Accurate and Scalable Image Clustering Based On Sparse Representation of Camera Fingerprint

Quoc-Tin Phan, Giulia Boato, Francesco G. B. De Natale

Abstract—**Clustering images according to their acquisition devices is a well-known problem in multimedia forensics, which is typically faced by means of camera Sensor Pattern Noise (SPN). Such an issue is challenging since SPN is a noise-like signal, hard to be estimated and easy to be attenuated or destroyed by many factors. Moreover, the high dimensionality of SPN hinders large-scale applications. Existing approaches are typically based on the correlation among SPNs in the pixel domain, which might not be able to capture intrinsic data structure in union of vector subspaces. In this paper, we propose an accurate clustering framework, which exploits linear dependencies among SPNs in their intrinsic vector subspaces. Such dependencies are encoded under sparse representations which are obtained by solving a LASSO problem with non-negativity constraint. The proposed framework is highly accurate in number of clusters estimation and image association. Moreover, our framework is scalable to the number of images and robust against double JPEG compression as well as the presence of outliers, owning big potential for real-world applications. Experimental results on Dresden and Vision database show that our proposed framework can adapt well to both medium-scale and large-scale contexts, and outperforms state-of-the-art methods.**

Index Terms—**Image clustering, sensor pattern noise, sparse subspace clustering, divide-and-conquer.**

I. INTRODUCTION

Getting information about the camera used to acquire an image provides forensic analysts with important cues to counterfeit digital crime. Occasionally, such information can be extracted from the attached metadata, e.g., the Exif header. However, this may be unavailable or can be easily modified or swept out even by non-experts. A more interesting option would be to detect the source directly from the image data. To this purpose, it has been observed that digital images intrinsically contain a Sensor Pattern Noise (SPN) caused by sensor imperfections of the capturing device [1], [2]. Just like human fingerprints, SPN uniquely identifies the acquisition source and can be considered as a *camera fingerprint*. In blind scenarios, where only a set of unsourced images are given, such fingerprint can reveal images that share the same source. We refer to this task as image clustering by source camera. Further investigations, for instance detecting how many cameras a suspect owns, or how likely an image is taken from the suspect's camera, can derive from clustering results. Indexing images by source camera also leads to direct applications in large-scale image retrieval.

In order to properly estimate the camera fingerprint, a number of smooth and uniformly bright images should be

collected [2]. Unfortunately, this requirement is usually not fulfilled in a blind scenario, where all images are unlabeled and no assumption can be made about the visual content. Consequently, SPN can just be coarsely approximated from the noise residual of a single image, which contains not only the pattern noise but also various other noise sources, such as shot noise and noise resulting from lossy compression or other filtering. Different methods have been proposed to enhance the SPN estimation and matching, such as averaging [1], PCA+LDA [3], spectrum equalization [4]. All those methods, however, require a labeled training set, making them unsuitable for image clustering by source camera in unsupervised scenarios.

Existing unsupervised techniques are typically based on the normalized correlation among SPNs, used as a similarity measure, whose degree of reliability is limited by the impact of multiple noise sources. In [5], an image is assigned to a group if the correlation between its noise residual and the relevant centroid exceeds a threshold, approximated by a quadratic model. Markov Random Fields are applied in [6], [7] to iteratively assign a class label to an image based on the consensus of a small set of SPNs, called membership committee. This raises another problem on how to choose a good committee, especially on asymmetric datasets where cluster cardinalities are unbalanced. In [8], [9], [10], a hierarchical partition - a binary tree containing singleton clusters as leaf nodes and whose root node is a cluster containing all data points - is built by hierarchical clustering. The major problem of existing hierarchical approaches is the sensitivity to noise and outliers, as a wrong assignment might result in the propagation of errors to higher tiers. Multiclass spectral clustering is applied in [11] to partition an undirected graph of unsourced images. The algorithm starts with two clusters and stops when it finds a cluster containing only one member. This stopping condition is heuristic, and as been improved by using normalized cut in [12]. Recently, in [13], [14], multiple base partitions are obtained on top of multiple binarized undirected graphs and then combined to form a complete clustering solution.

Another important problem that has to be taken into account is scalability. In practical applications, often the clustering has to be applied to large databases, containing huge numbers of high-resolution images. To the best of our knowledge, only the method in [15] addresses large-scale clustering of camera fingerprints, where the main idea is to split the dataset into small batches, which can be efficiently loaded on RAM, and to apply a coarse-to-fine clustering.

In the present work, we propose a clustering framework that exploits linear dependencies among SPNs in their intrinsic vector subspaces. Such dependencies are encoded under sparse representations, which are solutions of a constrained LASSO

The authors are with the Department of Information Engineering and Computer Science, University of Trento, Trento 38123, (email: {quoctin.phan, giulia.boato, francesco.denatale}@unitn.it) and also affiliated to the National Inter-University Consortium for Telecommunications (CNIT).

problem. Well-known clustering methods can then be applied on top of sparse representation matrix to obtain the final segmentation. Our framework is scalable despite of the complexity of LASSO thanks to a divide-and-conquer mechanism, which allows clustering on large datasets. Experimental tests on medium-scale and large-scale contexts exhibit advantages of sparse representations on clustering performance. The robustness of our proposed framework is demonstrated against the presence of outliers and double JPEG compression. A similar approach has been preliminarily described in [16]. Differently from [16], here we impose a non-negativity constraint that provides the *interpretability* of solutions, thus allowing the extension to large-scale contexts.

The rest of the paper is organized as follows: in Section II we describe the extraction of SPNs and the sparse subspace clustering method; in Section III we present the proposed optimization problem and its solution, as well as a clustering framework for large-scale datasets. Finally, discussions on computational complexity and extensive experimental analysis are provided in Section IV and V, respectively.

II. PRELIMINARIES

Throughout the paper, elements of a matrix or a vector are indicated by subscripts and single subscript $\mathbf{M}_i$ denotes the ith column of $\mathbf{M}$. With $\text{diag}(\mathbf{M})$ we denote the matrix containing only the diagonal of $\mathbf{M}$, all other entries being set to zero. Infinity norm of $\mathbf{y}$ is defined as $\|\mathbf{y}\|_\infty = \max_i |\mathbf{y}_i|$, while ℓ_1 and ℓ_2 norms of vector $\mathbf{y}$ are denoted as $\|\mathbf{y}\|_1$ and $\|\mathbf{y}\|_2$, respectively. For matrices, ℓ_1 norm, Frobenius norm and infinity norm are respectively defined as: $\|\mathbf{M}\|_1 = \sum_{i,j} |\mathbf{M}_{ij}|$,

$\|\mathbf{M}\|_F = \sqrt{\sum_{i,j} \mathbf{M}_{ij}^2}$, $\|\mathbf{M}\|_\infty = \max_{i,j} |\mathbf{M}_{ij}|$.

A. Sensor Pattern Noise

Given a grayscale image $\mathbf{Y}$, its noise residual $\mathbf{W}$ can be extracted by a denoising filter. A simplified model of $\mathbf{W}$ can be expressed as follows [17], [2]:

$$\mathbf{W} = \mathbf{TYK} + \mathbf{\Xi}, \tag{1}$$

where $\mathbf{\Xi}$ is a the matrix of independently and identically distributed (i.i.d) Gaussian random variables, $\mathbf{T}$ is an attenuation matrix, and $\mathbf{K}$ is referred to as Photo-Response Non-Uniformity (PRNU). In theory, PRNU can be used to cluster images with respect to the acquisition device. However, in a blind scenario this is difficult due to two main problems. First, the Cramer-Rao Lower Bound on the variance of PRNU estimate indicates that a number of smooth and bright (but not saturated) images are required for each camera [2] and this condition is hardly satisfied. Indeed, the only available information is the noise residual $\mathbf{W}$ for each image, which contains not only the PRNU but also the additive noise $\mathbf{\Xi}$, which limits the reliability of traditional similarity measures used in conventional clustering algorithms. Several methods have been proposed for SPN enhancement [18], [4] but it has been confirmed by [15] that such methods are not suitable for

unsupervised setting. Second, the dimension of camera fingerprints is usually high, due to the high resolution of camera sensors, thus their clustering requires huge computation and memory as long as the number of data increases.

Existing approaches use normalized correlation to measure the similarity between two flattened fingerprints $\mathbf{a}, \mathbf{b}$ of dimension d:

$$\rho(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^d (\mathbf{a}_i - \bar{\mathbf{a}})(\mathbf{b}_i - \bar{\mathbf{b}})}{\sqrt{\sum_{i=1}^d (\mathbf{a}_i - \bar{\mathbf{a}})^2}\sqrt{\sum_{i=1}^d (\mathbf{b}_i - \bar{\mathbf{b}})^2}}, \tag{2}$$

where scalars $\bar{\mathbf{a}}, \bar{\mathbf{b}}$ are the mean values of $\mathbf{a}$ and $\mathbf{b}$, respectively. Without loss of generality, if $\mathbf{a}, \mathbf{b}$ are normalized to have zero mean and unit norm, Eq. (2) simply becomes:

$$\rho(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^d \mathbf{a}_i \mathbf{b}_i,$$

which represents the cosine similarity between $\mathbf{a}$ and $\mathbf{b}$.

B. Sparse Representation

Given a set of data points arranged into the columns of a matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, a data point $\mathbf{y}$ can be expressed as a linear combination of the columns of $\mathbf{X}$. A *sparse* combination reveals columns in the same subspace which $\mathbf{y}$ happens to lie into. Sparse Subspace Clustering (SSC) [19] finds a sparse representation of $\mathbf{y}$ by solving the following optimization problem:

$$\underset{\mathbf{z}}{\text{minimize}} \quad \|\mathbf{z}\|_1 \quad \text{subject to} \quad \mathbf{Xz} = \mathbf{y}. \tag{3}$$

If columns of $\mathbf{X}$ are contaminated by noise or not well distributed, $\mathbf{Xz} = \mathbf{y}$ might never be reached. Works in [20], [21] have shown that SSC can deal with noisy data if Eq. (3) is reformulated as a LASSO problem:

$$\underset{\mathbf{z}}{\text{minimize}} \quad \|\mathbf{Xz} - \mathbf{y}\|_2^2 + \gamma\|\mathbf{z}\|_1, \tag{4}$$

where $\gamma \geq 0$ is a regularization hyperparameter.

Let us show a simple example to interpret the solutions of SSC. As illustrated in Figure 1, we have $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4] \in \mathbb{R}^{3 \times 4}$. We assume there are two subspaces spanned by $[\mathbf{X}_1, \mathbf{X}_2]$ and $[\mathbf{X}_3, \mathbf{X}_4]$. Without the regularization term, $\mathbf{y}$ can be expressed as linear combination of any 3 columns of $\mathbf{X}$. However, we would like to assign $\mathbf{y}$ to the closest subspace spanned by $[\mathbf{X}_1, \mathbf{X}_2]$, i.e., $\|\hat{\mathbf{y}} - \mathbf{y}\|_2 < \|\tilde{\mathbf{y}} - \mathbf{y}\|_2$, a preferable solution should satisfy:

$$\hat{\mathbf{y}} = \mathbf{z}_1 \mathbf{X}_1 + \mathbf{z}_2 \mathbf{X}_2.$$

Such solution can be reached if we encourage the sparseness of $\mathbf{z}$ by penalizing $\|\mathbf{z}\|_0$. Unfortunately, ℓ_0 optimization is usually intractable due to its non-convex and combinatorial nature. $\|\mathbf{z}\|_1$ can be used instead, being a good approximation of $\|\mathbf{z}\|_0$ [22]. Exploiting the ℓ_1 regularization term as in Eq. (4) and properly selecting γ, SSC finds a sparse solution that minimizes the reconstruction error.
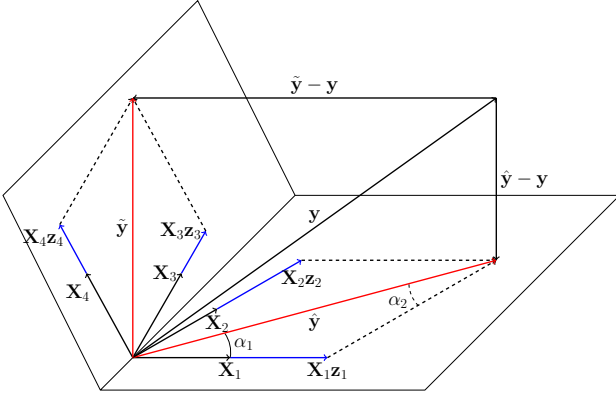
Fig. 1. Geometric interpretation of solution of SSC.

- *Proposed optimization* (Section III-B): we present a constrained optimization problem to retrieve sparse and interpretable solutions.
- *Extension to large-scale contexts* (Section III-C): we design a divide-and-conquer mechanism enabling large-scale clustering.

A. Fingerprint Extraction and Normalization

In this study, we do not make any assumption on image content, but we simply filter out dark images (if any) since dark or textured images are inappropriate for fingerprint estimation. An image is considered dark if more than 75% of pixels have values in $[0, 80]$.

A noise residual $\mathbf{W}^c$, $c \in \{\text{red, green, blue}\}$ is extracted from each color channel of $\mathbf{Y}$, by exploiting the wavelet-based denoising filter used in [1], [17], [2], [23], and then converted to one-channel noise residual. To further suppress non-unique artifacts caused by color interpolation of demosaicing algorithms and standard JPEG compression, we subtract from each row the mean of rows and from each column the mean of columns, and transform the obtained noise residual $\mathbf{W}$ into a one-dimensional unit-norm signal. We then obtain a data matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, n being the number of fingerprints, and d the number of pixels.

B. Proposed optimization

SSC learns a sparse representation $\mathbf{z}$ of $\mathbf{y}$, whose non-zero entries indicate data points closest to the orthogonal projection of $\mathbf{y}$ onto the relevant subspace. We can *interpret* the magnitude of $\mathbf{z}_i$ as a similarity measure: the closer $\mathbf{X}_i$ is to $\hat{\mathbf{y}}$, the more it contributes to the reconstruction of $\mathbf{y}$, resulting in a larger value of $\mathbf{z}_i$. Back to the example in Figure 1, denoted as $\alpha_i = \angle (\mathbf{X}_i, \hat{\mathbf{y}})$ the angle between $\hat{\mathbf{y}}$ and $\mathbf{X}_i$, it is easily to see that if $\alpha_i < \alpha_j$ then $|\mathbf{z}_i| > |\mathbf{z}_j|$:

$$\alpha_i < \alpha_j \Leftrightarrow \cos(\alpha_i) > \cos(\alpha_j) \Leftrightarrow \|\mathbf{z}_i \mathbf{X}_i\|_2 > \|\mathbf{z}_j \mathbf{X}_j\|_2$$

$$\Leftrightarrow |\mathbf{z}_i| > |\mathbf{z}_j| \, (\text{since } \|\mathbf{X}_i\|_2 = 1).$$

Thus, $|\mathbf{z}_i|$ is inversely propotional to α_i. The ℓ_1 regularization term encourages the sparseness of $\mathbf{z}$, whose non-zero entries should indicate data points *closest* to $\hat{\mathbf{y}}$. Due to the nature of ℓ_1 norm, however, negative and positive contributions are weighted equally. We provide in Figure 2 an example where solution $\tilde{\mathbf{z}} = [-\mathbf{z}'_1, -\mathbf{z}'_2]^T$ might be chosen instead of $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]^T$ as it is possible that $\|\tilde{\mathbf{z}}\|_1 < \|\mathbf{z}\|_1$. Note, in this example $\mathbf{z}_i > 0$ and $\mathbf{z}'_i > 0$, thus $\tilde{\mathbf{z}}$ contains negative entries. This is an unexpected solution since $\mathbf{X}'_1, \mathbf{X}'_2$ lie in another half-space, i.e., $\angle(\mathbf{X}'_1, \hat{\mathbf{y}}) > \pi/2$ and $\angle(\mathbf{X}'_2, \hat{\mathbf{y}}) > \pi/2$. In order to avoid this solution, all entries $\mathbf{z}_i$ can be constrained to non-negative values. Therefore, the optimal solution reveals data points lying in the subspace closest to $\mathbf{y}$ and correlated to the orthogonal projection of $\mathbf{y}$ on that subspace. Another intesting property is that if $\mathbf{z}_i \geq 0$, $\mathbf{z}_j \geq 0$ and $\alpha_i < \alpha_j$ then $\mathbf{z}_i > \mathbf{z}_j$. This motivates us to impose a non-negativity constraint on the optimization problem.

For each column $\mathbf{X}_i$, we expect to learn a sparse representation $\mathbf{Z}_i$ such that $\mathbf{X}_i = \mathbf{X}\mathbf{Z}_i$. To obtain a meaningful

C. Motivation

Even if two fingerprints come from the same camera, their normalized correlation is very weak due to the presence of irrelevant pixels. Only a subset of pixels are supposed to be relevant for each camera. Finding the subset of relevant pixels is connected to finding the subspace where fingerprints of a camera happen to lie in. Noticeable efforts in the direction of dimensionality reduction, such as fingerprint digest [23] or random projection [24], cannot be applied in clustering problems. In fact, [24] considers a single subspace, while different cameras have different subsets of relevant pixels. On the other hand, fingerprint digest is composed only by saturated values of the reference fingerprint, which can be extracted only if the common source of images is known.

SSC has been exploited to find structure of data in their intrinsic subspaces. Indeed, SSC formulated under LASSO problem works in broad conditions: theoretical guarantees had been provided when subspaces intersect [25], or in the presence of additive noise [20] even if the level of noise is higher than level of signal [21], demonstrating that in practice SSC can reliably recover cluster memberships. Moreover, SPNs are known as compressible signals: low-dimensional representations of SPNs are found in [3], [26], [27], [24], [28], thus implying the existence of subspaces that can well represent them.

Finally, we observe that the residual $\mathbf{W}$ is a noisy estimate of true camera fingerprint, thus the distributions of intra-class and inter-class correlations computed on $\mathbf{W}$ are heavily overlappped, making clustering algorithms less accurate. This challenge raises the need to eliminate inter-class data relationships and obtain unambiguous underlying data structure. By leveraging sparsity, SSC expresses each data point by a few linear relationships with other data points and extracts unambigous representation, which is essential for this problem.

III. PROPOSED METHOD

In this section, we present our clustering framework including three steps:

- *Fingerprint extraction and normalization* (Section III-A): given a set of color images as input, we extract, refine and normalize the corresponding noise residuals.
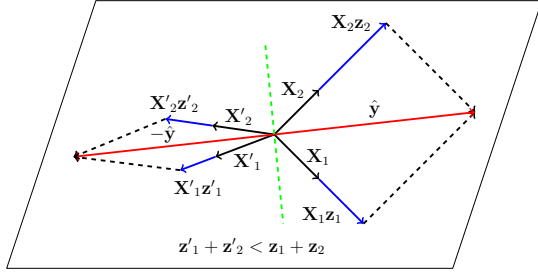
Fig. 2. Geometric interpretation of negative solution of SSC.

representation, a column should not be expressed by itself, thus requiring the constraint $\mathbf{Z}_{ii} = 0$. Accordingly, we have to solve the following optimization problem:

$$\underset{\mathbf{Z}}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{XZ} - \mathbf{X}\|_F^2 + \gamma \|\mathbf{Z}\|_1$$
$$\text{subject to} \quad \text{diag}(\mathbf{Z}) = 0, \mathbf{Z} \geq 0, \quad (8)$$

where $\gamma > 0$ is the regularization hyperparameter.

Many research efforts have been spent in solving the unconstrained version of Eq. (8) [29]. The ℓ_1 minimization problem does not have an analytical solution; its solution instead has to be obtained numerically. Among the proposed algorithms, Augmented Lagrange Multiplier (ALM) generally converges faster under a wide range of data [29]. In this paper, we adopt Alternating Direction Method of Multipliers (ADMM) [30] to solve the problem in Eq. (8), which couples the fast convergence of ALM with the decomposibility property, which is fundamental for distributed implementation in large-scale problems. ADMM introduces a complementary variable $\mathbf{V}$ and re-formulates the unconstrained version of Eq. (8) into the following equivalent form:

$$\underset{\mathbf{Z},\mathbf{V}}{\text{minimize}} \quad \gamma \|\mathbf{V}\|_1 + \frac{1}{2}\|\mathbf{XZ} - \mathbf{X}\|_F^2$$
$$\text{subject to} \quad \mathbf{Z} = \mathbf{V}. \quad (9)$$

Here, decomposibility means that $\mathbf{Z}$ and $\mathbf{V}$ can be updated separately, possibly on a distributed system, thus constraints in Eq. (8) can be imposed on $\mathbf{V}$. They are enforced during $\mathbf{V}$ update by Euclidean projections which are much simpler than ALM. The augmented Lagrangian form of Eq. (9) is

$$\mathcal{L}_\eta(\mathbf{Z}, \mathbf{V}, \mathbf{\Lambda}) = \gamma \|\mathbf{V}\|_1 + \frac{1}{2}\|\mathbf{XZ} - \mathbf{X}\|_F^2 + \langle \mathbf{\Lambda}, \mathbf{Z} - \mathbf{V} \rangle$$
$$+ \frac{\eta}{2}\|\mathbf{Z} - \mathbf{V}\|_F^2.$$

where $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is the Lagrangian multiplier and $\eta > 0$ is the augmented Lagrangian hyperparameter. ADMM iteratively optimizes $\mathbf{Z}, \mathbf{V}$ in an alternate fashion, by keeping one variable fixed and updating the others:

$$\mathbf{Z}^{t+1} = \arg\min_{\mathbf{Z}} \mathcal{L}_\eta \left(\mathbf{Z}, \mathbf{V}^t, \mathbf{\Lambda}^t \right),$$
$$\mathbf{V}^{t+1} = \arg\min_{\mathbf{V}} \mathcal{L}_\eta \left(\mathbf{Z}^{t+1}, \mathbf{V}, \mathbf{\Lambda}^t \right),$$
$$\mathbf{\Lambda}^{t+1} = \mathbf{\Lambda}^t + \eta \left(\mathbf{Z}^{t+1} - \mathbf{V}^{t+1} \right).$$

It is straightforward to demonstrate that $\mathbf{Z}$ can be updated by solving the linear equation:

$$(\mathbf{X}^T\mathbf{X} + \eta\mathbf{I})\mathbf{Z} = (\mathbf{X}^T\mathbf{X} - \mathbf{\Lambda} + \eta\mathbf{V}),$$

using Cholesky decomposition of $\mathbf{X}^T\mathbf{X} + \eta\mathbf{I}$. On the other hand, solution of $\mathbf{V}$ at each iteration is obtained through soft thresholding operator S defined as:

$$S_\nu(a) = \begin{cases} a - \nu & a > \nu \\ a + \nu & a < -\nu \\ 0 & |a| \leq \nu \end{cases}.$$

Details of its update are provided in Appendix A. After $\mathbf{V}$ update, the two following operators are applied to project $\mathbf{V}$ into the feasible set of solutions:

$$\Pi_D(\mathbf{M}_{ij}) = \begin{cases} \mathbf{M}_{ij} & i \neq j \\ 0 & i = j, \end{cases} \quad (11)$$

$$\Pi_N(\mathbf{M}_{ij}) = \begin{cases} \mathbf{M}_{ij} & \mathbf{M}_{ij} \geq 0 \\ 0 & \mathbf{M}_{ij} < 0. \end{cases} \quad (12)$$

The optimization procedure is reported in Algorithm 1: it converges efficiently to an acceptable solution as $\|\mathbf{Z} - \mathbf{V}\|_\infty \to 0$.

Algorithm 1 Constrained LASSO

procedure CONSTRAINED_LASSO($\mathbf{X}, \gamma, \eta$)
    initialize: $\mathbf{Z} \leftarrow 0, \mathbf{V} \leftarrow 0, \mathbf{\Lambda} \leftarrow 0, \varepsilon \leftarrow 10^{-4}$
    while convergence condition is not satisfied **do**
        Fix the others, update $\mathbf{Z}$

$$\mathbf{Z} \leftarrow (\mathbf{X}^T\mathbf{X} + \eta\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X} - \mathbf{\Lambda} + \eta\mathbf{V})$$

        Fix the others, update $\mathbf{V}$

$$\mathbf{V}_{ij} \leftarrow S_{\frac{\gamma}{\eta}} \left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta} \right)$$
$$\mathbf{V}_{ij} \leftarrow \Pi_D(\Pi_N(\mathbf{V}_{ij}))$$

        Fix the others, update $\mathbf{\Lambda}$: $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \eta(\mathbf{Z} - \mathbf{V})$
        Check convergence condition: $\|\mathbf{Z} - \mathbf{V}\|_\infty < \varepsilon$
    end while
    return $\mathbf{Z}$
end procedure

To visually compare sparse representation and normalized correlation matrix, we conduct an analysis on synthetic noise and another one on realistic noise. Synthetic noise is extracted from images generated by the simple imaging model described in [2] for smooth images (without the attenuation factor $\mathbf{T}$), that is $\mathbf{Y} = \mathbf{Y}^{(0)} + \mathbf{Y}^{(0)}\mathbf{K} + \mathbf{\Theta}$. The clean image $\mathbf{Y}^{(0)}$ is uniform, having pixel value of 0.9 (relatively bright). $\mathbf{K}_{ij}$ and $\mathbf{\Theta}_{ij}$ are reasonably assumed as white Gaussian noise. As the signal $\mathbf{K}$ is generally weaker than $\mathbf{\Theta}$, the variance of $\mathbf{K}_{ij}$ is selected as 0.001 and the variance of $\mathbf{\Theta}_{ij}$ is 0.1 (for pixel values in $[0, 1]$). We simulate the situation of 5 cameras corresponding to 5 different $\mathbf{K}$ patterns, considering 100 images for each camera, thus resulting into 500 different $\mathbf{\Theta}$ patterns. After that, we apply the same wavelet-based denoising filter to extract synthetic noise. A sample of extracted synthetic noise is depicted in Figure 3 (a). For the realistic setting, we select 5 cameras from the Vision dataset [31], 100 images for each camera, and apply the same denoising procedure. In Figure 3 (b) an example of realistic noise is shown. We intentionally group noise residuals of the same
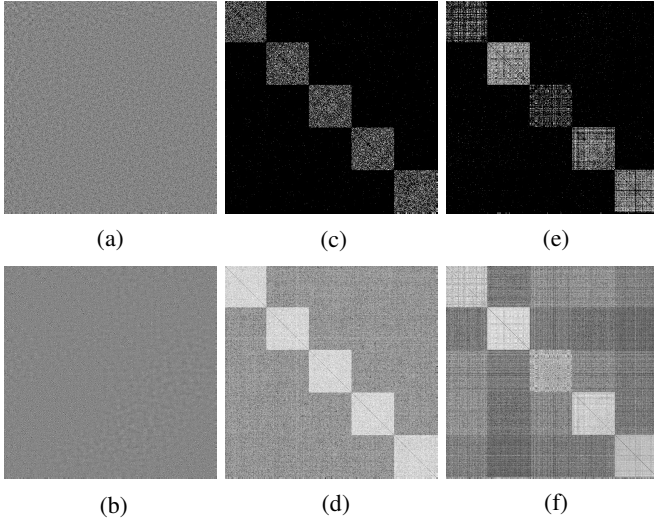
Fig. 3. Visual comparison of sparse representation and dense representation (obtained by normalized correlation): (a) synthetic noise sample, (b) realistic noise sample, (c) sparse representation matrix of synthetic noise, (d) dense representation matrix of synthetic noise, (e) sparse representation matrix of realistic noise, (f) dense representation matrix of realistic noise.

Fig. 4. Schema of the proposed LS-SSC.

camera so that the representation matrix is easily observable. We show sparse representation matrix of synthetic noise in Figure 3 (c), and of realistic noise in Figure 3 (e). The dense representation matrix in Figure 3 (d) and 3 (f) are obtained by computing pair-wise normalized correlation for synthetic noise and for realistic noise, respectively. Noticeably, solving the problem in Eq. (8) obtains meaningful representation where inter-class relations are effectively removed, revealing clearer block-diagonal structure compared to normalized correlation.

The sparse representation matrix captures asymmetric relationships among data points, i.e., $\mathbf{Z}_{ij} \neq \mathbf{Z}_{ji}$. For our clustering purpose, we build a weighted undirected graph $\mathbf{G}$ from $\mathbf{Z}$ as $\mathbf{G} = (\mathbf{Z} + \mathbf{Z}^T)/2$. To obtain the final segmentation, we apply the spectral clustering described in [32] to partition $\mathbf{G}$ into κ connected components or clusters. In practice, κ can be inferred from the number of small eigenvalues. Therefore, we adopt an approach based on *eigengap heuristic* [33] to infer the number of clusters, similarly as [16].

In summary, the proposed algorithm learns sparse representations of each camera fingerprint. To avoid any confusion with ordinary SSC in [16], we denote Sparse Subspace Clustering with Non-negativity Constraint as SSC-NC. This approach provides a good instrument to discover structures on high-dimensional data, but shows a major drawback on scalability, since all data must be loaded in RAM. In Section IV the computational complexity of Algorithm 1 will be demonstrated to be in the order of n^3, n being the number of fingerprints. Empirically, this is acceptable only for datasets with $n \leq 6000$.

C. Large-scale sparse subspace clustering

In this section we extend our methodology to cluster camera fingerprints in large-scale contexts, referring to such extension as large-scale SSC (LS-SSC). First, we address the memory issue using a *divide-and-conquer* strategy, so that compact
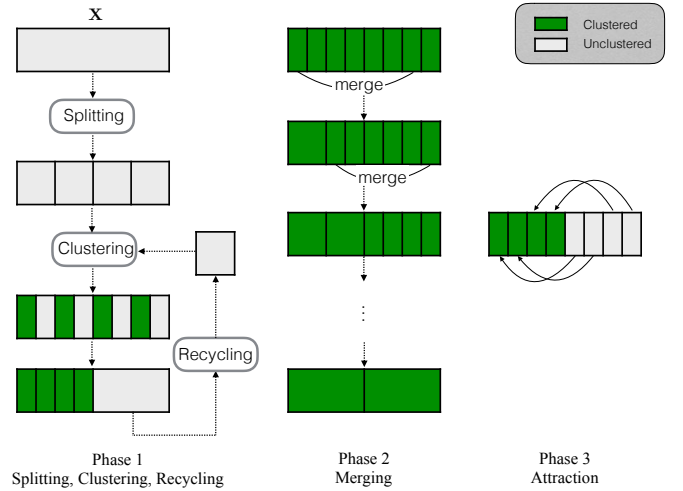
clusters could be discovered on small data batches. This process is followed by data re-cycling to increase the chance of discovering hidden clusters. Finally, we employ *merging* and *attraction* phases to finalize clustering results.

1) Splitting, Clustering and Recycling: The baseline strategy of our large-scale clustering is the divide-and-conquer paradigm, which breaks an intractable problem into several smaller tractable problems. We randomly split the set of all fingerprints $\mathcal{X}$ into B batches of equal size, $\mathcal{X} = \left\{ \mathcal{X}^l \right\}_{l=1,...,B}$, where B is originally set to $\lceil \frac{n}{p} \rceil$ and p is the batch size. Only one data batch at a time is loaded on RAM. We then apply Algorithm 1 on the data batch to learn sparse representations among fingerprints.

We hereby refer to *cluster purity* as the quality of a cluster. A pure cluster should contain only fingerprints of the same camera. The main purpose of this phase is to extract small-size but pure clusters that can be later merged to form larger clusters. As a result of splitting, a fingerprint might not be well reconstructed by only fingerprints from the same camera. To minimize the reconstruction error, the algorithm might select fingerprints from multiple cameras. Such representations are considered as *outliers*, similarly to [34].

Let us now consider a sparse representation matrix as a directed graph: outliers have connections to both outliers and inliers, while inliers have connections to inliers only. If we perform a random walk on the graph, the probability of ending at inliers is therefore higher than ending at outliers. We apply the random walk algorithm described in [34] with 1000 steps to acquire the state probabilities, we model such probabilities as a normal distribution, and we keep 80% of the distribution as inliers, thus classifying the rest as outliers.

To guarantee the purity of clusters, we avoid spectral clustering, but we attempt to localize dense regions using the interpretability property of our sparse representation. Accordingly, large values indicate *closest* fingerprints. Since our target is to discover small-size but pure clusters, we can further simplify the graph by retaining only K largest entries on each column of the sparse representation matrix, and setting other entries to

zero. Each remaining fingerprint is located in a region of K nearest neighbors, and fingerprints in the same cluster should have common neighbors, forming a dense region.

After that, we apply DBSCAN [35] to discover dense regions. This classical clustering technique is computationally feasible for large-scale datasets and does not require the number of clusters to be known. Two parameters need to be indicated as input of DBSCAN: radius ϵ and minimum number of neighbors $MinPts$. The radius ϵ should be selected on the basis of K largest values on each column of the sparse representation matrix, while $MinPts$ must be smaller or equal to K. If ϵ is too small, this results in many clusters. Conversely, very limited number of clusters are discovered, complicating the recycling process. On the other hand, if $MinPts > K$, there is no cluster discovered by DBSCAN. We empirically found that setting $MinPts = K$ and ϵ equal to the mean of non-zero entries, allows discovering pure clusters.

After clustering, we obtain the set of inliers and outliers, where inliers are used for the merging phase and outliers are fed to the recycling process. The aim of recycling is to combine outliers from each batch and feed them back to the clustering process, thus increasing the chance to discover hidden clusters. For that reason, clustering and recycling can be seen as an iterative procedure, as outlined in Algorithm 2.

Algorithm 2 Splitting, clustering, recycling

procedure SPLITTING_CLUSTERING_RECYCLING
    input: $\mathcal{X}, p, R, K, \gamma, \eta$ ▷ $\mathcal{X}$: dataset, p: batch size, R: number of recycling steps, K: number of nearest neighbors
    output: $\mathcal{X}_{\text{in}}, \mathcal{X}_{\text{out}}$ ▷ set of clustered and unclustered fingerprints
    $\mathcal{X}_{\text{in}} \leftarrow \emptyset$
    $B \leftarrow \lceil \frac{n}{p} \rceil$
    Split $\mathcal{X}$ into $\left\{ \mathcal{X}^l \right\}_{l=1,\ldots,B}$
    $\mathcal{X}_{\text{out}}^l \leftarrow \emptyset$,   ▷ $l = 1, \ldots, B$
    for $l = 1 \rightarrow B$ **do**
        $\tilde{\mathcal{X}}_{\text{in}}^l, \tilde{\mathcal{X}}_{\text{out}}^l \leftarrow$ PARTITION$(\mathcal{X}^l, K, \gamma, \eta)$
        Append $\tilde{\mathcal{X}}_{\text{in}}^l$ to $\mathcal{X}_{\text{in}}$ and append $\tilde{\mathcal{X}}_{\text{out}}^l$ to $\mathcal{X}_{\text{out}}^l$
    end for
    $t \leftarrow B, \tilde{B} \leftarrow B$
    repeat
        $\mathcal{X}_{\text{out}}^t \leftarrow \emptyset, \mathcal{X}^t \leftarrow \emptyset$
        for $l = 1 \rightarrow \tilde{B}$ **do**
            Pop out randomly $s^l = \frac{|\mathcal{X}_{\text{out}}^l| \times p}{\sum_{i=1}^{\tilde{B}} |\mathcal{X}_{\text{out}}^i|}$ fingerprints from $\mathcal{X}_{\text{out}}^l$
            Append s^l fingerprints to $\mathcal{X}^t$
        end for
        $\tilde{\mathcal{X}}_{\text{in}}^t, \tilde{\mathcal{X}}_{\text{out}}^t \leftarrow$ PARTITION$(\mathcal{X}^t, K, \gamma, \eta)$
        Append $\tilde{\mathcal{X}}_{\text{in}}^t$ to $\mathcal{X}_{\text{in}}$ and append $\tilde{\mathcal{X}}_{\text{out}}^t$ to $\mathcal{X}_{\text{out}}^t$
        $t \leftarrow t + 1, \tilde{B} \leftarrow \tilde{B} + 1$
    until $t \geq B + R$
    $\mathcal{X}_{\text{out}} \leftarrow \left\{ \mathcal{X}_{\text{out}}^l \right\}_{l=1,\ldots,B+R}$
end procedure
procedure PARTITION
    input: $\mathcal{X}, K, \gamma, \eta$ ▷ $\mathcal{X}$: dataset, K: number of nearest neighbors
    output: $\mathcal{X}_{\text{in}}, \mathcal{X}_{\text{out}}$ ▷ set of clustered and unclustered fingerprints
    Load fingerprints in $\mathcal{X}$ to $\mathbf{X}$ ▷ $\mathbf{X}$: matrix of fingerprints
    $\mathbf{Z} \leftarrow$ CONSTRAINED_LASSO$(\mathbf{X}, \gamma, \eta)$
    Remove outliers, obtain $\tilde{\mathbf{Z}}$. Append outliers to $\mathcal{X}_{\text{out}}$
    Keep only K largest entries on each column of $\tilde{\mathbf{Z}}$, obtain $\tilde{\mathbf{Z}}_{\text{KNN}}$
    Apply DBSCAN to discover clusters
    Append inliers to $\mathcal{X}_{\text{in}}$
    Append outliers to $\mathcal{X}_{\text{out}}$
end procedure

---

2) Merging: In the first phase, by increasing K we obtain larger clusters at the expense of a lower cluster purity. Conversely, we yield small-size pure clusters. The latter is preferable, as small-size clusters (subclusters) can be merged efficiently to form larger subclusters.

Let $\mathbf{W}^A$ and $\mathbf{W}^B$ be two noisy fingerprints of dimension d, i.e., two singleton subclusters, reasonably assumed to follow a normal distribution since the denoising filter extracts stationary Gaussian noise in wavelet domain (see Appendix A of [1]). $\mathbf{K}^A$ and $\mathbf{K}^B$ are the noise-free fingerprints residing in $\mathbf{W}^A$, $\mathbf{W}^B$. The merging problem can be formulated as a classical hypothesis test:

$$\mathrm{H}_0 : \mathbf{K}^A \neq \mathbf{K}^B,$$
$$\mathrm{H}_1 : \mathbf{K}^A = \mathbf{K}^B = \mathbf{K}.$$

Under null hypothesis, $\rho\left(\mathbf{W}^A, \mathbf{W}^B\right) \sim \mathcal{N}(0, \frac{1}{d})$ according to the Central Limit Theorem (CLT). Two subclusters can be merged if their normalized correlation exceeds $\frac{1}{\sqrt{d}} Q^{-1}(P_{FA})$, where $Q(t)$ is the probability that a standard normal variable is larger than t and P_{FA} is the expected false alarm rate [23]. More generally, if each cluster contains more than one fingerprint, $\mathbf{W}^A$ and $\mathbf{W}^B$ represent respectively the subcluster centroids. Under alternative hypothesis, the correlation between $\mathbf{W}^A$ and $\mathbf{W}^B$ increases if the cardinality of each subcluster increases, as random noise is effectively suppressed by averaging. Obviously, the merging phase will be more reliable if one knows not only the null distribution but also the alternative distribution.

To determine the alternative distribution, [36], [15] established a parametric model with some statistical assumptions and determined model parameters. For instance, the true fingerprints are assumed to be additive noise [36], [15], and the WGN of a camera presents always the same variance [36]. Nevertheless, if those assumptions are not guaranteed, and usually they are not, parameter estimation becomes extremely difficult.

We resort the merging problem into finding a threshold value τ that is able to exclude the null hypothesis and to adapt to the variation of the alternative hypothesis, based on real data. This is achieved by taking into account the cardinality and intra-class correlation within each subcluster. Let $\mathbf{X}^A$ and $\mathbf{X}^B$ be the two matrices containing n_A and n_B fingerprints of each subcluster, and ρ_A and ρ_B be the intra-class correlation within these subclusters. We learn the threshold adaptiveness via linear regression:

$$\mathcal{R}\left(n_A, n_B, \rho_A, \rho_B\right) = [n_A, n_B, \rho_A, \rho_B]\,\mathbf{w} + b,$$

where $\mathbf{w} \in \mathbb{R}^{4 \times 1}$ and $b \in \mathbb{R}$ are weights and bias, respectively. From real data, we calculate ρ_A, ρ_B and the regression output $\mathcal{R}(\cdot)$ as follows:

$$\rho_A = \frac{1}{n_A(n_A - 1)} \sum_{i=1}^{n_A} \sum_{j=1, j\neq i}^{n_A} \rho(\mathbf{X}_i^A, \mathbf{X}_j^A),$$

$$\rho_B = \frac{1}{n_B(n_B - 1)} \sum_{i=1}^{n_B} \sum_{j=1, j\neq i}^{n_B} \rho(\mathbf{X}_i^B, \mathbf{X}_j^B),$$

$$\mathcal{R}(\cdot) = \frac{\rho(\bar{\mathbf{X}}^A, \bar{\mathbf{X}}^B)}{2},$$

where $\bar{\mathbf{X}}^A, \bar{\mathbf{X}}^B$ are respectively two subcluster centroids, i.e., mean of columns in $\mathbf{X}^A$ and $\mathbf{X}^B$. The estimate of $\mathcal{R}(\cdot)$ is
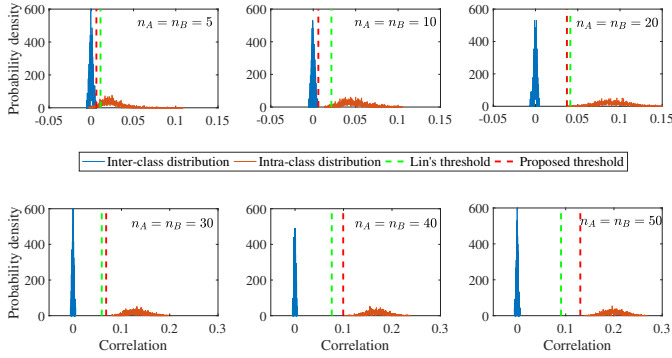
Fig. 5. Comparison of the proposed threshold and Lin's threshold [15] under two cameras: Kodak M1063 and Nikon CoolPix S710. Better viewed in color.

interpreted as the central value between mean of null and alternative distribution. The final regressor learnt from real data (we will mention this development set in Section V-B) has the form:

$$\mathcal{R}\left(n_A, n_B, \rho_A, \rho_B\right) = 0.0016\, n_A + 0.0016\, n_B +$$
$$2.2474\, \rho_A + 2.2474\, \rho_B - 0.0474.$$

Careful readers will notice that the regressor is symmetric in terms of cluster role, i.e., $\mathcal{R}\left(n_A, n_B, \rho_A, \rho_B\right) = \mathcal{R}\left(n_B, n_A, \rho_B, \rho_A\right)$. This is achieved by augmenting the training data with the role of two clusters exchanged. The threshold τ is finally calculated as:

$$\tau = \max\left\{\frac{1}{\sqrt{d}} Q^{-1}(P_{FA}), \mathcal{R}\left(\cdot\right)\right\},$$

where P_{FA} is chosen as 0.001 (0.1% false alarm rate).

The merging phase is conducted as an iterative procedure, which respectively selects pairs of subclusters having maximum centroid correlation and compares them to τ. If the correlation is larger than τ, the two subclusters are merged and relevant information is updated. The algorithm stops when no more pairs of subclusters exist that satisfy the merging condition.

In practical cases, a good regressor might not be linear, but for n_A, n_B within a reasonably small range, τ can be fitted by a linear function. Therefore, to calculate a reliable τ, we set $n_A = \min\{\tilde{n}_A, 50\}$ and $n_B = \min\{\tilde{n}_B, 50\}$ where $\tilde{n}_A, \tilde{n}_B$ are actual cluster cardinalities, and calculate ρ_A, ρ_B using bounded sets of fingerprints. The quantity 50 is suggested as a minimal cardinality for fingerprint estimation [37], [1].

To demonstrate the effectiveness of the proposed threshold, we compare with Lin's threshold [15], which was previously shown to be superior than thresholds in [5] and [38]. We select two cameras, namely Kodak M1063 and Nikon CoolPix S710 from Dresden database [39]. We randomly split images of one camera into two parts to simulate two same-camera subclusters. Images from different cameras are used to create cross-camera subclusters. The process is replicated 2000 times and intra-class and inter-class correlations are collected. Figure 5 shows how the proposed threshold and the threshold in [15] (Lin's threshold) separate null and alternative distribution. When the cardinality of same-camera subclusters increases,

the alternative distribution shifts towards the right, while the null distribution is centered at 0. The proposed threshold consistently splits the two distributions, while Lin's threshold tends to be unnecessarily confident when two distributions are close. An interesting behavior of the proposed threshold and Lin's threshold is their adaptiveness to distribution shifting.

3) Attraction: In attraction phase, we assign remaining fingerprints to available clusters. Let us denote $\mathbf{C} = \left[\bar{\mathbf{C}}_1, \bar{\mathbf{C}}_2, \ldots, \bar{\mathbf{C}}_L\right] \in \mathbb{R}^{d \times L}$ the matrix containing centroids of L final clusters, and $\mathbf{X}^{\text{out}} \in \mathbb{R}^{d \times U}$ the data matrix containing U unclustered fingerprints. Since the quality of camera fingerprints is generally non-homogeneous, cluster assignment should be performed for high-quality fingerprints first, in order to minimize assignment errors. The cluster membership $l, 1 \leq l \leq L$ of fingerprint $\mathbf{X}_i^{\text{out}}$, $1 \leq i \leq U$ is obtained iteratively by finding at each step the pair l and i such that $\rho(\mathbf{X}_i^{\text{out}}, \bar{\mathbf{C}}_l)$ is maximum and greater than $Q^{-1}(P_{FA})$ which is the threshold used to exclude null hypothesis in merging phase. After being attracted $\mathbf{X}_i^{\text{out}}$ is discarded, otherwise it is labeled to as *unclustered*. Since the remaining fingerprints to be merged have been classified as outliers after recycling, we can expect that they are low-quality samples. Therefore, to reduce false alarm rate, the cluster centroid is updated only when its cardinality does not exceed 50, consistently with the empirical value used in the merging phase.

Eventually, we obtain the cluster memberships of camera fingerprints in a large-scale database and a number of unclustered fingerprints.

IV. COMPUTATIONAL COMPLEXITY

In this section we discuss on the time complexity of our proposed SSC-NC, LS-SSC and two recent works: correlation clustering with consensus (CCC) [14] and Lin's large-scale method (Lin-LS) [15].

SSC-NC. SSC-NC is composed by CONSTRAINED_LASSO and spectral clustering. CONSTRAINED_LASSO consists of Cholesky decomposition, linear equation solving and soft thresholding. In the worst case, Cholesky decomposition requires $n^3/3$ flops. Solving linear equations requires $2n^2$ flops of forward and backward substitutions. Soft-thresholding operation on n^2 variables requires n^2 computations. Let T_1 be the bound number of iterations, total cost of CONSTRAINED_LASSO is $\mathcal{O}\left(n^3/3 + 3T_1 n^2\right)$. Spectral clustering consists of maximum $\mathcal{O}\left(n^3\right)$ computations for eigendecomposition and $\mathcal{O}\left(T_2 \kappa^2 n\right)$ for K-means clustering on n κ-dimensional eigenvectors, where T_2 is the bound number of iterations in K-means and κ is the number of clusters. The time complexity of SSC-NC is $\mathcal{O}\left(4n^3/3 + 3T_1 n^2 + T_2 \kappa^2 n\right)$.

CCC. Similarly to typical clustering methods, CCC computes the correlation matrix which costs $\mathcal{O}\left(n^2\right)$. Correlation clusterings are afterwards carried out by Adaptive Label Iterated Contitional Modes (AL-ICM) [40]. AL-ICM, a greedy algorithm, operates in iterative mode. Every fingerprint is initially assigned to a unique label. At each iteration, AL-ICM assigns to a fingerprint the label of its closest fingerprints. This process is repeated until convergence where no label is updated. If T_3 is the bound number of iterations,

the time complexity of AL-ICM is bounded to $\mathcal{O}\left(T_3 n^2\right)$. In CCC, correlation clustering is performed Q times where Q is the number of similarity thresholding values. Multiple base clusterings are combined to find the final clustering agreement by Weighted Evidence Accumulation Clustering (WEAC) [41]. The time complexity of WEAC is $\mathcal{O}\left((Q + \log n)n^2 + Qn\right)$. Finally, m obtained clusters are refined via a merging step which costs $\mathcal{O}\left(m^2 \log m\right)$. Total cost of CCC is $\mathcal{O}\left((QT_3 + Q + \log n + 1)n^2 + Qn + m^2 \log m\right)$.

LS-SSC. In large-scale contexts, we suppose that RAM can cache only p fingerprints. The dataset is split into B batches, $B = \lceil \frac{n}{p} \rceil$. Clustering each batch requires running CONSTRAINED_LASSO, finding K nearest neighbors and DB-SCAN. Finding K nearest neighbors requires sorting each column of sparse representation matrix, which is $\mathcal{O}\left(p^2 \log p\right)$. In the worst case, DBSCAN visits p points and scans for their neighbors, which costs $\mathcal{O}\left(p^2\right)$. Total cost of clustering B batches is $\mathcal{O}\left(B\left[p^3/3 + (3T_1 + \log p + 1)p^2\right]\right)$. In our large-scale experiments, recycling step is replicated $B/2$ times on batches of size p. Merging and attracting phase work similarly to agglomerative hierarchical clustering, and their time complexity is respectively $\mathcal{O}\left(L^2 \log L\right)$ and $\mathcal{O}\left(UL \log U\right)$, where L is the number of discovered clusters after the first phase and U is the number of unclustered images. Total cost of LS-SSC is $\mathcal{O}\left(1.5B^2\left[p^3/3 + (3T_1 + \log p + 1)p^2\right] + L^2 \log L + UL \log U\right)$.

Lin-LS. The time complexity of Lin-LS is analyzed for the first iteration. In the coarse step, the correlation calculation of B batches requires $\mathcal{O}\left(Bp^2\right)$. If the correlation matrix $n \times n$ has E non-zero entries, Graclus partitioning algorithm [42] has the time complexity of $\mathcal{O}\left(pE/n\right)$. Since the number of clusters in coarse step is fixed to $n^{1/4}$, the calculation of correlation matrix in the fining step costs $\mathcal{O}\left(n^{1/4}b^2\right)$ where b is the average size of clusters. Markov Clustering Algorithm (MCL) applied on $n^{1/4}$ coarse clusters is bounded to $\mathcal{O}\left(n^{1/4}bK^2\right)$, where K, for abuse of notation, is the maximal number of nonzero entries on each column of the binarized correlation matrix. Similarily to LS-SSC, merging and attraction of Lin-LS can be approximated to $\mathcal{O}\left(L^2 \log L\right)$ and $\mathcal{O}\left(UL \log U\right)$ where L is the discovered number of clusters and U refers to the number of unclustered fingerprints. Since both LS-SSC and Lin-LS aim to obtain high-quality clusters of small size, we can equalize U, L in LS-SSC and U, L in Lin-LS for easy comparison. The first iteration of Lin-LS totally costs $\mathcal{O}\left(Bp^2 + (K^2b + b^2)n^{1/4} + pE/n + L^2 \log L + UL \log U\right)$. The two parameters E and K depend on the cluster distribution in the dataset.

In medium-size datasets where no divide-and-conquer is needed, i.e., $p = n$, SSC-NC and LS-SSC are cubic while Lin-LS and CCC are approximately quadratic. In large-scale datasets, only algorithms designed with divide-and-conquer strategy can be run under the constraint on RAM as well as computational power. The time complexity of LS-SSC is cubic with respect to p, while Lin-LS is almost quadratic with respect to p and cluster distribution. In fact, when n becomes very large, we can fix p in LS-SSC as an upper bound, while Lin-LS requires to synthesize the correlation matrix $n \times n$ in the coarse step. Moreover, the time complexity of Lin-LS is analyzed only on the first iteration, the cost of following iterations have to be accounted. Although LS-SSC is cubic with respect to p due to Cholesky decomposition, we optimize this computation by exploiting LAPACK [43] whose implementation of Cholesky decomposition is extremely efficient. Our implementation will be made available upon paper acceptance.

V. EXPERIMENTS

In this section, we provide experimental analyses of the proposed clustering framework. Based on real data, hyperparameters are selected and used thorough all experiments. We validate the superiority of our method under intensive settings, both on medium and large-scale clustering contexts.

A. Experimental Settings

Dataset. All experiments are conducted on JPEG images of Dresden [39] and Vision [31]. The top-left regions of size 512×512 are cropped out for fingerprint extraction. We have tested diverse configurations whose quatitative details are outlined in Table I and Table II, considering:

- *Cluster symmetry*. On Dresden and Vision, we create symmetric datasets containing 100 images for each camera, and asymmetric datasets containing all available images on each camera. We denote such configuration on Dresden as $\mathcal{D}_c^a$ $\mathcal{D}_c^s$, and on Vision as $\mathcal{V}_c^a$ $\mathcal{V}_c^s$, where a and s stand for *symmetric* and *asymmetric*, respectively, and c is the number of cameras.
- *Multiple instances of the same model*. On Dresden, we create datasets containing 5 camera instances of each camera model. Combining with cluster symmetry, we obtain symmetric and asymmetric datasets of this configuration as $\mathcal{D}_c^{sm}$ and $\mathcal{D}_c^{am}$.
- *Number of cameras*. In medium-size datasets, we first select $c = 5$, and incrementally add 5 cameras till $c = 20$.
- *Large-scale clustering*. On Dresden, we first select $c = 30$, and incrementally add 5 cameras till the maximum $c = 74$, considering all cameras. Since Vision is smaller than Dresden, we start with $c = 21$ and incrementally add 3 cameras till $c = 33$. Such configurations on Dresden and Vision are respectively denoted as $\mathcal{LD}_c^a$ and $\mathcal{LV}_c^a$. All these configurations include cameras of same models.

Performance metric. We report performance in $\mathcal{F}$-measure and Adjusted Rand Index (ARI). In the presence of outliers (unclustered fingerprints), we follow [15] and treat outliers differently in the computation of True Positive ($\overline{TP}$) and False Positive ($\overline{TP}$). Specifically,

- True Positive ($\overline{TP}$): the number of image pairs from the same cluster which are assigned to the same cluster, *excluding outliers*.
- False Positive ($\overline{FP}$): the number of image pairs from different clusters which are assigned to the same cluster, *excluding outliers*.
- True Negative (TN): number of image pairs from different clusters which are assigned to different clusters.

TABLE I
TESTING CONFIGURATIONS ON MEDIUM-SIZE DATASETS.

Configuration		# cameras		# models		# images	
Dresden	Vision	Dresden	Vision	Dresden	Vision	Dresden	Vision
$\mathcal{D}_5^s$	$\mathcal{V}_5^s$	5		5		500	
$\mathcal{D}_{10}^s$	$\mathcal{V}_{10}^s$	10		10		1000	
$\mathcal{D}_{15}^s$	$\mathcal{V}_{15}^s$	15		15		1500	
$\mathcal{D}_{20}^s$	$\mathcal{V}_{20}^s$	20		20		2000	
$\mathcal{D}_5^a$	$\mathcal{V}_5^a$	5		5		1089	1041
$\mathcal{D}_{10}^a$	$\mathcal{V}_{10}^a$	10		10		1954	2110
$\mathcal{D}_{15}^a$	$\mathcal{V}_{15}^a$	15		15		3031	3208
$\mathcal{D}_{20}^a$	$\mathcal{V}_{20}^a$	20		20		4186	4435
$\mathcal{D}_5^{sm}$	–	5	–	1	–	500	–
$\mathcal{D}_{10}^{sm}$	–	10	–	2	–	1000	–
$\mathcal{D}_{15}^{sm}$	–	15	–	3	–	1500	–
$\mathcal{D}_{20}^{sm}$	–	20	–	4	–	2000	–
$\mathcal{D}_5^{am}$	–	5	–	1	–	855	–
$\mathcal{D}_{10}^{am}$	–	10	–	2	–	2663	–
$\mathcal{D}_{15}^{am}$	–	15	–	3	–	3558	–
$\mathcal{D}_{20}^{am}$	–	20	–	4	–	4540	–

TABLE II
TESTING CONFIGURATIONS ON LARGE-SCALE DATASETS.

Configuration		# cameras		# images	
Dresden	Vision	Dresden	Vision	Dresden	Vision
$\mathcal{LD}_{30}^a$	$\mathcal{LV}_{21}^a$	30	21	6596	4397
$\mathcal{LD}_{35}^a$	$\mathcal{LV}_{24}^a$	35	24	7538	5051
$\mathcal{LD}_{40}^a$	$\mathcal{LV}_{27}^a$	40	27	8545	5773
$\mathcal{LD}_{45}^a$	$\mathcal{LV}_{30}^a$	45	30	9635	6377
$\mathcal{LD}_{50}^a$	$\mathcal{LV}_{33}^a$	50	33	10765	7070
$\mathcal{LD}_{55}^a$	–	55	–	11673	–
$\mathcal{LD}_{60}^a$	–	60	–	12729	–
$\mathcal{LD}_{65}^a$	–	65	–	13995	–
$\mathcal{LD}_{70}^a$	–	70	–	14915	–
$\mathcal{LD}_{74}^a$	–	74	–	15677	–

- False Negative (FN): number of image pairs from the same cluster which are assigned to different clusters.

$\mathcal{F}$-measure is computed based on precision ($\mathcal{P}$) and recall ($\mathcal{R}$):

$$\mathcal{P} = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}, \quad \mathcal{R} = \frac{\overline{TP}}{\overline{TP} + FN}, \quad \mathcal{F} = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{C}}{\mathcal{P} + \mathcal{C}}.$$

Rand Index (RI) and ARI are computed as:

$$\text{RI} = \frac{\overline{TP} + TN}{\overline{TP} + TN + \overline{FP} + FN}, \quad \text{ARI} = \frac{\text{RI} - \mathbf{E}[\text{RI}]}{1 - \mathbf{E}[\text{RI}]},$$

where $\mathbf{E}[\text{RI}]$ is the expected value of RI and is computed based on the expected value of $\overline{TP}$ and TN.

$$\mathbf{E}[\text{RI}] = \frac{\mathbf{E}[\overline{TP}] + \mathbf{E}[TN]}{\overline{TP} + TN + \overline{FP} + FN}.$$

The readers can refer to [44] for more details of ARI computation. When the number of outliers is zero, $\mathcal{F}$-measure and ARI become canonically defined.

For comparing the number of clusters discovered by each algorithm, we follow [15] to report the ratio L_p/L_g where L_p refers to the number of predicted clusters and L_g the number of ground-truth clusters. Differently to [15] where L_p only accounts for *unique* predicted clusters, i.e., $L_p \leq L_g$, it is possible in our evaluation that $L_p/L_g > 1$ if an algorithm overestimates, or $L_p/L_g < 1$ if under-estimating the number of ground-truth clusters.

Performance comparison. We compare the results of the proposed methodologies with the state of the art. Tests have been done also with hierarchical clustering [9], Markov Random Field [6], and Spectral Clustering with Normalized Cut criterion [12], but for the sake of space and readability we only present comparisons with the following top performing works:

- *Multiclass Spectral Clustering (MSC)* [11]. A star graph is built with 5 nearest neighbors, as suggested in [11].
- *Lin's Large-Scale (Lin-LS) method* [15]. Lin-LS is implemented with all parameters recommended from [15]: compressed fingerprints (256×256) are binarized by threshold $t_b = 0.008$, while original-size fingerprints (1024×1024) are binarized by threshold $t_b = 0.005$. In order to take divide-and-conquer strategy into effect on medium-size datasets, each dataset is split into two equal batches and only one is loaded at once.
- *Correlation Clustering with Consensus (CCC)* [14]. Results of CCC are acquired from the implementation provided by the authors. No parameter needs to be specified.
- *Sparse Subspace Clustering (SSC)* [16]. SSC is implemented similarly to SSC-NC but without the non-negativity constraint.

We analyze the performance of SSC-NC to see the effectiveness of non-negativity constraint, and LS-SSC to verify its adaptation on medium-size and large-scale datasets. To simulate divide-and-conquer on medium-size datasets, LS-SSC splits each dataset into two equal batches and only one is loaded at once in the same manner as Lin-LS.

Under large-scale datasets, LS-SSC is compared only to Lin-LS since these methods are particularly designed for large-scale contexts. One matter of clustering on large-scale datasets is the lack of memory. Since only a limited number of fingerprints can be allocated on RAM, we fix this bound to 4000 (≈ 4 GBs are required to store fingerprints).

Due to some randomization used in MSC, CCC, Lin-LS and LS-SSC, those methods are run 10 times, and the average scores are reported.

B. Hyperparameter Selection

In order to select a number of parameters required by our methodologies we collect a dataset, obviously different from the test one. From RAISE dataset [45] we extract 200 raw images from Nikon D90 and 250 from D7000, and perform JPEG compression (quality factor 98). Since there are only 76 raw images of Nikon D40, we leave them out and instead select 300 JPEG images (default JPEG quality setting) from an external Canon 600D. We refer to this dataset as $\mathcal{D}_{\text{dev}}$ including 750 images from 3 cameras.

Selecting η. η is the augmented Lagrangian hyperparameter which stands for how much penalty added in order to enforce the equality $\mathbf{Z} = \mathbf{V}$. This parameter partially decides the convergence speed of CONSTRAINED_LASSO. Small η means slow convergence but with high accurate solutions, while large η accelerates convergence speed but results in modest accurate solutions. Since sparse representation learning is followed by a clustering procedure, solutions with modest accuracy are sufficient. On $\mathcal{D}_{\text{dev}}$, $\eta \in [1.0, 1.3]$ results in acceptable solutions and fast convergence. We adopt $\eta = 1.0$ in all experiments.
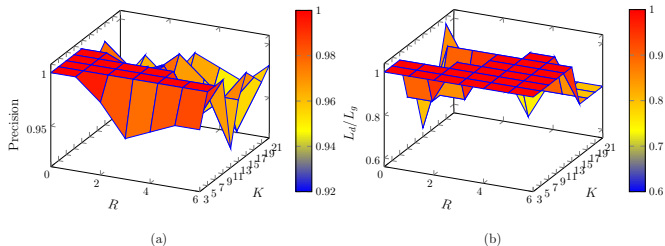
Fig. 6. Precision and L_d/L_g with respect to diverse values of K and # recycling steps.

Selecting γ. On $\mathcal{D}_{\text{dev}}$, we vary γ in the range $[0.0001, 0.02]$ and select $\gamma = 0.0018$ that minimizes the cost function defined in [16] taken into account normalized cuts and eigengaps as criterions.

Jointly selecting R **and** K. In LS-SSC, the main goal of recycling is to reduce the number of undiscovered ground-truth clusters. Let us denote as L_d, L_g the number of ground-truth clusters discovered after merging phase and the number of ground-truth clusters, respectively. The strategy is to adopt the number of recycling steps R such that $L_d/L_g \to 1$ and discovered clusters are pure, namely Precision $\to 1$. Another parameter which impacts on L_d is the number of nearest neighbors K. Small K means more ground-truth clusters are likely to be discovered, otherwise only noticeably dense clusters are discovered. We conduct experiments on an asymmetric dataset from Dresden containing 5 cameras coming from different models. We split the dataset into 6 equal batches of size ≈ 182 in order to simulate splitting step. Figure 6 depicts precision of discovered clusters after merging step in panel (a), and the ratio L_d/L_g in panel (b). It is clear that $K = 5$ is a reasonable choice for discovering pure ground-truth clusters. From these plots one can argue that selecting $R = 0$ allows to obtain the highest precision in this case. However, it is important to remember that recycling plays an important role since it helps discover more hidden clusters. In principle, high value of R should be chosen considering the computational complexity, but the precision is likely to drop if we run more recycling steps with big K. In large-scale contexts, we adopt $R = \lfloor B/2 \rfloor$, where B is the number of batches. In medium-scale contexts, where computational requirement is less important, we run recycling until there is no noticeable subclusters discovered.

C. Numeric Results on Medium-size Datasets

We report performance of all methods on medium-size datasets with the maximum number of images ranging from 4000 to 5000.

Results on Dresden suggest that MSC performs relatively well on symmetric (in Figure 7 (a)) and asymmetric (in Figure 7 (b)) datasets. MSC applies an extra step before clustering. It is the creation of a star graph among fingerprints, where noisy connections are partially eliminated. The star graph can be considered as a suboptimal sparse representation matrix of data. Differently to MSC, SSC finds a sparse representation of data by solving an optimization problem. In Figure 7,

SSC outperforms MSC in most configurations with high $\mathcal{F}$-measure. As an improved version of SSC, SSC-NC performs equally or better than SSC in the majority of symmetric and asymmetric datasets. Balanced precision and recall are obtained, gaining high $\mathcal{F}$-measure. The number of predicted clusters L_p obtained by SSC and SSC-NC are identical, approximating well the number of ground-truth clusters L_g. Such approximation is the best among all tested algorithms.

Although Lin-LS and LS-SSC are especially designed for large-scale datasets, they produce convincing results also on medium-size datasets. Lin-LS aims to obtain high-quality clusters of small size, resulting in high precision. Comparing to Lin-LS, LS-SSC obtains less precise clusters but the precision is still high without penalizing recall. Thanks to this balanced behavior, LS-SSC outperforms Lin-LS in terms of $\mathcal{F}$-measure and ARI. To keep precision high, both Lin-LS and LS-SSC tend to overestimate the number clusters in medium-size datasets.

Zooming into the cases where cameras of the same model share some commonalities in SPNs, this clearly introduces a certain level of ambiguity. In Figure 7 (c) and (d) we report results on datasets containing multiple camera models, each model with 5 camera instances. Despite the fact that all methods suffer from performance degradation, SSC-NC outperforms other methods in $\mathcal{D}_5^{sm}, \mathcal{D}_{10}^{sm}$, while LS-SSC is superior in all other configurations. In Figure 7 (c) and (d), the superiority of SSC-NC over SSC is evident. We argue that, in such complicated contexts where SPNs of the same camera model stay close to each other, SSC-NC can find a better representations of data.

We replicate the evaluation of all methods on medium-size datasets of Vision, see Figure 9. MSC, SSC-NC and LS-SSC perform on par with each other, but SSC-NC achieves more accurate estimation on the number of clusters. On the other hand, SSC-NC also obtains more accurate results than SSC in almost all configurations (7 out of 8). It seems that Lin-LS outperforms all other methods, however, we argue that its performance gain is partially due to high number of unclustered fingerprints it produces. We show in Figure 8 the number of unclustered fingerprints of LS-SSC and Lin-LS on medium-size datasets of Dresden and Vision. It is evident that Lin-LS produces more outliers than LS-SSC, thus gaining a certain advantage over precision, and then $\mathcal{F}$-measure as a consequence.

D. Numeric Results on Large-scale Datasets

In practice, there exist large-scale contexts where a large number of images need to be clustered. In Dresden, we conduct experiments on datasets containing 30 to 74 cameras, and the number of images exceeds 6000, while in Vision the number of cameras ranges from 21 to 33 and the number of images exceeds 4000. To the best of our knowledge, Lin-LS [15] is the only method proposed for large-scale clustering of camera fingerprints, thus results are compared only with it.

As depicted in Figure 10, Lin-LS achieves high precision, which means $\overline{FP}$ is negligible. Nevertheless, in order to keep high precision a noticeable number of fingerprints are
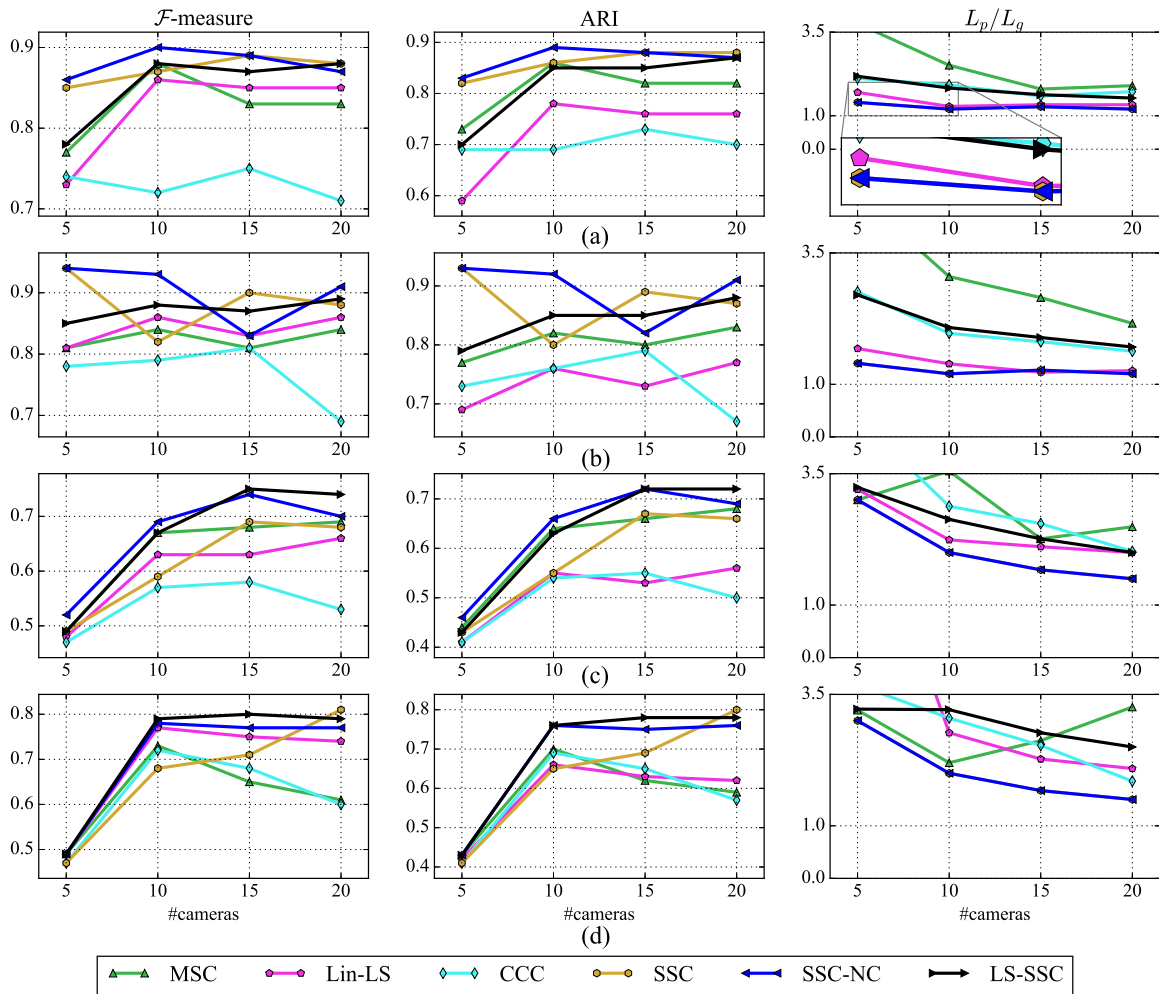
Fig. 7. Clustering performance on medium-size datasets of Dresden: (a) symmetric: $\mathcal{D}_5^s, \mathcal{D}_{10}^s, \mathcal{D}_{15}^s, \mathcal{D}_{20}^s$ (b) asymmetric: $\mathcal{D}_5^a, \mathcal{D}_{10}^a, \mathcal{D}_{15}^a, \mathcal{D}_{20}^a$ (c) symmetric + same model: $\mathcal{D}_5^{sm}, \mathcal{D}_{10}^{sm}, \mathcal{D}_{15}^{sm}, \mathcal{D}_{20}^{sm}$ (d) asymmetric + same model: $\mathcal{D}_5^{am}, \mathcal{D}_{10}^{am}, \mathcal{D}_{15}^{am}, \mathcal{D}_{20}^{am}$. Better viewed in color.
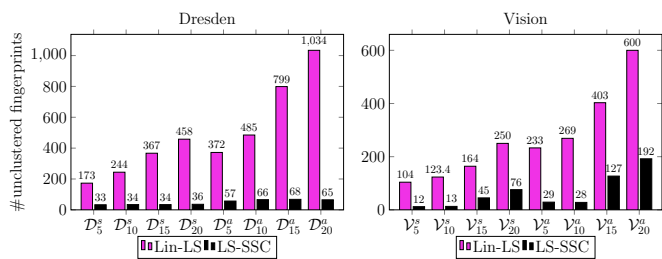

Fig. 8. Number of unclustered fingerprints on medium-size datasets of Dresden and Vision.

not clustered. Unclustered fingerprints essentially causes low recall, or equivalently high FN due to the separation of pairs belonging to the same cluster. On the contrary, LS-SSC produces less precise clusters with precision from 80% to 100%. One advantage of our method is the achievement of relatively high recall which slightly oscillates around 80%. Apart from keeping precision and recall balanced, we obtain high $\mathcal{F}$-measure. LS-SSC can cluster the whole Dresden dataset with $\mathcal{F}$-measure higher than 80% which substantially improves the 64% obtained by Lin-LS. The improvement of LS-SSC over

Lin-LS should be further amplified because Lin-LS requires to access 1024×1024 fingerprints in refining step while LS-SSC only works on 512×512 fingerprints. Moreover, as depicted in Figure 10 (last panel), LS-SSC produces a higher number of clusters than the ground-truth clusters, but the ratio between the two quantities is relatively constant when the dataset size grows. Vice versa, for Lin-LS this ratio rapidly increases.

Shown in Figure 11 are the performance of Lin-LS and LS-SSC on Vision dataset. Lin-LS again produces highly precise clusters, but tends to overestimate the number of ground-truth clusters. The $\mathcal{F}$-measure scores of the two methods are close since unclustered fingerprints are not accounted for precision computation.

In Lin-LS, the main cause of unclustered fingerprints are due to the merging step. If the merging threshold is too high, small subclusters cannot be merged to form larger subclusters, and thus filtered out in the end. On the other hand, in LS-SSC a fingerprint is unclustered if the correlation between fingerprint and all available cluster centroids is smaller than a threshold that was used to exclude the null hypothesis. Also for the case of large-scale datasets, we show the number of unclustered fingerprints in Lin-LS and LS-SSC, see last panel of Figure
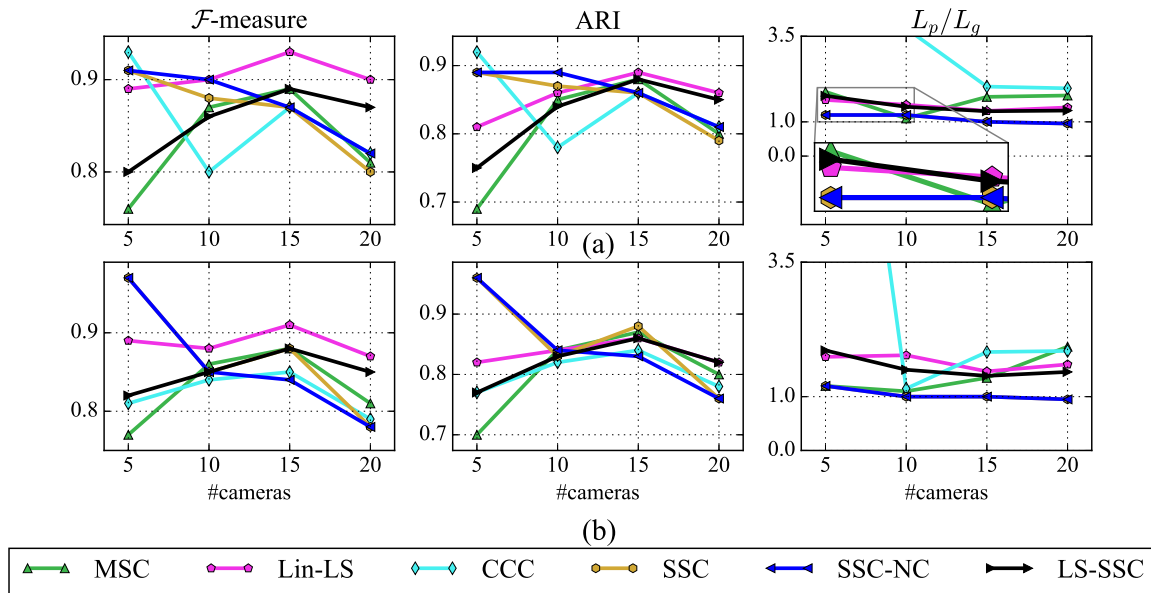
Fig. 9. Clustering performance on medium-size datasets of Vision: (a) symmetric: $\mathcal{V}_5^s, \mathcal{V}_{10}^s, \mathcal{V}_{15}^s, \mathcal{V}_{20}^s$ (b) asymmetric: $\mathcal{V}_5^a, \mathcal{V}_{10}^a, \mathcal{V}_{15}^a, \mathcal{V}_{20}^a$. Better viewed in color.
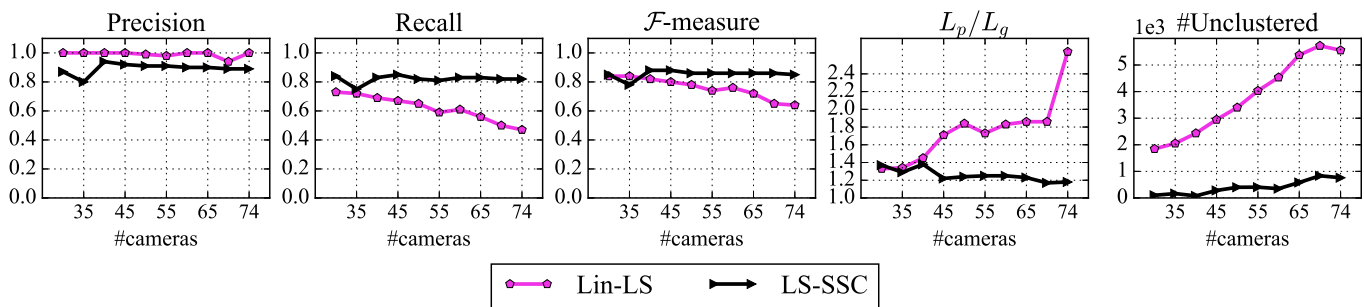

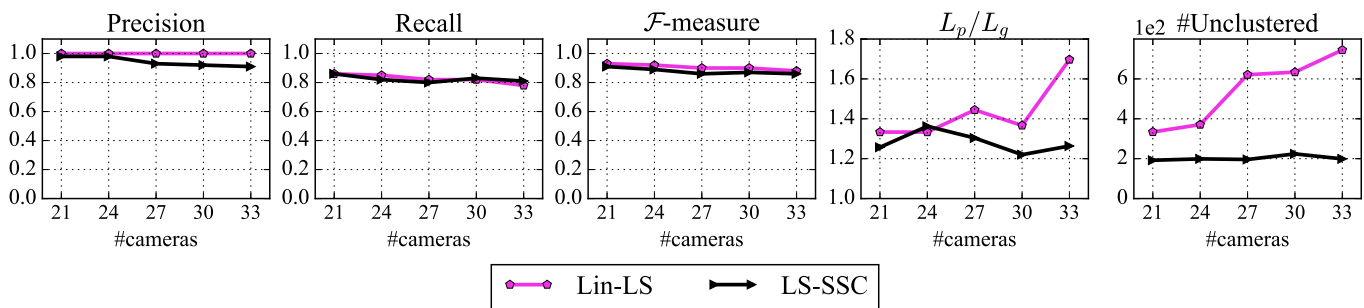Fig. 10. Clustering results on large-scale datasets of Dresden.

Fig. 11. Clustering results on large-scale datasets of Vision.

10, 11. In this scenario, it is clear that to keep precision high Lin-LS produces large number of unclustered fingerprints, not comparable with unclustered fingerprints in LS-SSC. The advantage of this mechanism is to reduce false alarm rate, but its downside is evident since data of interest could be ignored by the algorithm. LS-SSC provides a reasonable tradeoff allowing to cluster large-scale databases without skipping too many images which might be important for forensic analysis.

E. LS-SSC Robustness Analysis

In this section, we analyze the robustness of LS-SSC in more realistic testing configurations.

1) Presence of outliers: Firstly, we test the robustness of LS-SSC to outliers. We select images coming from 20 cameras of Vision, and add 50 images randomly collected from Facebook (from different entities) to make sure that they do not share the same source camera. On this dataset, LS-

SSC achieves $\mathcal{F}$-measure 0.89. Remarkably, LS-SSC assigns 69 images as unclustered, in which 33 out of 50 images are truthfully outliers.

2) Double JPEG compression: Images taken via smartphones usually undergo double JPEG compression once being available on social media sites. Therefore, we test the robustness of LS-SSC on images coming from 20 cameras of Vision, further compressed using `convert` tool provided by `ImageMagick`. The compression quality ranges from 50 to 95 (step 5). Results in Table III expose very reasonable and pretty stable performance of LS-SSC over different quality factors. Indeed, the algorithm is generally robust to double JPEG compression if the quality factor of the second compression is more than 65. Clustering performance starts to drop if images are aggressively compressed (quality factor smaller than 65).

TABLE III
NUMERIC RESULTS OF LS-SSC ON DOUBLE COMPRESSED IMAGES.

Metric	Quality factor									
	50	55	60	65	70	75	80	85	90	95
$\mathcal{P}$	0.79	0.81	0.84	0.88	0.90	0.93	0.94	0.95	0.93	0.96
$\mathcal{R}$	0.58	0.61	0.66	0.72	0.77	0.80	0.83	0.86	0.85	0.88
$\mathcal{F}$	0.67	0.70	0.74	0.79	0.83	0.86	0.88	0.90	0.89	0.92
ARI	0.65	0.68	0.73	0.78	0.82	0.85	0.87	0.89	0.88	0.88

In practice, images may come from online social networks where they undergo double compression. In such scenario, SPNs are further distorted due to resizing, and it has been confirmed by [14] that performance of all methods drop.

3) Different SPN sizes: Next, we validate the robustness of LS-SSC to different sizes of SPN. We pick the same set of images used in previous experiment, but crop the top-left region to 4 different sizes: $256 \times 256, 512 \times 512, 768 \times 768, 1024 \times 1024$. The hyperparameter γ is also re-estimated on the development set where images are cropped to similar sizes. The values of γ for each of corresponding size are $0.0045, 0.0018, 0.0012, 0.0008$. In Table IV, the performance generally improves if larger-size SPNs are used. Nevertheless, the results also suggest that using SPN sizes larger than 512×512 is not the key for the success of LS-SSC. Indeed, using 768×768 does not gain any improvement over 512×512 SPNs, and using 1024×1024 SPNs brings only a minor improvement.

TABLE IV
NUMERIC RESULTS OF LS-SSC ON SPNS OF DIFFERENT SIZES.

Metric	SPN size			
	256×256	512×512	768×768	1024×1024
$\mathcal{P}$	0.85	0.92	0.92	0.89
$\mathcal{R}$	0.84	0.84	0.85	0.90
$\mathcal{F}$	0.84	0.88	0.88	0.89
ARI	0.83	0.87	0.87	0.88

4) Few images per camera: In some specific contexts, forensic analysts might face with databases where the number of cameras is higher than the average number of images acquired by each camera (one camera per each model). To simulate such context, we start with an original set of 20 cameras selected from Dresden. The number of images on each camera alternatively ranges from 10 to 50 (step 10). For each image, we crop at 50 different positions, ending an
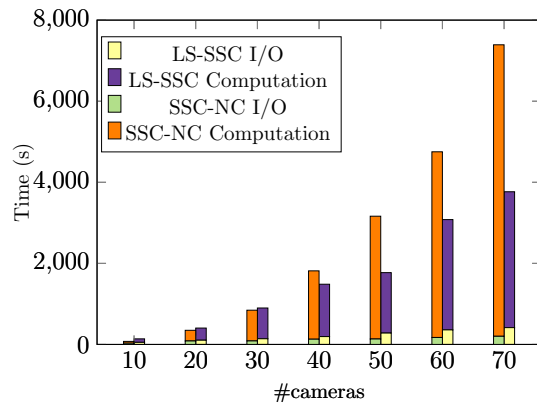

Fig. 12. Running time of SSC-NC and LS-SSC.

augmented set of images coming from 50 cameras. Finally we obtain a dataset of 12000 images of 400 cameras. It is a challenging dataset since the number of cameras is high, while the number of images for each camera is much lower. LS-SSC assigns images into 258 clusters, and 469 images remain unclustered. Obviously, many small-size clusters are hard to be discovered due to random splitting.

It is acknowledged in [15] that Lin-LS is especially designed to cope with such scenarios. However, such capability comes at a cost of discarding many outliers, which might leave images of interest out of consideration. Lin-LS assigns images into 892 clusters, while 4083 images remain unclustered. We obtain an $\mathcal{F}$-measure 47% in this dataset, while Lin-LS achieves 52%, at a price of a 10 times larger number of unclustered images.

In this scenario, LS-SSC performs not very well, but this is somehow inherently defined in the method itself. Indeed, we know from the theory that learning sparse representation of camera fingerprints requires a sufficient number of images per camera. Without this assumption, the algorithm might learn inexact representations which usually result in high $\overline{FP}$.

F. Running time analysis

We measure the running time of SSC-NC and LS-SSC on Dresden images, where the number of cameras ranges from 10 to 70. To observe the running time of SSC-NC we assume RAM is sufficient to catch all fingerprints of 70 cameras, and allows to solve the optimization in Eq. (8). Figure 12 reveals the fact that LS-SSC requires higher I/O cost due to extra reading/writing operations. SSC-NC, on the other hand, requires much higher computational cost, which are critical in practical usages. For LS-SSC, it takes approximately 1 hour and 20 minutes to cluster the whole Dresden dataset. In the case of limited RAM, LS-SSC requires more I/O time while SSC-NC cannot be operated.

VI. CONCLUSION

We have introduced a clustering framework by exploiting linear dependencies among SPNs in their intrinsic vector subspaces. Each SPN is expressed as a sparse linear combination of all other SPNs. Finding such sparse combinations is equivalent to solving LASSO with constraints, which is done

efficiently by ADMM method. Our algorithm can be extended to the case of large-scale databases thanks to the proposed divide-and-conquer strategy. Experiments prove the advantage of sparse representation over normalized correlation.

Future extensions will be dedicated to combining sparse representation learning and clustering into a unified end-to-end procedure. Moreover, we foresee to further study the impact of cluster cardinality on the method performance.

APPENDIX A
DERIVATION OF $\mathbf{V}$ UPDATE IN ALGORITHM 1

At each iteration of CONSTRAINED_LASSO, $\mathbf{V}$ is updated by:

$$\mathbf{V} = \arg\min_{\mathbf{V}} f(\mathbf{V}),$$

where

$$f(\mathbf{V}) = \gamma\|\mathbf{V}\|_1 + \langle \mathbf{\Lambda}, \mathbf{Z} - \mathbf{V}\rangle + \frac{\eta}{2}\|\mathbf{Z} - \mathbf{V}\|_F^2.$$

$$\frac{\partial f}{\partial \mathbf{V}_{ij}} = \gamma\frac{\partial|\mathbf{V}_{ij}|}{\partial \mathbf{V}_{ij}} + \eta\mathbf{V}_{ij} - \eta\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right)$$

$$= \begin{cases} -\gamma + \eta\mathbf{V}_{ij} - \eta\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right), & \mathbf{V}_{ij} < 0 \\ \gamma + \eta\mathbf{V}_{ij} - \eta\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right), & \mathbf{V}_{ij} > 0 \\ \text{undefined}, & \mathbf{V}_{ij} = 0 \end{cases}$$

$$\frac{\partial f}{\partial \mathbf{V}_{ij}} = 0 \text{ then}$$

$$\mathbf{V}_{ij} = S_{\frac{\gamma}{\eta}}\left(\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right)$$

$$= \begin{cases} \frac{\gamma}{\eta} + \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}, & \text{if} \quad \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta} < -\frac{\gamma}{\eta} \\ \frac{-\gamma}{\eta} + \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}, & \text{if} \quad \mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta} > \frac{\gamma}{\eta} \\ 0, & \text{if} \quad \left|\mathbf{Z}_{ij} + \frac{\mathbf{\Lambda}_{ij}}{\eta}\right| \le \frac{\gamma}{\eta} \end{cases}.$$

Solution $\mathbf{V} \in \mathbb{R}^{n \times n}$ might violate two constraints in Eq. (8). Denote $\mathcal{C}_1$ the set of all zero-diagonal and $\mathcal{C}_2$ the set of non-negative matrices. $\mathcal{C}_1, \mathcal{C}_2$ are convex. To impose the two constraints on $\mathbf{V}$, it is equivalent to find $\mathbf{V}^{(1,2)} \in \mathcal{C}_1 \cap \mathcal{C}_2$ that minimizes f. This can be obtained via von Neumann's alternating projections [46]: first Euclidean projection onto $\mathcal{C}_1$, and second Euclidean projection onto $\mathcal{C}_2$. Since $\mathbf{V} \in \mathbb{R}^{n \times n}$ is a minimizer of f, $\mathbf{V}^{(1,2)}$ can be obtained by two successive projections:

$$\mathbf{V}^{(1)} = \arg\min_{\tilde{\mathbf{V}} \in \mathcal{C}_1} \|\mathbf{V} - \tilde{\mathbf{V}}\|_F^2,$$

$$\mathbf{V}^{(1,2)} = \arg\min_{\tilde{\mathbf{V}} \in \mathcal{C}_2} \|\mathbf{V}^{(1)} - \tilde{\mathbf{V}}\|_F^2.$$

The two projections are implemented element-wise as in Eq. (11) and Eq. (12), respectively.

REFERENCES

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

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

[3] R. Li, C. T. Li, and Y. Guan, "A compact representation of sensor fingerprint for camera identification and fingerprint matching," in *Proc. of IEEE ICASSP*, 2015, pp. 1777–1781.

[4] X. Lin and C. T. Li, "Preprocessing reference sensor pattern noise via spectrum equalization," *IEEE Trans. on Information Forensics and Security*, vol. 11, no. 1, pp. 126–140, 2016.

[5] G. J. Bloy, "Blind camera fingerprinting and image clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 532–534, 2008.

[6] C. T. Li, "Unsupervised classification of digital images using enhanced sensor pattern noise," in *Proc. of IEEE ISCAS*, 2010, pp. 3429–3432.

[7] C.-T. Li and X. Lin, "A fast source-oriented image clustering method for digital forensics," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, 2017.

[8] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti, "Fast image clustering of unknown source images," in *Proc. of IEEE WIFS*, 2010, pp. 1–5.

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

[10] O. M. Fahmy, "An efficient clustering technique for cameras identification using sensor pattern noise," in *Proc. of IWSSIP*, 2015, pp. 249–252.

[11] B. b. Liu, H. K. Lee, Y. Hu, and C. H. Choi, "On classification of source cameras: A graph based approach," in *Proc. of IEEE WIFS*, 2010, pp. 1–5.

[12] I. Amerini, R. Caldelli, P. Crescenzi, A. D. Mastio, and A. Marino, "Blind image clustering based on the normalized cuts criterion for camera identification," *Signal Processing: Image Communication*, vol. 29, no. 8, pp. 831–843, 2014.

[13] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "Correlation clustering for PRNU-based blind image source identification," in *Proc. of IEEE WIFS*, 2016, pp. 1–6.

[14] ——, "Blind PRNU-based image clustering for source identification," *IEEE Trans. on Information Forensics and Security*, vol. 12, no. 9, pp. 2197–2211, 2017.

[15] X. Lin and C. T. Li, "Large-scale image clustering based on camera fingerprints," *IEEE Trans. on Information Forensics and Security*, vol. 12, no. 4, pp. 793–808, 2017.

[16] Q.-T. Phan, G. Boato, F. G.B. De Natale, "Image clustering by source camera via sparse representation," in *Proc. of the 2nd MFSec*, 2017, pp. 1–5.

[17] M. Chen, J. Fridrich, and M. Goljan, "Digital imaging sensor identification (further study)," in *Proc. of SPIE Electronic Imaging, Security, Steganography, Watermarking of Multimedia Contents IX*, vol. 6505, 2007.

[18] C. Li, "Source camera identification using enhanced sensor pattern noise," *IEEE Trans. on Information Forensics and Security*, vol. 5, no. 2, pp. 280–287, 2010.

[19] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[20] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, "Robust subspace clustering," *The Annals of Statistics*, vol. 42, 2014.

[21] Y.-X. Wang and H. Xu, "Noisy sparse subspace clustering," *Journal of Machine Learning Research*, vol. 17, no. 12, pp. 1–41, 2016.

[22] D. L. Donoho, "For most large underdetermined systems of equations, the minimal ℓ_1-norm near-solution approximates the sparsest near-solution," in *Technical report (Stanford University. Dept. of Statistics)*, 2004. [Online]. Available: https://books.google.it/books?id=dnX_GwAACAAJ

[23] M. Goljan, J. Fridrich, and T. Filler, "Managing a large database of camera fingerprints," in *Proc. of SPIE 7541, Media Forensics and Security II*, vol. 7541, 2010, pp. 7541–7541–12.

[24] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, "Compressed fingerprint matching and camera identification via random projections," *IEEE Trans. on Information Forensics and Security*, vol. 10, no. 7, pp. 1472–1485, 2015.

[25] M. Soltanolkotabi and E. J. Candés, "A geometric analysis of subspace clustering with outliers," *The Annals of Statistics*, vol. 40, 2011.

[26] Q. Rao and J. Wang, "Suppressing random artifacts in reference sensor pattern noise via decorrelation," *IEEE Signal Processing Letters*, vol. 24, no. 6, pp. 809–813, 2017.

[27] R. Li, C.-T. Li, and Y. Guan, "Inference of a compact representation of sensor fingerprint for source camera identification," *Pattern Recognition*, vol. 74, pp. 556–567, 2018.

[28] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, "Large-scale image retrieval based on compressed camera identification," *IEEE Trans. on Multimedia*, vol. 17, no. 9, pp. 1439–1449, 2015.

[29] A. Y. Yang, Z. Zhou, A. G. Balasubramanian, S. S. Sastry, and Y. Ma, "Fast ℓ_1-minimization algorithms for robust face recognition," *IEEE Trans. on Image Processing*, vol. 22, no. 8, pp. 3234–3246, 2013.

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[31] D. Shullani, M. Fontani, M. Iuliani, O. A. Shaya, and A. Piva, "VISION: a video and image dataset for source identification," *EURASIP Journal on Information Security*, vol. 2017, no. 1, 2017.

[32] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. of the 14th NIPS*, 2001, pp. 849–856.

[33] U. V. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[34] C. You, D. P. Robinson, and R. Vidal, "Provable self-representation based outlier detection in a union of subspaces," in *Proc. of IEEE CVPR*, 2017, pp. 4323–4332.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of the 2nd KDD*, 1996, pp. 226–231.

[36] J. Fridrich and M. Goljan, "Derivation of ROCs for composite fingerprints and sequential trimming," in *Technical Report (Binghamton University)*, 2010. [Online]. Available: http://www.ws.binghamton.edu/fridrich/Research/rocs.pdf

[37] J. Lukas, J. Fridrich, and M. Goljan, "Determining digital image origin using sensor imperfections," in *Proc. of SPIE - The International Society for Optical Engineering*, vol. 5685, 2005, pp. 5685–5685–12.

[38] J. Eklann, "Source camera classification and clustering from sensor pattern noise," Master's thesis, Dept. Appl. Inf. Tech., Chalmers Uni. Tech., Gothenburg, Sweden, 2012.

[39] T. Gloe and R. Böhme, "The 'Dresden Image Database' for benchmarking digital image forensics," in *Proc. of the 25th ACM SAC*, vol. 2, 2010, pp. 1585–1591.

[40] J. Besag, "On the statistical analysis of dirty pictures," *The Royal Statistical Society B*, vol. 48, no. 3, pp. 48–259, 1986.

[41] D. Huang, J.-H. Lai, C.-D. Wang, "Combining multiple clusterings via crowd agreement estimation and multi-granularity link analysis," *Neurocomputing*, vol. 170, pp. 240–250, 2015.

[42] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[43] "LAPACK," http://www.netlib.org/lapack/, accessed: 2017-08-31.

[44] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[45] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE: A raw images dataset for digital image forensics," in *Proc. of the 6th ACM MMSys*, 2015, pp. 219–224.

[46] W. Cheney and A. A. Goldstein, "Proximity maps for convex sets," *The American Mathematical Society*, vol. 10, no. 3, pp. 448–450, 1959.