of the green channel of each image and standardized to zero mean and unit variance.
- **Dimension reduction:** Project each of the d-dimensional fingerprints onto a k-dimensional subspace ($k < d$) using the *very sparse random projection* proposed in [20].
- **Fingerprint data storage:** Store the full-length fingerprint, the corresponding dimension-reduced fingerprint as well as the image name into a single file for each image.

Step 2: Coarse clustering

- **Correlation matrix approximation:** To avoid calculating the $n \times n$ pairwise correlation matrix $\boldsymbol{M}$, the dimension-reduced fingerprints and the potential-based eviction (to be discussed in Section III-B) are used to reduce the computational cost. In this way, $\boldsymbol{M}$ is replaced with a set of more RAM-efficient lists $\boldsymbol{L}$, of which each element $\boldsymbol{L}_i$ records the indices of the fingerprints that are similar enough to the ith fingerprint.
- **Coarse partition:** Based on the graph information recorded in $\boldsymbol{L}$, the n fingerprints are coarsely partitioned into n_c coarse clusters using the fast graph clustering algorithm in [21].

Step 3: Fine clustering

- **Cluster splitting:** For each of the n_c coarse clusters, a correlation matrix is calculated and binarized as in the coarse clustering stage, but using the d-dimensional full-length fingerprints. Because the size of each coarse cluster is small, the calculation of the correlation matrix is fast and efficient. Based on the binary correlation matrix corresponding to one coarse cluster, the cluster is naturally split into n_s sub-clusters with variable sizes using the flow simulation based graph clustering algorithm [22].
- **Cluster merging:** Each of the n_s sub-clusters is represented by the centroid averaged over the full-length member fingerprints. Based on the n_s centroids, a $n_s \times n_s$ pairwise correlation matrix is calculated and binarized using an adaptive threshold matrix $\boldsymbol{\tau}$ (the element τ_{ij} depends on the sizes and the qualities of sub-cluster S_i and sub-cluster S_j). Larger clusters are thus obtained by merging two or several sub-clusters. Meanwhile, the centroids and the qualities of the merged clusters are updated accordingly.

Step 4: Attraction

- **Centroid attraction:** The centroids are used as "attractors" to attract the unclustered fingerprints left in the database. Consequently, most of the fingerprints belonging to the discovered clusters will be absorbed to form larger clusters.

Step 5: Post-processing

- **Cluster screening and storage:** Place the fingerprints in the clusters with a size smaller than a threshold η back into the database and store the remaining clusters as the final clusters. The stored information includes the centroid, the names of the corresponding images, and the quality of the cluster.
- **Termination or continuation:** The algorithm ends if no more notable clusters can be found. Otherwise, place the unclustered fingerprints, as well as those in the clusters with a size smaller than η, back into the database for the next round of clustering.

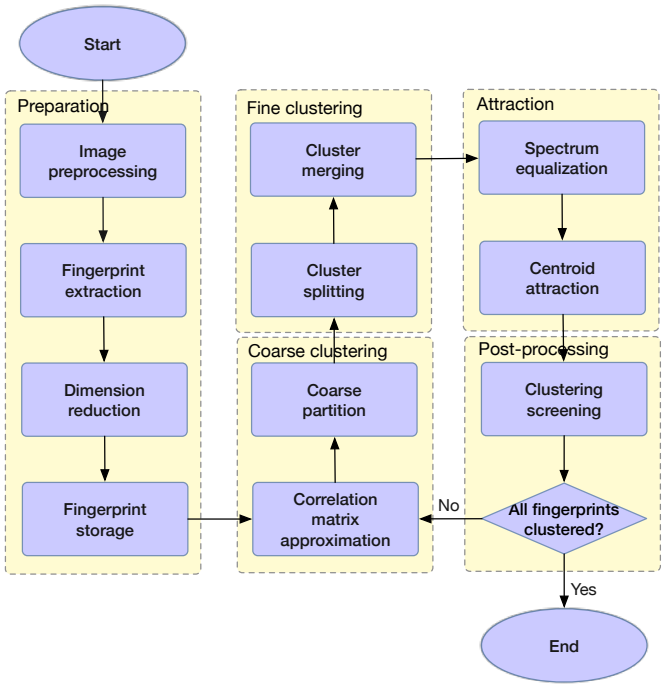


Fig. 1: Flow chart of the proposed framework.

In the following subsections, we will provide more details on each of the above steps.

A. Preparation

Since the fingerprint information is not present in the dark or saturated images [2], involving these images into the clustering process not only entails high computational burden, but also makes the clustering results unreliable and unpredictable. Hence, those images with insufficient fingerprint information will be ignored. Another concern comes from the orientation of images. As we know, images taken by the same camera can be in either the horizontal or vertical orientation depending on how the camera is held. For JPEG/TIFF images, the orientation information may have been stored in the exchangeable image file format (EXIF) header of the image file, but the EXIF information is untrustworthy and sometimes unavailable. We therefore rotate the vertically oriented images clockwise by 90° to ensure all images are horizontally oriented. Although this cannot completely solve the rotation problem, it successfully limits the freedom of rotation.

After the above preprocessing, a camera fingerprint $\boldsymbol{F}$ is extracted from the central block of the green channel of each image $\boldsymbol{I}$, i.e.,

$$\boldsymbol{F} = \boldsymbol{I} - \mathcal{G}(\boldsymbol{I}), \tag{1}$$

where $\mathcal{G}$ is the denoising filter. The normalized cross correlation (NCC) ρ is used as the similarity measurement between two fingerprints, $\boldsymbol{X}$ and $\boldsymbol{Y}$:

$$\rho(\boldsymbol{X},\boldsymbol{Y}) = \frac{\sum_{i=1}^{d}(\boldsymbol{X}[i]-\overline{\boldsymbol{X}})(\boldsymbol{Y}[i]-\overline{\boldsymbol{Y}})}{\sqrt{\sum_{i=1}^{d}(\boldsymbol{X}[i]-\overline{\boldsymbol{X}})^2}\sqrt{\sum_{i=1}^{d}(\boldsymbol{Y}[i]-\overline{\boldsymbol{Y}})^2}}, \tag{2}$$

where d is the length of the fingerprint, $\overline{\boldsymbol{X}}$ and $\overline{\boldsymbol{Y}}$ are the arithmetic mean of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. NCC is notorious for its high computational cost due to the calculation of the variances in the denominator of Equation (2). To alleviate this, each fingerprint is standardized to have zero mean and unit variance. After standardization, Equation (2) is simplified to the more efficient element-wise product:

$$\rho(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{d}\sum_{i=1}^{d}\hat{\boldsymbol{X}}[i]\hat{\boldsymbol{Y}}[i], \tag{3}$$

where $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ are the standardized version of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively.

Standardization also builds up the equivalence between NCC and other commonly-used similarity measures such as the inner product[1] and the Euclidean distance [23]. It reminds us that many similarity-preserving dimension reduction methods are potentially suited for use in reducing the computational cost. Unfortunately, dimension reduction techniques based on the discrete Fourier transform (DFT) [24], singular value decomposition (SVD) [25], discrete wavelets transform (DWT) [26] and piecewise aggregate approximation (PAA) [27] are not applicable to our application due to the nature of noise-like camera fingerprints. More advanced methods, such as principal components analysis (PCA) [28], isometry mapping [29], maximal variance unfolding [30] and locally linear embedding [31] are computationally infeasible for large-scale camera fingerprints due to the mutual dependencies of data. We therefore resort to another more efficient solution — *very sparse random projection* [20, 32] due to its appealing properties of Euclidean distance-preserving, computational efficiency and data independence. Let $d\times n$ matrix $\boldsymbol{D}$ be the set of n camera fingerprints in the d-dimensional space $\mathbb{R}^d$, and $\boldsymbol{R}$ be the $k\times d$ random projection matrix with $\boldsymbol{R}(i,j)=r_{ij}$, where $\{r_{ij}|1\le i\le k, 1\le j\le d\}$ are independent and identically distributed (i.i.d.) random variables drawn from the following probability distribution:

$$r_{ij} = \sqrt{\frac{s}{k}}\times\begin{cases}+1 & \text{with probability} \quad 1/2s,\\ 0 & \text{with probability} \quad 1-1/s,\\ -1 & \text{with probability} \quad 1/2s,\end{cases} \tag{4}$$

where $s=\sqrt{d}$ according to [20]. The dimension reduction of the n fingerprints from $\mathbb{R}^d$ to $\mathbb{R}^k$ is achieved by a matrix multiplication:

$$\boldsymbol{E} = \boldsymbol{R}\boldsymbol{D}, \tag{5}$$

where $\boldsymbol{E}$ is a $k\times n$ matrix, with its columns representing the dimension-reduced camera fingerprints.

Finally, the standardized fingerprint, the dimension-reduced counterpart, along with the image name are stored in a file with an unique filename. The dimension-reduced and the full-length fingerprint will be used for the following coarse clustering and fine clustering, respectively.

[1] $\rho(\boldsymbol{X},\boldsymbol{Y}) = \langle\frac{\hat{\boldsymbol{X}}}{\sqrt{d}},\frac{\hat{\boldsymbol{Y}}}{\sqrt{d}}\rangle$

B. Coarse clustering

One of the key challenges of clustering large-scale and high-dimensional camera fingerprints is to discover the potentially correlated fingerprint pairs. Calculating the complete, exhaustive set of similarities in the $n \times n$ pairwise correlation matrix $\boldsymbol{M}$ requires at least $n(n-1)/2$ comparisons, which becomes clearly prohibitive for large databases. Even if we have the oracle to know all of the entries of $\boldsymbol{M}$, it is still an issue to cache them in the RAM.

Practically speaking, the correlation matrix is very likely to be sparse because only the fingerprints of the same camera are correlated with each other. So instead of keeping all the correlations in the RAM, we use a threshold t_b to binarize the similarities (correlations). Since the intrinsic quality of fingerprints depends on many complex factors, the average similarities for different cameras may vary substantially. But in the sense of clustering, one class with a higher average intra-class similarity should be equivalent to another class with a lower average similarity. In this regard, binarization is an effective way to eliminate the divergences in different classes and make the clustering more resilient against the interferences in the fingerprints.

Another optimization is to calculate a small portion of the $n(n-1)/2$ matrix entries, then the key issue becomes what heuristics or criteria can be used to select the matrix entries for calculation. One such work was proposed in [9], but, as mentioned in Section II, the expensive I/O cost makes it unaffordable for large-scale databases. Given n dimension-reduced fingerprints, we randomly partition them into batches of equal size q, where q can be customized so that the RAM is sufficient for simultaneously accommodating two batches and the extra space for calculating the inter-batch correlations. Two batches are firstly brought into the RAM, and the correlations of the fingerprints in the two batches are calculated and binarized for updating a set of lists $\boldsymbol{L}$, of which each element $\boldsymbol{L}_i$ is a list recording the indices of the vertices adjacent to the ith fingerprint. All remaining batches are sequentially loaded from the disk one at a time. To proceed with the next batch, at least q fingerprints have to be evicted from the RAM. We consider each dimension-reduced fingerprint as a vertex $v_i, i = 1, 2, ..., n$ in a graph G, and define a *potential* measurement p_i characterizing its potential connectivities with other vertices:

$$p_i = \frac{\deg(v_i)}{A_i}, \tag{6}$$

where $\deg(v_i)$ is the *degree* of vertex v_i (i.e., the length of $\boldsymbol{L}_i$) and A_i is the accumulated number of vertices whose connectivities with vertex v_i have been investigated up to the latest batch. $p_i \in [0, 1]$ is initialized as 0 at the beginning. In order to accommodate the new incoming batch, the q vertices with the lowest *potentials* are evicted from the RAM. We refer to this strategy as the *potential-based eviction*. The underlying motivation is that *the more fingerprints a camera fingerprint is connected with among those that have been investigated, the more likely that it will be connected with more fingerprints among the ones that have not been investigated yet.* By taking advantage of the asymmetry of the class distribution, the potential-based eviction allows for discovering more correlated fingerprint pairs based on the limited number of correlation calculations. Meanwhile, the I/O cost is minimized because every dimension-reduced fingerprint is loaded only once. It is worth noting that we use p_i instead of $\deg(v_i)$ because the number of pairwise comparisons for the vertices in the posterior batches is less than that for the vertices in the previous batches. As a consequence, the vertices loaded earlier tend to have a higher *degree* than the ones loaded later.

With the potentially correlated fingerprint pairs recorded in $\boldsymbol{L}$, it is fed into a multilevel graph partitioning algorithm Graclus [21], which works by repeatedly coarsening the original graph (corresponding to $\boldsymbol{L}$) to a point where very few vertices remain in the graph, then a spectral clustering method [33] is applied on the coarsest graph, the clustering result of which will be refined level by level up to the original graph. By empirically specifying the cluster number $n_c = \lceil n^{\frac{1}{4}} \rceil$, where $\lceil \cdot \rceil$ is a ceiling operator, Graclus partitions the n fingerprints into n_c coarse clusters with various sizes, among which some large coarse clusters will be recursively bisected into small clusters to ensure that each of them can be fit into the RAM at a time. The purpose of coarse clustering is to gather potentially correlated fingerprints into the same batch and therefore increase the probability of forming reliable clusters in the following fine clustering stage.

C. Fine clustering

After coarsely partitioning the whole database, the coarse clusters will be further split and merged in the fine clustering stage. Specifically, for each of the resultant coarse clusters, an accurate correlation matrix is calculated and binarized as in the coarse clustering stage, but using the full-length (d-dimensional) rather than the dimension-reduced (k-dimensional) fingerprints. Because the number of fingerprints in each coarse cluster is small, the processing of each coarse cluster can be quite efficient. The binarized correlation matrix is input to the Markov cluster algorithm (MCL) [22], which iteratively applies the expansion (matrix multiplication) and inflation (entry-wise matrix product) operators until it converges to a steady state. Finally, the clusters are discovered by interpreting the resultant matrix. Given a small enough coarse cluster, many alternative algorithms, such as [8, 9, 11, 12, 34], are feasible for the clustering task. But the following merits make MCL preferable to other methods:

1) MCL can be significantly sped up by taking advantage of the sparseness of the correlation matrix and pruning the small matrix entries during each iteration.
2) MCL tends to result in small cluster granularity but high precision rate. Given a large volume of fingerprints, it is difficult to achieve high recall rate and high precision rate simultaneously. Mixing the fingerprints of different cameras in the same cluster (low precision rate) can lead to error propagation in the ensuing clustering process. However, dispersing the fingerprints of the same camera into several clusters (low recall rate) still ensures high correctness and reliability of the clustering despite the extra computational cost. Therefore, if we have to sacrifice in one aspect to gain in the other, we would usually

prefer the high precision rate to the high recall rate in practice.

The split of coarse clusters may produce many small or even singleton sub-clusters, but only the sub-clusters with a score ξ greater than a predefined threshold t_s are collected for further use:

$$\xi = \sqrt{|C|}\frac{\sum_{v_i \in C}\sum_{v_j \in C} e_{ij}}{|C|(|C|-1)} > t_s, \tag{7}$$

where $|C|$ is the size of cluster C, e_{ij} is a binary variable with 1 signifying that v_i is connected with v_j. But due to the limited RAM size, sub-clusters are sorted according to the score ξ and only the $\mathcal{H}$ sub-clusters with the highest scores are retained in the RAM, where $\mathcal{H}$ is adaptive to the RAM size. We refer to this strategy as *score sorting*. In such a manner, we can make better use of the limited RAM and guarantee that the larger classes (usually related to the sub-clusters with a higher score) will be clustered preferentially.

In the coarse clustering stage, the fingerprints of the same camera are likely to be partitioned into different coarse clusters. Even if those fingerprints are grouped into the same coarse cluster, they may still be split into different sub-clusters in the ensuing splitting stage, so it is desirable to merge the sub-clusters of the same camera for both efficiency and accuracy reasons before proceeding further. Hereafter, we will refer to the correlation between the centroids of two clusters from the same camera as the *intra-class* correlation, and the correlation between the centroids of two clusters from different cameras as the *inter-class* correlation. Intuitively, the intra-class correlation increases with the cluster size because the random noises in the centroid of a larger cluster have been suppressed more significantly, while the inter-class correlation remains the same. If we know how the correlation between two centroids changes with the sizes of clusters, an adaptive threshold can be adopted to determine whether they are from the same camera. The adaptive thresholding problem for camera fingerprint clusters merging has been studied in [35] and [36] by estimating the parameters of a prescribed function, but both of them do not generalize well across different cameras. According to the Central Limit Theorem (CLT), the NCC ρ between two d-dimensional centroids, $\boldsymbol{X}$ and $\boldsymbol{Y}$, from different cameras conforms to a normal distribution with zero mean and $1/d$ variance, i.e., $\rho(\boldsymbol{X},\boldsymbol{Y}) \sim \mathcal{N}(0, 1/d)$. If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the centroids of two sub-clusters S_x and S_y from the same camera, we can derive that the distribution approaches to a normal distribution when $d \to \infty$ (please refer to the Derivation of Correlation Distribution in the Appendix), i.e.,

$$\rho(\boldsymbol{X},\boldsymbol{Y}) \xrightarrow{d} \mathcal{N}(\mu, \Sigma), \tag{8}$$

where

$$\begin{cases} \mu = \sqrt{\frac{n_x n_y \sigma_x^2 \sigma_y^2}{[(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]}} \\ \Sigma = \frac{n_x n_y \sigma_x^2 \sigma_y^2 + [(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]}{[(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]d}. \end{cases} \tag{9}$$

Here, d is the length of fingerprints. n_x and n_y are the sizes (i.e., numbers of member fingerprints) of cluster S_x and S_y, respectively. σ_x^2 and σ_y^2 are the average qualities of the true SPN in each member fingerprint in S_x and S_y. When there is no ambiguity, the average quality of the true SPN in each member fingerprint of one cluster will be simply referred to as the quality of the cluster. For large clusters, the mean of the intra-class distribution is normally far from the zero mean of the inter-class distribution, so the threshold can be safely increased to reduce the false positives. An adaptive threshold τ characterizing the change in the intra-class distribution is therefore proposed as

$$\tau = \max\Big(t_b, \frac{\omega\sqrt{n_x n_y \sigma_x^2 \sigma_y^2}}{\sqrt{[(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]}}\Big), \tag{10}$$

where t_b is the threshold used for binarizing the similarity matrix in the fine clustering stage, ω is a predefined scaling factor. Two sub-clusters with a correlation between their centroids higher than τ are considered to be from the same camera. In Equation (10), τ is related to the dynamic intra-class distribution rather than the constant inter-class distribution, so it is more reliable than the threshold determined by the Neyman-Pearson criterion when the cluster size keeps increasing. In such a way, τ effectively prevents the cluster merging from propagating errors into the merged clusters. σ_x^2 and σ_y^2 in Equation (10) can be estimated by calculating the mean of correlations (please refer to μ_1 in Equation (25) with λ=1 in the Appendix). Therefore, when coarse clusters are split into n_s sub-clusters, the quality of each sub-cluster is initially estimated as

$$\sigma_i^2 = \frac{1}{q_i}\sum_{k=1}^{q_i} \rho_k, i = 1, 2, \ldots, n_s, \tag{11}$$

where q_i is the number of the calculated correlations within S_i. Following the analysis in Scenario 2 in the Appendix, if S_x and S_y are merged into $\hat{S}$ in the cluster merging stage, we update the quality of $\hat{S}$ as

$$\hat{\sigma}^2 = \frac{n_x\sigma_x^2 + n_y\sigma_y^2}{n_x + n_y}. \tag{12}$$

In particular, when σ_x^2=σ_y^2=σ^2, $\hat{\sigma}^2$ of the merged cluster $\hat{S}$ is the same as that of S_x or S_y, i.e., $\hat{\sigma}^2$=σ^2. Equation (12) can be easily generalized for merging c sub-clusters:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{c} n_i\sigma_i^2}{\sum_{i=1}^{c} n_i}, \tag{13}$$

where n_i and σ_i^2 are the size and the quality of S_i, respectively.

D. Attraction

The following stage is the centroid attraction, where the centroids of the merged clusters are used as the "attractors" to attract the fingerprints remaining in the database, so that the fingerprints belonging to the same cluster will be absorbed into the corresponding attractor. Specifically, one fingerprint $\boldsymbol{F}$ belongs to attractor $\boldsymbol{A}^*$ if

$$\boldsymbol{A}^* = \underset{\boldsymbol{A}_i \in \mathcal{A}}{\operatorname{argmax}}\, \rho(\boldsymbol{A}_i, \boldsymbol{F}), \rho(\boldsymbol{A}^*, \boldsymbol{F}) > \tau_\rho, \tag{14}$$

where $\mathcal{A}$ is the set of attractors and τ_ρ is calculated from Equation (10). Finally, the attracted fingerprints belonging to

the same cluster are merged into the corresponding cluster and the quality of the merged cluster is updated accordingly. Suppose an attractor $\boldsymbol{A}_x$ is averaged over n_x fingerprints with an average quality σ_x^2 of the true SPN in each fingerprint, then according to μ in Equation (9), the average quality σ_y^2 of the n_y attracted fingerprints can be estimated as

$$\sigma_y^2 = \frac{[(n_x-1)\sigma_x^2+1]\sum_{k=1}^{n_y}\rho_k^2}{n_x n_y \sigma_x^2}. \tag{15}$$

The set of fingerprints corresponding to $\boldsymbol{A}_x$ and the set of n_y attracted fingerprints can be viewed as two clusters with a quality of σ_x^2 and σ_y^2, respectively. So according to Equation (12), the quality of the attractor $\boldsymbol{A}_x$ and the number of member fingerprints are updated as

$$\begin{cases} \sigma_x^2 \leftarrow \dfrac{n_x^2\sigma_x^4 + [(n_x-1)\sigma_x^2+1]\sum_{k=1}^{n_y}\rho_k^2}{n_x(n_x+n_y)\sigma_x^2} \\ n_x \leftarrow n_x + n_y. \end{cases} \tag{16}$$

E. Post-processing

Up until now, the clustering process has formed a certain number of clusters, each of which is presented by a centroid, the names of the corresponding images, and the quality of the cluster. Notable clusters with a size no smaller than η are stored as the final clusters. While those clusters with a size less than η, as well as the remaining unclustered fingerprints, are placed back to the database for the next round of iteration starting from the coarse clustering stage. Due to the nature of the potential-based eviction in the coarse clustering stage and the score sorting in the fine clustering stage, the classes with a larger size are more likely to form notable clusters, so the majority of the fingerprints can be clustered in the first few iterations. During the coarse clustering stage of the next iteration, we put the unclustered fingerprints from the same batch in the previous iteration into different batches to increase the chance of discovering correlated fingerprint pairs. The algorithm terminates when no more notable clusters can be discovered.

IV. Discussion

After presenting the algorithm as above, the reasons why the proposed clustering framework can cope with large-scale camera fingerprint databases become clear. By taking advantage of the dimension reduction technique and the sparseness of the pairwise similarity matrix, the clustering problem of a large database is broken down into the clustering of several smaller databases with the help of the fast, but approximate, graph partition algorithm. The adaptive thresholding significantly reduces the computational complexity and allows the clustering results of the smaller databases to be combined to give the solution to the $NC \gg SC$ problem. The ability of spotting small classes is conferred by the iterative clustering manner and the adaptive value of coarse cluster number $n_c = \lceil n^{\frac{1}{4}} \rceil$. On one hand, the iterative manner (thanks to the potential-based eviction and the score sorting strategy) guarantees that the larger classes will be clustered in the first few iterations and the smaller classes will be more focused on in the ensuing iterations. On the other hand, with the decreasing number of coarse clusters, the probability that more fingerprints from smaller classes fall into the same coarse cluster increases, making them more easily to be discovered in the fine clustering stage.

The time complexity of the proposed framework depends on a complex interplay between the number of fingerprints n, the length of the fingerprints, the class distribution in the dataset, the number of edges $|E|$, and the parameter settings. In the coarse clustering stage, it involves nqk multiplications to calculate the sparse approximate correlation matrix, where q is the size of one batch and k is the reduced dimension. The fast graph partitioning algorithm Graclus [21] approximately has a time complexity $\mathcal{O}(q|E|/n)$. In the fine clustering stage, it involves around nbd multiplications to calculate the accurate correlation matrix, where b is the average size of the coarse clusters and usually much smaller than q, and d is the length of the full-length fingerprint. The MCL has a time complexity of $\mathcal{O}(nK^2)$ in the worst case [22], where K is the maximal number of nonzero entries in one column of the binarized correlation matrix. Finally, the attraction stage involves at most $ncd\theta$ multiplications, where $\theta \in (0, 1]$ is a factor accounting for the percentage of the c classes that have been discovered. Considering the very high dimension of the fingerprints, the time complexity $\mathcal{O}(q|E|/n)$ of Graclus is negligible because even for an extremely tight and dense graph, $\mathcal{O}(q|E|/n) = \mathcal{O}(nq)$ is trivial when compared with the time complexity $\mathcal{O}(nqk + nbd)$. So the overall time complexity of the proposed clustering framework is approximately $\mathcal{O}(nqk + nbd + nK^2 + ncd\theta)$ in one iteration. q and b can be fixed or made adapted to the RAM size, while K, c and θ depend on the class distribution of the dataset. With regard to the I/O cost of loading fingerprints from disk, it is $\mathcal{O}(2nd{+}nk)$ in the worst case. At the first glance, it is about two times as high as the minimal I/O cost $\mathcal{O}(nd)$. But given that one fingerprint needs to be loaded more than once to calculate the pairwise correlations for large datasets due to the limited size of RAM, $\mathcal{O}(2nd + nk)$ is still at an acceptable level and a considerable speedup can be expected by using the Solid State Drive (SSD).

V. Experiments

A. Experimental setup

The proposed clustering framework was evaluated on the Dresden image database [37, 38]. After the removal of the dark and saturated images, involved in the experiments are $15,840$ images taken by 74 cameras, covering 27 models and 14 brands. The number of images taken by each camera varies from 154 to 460. All experiments were conducted on a PC with windows 7 OS, 3.2 GHz Intel Quad Core Processor, 16 GB RAM and 1 TB Hard Disk Drive (HDD). To alleviate the vignetting effects [39] and ensure a consistent size for all fingerprints, camera fingerprints were extracted from the central 1024×1024 block of the green channel of the full resolution images (i.e., $d{=}1,048,576$). The method proposed in [1] was used to extract the camera fingerprints, which were further preprocessed by two operations, zero-meaning (ZM)

and Wiener filtering (WF) in the DFT domain [2], to suppress the non-unique artifacts. Note that we did not employ the spectrum equalization algorithm (SEA) [40], because the peaks in the spectrum of a single fingerprint are not so conspicuous as those in the spectrum of the reference SPN averaged over several fingerprints.

The proposed framework is designed for clustering large-scale image database, but we are also interested in its performance on small datasets. Therefore, based on the Dresden database, we set up four different small datasets. As we know, images taken by different devices of the same model undergo the same or similar in-camera processing procedures, so it is more challenging to correctly cluster the fingerprints of the devices of the same model. We categorized the clustering difficulties into *easy* and *hard* levels. In the easy level, images in different classes are acquired by devices of different models, while in the hard level, some images are taken by devices of the same model. Moreover, it is not uncommon in practical applications that the number of images captured by different devices varies widely, which results in different class distributions within the dataset. We, therefore, categorized the distributions of images in different classes into *symmetric* and *asymmetric*. Finally, we set up the following four datasets for the experiments:

- $\mathcal{D}_1$: Easy symmetric dataset. It consists of $1,000$ images taken by 25 cameras (each responsible for 40 images). The 25 cameras are of different models and cover nearly all of the popular camera brands, such as Canon, Nikon, Olympus, Pentax, Samsung and Sony.
- $\mathcal{D}_2$: Easy asymmetric dataset. 20, 30, 40, 50 and 60 images were alternatively chosen from the images taken by the same 25 cameras as in $\mathcal{D}_1$.
- $\mathcal{D}_3$: Hard symmetric dataset. It consists of $1,000$ images taken by 50 cameras (each responsible for 20 images). The 50 cameras only cover 12 popular models, so some of them are from the same model.
- $\mathcal{D}_4$: Hard asymmetric dataset. 10, 15, 20, 25 and 30 images were alternatively chosen from the images taken by the same 50 cameras as in $\mathcal{D}_3$.

Notice that the size of each dataset was fixed to $1,000$, while the number of classes, as well as the ratio of the number of classes to the average size of classes, was raised in $\mathcal{D}_3$ and $\mathcal{D}_4$. These four datasets will be used for investigating parameters, evaluating the capability of the proposed framework in conquering the $NC \gg SC$ problem, and comparing the performances of different clustering algorithms on small datasets. The results on the entire Dresden database can be found in the last part of Section V.

The quality of clustering is characterized in terms of the F1-measure

$$\mathcal{F} = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{R}}{\mathcal{P} + \mathcal{R}}, \tag{17}$$

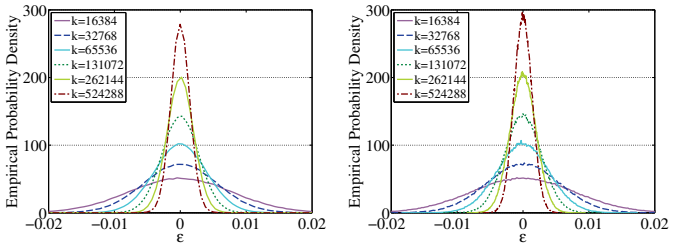
where the average precision rate $\mathcal{P}$ and the average recall rate $\mathcal{R}$ are calculated as

$$\begin{cases} \mathcal{P} = \sum |c_i \cap \psi_{j^\star}| / \sum |c_i| \\ \mathcal{R} = \sum |c_i \cap \psi_{j^\star}| / \sum |\psi_{j^\star}|. \end{cases} \tag{18}$$

Here, $|c_i|$ is the size of cluster c_i, and $|\psi_{j^\star}|$ is the size of the largest sub-cluster $\psi_{j^\star}$ in cluster c_i, i.e., $j^\star = \text{argmax}_j\{|c_i \cap \psi_j|\}$.

B. Parameter settings

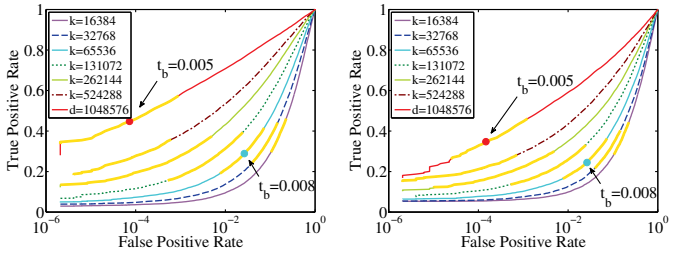
There are a few parameters that need to be set for our proposed framework. We will investigate the impact of these parameters on performance and discuss how to determine the appropriate parameter settings.



(a) (b)

Fig. 2: Probability density functions of the embedding error ϵ in pairwise correlations of $\mathcal{D}_1$ (a) and $\mathcal{D}_3$ (b).

The first parameter is the dimension k of the subspace of random projection. As remarked in [20], we can apply, with a high level of accuracy, the results of *conventional random projection*, i.e., the i.i.d. entries of $\boldsymbol{R}$ are drawn from the standard normal distribution $\mathcal{N}(0,1)$, to *very sparse random projection*. For example, we can determine the minimum k that achieves an embedding error ϵ in pairwise correlation preservation[2] using *Theorem 4* in [41]. But this gives a very conservative estimation of the minimum k. We drew the distributions of ϵ for different k using the camera fingerprints in dataset $\mathcal{D}_1$ ($n = 1,000$) and $\mathcal{D}_3$ ($n = 1,000$), as shown in Fig. 2a and Fig. 2b, respectively. According to *Theorem 4* in [41], if we want to preserve 80% of the pairwise correlations with an error less than $\epsilon = 0.005$ for a dataset consisting of $n = 1,000$ points, the required minimum k has to be higher than 2.8×10^6. But as can be seen in Fig. 2, the same level of accuracy can be achieved using $k = 65,536$, which is used in all of our experiments.



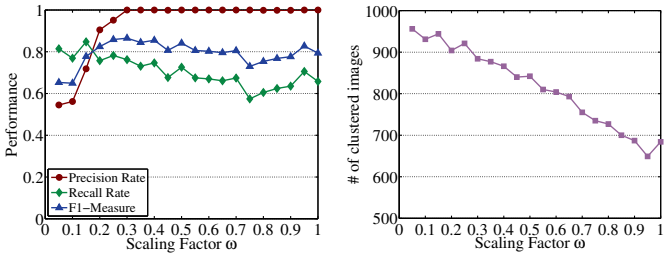
(a) (b)

Fig. 3: ROC curves obtained by varying a threshold from -1 to 1 and comparing it to the pairwise correlations of camera fingerprints in dataset $\mathcal{D}_1$ (a) and $\mathcal{D}_3$ (b).

[2]It states that let Q be a set of n unit-normal points in $\mathbb{R}^d$ and $\boldsymbol{R}$ be a random matrix with i.i.d. entries drawn from $\mathcal{N}(0,1)$. Then for all $\boldsymbol{u}, \boldsymbol{v} \in Q$, $\Pr(|\langle \boldsymbol{u}, \boldsymbol{v} \rangle - \langle \boldsymbol{Ru}, \boldsymbol{Rv} \rangle| \geq \epsilon) \leq n^2 e^{(2-(\epsilon^2-\epsilon^3)\frac{k}{4})}$.

The second parameter is the binarization threshold t_b. For binarizing the accurate correlation matrix in the fine clustering stage, we set $t_b = 0.005$ to give a theoretical false positive rate $\mathcal{Q}(d * t_b) \approx 1.5 \times 10^{-7}$, where $\mathcal{Q}$ is the complementary cumulative density function of the standard normal distribution $\mathcal{N}(0, 1)$. However, the actual false positive rate can be much higher due to the presence of non-unique artifacts. To see this, we obtained the ROC curves by varying a threshold from -1 to 1 and comparing it with the pairwise correlations of camera fingerprints in $\mathcal{D}_1$ and $\mathcal{D}_3$, as shown in Fig. 3a and Fig. 3b, respectively. The false positive rate is approximately 1×10^{-4} for the easy dataset $\mathcal{D}_1$ and can be even higher for the hard dataset $\mathcal{D}_3$. To compensate for the errors introduced by random projection, we increased t_b to 0.008 for binarizing the entries of the approximate correlation matrix in the coarse clustering stage. As shown in Fig. 3, $t_b = 0.008$ gives a false positive rate around 2.5×10^{-2}. Note that the much larger false positive rate in the coarse clustering does not necessarily result in a low precision rate of the final clustering result, because the coarse clustering is followed by the accurate fine clustering. To give a practical reference, the curves corresponding to thresholds in $[0.004, 0.006]$ for the original fingerprint and thresholds in $[0.006, 0.01]$ for the dimension-reduced fingerprint are highlighted in yellow.

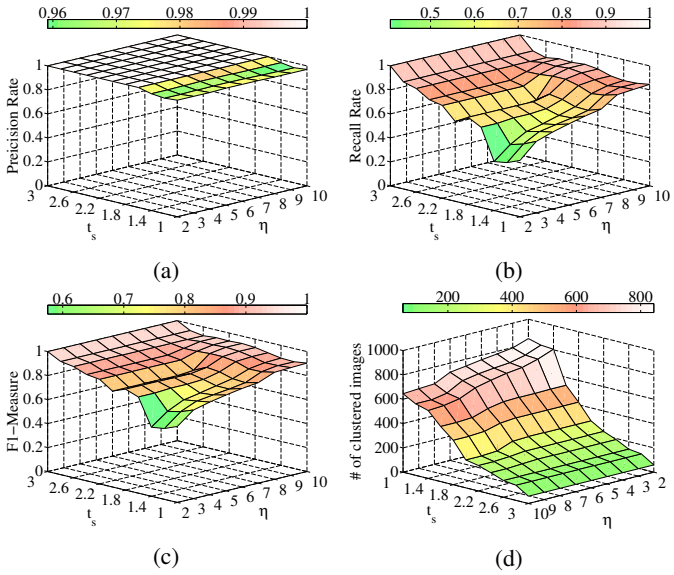
The third parameter is the scaling factor ω of the adaptive threshold τ in Equation (10). It actually determines a point that lies between the means of intra-class and inter-class distributions. To see how ω affects the clustering results, we conducted experiments on dataset $\mathcal{D}_1$. As shown in Fig. 4a, a high ω reduces the false attribution error, and therefore gives rise to the precision rate. But the drawback of a high ω is that it produces many small clusters with a size less than η, which excludes a large amount of images from the final results. As can be seen in Fig. 4b, the number of clustered images (i.e., the images in clusters with a size larger than η) decreases as ω goes up. We found that an ω in the range of $[0.3, 0.5]$ strikes a good balance between the performance and the number of clustered images. ω is set to 0.45 in our experiments.



(a) (b)

Fig. 4: Impact of ω on (a) the clustering performance and (b) the number of images included in the final results.

The fourth parameter is the score threshold t_s in Equation (7). Only the sub-clusters with a score larger than t_s are collected for further use in the merge and attraction stages. Since the score ξ is a function of the cluster size, t_s is therefore closely related with the minimal cluster size η. The effect of these two parameters on the performance will be investigated jointly. As could be expected, these two parameters are more sensitive to datasets with small average class size. Therefore, we clustered the challenging dataset $\mathcal{D}_3$ (with a small average class size of 20) using different combinations of $t_s \in [1, 3]$ and $\eta \in [2, 10]$. As can be seen in Fig. 5, the larger t_s, the better clustering performance. But the trade-off is that a higher t_s makes small sub-clusters less likely to be collected for further investigation, which limits the capability of discovering small classes and excludes the majority of the images in the final results, as can be seen in Fig. 5d. t_s is therefore set to $\sqrt{2}$ in our experiments to ensure a good capability of discovering small classes. If applicable, η can be set according to the prior information, such as the average class size, of the dataset. Otherwise, it is advised to set it to a value that is large enough for constructing a reliable cluster centroid. We set $\eta = 5$ in our experiments.



(a) (b) (c) (d)

Fig. 5: How the score threshold t_s and the size η of the minimal cluster affect the clustering results. (a) Precision rates. (b) Recall rates. (c) F1-Measures. (d) Number of clustered images.

C. Experimental investigations

We conducted a series of experiments to investigate: 1) the superiority of the potential-based eviction over the random eviction, 2) the effectiveness of the adaptive threshold τ, 3) the capability of conquering the $NC \gg SC$ problem, and 4) the comparison with other clustering algorithms on both small and large-scale datasets.

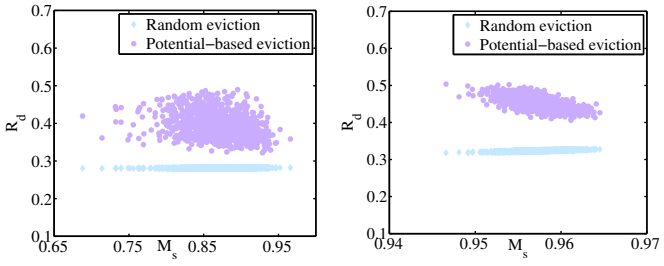
1) Superiority of the potential-based eviction: To demonstrate the advantages of the potential-based eviction (please refer to Equation (6)), we compared it with the random eviction, which is exactly the same as the potential-based eviction except that all fingerprints in the previous batch, rather than those with lower potentials, will be evicted from the RAM before loading the next batch. Suppose $n = 5,000$ fingerprints were equally divided into 10 batches (batch size $q = 500$), which will be sequentially loaded into the RAM. Two synthetic experiments were conducted: in the first experiment, the class

number c was fixed to 20 to simulate the scenario where the $NC \gg SC$ problem is absent. $1,000$ class distributions were randomly generated under the constraint that the sum of the class sizes equals $5,000$. Based on the limited correlation calculations and I/O operations, a good eviction strategy can well explore the connectivity information of the graph consisting of fingerprints. Therefore, we used R_d, defined as the ratio of the number of discovered edges to the total number of edges of the fingerprint graph, to measure the effectiveness of eviction strategies. Since the entropy is a good indicator of the symmetry of one distribution, we used the ratio of the entropy of the class distribution to that of the uniform distribution as the symmetry measurement M_s. Specifically,

$$M_s = \frac{-1}{n \log_2 c} \sum_{i=1}^{c} n_i \log_2 \frac{n_i}{n}, \tag{19}$$

where n_i is the number of fingerprints in the ith class, n is the total number of fingerprints, and c is the number of classes. The higher the M_s, the more symmetric the distribution, with 1 indicating the uniform distribution. In the second experiment, the same procedure was repeated with a different $c = 500$ to simulate the scenario where the number of classes is higher than the average class size (i.e., the $NC \gg SC$ problem is present).

Results of the two experiments are shown in Fig. 6a and 6b, respectively. As can be seen, R_d of the random eviction stays around the theoretical value $(3nq - 2q^2 - n)/(n^2 - n) \approx 0.28$, while R_d of the potential-based eviction are consistently higher than those of the random eviction. As M_s decreases, the class distribution becomes more unbalanced and the fingerprints from larger classes tend to have higher potentials. As a result, the edges connecting the fingerprints from larger classes are more likely to be discovered. Therefore, the advantage of the potential-based eviction grows as M_s decreases, as more clearly shown in Fig. 6b.



(a) (b)

Fig. 6: Superiority of the potential-based eviction over the random eviction. (a) $c = 20$. (b) $c = 500$.

2) Effectiveness of the adaptive threshold: To investigate the effectiveness of the adaptive threshold τ in Equation (10), we randomly chose 900 images taken by 3 cameras (each responsible for 300 images), namely one Agfa DC-830i (referred as *Cam 1*) and two Kodak M1063s (referred as *Cam 2* and *Cam 3*), to simulate the inter-class and intra-class correlation distributions. Three sets of values of n_x and n_y, i.e., $(n_x = 1, n_y = 10)$, $(n_x = 10, n_y = 30)$ and $(n_x = 30, n_y = 50)$, were tested. We first excluded the 300 images acquired by *Cam 3* to simulate the case that two cameras are of different models. The NCC ρ between two centroids averaged over n_x and n_y camera fingerprints, which were randomly selected from the 300 images of *Cam 1* and the other 300 images of *Cam 2*, respectively, was calculated as one inter-class correlation. To prevent the same fingerprint from contributing to both centroids, the 300 images of *Cam 2* were equally divided into two groups. n_x and n_y images were randomly selected from the two groups and the correlation ρ between the corresponding two centroids was calculated as one intra-class correlation. As indicated in Equation (11), σ_x^2 and σ_y^2 were estimated from the pairwise correlations of the camera fingerprints that were used to estimate the two centroids, respectively. We repeated the above procedure $2,000$ times to generate $2,000$ inter-class correlations and $2,000$ intra-class correlations, which were used to draw the inter-class distribution and intra-class distribution, respectively. To simulate the case where two cameras are of the same model, we replaced the images of *Cam 1* with those of *Cam 3* and repeated the above procedure again. Comparison with the other two adaptive thresholds proposed in [35] and [36] were also performed. For the threshold proposed in [35], we set $n = \max(n_x, n_y)$ to make it applicable to more general cases.

As illustrated in the first row of Fig. 7, the inter-class distribution remains unchanged with regard to the increasing n_x and n_y, while the intra-class distribution keeps shifting towards the right. The threshold proposed in [35] seems to be working well for large n, but closer inspection reveals that it almost stays the same with regard to the increasing n. As a consequence, it is either too conservative for large n or overly aggressive for small n. Although the threshold proposed in [36] also shifts to the right, its shift is too aggressive, making some of the intra-class correlations misclassified as inter-class correlations. In fact, the unsatisfactory performance of the thresholds in [35] and [36] is to be expected because they only consider the number of fingerprints and ignore the fact that the correlation also heavily depends on the quality of fingerprints. That is the key reason why the threshold τ in Equation (10) is capable of adaptively finding a suitable breaking point between the inter-class distribution and the intra-class distribution.

Similar results can be observed in the second row of Fig. 7, but the inter-class distribution moves to the right as n_x and n_y increase due to the non-unique artifacts shared by devices of the same model. What is worse, in practice it is possible that the fingerprints used to calculate the centroid are not solely from the same class due to the potential errors, which probably lengthens the right tail of the inter-class distribution and the left tail of the intra-class distribution. These two phenomenons make the two distributions more likely to overlap and therefore complicate the situation. A threshold indicated in Equation (10) approximately divides the margin evenly between the inter-class distribution and the intra-class distribution, so it can be expected to deliver a good performance in practice.

3) Capability of conquering the $NC \gg SC$ problem: To simulate the $NC \gg SC$ problem, we cropped 50 image blocks sized 1024×1024 at 50 different locations from each image in $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}_4$. Since the SPN is location-based, such a simple and efficient way of image cropping simulates the

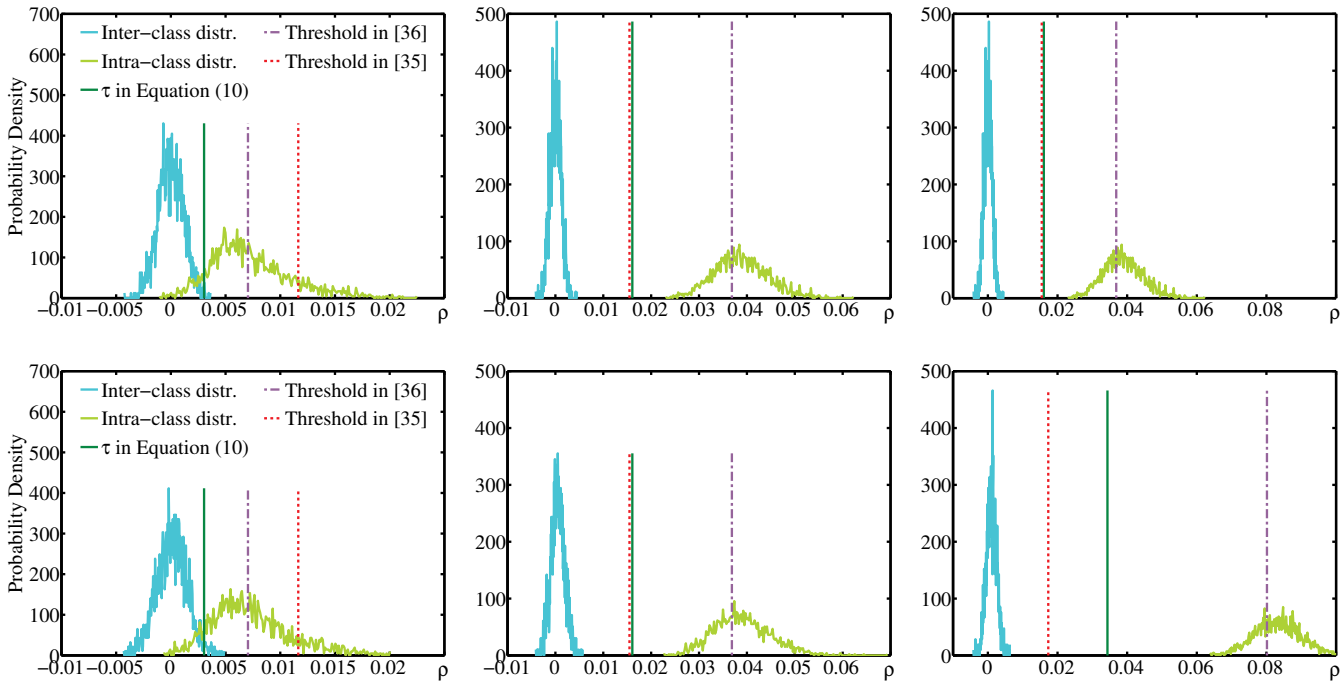


Fig. 7: Comparison of the effectiveness of thresholds. The first and second rows show the results for two cases, i.e., cameras are of different models or the same model, respectively. From left to right, the columns show the results for four sets of values, i.e., $(n_x = 1, n_y = 10)$, $(n_x = 10, n_y = 30)$ and $(n_x = 30, n_y = 50)$, respectively.

process of generating images acquired by different sensors. For example, cropping 50 image blocks at different locations from each of the 40 images taken by one camera results in $2,000$ image blocks taken by 50 cameras (each accounting for 40 image blocks). Notice that the images blocks cropped from the same image undergo the same in-camera processing procedures, which possibly introduces non-unique artifacts and therefore makes the clustering task on the generated datasets even more challenging. The datasets generated from $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}_4$ are referred to as $\mathcal{D}'_1$, $\mathcal{D}'_2$, $\mathcal{D}'_3$ and $\mathcal{D}'_4$, respectively, each of which consists of $50,000$ image blocks.

TABLE I: Clustering results on the generated datasets

Dataset	n_g	n_d	Iteration				
			1	2	3	4	5
$\mathcal{D}'_1$	1,250	1,145	586	445	253	58	–
$\mathcal{D}'_2$	1,250	1,157	601	451	216	67	–
$\mathcal{D}'_3$	2,500	1,982	637	532	606	304	108
$\mathcal{D}'_4$	2,500	1,912	638	529	568	279	146

The clustering results are presented in Table I, where n_g is the ground-truth class number, n_d is the number of the discovered *unique* classes with a size no smaller than η in the final clustering results, and the remaining columns show the numbers of the discovered *unique* classes in each iteration. As can be seen, the clustering is finished in a few iterations and the number of the discovered unique classes decreases significantly after the first few iterations. For the easy datasets $\mathcal{D}'_1$ and $\mathcal{D}'_2$, about $92\%\sim93\%$ of the classes can be discovered and the clustering can be finished in less iterations. But for the two more challenging datasets $\mathcal{D}'_3$ and $\mathcal{D}'_4$, only $76\%\sim79\%$ of the classes are discovered mainly due to the small classes in them. Certainly, the number of the discovered unique classes can be increased by specifying a smaller η, but given the results in the first column of Fig. 7, if the cluster size is small, it is more likely that the two distributions are overlapped, which gives rise to false positives or false negatives even an appropriate threshold is chosen. Therefore, if the discovered clusters are about to be used for further merging or query (e.g., retrieving the similar clusters for the new incoming fingerprints), it is advisable to specify a reasonable high η so as to improve the reliability of the system. So it is important to investigate the performance of the proposed framework when η is set to a relatively high value.

Further investigations reveal that at least three reasons account for the difficulties of discovering small classes:

(a) Although the images are rotated to ensure the same horizontal orientation, they may still be incorrectly rotated (e.g., supposed to be rotated clockwise but rotated anti-clockwise). As a result, images from the same class are split into two classes, making small classes even more difficult to be identified.

(b) A threshold t_b is specified to binarize the correlations. In order to guarantee the high precision rate of the clusters, t_b should be large enough. But the side effect is that some fingerprints of the same class will be considered as unrelated due to the insufficiently large correlations, which has a more severe impact on small classes due to the limited number of intra-class correlations.

(c) Smaller classes are more sensitive to misclassification. Considering two classes, one consisting of five fingerprints and the other consisting of ten fingerprints, if four

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.9756	0.2367	0.5080	**0.9980**
$\mathcal{R}$	0.8205	**0.9833**	0.8467	0.7320
$\mathcal{F}$	**0.8914**	0.3816	0.6350	0.8442
n_d/n_g	**25**/25	6/25	13/25	**25**/25

(a)

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.9533	0.2605	0.5577	**0.9879**
$\mathcal{R}$	0.8500	**1.0000**	0.8424	0.7382
$\mathcal{F}$	**0.8987**	0.4134	0.6711	0.8446
n_d/n_g	**25**/25	5/25	15/25	**25**/25

(b)

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.8099	0.0600	0.8262	**0.9940**
$\mathcal{R}$	0.7538	**1.0000**	0.6681	0.6023
$\mathcal{F}$	**0.7809**	0.1132	0.7388	0.7499
n_d/n_g	42/50	3/50	**45**/50	43.3/50

(c)

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.7027	0.0301	0.9478	**0.9941**
$\mathcal{R}$	0.7942	**1.0000**	0.7236	0.6853
$\mathcal{F}$	0.7456	0.0584	**0.8207**	0.8111
n_d/n_g	31/50	1/50	**46**/50	42.2/50

(d)

TABLE II: Comparison of 4 different clustering algorithms on 4 different datasets: (a) $\mathcal{D}_1$, (b) $\mathcal{D}_2$, (c) $\mathcal{D}_3$, and (d) $\mathcal{D}_4$.

fingerprints in both the first class and the second class are wrongly classified, the first class will disappear from the final clustering results, while there is still a chance to form clusters for the second class.

It is the interplay of these reasons that makes discovering small classes in large-scale databases very challenging. Considering the fact that the size of classes in $\mathcal{D}_3'$ and $\mathcal{D}_4'$ can be as small as 10 or 20, and η is set to a relatively high value 5, $76\%\sim79\%$ is a very promising result.

4) Comparison with other clustering algorithms: In this section, we will compare the proposed clustering framework with other camera fingerprints clustering algorithms. Apart from $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{F}$, we will also show the ratio of the number of discovered *unique* clusters to the ground-truth class number, i.e., n_d/n_g. It was observed in our experiments that the algorithm in [12] performs better and slightly faster than that in [11], so we will use [12] to represent the hierarchical clustering based algorithms. For the sake of convenience, we refer to Li's Markov random field based algorithm [8] as MRF, Liu's multi-class spectral clustering based algorithm [9] as SC, and Villalba's hierarchical clustering based algorithm [12] as HC. It is worth mentioning that $1/5$ of the entire dataset is used as the membership committee in the first iteration of MRF to reduce the computational cost. For the proposed algorithm, the whole dataset is randomly partitioned into batches of equal size in the coarse clustering stage, so as to give convincing results, we repeated the experiments 10 times and showed the average results for the proposed algorithm.

In the first experiment, four clustering algorithms were tested on the four fixed-size small datasets, namely $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}_4$. Results are shown in Table II, where the highest value in each row is highlighted in bold. As can be seen, SC performs worst among the four algorithms in terms of the F1-measure. The underlying reasons is that SC terminates when the size of the smallest cluster equals 1. Due to the intrinsic characteristics of the fingerprint, if one camera fingerprint is severely contaminated, it can easily be regarded as unrelated to all other fingerprints and result in a singleton cluster. Such premature termination happens more often when the size of class is small. As a consequence, the number of unique clusters n_d discovered by SC is significantly lower than that of the other three algorithms. MRF has the best performance on the easy datasets, but its performance degrades on the hard datasets, where the fingerprints of the cameras of the same model are more ambiguous to the clustering algorithms. Surprisingly, HC performs worse on the easy datasets than on the hard datasets. We looked into the clusters generated on the two easy datasets and found that there are several large clusters containing the fingerprints from several cameras. So the rather contradictory results are caused by the incorrect agglomeration at an earlier stage, which consequently misleads the succeeding agglomerations. For the proposed clustering algorithm, the overall performance is balanced and stable across different datasets. The precision rate of the proposed algorithm keeps staying at $98\%\sim100\%$ at the expense of a slightly lower recall rate, around $61\%\sim73\%$. This consequently retains the F1-measure of the proposed algorithm at a favorable level. But because of the reasons mentioned in Section V-C3, some classes in $\mathcal{D}_3$ and $\mathcal{D}_4$ are missing in the final results.

In the second experiment, we aim to compare the time complexities and the clustering qualities of the four algorithms on various-size datasets. To generate datasets of various sizes, we incrementally added $1,000$ images captured by 10 cameras (100 images per camera) to an empty dataset until all the 74 cameras in the Dresden database had been covered. The four algorithms were run and evaluated on each of these datasets. For the sake of completeness, we also evaluated the algorithms on the whole Dresden database, i.e., $15,840$ images.

The running time (in seconds) is shown in Fig. 8a. Since MRF, SC and HC require the calculation of pairwise correlations before clustering, the time used to load fingerprints from the disk and calculate the pairwise correlations is highlighted in green in the stacked bar. For the proposed clustering framework, the time used to load the projection matrix and calculate random projection is highlighted in pink in the stacked bar. What can be observed in Fig. 8a is that MRF is most time-consuming, followed by HC and SC. One interesting observation comes from MRF, for which the running time significantly

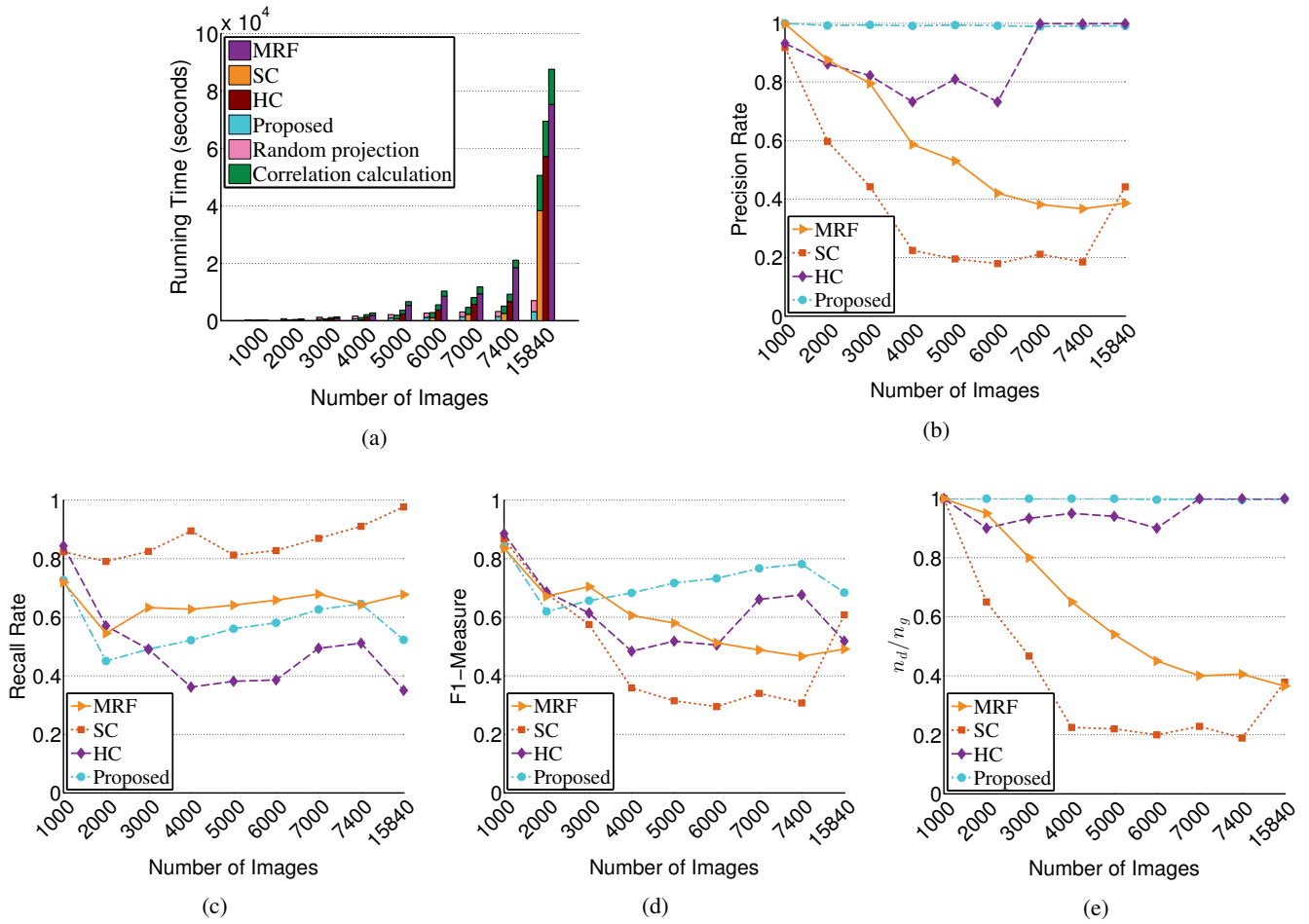


Fig. 8: Running times and clustering qualities of different clustering algorithms on datasets with various sizes. (a) Running time (in seconds). (b) Precision rates. (c) Recall rates. (d) F1-Measures. (e) n_d/n_g.

increases by almost 100% when the image number slightly increases from $7,000$ to $7,400$. It was found that the required iteration number for the algorithm to converge becomes much higher for the slightly larger dataset, so the running time of MRF not only depends on the size but also the characteristics of the dataset, such as the quality of fingerprint and the class distribution. By applying the dimension reduction technique and the divide-and-conquer strategy, the running time of the proposed clustering framework is much lower than those of all the other three algorithms. For example, it only requires about 45 minutes to cluster the whole Dresden database on our ordinary desktop machine.

The precision rates, recall rates, F1-measures, and the ratios n_d/n_g are illustrated in Fig. 8b, 8c, 8d, and 8e, respectively. Compared with the results given in Table II, the performance of SC becomes better on the datasets containing large classes, because with more fingerprints in each class, it has a lower chance to trigger the premature termination. But compared with the other three algorithms, SC is still the worst algorithm and far from satisfactory. MRF is quite stable in terms of the recall rate, but the precision rate witnesses a remarkable and steady decrease as the image number increases, making it unsuitable for large-scale databases. For HC, both the number of discovered unique clusters and the precision rate stay at a high level, but its low recall rate indicates that it tends to produce small-size clusters. Moreover, the sudden increase of the precision rate of HC also indicates its instability across different datasets. While for the proposed algorithm, as the size of dataset increases, it not only achieves high and stable performance, but also discovers all the classes in the database. The high precision rate and the good capability of discovering classes makes the proposed algorithm attractive in many practical scenarios.

VI. Conclusion

In this work, a novel clustering framework, without a training process, has been developed to address the $NC \gg SC$ problem commonly encountered when clustering large-scale images based on camera fingerprints. By continually updating the qualities of clusters and applying the adaptive thresholding, the proposed framework works in a divide-and-conquer manner, and thus can be adopted to large-scale camera fingerprint databases. In comparison with the state-of-the-art camera fingerprints clustering algorithms, the proposed algorithm is much faster and delivers good clustering quality, especially for large databases. The high precision rate makes the proposed

framework attractive in many practical applications such as information searching and retrieval. However, given the results in Table I and II, some small classes cannot be discovered by the proposed algorithm due to the improper binarization and the potential misclassification errors. In fact, because of the nature of the noise-like camera fingerprints, it is hard to solve this issue especially in large-scale databases and further studies are needed to work around it. Another line of our future work will be the parallelization of the proposed framework. Different stages of the proposed framework, such as the calculation of the approximate correlation matrix, the fine clustering of the coarse clusters, and the centroid attraction can be processed in parallel. We believe that our research will serve as the effective tool for clustering large-scale camera fingerprints.

APPENDIX A
Derivation of Correlation Distribution

To simplify the derivation of the correlation distribution, some assumptions have to be made. Following the assumptions in [42], we assume that each d-dimensional fingerprint has the same quality and is standardized to have zero mean and unit variance before calculating the correlation. Presumably, each standardized fingerprint can be modeled as the sum of the true SPN f_i and other interferences α_i:

$$\boldsymbol{F}[i] = f_i + \alpha_i, \;\; i = 1, 2, 3, ..., d, \tag{20}$$

where d is the length of the estimated fingerprint, f_i follows a normal distribution $\mathcal{N}(0, \sigma^2)$ and $\alpha_i \sim \mathcal{N}(0, 1-\sigma^2)$ is White Gaussian Noise (WGN). For convenience, σ^2 will be referred to as the quality of the true SPN, while $1-\sigma^2$ will be referred to as the level of the interferences in the standardized fingerprint. We further assume that the fingerprints of two different cameras are independent. The normalized cross correlation (NCC) ρ between two fingerprints or centroids, $\boldsymbol{X}$ and $\boldsymbol{Y}$, is given in Equation (3), where $\hat{\boldsymbol{X}}[i]=x_i+\alpha_i$, $\hat{\boldsymbol{Y}}[i]=y_i+\beta_i$, $x_i \sim \mathcal{N}(0, \sigma_x^2)$, $\alpha_i \sim \mathcal{N}(0, 1-\sigma_x^2)$, $y_i \sim \mathcal{N}(0, \sigma_y^2)$, and $\beta_i \sim \mathcal{N}(0, 1-\sigma_y^2)$. Therefore, we can rewrite Equation (3) as

$$\rho(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{d}\sum_{i=1}^{d}(x_i y_i + \alpha_i y_i + \beta_i x_i + \alpha_i \beta_i). \tag{21}$$

When it comes to the situation of determining whether to merge two clusters, we will consider the two centroids averaged over the n_x fingerprints in one cluster and the n_y fingerprints in the other cluster, respectively. Next, we will derive the distribution of inter-class correlation and intra-class correlation in different scenarios.

A. *Scenario 1:* $n_x{=}n_y{=}1$, $\sigma_x^2 \neq \sigma_y^2$

In this scenario, we assume the qualities of the true SPN in $\boldsymbol{X}$ and $\boldsymbol{Y}$ are different. Notice that it does not conflict with the assumption that the true SPNs of the same class are of the same quality. Because even the qualities of the true SPN in all individual fingerprints are the same, when different numbers of fingerprints are averaged to estimate the centroids, the qualities of the true SPN in the resultant centroids may vary significantly. In this scenario, although $\boldsymbol{X}$ and $\boldsymbol{Y}$ are referred to as two fingerprints with different qualities, they can actually be viewed as two centroids averaged over two clusters with different numbers of fingerprints. Under this circumstance, we assume $x_i \sim \mathcal{N}(0, \sigma^2)$, $\alpha_i \sim \mathcal{N}(0, 1-\sigma^2)$, $y_i \sim \mathcal{N}(0, \lambda\sigma^2)$ and $\beta_i \sim \mathcal{N}(0, 1-\lambda\sigma^2)$. For two fingerprints of different cameras, using the Central Limit Theorem (CLT), $\rho(\boldsymbol{X}, \boldsymbol{Y})$ approaches to a normal distribution $\mathcal{N}(0, 1/d)$ when $d \to \infty$. But if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are of the same camera, we have $y_i = \sqrt{\lambda} x_i$. Therefore, Equation (21) can be rewritten as

$$\rho(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{d}\sum_{i=1}^{d}(\sqrt{\lambda}x_i^2 + \sqrt{\lambda}\alpha_i x_i + \beta_i x_i + \alpha_i \beta_i). \tag{22}$$

It is known that $x_i \sim \mathcal{N}(0, \sigma^2)$, therefore x_i^2/σ^2 follows the Chi-squared distribution with 1 degree of freedom $\chi^2(1)$. We can easily obtain the mean and variance for x_i^2: $E[x_i^2]{=}\sigma^2$, $Var[x_i^2]{=}2\sigma^4$. Based on the assumption that x_i, α_i, and β_i are mutually independent, we can easily derive the mean and variance of the ith element as

$$\begin{cases} E[\sqrt{\lambda}x_i^2 + \sqrt{\lambda}\alpha_i x_i + \beta_i x_i + \alpha_i \beta_i] = \sqrt{\lambda}\sigma^2 \\ Var[\sqrt{\lambda}x_i^2 + \sqrt{\lambda}\alpha_i x_i + \beta_i x_i + \alpha_i \beta_i] = 1 + \lambda\sigma^4. \end{cases} \tag{23}$$

According to CLT, when $d \to \infty$,

$$\rho(\boldsymbol{X}, \boldsymbol{Y}) \xrightarrow{d} \mathcal{N}(\mu_1, \Sigma_1), \tag{24}$$

where

$$\begin{cases} \mu_1 = \sqrt{\lambda}\sigma^2 \\ \Sigma_1 = (1 + \lambda\sigma^4)/d. \end{cases} \tag{25}$$

B. *Scenario 2:* $n_x > 1$, $n_y > 1$, $\sigma_x^2 \neq \sigma_y^2$

Suppose $\boldsymbol{X}$ and $\boldsymbol{Y}$ are two centroids generated by averaging n_x fingerprints and n_y fingerprints, respectively. In this more complicated and general scenario, if we can figure out the qualities of the true SPN in $\boldsymbol{X}$ and $\boldsymbol{Y}$, then we can use the conclusion of Scenario 1 to determine the distribution of the correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$. For the centroids from two different classes, we still have the same conclusion $\rho(\boldsymbol{X}, \boldsymbol{Y}) \xrightarrow{d} \mathcal{N}(0, 1/d)$. For two centroids from the same class, we assume that the qualities of the true SPN in all individual fingerprints in the same cluster are the same, σ^2. If n such fingerprints are averaged before standardization, the quality of the true SPN, σ^2, remains unchanged, but the level of interferences declines to $(1-\sigma^2)/n$. So after standardization, the quality of the true SPN in the centroid becomes

$$\frac{\sigma^2}{\sigma^2 + (1-\sigma^2)/n} = \frac{n\sigma^2}{(n-1)\sigma^2 + 1}. \tag{26}$$

Replacing n in Equation (26) with n_x and n_y, σ^2 with σ_x^2 and σ_y^2 yields the distributions for the ith element of the two centroids:

$$\begin{cases} x_i \sim \mathcal{N}(0, n_x\sigma_x^2/[(n_x-1)\sigma_x^2 + 1]) \\ \alpha_i \sim \mathcal{N}(0, (1-\sigma_x^2)/[(n_x-1)\sigma_x^2 + 1]) \end{cases} \tag{27}$$

and

$$\begin{cases} y_i \sim \mathcal{N}(0, n_y\sigma_y^2/[(n_y-1)\sigma_y^2 + 1]) \\ \beta_i \sim \mathcal{N}(0, (1-\sigma_y^2)/[(n_y-1)\sigma_y^2 + 1]). \end{cases} \tag{28}$$

Following the conclusion of Scenario 1, when $d \to \infty$, the distribution of ρ between two centroids of the same class approaches to a normal distribution

$$\rho(\boldsymbol{X}, \boldsymbol{Y}) \xrightarrow{d} \mathcal{N}(\mu_2, \Sigma_2), \tag{29}$$

where

$$\begin{cases} \mu_2 = \sqrt{\frac{n_x n_y \sigma_x^2 \sigma_y^2}{[(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]}} \\ \Sigma_2 = \frac{n_x n_y \sigma_x^2 \sigma_y^2 + [(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]}{[(n_x-1)\sigma_x^2+1][(n_y-1)\sigma_y^2+1]d}. \end{cases} \tag{30}$$

By setting $n_x{=}n_y{=}1$, Equation (30) becomes Equation (25). But in practice, λ varies from different fingerprints, making the intra-class correlation distribution a Gaussian mixture distribution rather than a unimodal Gaussian distribution. The mean indicated in Equation (30) indicates where most of the correlations are scattered around.

ACKNOWLEDGMENT

The authors would like to thank Functional Technologies Ltd, UK, for its support of this work, which has led to a pending UK Patent (Application Number GB1515615.1). The authors would also like to thank the anonymous reviewers for their comments and suggestions that helped improve the presentation of this paper.

REFERENCES

[1] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 2, pp. 205–214, 2006.

[2] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, "Determining image origin and integrity using sensor noise," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 1, pp. 74–90, 2008.

[3] C.-T. Li, "Source camera identification using enhanced sensor pattern noise," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 2, pp. 280–287, 2010.

[4] X. Kang, Y. Li, Z. Qu, and J. Huang, "Enhancing source camera identification performance with a camera reference phase sensor pattern noise," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 393–402, 2012.

[5] X. Kang, J. Chen, K. Lin, and P. Anjie, "A context-adaptive SPN predictor for trustworthy source camera identification," *EURASIP J. Image Video Process.*, vol. 2014, no. 1, pp. 1–11, 2014.

[6] A. Lawgaly, F. Khelifi, and A. Bouridane, "Weighted averaging-based sensor pattern noise estimation for source camera identification," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2014, pp. 5357–5361.

[7] X. Lin and C.-T. Li, "Enhancing sensor pattern noise via filtering distortion removal," *IEEE Signal Process. Lett.*, vol. 23, no. 3, pp. 381–385, March 2016.

[8] C.-T. Li, "Unsupervised classification of digital images using enhanced sensor pattern noise," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 3429–3432.

[9] B. Liu, H.-K. Lee, Y. Hu, and C.-H. Choi, "On classification of source cameras: A graph based approach," in *Proc. IEEE Int. Workshop Inf. Forensics Security*, Dec. 2010, pp. 1–5.

[10] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proc. IEEE Int. Conf. Computer Vision*, 2003, pp. 313–319.

[11] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti, "Fast image clustering of unknown source images," in *Proc. IEEE Int. Workshop Inf. Forensics Security*, Dec. 2010, pp. 1–5.

[12] L. J. G. Villalba, A. L. S. Orozco, and J. R. Corripio, "Smartphone image clustering," *Expert Syst. with Appl.*, vol. 42, no. 4, pp. 1927–1940, 2015.

[13] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[14] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proc. Int. Conf. Knowl. Discovery Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[16] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," in *ACM SIGMOD Record*, vol. 27, no. 2, 1998, pp. 73–84.

[17] S. Guha, R. Rastogi, and K. Shim, "ROCK: a robust clustering algorithm for categorical attributes," in *Proc. Int. Conf. Data Eng.*, 1999, pp. 512–521.

[18] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2, 1996, pp. 103–114.

[19] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[20] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 287–296.

[21] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, 2007.

[22] S. M. Van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, Netherlands, May, 2000.

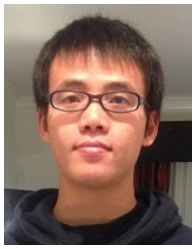
[23] D. Rafiei and A. Mendelzon, "Similarity-based queries for time series data," in *ACM SIGMOD Record*, vol. 26, no. 2. ACM, 1997, pp. 13–25.

[24] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proc. Int. Conf. Foundations Data Organization Algorithms*, 1993, pp. 69–84.

[25] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith, "Efficient retrieval for browsing large image databases," in *Proc. Int. Conf. Inf. Knowledge Management*, 1996, pp. 11–18.

[26] K.-P. Chan and A.-C. Fu, "Efficient time series matching by wavelets," in *Proc. Int. Conf. Data Eng.*, 1999, pp.

126–133.
[27] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge Inf. Syst.*, vol. 3, no. 3, pp. 263–286, 2001.
[28] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
[29] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[30] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *Int. J. Computer Vision*, vol. 70, no. 1, pp. 77–90, 2006.
[31] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[32] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. Computer System Sci.*, vol. 66, no. 4, pp. 671–687, 2003.
[33] I. S. Dhillon, Y. Guan, and B. Kulis, "A unified view of kernel k-means, spectral clustering and graph cuts," Dept. Comput. Sci., Texas Univ., Austin, Texas, Tech. Rep. TR-04-25, February 2004.
[34] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: analysis and an algorithm," *Advances in Neural Inf. Process. Syst.*, vol. 2, pp. 849–856, 2002.
[35] G. J. Bloy, "Blind camera fingerprinting and image clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 532–534, 2007.
[36] J. Eklann, "Source camera classification and clustering from sensor pattern noise," Master's thesis, Chalmers University of Technology, Sweden, 2012.
[37] T. Gloe and R. Böhme, "The 'Dresden Image Database' for benchmarking digital image forensics," *J. of Digital Forensic Practice*, vol. 3, no. 2-4, pp. 150–159, 2010.
[38] T. Gloe, S. Pfennig, and M. Kirchner, "Unexpected artefacts in PRNU-Based camera identification: A 'Dresden Image Database' case-study," in *Proc. ACM Workshop Multimedia Security*, 2012, pp. 109–114.
[39] C.-T. Li and R. Satta, "Empirical investigation into the correlation between vignetting effect and the quality of sensor pattern noise," *IET Computer Vision*, vol. 6, no. 6, pp. 560–566, 2012.
[40] X. Lin and C.-T. Li, "Preprocessing reference sensor pattern noise via spectrum equalization," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 126–140, 2016.
[41] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, "Compressed fingerprint matching and camera identification via random projections," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1472–1485, 2015.
[42] J. Fridrich and M. Goljan, "Derivation of ROCs for composite fingerprints and sequential trimming," Dept. Elect. Comput. Eng., Binghamton Univ., Binghamton, NY, USA, Tech. Rep., January 2010. [Online]. Available: http://www.ws.binghamton.edu/fridrich/Research/rocs.pdf



Xufeng Lin received the B.E. degree in electronic and information engineering from Hefei University of Technology, Hefei, China, in 2009, and the M.E. degree in signal and information processing from South China University of Technology, Guangzhou, China, in 2012. He is currently pursuing his Ph.D. degree in computer science at the University of Warwick, Coventry, U.K. His research interests include digital forensics, multimedia security, machine learning, and data mining.

Chang-Tsun Li received the B.E. degree in electrical engineering from the Chung-Cheng Institute of Technology (CCIT), National Defense University, Bade City, Taiwan, in 1987, the M.Sc. degree in computer science from the Naval Postgraduate School, Monterey, CA, USA, in 1992, and the Ph.D. degree in computer science from the University of Warwick, Coventry, U.K., in 1998. He was an Associate Professor with the Department of Electrical Engineering, CCIT, from 1998 to 2002, and a Visiting Professor with the Department of Computer Science, Naval Postgraduate School, in the second half of 2001. He is currently a Professor with the Department of Computer Science, University of Warwick. He was the Editor-in-Chief of the International Journal of Digital Crime and Forensic from 2009 to 2013. He is currently an Associate Editor of the EURASIP Journal on Image and Video Processing. He has been involved in the organization of many international conferences and workshops, and also served as a member of the International Program Committees for many international conferences. His research interests include biometrics, digital forensics, multimedia security, computer vision, image processing, pattern recognition, evolutionary computation, machine learning, data mining bioinformatics, and content-based image retrieval.