

# Large-Scale Image Clustering Based on Camera Fingerprints

Xufeng Lin, *Student Member, IEEE* 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., number of cameras) is much higher than the average Size of Class (i.e., 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  $NC \gg SC$  issue 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 camera 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. The 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 are 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, but 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 are able to associate different crime scenes and would be in a better position to link these evidences to the seized hardwares that are 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

extracted from one image can be viewed as the noisy version of SPN, we will refer to the true SPN components in the noise residue as the “true SPN”. While the noise residue will be simply referred to as the “camera fingerprint” in the rest of 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 dataset 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 the clustering, the dimension of the fingerprint has to be very large, for example,  $512 \times 512$  pixels or above. The high dimension of 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 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 datasets much more efficient.

The remainder of this paper is organized as follows. In Section II, we will revisit the previous works on image clustering based on camera fingerprints. In Section III, the details of the proposed clustering framework will be presented. Section IV discusses the reasons why the proposed framework can cope with large-scale database. The time complexity and I/O cost of the proposed framework will also be analyzed in this section. 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 is 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 probability in its membership committee. The clustering process stops when there are no label changes after two consecutive iterations. Finally, each fingerprint in the rest of the dataset is classified to its closest cluster. This algorithm performs well on small datasets, 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 fingerprints, so it becomes obviously computationally prohibitive for large-scale datasets. Another limitation is that when the  $NC \gg SC$  problem presents, the size of the training set has to be large enough to guarantee most of the classes are present in the training set. These two limitations make it computationally infeasible even for medium-sized datasets.

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 represented by the similarity between the two fingerprints linked by the edge. To avoid the time-consuming pairwise similarity calculation, a sparse graph is constructed as follows: A vertex is randomly selected as the initial center and its edge weights with all other vertices are calculated. The  $(u + 1)$ th closest vertex to the initial center is then selected as the second center and its edge weights with all other vertices except the first center are calculated, where  $u$  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  $u$ . A multi-class spectral clustering algorithm [10] is then employed on the constructed sparse 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 sparse graph. When it comes to large-scale datasets, the algorithm becomes inefficient due to the high I/O overhead because one fingerprint may need to be read from the disk many times for calculating its similarities with the centers. The time complexity of the algorithm in [10] is  $\mathcal{O}(n^{\frac{3}{2}}\kappa + n\kappa^2)$ , where  $\kappa$  is the number of partitions. The optimal number of partitions is determined by specifying an increasing  $\kappa$  and repeating the spectral clustering until the size of the smallest cluster equals 1. Putting aside the efficiency, the aptness of the manner of finding the optimal partitions number is still an issue for large-scale camera fingerprint datasets.

Another algorithm based on the hierarchical clustering was proposed in [11]. Similar with [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 similar 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. This therefore limits its applicability to large-scale databases.

It can be observed that existing methods either initially 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. 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 threshold set by the user, 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.

### III. PROPOSED 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.  $d$ -dimensional fingerprint is extracted from the center of the green channel of each image and standardized to zero mean and unit variance.
- **Dimension reduction:** Each of the  $n$  extracted fingerprints is projected onto a  $k$ -dimensional subspace ( $k < d$ ) using the *very sparse random projection* proposed in [13].
- **Fingerprint data storage:** Store the full-length fingerprint, the corresponding dimension-reduced fingerprint as well as the image name into a single file.

#### Step 2: Coarse clustering

- **Correlation matrix approximation:** To avoid calculating the  $n \times n$  pairwise correlation matrix  $M$ , 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,  $M$  is replaced with a set of more RAM-efficient lists  $L$ , of which each element  $L_i$  records the indices of the fingerprints that are similar enough to the  $i$ th fingerprint.
- **Coarse partition:** Based on the graph information recorded in  $L$ , the  $n$  fingerprints are coarsely partitioned into  $n_c$  coarse clusters using the fast graph clustering algorithm in [14].

#### 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 one 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 [15].
- **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  $\tau$  (the

element  $\tau_{ij}$  depends on the sizes and the qualities of sub-cluster  $S_i$  and sub-cluster  $S_j$ ). Larger clusters are therefore obtained by merging two or several sub-clusters, meanwhile the centroids and the qualities of the merged clusters are updated accordingly.

#### Step 4: Attraction

- **Spectrum equalization:** Equalize the spectra of the centroids calculated from the merged clusters using the method proposed in [16]. The equalization aims at eliminating the non-unique artifacts to reduce the false positives, so that the fingerprints coming from different devices of the same model or brand will not be mixed up.
- **Centroid attraction:** The equalized centroids are used as “attractors” to attract the unclustered fingerprints left in the database. Consequently, most of the fingerprints belong to the discovered clusters will be absorbed to form larger clusters.

#### Step 5: Post-processing

- **Cluster screening and storage:** Replace the fingerprints in the clusters with a size smaller than a threshold  $\eta$  back into the database and store the remaining clusters as the final or intermediate clusters. The stored information include the centroid, the number of member fingerprints 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  $\eta$  back into the database for the next round of clustering.

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 that 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  $90^\circ$  to ensure all images are horizontally oriented. Although this cannot completely solve the rotation problem, but it indeed successfully limits the freedom of rotation.

After the above preprocessing, the camera fingerprint  $F$  is extracted from the central block of the green channel of the image  $I$ , i.e.,

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

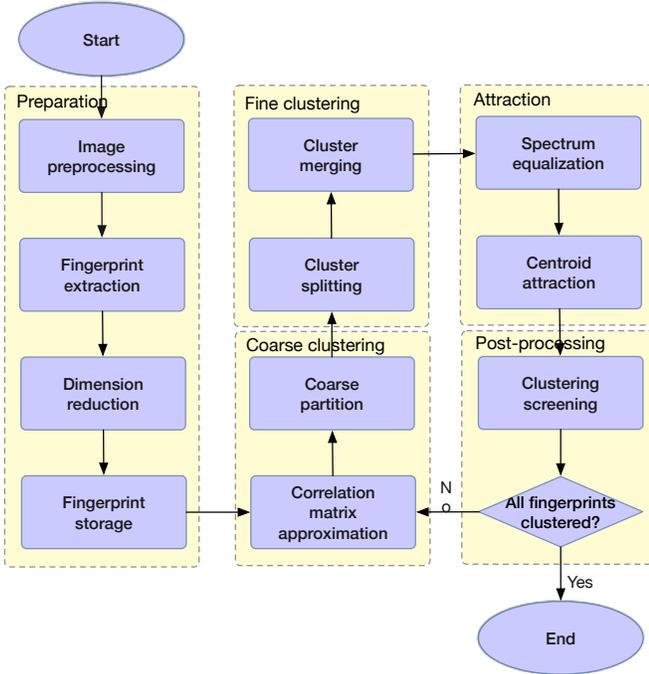


Fig. 1: Flow chart of the proposed framework.

where  $\mathcal{G}$  is the denoising filter described in Appendix A of [1]. The normalized cross correlation (NCC)  $\rho$  is used as the similarity measurement between two fingerprints,  $\mathbf{X}$  and  $\mathbf{Y}$ :

$$\rho(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^d (\mathbf{X}[i] - \bar{\mathbf{X}})(\mathbf{Y}[i] - \bar{\mathbf{Y}})}{\sqrt{\sum_{i=1}^d (\mathbf{X}[i] - \bar{\mathbf{X}})^2} \sqrt{\sum_{i=1}^d (\mathbf{Y}[i] - \bar{\mathbf{Y}})^2}}, \quad (2)$$

where  $d$  is the length of the fingerprint,  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  are the arithmetic mean of  $\mathbf{X}$  and  $\mathbf{Y}$ , respectively. NCC is notorious for its high computational cost due to the calculation 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:

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

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

Standardization also builds up the equivalence between NCC and other commonly-used similarity measures such as the inner product<sup>1</sup> and the Euclidean distance [17]. 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) [18], singular value decomposition (SVD) [19], discrete wavelets transform (DWT) [20] and piecewise aggregate approximation (PPA) [21] are not applicable to our application due to the nature

of noise-like camera fingerprints. More advanced methods, such as principal components analysis (PCA) [22], isometry mapping [23], maximal variance unfolding [24] and locally linear embedding [25] 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* [13, 26] due to its appealing properties of Euclidean distance-preserving, computational efficiency and data independence. Let  $d \times n$  matrix  $\mathbf{D}$  be the set of  $n$  camera fingerprints in the  $d$ -dimensional space  $\mathbb{R}^d$ , and  $\mathbf{R}$  be the  $k \times d$  random projection matrix with  $\mathbf{R}(i, j) = r_{ij}$ , where  $\{r_{ij} | 1 \leq i \leq k, 1 \leq j \leq d\}$  are independent and identically distributed (i.i.d.) random variables drawn from the following probability distribution:

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

where  $s = \sqrt{d}$  according to [13]. The dimension reduction of the  $n$  fingerprints from  $\mathbb{R}^d$  to  $\mathbb{R}^k$  is achieved by a matrix multiplication:

$$\mathbf{E} = \mathbf{R}\mathbf{D}, \quad (5)$$

where  $\mathbf{E}$  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.

### 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. The exhaustive search requires at least  $n(n-1)/2$  pairwise comparisons, which becomes clearly prohibited even for the dimension-reduced fingerprints when  $n$  is large. Even if we have the oracle to know all the entries of the pairwise correlation matrix  $\mathbf{M}$ , it is still an issue to cache all of the entries 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 matrix entries. 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 overhead makes

<sup>1</sup> $\rho(\mathbf{X}, \mathbf{Y}) = \langle \frac{\hat{\mathbf{X}}}{\sqrt{d}}, \frac{\hat{\mathbf{Y}}}{\sqrt{d}} \rangle$

it unaffordable for large-scale databases. Given  $n$  dimension-reduced fingerprints, we 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  $\mathbf{L}$ , of which each element  $\mathbf{L}_i$  is a list recording the indices of the vertices adjacent to the  $i$ th 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. If each dimension-reduced fingerprint is represented by a vertex  $v_i, i = 1, 2, \dots, n$  in a graph  $G$ , we define a *potential* measurement  $p_i$  characterizing its potential connectivities with other vertices:

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

where  $\deg(v_i)$  is the *degree* of vertex  $v_i$  (i.e., the length of  $\mathbf{L}_i$ ) 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 coming batch, the  $q$  vertices with the lowest *potentials* are evicted from the RAM. We refer to this method 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  $\mathbf{L}$ , it is fed into a multilevel graph partitioning algorithm Graclus [14], which works by repeatedly coarsening the original graph (corresponding to  $\mathbf{L}$ ) to a point where very few vertices remain in the graph, then a spectral clustering method [27] 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  $n_c = \sqrt{n}$ , 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 dataset, 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) [15], 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, 28], are actually 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  $\xi$  greater than a predefined threshold  $t_s$  are collected for further use:

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

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  $\xi$  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 [29] and [30] 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  $\rho$  between two  $d$ -dimensional centroids,  $\mathbf{X}$  and  $\mathbf{Y}$ , from different cameras conforms to a normal distribution with zero mean and  $1/d$  variance, i.e.,  $\rho(\mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(0, 1/d)$ . If  $\mathbf{X}$  and  $\mathbf{Y}$  are the centroids of two sub-clusters  $S_x$  and  $S_y$  from the same camera, we can derive that the distribution is also a normal distribution (please refer to the Derivation of Correlation Distribution in the Appendix)

$$\rho(\mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(\mu, \Sigma), \quad (8)$$

where

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

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.  $\sigma_x^2$  and  $\sigma_y^2$  are the qualities of the true SPN in each member fingerprint in  $S_x$  and  $S_y$ . When there is no ambiguity, the 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 but without giving rise to false negatives. An adaptive threshold  $\tau$  characterizes the change in the intra-class distribution is therefore proposed as

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

where  $\omega$  is a predefined scaling factor. Two sub-clusters with a correlation between their centroids higher than  $\tau$  are considered to be from the same camera. In Equation (10),  $\tau$  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,  $\tau$  effectively prevents the cluster merging from propagating errors into the merged clusters.  $\sigma_x^2$  and  $\sigma_y^2$  in Equation (10) can be estimated by calculating the mean of correlations (please refer to  $\mu_1$  in Equation (25) with  $\lambda=1$  in the Appendix). Therefore, when coarse clusters are split into  $n_s$  sub-clusters, the quality of each sub-cluster is initially estimated as

$$\sigma_i^2 = \frac{1}{q_i} \sum_{k=1}^{q_i} \rho_k, \quad i = 1, 2, \dots, n_s, \quad (11)$$

where  $q_i$  is the number of the calculated correlations within  $S_i$ . If  $S_x$  and  $S_y$  are merged into  $\hat{S}$  in the cluster merging stage, the quality of  $\hat{S}$  is updated as

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

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

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

where  $n_i$  and  $\sigma_i^2$  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. One fingerprint  $\mathbf{F}$  belongs to attractor  $\mathbf{A}^*$  if

$$\mathbf{A}^* = \operatorname{argmax}_{\mathbf{A}_i \in \mathcal{A}} \rho(\mathbf{A}_i, \mathbf{F}), \rho(\mathbf{A}^*, \mathbf{F}) > \tau_\rho, \quad (14)$$

where  $\mathcal{A}$  is the set of attractors and  $\tau_\rho$  is calculated from Equation (10). As pointed out in [2], some non-unique artifacts, shared amongst cameras undergo the same or similar in-camera processing procedures, often give rise to false positives. Therefore, we preprocessed the attractors using the spectrum equalization algorithm (SEA) [16] before calculating the correlations with the remaining fingerprints in the database. Finally, the attracted fingerprints belonging to the same cluster are merged into the corresponding attractor and the quality of the merged attractor is updated accordingly. Suppose the attractor  $\mathbf{A}_x$  is averaged over  $n_x$  fingerprints with a quality  $\sigma_x^2$  of the true SPN in each fingerprint, then according to  $\mu$  in Equation (9), the quality  $\sigma_y^2$  for each of the  $n_y$  attracted fingerprints can be estimated as

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

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

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

#### E. Post-processing

Up to 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, the number of member fingerprints, and the quality of the cluster. Notable clusters with a size no smaller than  $\eta$  are stored as the final clusters or

the intermediate clusters that can be used for further merge. While those clusters with a size less than  $\eta$  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 so as to increase the probability 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 dataset is broken down into the clustering of several smaller datasets 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 datasets 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  $n_c = \sqrt{n}$ . 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 [14] approximately has a time complexity  $\mathcal{O}(q|E|/n)$ . In the fine clustering stage, it involves around  $nb d$  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 [15], 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  $\theta$  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, the I/O cost  $\mathcal{O}(2nd + nk)$  is still at an acceptable level and a considerable performance gain 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 [31]. 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 [32] and ensure a consistent size for all fingerprints, camera fingerprints were extracted from the central  $1024 \times 1024$  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-mean and Wiener filtering in the DFT domain, to suppress the non-unique artifacts.

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 common in practical applications that the numbers of images captured by different devices vary 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 nearly cover

all of the popular camera brands, such as Cannon, 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 of 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

$$\mathcal{F} = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{R}}{\mathcal{P} + \mathcal{R}}, \quad (17)$$

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

$$\begin{cases} \mathcal{P} = \frac{\sum_i |o_i|}{\sum_i |c_i|} \\ \mathcal{R} = \frac{\sum_i |o_i|}{\sum_i |\psi_i|}, \end{cases} \quad (18)$$

where  $|c_i|$  is the size of cluster  $c_i$ ,  $|\psi_i|$  is the size of the most frequent class  $\psi_i$  in cluster  $c_i$ , and  $|o_i|$  is the size of the overlap between cluster  $c_i$  and class  $\psi_i$ .

## 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.

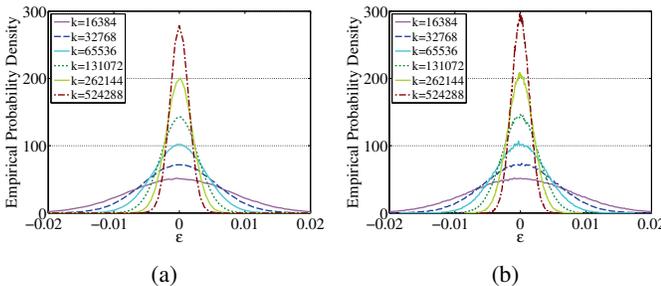


Fig. 2: Distributions of the embedding error  $\epsilon$  in pairwise correlations of  $\mathcal{D}_1$  (a) and  $\mathcal{D}_3$  (b).

The first parameter is the dimension  $k$  of the subspace of random projection. As remarked in [13], we can apply, with a high level of accuracy, the results of *conventional random*

*projection*, i.e., the i.i.d. entries of  $\mathbf{R}$  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  $\epsilon$  in pairwise correlation preservation<sup>2</sup> using *Theorem 4* in [33]. But this gives a very conservative estimation of the minimum  $k$ . We drew the distributions of  $\epsilon$  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. 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  $2.8 \times 10^6$  according to *Theorem 4* in [33]. 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.

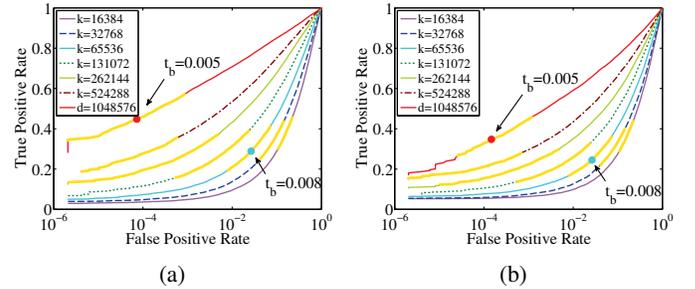


Fig. 3: ROC curves obtained by varying a threshold from  $-1$  to  $1$  and comparing it to the pairwise correlations of camera fingerprints in dataset  $\mathcal{D}_1$  (a) and  $\mathcal{D}_3$  (b).

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  $1 \times 10^{-4}$  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  $2.5 \times 10^{-2}$ . Note that the much larger false positive rate in the coarse clustering does not necessarily results in a low precision rate of the final clustering result, as 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 color.

The third parameter is the scaling factor  $\omega$  of the adaptive

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

threshold  $\tau$  in Equation (10). It actually determines a point that lies between the means of intra-class and inter-class distributions. To see how  $\omega$  affects the clustering results, we conducted experiments on dataset  $\mathcal{D}_1$ . As shown in Fig. 4a, a high  $\omega$  reduces the false attribution error and therefore gives rise to the precision rate. But the drawback of a high  $\omega$  is that it produces many small clusters with a size less than  $\eta$ , 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  $\eta$ ) decreases as  $\omega$  goes up. We found that an  $\omega$  in the range of  $[0.3, 0.5]$  strikes a good balance between the performance and the number of clustered images.  $\omega$  is set to 0.45 in our experiments.

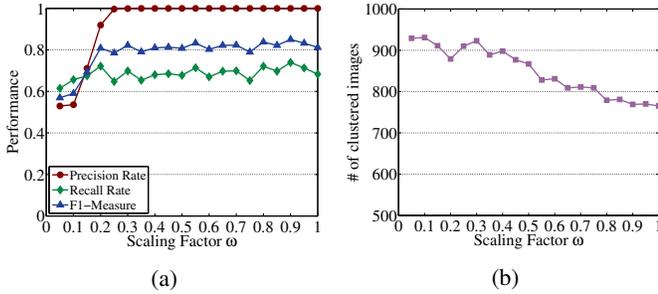


Fig. 4: Impact of  $\omega$  on (a) the clustering performance and (b) the number of images included in the final results.

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  $\xi$  is a function of the cluster size,  $t_s$  is therefore closely related with the minimal cluster size  $\eta$ . 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,  $\eta$  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.

### C. Experimental investigations

We conducted a serial of experiments to investigate: 1) the superiority of the potential-based eviction over the random eviction, 2) the effectiveness of the adaptive threshold  $\tau$ , 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.

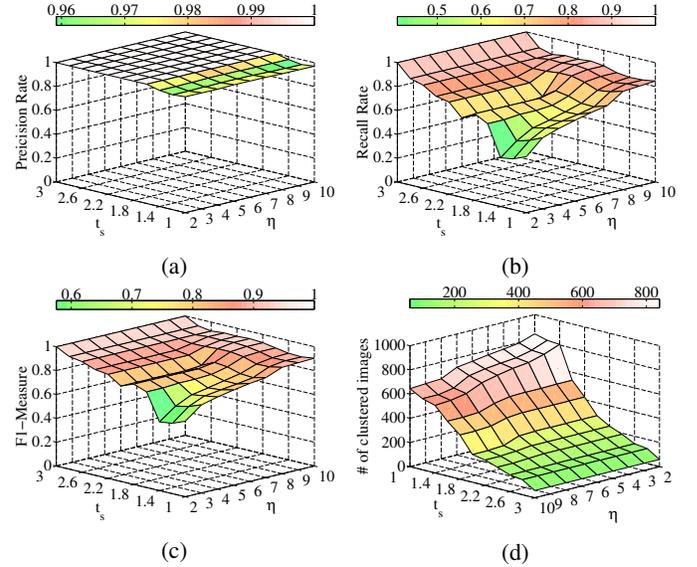


Fig. 5: How the score threshold  $t_s$  and the size  $\eta$  of the minimal cluster affect the clustering results. (a) Precision rates. (b) Recall rates. (c) F1-Measures. (d) Number of clustered images.

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,

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

where  $n_i$  is the number of fingerprints in the  $i$ th 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 stay 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.

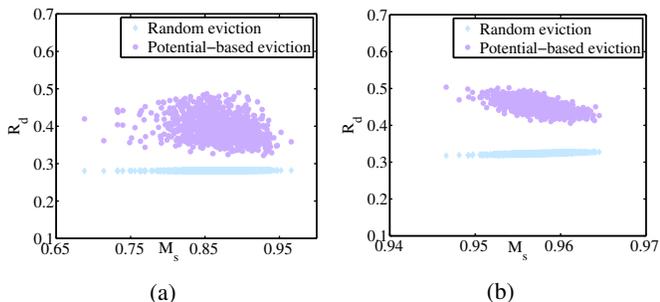


Fig. 6: Superiority of the potential-based eviction over the random eviction. (a)  $c = 20$ . (b)  $c = 500$ .

2) *Effectiveness of the adaptive threshold*: To investigate the effectiveness of the adaptive threshold  $\tau$  in Equation (10), we randomly chose 900 images taken by 3 cameras (each responsible for 300 images), namely 1 Agfa DC-830i (referred as *Cam 1*) and 2 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  $\rho$  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  $\rho$  between the corresponding two centroids was calculated as one intra-class correlation. As indicated in Equation (11),  $\sigma_x^2$  and  $\sigma_y^2$  were estimated from the pairwise correlations of the camera fingerprints that were used to estimate the two centroids, respectively. We repeated the above procedure for 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 [29] and [30] were also performed. For the threshold proposed in [29], we set

$n = \max(n_x, n_y)$  to make it applicable to more general cases.

As can be seen 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 [29] seems 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 [30] 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 [29] and [30] 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  $\tau$  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  $n_y$  increasing 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 purely 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 phenomena 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 have 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  $1024 \times 1024$  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 very well simulates the 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	6
$\mathcal{D}'_1$	1,250	1,151	529	419	265	67	—	—
$\mathcal{D}'_2$	1,250	1,167	501	436	247	67	29	—
$\mathcal{D}'_3$	2,500	2,011	624	506	544	328	178	106
$\mathcal{D}'_4$	2,500	1,943	633	526	519	270	147	72

The clustering results are presented in Table I, where  $n_g$

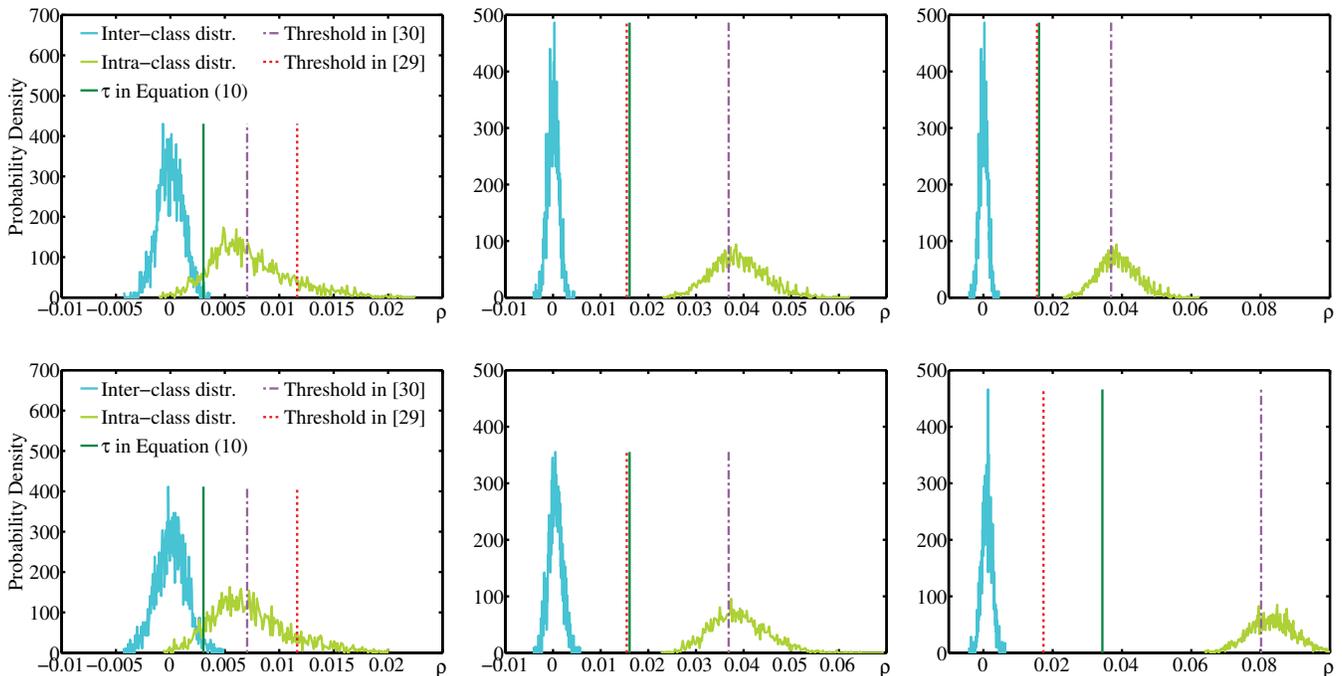


Fig. 7: Comparison of the effectiveness of thresholds. The first and second rows show the results for two cases, i.e., cameras are of different models or the same model, respectively. From left to right, the columns show the results for four sets of values, i.e.,  $(n_x = 1, n_y = 10)$ ,  $(n_x = 10,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

is the ground-truth class number,  $n_d$  is the number of the discovered *unique* classes with a size no smaller than  $\eta$  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  $\mathcal{D}'_2$ , 92%~94% 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 78%~80% of the classes are discovered mainly due to the small classes in them. By all means, the number of the discovered unique classes can be increased by specifying a smaller  $\eta$ , 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 coming fingerprints), it is advisable to specify a reasonable high  $\eta$  so as to improve the reliability of the system. So it is important to investigate the performance of the proposed framework when  $\eta$  is set to a relatively high value.

Further investigations reveal that at least two reasons account for the difficulties of discovering small classes. Firstly, as mentioned in Section III-B, a threshold  $t_b$  is specified to binarize the correlation values. 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 with each other due to the insufficiently large correlation values, which has a more

striking impact on the small classes due to the limited number of intra-class correlation values. How to adaptively select an appropriate binarization threshold for different class is yet to be studied. The second reason is that small classes are more sensitive to misclassification. Imagine a small class consisting of 10 fingerprints. If 9 of the 10 fingerprints are wrongly classified into other clusters, this class will disappear from the final clustering results. It is the interplay of these two 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  $\eta$  is set to a relatively high value 5, 78%~80% is a very promising result.

4) *Comparison with other clustering algorithms*: In this section, we will compare the proposed clustering framework with other camera fingerprint clustering algorithms. It was observed in our experiments that the algorithm proposed in [29] is unable to accomplish the clustering task, so it will be excluded from our subsequent experiments. We also found that the algorithm in [12] performs better and 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. We conducted two separated experiments, one on fixed-size datasets while the other on various-size datasets.

In the first experiment, four clustering algorithms were tested on the four fixed-size small datasets, namely  $\mathcal{D}_1$ ,  $\mathcal{D}_2$ ,

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.9756	0.2367	0.5080	<b>1.0000</b>
$\mathcal{R}$	0.8205	<b>0.9833</b>	0.8467	0.7100
$\mathcal{F}$	<b>0.8914</b>	0.3816	0.6350	0.8304

(a)

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.8099	0.0600	0.8262	<b>0.9878</b>
$\mathcal{R}$	0.7538	<b>1.0000</b>	0.6681	0.6000
$\mathcal{F}$	<b>0.7809</b>	0.1132	0.7388	0.7465

(c)

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.9533	0.2605	0.5577	<b>0.9886</b>
$\mathcal{R}$	0.8500	<b>1.0000</b>	0.8424	0.7373
$\mathcal{F}$	<b>0.8987</b>	0.4134	0.6711	0.8447

(b)

Quality	Clustering algorithms			
	MRF [8]	SC [9]	HC [12]	Proposed
$\mathcal{P}$	0.7027	0.0301	0.9478	<b>0.9878</b>
$\mathcal{R}$	0.7942	<b>1.0000</b>	0.7236	0.7198
$\mathcal{F}$	0.7456	0.0584	0.8207	<b>0.8328</b>

(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)  $\mathcal{D}_4$ .

$\mathcal{D}_3$  and  $\mathcal{D}_4$ . Results are shown in Table II, where the highest value in each row is highlighted in bold font. As can be seen, SC performs worst among the four algorithms in terms of the F-measure. The underlying reason is that the algorithm terminates when the size of the smallest cluster equals 1. For the four datasets we have tested, the size of the smallest class is no more than 40, so it is very likely to form singleton clusters for small classes. Additionally, 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. These two reasons lead to premature termination and weaken the capability of determining the optimal cluster number. MRF has the best performance on the easy datasets, but it is not surprising to see its deteriorating performance on the hard datasets, where the fingerprints of the cameras of the same model are more ambiguous and misleading for the clustering algorithm. Surprisingly, while the clustering results on the easy datasets are generally better than those on the hard datasets, HC performs even worse on the easy datasets. The reasons for the rather contradictory results are probably that the hierarchical clustering algorithm is sensitive to outliers and cannot undo what has been done previously. 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. The undesirable results are caused by the incorrect agglomeration at an earlier stage, which will mislead and spoil 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%~100% at the expense of a slightly lower recall rate, 60%~74%. This consequently retains the F-measure of the proposed algorithm at a favorable level and leads to its superior position in the most challenging dataset  $\mathcal{D}_4$ .

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 have 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 times are shown in Fig. 8a, while the precision rates, recall rates and F1-measures are shown in Fig. 8b, 8c, and 8d, respectively. 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 each of the stacked bar. What can be observed in Fig. 8a is that MRF is most time-consuming, followed by HC and SC. 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. Concerning the quality of clustering, the performance of SC is much better than that on the datasets containing very small classes (e.g.,  $\mathcal{D}_1$ ,  $\mathcal{D}_2$ ,  $\mathcal{D}_3$  and  $\mathcal{D}_4$ ). 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. The sudden increase of the precision rate in Fig. 8b for HC also indicates its instability across different datasets. MRF performs quite stable in terms of recall rate, but the precision rate witnesses a remarkable and steady decrease as the image number increases, making it unsuitable for large-scale databases. The proposed algorithm performs very stable in terms of both the precision and recall rate as the size of dataset increases. Compared with the other three algorithms, the proposed algorithm has the highest precision rate and F1-measure, making it attractive in many practical applications.

## 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 fingerprint. 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

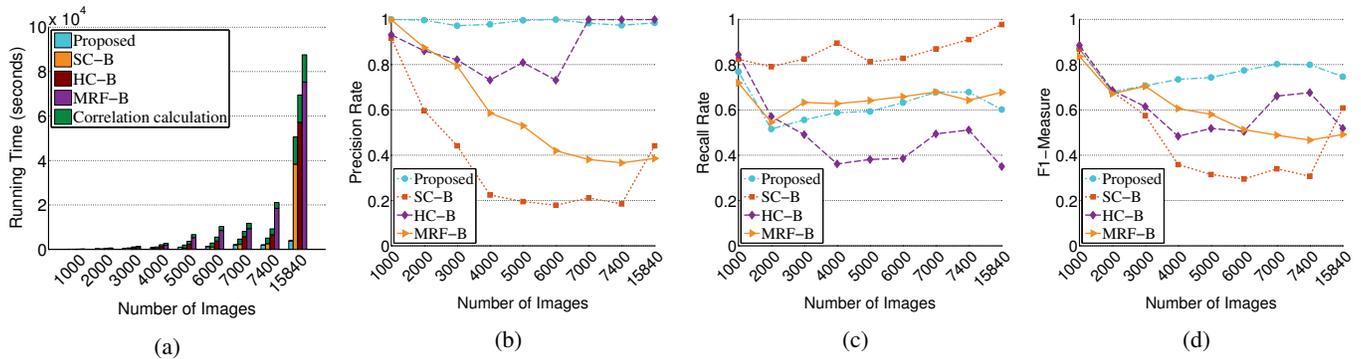


Fig. 8: Running times and clustering qualities of different clustering algorithms on datasets with various sizes. (a) Running time (in seconds). (b) Precision rates. (c) Recall rates. (d) F1-Measures.

camera fingerprints clustering algorithms, it has better and more stable performance especially for large datasets. The high precision rate makes the proposed framework attractive in many practical applications such as information searching and retrieval. However, given the results in Section V-C3, there is a probability that some small classes cannot be discovered due to the improper binarization and the potential misclassification errors. In fact, because of the nature of the noise-like camera fingerprints, it is extremely hard to solve this issue especially in large-scale databases and further studies are needed to work around it. The main line of 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 [34], 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  $\alpha_i$ :

$$\mathbf{F}[i] = f_i + \alpha_i, \quad i = 1, 2, 3, \dots, d, \quad (20)$$

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,  $\sigma^2$  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)  $\rho$  between two fingerprints or centroids,  $\mathbf{X}$  and  $\mathbf{Y}$ , is given in Equation (3), where  $\hat{\mathbf{X}}[i] = x_i + \alpha_i$ ,  $\hat{\mathbf{Y}}[i] = y_i + \beta_i$ ,  $x_i \sim \mathcal{N}(0, \sigma_x^2)$ ,  $\alpha_i \sim \mathcal{N}(0, 1 - \sigma_x^2)$ ,  $y_i \sim \mathcal{N}(0, \sigma_y^2)$ ,

and  $\beta_i \sim \mathcal{N}(0, 1 - \sigma_y^2)$ . Therefore, we can rewrite Equation (3) as

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

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  $\mathbf{X}$  and  $\mathbf{Y}$  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  $\mathbf{X}$  and  $\mathbf{Y}$  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),  $\rho(\mathbf{X}, \mathbf{Y})$  follows a normal distribution  $\mathcal{N}(0, 1/d)$ . But if  $\mathbf{X}$  and  $\mathbf{Y}$  are of the same camera, we have  $y_i = \sqrt{\lambda} x_i$ . Therefore, Equation (21) can be rewritten as

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

It is known that  $x_i \sim \mathcal{N}(0, \sigma^2)$ , therefore  $x_i^2 / \sigma^2$  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  $x_i$ ,  $\alpha_i$ , and  $\beta_i$

are mutually independent, we can easily derive the mean and variance of the  $i$ th element as

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

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

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

where

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

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

Suppose  $\mathbf{X}$  and  $\mathbf{Y}$  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  $\mathbf{X}$  and  $\mathbf{Y}$ , then we can use the conclusion of Scenario 1 to determine the distribution of the correlation between  $\mathbf{X}$  and  $\mathbf{Y}$ . For the centroids from two different classes, we still have the same conclusion  $\rho(\mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(0, 1/d)$ . 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,  $\sigma^2$ . If  $n$  such fingerprints are averaged before standardization, the quality of the true SPN,  $\sigma^2$ , 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

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

Replacing  $n$  in Equation (26) with  $n_x$  and  $n_y$ ,  $\sigma^2$  with  $\sigma_x^2$  and  $\sigma_y^2$  yields the distributions for the  $i$ th element of the two centroids:

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

and

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

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

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

where

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

By setting  $n_x = n_y = 1$ , Equation (30) becomes Equation (25). But in practice,  $\lambda$  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).

## 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.*, accepted for publication in 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. 2010 IEEE Int. Workshop Inf. Forensics Security*, Dec 2010, pp. 1–5.
- [10] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proc. the 9th 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] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proc. the 12th ACM SIGKDD Int. Conf. on Knowledge Discovery Data Mining*, 2006, pp. 287–296.
- [14] 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.

- [15] S. M. Van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, Netherlands, May, 2000.
- [16] 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.
- [17] 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.
- [18] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient Similarity Search In Sequence Databases," in *Proc. the 4th Int. Conf. Foundations Data Organization Algorithms*, 1993, pp. 69–84.
- [19] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith, "Efficient retrieval for browsing large image databases," in *Proc. the 5th Int. Conf. Inf. Knowledge Management*, 1996, pp. 11–18.
- [20] K.-P. Chan and A.-C. Fu, "Efficient time series matching by wavelets," in *Proc. the 15th Int. Conf. Data Engineering*, 1999, pp. 126–133.
- [21] 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.
- [22] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
- [23] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Sci.*, vol. 290, no. 5500, pp. 2319–2323, 2000.
- [24] 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.
- [25] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Sci.*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [26] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. Computer System Sci.*, vol. 66, no. 4, pp. 671–687, 2003.
- [27] I. S. Dhillon, Y. Guan, and B. Kulis, "A Unified View of Kernel k-means, Spectral Clustering and Graph Cuts," Department of Computer Sciences, University of Texas, Austin, Texas, Tech. Rep. TR-04-25, February 2004.
- [28] 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.
- [29] G. J. Bloy, "Blind camera fingerprinting and image clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 532–534, 2007.
- [30] J. Eklann, "Source Camera Classification and Clustering from Sensor Pattern Noise," Master's thesis, Chalmers University OF Technology, Sweden, 2012.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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>