Fast Image Clustering based on Compressed Camera Fingerprints

Sahib Khan, Tiziano Bianchi[1]

DET, Politecnico di Torino, Corso Duca degli Abruzzi, 24 10129 TO, Italy

Abstract

Every camera sensor leaves unique traces on the acquired images that can be thought of as a camera fingerprint. This work presents an efficient algorithm for clustering images based on their camera fingerprints. The algorithm performs a fast preliminary clustering based on a compressed representation of the camera fingerprints, then it refines the initial clusters using full-size fingerprints. The efficiency of the method is further improved by scanning the images according to a ranking index that depends on fingerprint estimation quality. The results confirm that the proposed method achieves a performance comparable to the state of the art approaches, with a significantly lower computational complexity, especially on large datasets. The method can also handle cases in which the number of clusters is much larger than the average size of the clusters.

Keywords: Image clustering, photo response non-uniformity, computational complexity, source camera identification

1. Introduction

Capturing life events and keeping them safe in the form of digital images is a common practice nowadays. The trend is increasing rapidly day by day due to the availability of high-resolution cameras in mobile phones at reasonably low prices. The contents and quality of the images are of main interest to the general viewer. However, for forensic experts, the knowledge about the source of an image is essential, because it can help in finding non-obvious clues and solving criminal cases. This information is available in the metadata, e.g., the Exif header, of the image captured with some camera models, but not available in all cameras models. Along with this, metadata are modifiable and can be easily changed or even removed. Forensic applications need stable, irremovable, unique camera features, which can be reliably considered and used as source information about the cameras. In the literature, it has been found that due to

*Corresponding authors

Email address: `sahib.khan@polito.it, tiziano.bianchi@polito.it` (Sahib Khan, Tiziano Bianchi)

manufacturing imperfections, the sensors of the image acquisition devices leave some traces, called sensor pattern noise (SPN), which can be used for source identification of images. The SPN is mostly contributed by photo response non-uniformity (PRNU)[1, 2]. Due to the uniqueness and stable nature of PRNU, it can be used as a camera fingerprint [3, 4] to find the source of an image, link an image with the source device, and group images from the same camera.

In this paper, we will consider the problem of analyzing a set of images and grouping them according to the source device. The problem is illustrated in Fig. 1. A set of images is provided to the forensic analyst, possibly without any prior information regarding the different cameras that acquired them. The aim of the forensic analyst is to divide the images into different clusters, where each cluster should contain images acquired by the same camera. This process can be useful in several practical settings. For example, one can analyze a gallery of images in a suspect's social network profile and infer all the different cameras used by the suspect. This can be later used to link images from different crime scenes with the suspect's cameras or to match different user profiles across social platforms [5, 6].

The camera fingerprints are usually estimated by acquiring a certain number of images from the cameras under test. A proper and reliable estimation of a camera fingerprint can be made from a sufficient number of flat, unsaturated, and uniformly bright images [2]. However, when the source camera is not available, it is impossible to have a suitable number of flat, unsaturated, and uniformly bright images from the same camera. Moreover, in the absence of any side information on the analyzed images, we do not know a priori which images have to be grouped to estimate the camera fingerprint. Therefore, in the most challenging clustering scenario, the camera fingerprint is simply determined from the noise residual of a single image, by subtracting the de-noised image from the original image [7, 8].

The clustering of camera fingerprints presents some intrinsic problems like high computational cost, I/O cost, large memory requirements, sensitivity to outliers, and the absence of prior information, which make most of the classical clustering algorithms [9, 10] unsuitable for this problem. Almost all currently available camera fingerprint-based clustering algorithms rely on the normalized correlation and use it as a similarity measure for clustering. The pairs of fingerprints are considered from the same source if the normalized correlation between them is above a specific threshold. Along with this, most of the existing image clustering algorithms [11, 12, 13, 14] can be computationally expensive, since they compute the full cross-correlation matrix among n fingerprints requiring $(n(n-1))/2$ correlations. Another problem occurs when the number of cameras (NC) is much larger than the average number of images captured by a single camera (SC). Several existing algorithms [11, 14, 15, 10] have low performance in this case.

The main objective of the proposed algorithm is to cluster images based on camera fingerprints obtaining an accuracy of the output clusters comparable to that of state of the art algorithms but reducing the computational complexity. There are many ideas in the proposed approach: 1) a simple clustering

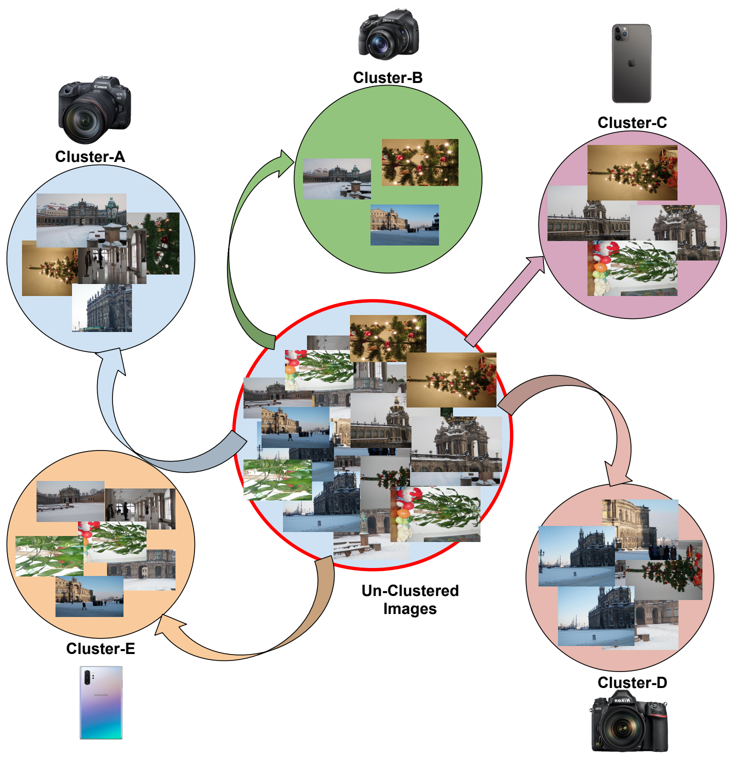


Figure 1: Depiction of the clustering problem considered in this paper. A set of images should be divided in smaller subsets, where each subset corresponds to images acquired by the same camera. In the most challenging scenario, we assume that there is no prior information regarding the cameras that acquired the set of images.

strategy uses a suitably selected reference fingerprint among the unclustered fingerprints to construct a cluster and avoid computing $n(n-1)/2$ correlations; 2) the fingerprints are sequentially processed using a ranking index depending on their quality, to increase performance; 3) a preliminary clustering stage uses compressed fingerprints, to further reduce complexity.

Some of the above ideas were first introduced in two our previous papers. A simple reduced complexity image clustering algorithm using camera fingerprints was devised in [16], where a fingerprint is randomly selected as reference fingerprint to attract other fingerprints close to it. The clustering process can be made faster using a proper fingerprint ordering [17]. The camera fingerprints are arranged in descending order of quality, and clusters are constructed by selecting the best fingerprint among the unclustered fingerprints as a reference fingerprint. With respect to the above papers, in this work we add several important contributions. The clustering process is split in two stages. In an initial stage, a compressed version of the fingerprint is used to obtain a preliminary clustering with very low complexity. Then, the clusters are refined using merged full fingerprints of the constructed clusters. Different compression strategies have been analyzed, in order to find the configuration offering the best performance/complexity trade-off. Moreover, the proposed algorithm has been experimentally evaluated both on small and large scale datasets, providing insights on its behavior when there is large variability in the number of cameras and the number of images per camera. The results reveal that the compressed fingerprints based clustering has lower computational complexity than state-of-the-art clustering algorithms [18, 19] and our previously proposed algorithms [16, 17], while it constructs clusters of comparable quality. Along with lower complexity, the proposed algorithm does not suffer from the $NC \gg SC$ problem.

The rest of the paper is organized as follows. Section 2 presents a literature review of previous works on the subject. Section 3 presents the detailed implementation of the proposed algorithm. The experimental setup, results, and analysis of the proposed algorithm is described in Section 4. The comparison of the proposed algorithm with state-of-the-art methods is made in Section 4.8. Concluding remarks are given in Section 5.

2. Related Work

This section presents a brief literature review of camera fingerprints-based image clustering techniques. One of the first attempts was made in the work of Bloy [18], where the author used the modified pairwise nearest neighbor (PNN) algorithm [20]. PNN algorithm operates recursively and initially considers all the elements of the dataset as individual clusters and calculates pairwise distances between them. The closest clusters are merged to make a new cluster. The process continues until a stopping condition is met. Bloy modified the PNN algorithm by randomly picking a couple of clusters, searching all neighbors, merging the closest ones, and then proceeding with another pair of clusters. This technique needs a suitable threshold for merging a pair of clusters.

With the addition of fingerprint enhancement, other variants of this technique are presented in [8, 11]. In [11], Li used enhanced fingerprints as random variables, and a Markov random field (MRF) is used to cluster these fingerprints iteratively. A subset of images is randomly chosen, and a pairwise similarity matrix is generated. The reference similarity and membership committee are determined, based on the matrix. The likelihood probability of belonging to a class is calculated for the corresponding fingerprint, which is assigned to a class on the basis of the highest likelihood probability in the membership committee. This process ends when no change is observed in the class label after two consecutive iterations. In the end, fingerprints not belonging to the training set are assigned to their closest clusters identified in the training set. This algorithm is very efficient when used for clustering small databases. However, it is computationally expensive and not suitable for situations when the number of cameras NC is much larger than the average size of cluster SC.

Liu et al. treated camera fingerprint clustering as a graph partitioning problem in [12] and considered it as a weighted unidirectional graph, with fingerprints as vertexes and similarity between two fingerprints as the weight of edge that link the fingerprints. The K-nearest neighbor graph is constructed, where K is a parameter controlling the sparsity of the graph. The edge weights of a randomly selected vertex with all the other vertices are calculated. The $(K+1)^{th}$ closest vertex to the initial center is chosen as the new center, and excluding the first center, its edge weights with all the other vertices are calculated. The process ends when the number of vertices not used as a center is no larger than K. Afterwards, a multi-class spectral clustering algorithm [13] is used to partition the vertices of the constructed K-nearest graph. The spectral clustering is repeated until the size of the smallest cluster equals 1. This algorithm is more efficient than Li's algorithm [11], but it has high I/O cost and needs to know the number of partitions in advance.

In [14], a faster solution based on hierarchical clustering is proposed, together with a criterion based on a silhouette coefficient [21]. The fingerprints are enhanced, and instead of a complete dataset, a randomly selected training set is used for clustering. All selected fingerprints are treated as an individual cluster, and a pairwise similarity matrix is generated. A couple of clusters with maximum similarity are merged, and the matrix is updated. After updating, the silhouette coefficient 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. In [22], Gisolf reduced PRNU-related computation, and the compressed fingerprints were used instead of the original fingerprints for iterative clustering. In each iteration, a couple of clusters with the highest correlation are selected as candidates. Finally, in [21], a refinement step based on Hu's moment vector is applied to improve performance. The framework with some modifications is used in [15] for smart-phone clustering. The silhouette coefficient is calculated for each cluster instead of each individual fingerprint. It is claimed in [14], that the silhouette coefficient based algorithm is faster than [11] and provides comparable accuracy. However, its computational cost is still very high and makes it unsuitable for large databases.

The algorithms discussed earlier are either computationally too expensive, or the computational cost is reduced by clustering a randomly selected training set, and using the obtained clusters for attracting the remaining fingerprints. However, the training set should adequately represent the complete dataset for successful classification. The dataset and training set must include a suitable number of representatives of all clusters. If the dataset does not have a sufficient number of fingerprints for all clusters, i.e., we are in the $NC \gg SC$ scenario, it is almost impossible to select a suitable training set which represents well the complete dataset. As a result of the bad selection of training sets, the remaining unclassified fingerprints are miss-classified.

A solution to the clustering of a large database under the $NC \gg SC$ problem was proposed by Lin and Li in [19]. The proposed algorithm splits a large dataset in several small datasets that can be quickly loaded on RAM. Then coarse clusters are obtained using a simple clustering algorithm. The coarse clusters are then refined using fine clustering, followed by attraction and post-processing, to enhance the clustering results. Lin and Li's technique generates high quality clusters in a much faster way, especially for large databases.

In a recent approach, Li and Lin, in [23], presented a fast source-oriented image clustering technique based on the pairwise correlation among camera fingerprints. By considering each fingerprint as a random variable, Markov random field (MRF) technique is used to assign a class label to each fingerprint. Li and Lin formulate a cost function and assign different voting power to neighbors to determine class labels, depending on their similarity. This algorithm does not require any prior information about the dataset. The algorithm reduced the computational complexity of each iteration and accelerated the speed of convergence. However, this algorithm did not addressed the $NC \gg SC$ problem. In [5, 24], Phan et al., presented a sparse subspace clustering (SSC) based technique [25]. The SSC technique is based on the concept that a data point can be written by the linear combination of other points in the same subspace. This technique uses the sparse representation of each camera fingerprint, using $l1$-regularized least squares and estimating appropriate parameters in a data-driven manner.

The classical clustering algorithms are not used for clustering digital images using camera fingerprints, because these algorithms have some serious limitations in this scenario. The K-means [26] and CLARANS [9] clustering algorithms need prior information about the number of clusters and requires high computational resources to process large datasets. DBSCAN [27] has large memory requirements and has large I/O cost [10]. The hierarchical clustering algorithms like [10] and [28] reduce the input size for large databases but suffer from the $NC \gg SC$ problem. Some hierarchical clustering algorithms, such as BIRCH [29] and CHAMELEON [30], are designed for large-scale databases, but these are very sensitive to outliers and have high I/O cost due to the generation of a K-nearest neighbor graph.

The clustering of images based on camera fingerprints is a computationally expensive process. It also has high I/O cost and large memory requirements. Although the existing clustering algorithms result in high quality clusters, they suffer from all or some of the above problems. Many of the above clustering algorithms [5, 11, 12, 13, 24] compute the full cross-correlation matrix among the n fingerprints, which requires $(n(n-1))/2$ correlations and can be computationally intensive, especially in the case of large datasets. Along with these problems, some algorithms, such as [14, 19, 9, 29], suffer when the number of cameras (NC) is much larger than the average number of images captured by a single camera (SC).

Very recently, machine learning techniques using convolutional neural networks (CNN) have been applied to camera identification. Whereas classical supervised techniques seems to provide good results only for camera model identification [31], a recent approach based on a Siamese CNN named noiseprint [32] has been successfully extended to camera sensor identification [33, 34]. Using the above techniques in a large scale clustering scenario has not been investigated yet, however it can provide an interesting avenue for future research.

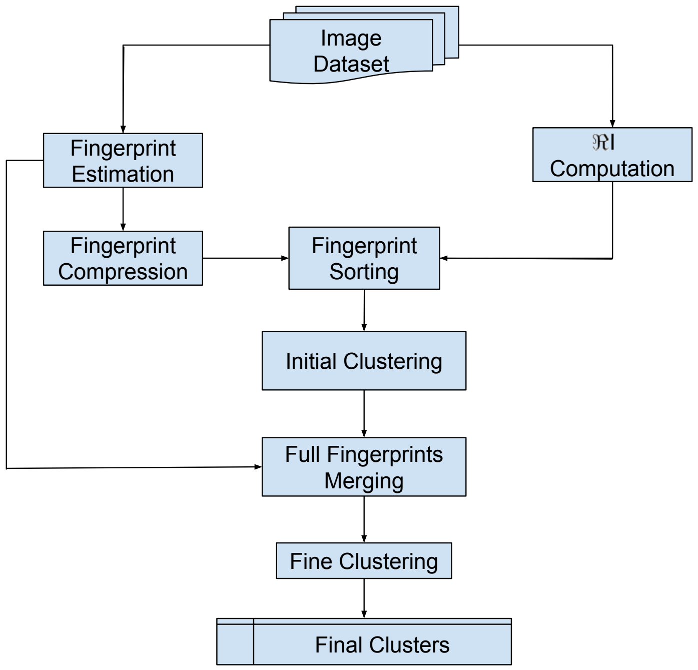


Figure 2: Block diagram of the proposed algorithm.

3. Proposed Algorithm

In this section, we provide a detailed description of the different steps composing the proposed algorithm, hereafter named as Fast Image Clustering based on Compressed Camera Fingerprints (FIC3F). The block diagram of the proposed algorithm is shown in Fig. 2. We consider a scenario in which the dataset of images to be analyzed have all the same size, so that there is no side information apart from the camera fingerprint. As a first step, we compute a ranking index $\Re I$ depending on the image content, which tells how reliably a fingerprint can be estimated from that image. Then the camera fingerprint is estimated from each of the images by subtracting the de-noised image from the original image. This estimated camera fingerprint is denoted as full fingerprint F. The full fingerprints are further compressed to get reduced fingerprints Fr. The reduced fingerprints are sorted in decreasing order of the respective $\Re I$. The FIC3F algorithm uses the reduced and sorted camera fingerprints Fr to compute initial clusters. Then, for each of the initial clusters, a reference fingerprint for the cluster is obtained by averaging the full fingerprints belong to that cluster. The new reference fingerprints are then used in a fine clustering step to refine the clusters.

The different steps of the FIC3F algorithm are described in the following sections.

3.1. Fingerprint Estimation

As a preliminary step, the camera fingerprint is estimated for each image in the dataset. The estimation of the camera fingerprint is a standard procedure when a sufficiently large number of image samples are available [1]. However, in the case of image clustering, it is not possible to use multiple images, since there is no side information to infer the source camera. Therefore, each image is treated individually.

Camera fingerprint estimation is based on a simplified linear model of camera sensor output [2], each image is de-noised using Mihcak filter operation [1], and subtracted from the original image to get the camera fingerprint [2, 35]. A set of, initially un-clustered, camera fingerprints M are obtained from a dataset of images I and standardized to zero mean and unit variance using Eq. 1.

$$M = \{F_i | F_i = \Phi(X_i - D(X_i)) \land 1 \leqslant i \leqslant n, X_i \in I\} \tag{1}$$

Where $D(.)$ is the de-noising function, $\Phi(.)$ denotes common post processing operations including Wiener filtering and mean removal to suppress non unique artifacts (NUAs) as explained in [36], as well as normalization to zero mean and unit variance, n is the number of images in dataset, X_i is the i^{th} image in dataset and F_i is the camera fingerprint obtained from X_i.

3.2. Fingerprint Compression

The compression of camera fingerprints is one of the key processes in the implementation of the FIC3F algorithm. The compression process reduces the size of the camera fingerprint, which in turn helps to reduce the computational cost and memory requirement of the clustering process. Several techniques have been proposed to compress camera fingerprints. These include trimming and cropping [37], fingerprint digest [37], Gaussian random projections [38] and binarization [39]. Trimming unwraps camera fingerprint column-wise and trim the fingerprint by preserving only the P_r first samples. Cropping preserves only the center portion of the camera fingerprint. The fingerprint digest technique builds a digest by keeping the P_r highest energy components and their positions. This technique relays on the assumption that the most prominent peaks of the extracted camera fingerprints can be used as a suitable camera attribute. The Gaussian random projections based compression technique was introduced by Valsesia et al. The basic idea is to project the one-dimensional unwrapped camera fingerprint from a vector space of a large dimension to a subspace of reduced dimension P_r. The binarization technique obtains binarized camera fingerprint by performing element-wise quantization. This can be used even after the earlier mentioned camera fingerprints compression techniques.

In this paper, we explore two different methods for fingerprint compression. The first method is based on decimation, random projection, and dead-zone quantization, as presented in [40]. This compression technique is presented in Algorithm 1. The second method directly applies dead-zone quantization to the decimated camera fingerprints and is presented in Algorithm 2. The aim of introducing both algorithms is to investigate whether random projections can provide any advantage in this setting. The reduced fingerprints obtained by the above methods are approximate representations of their respective full camera fingerprints. However, they preserve the correlation among fingerprints and can be used to obtain a reliable initial clustering.

The steps involved in the compression of the fingerprints are presented in the following subsections.

3.2.1. Decimation

In [40], Bondi et al. considered the estimated fingerprints extracted from flat-field and natural images to analyze the effect of JPEG compression on the power spectral density (PSD) for different quality factors. The analysis reveals that increasing compression level lower the power of the residue at high spatial frequencies. At the same time, camera fingerprint contributions in high-frequency bins are combined with residuals of blockiness artifacts due to JPEG compression that cannot be entirely removed by the residue extraction process [41, 40]. Taking this into account, the authors in [40] propose to attenuate the high-frequency components by decimating F by a factor $d > 1$ along rows and columns. The operation is accomplished via interpolation with a cubic kernel [42] $H_c(z)$ defined as

$$h_c(z) = \begin{cases} 1.5|z|^3 - 2.5|z|^2 + 1 & \text{if } |z| \geq 1 \\ -0.5|z|^3 + 2.5|z|^2 - 4|z| + 2 & \text{if } 1 < |z| \leq 2 \\ 0 & \text{otherwise} \end{cases} . \tag{2}$$

Given a vector y of length L_y, if the vector y is decimated by a factor d then the i^{th} element of the decimated vector y_d is

$$y_d(i) = \sum_{j=0}^{L_y-1} h_c(j - id)y(j), \forall i \in \{0, ..., \lfloor L_y/d \rfloor\}. \tag{3}$$

After applying a decimation process on a camera fingerprint F, we get a decimated camera fingerprint F_d of reduced size $|F|/d^2$.

3.2.2. Random Projection

After decimation, the next step is to generate random projections for the given decimated camera fingerprint F_d. The random projections are obtained using a Gaussian sensing matrix [38], which has proven to be an effective way of compressing camera fingerprints. The sensing matrix Ψ of dimension $P_r \times |F_d|$ is generated with samples being extracted from a i.i.d zero-mean Gaussian distribution. The resulting projection RP of the decimated camera fingerprint F_d is obtained by taking a simple matrix product between the sensing matrix Ψ and the decimated camera fingerprint F_d

$$RP = \Psi F_d \tag{4}$$

The random projection process reduces the size of camera fingerprint from $|F_d|$ to P_r, where $P_r \leq |F_d|$.

3.2.3. Dead-Zone Quantization

Binarization of random projections is an effective technique to preserve good performance in terms of detection [40]. Here, we binarize the fingerprints by using dead-zone quantization. Given σ, the standard deviation of RP, the i^{th} element of Fr for $i = 1, \ldots, P_r$ is obtained as

$$Fr(i) = \begin{cases} +1 & \text{if } RP(i) > \delta\sigma \\ 0 & \text{if } -\delta\sigma \leq RP(i) \leq \delta\sigma \\ -1 & \text{if } RP(i) < -\delta\sigma \end{cases} . \tag{5}$$

where δ is a parameter controlling the sparseness of the quantized fingerprints. Dead-zone quantization has two advantages; first, it preserves the peaks, which are very important in terms of cross-correlation. Secondly, the variable threshold of the quantization process allows reducing the bit-rate of Fr via entropy coding by increasing δ while keeping P_r fixed. However, in this paper we prefer a simple two-bit encoding of ternary values versus optimized entropy coding because this simplifies computing the correlation of binarized fingerprints. Hence, the size of the Fr in terms of bits will be $2 \times P_r$ bits.

Algorithm 1 Fingerprints compression using decimation, random projections and dead zone quantization

```
Input: $F$ : Camera fingerprint, $\sigma$ : Standard deviation of camera fingerprint,
    $d$ : Decimation factor, $P_r$ : Number of random projections, $\Psi$ : Sensing
    matrix, $\delta$ : Dead-Zone threshold
1: $F_d = Decimate(F, d)$
2: $RP = \Psi F_d$
3: $Fr(i) = \begin{cases} +1 & \text{if } RP(i) > \delta\sigma \\ 0 & \text{if } -\delta\sigma \leq RP(i) \leq \delta\sigma \\ -1 & \text{if } RP(i) < -\delta\sigma \end{cases} .$
```

Algorithm 2 Fingerprints compression using decimation and dead zone quantization

```
Input: $F$ : Camera fingerprint, $\sigma$ : Standard deviation of camera fingerprint,
    $d$ : Decimation factor, $P_r$ : Number of random projections, $\delta$ : Dead-Zone
    threshold
1: $F_d = Decimate(F, d)$
2: $Fr(i) = \begin{cases} +1 & \text{if } F_d(i) > \delta\sigma \\ 0 & \text{if } -\delta\sigma \leq F_d(i) \leq \delta\sigma \\ -1 & \text{if } F_d(i) < -\delta\sigma \end{cases} .$
```

3.3. Fingerprint Sorting

Clustering images according to a common source would be much easier if the centroids of clusters were known in advance, since the centroids could be used to attract images from the same camera with linear complexity in the number of fingerprints. In practice we do not have any information regarding the centroids, and we can only blindly select one of the fingerprints as reference. However, we can assume that fingerprints with lower estimation error would be closer to the true reference fingerprint of the respective camera. We know that camera fingerprints estimated from a uniformly bright, flat and unsaturated image have the least estimation error, so they can be considered as the most suitable candidate for being a centroid, among the available camera fingerprints. The FIC3F algorithm is based on the same assumption. The algorithm computes a ranking index $\Re I$, using the inherent gray-level, saturation, and texture information of the images. The $\Re I$ index indicates the quality of estimated camera fingerprints: the higher the $\Re I$ of an image the better the quality of the camera fingerprint estimated from it.

To compute $\Re I_i$ for an image $X_i \in I$ the average and normalized gray-level G_i and saturation S_i are calculated as in Eq. 6 and Eq. 7, respectively.

$$G_i = \frac{\sum_{j=1}^{|X_i|} X_i(j)}{255 \times |X_i|} \tag{6}$$

$$S_i = \frac{\sum_{j=1}^{|X_i|}(X_i(j) == 255)}{|X_i|} \tag{7}$$

The texture information of an image X_i is obtained by calculating the normalized edges energy. The edges L_i of an image X_i are computed using a Laplacian filter, applied to the image X_i, which highlights the regions of rapid gray-level change. The edges L_i of an image X_i are given by Eq. 8.

$$L_i = imfilter(X_i, A) \tag{8}$$

Where, $imfilter(.)$ denotes 2D filtering, and A is a kernel that approximates the second-order derivative, given by

$$A = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{9}$$

The texture level T_i is obtained by computing the energy of edges normalized with respect to the total energy of the image X_i, as given in Eq. 10.

$$T_i = \frac{\sum_{j=1}^{|X_i|} L_i(j)^2}{\sum_{j=1}^{|X_i|} X_i(j)^2} \tag{10}$$

Finally, the $\Re I_i$ for image X_i is obtained by combining the values of G_i, S_i, and T_i according to the following equation.

$$\Re I_i = G_i^{\frac{1}{\alpha}} \times (1 - S_i)^{\frac{1}{\beta}} \times (1 - T_i)^{\frac{1}{\gamma}} \tag{11}$$

where, α, β and γ are factors defining the contribution of G_i, S_i and T_i in $\Re I_i$, respectively.

Eq. 11 shows that unsaturated and flat images with average grayscale values will results in high values of $\Re I$, whereas saturated, highly textured, or dark images will result in lower values of $\Re I$. So, we assume that images with high values of $\Re I$ will yield fingerprints characterized by a lower estimation error.

After computing the $\Re I$ for each image, the camera fingerprints F and quantized camera fingerprints Fr are arranged in decreasing order of $\Re I$, to get a set of sorted fingerprints M_O. These fingerprints are then used for clustering.

3.4. Initial Clustering

The processes of camera fingerprints estimation, computation of compressed camera fingerprints, $\Re I$ computation, and sorting of camera fingerprints based on $\Re I$, are followed by clustering. The clustering is performed in two steps. The first step is denoted as initial clustering, the second step is called fine clustering. The initial clustering is performed using reduced camera fingerprints Fr. The clusters are refined in the fine clustering step using full camera fingerprints F. The clustering process works iteratively, performing different rounds, where each round results in a new cluster. Each clustering round is denoted by the cluster index K. In each round K, a cluster $C_K = \emptyset$ is initialized by the algorithm. The un-clustered fingerprints are assigned to a set UC_K.

At the start of clustering, we set $K = 1$ and all sorted camera fingerprints are assigned to the set of un-clustered fingerprints UC_K i.e., $UC_1 = M_O$. To build the K^{th} cluster i.e., C^r_K, the FIC3F algorithm always selects as reference fingerprint RF^r_K the best-reduced fingerprint F_r from the set of sorted and un-clustered fingerprints UC_K and the corresponding full fingerprint F is assigned to cluster C^r_K. If the ranking index is consistent, RF^r_K will be the best-estimated fingerprint among all the un-clustered fingerprints UC^r_K and the best representative of the respective cluster C^r_K. The normalized cross-correlation (NCC) ρ between the reference fingerprint RF^r_K and all other fingerprints Fr_i is used to decide whether the given Fr_i belongs to the same camera as that of RF^r_K, or not.

The normalized cross correlation ρ between Fr_i and RF^r_K can be computed as in Eq. 12.

$$\rho(i) = \sum_{x=1}^{P_r} RF^r_K[x]Fr_i[x] \tag{12}$$

Where P_r is the dimension of the reduced fingerprint Fr_i.

If the NCC ρ between the reduced fingerprint F^r_i and reference fingerprint RF^r_K has a value greater than or equal to a threshold value Th, F_i is assigned to the cluster C^r_K, otherwise the reduced fingerprint F^r_i and corresponding full fingerprint F_i are assigned to the set of un-clustered fingerprints UC_{K+1}. In order to guarantee a certain probability of false alarm PFA, i.e., the probability of assigning a wrong fingerprint to C^r_K, under the assumption that the NCC ρ for two unrelated fingerprints can be modeled by a Gaussian distribution with zero mean and standard deviation σ_ρ, the threshold value can be computed as follows

$$Th = \sigma_\rho \times \sqrt{2}\ erfc^{-1}(2 \times PFA) \tag{13}$$

where $erfc^{-1}(.)$ is the inverse of the complementary error function. The variance of ρ can be computed as follows. The reduced camera fingerprints are quantized in the dead zone quatization process, and each entry of these quantized fingerprints has only three possible values among $+1$, 0 and -1. Let us denote the probability of having an entry equal to zero as P_o, so that the probability of $+1$ or -1 is $(1-P_o)/2$. Hence, the variance of ρ when the reduced fingerprints come from different cameras can be computed as $\sigma_\rho^2 = P_r(1-P_o)$, leading to the following expression for the threshold:

$$Th = \sqrt{2 \times P_r(1-P_o)}\ erfc^{-1}(2 \times PFA) \tag{14}$$

While constructing the cluster C_K^r, a total of $|UC_K|-1$ correlation operations are performed, and a total of $|UC_{K+1}| = |UC_K| - |C_K^r|$ fingerprints are left un-clustered. To cluster the remaining fingerprints, if any, the cluster index K is incremented by 1 and the un-clustered UC_{K+1} fingerprints are processed to construct a new cluster C_{K+1}^r by repeating the same procedure. The process continues till all fingerprints are assigned to a cluster, and UC_{K+1} gets empty. At the end of each round, the full fingerprints F in each cluster C_K^r are merged by averaging them, to compute a full reference fingerprint RF_K^f for each cluster. The full reference fingerprints RF_K^f is then used in the fine clustering stage to attract other clusters.

The complete process is detailed in Algorithm 3.

3.5. Fine Clustering

The fine clustering process uses the full reference fingerprints RF_K^f for the possible attraction of clusters C_K^r. Initially, we have a set of full reference fingerprints M_{RF^f}. All the reference fingerprints are treated as un-clustered fingerprints. The fine clustering index H is initiated and set to one i.e., $H = 1$, and all reference fingerprints are assigned to the set of un-clustered fingerprints, i.e., $UC_1 = M_{RF^f}$. To construct the H^{th} fine cluster C_H^F, the FIC3F algorithm always selects as reference fingerprint RF_H^f the first full reference fingerprint RF_1^f from the un-clustered full fingerprints UC_H and all the fingerprints in the corresponding cluster C_H^r are assigned to the fine cluster C_H^F, i.e. $C_H^F \leftarrow C_H^r$. Due to the way average full fingerprints are constructed in the initial clustering stage, full fingerprints inherit the order according to the ranking index, so the first full reference fingerprint can be assumed the best candidate for attracting the other clusters. The NCC ρ^f between the full reference fingerprint RF_H^f and all other full reference fingerprints RF_i^f is calculated as follows

$$\rho^f(i) = \sum_{x=1}^{\bar{d}} RF_H^f[x] RF_i^F[x] \tag{15}$$

Algorithm 3 Initial clustering

```
Input: $M_O$, $K$, $PFA$, $P_r$
Output: $C^r_K$,$RF^f_K$
    $Initialization : K = 1 , UC_K = |M_O|$
 1: $Th = \sqrt{2 \times P_r(1 - P_o)}\ erfc^{-1}(2 \times PFA)$
 2: while $(|UC_K| \neq 0)$
 3: $UC_{K+1} = \emptyset$
 4: $C^r_K = \emptyset$
 5: $RF^r_K = Fr_1$
 6: $C^r_K \leftarrow F_1$
 7: for $j = 2\ to\ |UC_K|$ do
 8:     $\rho(j) = \sum_{x=1}^{P_r} RF^r_K[x] Fr_j[x]$
 9:     if $(\rho(j) \geq Th)$ then
10:         $C^r_K \leftarrow F_j$
11:     else
12:         $UC_{K+1} \leftarrow Fr_j$
13:         $UC_{K+1} \leftarrow F_j$
14:     end if
15: end for
16: $RF^f_K = \frac{\sum_{i=1}^{|C^r_K|} F_i}{|C^r_K|}$ where $F_i \in C^r_K$
17: $K = K + 1$
18: endwhile
```

If the NCC ρ^f between the two reference fingerprints has a value greater than a threshold value T, all the fingerprints in cluster C_i^r, are assigned to the cluster C_H^f and the reference fingerprints are merged, otherwise the reference fingerprint RF_i^f is assigned to the set of un-clustered fingerprints UC_{H+1} and the corresponding C_i^r is left unaffected. Since reference fingerprints are normalized vectors of dimension $\bar{d}$, the variance of ρ^f for unrelated fingerprints can be obtained as $\sigma^2_{\rho^f} = \bar{d}$ and, according to (13), the threshold value T is computed as

$$T = \sqrt{2 \times \bar{d}}\; erfc^{-1}(2 \times PFA) \tag{16}$$

At the end of round H a fine cluster C_H^f is constructed. While constructing the cluster C_H^f, a total of $|UC_H| - 1$ correlation operations are performed, and a total of $|UC_{H+1}| = |UC_H| - |C_H^f|$ fingerprints are left un-clustered.

The cluster index H is incremented by 1 and the full reference fingerprints in UC_{H+1}, are processed to construct a new fine cluster C_{H+1}^f by repeating the same procedure. The process continues till all fingerprints of the C_H^r are assigned to a fine cluster C_H^f, and UC_{H+1} gets empty.

The complete procedure is detailed in Algorithm 4

Algorithm 4 Fine clustering

```
Input: $M_{RF^f}$, $H$, $RF_H^f$, $PFA$
Output: $C_H^f$
    $Initialization : H = 1 , UC_K = |M_{RF^f}|$
 1: $T = \sqrt{2 \times \bar{d}}\; erfc^{-1}(2 \times PFA)$
 2: while $(|UC_H| \neq 0)$
 3: $UC_{H+1} = \emptyset$
 4: $C_H^f = \emptyset$
 5: $RF_H^f = RF_1^f$
 6: $C_H^f \leftarrow C_H^r$
 7: for $j = 2$ to $|UC_H|$ do
 8:     $\rho^f(j) = \sum_{x=1}^{\bar{d}} RF_H^f[x] RF_j^f[x]$
 9:     if $(\rho^f(j) \geq T)$ then
10:         $C_H^f \leftarrow C_j^r$
11:         $RF_H^f = \frac{(RF_H^f + RF_j^f)}{2}$
12:     else
13:         $UC_{H+1} \leftarrow RF_j^f$
14:         $C_j^r$ un-affected
15:     end if
16: end for
17: $H = H + 1$
18: endwhile
```

The fine clustering uses the average of multiple full fingerprints F as reference fingerprints which are a more stable and reliable estimation of the true fingerprint [1]. Therefore, in the fine clustering stage many clusters composed by a single fingerprint are usually attracted by the correct cluster.

3.6. Computational Complexity, I/O Cost and RAM Requirements

In this section, we discuss the total computational complexity, I/O cost and RAM requirements and the reasons for the suitability of the FIC3F algorithm for large scale clustering.

Computational Complexity

The FIC3F algorithm is composed of initial clustering and fine clustering. Both of these stages contributes to the computational cost. The computational cost T_c^r of initial clustering is given by Eq. 17.

$$T_c^r = \zeta \times \left(\sum_{i=1}^{NCr} |UC_i| - NCr \right) \tag{17}$$

where NCr is the total number of clusters constructed in initial clustering and ζ is the ratio between the sizes (in bits) of the reduced and full fingerprints.

The computational cost of the fine clustering stage is measured only in terms of the total number of correlations performed. The computational cost T_c^f of the fine clustering stage is given by Eq. 18.

$$T_c^f = \sum_{i=1}^{NCf} |UC_i| - NCf \tag{18}$$

where, NCf is the total number of clusters obtained at the end of fine clustering.

The total computational cost T_c^t of the FIC3F algorithm is the sum of the computational cost of fine clustering T_c^f, scaled computational cost of initial clustering T_c^f and the cost of merging fingerprints $Cost_{merging}$, given by Eq. 19.

$$T_c^t = T_c^f + T_c^r + Cost_{merging} \tag{19}$$

The merging of fingerprints requires only additions so its cost can be assumed as negligible compared to the combined cost of initial clustering and fine clustering. Overall, the total cost of the FIC3F algorithm is dominated by the number of correlations, which is significantly less than the reference value $n(n-1)/2$ when the number of images per camera SC is greater or equal to 2 i.e., $SC \geq 2$, while for $SC = 1$ the computational cost is comparable to the reference complexity. For example, if we have a dataset of n fingerprints with $SC = 1$ and the fingerprints are correctly clustered without any false positive, then the FIC3F algorithm will perform $\zeta(n(n-1)/2)$ and $n(n-1)/2$ correlations in initial and fine clustering, respectively. Hence, the total cost of clustering will be $(\zeta+1)(n(n-1)/2)$ which is $\zeta+1$ times larger than the reference complexity i.e., $n(n-1)/2$. However, if $SC \geq 2$, the computational complexity of the FIC3F algorithm, as compared to the reference complexity, decreases as the size of the dataset increases. This make the algorithm suitable for large scale clustering.

RAM Requirements

Along with the computational cost, it is essential to discuss the maximum RAM requirements of the FIC3F algorithm. During the initial clustering, the FIC3F algorithm has one reduced fingerprint Fr_j and a reduced reference fingerprint RF_K^r in RAM. The maximum RAM occupied during the initial clustering always remains constant and is equal to $2 \times P_r$. After the initial clustering, the full fingerprints of each cluster are merged together by averaging them. The average full fingerprints remain in the RAM and are used in fine clustering stage. Therefore, the RAM usage reaches its peak which gradually decreases with the possible merging of clusters during the fine clustering. The maximum RAM RAM_{max} required by the FIC3F algorithm is given by Eq. 20.

$$RAMmax^r = 64 \times NCr \times |F| + 2 \times P_r \tag{20}$$

where $64 \times |F|$ is the size of full fingerprints in terms of bits.

I/O Cost

Here it is also important to discuss the I/O cost of the FIC3F algorithm. The I/O cost of the algorithm depends on the number of clusters constructed in the initial clustering. The I/O of the FIC3F algorithm is given in Eq. 21.

$$I/O = \sum_{i=1}^{NCr} |UC_i| + n \tag{21}$$

where n is representing the I/O cost of full fingerprints. Each full fingerprint is loaded once while computing the average full fingerprints and the I/O cost due to full fingerprints is equal to n.

The FIC3F algorithm has a lower RAM requirement, however the I/O is quite large. The I/O cost can be reduced by loading all reduced fingerprints on RAM during the initial clustering. In this case, the I/O cost and the maximum RAM $RAMmax$ required are given in Eq. 22 and Eq. 23, respectively.

$$I/O = (1 + \zeta)n \tag{22}$$

$$RAMmax = 64 \times NCr \times |F| + n \times 2 \times P_r \quad bits \tag{23}$$

This can result in a significant reduction in I/O cost. However, it increases the load on RAM. Hence, a trade-off can be made between the I/O cost and maximum RAM requirements.

4. Experiments

In this section, we provide an experimental evaluation of the FIC3F clustering algorithm. First, a set of experiments is conducted to find the optimal value of parameters, like sigma σ, and the size of compressed camera fingerprints P_r, which are selected and used throughout all experiments. Then, we validate the performance of our algorithm under different settings, considering both medium and large scale datasets and different distributions of clusters. The algorithm is also analyzed under the $NC \gg SC$ scenario.

4.1. Dataset

The FIC3F clustering algorithm has been evaluated on the Dresden image database [43, 44]. The dataset is composed of 10960 images from 53 cameras of 18 different models and 10 different brands. Starting from the original Dresden dataset, different datasets are selected according to different requirements. As we know, it is very challenging to cluster images of different cameras of the same model based on camera fingerprints, because images taken by different devices of the same model undergo the same or similar in-camera processing procedures. Therefore, we classify the clustering task into easy and hard levels. The easy level includes only images taken by cameras of different models, while the hard level, considers images from devices of the same model.

Along with this, in real scenarios, the number of images captured by different cameras varies widely, which results in different contributions of cameras and different class distributions within the dataset. Therefore, we classify datasets into symmetric and asymmetric. In symmetric datasets, all cameras contribute equally to the dataset while in asymmetric datasets, the contribution is not equal.

To have a detailed analysis of the $NC \gg SC$ problem, we need a larger number of cameras. A dataset with a large number of cameras is built using the images of the existing cameras in the Dresden dataset. The images of an existing camera are divided into different non-overlapping patches of size 1023×1023 which are considered as separate images. The patches do not share any part of camera sensors; therefore, each created image will have unique PRNU. The part of the sensor array that has captured the particular patch is considered as a single camera. A dataset of images with 295 cameras and a contribution of 20 images by each camera is built in this way.

Finally, we set up the following seven datasets for the experiment:

- $D0$: Small dataset. It consists of 600 images taken by 15 cameras, each equally contributing 40 images. The 15 cameras are of different models and nearly cover 8 all of the popular camera brands.
- $D1$: Easy symmetric dataset. It consists of $1,000$ images taken by 25 cameras, each equally contributing 40 images.The 25 cameras are of different models and nearly cover 8 all of the popular camera brands, such as Cannon, Nikon,Olympus, Pentax, Samsung, and Sony.
- $D2$: Easy asymmetric dataset. The dataset is also composed of $1,000$ images taken by the same 25 cameras as in $D1$. These camera alternatively contribute 20, 30, 40, 50 and 60 images.

- $D3$: Hard symmetric dataset. It consists of $1,000$ images taken by 50 cameras, each contributing 20 images. The 50 cameras only cover 12 popular models, so some of them are of the same model.
- $D4$: Hard asymmetric dataset. The same 50 cameras as in $D3$ are part of this dataset, alternatively contributing 10, 15, 20, 25 and 30 images.
- $D5$: Dresden dataset. The dataset is composed of 10960 images from 53 cameras of 18 different models and 10 different brands.
- $D6$: Large Number of Cameras dataset. It consists of 5900 images from 295 cameras, each contributing 20 images.

The $D0$ dataset is used to investigate the choice of parameters for the FIC3F algorithm. The other four datasets, i.e., $D1$, $D2$, $D3$, and $D4$, are used for examining the performance of the algorithm on different types of small datasets. These datasets are also used for comparing the FIC3F algorithm with state-of-the-art algorithms. The $D5$ and $D6$ datasets are used for large scale clustering analysis and investigating the $NC \gg SC$ problem.

4.2. Evaluation Metric

A measure of agreement is essential for the evaluation of the FIC3F clustering algorithm and its comparison with state-of-the-art techniques. The most popular metrics used for assessment of clustering algorithms, are based on the matching of sets, e.g., Precision P, Recall R and F-measure, on information theory, e.g., mutual information MI and normalized mutual information NMI, and pair of objects counting e.g., rand index RI and adjusted rand index ARI [45, 46]. However, clustering solutions with many clusters results in higher values of NMI when compared with ground truth classes [47]. This may be misleading while comparing different clustering algorithms yielding different numbers of clusters. Hence, MI and NMI are not used in this paper.

The P, R, $F-measure$, RI and ARI are used for evaluating the proposed clustering framework and comparing it with the state-of-the-art algorithms. These metrics are computed using ground truth classes Ω and generated clusters C. Let's denote the ground truth as

$$\Omega = \{\omega_1, \omega_2, \omega_3, \ldots, \omega_{NC}\} \tag{24}$$

where each ω denotes a set of fingerprints coming from the same camera. C is the set of clusters generated by clustering algorithm and is given as

$$C = \{c_1, c_2, c_3, \ldots, c_y\} \tag{25}$$

where each c denotes a set of fingerprints assigned to a cluster. The precision P and recall R are calculated from the classes and clusters as given in the following equations.

$$P = \frac{\sum_k (max_j |c_k \cap \omega_j|)}{\sum_k |c_k|} \tag{26}$$

$$R = \frac{\sum_j (max_k|c_k \cap \omega_j|)}{\sum_j |\omega_j|} \tag{27}$$

where $|c_k|$ and $|\omega_j|$ are cardinalities of cluster c_k and ground truth class ω_j, respectively, $max_j|c_k \cap \omega_j|$ is used to find the largest number of fingerprints in cluster c_k that comes from a ground truth class and $max_k|c_k \cap \omega_j|$ returns the largest number of fingerprints in ground truth class ω_j that are also in a recovered cluster.

A high precision value means that each of the recovered clusters mostly contains images coming from a single camera. Conversely, if the clusters are polluted by images coming from different cameras, the precision will decrease. A high recall value means the the obtained clusters have recovered most of the ground truth clusters. When a ground truth cluster is split into two or more recovered clusters, the recall will decrease.

As an overall measure of clustering accuracy, we use the $F-measure$ defined as the harmonic mean of P and R:

$$F - measure = 2 \times \frac{(P \times R)}{(P + R)}. \tag{28}$$

The RI and ARI are computed using Eq. 29 and Eq. 30, respectively [48, 49, 50].

$$RI = \frac{(a + d)}{a + b + c + d} = \frac{(a + d)}{\binom{n}{2}}. \tag{29}$$

Where, a is the number of pairs of fingerprints which are in the same set in Ω and in the same set in C, b is the number of pairs of fingerprints which are in the same set in Ω and in different sets in C, c is the number of pairs of fingerprints which are in different sets in Ω and in the same set in C and d is the number of pairs of fingerprints which are in different sets in Ω and in different sets in C [51].

$$ARI = \frac{RI - \mathbf{E}[RI]}{1 - \mathbf{E}[RI]}. \tag{30}$$

The expected value of RI between two random partitions is not a constant. The problem is corrected by the ARI that assumes the generalized hypergeometric distribution as the model of randomness. The ARI has maximum value 1, and its expected value is 0 in the case of random clusters. A larger ARI means a higher agreement between two partitions. Therefore ARI is recommended for measuring agreement even when the partitions compared have different numbers of clusters.

Along with the quality of clusters of an algorithm, it is also essential evaluate its computational complexity, and for this purpose and a metric called complexity reduction Cr [16, 17] is introduced. The Cr evaluates the relative complexity of an algorithm with respect to the reference complexity $n(n-1)/2$ and is computed as given by Eq. 31

$$Cr = \frac{n \times (n-1)}{2 \times T_c^t}. \tag{31}$$

The higher the value of Cr, the faster the algorithm is and vice versa. The total computational complexity T_c^t is measured as the sum of the equivalent number of correlations between full camera fingerprints. This can be estimated as the actual number of correlation between full fingerprints NCC_Full plus a weighted number of correlations between the compressed camera fingerprints NCC_R, i.e., $T_c^t = NCC\_Full + \zeta NCC\_R$, where ζ is ratio of the size of compressed fingerprint in bits to the size of full fingerprint in bits, i.e., $\zeta = \frac{2 \times P_r}{64 \times |F|}$.

4.3. Experimental setup

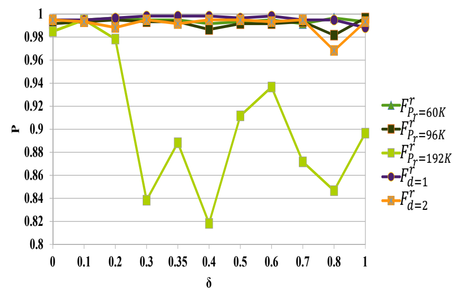
The datasets are first pre-processed so that all images share the same size. All images are center cropped to 1023×1023 pixels and camera fingerprints are extracted using the technique mentioned in [1, 2]. These are called full camera fingerprints. The center cropped images are then used to compute the ranking index $\Re I$ for each image. Each image X, in each dataset is used to compute average and normalized gray level G, saturation level S and texture T, using Eq. 6, Eq. 7 and Eq. 10, respectively. The G, S and T are further used to compute $\Re I$ for the image X, with $\alpha = 2$, $\beta = 0.5$ and $\gamma = 2$, as given in Eq. 12. The extracted full camera fingerprints are sorted in the decreasing order of $\Re I$.

In all experiments, the threshold is computed by setting $PFA = 10^{-6}$.

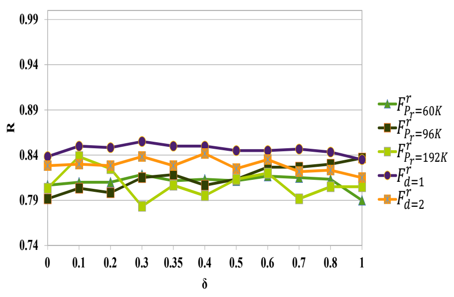
4.4. The analysis of FIC3F algorithm using different reduced fingerprints.

Reduced camera fingerprints can be obtained according to different settings. We can use decimation, random projections, and dead-zone quantization, as explained in Algorithm. 1 or we can apply dead-zone quantization directly to full fingerprint after decimation or without decimation as given in Algorithm. 2. Along with these, we may use a different number of projections and values of quantization factor δ to get the reduced fingerprints. Therefore, it is important to investigate which choice of parameters leads to the best clustering performance.

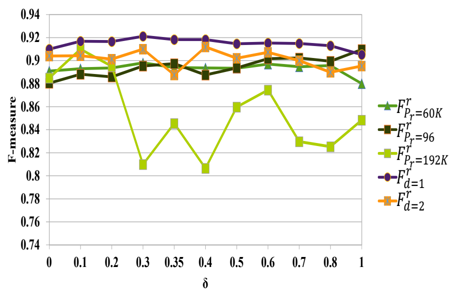
For Algorithm 1, we tested a decimation factor equal to 2, a random projection length P_r of 60000, 96000, or 192000, and the parameter δ for dead-zone quantization was chosen in the set $\{0, 0.1, 0.2, 0.3, 0.35, 0.4, 0.5, 0.6, 0.7, 0.8, 1\}$. For Algorithm 2, we tested the decimation factors $d = 1$ and $d = 2$, while the parameter δ of dead-zone quantization is chosen in the same set as above. To differentiate between the two types of reduced fingerprints, the reduced fingerprints estimated using the first method are represented by $F^r_{P_r}$ and those computed with the second method are represented by F^r_d.



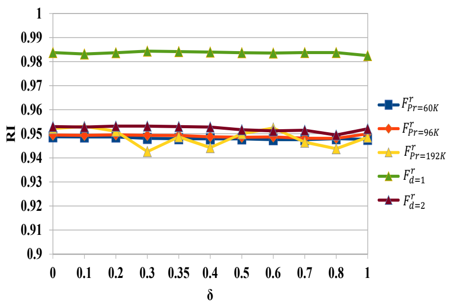
(a) P.



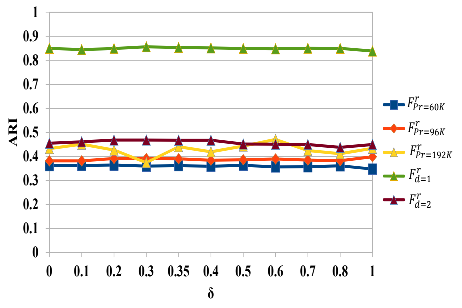
(b) R.



(c) $F - measure$.



(d) RI.

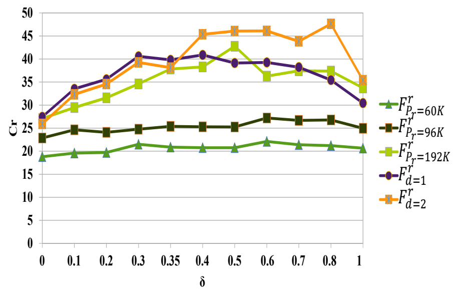


(e) ARI.

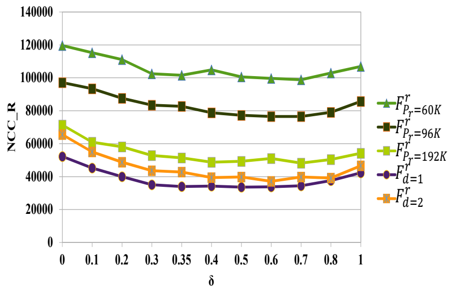
Figure 3: Evaluation measure based analysis of FIC3F algorithm using different reduced fingerprints Fr for different δ.

The results in Fig. 3a show that F_d^r at $d = 1$ results in clusters achieving the highest P of 0.998 for δ equal to 0.3, 0.35, 0.4 and 0.6. Along with the precision, the highest value 0.855 of R and highest value 0.921 of $F - measure$ are achieved by F_d^r at $d = 1$ for δ equal to 0.3, as shown in Fig. 3b and Fig. 3c, respectively. Similarly, the FIC3F algorithm using reduced fingerprint F_d^r at $d = 1$ and $\delta = 0.3$ results in the highest values of RI and ARI, as shown in Fig. 3d .

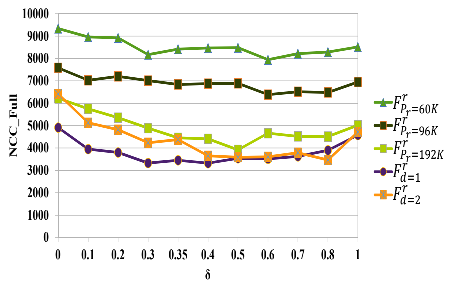
The results of Cr, NCC_R, and NCC_Full for different reduced fingerprints at different values of δ are shown in Fig. 4a, Fig. 4b and Fig. 4c, in that order. Considering the performance of the FIC3F algorithm in terms of complexity reduction, it has been observed that the FIC3F algorithm obtains the minimum value of NCC_R and NCC_Full when using F_d^r at $d = 1$ for δ equal to 0.35 and 0.3, respectively.



(a) Cr.



(b) NCC_R.



(c) NCC_Full.

Figure 4: Computational complexity based analysis of FIC3F algorithm using different reduced fingerprints Fr for different δ.

In short, the configuration that maximizes the complexity reduction without affecting the performance is that using F_d^r at $d = 1$ and $\delta = 0.3$. This can be explained because using only quantization avoids the approximation due to random projections, while still achieving a significant reduction in the complexity of fingerprint correlation operations. From now on, with reduced fingerprints

Fr we denote only the fingerprints estimated from full fingerprints without any decimation and using dead-zone quantization at $\delta = 0.3$.

4.5. Small Scale Clustering

In this section, we are investigating the performance of the FIC3F algorithm on symmetric easy $D1$, asymmetric easy $D2$, symmetric hard $D3$ and asymmetric hard $D4$ datasets, varying the number of images per camera SC while keeping the number of cameras fixed. The number of cameras in symmetric easy and asymmetric easy datasets is 25, while in the other two datasets the number of contributing cameras NC is 50. Full camera fingerprints F and reduced camera fingerprints F^r are estimated for each image of each dataset.

A number of 200, 400, 600, 800 and 1000 fingerprints are selected in each experiment corresponding to a SC of 8, 16, 24, 32 and 40 respectively in case of easy datasets and to a SC of 4, 8, 12, 16 and 20 in the case of hard datasets.

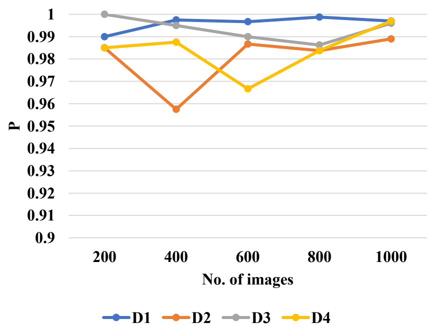
The experimental results for the different performance metrics are shown in Fig. 5. The results in Fig. 5a show that the FIC3F algorithm results in high values of P for almost all cases. A small reduction in P has been observed for 400 and 600 images for $D2$ and $D4$ datasets, respectively. The reduction in P can be due to the attraction of wrong fingerprints during the fine clustering stage. The results obtained for R and $F - measure$ in Fig. 5b-c show that FIC3F obtains a recall between 75% and 90% for the different datasets. The results also show that the recall is higher for the easy dataset compared to the hard ones. A similar behavior can be observed for the ARI on Fig. 3e, while the values of RI remains similar for easy and hard datasets.

The results in Fig. 5f show that the computational complexity of FIC3F, with respect to the reference complexity of $n(n-1)/2$, decreases with an increase in the number of fingerprints. Moreover, the complexity reduction is higher in case of easy datasets i.e., $D1$ and $D2$ than in the case of the hard datasets i.e., $D3$ and $D4$, showing that the FIC3F algorithm tends to be faster when there are less clusters.

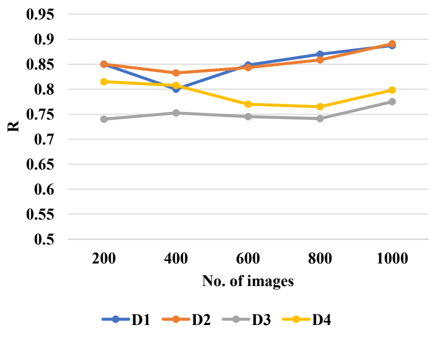
4.6. Large Scale Clustering

For large scale analysis the FIC3F algorithm is applied to different subsets of images selected from $D5$ dataset using the same number of cameras, i.e., $NC = 53$, and varying the average number of images from each camera SC. The experiments are performed using 106, 265, 371, 477, 530, 795, 1325, 1855, 2385, 2915, 3445 and 3975 images.

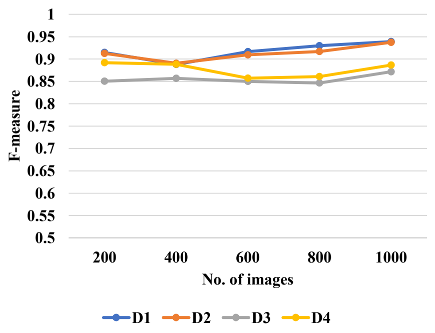
The experimental results in Fig. 6a show that the values of P, R and $F-measure$ decrease, when SC increases. The reduction in R and $F-measure$ can be due to the construction of more singleton clusters, i.e., clusters formed by a single fingerprint, in the initial clustering stage when SC increases and due to the difficulty of attracting singleton clusters in the fine clustering stage. While the reduction in P can be due to the attraction of some wrong fingerprints during the fine clustering stage. Fig. 4b shows that the values of RI are very high and remains stable while the values of ARI tends to decrease with the increasing number of images.



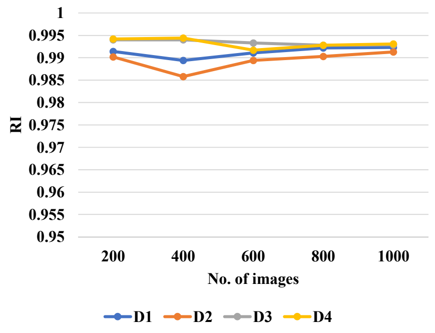
(a) P.



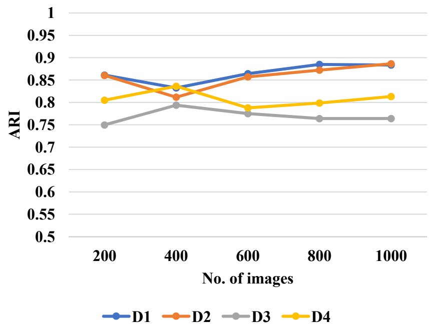
(b) R.



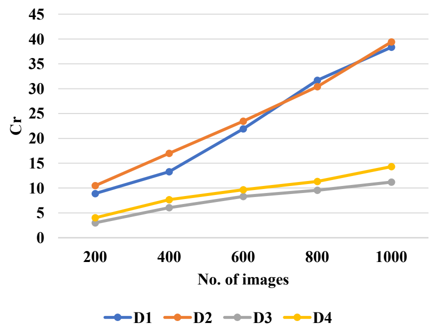
(c) $F - measure$.



(d) RI.

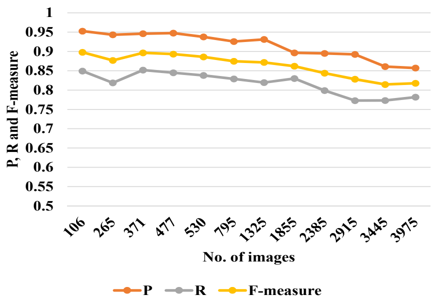


(e) ARI.

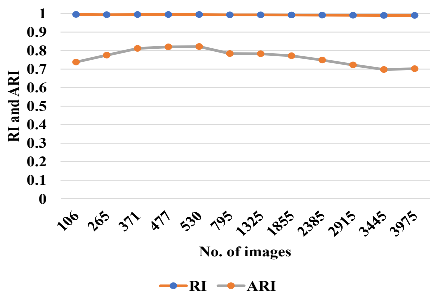


(f) Cr.

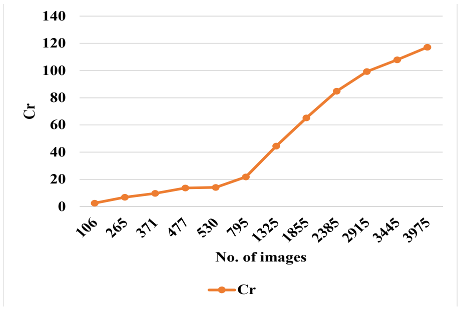
Figure 5: Performance of FIC3F algorithm on small datasets, i.e., $D1$, $D2$, $D3$ and $D4$



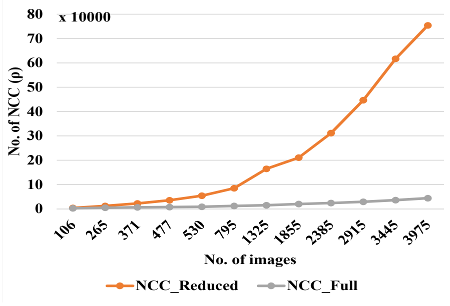
(a) P, R and $F-measure$.



(b) ARI.



(c) Cr.



(d) NCC_R and NCC_Full.

Figure 6: Large scale dataset analysis of FIC3F algorithm.

The results in Fig. 6c-d show that the complexity reduction of the FIC3F algorithm increases as the SC increases. It can be observed that $NCC\_Reduced$ and NCC_Full increase with an increase of SC, as shown in Fig. 6d. This increase is due the increase in the size of dataset. The clustering of a large number of images will perform a large number of NCC, both NCC_R and NCC_Full. However, the NCC_R and NCC_Full grow slower than the $n(n-1)/2$, as n increases. Hence, the overall Cr, which compares the total computational cost with the reference complexity $n(n-1)/2$, decreases.

4.7. $NC \gg SC$ analysis

In this section, we are evaluating the robustness of the FIC3F algorithm under the $NC \gg SC$ scenario on $D6$ dataset, which has a significantly larger NC than the datasets used in the previous experiments. The experiments are performed by keeping the size of clusters fixed and varying the number of cameras NC. The experimental results obtained for NC equal to 50, 75, 100, 125, 150, 175, 200, 250 and 295 with fixed $SC = 20$, are shown in Fig. 7. The results show that as the NC gets much larger than SC, the evaluation metrics of P, R and $F-measure$ improve, while the resulting RI and ARI values remain almost constant for the different configurations.

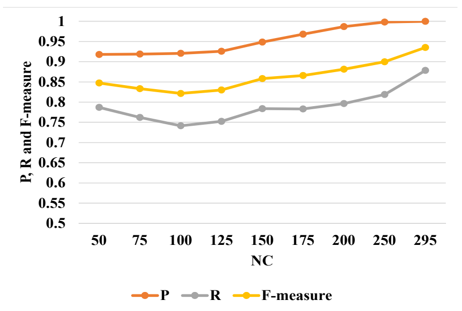
The results also show that the complexity reduction is not affected significantly by the number of clusters. This can be explained by the fact that the complexity reduction is mostly affected by the average size of the clusters, which is constant in this experiment.

Further experiments are performed for fixed $NC = 295$ and varying the SC. The experimental results obtained for SC equal to 2, 3, 5, 10, 15 and 20, are shown in Fig. 8. The results show that the evaluation metrics of P, R and $F-measure$ slightly decrease when SC increases. While the values of RI and ARI tends to be almost constant. Fig. 8c shows that as the average number of images per camera SC increases, the complexity reduction increases. Confirming that the complexity of the proposed algorithm mainly depends on the parameter SC.

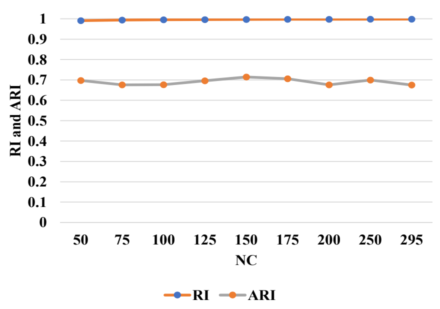
Overall, the above results show that FI3CF algorithm is robust in the $NC \gg SC$ scenario, since the performance is not significantly affected even in very unbalanced scenarios, while the complexity reduction mainly depends on the average size of clusters and is not affected by the number of clusters.

4.8. Comparison

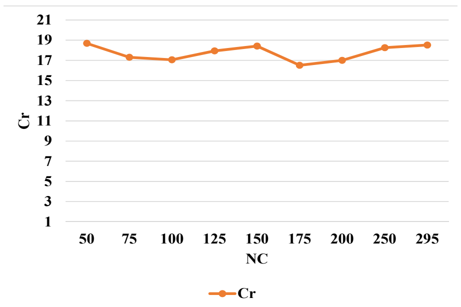
The main focus of the FIC3F algorithm is to reduce the computational complexity per image while preserving the quality of the constructed clusters. In this section, the FIC3F algorithm is compared with state-of-the-art algorithms for camera fingerprint clustering, namely the blind camera fingerprinting, image clustering (BCFIC) algorithm [18], the large scale clustering (LSC) algorithm [19], the reduced complexity image clustering algorithm without attraction (RCIC) and with attraction (RCIC-A) [16] and fast image clustering algorithm based on camera fingerprint ordering without attraction (FICFO) and



(a) P, R and $F - measure$.

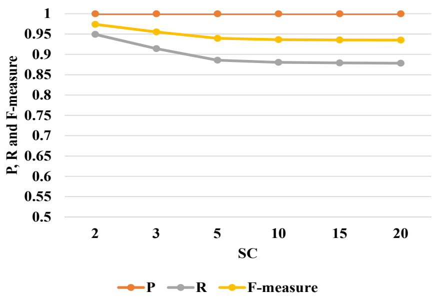


(b) ARI.

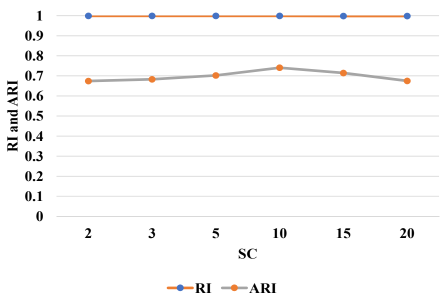


(c) Cr.

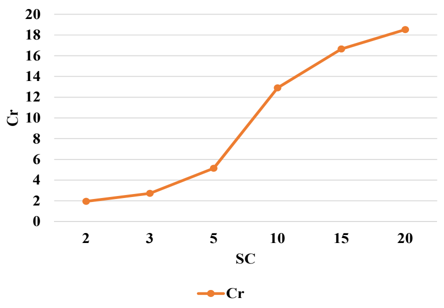
Figure 7: The robustness of FIC3F algorithm to $NC \gg SC$ problem, for various values of NC and fixed $SC = 20$.



(a) P, R and $F - measure$.



(b) ARI.



(c) Cr.

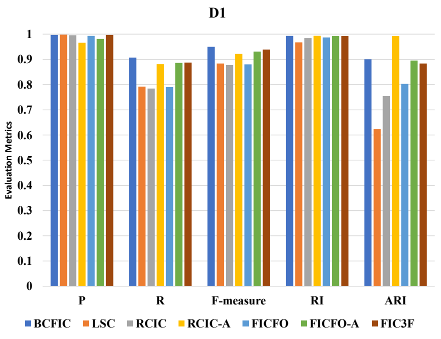
Figure 8: The robustness of FIC3F algorithm to $NC \gg SC$ problem, for various values of SC and fixed $NC = 295$.

with attraction (FICFO-A) [17] using the four datasets $D1$, $D2$, $D3$, and $D4$ [19]. Since the RCIC and FICFO algorithms can be seen as a baseline version of the proposed algorithm without using compression, the following results permit to directly appreciate the effect of compression on the clustering.

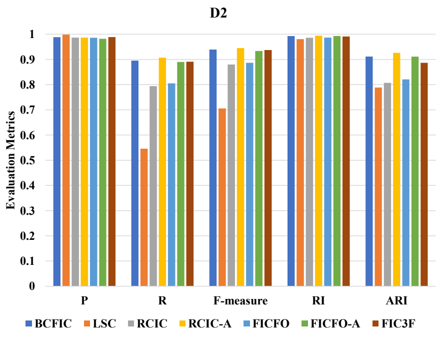
The results in Fig. 9a show that the FIC3F algorithm has a comparable P with state-of-the-art BCFIC and LSC algorithms. However, the FIC3F algorithm performs similar or better than RCIC, RCIC-A, FICFO and FICFO-A algorithms in terms of P. While, the R and $F - measure$ of the proposed algorithm is comparable with BCFIC algorithm and higher than that obtained by LSC, RCIC, RCIC-A, FICFO and FICFO-A algorithms. However, in case of $D4$ the FICFO-A has slightly higher R than the proposed algorithm, as shown in Fig. 9d. The resulting values of RI show that the FIC3F algorithm has a performance equivalent to that of the other clustering algorithms. In terms of ARI, the FIC3F algorithm is less efficient than BCFIC, RCIC-A and FICFO-A on easy datasets. Similarly, on hard datasets the RCIC-A and FICFO-A perform better than the FIC3F algorithm, but the performance of the BCFIC algorithm is lower than the proposed algorithm. The ARI of the proposed algorithm is higher than the rest of the clustering algorithms i.e., LSC, RCIC and FICFO.

The complexity reduction Cr obtained by all the algorithms is presented in Fig. 9e. The total complexity t_c of BCFIC, RCIC, RCIC-A, FICFO, and FICFO-A algorithms is computed by counting the number of correlations, since these algorithms use only full camera fingerprints. However, the FIC3F algorithm and the LSC algorithm use both reduced and full camera fingerprints for clustering, therefore, the total complexity t_c for these algorithms is computed in a different way. Since the two algorithms use different versions of reduced fingerprints, the number of correlation operations performed on reduced and full fingerprints are weighted differently. In case of LSC, the total complexity t_c is calculated as $t_c = ncf + (r/|F|) \times ncr$, where, ncf and ncr are the number of correlation among full and reduced fingerprints respectively, while r is the sizes of reduced fingerprints. While, in the case of FIC3F, total computational cost is computed as $t_c = NCC\_Full + (\zeta \times NCC\_R)$, where $\zeta = \frac{2 \times P_r}{64 \times |F|}$, while P_r and $|F|$ are the dimensions of the reduced fingerprints and full fingerprints, respectively. Here it is important to mention that FICFO, FICFO-A, and FIC3F algorithms perform some computation while calculating G, S, T, and $\Re I$ and also in sorting fingerprints. However, the cost of calculating G, S, T and $\Re I$ is negligible with respect to the estimation of fingerprints. The cost of sorting fingerprints is also far less than computing the correlations of very long vectors. Therefore, the cost of computing $\Re I$ and sorting fingerprints is neglected, while computing the total computational complexity t_c and the corresponding complexity reduction.

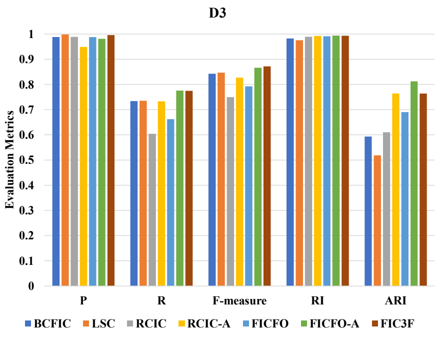
The results show that the FIC3F algorithms has lower computational complexity than the state-of-the-art methods. The reduction in the computational cost is mainly due to the use of sorted reduced fingerprints for clustering. BCFIC algorithm has a high computation cost because it performs three rounds to construct a single cluster. These rounds are repeated for each cluster. Conversely,



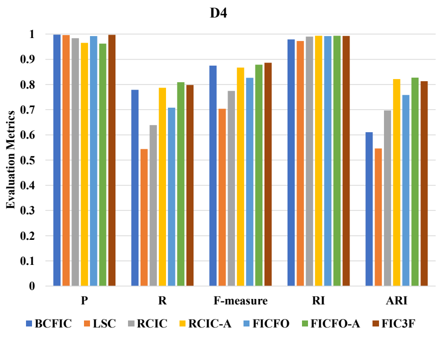
(a) $D1$.



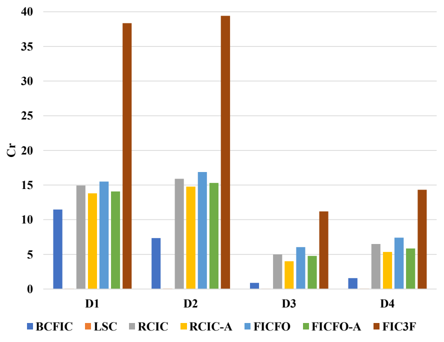
(b) $D2$.



(c) $D3$.



(d) $D4$.



(e) Cr.

Figure 9: Comparison of the FIC3F with state-of-the-art clustering techniques.

the proposed algorithm picks the best reduced fingerprint to build a cluster and performs a small number of correlations on full fingerprints in the fine clustering stage. The complexity of the LSC algorithm is quite evident due to coarse clustering, fine clustering, and attraction. The complexity of RCIC, RCIC-A, FICFO, and FICFOA algorithms is higher than the proposed algorithm due to the use of full fingerprints for clustering as well as in attraction, whenever used.

5. Conclusions

In this paper we have introduced a fast and efficient image clustering algorithm to group images based on their camera fingerprints. The proposed algorithm computes a ranking index $\Re I$ indicating the quality of each estimated fingerprint and sorts all fingerprints in descending order of $\Re I$. Using the highest quality fingerprints as attractors, an initial clustering stage constructs coarse clusters in a fast way based on a compressed version of the fingerprints. Then, the full fingerprints of each initial cluster are merged together by averaging them and the highest quality clusters are used as attractors to merge and refine the clusters. The results obtained on different subsets of the Dresden dataset show that the proposed clustering algorithm performs similarly or better than prior related work, with a significantly lower computational complexity. Namely, the results show that the initial clustering stage perform most of the correlations among the reduced fingerprints, while a very small number of correlations are performed on full camera fingerprints during the fine clustering stage. The proposed algorithm is suitable for large datasets since computational complexity per image decreases as the size of the image dataset increases. At the same time, the proposed algorithm is also robust when the size of clusters is small compared to the number of cameras, which is a typical problem in this kind of application.

References

[1] J. Lukás, J. Fridrich, , M. Goljan, Digital camera identification from sensor pattern noise, IEEE Trans. Inf. Forensics Security. 1 (2) (2006) 205–214.

[2] M. Chen, J. Fridrich, M. Goljan, J. Lukás, Determining image origin and integrity using sensor noise, IEEE Trans. Inf. Forensics Security. 3 (1) (2008) 74–90.

[3] C. Li, Y. Li, Digital camera identification using colour-decoupled photo response non-uniformity noise pattern, in: Proc. IEEE Int. Symp. Circuits Syst., IEEE, 2010, pp. 3052–3055.

[4] T. Filler, J. Fridrich, M. Goljan, Using sensor pattern noise for camera model identification, in: 15th IEEE Int. Conf. Image Process., IEEE, 2008, pp. 1296–1299.

[5] Q. Phan, G. Boato, F. De Natale, Image clustering by source camera via sparse representation, in: Proc. Int. Workshop on Multi. Forensics Secur., ACM, 2017, pp. 1–5.

[6] F. Bertini, R. Sharma, A. Iannì, D. Montesi, Profile resolution across multilayer networks through smartphone camera fingerprint, in: Proceedings of the 19th International Database Engineering & Applications Symposium, IDEAS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 23–32.

[7] S. Georgievska, R. Bakhshi, A. Gavai, A. Sclocco, B. van Werkhoven, Clustering image noise patterns by embedding and visualization for common source camera detection, Digital Investigation. 23 (2017) 22–30.

[8] C. Li, Source camera identification using enhanced sensor pattern noise, IEEE Trans. Acoust., Speech, Signal Process. 5 (2) (2010) 280–287.

[9] R. Ng, J. Han, CLARANS: A method for clustering objects for spatial data mining, IEEE Trans. Knowl. Data Eng. 14 (5) (2002) 1003–1016.

[10] S. Guha, R. Rastogi, K. Shim., CURE: An efficient clustering algorithm for large databases, ACM SIGMOD Rec. 27 (2) (1998) 73–84.

[11] C. Li, Unsupervised classification of digital images using enhanced sensor pattern noise, in: Proc. IEEE Int. Symp.Circuits Syst., IEEE, 2010, pp. 3429–3432.

[12] B. Liu, H. Lee, Y. Hu, C. Choi, On classification of source cameras: A graph based approach, in: IEEE Int. Workshop Inf. Forensics Secur., IEEE, 2013, pp. 1–5.

[13] S. Yu, J. Shi, Multiclass spectral clustering, in: IEEE Int. Conf. Comput. Vis., IEEE, 2003, pp. 313–319.

[14] R. Caldelli, I. Amerini, F. Picchioni, M. Innocenti, Fast image clustering of unknown source images, in: Proc. IEEE Int. Workshop Inf. Forensics Secur., IEEE, 2010, pp. 1–5.

[15] L. Villalba, A. Orozco, J. Corripio, Smartphone image clustering, Expert Syst. Appl. 42 (2015) 1927–1940.

[16] S. Khan, T. Bianchi, Reduced complexity image clustering based on camera fingerprints, in: 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 2682–2688.

[17] S. Khan, T. Bianchi, Fast image clustering based on camera fingerprint ordering, in: International Conference on Multimedia and Expo (ICME) 2019, Shanghai, China, IEEE, 2019.

[18] G. Bloy, Blind camera fingerprinting and image clustering, IEEE Trans. Pattern Anal. Mach. Intell. 30 (3) (2008) 532–534.

[19] X. Lin, C. Li, Large-scale image clustering based on camera fingerprints, IEEE Trans. Inf. Forensics Security. 12 (4) (2017) 793–808.

[20] W. Equitz, A new vector quantization clustering algorithm, IEEE Trans. Acoust., Speech, Signal Process. 37 (10) (1989) 1568–1575.

[21] O. Fahmy, An efficient clustering technique for cameras identification using sensor pattern noise, in: Proc. Int. Conf. Syst., Signals and Image Process., IEEE, 2015, pp. 249–252.

[22] F. Gisolf, P. Barens, E. Snel, A. Malgoezar, M. Vos, A. Mieremet, Z. Geradts, Common source identification of images in large databases, Forensic Sci. Int. 44 (2014) 222–230.

[23] C. Li, X. Lin, A fast source-oriented image clustering method for digital forensics, EURASIP J. Image Video Process. 1 (2017) 69.

[24] Q. Phan, G. Boato, F. D. Natale., Accurate and scalable image clustering based on sparse representation of camera fingerprint, arXiv preprint arXiv:1810.07945.

[25] E. Elhamifar, R. Vidal, Sparse subspace clustering, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., IEEE, 2009, pp. 2790–2797.

[26] A. K. Jain, R. C. Dubes, Algorithms for clustering data.

[27] M. Ester, H. P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.

[28] S. Guha, R. Rastogi, K. Shim, ROCK: A robust clustering algorithm for categorical attributes, Information systems 25 (5) (2000) 345–366.

[29] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: ACM Sigmod Record, Vol. 25, ACM, 1996, pp. 103–114.

[30] G. Karypis, E. H. Han, V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.

[31] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, S. Tubaro, First steps toward camera model identification with convolutional neural networks, IEEE Signal Processing Letters 24 (3) (2017) 259–263.

[32] D. Cozzolino, L. Verdoliva, Noiseprint: A CNN-based camera model fingerprint, IEEE Transactions on Information Forensics and Security 15 (2020) 144–159.

[33] D. Cozzolino, F. Marra, D. Gragnaniello, G. Poggi, L. Verdoliva, Combining PRNU and noiseprint for robust and efficient device source identification, EURASIP Journal on Information Security 2020 (1).

[34] S. Mandelli, D. Cozzolino, P. Bestagini, L. Verdoliva, S. Tubaro, CNN-based fast source device identification, IEEE Signal Processing Letters 27 (2020) 1285–1289.

[35] J. Janesick, Scientific charge-coupled devices, in: SPIE press, Vol. 83, Bellingham, Washington USA, 2001.

[36] M. Chen, J. Fridrich, M. Goljan, Digital imaging sensor identification (further study), in: Security, steganography, and watermarking of multimedia contents IX, Vol. 6505, International Society for Optics and Photonics, 2007, p. 65050P.

[37] M. Goljan, J. Fridrich, T. Filler, Managing a large database of camera fingerprints, Proc. SPIE 7541 (2010) 754108.

[38] D. Valsesia, G. Coluccia, T. Bianchi, E. Magli, Compressed fingerprint matching and camera identification via random projections, IEEE Trans. Inf. Forensics Security 10 (7) (2015) 1472–1485.

[39] S. Bayram, H. T. Sencar, N. Memon, Efficient sensor fingerprint matching through fingerprint binarization, IEEE Trans. Inf. Forensics Security 7 (4) (2012) 1404–1413.

[40] L. Bondi, P. Bestagini, F. Perez-Gonzalez, S. Tubaro, Improving PRNU compression through preprocessing, quantization, and coding, IEEE Trans. Inf. Forensics Security 14 (3) (2019) 608–620.

[41] M. Goljan, M. Chen, P. Comesaña, J. Fridrich, Effect of compression on sensor-fingerprint based camera identification, Electron. Imag. 2016 (8) (2016) 1–10.

[42] R. Keys, Cubic convolution interpolation for digital image processing, IEEE Trans. Acoust., Speech, Signal Process. 6 (3) (9) 1153–1160.

[43] T. Gloe, R. Böhme, The ‘Dresden image database’ for benchmarking digital image forensics, J. Digit. Forensic Pract. 3 (2-4) (2010) 1584–1590.

[44] T. Gloe, S. Pfennig, M. Kirchner, Unexpected artefacts in PRNU based camera identification: A ‘dresden image database’ case-study, in: Proc. ACM Workshop Multi. Secur., ACM, 2012, pp. 109–114.

[45] M. de Souto, A. Coelho, K. Faceli, T. Sakata, V. Bonadia, I. Costa, A comparison of external clustering evaluation indices in the context of imbalanced data sets, in: 2012 Brazilian Symposium on Neural Networks, Bellingham, Washington USA, 2012, pp. 49–54.

[46] N. Vinhn, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary?, in: Proceedings of the 26th annual international conference on machine learning, ACM, 2009, pp. 1073–1080.

[47] A. Amelio, C. Pizzuti, Is normalized mutual information a fair measure for comparing community detection methods?, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, Paris, France, ACM, 2015, pp. 1584–1585.

[48] W. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association. 66 (336) (1971) 846–850.

[49] L. Hubert, P. Arabie, Comparing partitions, Journal of classification. 2 (1985) 193–218.

[50] K. Yeung, W. Ruzzu, Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data, Bioinformatics. 17 (9) (2001) 763–774.

[51] M. Huffman, D. Steinley, M. Brusco, A note on using the adjusted rand index for link prediction in networks, Social networks. 42 (2015) 72–79.