POLITECNICO DI TORINO

Master's Degree in COMMUNICATIONS AND COMPUTER NETWORKS ENGINEERING

Master's Degree Thesis

Reduced complexity correlation-based blind clustering of camera fingerprints

Supervisor

Prof. Tiziano BIANCHI

Candidate

Matteo SPANO'

April 2021

Summary

Every camera leaves stable and deterministic invisible traces on each captured photo which are due to small sensor imperfections. These traces, or fingerprints, can be used to solve various problems, for example deciding whether two (or more) images comes from the same sensor, or which sensor, from a given list, took a picture under exam.
In this thesis we consider the problem of common image source detection. We experiment with low complexity image clustering algorithms to regroup pictures based on the similarities of their camera fingerprints. The proposed improvements in the clustering and attraction phases are designed to work on top FICFO [1] and RCIC [2] algorithms. Namely we consider the following reference algorithms:
• Bloy algorithm [3] randomly searches for a pair with correlation value greater than a threshold. Bloy generates hierarchical clustering using a sequence of merge operations. Iteratively, it compares the references and the remaining fingerprints.
• FICFO computes a ranking index RI as a metric of the quality of each estimated fingerprint and it stores all the fingerprints sorted in descending order of RI. FICFO produces a partition that is usually an overclustering of little pure (or quite pure) groups, following a fast and efficient procedure.
• RCIC instead, saves the fingerprints in a random order and it has to regroup without any knowledge of the goodness of the attractors, producing stochastic outputs usually in a slightly less efficient way.
• Khan attraction [1] conducts a merge of pairs clusters having maximum correlation between their centroids, until they are greater than the threshold.
We always work in a supervised blind scenario, where the information about the cameras is not available, but we can still set some parameters to tune the algorithms. Our thesis presents an ablation study of the best methods proposed, where some parameters are tested and removed one by one, in order to understand their contribution to the performances.

For the clustering phase we compare the following methods:
• Full Matrix [4]: this is based on computing a matrix of distances between every possible couple of fingerprints, for a complete knowledge of the problem, followed

by a hierarchical clustering.
• First reference: chooses the first fingerprint in the list of currently unclustered fingerprints as reference for attracting other fingerprints, without updating it.
• Fair: updates the reference fingerprint in the above scheme by computing a fair average of the currently attracted fingerprint.
• Weighted: updates the reference fingerprint by computing a weighted average of the currently attracted fingerprint, where the weights are proportionally to the ordering factor RI.

After this first step, we are free to consider the obtained partition already as the final image clustering we were searching for, or we can apply an optional attraction phase, aiming for a better clustering effectiveness, chosen between the following methods:
• Full Matrix attraction: the Full Matrix method is applied to a preexisting partition.
• Hierarchical Agglomerative [4]: builds the binary tree of distances between all the centroids. At a pre-computed height, it splits the tree into a cluster partition.
• Static/Dynamic Pure attraction: refines the clusters internally, moving some fingerprints to the correct cluster they should lie in, according to a k-means clustering strategy.

In the thesis we compare the reference methods with the proposed technique, considering four datasets with different number of source cameras and different numbers of images per camera. For each dataset, several choices parameters were tested. The results show that even fast low complexity approaches can achieve very good results, maintaining a high level of precision and scaling the computations needed for the Full Matrix technique.
Our experiments reveal that some instances of FICFO-A Fair and RCIC-A Fair may be used as fast, low complexity codes for image clustering. This is particularly true in a scenario in which we have access to a huge number of big sized, bright and not saturated fingerprints.

More in details, both RCIC and FICFO Fixed Reference clustering perform always worse compared to the other clustering algorithms. Even in the FICFO scenario, where the first fingerprint has the highest value of reliability, it is not wise to fix the first attractor as the centroid of the cluster. The Fair approach instead, computing a mean of all the fingerprints we merged in the cluster and obtaining the centroid from all of them, permits to average the errors and possibly remove the noise components, obtaining better performances.
The Weighted version gives quite the same performances of the average, but it is based on the assumption of some very high-quality image present in the dataset,

therefore we prefer not to use it as a base for attraction, also because it is not compatible with the RCIC algorithm.

If we relax the problem and we do not account for the computational complexity, Full Matrix offers an extremely accurate solution for easy datasets, while Bloy seems even a bit better in facing the challenge of clustering on hard datasets. The scarcity of information about the cameras has a strong impact on the Full matrix, indeed very dependent on the correlation values it pre-computes.

RCIC appears to be a good fast clustering algorithm, but because of its worse initial performances respect to FICFO, it is not very suitable for an attraction phase. Indeed, the number of computations increases a lot and overall, we have a noticeable degradation of performances. For the same dataset and input parameters, RCIC-A performs always worse respect to the simpler FICFO Fair, plus we must take into account the stochastic behavior of RCIC. If we need to execute each RCIC code many times to obtain a stable average of the clusters, we must multiply the computational complexity times the number of runs we tested, receiving a big disadvantage respect to the deterministic algorithm.

The purification step, in both its forms, static or dynamic, introduces a non-negligible complexity to the attraction phase. The inputs are already precise as they are computed by FICFO and RCIC, especially in their Fair version. In conclusion, the pure algorithms perform bad both as an attraction phase and as a pre-attraction step. They have to check the full set of fingerprints and they re-assign too many items, usually to wrong clusters or they split the partition to a huge amount of groups, very hard to work with.

As a future work, we may work with a sparse representation [5] of the fingerprints. Each proposed algorithm can be also extended to the case of large-scale databases [1] in a divide-and-conquer strategy, loading in memory only parts of the dataset. We can formulate a better metric for Weighted clustering, to distinguish few very good images, about one for each camera, and work by regrouping around them. An efficient low-complexity check phase should be able to precisely locate the errors (without checking every fingerprint) and to move them to the right cluster.

Acknowledgements

I am deeply grateful to my tutor Professor Tiziano Bianchi, with his precious advice, his scrupulous and punctual support, his in-depth knowledge on the subject and availability, helped me in every step of my thesis.

An affectionate thanks goes to my mum Paola and my dad Felice who always encouraged my study choices and supported me to achieve my successes.

I must also warmly thank my best friend Mirna, with her kindness and patience, she made my coffee breaks relaxing, giving me new strength to face every difficulty.

Thanks to my relatives and my friends who, in this pandemic time, continue to show their love and presence, for being valuable people in my life and for making this amazing milestone possible.

"Ad maiora semper!"

My thoughts are also with my grandma Franca who continues to be proud of me from up there and holds out her hand to me.

Table of Contents

List of Tables

List of Figures

Acronyms

ARI
Adjusted Rand Index

BCFIC
Blind Camera Fingerprinting and Image Clustering

CFA
Color Filter Array

CFIC
Compressed Fingerprints based Image Clustering

CR
Complexity reduction

CRLB
Cramer-Rao Lower Bound

F
F-measure, harmonic mean of P and R

FICFO
Fast Image Clustering based on Fingerprints Ordering

FICFO-A
Fast Image Clustering based on Fingerprints Ordering with Attraction

FM
Full Matrix

FPN

Fixed Pattern Noise

GOP

Group Of Pictures

$\mathbf{G}_T$

Ground truth

JPEG

Joint Photographic Experts Group

LASSO

Least Absolute Shrinkage and Selection Operator

LSIC

Large Scale Image Clustering

MSE

Mean Square Error

NC

Number of Classes

NCC

Normalized Cross Correlation

NMI

Normalized mutual information

NUA

Non-Unique Artifacts

P

Precision, percentage of items correctly clustered

PCE

Peak to Correlation Energy

PFA

Probability of False Alarm

PNN

Pairwise Nearest Neighbor

PRNU

Photo Response non Uniformity

PSD

Power Spectral Density

PSNR

Peak Signal to Noise Ratio

R

Recall, given a ground truth, percentage of unclustered images

RCIC

Reduced Complexity Image Clustering

RCIC-A

Reduced Complexity Image Clustering with Attraction

RI

Rand Index

SC

Size of Class

SPN

Sensor Pattern Noise

SSC

Sparse Subspace Clustering

WGN

White Gaussian Noise

Chapter 1

Introduction

1.1 Problem Statement

Communication through images has a greater immediacy and emotional value rather than the written text. The problem of deep fakes, images and videos generated by artificial intelligence, arises more and more often.
The Digital Forensic analysis of images is an important field of investigation. It branches in Computer Forensics and Multimedia Forensics. Nowadays, Multimedia Forensics is largely used in companies, industries, medias, and so on to support or refute an hypothesis in criminal or civil courts, to fight against cybercrime, to solve insurance claims and scientific frauds. The main tasks regarding Multimedia Forensics are:
• common source device clustering: given a set of images, we would like to find out which images were obtained using the same camera.
• device identification: prove that a given image was obtained by a specific device.
• integrity verification or forgery detection: discover malicious processing, detecting local inconsistencies, odd items added to or removed from the image during digital processing.
• recovery of processing history: lossy compression, multiple compressions (double JPEG problem), manipulations, filtering, cropping, resizing, contrast and brightness adjustment.

An active approach to the problems of source identification and integrity verification is the use of authentication: some features are extracted for generating authentication signature at the source side. The image integrity is verified by signature comparison at the receiver side. Even those watermarks, visible or invisible, embedded in the firmware, can be removed with different attacks.
The signal processing chain in digital cameras is complex and vary for different

camera types and models. It typically includes signal quantization, white balance, demosaicking (color interpolation), color correction, gamma correction, filtering, and JPEG compression [6].
Because details about the processing are usually hard-wired or proprietary, it is crucial to be able to reconstruct part of those data from the unique features of the pictures themselves, in a passive and blind approach. The advantages of a passive approach are that we don not need dedicated devices, a trusted infrastructure and a standard to make them inter-operable.

Common source device clustering estimates the origin of the pictures, the camera type and the conditions under which the image has been taken. These statistics may be already present in the header file and in the metadata. However, the metadata can be easily altered, so the information of their origin may be lost or corrupted.
We can therefore perform a "calligraphic expertise" on the matrix of pixels that constitute the image and extract an estimate of its fingerprint. A fingerprint is a unique deterministic trace left by each sensor on every photo produced by that specific camera.
Respect to source identification, common source clustering is much more computationally intensive, as $O(N^2)$ comparisons have to be made for N images, whereas source identification only needs N comparisons.

Image clustering faces the challenge of grouping big datasets of images into as few clusters as possible while maintaining high precision. In our case, each group has to contain only photos acquired from the same camera. Image clustering relying on camera fingerprints is a blind problem, where clustering is performed in the absence of any prior information, using passive authentication techniques that exploit camera fingerprints to regroup images taken by the same device.
Many approaches are referred to as supervised. They require a similarity threshold parameter inserted as input, in order to tune the algorithm on a benchmark dataset, during the hierarchical clustering step. This drawback is present in Bloy2008 [3], which contains a parameter that varies for different camera models, limiting its scalability and applicability in new environments [7].
In a unsupervised scenario, as the algorithm proposed by Li [8], image classification is considered with unknown number of cameras of unknown types. The classifier can perform at very accurate quality level and a contained error rate, without the user providing extra information about the dataset. The problem is that the images have to be analyzed in full-sized to compensate for the influence from the scene details. This task raises a lot the complexity and is not feasible in large image dataset classification. Because of that, the development of unsupervised methods has suffered for delays [8].

1.2 Contributions and organization of the thesis

In this thesis we consider the problem of common source device clustering. We experiment with low complexity image clustering algorithms to regroup pictures based on the similarities of their camera fingerprints. The proposed improvements in the clustering and attraction phases are designed to work on top of FICFO [1] and RCIC [2] algorithms.

☐ **Chapter 2** presents a background on PRNU and clustering literature. In particular, it introduces the reader to the camera sensor output model, the PRNU camera fingerprints estimation and the matching procedure. It contains also theoretical notions and different strategies for image clustering based on camera fingerprints.

☐ **Chapter 3** review the most important approaches in the literature of image clustering algorithms based on camera fingerprint:
• Bloy algorithm [3] randomly searches for a pair with correlation value greater than a threshold. Bloy generates hierarchical clustering using a sequence of merge operations. Iteratively, it compares the references and the remaining fingerprints.
• FICFO [1] computes a ranking index RI as a metric of the quality of each estimated fingerprint and it stores all the fingerprints sorted in descending order of RI. FICFO produces a partition that is usually an overclustering of little pure (or quite pure) groups, following a fast and efficient procedure.
• RCIC [2] instead, saves the fingerprints in a random order and it has to regroup without any knowledge of the goodness of the attractors, producing stochastic outputs usually in a slightly less efficient way.
• Khan attraction [1] conducts a merge of pairs clusters having maximum correlation between their centroids, until they are greater than the threshold. Here we discuss the advantages and the problems of these algorithms, especially their suitability for a blind supervised scenario, their computational complexity and their robustness when the number of clusters is much larger than the average number of images per camera.

☐ **Chapter 4** describes the different image clustering algorithms that are proposed in the thesis, the steps involved and the detailed implementation of their pseudo-codes. For the clustering phase we compare:
• Full Matrix [4]: this is based on computing a matrix of distances between every possible couple of fingerprints, for a complete knowledge of the problem,

followed by a hierarchical clustering.
• First reference: chooses the first fingerprint in the list of currently unclustered fingerprints as reference for attracting other fingerprints, without updating it.
• Fair: updates the reference fingerprint in the above scheme by computing a fair average of the currently attracted fingerprint.
• Weighted: updates the reference fingerprint by computing a weighted average of the currently attracted fingerprint, where the weights are proportionally to the ordering factor RI.

After this first step, we are free to consider the obtained partition already as the final image clustering we were searching for, or we can apply an optional attraction phase, aiming for a better clustering effectiveness, chosen between the following methods:
• Full Matrix attraction: the Full Matrix method is applied to a preexisting partition.
• Hierarchical Agglomerative [4]: builds the binary tree of distances between all the centroids. At a pre-computed height, it splits the tree into a cluster partition.
• Static/Dynamic Pure attraction: refines the clusters internally, moving some fingerprints to the correct cluster they should lie in, according to a k-means clustering strategy.

□ **Chapter 5** contains a discussion about the experimental setup and the evaluation metrics.
Our thesis presents an ablation study of the best methods proposed, where some parameters are tested and removed one by one, in order to understand their contribution to the performances. In this chapter we analyze the results in tables of performances. The final section compares the proposed algorithms among themselves and with the State of the art techniques.

□ **Chapter 6** concludes the work with a comment about the results obtained and some possible future refinement work, such as a sparse representation for the fingerprints, to be implemented over our proposed algorithms.

Chapter 2

Background on PRNU and clustering

In this chapter we define the technical terms we adopt in the thesis, as PRNU noise, cosine similarity and clustering. We introduce the reader to the camera sensor output model, the camera fingerprints estimation and the PRNU matching. We also present an overview of the procedures adopted in the literature for image clustering, such as sparse representation, k-means and Gaussian mixture.

2.1 PRNU noise

The fingerprint is related to the sensor pattern noise (SPN) of the camera and is unique for each brand, model and every instance of a device. The pattern noise is a combination of fixed pattern noise (FPN) and the photo response non uniformity (PRNU) [9]. The PRNU is the main component of the pattern noise and it is present in every image taken from a camera, due to the pixel non uniformity.
Due to intrinsic disconformities during the manufacturing process, each pixel has a slightly different sensitivity to light, that is why the PRNU is a unique sensor feature, like an intrinsic authentication watermark and it is related to a specific device.
When the pixel is hit by a fixed number of photons, it responds with a quantity of electrons which is proportional to a factor modelled by one plus a zero-mean noise-like signal K. This factor has slight variations for each pixel in the camera, hence leading to a Multiplicative Noise Pattern proper of each sensor.
The pattern noise is deterministic and remains a stable noise component, unaffected by exposure time, temperature or humidity [10].

The first easy approach to sensor biometrics is the investigation of defective pixels, hot and dead pixels, that scatter the sensor. Nowadays, we rely on PRNU noise that contain more information regarding all the array of pixels. PRNU can also be used in parallel with machine learning for reliable device identification.

2.1.1 PRNU Sensor Output Model

The quantized signal I is computed starting from the incident light intensity Y, the quantization noise Q, the PRNU factor K and considering some additive, zero-mean independent random noise L (including the shot noise, the dark current and the photon noise effects):

$$I = g^{\gamma}[(1+K)Y + L]^{\gamma} + Q \tag{2.1}$$

where g is the color channel gain factor that adjusts the pixel intensity level according to the sensitivity to spectral bands in order to obtain the correct white balance. γ is the gamma correction factor, typically set to γ= 0.45.

By applying the Taylor expansion to equation 2.1 and keeping the first two terms, each photo can be approximately described by a sum between a noise free term I_0, the signal I_0K_0 that introduces to the flat-field image a unique multiplicative noise pattern K_0= γ K and an independent random noise component R.

$$I \approx (gY)^{\gamma}\,(1 + \gamma K + \gamma\frac{L}{Y}) + Q \tag{2.2}$$

$$I \approx I_0 + I_0K_0 + R \tag{2.3}$$

The equivalent additive noise R comes by regrouping many noise terms considered as undesired disturbances to simplify the notation and it can be obtained as:

$$R = \gamma\, I_0\, \frac{L}{Y} + Q \tag{2.4}$$

2.2 PRNU detection

The PRNU detection, is obtained by comparing the noise residual W to a reference fingerprint K, according to a predefined threshold in a Neyman-Pearson approach, distinguishing between two hypotheses in a binary test:

- H_0: W = I K_0 + R_D where $K_0 \neq$ K (image not acquired by that sensor)
- H_1: W = I K + R_D where K_0 = K (image acquired by the same sensor)

H_1 hypothesis represents the Inter-class correlation, distributed in a sparse area with a big variance and positive mean, as in figure 2.1.
In the H_1 hypothesis the cross-correlation value is greater than the threshold.

$$C = \frac{\langle I\,K, I\,K_0 + R_D\rangle}{\|I\,K\|\,\|I\,K + R_D\|} > Th \tag{2.5}$$

Applying the Central Limit Theorem we obtain:

$$E\langle I\,K, I\,K + R_D\rangle = \|I\,K\|^2 \tag{2.6}$$
$$Var\langle I\,K, I\,K + R_D\rangle = \|I\,K\|^2\,\sigma^2(R_D) \tag{2.7}$$

H_0 hypothesis instead, is the Intra-class correlation, among different cameras (cross-camera). All the same camera fingerprints take place in a zero mean Gaussian shape, with variance equal to the inverse of the crop size of the image I the fingerprint is estimated from.
In the H_0 hypothesis the cross-correlation value is lower than the threshold.

$$C = \frac{\langle I\,K, I\,K_0 + R_D\rangle}{\|I\,K\|\,\|I\,K_0\,R_D\|} < Th \tag{2.8}$$

Applying the Central Limit Theorem we obtain:

$$E\langle I\,K, I\,K_0 + R_D\rangle = 0 \tag{2.9}$$
$$Var\langle I\,K, I\,K_0 + R_D\rangle = \|I\,K\|^2\,(\sigma^2(R_D) + \sigma^2(K)) \tag{2.10}$$

2.2.1 Correlation detector

False alarm threshold

Our clustering problem is based on the decision rule, the algorithm that takes as input C, the cross-correlation between a fingerprint and a group, and has to choose either H_0 or H_1 hypothesis.
If $C < Th$ we expect as output H_0, the correct rejection of the fingerprint. The likelihood instead of accepting C is P_{FA}, the probability of false alarm (prob. of deciding H_1 when H_0 is true).

Neyman-Pearson criterion maximize the probability of detection P_D (prob. of deciding H_1 when H_1 is true) and compute the similarity threshold Th starting

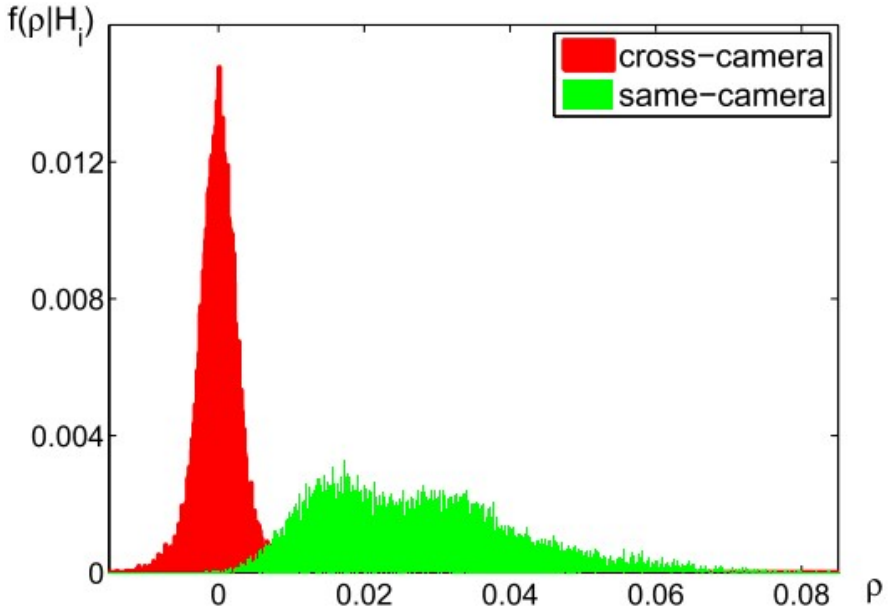

Figure 2.1: same-camera and cross-camera distribution

from P_{FA} as:

$$Th = \sqrt{2\,\sigma(C)}\; erfc^{-1}(2\,P_{FA}) \tag{2.11}$$

$$\|I\,K_0 + R_D\| = \sqrt{L\,\sigma^2(R_D) + L\,\sigma^2(K)} \tag{2.12}$$

$$E(C) = 0 \tag{2.13}$$

$$Var(C) = \frac{1}{L} \tag{2.14}$$

therefore, we can approximate the variance of the cross-correlation $\sigma(C)$ in the H_0 hypothesis, from equation 2.7, simply as the inverse of the crop size L.

Lin threshold

An alternative similarity threshold is the one proposed by Lin and Li in their article [11]. The adaptive thresholding significantly reduces the computational complexity and allows the clustering results of the smaller databases to be combined to give the solution to the $NC \gg SC$ problem, because it is capable of adaptively finding a suitable breaking point between the inter-class distribution and the Intra-class distribution. The threshold adjust itself by following distribution shifting, so it considers also the non-zero means null hypothesis.
Nevertheless, it tends to be unnecessarily confident when two distributions are

close. It is either too conservative for large datasets or overly aggressive for the small ones, making some of the Intra-class correlations misclassified as inter-class correlations. The following figure 2.3 shows in green Lin's threshold and in red a more ideal one proposed in the article [12].

Gaussian mixture threshold

We can adjust the threshold to follow the distribution shifting, by considering also the non-zero means inter-correlation hypothesis.
Once we have a first estimation of the clusters, we can have a successive attraction phase, in which we must also refine our threshold to work with centroids instead of fingerprints.
We can process our data to retrieve a split into a two-components Gaussian distribution, one component for the inter and the other for the Intra fingerprints. It makes use of the k-means++ algorithm (subsection 2.5.1). The average Intra-class similarity represents the density of the PRNU clusters.

$$\sigma_0 = variance(Gauss\_mix_0), \tag{2.15}$$
$$\mu_0 = mean(Gauss\_mix_0), \tag{2.16}$$
$$Th = \mu_0 + \sqrt{2 * \sigma_0} * erfc^{-1}(2 * P_{FA}) \tag{2.17}$$

We expect the H_0 hypothesis Gaussian distribution to behave very similarly to the approximations computed in formulas 2.13 and 2.14. We can also retrieve an

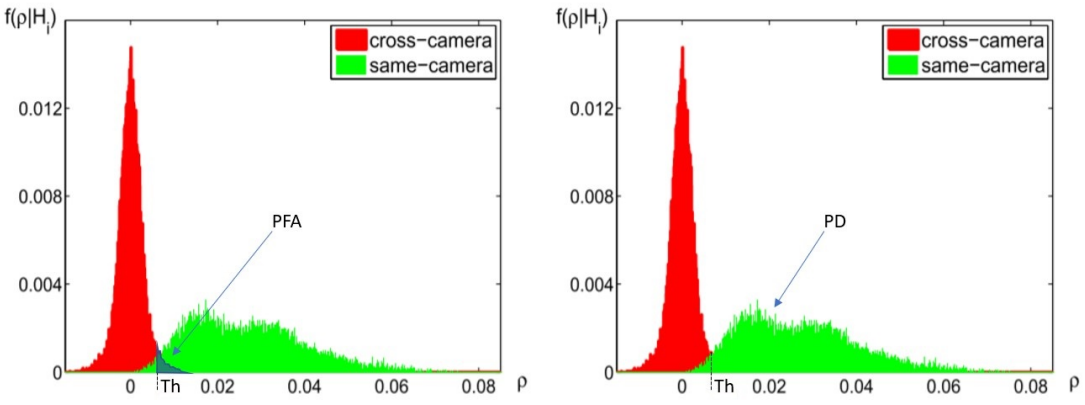

Figure 2.2: Neyman-Pearson Threshold and P_{FA}

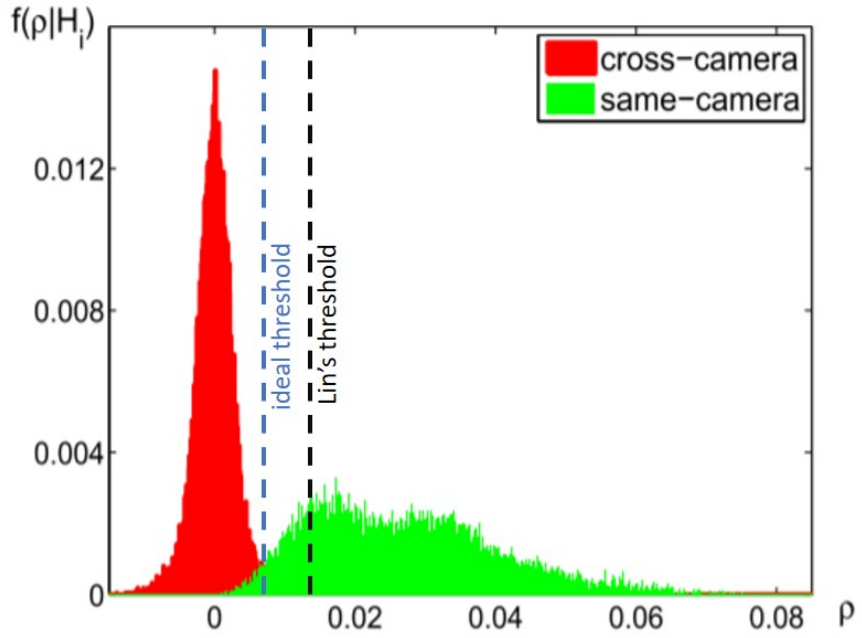

Figure 2.3: Lin's threshold in black, Inter-class distribution in green, Intra-class distribution in red

estimate for the H_1 hypothesis if we approximate the formulas 2.6 and 2.7 as:

$$\|I\,K + R_D\| = \sqrt{\|I\,K\|^2 + L\,\sigma^2(R_D)} \tag{2.18}$$

$$E(C) = \frac{\|I\,K\|}{\|I\,K + R_D\|} \tag{2.19}$$

$$Var(C) = \frac{\sigma^2(R_D)}{\|I\,K + R_D\|^2} \tag{2.20}$$

So, the Gaussian mixture must output:

$$H_0 : C = G(\mu_0, \sigma_0) \sim G(0, \frac{1}{L}) \tag{2.21}$$

$$H_1 : C = G(\mu_1, \sigma_1) \sim G\Big(\frac{\|I\,K\|}{\|I\,K + R_D\|}, \frac{\sigma^2(R_D)}{\|I\,K + R_D\|^2}\Big) \tag{2.22}$$

The Gaussian mixture threshold is a likelihood-based measures of model fit that include a penalty for complexity.

2.2.2 Peak to Correlation Energy

Another detector, instead of the similarity threshold, is the comparison of the Peak to Correlation Energy. In the PCE formula, $|C(0,0)|$ denotes the peak amplitude, the the height of the maximum in the cross-correlation between the extracted PRNU noise for a given image and that same image multiplied by the PRNU pattern of the camera to be verified.

$$PCE = \frac{|C(0,0)|^2}{\frac{1}{MN - |\Delta|} \sum_{ii} \sum_{jj} C(ii,jj)^2} \tag{2.23}$$

The denominator of PCE is the energy of the cross-correlation between the two PRNU patterns, multiplied by a factor, where MN represents the dimension of the normalized cross-correlation matrix C [13], Δ is a small square area around zero where a peak correlation is expected for correlated vectors and $|\Delta|$ is its cardinality. PCE shows good performance in PRNU comparison [14] because it is more robust to fluctuations in simple correlation and it can be used to manage PRNU synchronization like cropped images [15].

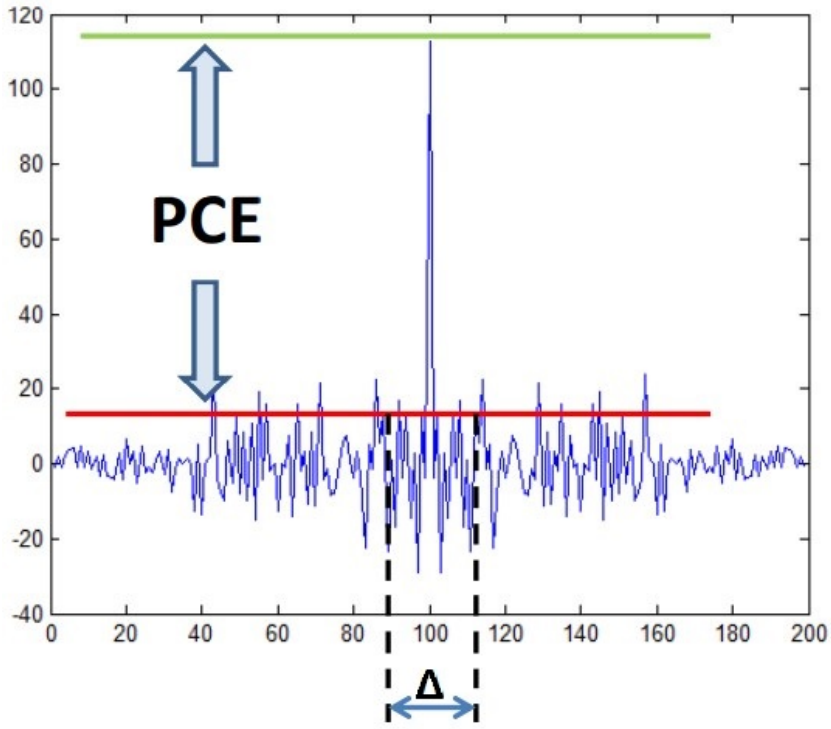

Figure 2.4: Peak to Correlation Energy

2.2.3 Sparse representation

A good strategy for decreasing the complexity is to work with reduced fingerprints. For clustering digital images, we can suppose only a subset of pixels is relevant in our analysis. The goal of sub-space clustering is to identify the number of sub-spaces, their dimensions, a basis for each sub-space, and the membership of each data point to its correct subspace. Each data point y can be expressed as the linear combination of a proper reduced basis of its data matrix X.

$$y = z_1X_1 + z_2X_2 + ... + z_iX_i + ...z_nX_n \tag{2.24}$$

The basis captures asymmetric relationships among data points and it must still contain information about the subspace where fingerprints of a camera lie in.
If the columns of X are contaminated by noise, as in every real scenario, or they are not well distributed, the formula:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \|Z\|_1 \\ \text{subject to} \quad & XZ = y \end{aligned} \tag{2.25}$$

might never output an analytic solution. The problem is NP-hard, and hence, for large graphs, finding the exact solution may become exceedingly complex and require an Integer Linear Programming (ILP) algorithm (by soft thresholding and computing analytical solutions) or some greedy approximations.

The sparse combination problem is efficiently solved in many articles as a LASSO (Least Absolute Shrinkage and Selection Operator) with constraints, by the Alternating Direction Method of Multipliers [12], when is reformulated as:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \frac{1}{2}\|XZ - X\|_F^2 + \gamma\|Z\|_1 \\ \text{subject to} \quad & \gamma \geq 0, \\ & Z = 0, \\ & diag(Z) = 0 \end{aligned} \tag{2.26}$$

The new representation is obtained by the l_1-regularized least squares. The regularization hyperparameter γ controls the trade-off between the sparsity of the solution we try to achieve and the reconstruction error. A too dense solution merges unrelated clusters, a too sparse results in many small clusters [5].
It is possible to compute the spectrum, or eigenvalues, of the similarity matrix of the data (performing dimensionality reduction) to apply after spectral clustering on it.

Experiments already done by other researchers, prove the advantage of sparse representation over normalized correlation. In our experiments, we always rely on the full fingerprint reference, hence we ensure more precise outputs.

2.3 PRNU Fingerprint matching

The main forensic tasks [16] performed using the camera sensor output model from the previous section are:
• Device identification: $corr(W, K)$ tests whether a given picture I was taken by a specific device. An estimate of the reference fingerprint K of the device is extracted from a dataset of training pictures. The noise residual W comes from the image and is correlated with the reference K of the selected device, modulated by the image intensity.
• Device linking: given two images I_1 and I_2, $corr(W_1, W_2)$ compares their noise residuals and determines whether the two pictures have been acquired by the same device.
• Fingerprint matching: from a set of pictures I acquired by the same camera, it estimates the reference fingerprint K of the group of pictures and it compare K with each entry of a given list of fingerprints Ref_list. $corr(Ref\_list_{ii}, K)$ for each entry in the list, determines which device in the database has acquired the given pictures. If no device provides a correlation higher than a given threshold, the device is declared absent from the list.

The extracted fingerprint from an image can be severely contaminated by interferences. In order to guarantee the reliability and the accuracy of clustering, the dimension of camera fingerprint has to be very large, for example, 511 × 511 pixels or above [11], in order to fully exploit the curse of dimensionality property, in figure 2.9.

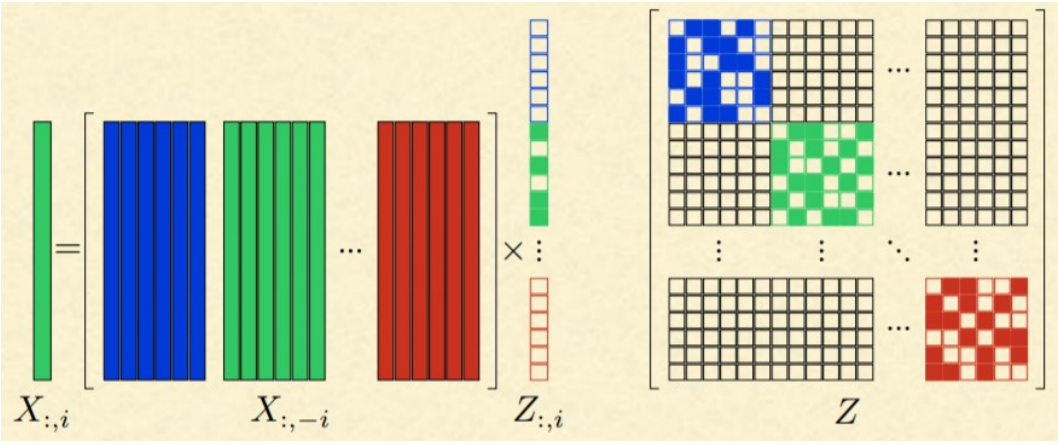

Figure 2.5: Sparse representation basis

2.3.1 PRNU fingerprint estimation

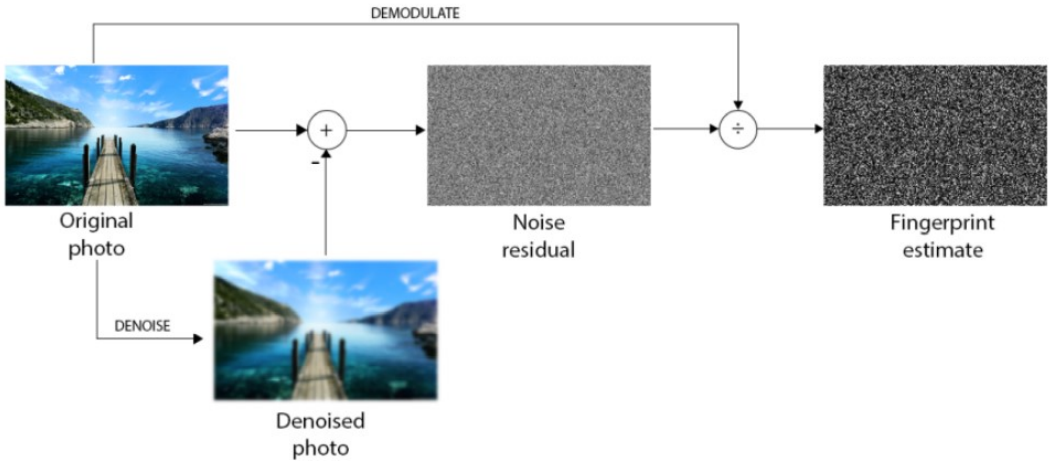

Figure 2.6: PRNU fingerprint estimation

We estimate the fingerprints of each sensor from a set of images taken by the camera. We suppress the noiseless image I_0 by subtracting from both sides of the equation 2.3 the denoised image $\hat{I}_0 = F(I)$, obtained using a denoising filter F.
In our experiments, each image is denoised using Mihcak filtering operation [17] and subtracted from the original image to get the noise residual W, as in figure 2.6.

$$W = I - \hat{I}_0 = I\,K_0 + R_D \tag{2.27}$$

The Mihcak filter denoising step apply a Wiener filter to the wavelet coefficients, where the parameters of the Wiener filter are estimated locally.
The Wavelet Transform exploits the sparsifying property. It concentrates the details, edges and structure of an image on some few large coefficients making the noise appear only as many small coefficients. Its core functioning iteratively applies low/high pass filtering and down-sampling to the image. The purpose is to obtain a multi-resolution version of the image since at each iteration it halves the spatial resolution while doubling the frequency resolution.
The output is the noiseless image I_0, from which the residual noise W is evaluated. The formula for optimal denoising is based on the principle of the MSE(minimum square error) estimator, hence, we expect to suppress many periodic and non-periodic artifacts.
In the demodulation step the noise residual W is processed to estimate the PRNU $\hat{K}$.
If we filter the coefficients of the Wavelet Transform with a threshold cutoff value to vary depending on the image statistics, we can take out most of the noise

zeroing low coefficients, that contain the noise components, and leaving only the one representing the image details [18].

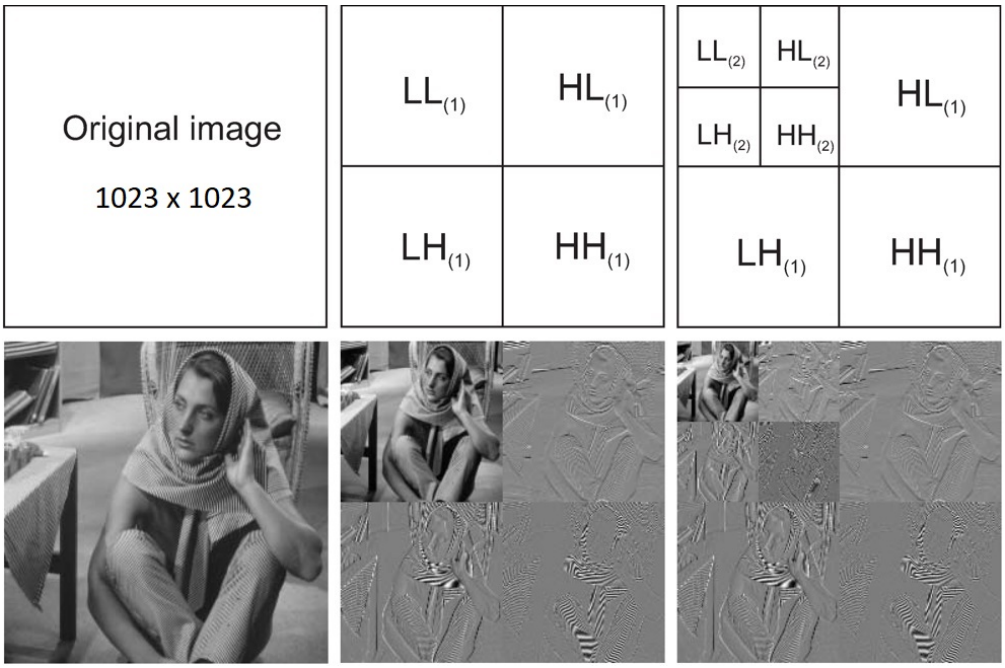

Figure 2.7: Wavelet decomposition

The Mihcak denoising filter is implemented with a 4 level Wavelet decomposition using the Daubechies 8 tap Wavelet filter [2]. Each wavelet coefficient is modeled as a random variable with a Generalized Gaussian (GG) distribution having unknown parameters. Parameter estimation, as the variance of the white additive Gaussian noise to remove, is usually carried out by context modeling [19]. It starts by estimating the underlying variance field using a Maximum Likelihood (ML) rule and then applying the Minimum Mean Squared error (MMSE) estimation procedure. The effect of edges is captured with use an adaptive window-based estimation procedure.

The image content is quite cancelled in the noise residual, therefore we can better estimate the PRNU from W respect than from I. This procedure improves the signal-to-noise ratio SNR between the signal of interest I_0K_0 and the observed data I. The energy of the PRNU signal $I\,K$ is small compared to the noise term R, this is the reason why we can assume R_D= R plus the terms introduced by the denoising filter, is unrelated to $I\,K$. R_D is non-stationary and has variance way bigger in textured areas.

The maximum likelihood estimator is used to compute an estimation of the

fingerprint $\hat{K}$, considering the dimension $|Intra|$ of the images taken from the same camera, namely the Intra-class, as in the equation 2.28:

$$\hat{K} = \frac{\sum_{k=1}^{|Intra|} I_k W_k}{\sum_{k=1}^{|Intra|} I_k^2} \tag{2.28}$$

The estimated factor $\hat{K}$ contains components that are systematically present in every image, such as some weak artifacts of color interpolation, onsensor signal transfer and sensor design [17].
Since we work with a linear model, the ML estimator is minimum variance unbiased (MVU). Given N the number of images, the Cramer–Rao lower bound (CRLB) determines the variance of the estimation $\hat{K}$ as:

$$var(\hat{K}) \sim \frac{1}{N} + Q \tag{2.29}$$

For smooth (low $var(\hat{K})$,), bright and not saturated images, should approximate very well the variance of the sum of image-acquisition noise sources R and the terms $I_0 - \hat{I}_0$ introduced by denoising.
The estimated factor contains weak artifacts of color interpolation and blockiness artifacts (depending on the strength of the JPEG compression) that are shared among cameras using the same imaging sensor design.
The PRNU of two different cameras is usually slightly correlated. Thus, the hypothesis of inter-correlation= 0 between groups is not verified, we expect an increase of the false positives rate and a decrease of the reliability of camera identification.

In the case of color images, the estimation must be performed separately on each matrix: the green, red and blue planes. Green noise residual components W_G is the most robust color to JPEG compression[3]. The green and red colors contribute for only half of the surface of the image, in a typical Bayer pattern.
A global grayscale PRNU fingerprint is obtained applying the RGB–to–gray conversion, giving different weights to each color matrix.

$$\hat{K} = 0.2989\hat{K_R} + 0.5870\hat{K_G} + 0.1140\hat{K_B} \tag{2.30}$$

In the notation up to now, $\hat{K}$ refers to any one of the tree colors considered. From this step on, $\hat{K}$ represents the gray-like PRNU fingerprint.

The estimated PRNU factor has to be further refined in the final step in preprocessing by converting it into the Fourier domain. The noise components are extracted using a zero-mean filter and Wiener filter operator W_f in the DFT domain to suppress the periodic and non-periodic artifacts not unique to the camera sensor

[1], caused by color interpolation of demosaicing algorithms and standard JPEG compression. Visually identifiable patterns are removed and the resulting PRNU has a flatter frequency spectrum [17].

For real images, with a complex non smooth content, we should make use of a more refined formula for the noise residual W. Equation 2.27 has to be re-written as:

$$W = TI\hat{K} + C_G \tag{2.31}$$

where T is a pixel-wise multiplicative attenuation factor and C_G is a colored Gaussian noise, a sequence of independent Gaussian variables with unequal variances. Though for simplicity, we rely on the formula 2.27 for computing the noise residual of the images in our dataset.

2.4 Clustering

The camera fingerprint is estimated from an image by subtracting the denoised image from the original image. With this technique, we can reliably recover cluster memberships even if the level of noise is higher than level of the signal. By averaging multiple images of the same camera, random noise components can be reduced, improving the readability of its pattern noise. Adding new images, the cluster becomes more and more united and defined around its own center, the centroid of the group.
A cluster SET is defined as the set of N data vectors X=x_1, x_2, x_N, divided in a partition P=$\{p_1, p_2, p_N\}$, such that they belong to the same partition p':

$$SET = \{x_{ii} | p_{ii} = p'\}. \tag{2.32}$$

Usually the output of a clustering phase is the list of clusters C_G and the number of groups $G = |C_G|$.

The clustering procedure can be done in many ways. In our work we have always to consider the trade-off between computational cost and performances of our algorithms. In our work, we propose blind techniques that does not require any reference information, parameter or bound imposed by humans. Every detail, like the number of final clusters we get in the output, is computed by the program itself.

Clustering can be treated as a graph partitioning challenge, an NP-complete combinatorial optimization problem [20]. It is based just on the linear dependencies

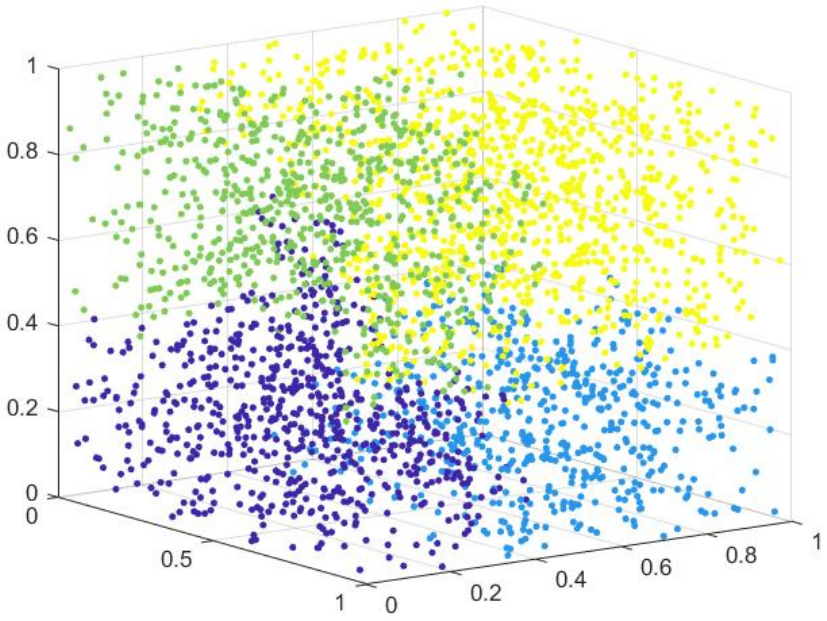

Figure 2.8: example of a 3-dimensional clustering

among Sensor Pattern Noises. [10]. For this computation we can store the full cross-correlation between every image of the dataset (section 2.4.2). This is the main phase of the Full matrix approach, in figure 4.1, but it requires a huge cost of computation.
Instead, we want to adopt greedy techniques, which provide slightly sub-optimal solutions but in a much shorter time. The implementation of those alternatives is the focus of this thesis, as shown in figure 4.1.

A big challenge we encountered is referred to as the $NC \gg SC$ problem. It comes from the scenarios in which the dataset presents a number of cameras (Number of Classes NC) larger than the average dimension of the set per each camera (Size of Class SC). The entire dataset might be underrepresented by the training set, so that means limiting the scalability of our approach.

2.4.1 Curse of dimensionality

The downside of non-parametric methods is that their flexibility can increase computational complexity. They often depend on some definition of distance in the

data space that can impact on the results. This phenomenon is known as "curse of dimensionality". If we have a collection of stochastic variables, the differences in each dimension will be random, because squared Euclidean distances are just sums over dimensions. The central limit theorem ensure we will observe a distribution converging into a Gaussian. This leads to the distances becoming a way less distinct and meaningful metric in higher dimensions.[21] In our experiments the problem of the curse of dimensionality does not arise: we work with full fingerprint extracted from big crops. A crop of 1023 pixels means that we have to deal with 1023^2 dimensions. However, we exploit the curse of dimensionality at our advantage. we do not have to learn from a dataset how to distinguish the fingerprints (for which we would need thousands of examples), but we already have a reasonable supervised model that explains how the correlation between fingerprints behaves. Increasing the crop dimension, the distance between matching and non-matching increases; we are more and more sure that two items are really related to each other and they are not simply stochastic variables with similar values.

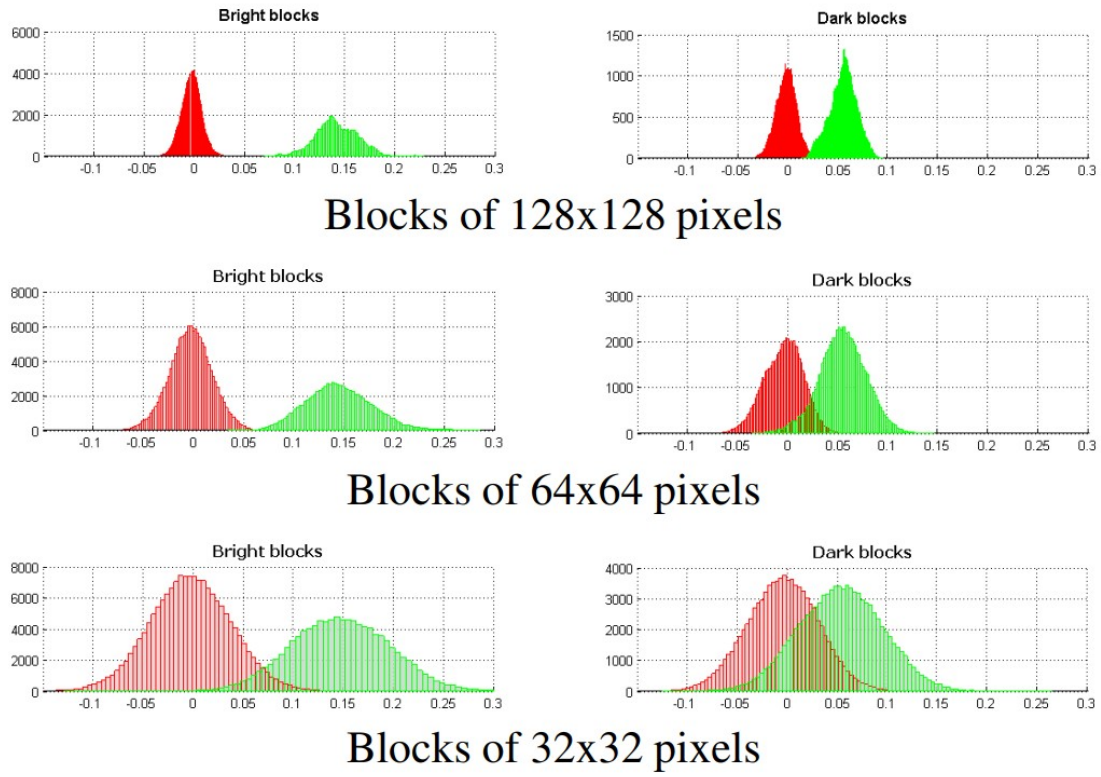

Figure 2.9: curse of dimensionality. Histograms of correlation for bright and dark blocks (crops) of images.

Our aim is to design an algorithm able to improve the metrics of the groups,

respect to a given ground truth of the data. The algorithm must also keep the precision P very stable and ideally always at 100%, without introducing false positives, even in presence of the $NC \gg SC$ problem.

2.4.2 Cross-correlation matrix

The additive noise N limits the reliability of traditional similarity measures used in conventional clustering algorithms. To overcome this problem, we applied the cosine similarity to measure the correlation between two normalized flattened fingerprints, as in the formula 2.33.
We compute the cosine of the angle between two vectors as the inner product between them over their magnitude; in other words it is the dot product of the vectors once they are both normalized. In particular, in the field of data mining, this formula is used to measure cohesion within clusters, where each term we are comparing is assigned to a different dimension and the cosine similarity returns how similar two items are likely to be, in terms of their content. The value is bound between -1 (complete opposite correlation) and +1 (fully identical content), where low module values indicate large angles and uncertainty to classify the vectors as related.
Given two vectors V_1 and V_2 the cross-correlation is expressed by:

$$C(V_1,\, V_2) = \left(\frac{V_1 \cdot V_2}{||V_1||\,||V_2||} \right) = \frac{\sum V_1(ii)\, V_2(ii)}{\sqrt{\sum V_1(ii)^2}\sqrt{\sum V_2(ii)^2}} \tag{2.33}$$

Pairs of fingerprints with cosine similarity, above a predefined threshold value Th are grouped together and they are considered taken from the same sensor; the corresponding images are seen as captured with the same camera, as in formula 2.33. By definition, the diagonal of the matrix represents the auto-correlation. It has all entries equal to one, so it is not considered by the clustering algorithms.
The cosine similarity between every couple of camera fingerprint is computationally very expensive, especially when the number and size of images become very large [12],

$$\mathcal{O}\left(\frac{n * (n-1)}{2} \right) correlations \tag{2.34}$$

where n= number of fingerprints examinated.
This step serves as a preliminary phase for many of the proposed algorithms.

After the preliminary step, the clustering phase aims at obtaining a first conservative grouping of residuals, where the merging of unrelated residuals has very low probability and would alter a lot our results. Each cluster is represented by its centroid, computed by averaging all the fingerprints in that group. Larger clusters

are constructed by merging the groups that satisfy the threshold criteria.

As we discussed in section 2.2, the main idea of the clustering phase is the estimation of a sensor K, in order to distinguish between two hypotheses:

- H_0: $\mathrm{W} = \mathrm{I}\,K_0 + R_D$ where $K_0 \neq \mathrm{K}$ (image not acquired by that sensor)
- H_1: $\mathrm{W} = \mathrm{I}\,\mathrm{K} + R_D$ where $K_0 = \mathrm{K}$ (image acquired by the same sensor)

Pure clusters are obtained by localizing common neighborhood of points. Large values of correlation indicate dense region of the closest fingerprints, that we can consider as part of the same cluster. This step is not very expensive because we consider only one group at time, comparing its correlation with the unclustered fingerprints, all the groups already done are no more considered by the algorithm; moreover, it works without applying spectral clustering to the data.

2.4.3 Divide and conquer split

Algorithms can be generalized to the case of large-scale databases[1]. Therefore, we need to work in a divide-and-conquer strategy. We can load in memory only part of the dataset and proceed by grouping sub-clusters based on the binary correlation matrix. Each sub-cluster can be represented by its centroid, computed by averaging all the fingerprints in that sub-cluster and successively merged with the others.

2.5 Image clustering algorithms

2.5.1 k-means algorithm

Lloyd's k-means algorithm impose the centroid of the cluster as its center, by considering the first k neighbors of each group and computing a hierarchical binary tree (as in figure 2.10) to subdivide the points. Given any set of centers Z, and z=|Z|, for each center $z \in Z$, let V(z) denote its neighborhood, that is, the set of data points for which z is the nearest neighbor. Those points are distributed in the Voronoi cell of z, as we can check from any graphical representation.
Each cycle of Lloyd's algorithm shifts every center point z to the centroid of V(z) and then updates the distance from each point to its nearest center and the position of V(z). These steps are repeated until some convergence condition is met.[22] the algorithm will eventually converge to a point that is a local minimum for the distortion, but it is not guarantee it will also be a global minimum. A simple implementation of Lloyd's algorithm can be quite slow because it is very influenced

by the cost of computing the nearest neighbors. This is why it is useful to scale k-means clustering for large data sets through sampling and pruning. Given N points, the algorithm produces a tree with $\mathcal{O}(N)$ nodes and $\mathcal{O}(logN)$ depth. The overall running time is $\mathcal{O}(kN)$, which lay in the same order of a brute-force algorithm. k-means clustering algorithms requires the number of clusters in input, which is very hard to achieve in practical scenarios.

2.5.2 Full Matrix algorithm

Full Matrix [4] is a deterministic algorithm able to merge together the best couple of fingerprints in the dataset at each step.
We can apply an exhaustive hierarchical clustering in which we compute the cross correlations of all the fingerprints, as in the formula equation 2.33.
After this pre-clustering step, we treat the fingerprints as references of their unclustered groups. Therefore, the algorithm starts by considering G groups, where G is the number of fingerprints in the dataset and it computes the full $G \times G$ cross-correlation matrix. In this way, we exploit an exhaustive knowledge of the whole set, therefore the method is very precise but extremely expensive in terms of computational complexity in its pre-clustering step.

The cluster membership is obtained iteratively by finding at each step of a while loop the pair of nearest fingerprints F_{ii} and F_{jj} that has, by definition, the same indices of the maximum cross correlation. This step is explained in detail in the algorithm 1.
We merge the union of those groups and we store the reference fingerprint in the matrix instead of their previous versions F_{ii} and F_{jj}. In practice, we update the cluster C_{ii}, while we get rid of C_{jj} and we decrease G by one, until we exhaust all what we can attract.
The first cycles represent a very delicate and critical phase. From a forest of unclustered fingerprints we have to store the union of the best ones and keep on merging the little groups between them and with new items. When we deal with little groups, we have few information about their central reference and we find harder to get a precise esteem of their centroids.
The computational complexity of the clustering stage is upper bounded by $\mathcal{O}(G)$, where G= dimension of the matrix before clustering. We can assign to the overall method the complexity of creating the full matrix in its first step, in equation 2.33, because the clustering procedure alone introduces a negligible amount of computations.

Algorithm 1 Full matrix algorithm
Input: Th: threshold, corr: normalized correlation matrix, C_G: unclustered fingerprints.
Output: G: number of groups, C_G: updated clusters.

procedure FM_base(Th, corr, C_G)
  $[C, ii, jj] = max(corr)$ ▷ max in the matrix and its indices
  $len\_Gr = |C_G|$
  while $C > Th$ **do** ▷ there still exist a corr value greater than Th
    $C_G(ii) \leftarrow C_G(jj)$ ▷ $Group_{jj}$ absorbed by $Group_{ii}$
    $C_G(jj) = \emptyset$
    $len\_Gr(ii) = len\_Gr(ii) + len\_Gr(jj)$
    $corr(ii, :) = corr(:, ii)$ ▷ corr matrix is symmetrical
    $corr(ii, ii) = Null$ ▷ value out of bounds, never considered again
    $corr(jj, :) = corr(:, jj) = Null$
    $G = G - 1$ ▷ groups and matrix dimension reduced by one
    $[C, ii, jj] = max(corr)$ ▷ max in the matrix and its indices
  end while
  $list \leftarrow C_G$
end procedure

2.5.3 Hierarchical Agglomerative clustering

Agglomerate clustering [4] builds the binary tree of distances between all the centroids. It output a diagram, the dendrogram, that regroups our data in a deterministic tree graph, where the leaves represent the original fingerprints and the y-axis shows the height at which a couple is regrouped into one set, as in the example shown in figure 2.10. The tree starts at the bottom, considering every fingerprint as its own singleton cluster, and merges groups together computing the pairwise similarity matrix, until we are left at the root with is a single cluster that contains all the data.
The algorithm maintains an active set of clusters and at each loop decides which two clusters should merge. When two clusters are merged, the similarity matrix is updated by replacing the corresponding two rows with their union and the new group is added to the tree, with a label to the height they were regrouped. After the update, a silhouette coefficient is calculated for each fingerprint. The silhouette coefficients measure the cohesion within each cluster and the separation among clusters. An average of silhouette coefficients gives an esteem of the global aptness of the current partition. When all fingerprints have been merged into one cluster, the partition corresponding to the highest suitability is interpreted as the optimal partition.

A feasible clustering comes from a legit dendrogram, where the group distance vector is monotonically increasing by construction.

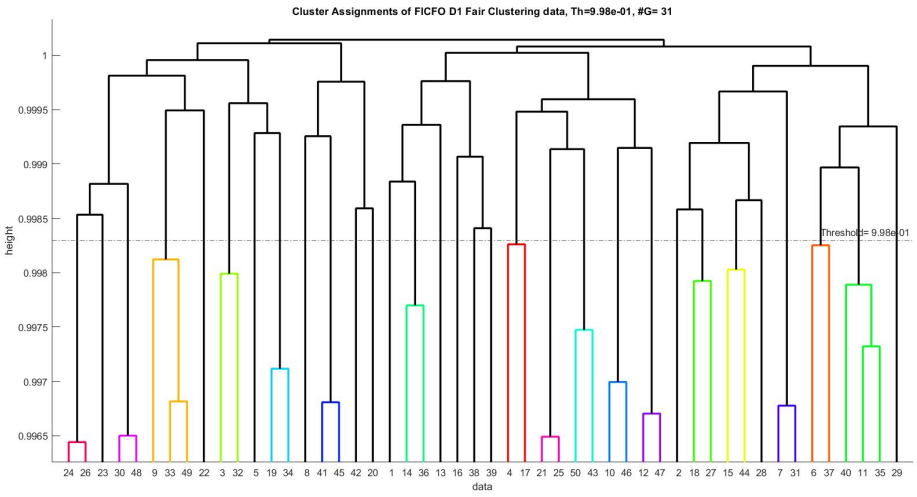

Figure 2.10: Dendogram example

2.5.4 Graclus graph

The Graclus graph partitioning algorithm [23] is able create coarse clusters. Graclus is a fast graph clustering software that computes normalized cut and ratio association from an undirected graph, already implemented as a Matlab extension in: `https://www.cs.utexas.edu/users/dml/Software/graclus.html` It can run a kernel k-means type of iterative algorithm to minimize general cut or association objectives and it does not require any eigenvector computation.

Chapter 3

State of the art methods for PRNU clustering

In this chapter we review the most important approaches in the literature of image clustering algorithms, based on camera fingerprints. The algorithms need to work in a blind supervised scenario, without references to the source cameras present in the datasets but with the possibility to tune some parameters from a testbench dataset.
Here we discuss the advantages and the problems of these algorithms, particularly their computational complexity and their robustness against the $NC \gg SC$ problem.

For all the method presented in this chapter, we assume that the input is a set of camera fingerprints having the same size, each normalized to have zero mean and unit variance. In the following algorithm pseudo-codes, the variable Set represents the group of each standardized fingerprint F, for every image in the dataset under exam. We obtain the initial set of fingerprints Set from the dataset of images I as given in the equation 3.1.
Given an image I in the dataset, we can compute F, the relative camera fingerprint, as the standardization (Z_score) of the Wiener filter operator W_f applied on the PRNU of the image [2].

$$F = Z\_score\left(W_f\left(\frac{I\,W}{I^2}\right)\right) \tag{3.1}$$

Normalization is an optional preliminary step that can be helpful to compare different values on the same scale and to return correlation values in the [-1,1] range.

3.1 Bloy algorithm

The pairwise nearest neighbor (PNN) method, also known as Ward's method belongs to the class of algorithms known as Agglomerative clustering [20]. PNN generates hierarchical clustering using a sequence of merge operations until the desired number of clusters is obtained. This method selects the cluster pair to be merged if the objective function value will benefit by this operation. The main drawback of the PNN method is its slowness, the complexity of the fastest known exact implementation of the PNN method is lower bounded by $\mathcal{O}(N^2)$, where N is the number of data objects.

G.J.Bloy in 2008 implemented the pairwise nearest neighbor (PNN) technique in a reduce complexity version [3]. Bloy algorithm is a very expensive well-known solution that we used mainly as a performance reference.

The process of Bloy algorithm, in $PHASE_1$ randomly searches for a pair with correlation value greater than a pre-calculated threshold, or it stops if it reaches a bound number of attempts N_{attMAX}, declaring the fingerprints as unclustered. The pairs of fingerprints are merged form a fingerprint. They became part of the same group and the centroid is computed by averaging them.
Then in $PHASE_2$, Bloy performs the first pass on the dataset. It compares the reference fingerprints and the remaining fingerprints, working with the new groups instead of the full dataset as it proceed, until all images have been tried or it reaches an average of n= 50 images for computing each reference.
This fixed value n is clearly non robustness against the $NC \gg SC$ problem.
The $PHASE_3$ loops over all the unclustered images a second time, but allow more than n= 50 images to be associated with the fingerprint without updating the centroid.
The groups and the unclustered fingerprints undergo the clustering process again and the three phases are repeated in a loop until $PHASE_1$ has tried enough pairs without success, reaching a bound number of groups G_{MAX}, or until every fingerprint is regrouped. The algorithm has to be supervised by inputting the parameters N_{attMAX}, G_{MAX}, n and a value for the threshold Th.
A pseudo-code for Bloy algorithm is presented in algorithm 2.

The time complexity can be optimized by checking the value of a stop_flag at every cycle, in order to "prune" useless loops, but the computational complexity remains bound by $\mathcal{O}(N^2)$

Algorithm 2 Bloy Algorithm

Input: Set: fingerprints, Th: threshold, d: dimension of the cropped fingerprints, N_{attMAX}: max number of attempts, G_{MAX}: max number of groups, n: max size for thresholds. Output: G: number of groups, C_G: clusters, K_G: References.

procedure BLOY(Set, Th, d, n)
  $G = 1, N_{att} = 0, stop\_flag = 0$
  $C_G = \emptyset$
  while $(N \neq 0)$ **do** ▷ we still have items to assign to groups
    $runPHASE_1$ ▷ creates the first couples from the attractors
    $runPHASE_2$ ▷ attracts new F_{ii} in group C_G, compute K_{ii}
    $runPHASE_3$ ▷ scrolls Uncl for others $F_{ii} \in C_G$, with a fixed K_{ii}
    $Set = Uncl$ ▷ fingerprints still in exam
    $N = |Set|$ ▷ N= number of fingerprints
    $G = G + 1$
    if $G \geq G_{MAX}$ or $N_{att} \geq N_{attMAX}$ **then** ▷ returns if it procured
      $stop\_flag = 1$ ▷ too many groups or if is looping too many times
    end if
  end while
end procedure

Bloy Algorithm- PHASE 1

procedure PHASE$_1$
  $N = |Set|$ ▷ N= number of fingerprints
  for $ii = 1$ to $N - 1$ and stop_flag== 0 **do**
    $F_{ii} = load(Set(ii))$ ▷ Read zero mean full camera fingerprint
    $C_G \leftarrow F_{ii}$
    for $jj = ii + 1$ to N and stop_flag== 0 **do**
      $F_{jj} = load(Set(jj))$ ▷ Read zero mean full camera fingerprint
      $C(jj) = \frac{1}{d} \sum_{x=1}^{d} F_{ii}[x] F_{jj}[x]$ ▷ cross correlation
      if $C(jj) \geq Th$ **then**
        $C_G \leftarrow F_{jj}$ ▷ new item included in the group
        $K_G = \frac{F_{ii} + F_{jj}}{2}$
        $norm\_K = \frac{\sum K_G}{|K_G|}$
        $K_G = \frac{K_G}{norm\_K}$
        $stop\_flag = 1$
      end if
    end for
  end for
end procedure

Bloy Algorithm- PHASE 2

procedure PHASE$_2$
  if $K_G \neq 0$, $|K_G| \leq n$ **then** ▷ n is set to 50 images
    $Uncl = \emptyset$
    for $ii = 1$ to N **do**
      $F_{ii} = load(Set(ii))$ ▷ Read zero mean full camera fingerprint
      $C(ii) = \frac{1}{d}\sum_{x=1}^{d} F_{ii}[x]K_G[x]$ ▷ cross correlation
      if $C(ii) \geq Th$ **then**
        $|K_G| = |K_G| + 1$
        $K_G = \frac{K_G + F_{ii}}{|K_G|}$
      else
        $Uncl \leftarrow F_{ii}$
      end if
    end for
  end if
end procedure

Bloy Algorithm- PHASE 3

procedure PHASE$_3$
  for $ii = 1$ to $|Uncl|$ **do**
    $F_{ii} = load(Set(ii))$ ▷ Read zero mean full camera fingerprint
    $C(ii) = \frac{1}{d}\sum_{x=1}^{d} F_{ii}[x]K_G[x]$ ▷ cross correlation
    if $C(ii) \geq Th$ **then**
      $|K_G| = |K_G| + 1$
      $K_G = \frac{K_G + F_{ii}}{|K_G|}$
      $Uncl =$ remove F_{ii}
    end if
  end for
end procedure

3.2 RCIC algorithm

We focused on Low Complexity Clustering Algorithms such as RCIC (Reduced Complexity Image Clustering) and FICFO (Fast Image Clustering based on Fingerprints Ordering) and we used them as a base for our methods.
FICFO[1] and RCIC[1] work in a blind scenario, facing the $NC \gg SC$ problem without any information or bound regarding the candidate images, the source cameras, the number of cameras and the number of images per camera.
The total complexity of the regrouping phase is measured in terms of the number

of NCC operations performed during the clustering, as in the equation 2.33. Both FICFO and RCIC are able to exploit low complexity and fast computation cost. Because of the few comparisons they need to do, they load in RAM memory a single reference fingerprint (cluster centroid) and a full fingerprint (from the image) at a time.
The I/O cost is quite high because for each group, all the images still unclustered have to be read again in order to be tested for the current group.

In RCIC algorithm, a normalized fingerprint Imi is randomly selected from the dataset as the reference fingerprint of a new group and it is moved to the cluster G. The clustering is done iteratively by calculating the normalized cross-correlation C between the reference fingerprint and the items still present in the dataset.
If the generic fingerprint Imi shows a correlation value greater than a threshold Th, it is assigned to the cluster G; otherwise, it is left unclustered. The probability of regrouping into the wrong cluster G a fingerprint from a different camera is bounded by P_{FA}, that impose the value of Th as in the formula 2.11.
A pseudo-code for RCIC algorithm is presented in algorithm 3.

After processing all fingerprints a total of $NCC - 1$ correlation operations are performed to construct cluster G. The fingerprints grouped in cluster G are removed from the dataset and we are left with $NCC - G$ unclustered fingerprints. A new cluster $G + 1$ is initiated and a fingerprint Imi is randomly selected from the dataset a reference fingerprint. The unclustered fingerprints are processed by repeating the same procedure used for constructing the first cluster G. The algorithm stops when all fingerprints are assigned to a cluster or they constitute a cluster by themselves.
Each time a new fingerprint is included in a cluster, the corresponding reference is updated by adding to itself the fingerprint and dividing the result by two.

$$K_G = \frac{K_G + F_{jj}}{2} \tag{3.2}$$

This simple formula does not take into account the length of the groups, so it gives the new attracted fingerprint half of the importance of the total. This procedure is usually dangerous in presence of false positives that drag the centroid very far from its real position. Moreover, it divides by two the weight of the already present fingerprints. In particular, the first fingerprint is an attractor and a good approximation of the centroid. For a group containing G items, the first fingerprint only contributes as $\frac{1}{2^G}$ to the solution.

The algorithm never checks again the clusters already formed, this is the task performed by an optional attraction step that may further merge the groups.

The attraction always suffers the same problem of RCIC in its base form, the randomness of the data, that impose us to consider the variance in which our results lay in. Every output is a stochastic instance of a run of the algorithm, instead of a deterministic representation of the clusters.

Algorithm 3 RCIC Algorithm
Input: Set: Fingerprints, Th: Threshold, d: dimension of cropped fingerprints.
Output: G: number of groups, C_G: Clusters, K_G: Reference fingerprints.

procedure RCIC(Set, Th, d)
  $G = 1$
  $Uncl = randomize(Set)$ ▷ unclustered set, full of randomized fingerprints
  $N = |Uncl|$ ▷ N= number of fingerprints
  while $(N \neq 0)$ **do** ▷ we still have items to assign to groups
    $Unc_{k+1} = \emptyset$
    $C_G = \emptyset$
    $F_1 = load(Set(1))$ ▷ Read zero mean full camera fingerprint
    $K_G = F_1$
    $C_G \leftarrow F_1$ ▷ attractor included in the group
    for $ii = 2$ to N **do**
      $F_{ii} = load(Set(ii))$ ▷ Read zero mean full camera fingerprint
      $C(ii) = \frac{1}{d}\sum_{x=1}^{d} K_G[x]F_{ii}[x]$ ▷ cross correlation
      if $C(ii) \geq Th$ **then**
        $C_G \leftarrow F_{ii}$ ▷ new item included in the group
        $K_G = \frac{K_G + F_{ii}}{2}$
        $norm\_K = \frac{\sum K_G}{|K_G|}$
        $K_G = \frac{K_G}{norm\_K}$
      else
        $Uncl \leftarrow F_{ii}$ ▷ item included in $Uncl$ for the next cycle
      end if
    end for
    $Uncl = randomize(Uncl)$
    $G = G + 1$
    $N = |Uncl|$
  end while
end procedure

3.3 FICFO algorithm

The fast image clustering based on the fingerprint ordering (FICFO) algorithm select the best fingerprints with lower estimation error as reference fingerprints. Those fingerprints are the closest to the respective centroids, so we can already assign them as the first reference of each group.
The ordering factor depends on the quality of the fingerprints and it can be referred as the ranking index RI, as expressed in the equation 3.3. As a first step of FICFO, we compute a ranking index RI depending on the image content, which tells how reliably a fingerprint can be estimated from that image. One of the best example we can provide of real image with very high RI are flat image uniformly bright and not saturated, such as out of focus cloudy skies [17].
The RI parameter reduces computational complexity, because the good images are already regrouped in the first positions of their groups, based on the assumption that fingerprints with lower estimation errors are the closest to their centroids.
The FICFO algorithm proceeds by checking the other images in the dataset. It correctly clusters them and, if possible, it merges some groups that have very high correlation.[2]
FICFO instead of randomly appending images to the existing groups, as RCIC does, it sorts them in descending order, using as sorting key the computation of the Rand Index RI of each fingerprint.

$$RI_{ii} = Gl_{ii}^{1/\alpha} + (1 - S_{ii})^{1/\beta} + (1 - T_{ii})^{1/\gamma} \tag{3.3}$$

with Gl as the average gray level, S as the saturation level, T as the texture; $\alpha > 1$, $\beta > 0$ and $\gamma > 0$ are the factors defining the contribution of Gl, S and T respectively. RI increases with an increase of α and γ, while it is inversely proportional to β. FICFO works in a supervised way, where α, β and γ have to be carefully chosen for the images we want to process.

The cluster reference is computed with the formula 3.2, as in the RCIC code. RI is used by the clustering phase as an index of purity of the picture, the ordering factor that shows the reliability of using the reference as an attractor for other images.
Therefore, the main difference between FICFO and RCIC is the introduction of the parameter RI. It adds an initial computation cost in the fingerprint pre-processing phase with a negligible complexity, but it guarantees deterministic and reproducible results in the output. The data are easier to compare with other methods and we never have to run again the algorithm to provide new estimates of the final clusters.

FICFO, as RCIC, never backtrack over the clusters already computed, this task can be performed by an optional attraction step that may further merge the groups.

A pseudo-code for FICFO algorithm is presented in algorithm 4.

Algorithm 4 FICFO Algorithm
Input: Set: Fingerprints, RI: Ranking Indexes, Th: Threshold, d: dimension of cropped fingerprints.
Output: G: number of groups, C_G: Clusters, K_G: Reference fingerprints.

procedure FICFO(Set, RI, Th, d)
  $G = 1$
  $Uncl = sort(Set, RI)$ ▷ unclustered set, full of sorted fingerprints
  $N = |Uncl|$ ▷ N= number of fingerprints
  while $(N \neq 0)$ **do** ▷ we still have items to assign to groups
    $Unc_{k+1} = \emptyset$
    $C_G = \emptyset$
    $F_1 = load(Set(1))$ ▷ Read zero mean full camera fingerprint
    $K_G = F_1$
    $C_G \leftarrow F_1$ ▷ attractor included in the group
    for $ii = 2$ to N **do**
      $F_{ii} = load(Set(ii))$ ▷ Read zero mean full camera fingerprint
      $C(ii) = \frac{1}{d}\sum_{x=1}^{d} K_G[x]F_{ii}[x]$ ▷ cross correlation
      if $C(ii) \geq Th$ **then**
        $C_G \leftarrow F_{ii}$ ▷ new item included in the group
        $K_G = \frac{K_G + F_{ii}}{2}$
        $norm\_K = \frac{\sum K_G}{|K_G|}$
        $K_G = \frac{K_G}{norm\_K}$
      else
        $Uncl \leftarrow F_{ii}$ ▷ item included in $Uncl$ for the next cycle
      end if
    end for
    $G = G + 1$
    $N = |Uncl|$
  end while
end procedure

3.4 Khan attraction

The merging phase of Khan clustering [1] is conducted as an iterative procedure, as in the pseudo-code of algorithm 5, which respectively selects pairs of groups having maximum correlation between their centroids and compares them with the

threshold Th. The algorithm stops when no more pairs of groups exist that satisfy the merging condition.

The centroid K_G for every group is computed as

$$K_G = \frac{\sum F_{ii} \in C_G}{|F_{ii}|} \tag{3.4}$$

Algorithm 5 Khan Attraction Algorithm
Input: K_G: Reference fingerprints, K: previous number of groups, Th: Threshold, d: dimension of cropped fingerprints.
Output: G: number of groups, C_G: Clusters.

procedure *Khan_attraction*(K_G, K, Th, d)
  $G = 0$
  for $ii = 1$ to K **do**
    $G = G + 1$
    for $jj = 1$ to k and $ii \neq jj$ **do**
      $C(j) = \frac{1}{d} \sum_{x=1}^{d} K_{ii}[x] K_{jj}[x]$ ▷ cross correlation
      if $C(j) \geq Th$ **then**
        $C_{ii} \leftarrow C_{jj}$ ▷ $Group_{jj}$ absorbed by $Group_{ii}$
      end if
      $ncc = ncc + 1$
    end for
  end for
end procedure

The computational complexity is: $\mathcal{O}(K^2)$ where K is the number of groups in the first attraction.

Chapter 4

Proposed methods

We implemented different clustering techniques and attraction phases in order to increase as much as we could the performances of the State of the art algorithms mentioned above. Our thesis presents an ablation study of the best methods proposed, where some parameters are tested and removed one by one, in order to understand their contribution to the performances. In the final results we will show the best attraction steps applied to the best clustering technique, in accordance with the metric we used.

The clusters produced by FICFO and RCIC, indeed, can be further refined by using an attraction stage. The main idea is to consider each group as a single fingerprint, its reference, and to proceed with a second clustering phase that conglomerate groups of groups. Usually, FICFO and RCIC output many little clusters and some unclustered outliers.
The attraction phase should be able to reduce the number of groups, merging items that the previous step considered different. In the attraction phase, we work with the mean of the groups, so many noise components ideally average to zero. These versions of the codes that make use of the base algorithm plus the attraction are called FICFO-A and RCIC-A.
We propose multiple methods of attraction and all of them are compatible with both of our algorithm RCIC and FICFO. Because of their dependence on the base version we start from, the final clusters will be stochastic or deterministic as the respective input data.

It is very important to notice that groups are not checked internally. Whichever error occurred in the first phase, whatever false positive they contain, it remains inside of them, and may lead to an incorrect assumption of combining it with another group. The distance between groups is an optimum merging parameter only for pure groups, while for corrupted clusters, which compute wrong centroids,

it makes us to combine unrelated groups, dropping the precision of our results. Therefore, we can introduce an attraction method able to check the groups, by re-assigning the fingerprints to the clusters, or we rely on the strong assumption that the groups are already very pure as they are, when computed by FICFO and RCIC.

For the clustering phase we compare the following methods:
• Full Matrix [4]: this is based on computing a matrix of distances between every possible couple of fingerprints, for a complete knowledge of the problem, followed by a hierarchical clustering.
• First reference: chooses the first fingerprint in the list of currently unclustered fingerprints as reference for attracting other fingerprints, without updating it.
• Fair: updates the reference fingerprint in the above scheme by computing a fair average of the currently attracted fingerprint.
• Weighted: updates the reference fingerprint by computing a weighted average of the currently attracted fingerprint, where the weights are proportionally to the ordering factor RI.

After this first step, we are free to consider the obtained partition already as the final image clustering we were searching for, or we can apply an optional attraction phase, aiming for a better clustering effectiveness, chosen between the following methods:
• Full Matrix attraction: the Full Matrix method is applied to a preexisting

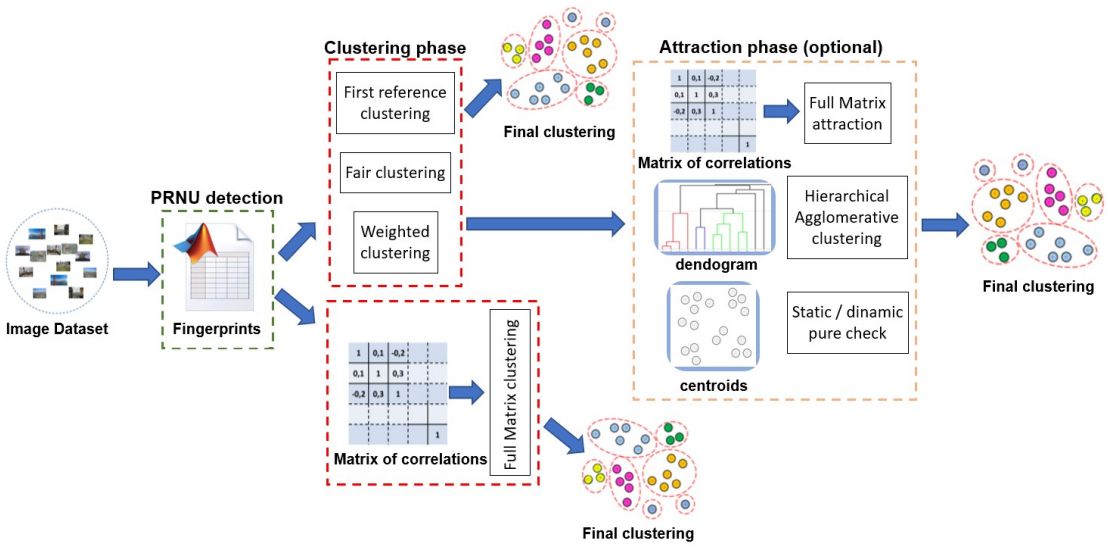

Figure 4.1: Procedure steps

partition.
• Hierarchical Agglomerative [4]: builds the binary tree of distances between all the centroids. At a pre-computed height, it splits the tree into a cluster partition.
• Static/Dynamic Pure attraction: refines the clusters internally, moving some fingerprints to the correct cluster they should lie in, according to a k-means clustering strategy.

4.1 Clustering phase

4.1.1 Full Matrix clustering with normalized threshold

This clustering technique has been obtained from the Full Matrix algorithm, presented in the literature review in section 2.5.2. The key of this modification is the loop while $C > Th$. The merge of fingerprint could end too soon because, as we proceed, we find ourselves in front of bigger and bigger groups. If the best of them has correlation lower than the threshold we exit the loop, but this does not automatically mean we could not attract other little groups. Here comes the importance to normalize the threshold to the length and the statistics of each possible merge.

The vector w_corr stores the squared norm of the fingerprints at each step of the algorithm. When we start from the initial fingerprints, w_corr is set equal to 1 for every unclustered fingerprint. As the merging process proceeds, it is updated according to the groups dimension weighted by their correlation goodness, namely the correlation with the new fingerprint that is joining the cluster.
Thanks to this factor, we can store the degree of proximity of the items in the groups. Given for instance only two fingerprints F_{ii} and F_{jj}, their cross-correlation is:

$$corr(ii, jj) = \langle F_{ii}, F_{jj} \rangle \tag{4.1}$$

The parameter w_corr, corresponding to the sum of the two fingerprints, can be obtained as:

$$w\_corr = \|F_{ii} + F_{jj}\|^2 = \|F_{ii}\|^2 + \|F_{jj}\|^2 + 2\, corr(ii, jj) \tag{4.2}$$

and it varies from $\sqrt{2}$ to 2! =2 ; where $\sqrt{2}$ is the weight factor for extremely far groups (with cross-correlation =0). and 2 represent identical items (with cross-correlation =1) The parameter w_{corr} tunes the values of the cross correlation matrix and influences the successive choices of the maximum(corr).

The parameter γ represents a correction factor that highlight the length of the groups, raising the threshold Th accordingly the the sum of the items in the groups we want to join, so that it is harder to merge bigger groups together. The cross correlation between the two references, normalized by the length matrix (elevated to the γ power), has to be bigger than the normalized threshold. The parameter γ can easily turned off if it is set to zero, therefore we can consider every version of the algorithm (algorithm 6) as a generalization of the γ =0 case. In the attraction phase is not valid anymore the hypothesis of w_corr= 1, for every fingerprint. We are working with clusters C_G, each of them with its own statistics.

Algorithm 6 Full matrix clustering with normalized threshold
Th: threshold, corr: normalized correlation matrix, γ: gamma parameter, C_G: unclustered fingerprints.
Output: G: number of groups, C_G: updated clusters.

procedure FM_norm(Th, corr, γ, C_G)
  $[C, ii, jj] = max(corr)$ ▷ max in the matrix and its indices
  $len\_Gr = |C_G|$
  $w\_corr = 1$
  while $C > Th$ **do** ▷ there still exist a norm(C) value greater than Th
    $corr\_coeff = C * \sqrt{w\_corr(ii)} * \sqrt{w\_corr(jj)}$
    $corr(:, ii) = \frac{\sqrt{w\_corr(ii)} * corr(:, ii) + \sqrt{w\_corr(jj)} * corr(:, jj)}{\sqrt{w\_corr(ii) + w\_corr(jj) + 2 * corr\_coeff}}$
    $w\_corr(ii) = w\_corr(ii) + w\_corr(jj) + 2 * corr\_coeff$
    $C_G(ii) \leftarrow C_G(jj)$
    $C_G(jj) = \emptyset$
    $len\_Gr(ii) = len\_Gr(ii) + len\_Gr(jj)$ ▷ $Group_{jj}$ absorbed by $Group_{ii}$
    $corr(ii, :) = corr(:, ii)$ ▷ corr matrix is symmetrical
    $corr(ii, ii) = Null$ ▷ value out of bounds, never considered again
    $corr(jj, :) = corr(:, jj) = Null$
    $G = G - 1$ ▷ groups and matrix dimension reduced by one
    $len\_matrix = |len\_Gr \times len\_Gr|^{\gamma}$
    $[C, ii, jj] = max(\frac{corr}{len\_matrix})$ ▷ max in norm matrix and its indices
  end while
end procedure

4.1.2 Full Matrix clustering with merge singleton

This other version of the Full Matrix algorithm, presented in algorithm 7, introduces another important parameter to the algorithm variant presented in section 4.1.1. It makes use of a binary vector that, thanks to its mask, splits the groups into the unclustered images with length= 1 and the bigger clusters. The vector forces the normalization introduced by len_matrix to be reset to 1, in case one of the two merged groups (or both) has length $|C_G| = 1$.

The matrix bin_matrix is computed as the external product between bin_vect and itself, where bin_vect is a binary mask of length. It appears to have benefits over all the datasets in the base version because if forces the normalization introduced by len_matrix to be reset to 1, in case one of the two merged groups (or both) has length $|C_G| = 1$. This solution introduces an important alteration in the way the algorithm deals with unclustered and little groups, improving the quantity and the correctness of the attractions done.

4.1.3 First reference clustering

The easiest implementation of a clustering algorithm relies on fixing the first fingerprint attracted by each group as the centroid. With this method, we give extreme importance to the attractor and we are not able to update the centroid to shift toward the real center of the final cluster.
In each step of the pseudo-code in algorithm 8, we consider every cluster we obtained up to now as an overclustering of little pure groups. We attract them as if they were single items, considering only the unclustered fingerprints and the centroids of each group, without having to access each single fingerprint of the dataset again. The result is a merge of only the fingerprints that have very low distance from the first one, so in the best scenario it creates a huge number of pure groups, while in many other realistic cases it also introduces some false positives.

4.1.4 Fair clustering

The centroid of each cluster can be iteratively computed as a fair mean of all the items in our clusters, increasing the number of items Av_fact by one each time we merge an unclustered fingerprint. With this strategy, we give the same weight to each fingerprint, moving the centroid toward the real center of the final cluster with a weight proportional to the number of fingerprint present in the cluster. In a huge group, the presence of an outlier produces a little variation and deviation of the centroid, corrected by all the true positives included in the cluster. Since PRNU estimation improves steadily with the dimension of the cluster, the biggest the group is, the strongest it is in producing an effective representation similar to the ground truth.

Algorithm 7 Full Matrix clustering with merge singleton
Input: Th: threshold, corr: cross correlation matrix, γ: gamma parameter, C_G: unclustered fingerprints.
Output: C_G: clusters.

procedure $FM\_singletone$(Th, corr, γ, C_G)
  $[C, ii, jj] = max(corr)$ ▷ max in the matrix and its indices
  $len\_Gr = |C_G|$
  $w\_corr = 1$
  while $C > Th$ **do** ▷ there still exist a norm(C) value greater than Th
    $corr\_coeff = C * \sqrt{w\_corr(ii)} * \sqrt{w\_corr(jj)}$
    $corr(:, ii) = \frac{\sqrt{w\_corr(ii)} * corr(:, ii) + \sqrt{w\_corr(jj)} * corr(:, jj)}{\sqrt{w\_corr(ii) + w\_corr(jj) + 2 * corr\_coeff}}$
    $w\_corr(ii) = w\_corr(ii) + w\_corr(jj) + 2 * corr\_coeff$
    $C_{ii} \leftarrow C_{jj}$
    $C_G(jj) = \emptyset$
    $len\_Gr(ii) = len\_Gr(ii) + len\_Gr(jj)$ ▷ $Group_{jj}$ absorbed by $Group_{ii}$
    $corr(ii, :) = corr(:, ii)$ ▷ corr matrix is symmetrical
    $corr(ii, ii) = Null$ ▷ value never considered again
    $corr(jj, :) = corr(:, jj) = Null$
    $G = G - 1$ ▷ groups and matrix dimension reduced by one
    $len\_matrix = |len\_Gr \times len\_Gr|^{\gamma}$
    $bin\_vect = len\_Gr > 1$ ▷ binary mask of length
    $bin\_matrix = bin\_vect \times bin\_vect'$ ▷ compute the mask
    $len\_matrix(not(bin\_matrix)) = 1$ ▷ force some lengths to 1
    $[C, ii, jj] = max\frac{corr}{len\_matrix}$ ▷ max in norm matrix and its indices
  end while
end procedure

The problem of this technique is the importance that false positive may have in attracting other wrong items in the cluster, especially when they are included in little groups. False positives may lead to the attraction of more and more outliers, adding random noise to the reference vector and destroying the performances of our algorithm.
By definition, the first fingerprint of our cluster is never a false positive and it is rare to encounter a wrong fingerprint in the first positions of a group, because it has to have a very high correlation and it has to lay in the very proximity of the centroid of a wrong group to be attracted before of the true positives.
A pseudo-code for FICFO Fair clustering is presented in algorithm 9.

Algorithm 8 First reference clustering
Input: d: dimension of cropped fingerprints, Th: threshold, Set: fingerprints.
Output: G: number of groups, C_G: Clusters, K_G: Reference fingerprints.

procedure $First\_ref\_clust$(d, Th, Set)
  $G = 1$
  while $(N \neq 0)$ **do** ▷ we still have items to assign to groups
    $F_1 = load(Set(1))$ ▷ Read zero mean full camera fingerprint
    $C_G(G) \leftarrow F_1$
    $K_G = F_1$ ▷ reference equal to the first image
    for $jj = 2$ to $|Set|$ **do**
      $F_{jj} = load(Set(jj))$ ▷ Read zero mean full camera fingerprint
      $C = \frac{1}{d}\sum_{x=1}^{d} K_G[x]F_{jj}[x]$ ▷ cross correlation
      if $C \geq Th$ **then**
        $C_G(G) \leftarrow F_{jj}$ ▷ new item included in the group
        Set remove F_{jj}
      end if
    end for
    $N = N - 1$
  end while
end procedure

4.1.5 Weighted clustering

The Fair clustering main strength is the mean of fingerprints that gradually gives less importance to the new items, without fully excluding the new entries as in the First reference technique. However, the algorithm has no way to customize the shift of the centroid, depending on the importance of the new attracted fingerprint and the confidence that we are in presence of a true positive.
Another proposed solution is to consider a weighted average that considers more reliable the fingerprints of flat image uniformly bright and not saturated, the ones easier to cluster in the right group. Those images can give us a better estimation of the correct position of the centroid but, because we work in a $NC \gg SC$ scenario, we do not have enough images to settle for only the best of them, so we need to retrieve useful information also from the other images attracted in the group, with a lower degree of participation. The proposed algorithm 10 makes use of the ordering factor RI already computed for each fingerprint of FICFO, therefore this technique is not compatible with the RCIC scenario. Weighted clustering is a generalized Fair clustering. Fair clustering, indeed, can be seen as a version of this algorithm that simply impose the fingerprint weight $RI = 1$ for each item.

Algorithm 9 Fair clustering
Input: d: dimension of cropped fingerprints, Th: threshold, Set: fingerprints.
Output: G: number of groups, C_G: Clusters, K_G: Reference fingerprints.

procedure $Fair\_clust$(d, Th, Set)
  $Av\_fact = 1$ ▷ length of the current group
  while $(N \neq 0)$ **do** ▷ we still have items to assign to groups
    $F_1 = load(Set(1))$ ▷ Read zero mean full camera fingerprint
    $C_G(G) \leftarrow F_1$
    $K_G = F_1$ ▷ reference equal to the first image
    for $jj = 2$ to $|Set|$ **do**
      $F_{jj} = load(Set(jj))$ ▷ Read zero mean full camera fingerprint
      $C = \frac{1}{d}\sum_{x=1}^{d} K_G[x]F_{jj}[x]$ ▷ cross correlation
      if $C \geq Th$ **then**
        $C_G(G) \leftarrow F_{jj}$ ▷ new item included in the group
        $K_G = \frac{(K_G * Av\_fact)+F_{jj}}{Av\_fact+1}$ ▷ K_G= updated reference
        $norm\_Ref = \frac{\sum K_G}{|K_G|}$
        $K_G = \frac{K_G}{norm\_Ref}$ ▷ normalized Ref
        $Av\_fact = Av\_fact + 1$ ▷ updates number of items in $C_G(G)$
      end if
    end for
    $N = N - 1$
  end while
end procedure

4.2 Attraction phase, clustering refinement

Lastly, we may need an attraction phase, in which all the centroids are considered as single fingerprints and we proceed on clustering them again in fewer groups. In an attraction phase, when we have way less components respect to the beginning, we can treat our clusters as single items and we can try to attract them by computing a new threshold that takes into account the common noise [24]. The cross correlation between groups indeed suffers for a non-negligible inter-class noise, therefore in this phase, we have to tune the previous clustering threshold to work with groups. The fingerprints with correlation smaller than a threshold value, respect to all the others, are assigned to the unclustered set of fingerprints, and the remaining clusters are declared as final clusters configuration.

The imperfections in the PRNU fingerprint depend on artifacts which may be common to multiple cameras. For instance, the color filter array produces artifacts

Algorithm 10 Weighted clustering
Input: d: dimension of cropped fingerprints, Th: threshold, Set: fingerprints.
Output: G: number of groups, C_G: Clusters, K_G: Reference fingerprints.

procedure $Weighted\_clust$(d, Th, Set)
  $Av\_fact = 0$ ▷ length of the current group
  while $(N \neq 0)$ **do** ▷ we still have items to assign to groups
    $F_1 = load(Set(1))$ ▷ Read zero mean full camera fingerprint
    $C_G(G) \leftarrow F_1$
    $K_G = F_1$ ▷ reference equal to the first image
    for $jj = 2$ to $|Set|$ **do**
      $F_{jj} = load(Set(jj))$ ▷ Read zero mean full camera fingerprint
      $C = \frac{1}{d}\sum_{x=1}^{d} K_G[x]F_{jj}[x]$ ▷ cross correlation
      if $C \geq Th$ **then**
        $C_G(G) \leftarrow F_{jj}$ ▷ new item included in the group
        $RI = Set(j).ordering\_factor$ ▷ fingerprint weight
        $K_G = \frac{(K_G * Av\_fact) + F_{jj} * RI}{Av\_fact + RI}$ ▷ K_G= updated reference
        $norm\_Ref = \frac{\sum K_G}{|K_G|}$
        $K_G = \frac{K_G}{norm\_Ref}$ ▷ normalized Ref
        $Av\_fact = Av\_fact + RI$ ▷ updates number of items in $C_G(G)$
      end if
    end for
    $N = N - 1$
  end while
end procedure

in the interpolation and the 8x8 pattern of the JPEG compression matrix generates block artifacts, common to all the cameras.
We set the random components almost to zero, while computing the mean of the fingerprints, the centroid of the group, but we may also enhance the noises due to artifacts. The hypothesis of inter-correlation= 0, true for single fingerprints, is no longer verified for groups. It is increased by the positive value of the noise due to artifacts, as discussed in section 2.2.
Therefore, during the attraction step, the threshold needs to be enhanced, not to obtain false positives. In this phase, we can apply the Gaussian mixture threshold, as in equation 2.17, making use of the information we got from the clustering (the mean μ and the variance σ of the Gaussian distribution), or we can empirically increase the threshold by a multiplicative factor ρ.

$$Th_{attr} = \rho\, Th \tag{4.3}$$

This easy static approach does not rely on the output of the clustering phase, but because of this very reason, we are sure the threshold will not lead to unnecessarily confident conclusions.

The Gaussian mixture model with 2 components consistently splits the data into the mixture of two distributions, the cross camera Inter-class correlation and the same camera Intra-class correlation, as in figure 2.1.
For the attraction phase we may update the threshold to better exclude null hypothesis in the merging phase with a refined formula:

$$Th = \mu(Inter) + \sqrt{2\,\sigma(Inter)}\,erfc^{-1}(2\,P_{FA}). \tag{4.4}$$

4.2.1 Full Matrix attraction with merge singleton

The algorithm described in 4.1.1 can be also applied as an attraction phase, if we input the cross correlation between the centroids of the groups, instead of the full cross correlation of fingerprints, obtained after a previous clustering step.
The complexity is again heavily dependent on the size of cross-correlation; the upper bound is $\mathcal{O}(G^2)$, where the clustering phase should have already greatly reduced the number of groups G respect to the number of fingerprints in the dataset.

The full clustering technique can be refined by making use of vectors and matrices that exploit better the statistics in our possession about the data, such as the knowledge of the clusters dimension. Taking into account those considerations, we can prefer to merge together unclustered fingerprints or little groups, and we can normalize the threshold by different factors to be customized for specific cases.
Full Matrix, in algorithm 11, at each step of the iterative algorithm, computes the maximum value from the matrix and it tries to form a new pair of groups, until there are no normalized correlations values still over the threshold.

The parameter w_corr asks the clustering phase to save a metric in the normalization step of each group. It retrieves σ_G, the standard deviation of the group.
w_corr is initialized to σ_G^2 that is a value greater or equal to the cluster size $|C_G|$ but lower than $|C_G|^2$. Then, as for the Full matrix clustering, w_corr is updated each time the group undergo a merging operation.

4.2.2 Hierarchical Agglomerative clustering

Referring to the presented Hierarchical Agglomerative clustering in section2.5.3 in the literature, we tried different distances criterion in order to find the partition that best represents the clustering attraction we want to obtain and grant a legit

Algorithm 11 Full Matrix attraction Input: Th: threshold, corr: normalized correlation matrix, γ: gamma parameter, C_G: clusters, σ_G: clusters standard deviation.
Output: G: number of groups, C_G: updated clusters.

procedure CC_bin_vect(Th, corr, γ, C_G, σ_G)
  $[C, ii, jj] = max(corr)$ ▷ max in the matrix and its indices
  $len\_Gr = |C_G|$
  $w\_corr = \sigma_G^2$
  while $C > Th$ **do** ▷ there still exist a norm(C) value greater than Th
    $corr\_coeff = C * \sqrt{w\_corr(ii)} * \sqrt{w\_corr(jj)}$
    $corr(:, ii) = \frac{\sqrt{w\_corr(ii)} * corr(:, ii) + \sqrt{w\_corr(jj)} * corr(:, jj)}{\sqrt{w\_corr(ii) + w\_corr(jj) + 2 * corr\_coeff}}$
    $w\_corr(ii) = w\_corr(ii) + w\_corr(jj) + 2 * corr\_coeff$
    $C_G(ii) \leftarrow C_G(jj)$
    $C_G(jj) = \emptyset$
    $len\_Gr(ii) = len\_Gr(ii) + len\_Gr(jj)$ ▷ $Group_{jj}$ absorbed by $Group_{ii}$
    $corr(ii, :) = corr(:, ii)$ ▷ corr matrix is symmetrical
    $corr(ii, ii) = Null$ ▷ value out of bounds, never considered again
    $corr(jj, :) = corr(:, jj) = Null$
    $bin\_vect = len\_Gr > 1$ ▷ binary mask of length
    $bin\_matrix = bin\_vect \times bin\_vect'$ ▷ compute the mask
    $len\_matrix(not(bin\_matrix)) = 1$ ▷ force the length to 1
    $G = G - 1$ ▷ groups and matrix dimension reduced by one
    $len\_matrix = |len\_Gr \times len\_Gr|^{\gamma}$
    $bin\_vect = len\_Gr > 1$ ▷ binary mask of length
    $bin\_matrix = bin\_vect \times bin\_vect'$ ▷ compute the mask
    $len\_matrix(not(bin\_matrix)) = 1$ ▷ force some lengths to 1
    $[C, ii, jj] = max \frac{corr}{len\_matrix}$ ▷ max in the norm matrix and its indices
  end while
end procedure

dendrogram, where the group distance vector is monotonically increasing, as in figure 2.10.
The Average-Linkage Criterion, rather than computing the distance between centroids, averages over all possible pairs of items between the two groups:

$$av\_dist(C\_G_1, C\_G_2) = \frac{\sum \sum ||C\_G_1 - C\_G_2||}{|C\_G_1||C\_G_2|} correlations \tag{4.5}$$

It produces compact clusters that may have some elongated shape.
As we experienced in Matlab, the centroid linkage criterion breaks the assumption of monotonicity of merges and can result in an inversion in the dendrogram [21], so we never use it in our experiments.
For retrieving our updated groups, we have to compare the dendogram with a threshold that imposes a specific height limit, as it is expressed in algorithm 12. Everything lays under the bound is considered as the partition of our data, the clustering representation we were searching for.

The computational complexity is:

$$\mathcal{O}\left(G^2 log(G)\right) correlations \tag{4.6}$$

where G is the number of groups in the first attraction.

Algorithm 12 Hierarchical Agglomerative clustering
Input: Th: threshold, corr: normalized correlation matrix, C_G: clusters.
Output: G: number of groups, C_G: updated clusters.

procedure $Aggl\_clust$(Th, corr, C_G)
  $Th_{AC} = 1 - Th$ ▷ it requires a Th_{AC} in the form of 1-distance
  $T = \emptyset$ ▷ tree initialization
  $A = C_G$ ▷ active set initialization
  while $|A| \geq 1$ **do** ▷ Choose the pair in A with best average distance
    $[C, C\_G_1, C\_G_2] = min(av\_dist(corr))$ ▷ best pair and its indices
    $C\_G_1 \leftarrow C\_G_2$
    $C\_G_2 = \emptyset$
    $T \leftarrow C\_G_1$
    $T.height \leftarrow C$ ▷ stores all the attraction heights
  end while
  $plot(dendogram)$ ▷ output a plot as in figure 2.10
  $G = max(T(Th_{AC}))$ ▷ update number of groups for height $\leq T(Th_{AC})$
end procedure

4.2.3 Pure attraction

The purification check is an optional step that aims to increase the precision of the clusters by questioning the previous results.
Each fingerprint is compared with the list of centroids to find the best groups they should have been clustered in. The purification check, or Pure attraction, can be

considered as an attraction step, but it aims to refine the clusters internally, not to regroup them in better representations, at maximum it merges some unclustered images in other groups.
Another solution is to deem the purification check just as a pre-processing of the attraction phase. We usually skip this phase because is computationally very expensive and in our experiments, it always leaves some false positives in the groups.
In its Dynamic version, the check may also lead to many splits of the previous groups. The clusters dimension drops quite to simply unclustered fingerprints and number of groups G tends to explode, this increases a lot the complexity of a successive attraction phase in which all those items are very hard to merge again. This is the reason why in our results we present the purification check just as an attraction step.
In the following sections we propose two different approaches to this phase. They propose two unsupervised solution, because they do not require any threshold to be set when comparing the groups.

Static Pure attraction

Following the example of the k-mean algorithm, each fingerprint is extracted from its own group and inserted again in the cluster with the lowest distance from itself. Once it is finished, we update all the centroids of the groups.
The main issue of this method is the lack of regrouping. As we said, FICFO and RCIC may have left many unclustered images, that form groups by their own. When we compute which group fits them best, we always end up with the group they were already in, where the centroid was the fingerprint itself and so it has auto-correlation= 1. Therefore, the algorithm considers the unclustered groups as the best solution again.
A pseudo-code for the Static Pure attraction is presented in algorithm 13.

Dynamic Pure attraction

Each fingerprint F_{kk} in algorithm 14 is iteratively extracted from its own cluster. We compute the reference K'_{ii} the group would have been excluding F_{kk}:

$$K'_{ii} = \frac{(K_{ii} * len\_i) - F_{kk}}{len\_i - 1} \tag{4.7}$$

as if we have already clustered all the other fingerprints but we still need to decide where to attract the current one. We compare the cross correlation between F_{kk} and K'_{ii} with the correlation obtained respect to each other cluster K_{jj}. In this way, we always work with big groups, that should contain a good estimation of their reference.

We impose the index of the best group equal to ii. In case the new correlation with group C_{jj} is higher, we update the index to jj and we proceed to consider all the remaining groups.
Once the cycle is over, we may find a better cluster C_{jj} in which the fingerprint should lay, the best of the list, or we still have index= ii.
For index $\neq$ ii, we update the reference centroids of the old and new clusters, while moving the fingerprint F_{kk}, in the method 15. This solution forces every unclustered to find the best group excluding itself, that now contains zero elements. Those empty groups always give as correlation corr_old= 0, so every positive cross correlation with other groups or unclustered image updates the index.

Algorithm 13 Static Pure clustering
Input: Set: dataset source files, d: dimension of cropped fingerprints, C_G: clusters, K_G: Reference fingerprints.
Output: G: number of groups, C_G: updated clusters.

1: **procedure** *static_attraction*(Set, d, C_G, K_G)
2: $C_G = \emptyset$ ▷ empty every group
3: **for** $kk = 1$ to $|Set|$ **do**
4: $F_{kk} = Set(kk)$ ▷ consider fingerprint F_{kk} from the dataset
5: $corr\_best = 0$ ▷ initialize the best corr to 0
6: **for** $jj = 1$ to $|C_G|$ **do**
7: $corr\_new = \frac{F_{kk} \times K_{jj}}{d * d}$
8: **if** $corr\_new > corr\_best$ **then**
9: $corr\_best = corr\_new$
10: $index = jj$ ▷ better group found
11: **end if**
12: **end for**
13: $C_G(index) \leftarrow F_{kk}$ ▷ add fingerprint(kk) to the group
14: **end for**
15: **end procedure**

Algorithm 14 Dynamic Pure clustering
Input: Set: dataset source files, d: dimension of cropped fingerprints, C_G: clusters, K_G: Reference fingerprints. Output: G: number of groups, C_G: updated clusters.

1: **procedure** *dynamic_attraction*(Set, d, C_G, K_G)
2: **for** $ii = 1$ to $|C_G|$ **do**
3: $kk = 1$ ▷ fingerprint index
4: **while** $kk < len_i$ and $len_i > 0$ **do**
5: $F_{kk} = Set(kk)$
6: $K'_{ii} = \frac{(K_{ii} * |C_G(ii)|) - F_{kk}}{|C_G(ii)| - 1}$ ▷ K'_{ii} = reference of $\langle C_G(ii) - F_{kk} \rangle$
7: $norm\_K'_{ii} = \frac{\sum K'_{ii}}{|K'_{ii}|}$
8: $K'_{ii} = \frac{K'_{ii}}{norm\_K'_{ii}}$
9: $corr\_old = \dfrac{F_{kk} \times K'_{ii}}{d * d}$
10: **for** $jj = 1$ to $|C_G|$, $jj \neq ii$ **do** ▷ search for a better group
11: $corr\_new = \dfrac{F_{kk} \times K(jj)}{d * d}$
12: **if** $corr\_new > corr\_old$ **then**
13: $corr\_old = corr\_new$
14: $index = jj$ ▷ better group found
15: $len\_index = |C_G(jj)|$ ▷ stores the length of $C_G(jj)$
16: **end if**
17: run $shift\_fingerprints$ ▷ moves F_{kk} to $C_G(index)$ in script 15
18: **end for**
19: $kk = kk + 1$
20: **end while**
21: **end for**
22: **end procedure**

Algorithm 15 Shift fingerprints script

1: **procedure** $shift\_fingerprints$
2: **if** $index \neq ii$ **then** ▷ only if the group has changed
3: $C_G(index) \leftarrow F_{kk}$ ▷ add Set(kk) in the group
4: **if** $len_i > 1$ **then** ▷ remove 1 fingerprint from a big group
5: $C_G(ii)$ remove F_{kk}
6: $K_{ii} = K'_{ii}$ ▷ overwrite K(ii)
7: **else** ▷ remove 1 fingerprint from an unclustered group
8: $C_G(ii) = \emptyset$
9: **end if**
10: $K_{index} \frac{(K_{index} * |C_G(ii)|) + F_{kk}}{|C_G(ii)|+1}$ ▷ updates $K(jj)|_{jj=index}$
11: $norm\_K_{index} = \frac{\sum K_{index}}{|K_{index}|}$
12: $K_{index} = \frac{K_{index}}{norm\_K_{index}}$
13: **end if**
14: **end procedure**

Chapter 5

Experimental results

This chapter contains a discussion about the experimental setup and the evaluation metrics we applied to our work. It also analyzes the results in tables of performances. The final section compares the best proposed algorithms among themselves and with State of the art techniques.

5.1 Datasets

We considered 4 well known datasets, each consisting of 1000 non-overlapping images taken from the Dresden datasets [25], as well as a small dataset of 80 images used for initial tests.
The datasets include also their the ground truth, the list of corrected assignment of fingerprints for each camera. All images are center-cropped to 1023x1023 pixels, allowing us to compare all the fingerprints.
In symmetric datasets, each camera contributes equally to the dataset, while the distribution of pictures is not constant in the asymmetric ones. Thus, there are folders containing very few images that are harder to cluster. Our proposed algorithms can be analyzed in the $NC \gg SC$ scenario, by modifying the average number of images per camera SC:

- D0: It is an easy small dataset. It consists of 80 images taken by 4 cameras, each equally contributing 20 images. The 4 cameras belong to different models and brands. D0 dataset is just useful as a testbench, to investigate the parameters of the algorithms and for debugging purposes. It it not a very challenging dataset because of the very distinct pictures it is made of.

- D1: It is an easy symmetric dataset, it comprises a folder with 40 images for each of the 25 cameras, 1000 images in total. The 25 cameras are originated by

different models and cover the most popular brands: Agfa, Canon, Casio, FujiFilm, Kodak, Nikon, Olympus, Panasonic, Pentax, Praktica, Ricoh, Rollei, Samsung and Sony.

- D2: It is an easy asymmetric dataset, consisting of folders of images taken from 25 cameras, 1000 images in total. The contribution of each camera varies from 20 to 60 images.

- D3: It is a hard symmetric dataset, consisting of a folder of 20 images times 50 cameras, 1000 images in total. The 50 camera models cover 8 brands, so we have some brands contributing with many models. This complicates the work of our algorithms because we expect intra-brand correlation quite high, a strong presence of some false positives in our clusters or many more groups than what the ground truth suggests.

- D4: It is a hard asymmetric dataset. The same 50 cameras as in D3 constitute the dataset, but with a contribution varying from 10 to 30 images. We expect this last dataset to output the worst performances of our experiments.

5.2 Our metrics

We base our metrics on clustering theory: Precision P, Recall R, F-measure F, Rand Index RI and Adjusted Rand Index ARI. In order to evaluate P, R, F, RI and ARI, we load in memory the ground truth G_T.
Moreover, we compute the number of cross-correlations NCC as evaluation of complexity and the number of groups G. These two metrics can be obtained in absence of the ground truth, but they do not inform us enough about the performances of our codes. We can also obtain the complexity reduction as

$$Cr = \frac{N * (N-1)}{2\,NCC} \tag{5.1}$$

Cr offers a normalization of NCC over the number of images N in our dataset, but as in our experiments N is fixed to 1000 images, we omit to compute it.

$$P = \frac{\sum_{ii=1}^{G} \max_{jj=1}^{|G_T|} |C_{ii} \cap G_T(jj)|}{\sum_{ii=1}^{G} |C_{ii}|} \tag{5.2}$$

P is computed as the sum of each largest number of fingerprints in cluster C_{ii} that comes from a ground truth class $G_T(jj)$, namely their intersection, over the total number of items in the ground truth class. P represents the ratio of the true

positives over the positives (true and false).

$$R = \frac{\sum_{jj=1}^{|G_T|} \max_{ii=1}^{G} |C_{ii} \cap G_T(jj)|}{\sum_{jj=1}^{|G_T|} |G_T(jj)|} \tag{5.3}$$

R is computed as the total intersection of the fingerprints in cluster C_{ii} and the best equivalent ground truth class $G_T(jj)$, over the the size of the cluster C_{ii}. R represents the ratio of the true positives over the cluster (true positives plus false negatives). Notice that because the dimension of the cluster and the ground truth partitions is the same and is equal to the total number of fingerprints N in the dataset,

$$\sum_{ii=1}^{G} |C_{ii}| = \sum_{jj=1}^{|G_T|} |G_T(jj)| = N$$

The F-measure is the arithmetic mean of P and R, so

$$F = 2\,\frac{P\,R}{P+R} \tag{5.4}$$

F is useful to express in one parameter the trend of both P and R.
R increases as the inverse of the threshold Th and so proportionally to the PFA that we apply. Dealing with more groups usually means a better precision in the clusters, because there are less outlier and fewer errors we may make, but at the same time it shows a worse recall and ARI, because all those little groups are quite far from matching the ground truth.

The Rand Index RI, in algorithm 16, between our result C_G and the ground truth G_T, can be computed as the number A of the couples falling in the same cluster in both partitions, minus the value B of the couples falling in the same cluster in C_G and in different clusters in G_T. Vice versa, C represents the couples falling in different clusters in C_G and in the same cluster in G_T. This computation has to be divided by the binomial coefficient of N fingerprints choose a couple ($k = 2$), representing all the couple points C_P.

The Adjusted Rand Index represents the normalized agreement between two partitions.

$$ARI = \frac{RI - E[RI]}{1 - E[RI]} \tag{5.5}$$

where E[RI] is the expected value of RI, obtained from the probability P(RI).

$$E[RI] = \sum_{ii} RI_{ii} \times P(RI_{ii}) \tag{5.6}$$

Algorithm 16 RI computation
Input:C_G: clusters, G_T: Ground truth, N: number of fingerprints.
Output: RI parameter

procedure *RI_computation*(C_G, G_T, N)
  $Matrix = C_G(ii) \cap G_T(jj)$ ▷ total intersection of C_G and G_T
  $C_P = \binom{N}{2}$ ▷ number of couple points
  $A = \sum_{ii} \sum_{jj} (Matrix^2)$ ▷ couples in the same partition in both
  $B = \frac{\sum_{ii} Matrix^2(ii,:)}{2}$ ▷ couples in the same partition just in C_G
  $C = \frac{\sum_{ii} Matrix^2(:,ii)}{2}$ ▷ couples in the same partition just in G_T
  $RI = \frac{C_P + A - B - C}{C_P}$
end procedure

Our aim is to design an algorithm able to improve the F-measure of the groups and the Adjust Random Index ARI, while keeping the precision P ideally always at 100%, without introducing false positives, even in presence of the $NC \gg SC$ problem.

5.3 Randomness variation

In the RCIC codes the clustering algorithm randomly permutes the fingerprint source file and it selects different reference fingerprints as cluster attractors for each run. Therefore, every experiment has to be repeated a number of times τ to obtain an average performance metric.
We executed $\tau = 10$ runs of the codes, for each experiment, and we tested the stability of RCIC by computing the first quartile $Q_1(data)$ and third quartile $Q_3(data)$. The first quartile is able to split the first 25% of the sorted data from the rest of the measurements, the third quartile similarly splits the first 75% of data. The second quartile $Q_2(data)$, also known as median, points to the central data and it represents a good alternative to the average for deriving the mean run of the code, one of the most probable output in a Gaussian distributed vector of data.

The following table 5.1, shows the difference Δ between the third quartile and the first quartile, that represents the central 50% of data.

$$\Delta(data) = Q_3(data) - Q_1(data) \tag{5.7}$$

Generally, the Δ values are quite near to zero, so the randomness does not have a big impact, especially for the easy datasets. RCIC is a very stable algorithm in base version but it generally shows a bigger variance of the data (up to ≈ 0.11) in the codes with the attraction phase.
The next subsections, regarding RCIC or RCIC-A, contain the median value, the mean and the Δ values of the performance metrics, computed on $\tau = 10$ runs of the code, for each case in exam.

RCIC Fair					
dataset	$\Delta(P)$	$\Delta(R)$	$\Delta(F)$	$\Delta(RI)$	$\Delta(ARI)$
D1	0.0010	0.0335	0.0209	0.0014	0.0246
D2	0.0042	0.0265	0.0149	0.0019	0.0295
D3	0.0018	0.0135	0.0097	0.0004	0.0146
D4	0.0025	0.0113	0.0071	0.0001	0.0055
RCIC-A Fair Complete					
dataset	$\Delta(P)$	$\Delta(R)$	$\Delta(F)$	$\Delta(RI)$	$\Delta(ARI)$
D1	0.0488	0.0434	0.0269	0.0092	0.0953
D2	0.0358	0.0300	0.0244	0.0135	0.1150
D3	0.0309	0.0263	0.0162	0.0021	0.0363
D4	0.0447	0.0092	0.0255	0.0030	0.0440

Table 5.1: Example of Δ values for RCIC Fair and RCIC-A Fair Complete

5.4 Our experiments

We compare each code in a strict, average and relaxed scenario. We set the threshold to Th=0.0050, Th=0.0040 or Th=0.0030 and consequently we fix the value of P_{FA}, for a square crop of $\frac{1}{\sigma^2} = 1023^2$ pixels, as we can compute from the inverse threshold formula:

$$P_{FA} = \frac{1}{2} erfc(\frac{Th}{\sqrt{2}\,\sigma}) \tag{5.8}$$

Our big concern is the value of P_{FA}. If it is too big it may fill the clusters with false positives. This situation can bring us to a wrong computation of the centroids of our groups and the consequential attraction of more and more false positives in the groups. A very low P_{FA} instead, may not attract enough and would output too many little clusters.

The attraction phase make use of the static empirically increase threshold described in the formula 4.3, with $\rho = 1.25$. The threshold becomes Th=0.0063, Th=0.0050 or Th=0.0037. All the corresponding values of P_{FA} have been reported in the equivalence table 5.2.

Th	0.0063	0.0050	0.0040	0.0037	0.0030
P_{FA}	$8.0933\,10^{-11}$	$1.5687\,10^{-07}$	$2.1383\,10^{-05}$	$6.2464\,10^{-05}$	$1.0739\,10^{-03}$

Table 5.2: Threshold values and P_{FA} equivalents

For the Full Matrix algorithms, we experimented with many values of the index γ, even in absence of it, for γ= 0, We found a compromise between high γ values, that has a big impact on our data and avoids introducing many false positives, and low γ values that does not force the algorithms to stop too early. In all the following tables present from table 5.3 to table 5.47 we fix γ= 0.1 when needed.

5.4.1 Full Matrix

We conduct an ablation study by removing one by one certain parameters, in order to understand their contribution to the performances. The groups form in a slightly different way, considering their dimension and avoiding unsure merging.
We check the usefulness of out parameters: corr_coeff, len_matrix and bin_vect for different values of Th and γ. In order not to introduce too many tables, we show only the difference between not considering the bin_vect (table 5.3) and using it (table 5.4), in the next section.

The first version of the code in table 5.3 is obtained by including all of the proposed solutions except for bin_vect. The Full Matrix already performs very good because it normalizes each correlation by the length of the groups it tries to merge, raised to the power of γ.
The precision value for the first dataset is maximum, for the strict threshold and it remains $P = 1$ even for the average threshold Th =0.0040. It means that there are no false positives in any group. For the other datasets we obtain as output quite pure clusters, with precision P always greater than 0.95 for high thresholds.
The recall shows two different behaviors for easy and hard datasets. As expected, R grows as the opposite of the threshold, varying for example from 0.9280 to 0.9480 in the second dataset.
It remains very high ($R > 0.91$) for easy datasets, where the fingerprint distances are more distinct. For D3 and D4 instead, the algorithm starts to merge less groups, leaving the recall values between $0.82 < R > 0.89$. The F-measure $0.87 < F > 0.96$ confirms the trends that we notice in the first two metrics.

The rand index, with its value always greater than 0.99, highlights a strong agreement between the cluster partitions and the ground truth. In its normalized version, ARI, it shows even more which versions perform better respect to the others. As for the recall, it presents two different trends for easy and hard datasets. $ARI \approx$ 0.90 to 0.94 confirms the strength of this algorithm over the easy datasets, while $ARI \approx 0.79$ to 0.85 shows the difficulty encountered in the hard datasets.

Because every modification of the code we introduce is irrelevant in front of the $\mathcal{O}(G^2)$ complexity, we ask ourselves if we can still improve the performances. From our tests, the difference between enhancing the correlation with the weight w_corr and not applying it, is minimal and irrelevant.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	499,500	1.0000	0.9160	0.9562	0.9950	0.9283	72
D1	0.004	499,500	1.0000	0.9270	0.9621	0.9957	0.9391	58
D1	0.003	499,500	0.9590	0.9340	0.9463	0.9930	0.9059	45
D2	0.005	499,500	0.9990	0.9280	0.9622	0.9953	0.9411	77
D2	0.004	499,500	0.9640	0.9410	0.9524	0.9935	0.9217	59
D2	0.003	499,500	0.9610	0.9480	0.9545	0.9940	0.9281	43
D3	0.005	499,500	0.9780	0.8260	0.8956	0.9948	0.8458	159
D3	0.004	499,500	0.9670	0.8380	0.8979	0.9947	0.8458	137
D3	0.003	499,500	0.9000	0.8600	0.8795	0.9923	0.7939	94
D4	0.005	499,500	0.9920	0.8200	0.8978	0.9947	0.8575	180
D4	0.005	499,500	0.9520	0.8530	0.8998	0.9940	0.8489	136
D4	0.005	499,500	0.9010	0.8820	0.8914	0.9920	0.8141	95

Table 5.3: Full matrix

5.4.2 Full Matrix merge singleton

The matrix bin_matrix is computed as the external product between bin_vect and itself, where bin_vect is a binary mask of length. It appears to have benefits over all the datasets in the base version because if forces the normalization introduced by len_matrix to be reset to 1, in case one of the two merged groups (or both) has length $|C_G| = 1$. This solution introduces an important alteration in the way the algorithm deals with unclustered and little groups, improving the quantity and the correctness of the attractions done.

From table 5.4 we can discuss the results obtained by including all of the proposed parameters. The precision value for the first dataset is again maximum, for the strict and the average thresholds. For the other datasets we obtain as output quite pure clusters, with precision P always greater than 0.95 for high thresholds. The recall has every value improved respect to the first proposal and it maintains two different behaviors for easy and hard datasets.
R remains very high ($R \geq 0.93$) for datasets D1 and D2, for D3 and D4 instead, the algorithm merges less groups, leaving the recall values between $0.83 < R > 0.89$.
We can notice that the number of groups is constantly bigger respect to the ground truth, quite the double respect to the correct value of $G = 25$ for D1 and D2 and $G = 50$ for the last datasets. As G is lower than the in first table, this algorithm returns a solution more near to the ground truth. The F-measure $0.87 < F > 0.97$ confirms the trends of P and R and it shows once again the slight improvements respect to table 5.3.

The rand index is stably over the value of 0.99 and the adjusted rand index ARI drops under 0.81 only for the relax threshold in the D4 scenario. As for the recall, it presents two different trends for easy and hard datasets. $ARI \approx 0.90$ to 0.94 confirms the strength of this algorithm over the easy datasets, while $ARI \approx$ 0.79 to 0.85 shows the difficulty encountered in the hard datasets.
The trade-off between R and ARI in our solutions seems always to suggest the strict threshold as the best. Further computations not reported in the tables confirm the ideal working point for every of our datasets as Th= 0.0050 and γ= 0.1.

5.4.3 Bloy base

Bloy2008 algorithm present one of the best outcomes in terms of metric values. Bloy obtains a good overview about the dataset, thanks to the big amount of checks and computations that it needs to do. The complexity is bound by $\mathcal{O}(N^2)$ and it reaches very high values, such as NCC= 305,894, more than half of the complete knowledge about every correlation exploited in the Full Matrix, with a fixed complexity of NCC= 499,500.
The presence of some $stop\_flag$ optimization avoids some useless loops, but it maintains Bloy very far from being defined as a low complexity algorithm.
As the table 5.5 highlights, Bloy is not able to maintain very high precision values for easy datasets; P=1 just in one case and P drops under 0.80 even for easy datasets. Despite that, it works very well in the hard scenarios with a very strict threshold.
As we can see from ARI< 0.60, the first two datasets are very far from the ground truth if we apply a relaxed threshold of Th=0.0030 and the situation becomes way worse in the last two datasets, with ARI values around and under 0.35. Our

consideration about the best threshold as Th= 0.0050 is even more valid for this algorithm, where we discourage the use of average and relaxed threshold.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	499,500	1,0000	0.9300	0.9637	0.9959	0.9421	58
D1	0.004	499,500	1,0000	0.9360	0.9669	0.9963	0.9487	47
D1	0.003	499,500	0.9580	0.9390	0.9484	0.9931	0.9084	39
D2	0.005	499,500	0.9990	0.9430	0.9702	0.9964	0.9555	62
D2	0.004	499,500	0.9620	0.9530	0.9575	0.9942	0.9310	49
D2	0.003	499,500	0.9580	0.9540	0.9560	0.9942	0.9307	37
D3	0.005	499,500	0.9790	0.8370	0.9024	0.9952	0.8567	149
D3	0.004	499,500	0.9690	0.8580	0.9101	0.9952	0.8610	121
D3	0.003	499,500	0.9140	0.8750	0.8941	0.9931	0.8161	84
D4	0.005	499,500	0.9920	0.8330	0.9056	0.9951	0.8696	169
D4	0.004	499,500	0.9510	0.8690	0.9082	0.9945	0.8618	121
D4	0.003	499,500	0.8650	0.8850	0.8749	0.9901	0.7810	81

Table 5.4: Full Matrix merge singleton

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	72,671	1,0000	0.9050	0.9501	0.9975	0.9656	33
D1	0.004	51,722	0.9774	0.9000	0.9371	0.9938	0.9154	34
D1	0.003	22,579	0.7379	0.7620	0.7497	0.9649	0.5688	30
D2	0.005	41,790	0.9957	0.8910	0.9405	0.9978	0.9738	34
D2	0.004	30,889	0.9825	0.9070	0.9432	0.9951	0.9404	36
D2	0.003	22,514	0.7978	0.7800	0.7888	0.9693	0.6490	32
D3	0.005	232,105	0.9845	0.8150	0.8918	0.9970	0.9180	68
D3	0.004	180,278	0.9119	0.7700	0.8350	0.9927	0.7991	74
D3	0.003	46,818	0.5463	0.5500	0.5481	0.9695	0.3262	57
D4	0.005	305,894	0.9910	0.8180	0.8962	0.9980	0.9530	69
D4	0.004	131,696	0.9356	0.8070	0.8666	0.9939	0.8521	72
D4	0.003	50,767	0.5578	0.5470	0.5524	0.9683	0.3524	64

Table 5.5: Bloy base

For Th= 0.0050, the F-measure suggests Bloy a bit worse than the Full Matrix, while the ARI presents the opposite consideration. This behavior suggests that the two methods are both very valid and not too far from each other in their best form. Also in Bloy, every value of RI is greater than 0.99.
We may choose to work with Bloy on the hard datasets for its very low values of G and its stable ARI>0.91.
We have to consider that the solutions presented up to now are usually too expensive to be applied in a real scenario, that is why we should rely on good solutions that are fast and efficient.

5.4.4 FICFO base

FICFO base is the version of the FICFO algorithm already proposed by Khan in his thesis. It does not make use of any attraction stage and it works with a clustering step that updates the centroids by adding the new reference and dividing the result by two, as in formula 3.2. The attractor fingerprints have an important role in determining the centroid. This approach tends to penalize the first fingerprints respect to the last added, from which we can extract only a rough estimation of the centroid, because of their lower RI ordering factor. The formula 3.2 moves the centroid too much, following the new and worse items included.

The algorithm in table 5.6 shows a low complexity, always lower than 30,000 for easy datasets and 62,000 for hard datasets.
The precision P for the strict threshold is strong and ≈ 0.99. This fact is simply due to the huge amount to quite pure cluster that we see in the output; G keeps values from 155 to 234, much higher than the ground truth dimension (G= 25 for easy datasets and G= 50 for hard datasets). Consequently, R, F and ARI stay in the range 0.80 to 0.90. RI maintains very high values ≈ 0.99, this is the sign that RI is not a good metric by itself.
Lower thresholds drop every performance result. The precision reaches $0.55<P<0.70$, the recall ≈ 0.50 in three cases and ARI is a little higher than 0.30.
The algorithm starts to merge groups which it too confident in considering near to each others. This misbehavior is surely dependent on the presence of some false positives already in the early stages of the clustering. Moving to lower threshold, we always increase the possibility for further fingerprints to be merged in a wrong group and moreover we move the centroid to follow a wrong direction, opening the possibility for many other errors to be merged in the same cluster.
The usual output is a partition with many little pure clusters and few very big and very wrong ones.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	28,818	0.9990	0.8220	0.9019	0.9891	0.8333	155
D1	0.004	24,204	0.9210	0.7730	0.8405	0.9822	0.7365	117
D1	0.003	16,789	0.6200	0.5000	0.5536	0.9538	0.3471	78
D2	0.005	29,951	0.9980	0.8320	0.9075	0.9890	0.8517	163
D2	0.004	23,454	0.9750	0.8230	0.8926	0.9875	0.8328	121
D2	0.003	15,113	0.6990	0.5830	0.6358	0.9563	0.4409	74
D3	0.005	53,121	0.9880	0.7730	0.8674	0.9935	0.7954	211
D3	0.004	43,314	0.9100	0.7390	0.8156	0.9898	0.6975	170
D3	0.003	29,015	0.5640	0.4980	0.5289	0.9748	0.3238	113
D4	0.005	61,676	0.9910	0.7560	0.8577	0.9924	0.7828	234
D4	0.004	47,921	0.9320	0.7320	0.8200	0.9900	0.7214	182
D4	0.003	29,437	0.6030	0.5030	0.5485	0.9720	0.3322	124

Table 5.6: FICFO base

5.4.5 RCIC base

RCIC base is the version of the RCIC algorithm introduced by Khan in his thesis. It is very similar to FICFO base. It uses the same approach for computing the reference mean of fingerprints as in formula 3.2 and it output the final partition without any attraction step. The only differences are the introduction of a stochastic behavior because of the randomization of fingerprints in the source list and the suppression of the ordering factor RI.
This approach contains the same issues of FICFO, moving the centroid too much, following the new and worse items included, and in addition it does not rely on good attractors in each group, leading to similar or even worse performances.
In tables 5.7-5.10, ARI keeps each value under 0.80 and the recalls waves around 0.50 to 0.75.
Because of the random behavior of the algorithm, both a strict and a relaxed threshold perform very bad, while an average threshold maintains better performances in the average of its runs (the mean) and in its typical run (the median). Even the best results on dataset D2: R= 0.7590 and ARI= 0.7717 are significantly lower than their FICFO counterparts R= 0.8230 and ARI= 0.8328.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	33,331	0.9897	0.7540	0.8559	0.9858	0.7732	168
$Q_2(data)$	0.005	33,328	0.9850	0.7510	0.8523	0.9853	0.7651	168
$\Delta(data)$	0.005	2,898	0.0280	0.0180	0.0221	0.0031	0.0486	7
$E(data)$	0.004	23,780	0.9550	0.7910	0.8653	0.9860	0.7873	114
$Q_2(data)$	0.004	23,780	0.9550	0.7910	0.8653	0.9860	0.7873	114
$\Delta(data)$	0.004	4,185	0.0180	0.0060	0.0038	0.0019	0.0253	4
$E(data)$	0.003	16,929	0.6765	0.5500	0.6063	0.9591	0.4122	86
$Q_2(data)$	0.003	16,929	0.6765	0.5500	0.6063	0.9591	0.4122	86
$\Delta(data)$	0.003	2,178	0.0430	0.0260	0.0015	0.0021	0.0087	2

Table 5.7: RCIC base- D1

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	27,931	0.9941	0.6012	0.7485	0.9770	0.6096	187
$Q_2(data)$	0.005	27,893	0.9940	0.6000	0.7483	0.9756	0.6089	188
$\Delta(data)$	0.005	3,241	0.0318	0.0190	0.0211	0.0031	0.0489	8
$E(data)$	0.004	26,429	0.9700	0.7590	0.8516	0.9835	0.7717	131
$Q_2(data)$	0.004	26,852	0.9720	0.7610	0.8553	0.9860	0.7733	128
$\Delta(data)$	0.004	3,185	0.0280	0.0090	0.0048	0.0029	0.0243	5
$E(data)$	0.003	16,498	0.7480	0.6020	0.6671	0.9611	0.4955	83
$Q_2(data)$	0.003	16,629	0.7465	0.6004	0.6639	0.9591	0.4922	86
$\Delta(data)$	0.003	1,978	0.0330	0.0262	0.0015	0.0021	0.0080	2

Table 5.8: RCIC base- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0,005	65,087	0.9906	0.7119	0.8284	0.9922	0.7395	243
$Q_2(data)$	0,005	65,311	0.9900	0.7130	0.8290	0.9922	0.7409	242
$\Delta(data)$	0,005	2,303	0.0015	0.0048	0.0026	0.0001	0.0069	3
$E(data)$	0,004	47,703	0.9370	0.7128	0.8097	0.9910	0.7122	180
$Q_2(data)$	0,004	47,859	0.9370	0.7200	0.8143	0.9911	0.7185	176
$\Delta(data)$	0,004	2,903	0.0300	0.0180	0.0228	0.0017	0.0529	11
$E(data)$	0,003	30,580	0.6325	0.5209	0.5712	0.9779	0.3716	126
$Q_2(data)$	0,003	30,221	0.6220	0.5220	0.5639	0.9773	0.3613	123
$\Delta(data)$	0,003	956	0.0398	0.0133	0.0203	0.0018	0.0327	9

Table 5.9: RCIC base- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0,005	70,499	0.9878	0.6800	0.8055	0.9905	0.7174	274
$Q_2(data)$	0,005	71,316	0.9890	0.6810	0.8083	0.9905	0.7150	276
$\Delta(data)$	0,005	3,668	0.0050	0.0030	0.0004	0.0003	0.0096	27
$E(data)$	0,004	53,719	0.9138	0.6573	0.7643	0.9876	0.6423	203
$Q_2(data)$	0,004	54,819	0.9150	0.6390	0.7470	0.9867	0.6137	209
$\Delta(data)$	0,004	7,181	0.0210	0.0622	0.0467	0.0025	0.0854	25
$E(data)$	0.003	31,874	0.6496	0.5227	0.5791	0.9762	0.3949	134
$Q_2(data)$	0.003	31,341	0.6500	0.5290	0.5877	0.9759	0.4058	133
$\Delta(data)$	0.003	2,967	0.0143	0.0398	0.0292	0.0017	0.0425	4

Table 5.10: RCIC base- D4

5.4.6 FICFO First reference clustering

The First reference approach in table 5.11 shows an impressive number of clusters, G keeps values from 97 to 351, and consequently a very low value of recall.
The result of First reference clustering is a merge of only the fingerprints that have very low distance from the first one, so in the best scenarios it creates a huge amount of pure groups (with precision greater than 0.99), while in many other realistic cases it also introduces some false positives (as the precision becomes $0.76<P<0.89$).
NCC raises over 100,000 in the last dataset and in general it has high values because of the big quantity of fingerprints that remain unclustered for many cycles. This solution is very bad in terms of computational complexity and performances, presenting $0.50<ARI<0.70$, so we never adopt it as a base for the attraction step.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	61,772	0.9960	0.0097	0.0192	0.9783	0.6065	271
D1	0.004	40,851	0.9870	0.0172	0.0339	0.9819	0.6937	187
D1	0.003	23,332	0.8840	0.0321	0.0620	0.9789	0.6713	97
D2	0.005	58,358	0.9940	0.0110	0.0217	0.9776	0.6490	261
D2	0.004	39,920	0.9830	0.0180	0.0354	0.9809	0.7179	192
D2	0.003	22,544	0.8810	0.0328	0.0633	0.9770	0.6784	98
D3	0.005	86,359	0.9930	0.0070	0.0139	0.9896	0.6202	317
D3	0.004	61,297	0.9530	0.0111	0.0219	0.9907	0.6847	231
D3	0.003	37,680	0.8040	0.0178	0.0348	0.9861	0.5689	133
D4	0.005	100.698	0.9920	0.0056	0.0112	0.9874	0.5826	351
D4	0.004	68,247	0.9450	0.0093	0.0184	0.9885	0.6415	244
D4	0.003	40,704	0.7660	0.0150	0.0294	0.9828	0.5037	141

Table 5.11: FICFO First reference clustering

5.4.7 RCIC First reference clustering

The First reference approach has even worse performances when applied to RCIC, as we notice in tables 5.12-5.15. The NCC value grows very much, especially for the hard dataset, up to 191,264 and it quite reaches the complexity of Bloy, while the algorithm remains very far in terms of goodness of results. ARI struggles to reach 0.60 even on easy datasets. This solution contains the majority of the R values under 0.0050. It is meaningless and it does not deserve to be applied.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	74,302	0.9932	0.0489	0.0932	0.9742	0.4977	298
$Q_2(data)$	0.005	74,122	0.9935	0.0488	0.0930	0.9743	0.5001	300
$\Delta(data)$	0.005	3,854	0.0025	0.0032	0.0028	0.0012	0.034	9
$E(data)$	0.004	48,311	0.9676	0.0574	0.1084	0.9771	0.5844	208
$Q_2(data)$	0.004	48,824	0.9690	0.0572	0.1080	0.9771	0.581	210
$\Delta(data)$	0.004	3,490	0.0070	0.0039	0.0050	0.0020	0.0447	9
$E(data)$	0.003	25,604	0.8434	0.0615	0.1146	0.973	0.554	113
$Q_2(data)$	0.003	25,739	0.8470	0.0614	0.1145	0.9734	0.5562	114
$\Delta(data)$	0.003	1,655	0.0253	0.0035	0.0061	0.0022	0.0426	5

Table 5.12: RCIC First reference clustering- D1

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	70,429	0.9918	0.0531	0.1008	0.9725	0.5362	295
$Q_2(data)$	0.005	70,250	0.9915	0.0524	0.0995	0.9723	0.5301	295
$\Delta(data)$	0.005	2,669	0.0038	0.0019	0.0025	0.0008	0.0174	10
$E(data)$	0.004	43,516	0.9711	0.0606	0.1141	0.9757	0.6152	199
$Q_2(data)$	0.004	42,095	0.9715	0.0609	0.1146	0.9759	0.6179	199
$\Delta(data)$	0.004	2,466	0.0035	0.0041	0.0038	0.0019	0.0404	5
$E(data)$	0.003	24,076	0.8518	0.0631	0.1174	0.9718	0.5897	107
$Q_2(data)$	0.003	23,998	0.8510	0.0635	0.1182	0.9720	0.5996	109
$\Delta(data)$	0.003	372	0.0060	0.0025	0.0035	0.0015	0.0306	5

Table 5.13: RCIC First reference clustering- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	158,622	0.9880	0.0050	0.0100	0.9876	0.5145	346
$Q_2(data)$	0.005	144,385	0.9823	0.0050	0.0100	0.9882	0.5140	356
$\Delta(data)$	0.005	5,003	0.0118	0.0001	0.0003	0.0137	0.0161	14
$E(data)$	0.004	106,044	0.9742	0.0051	0.0101	0.9896	0.5248	266
$Q_2(data)$	0.004	105,338	0.9757	0.0051	0.0102	0.9899	0.5272	264
$\Delta(data)$	0.004	4,341	0.0200	0.0003	0.0006	0.0075	0.0192	17
$E(data)$	0.003	52,478	0.9240	0.0063	0.0125	0.9857	0.5381	180
$Q_2(data)$	0.003	52,782	0.9294	0.0061	0.0122	0.9858	0.5383	178
$\Delta(data)$	0.003	1,736	0.0400	0.0458	0.0427	0.0016	0.0438	16

Table 5.14: RCIC First reference clustering- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	191,264	0.9870	0.0047	0.0094	0.9852	0.4729	376
$Q_2(data)$	0.005	196,008	0.9850	0.0054	0.0107	0.9857	0.5141	374
$\Delta(data)$	0.005	63,222	0.0599	0.0040	0.0075	0.0198	0.0410	23
$E(data)$	0.004	83,754	0.9870	0.0047	0.0094	0.9852	0.4729	297
$Q_2(data)$	0.004	83,301	0.9850	0.0053	0.0105	0.9857	0.5141	293
$\Delta(data)$	0.004	63,222	0.0099	0.0020	0.0033	0.0773	0.0241	15
$E(data)$	0.003	50,961	0.8650	0.0047	0.0093	0.9852	0.4729	200
$Q_2(data)$	0.003	50,920	0.8740	0.0054	0.0107	0.9857	0.5141	198
$\Delta(data)$	0.003	63,222	0.0140	0.0030	0.0049	0.0037	0.0210	6

Table 5.15: RCIC First reference clustering- D4

5.4.8 FICFO Fair clustering

FICFO Fair comes from a very simple idea: to evenly consider every fingerprint. We always work in the assumption of few errors in the clusters, if any. The fair mean fits surely very well in a blind scenario, it does not fix any reference as a drastically more important attractor respect to the others. The more we merge a group, the more we contribute to refine the centroid location. False positives have an impact of introducing biased noise, that makes it harder to retrieve a good esteem of the reference. This error often appears in big clusters, when the threshold makes us also include some far fingerprints in the group by mistake.
Because of the rarity to have many errors in the first positions of our clusters, the impact of false positives is quite cancelled out and the cluster precision remain very high, avoiding attracting other wrong fingerprints.
Once we avoid the first errors, PRNU estimation improves steadily with the dimension of the cluster. The biggest the pure (or quite pure) group is, better it produces an effective representation similar to the ground truth.
Due to the simplicity and genericity of this algorithm, it can be applied to every dataset with very high performances and low value of $NCC < 55{,}000$. In table 5.16, we observe G=48 to 195 clusters, $R \approx 0.90$ $ARI > 0.91$ for D1 and D2 and $R \approx 0.80$ $ARI >$0.83 for D3 and D4.
As usual, the strict threshold has to be preferred to the others.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	19,381	0.9970	0.9050	0.9488	0.9941	0.9155	78
D1	0.004	18,172	0.9820	0.9030	0.9408	0.9929	0.8998	60
D1	0.003	11,907	0.7120	0.7720	0.7408	0.9609	0.5507	48
D2	0.005	21,832	0.9990	0.9140	0.9546	0.9941	0.9258	92
D2	0.004	15,882	0.9260	0.9060	0.9159	0.9846	0.8216	70
D2	0.003	12,478	0.7810	0.8200	0.8000	0.9703	0.6681	55
D3	0.005	47,966	0.9870	0.8070	0.8880	0.9946	0.8351	174
D3	0.004	41,781	0.9560	0.8120	0.8781	0.9937	0.8134	147
D3	0.003	27,225	0.7090	0.6550	0.6809	0.9813	0.4993	103
D4	0.005	54,478	0.9950	0.8020	0.8881	0.9943	0.8439	195
D4	0.004	45,241	0.9530	0.7820	0.8591	0.9918	0.7782	161
D4	0.003	27,821	0.7020	0.6320	0.6652	0.9794	0.4906	100

Table 5.16: FICFO Fair clustering

5.4.9 RCIC Fair clustering

A great advantage of the arithmetic mean proposed by FICFO Fair is the invariance property from the fingerprint order that makes Fair clustering suitable also for the RCIC algorithm. Its applicability to every scenario is the main reason why we chose the Fair clustering solution as the starting point of every attraction step we tested. Further runs of other techniques in exam show that FICFO-A and RCIC-A really exploit their full potential only in their Fair clustering form.
In tables 5.17-5.20, we observe in average G=67 to 270 clusters, $R \approx 0.80$ $ARI >$ 0.98 for D1 and D2 and $R \approx 0.70$ $ARI >$0.99 for D3 and D4.
As usual, the strict threshold has to be preferred to the others.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	30,111	0.9949	0.8022	0.8880	0.9885	0.8222	141
$Q_2(data)$	0.005	30,090	0.9980	0.8005	0.8888	0.9888	0.8266	140
$\Delta(data)$	0.005	3,148	0.0010	0.0335	0.0209	0.0014	0.0246	3
$E(data)$	0.004	22,644	0.9738	0.8308	0.8965	0.9887	0.8313	97
$Q_2(data)$	0.004	22,474	0.9795	0.8445	0.9021	0.9894	0.8414	97
$\Delta(data)$	0.004	2,309	0.0235	0.0350	0.0255	0.0021	0.0341	4
$E(data)$	0.003	15,899	0.7652	0.7282	0.7458	0.9684	0.5825	67
$Q_2(data)$	0.003	15,674	0.7705	0.7340	0.7407	0.9690	0.5773	68
$\Delta(data)$	0.003	1,569	0.0453	0.0453	0.0314	0.0032	0.0310	3

Table 5.17: RCIC Fair clustering- D1

5.4.10 FICFO Weighted clustering

FICFO Weighted tries to improve the esteem of the Fair mean by giving greater importance to the high value of ordering factor RI. Although this strategy seems a better solution, it heavily depends on the statistics of the images we are clustering. The Dresden datasets [25] do not contain many flat images. Many photos are not uniformly bright and they tend to be saturated, we do not have a perfect attractor for each group. In a real scenario, working with average-quality images ($RI \approx$ 0.50), the Fair mean establish a stronger partition of the groups, because it does not rely too much on the single images.
The table 5.22 presents some information about our source file, the RI index of each fingerprint. It shows that there are very good images in each dataset (as the maximum is $\approx$ 1) and also some very bad. Given the value G of groups in the

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	30,331	0.9960	0.7961	0.8847	0.9874	0.8261	151
$Q_2(data)$	0.005	29,681	0.9960	0.8025	0.8893	0.9877	0.8323	150
$\Delta(data)$	0.005	4,398	0.0042	0.0265	0.0149	0.0019	0.0295	5
$E(data)$	0.004	22,545	0.9728	0.8355	0.8988	0.9883	0.8462	106
$Q_2(data)$	0.004	22,144	0.9740	0.8435	0.9029	0.9882	0.8463	104
$\Delta(data)$	0.004	1,777	0.0168	0.0318	0.0224	0.0022	0.0273	9
$E(data)$	0.003	14,752	0.7816	0.7295	0.7543	0.9671	0.6107	68
$Q_2(data)$	0.003	14,373	0.7730	0.7290	0.7455	0.9662	0.5933	69
$\Delta(data)$	0.003	1,291	0.0145	0.0368	0.0217	0.0034	0.0506	10

Table 5.18: RCIC Fair clustering- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	63,031	0.9915	0.7170	0.8322	0.9924	0.7485	233
$Q_2(data)$	0.005	63,024	0.9915	0.7140	0.8304	0.9924	0.7498	235
$\Delta(data)$	0.005	4,905	0.0018	0.0135	0.0097	0.0004	0.0146	14
$E(data)$	0.004	49,766	0.9463	0.7343	0.8269	0.9917	0.7381	177
$Q_2(data)$	0.004	49,242	0.9465	0.7365	0.8275	0.9917	0.7371	176
$\Delta(data)$	0.004	2,894	0.0133	0.0200	0.0145	0.0005	0.0128	11
$E(data)$	0.003	28,449	0.6708	0.6236	0.6463	0.9790	0.4507	114
$Q_2(data)$	0.003	28,876	0.6685	0.6195	0.6435	0.9790	0.4507	113
$\Delta(data)$	0.003	1,214	0.0280	0.0320	0.0320	0.0011	0.0306	11

Table 5.19: RCIC Fair clustering- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	69,334	0.9909	0.7022	0.8219	0.9910	0.7349	268
$Q_2(data)$	0.005	69,326	0.9910	0.7020	0.8226	0.9911	0.7353	270
$\Delta(data)$	0.005	5,395	0.0025	0.0113	0.0071	0.0001	0.0055	1
$E(data)$	0.004	54,742	0.9375	0.7075	0.8063	0.9875	0.6351	270
$Q_2(data)$	0.004	54,166	0.9365	0.7155	0.8112	0.9874	0.6366	270
$\Delta(data)$	0.004	3,184	0.0172	0.0265	0.0238	0.0014	0.0451	0
$E(data)$	0.003	31,294	0.7175	0.6106	0.6595	0.9755	0.3588	270
$Q_2(data)$	0.003	31,764	0.7199	0.6105	0.6613	0.9760	0.3589	270
$\Delta(data)$	0.003	1,335	0.0446	0.0243	0.0232	0.0019	0.0364	0

Table 5.20: RCIC Fair clustering- D4

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	19,609	0.9970	0.9030	0.9477	0.9940	0.9138	80
D1	0.004	18,295	0.9880	0.9050	0.9447	0.9934	0.9065	61
D1	0.003	12,140	0.7280	0.7690	0.7479	0.9615	0.5554	50
D2	0.005	20,915	0.9990	0.9130	0.9541	0.9941	0.9250	93
D2	0.004	16,542	0.9600	0.9170	0.9380	0.9916	0.8972	69
D2	0.003	11,991	0.7640	0.7940	0.7787	0.9681	0.6425	53
D3	0.005	47,267	0.9890	0.8050	0.8876	0.9946	0.8355	176
D3	0.004	41,544	0.9660	0.8160	0.8847	0.9942	0.8264	145
D3	0.003	27,095	0.7170	0.6690	0.6922	0.9816	0.5131	102
D4	0.005	55,495	0.9940	0.7960	0.8840	0.9940	0.8371	200
D4	0.004	44,971	0.9550	0.7720	0.8538	0.9919	0.7786	161
D4	0.003	28,914	0.7110	0.6270	0.6664	0.9798	0.4936	107

Table 5.21: FICFO Weighted clustering

ground truth, we expect at least to have the attractor fingerprints AF_G (the best one for each group) with $RI(AF_G) \approx 1$.
In table 5.21, we observe G= 50 to 200 clusters, $R \approx 0.90$ $ARI > 0.91$ for D1 and D2 and $R \approx 0.80$ $ARI >$0.83 for D3 and D4.

dataset	max	min	mean	G	mean(AF_G)	var(AF_G)
D1	0.9903	0.2226	0.4730	25	0.6731	0.0157
D2	0.9936	0.1959	0.4762	25	0.7134	0.0188
D3	0.9996	0.1573	0.4777	50	0.6452	0.0148
D4	0.9996	0.1573	0.4777	50	0.6444	0.0147

Table 5.22: Ordering factor for our datasets

5.4.11 FICFO-A Fair Full Matrix (FM)

FICFO-A Fair FM works in the same way as the Full Matrix clustering, but it has the big computational advantage to start from a first partition that reduces the number correlation matrix we need to NCC from $\approx 0.4\%$ to $\approx 3\%$ of its clustering version, depending on the dataset and the threshold in exam.
In table 5.23, we observe G= 32 to 155 clusters, $R \approx 0.91$ $ARI > 0.99$ for D1 and

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	22,384	0.9747	0.9130	0.9428	0.9921	0.8912	72
D1	0.004	19,942	0.9200	0.9220	0.9210	0.9862	0.8261	52
D1	0.003	13,035	0.5130	0.8970	0.6527	0.9127	0.3904	32
D2	0.005	26,018	0.9767	0.9280	0.9517	0.9926	0.9093	78
D2	0.004	18,297	0.8780	0.9360	0.9061	0.9725	0.7288	62
D2	0.003	13,963	0.5870	0.9080	0.7130	0.9354	0.5057	38
D3	0.005	63,017	0.9736	0.8320	0.8973	0.9943	0.8333	151
D3	0.004	52,512	0.4230	0.8170	0.5574	0.9510	0.3470	52
D3	0.003	32,478	0.5620	0.7460	0.6411	0.9682	0.4085	67
D4	0.005	69,529	0.9654	0.8350	0.8955	0.9912	0.7827	155
D4	0.004	58,121	0.9040	0.8350	0.8681	0.9907	0.7744	134
D4	0.003	32,771	0.5720	0.7240	0.6391	0.9666	0.4210	64

Table 5.23: FICFO-A Fair Full matrix

D2 and $R \approx 0.83$ ARI >0.78 for D3 and D4.
The values drop very much for lower thresholds.

5.4.12 RCIC-A Fair Full Matrix (FM)

RCIC-A Fair FM is very similar to the FICFO version. From their tables 5.24-5.27 we can notice a decrements in the performances respect to the complete Full Matrix clustering, but at the same time a relaxation of the number NCC. The introduction of the parameter w_corr seems to have a very little advantage and also bin_vect shows to have a scarce impact on the results.
We observe in average G= 67 to 270 clusters, $R \approx 0.80$ $ARI > 0.98$ for D1 and D2 and $R \approx 0.70$ ARI >0.99 for D3 and D4.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	40,004	0.7341	0.7914	0.7606	0.9637	0.5786	55
$Q_2(data)$	0.005	39,544	0.7405	0.7930	0.7677	0.9632	0.5698	55
$\Delta(data)$	0.005	3,250	0.0488	0.0434	0.0269	0.0092	0.0953	5
$E(data)$	0.004	27,312	0.7662	0.6036	0.6731	0.9527	0.5297	43
$Q_2(data)$	0.004	27,287	0.7766	0.5828	0.6625	0.9516	0.5282	43
$\Delta(data)$	0.004	2,612	0.0285	0.0745	0.0540	0.0051	0.0246	4
$E(data)$	0.003	18,110	0.7387	0.5517	0.6311	0.9462	0.5178	32
$Q_2(data)$	0.003	17,952	0.7397	0.5595	0.6395	0.9472	0.5126	33
$\Delta(data)$	0.003	1,569	0.0096	0.0323	0.0237	0.0072	0.0197	4

Table 5.24: RCIC-A Fair Full Matrix- D1

5.4.13 FICFO-A Fair Hierarchical Agglomerative

Agglomerate clustering is an efficient attraction stage. It requires a first knowledge of pure clusters and it build the binary tree of distances between all the centroids, to output a valid partition of groups of groups.
The algorithm suffers the same problems of the Full Matrix, its critical step is the merging of unclustered fingerprints. The binary tree can easily group false positives way before the good neighbors of each reference. The more the groups grow, the more strong and trustable the algorithm becomes in its decisions. This is the reason why it works way better as an attraction step.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	41,650	0.6873	0.7967	0.7371	0.9517	0.5186	59
$Q_2(data)$	0.005	40,858	0.6855	0.7959	0.7302	0.9522	0.5265	57
$\Delta(data)$	0.005	5,566	0.0358	0.0300	0.0244	0.0135	0.1150	6
$E(data)$	0.004	28,083	0.6658	0.7277	0.6947	0.9364	0.4591	42
$Q_2(data)$	0.004	27,406	0.6750	0.7199	0.6853	0.9390	0.4629	43
$\Delta(data)$	0.004	2,162	0.0437	0.0609	0.0547	0.0097	0.0510	6
$E(data)$	0.003	17,051	0.6757	0.5570	0.6094	0.9336	0.5525	30
$Q_2(data)$	0.003	16,479	0.6797	0.5578	0.6146	0.9315	0.5591	29
$\Delta(data)$	0.003	1,649	0.0367	0.0732	0.0326	0.0056	0.0189	4

Table 5.25: RCIC-A Fair Full Matrix- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	90,083	0.6730	0.6868	0.6791	0.9773	0.4882	109
$Q_2(data)$	0.005	90,629	0.6682	0.7030	0.6844	0.9768	0.4856	108
$\Delta(data)$	0.005	8,704	0.0309	0.0263	0.0162	0.0021	0.0363	12
$E(data)$	0.004	65,346	0.6239	0.6430	0.6319	0.9706	0.4305	87
$Q_2(data)$	0.004	64,801	0.6194	0.6410	0.6245	0.9710	0.4153	87
$\Delta(data)$	0.004	4,392	0.0458	0.0286	0.0306	0.0028	0.0397	3
$E(data)$	0.003	34,853	0.5392	0.4842	0.5098	0.9560	0.3551	54
$Q_2(data)$	0.003	35,724	0.5457	0.4797	0.5051	0.9556	0.3573	53
$\Delta(data)$	0.003	1,506	0.0356	0.0337	0.0210	0.0028	0.0275	5

Table 5.26: RCIC-A Fair Full Matrix- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	103,080	0.6661	0.7029	0.6835	0.9737	0.4643	118
$Q_2(data)$	0.005	102,313	0.6550	0.7039	0.6780	0.9724	0.4416	118
$\Delta(data)$	0.005	2,092	0.0447	0.0092	0.0255	0.0030	0.0440	4
$E(data)$	0.004	73,842	0.5612	0.6462	0.5991	0.9641	0.3586	90
$Q_2(data)$	0.004	74,504	0.5614	0.6640	0.5963	0.9640	0.3631	92
$\Delta(data)$	0.004	1,909	0.0374	0.0712	0.0261	0.0019	0.0247	6
$E(data)$	0.003	38,043	0.5347	0.4943	0.5114	0.9538	0.3559	56
$Q_2(data)$	0.003	38,760	0.5247	0.4726	0.5040	0.9536	0.3517	56
$\Delta(data)$	0.003	1,947	0.0551	0.0740	0.0173	0.0051	0.0775	3

Table 5.27: RCIC-A Fair Full Matrix- D4

The presence of many unclustered left by the first step of FICFO and RCIC have still a big impact on Fair Hierarchical Agglomerative. It works very well in finding the last fingerprint left outside big groups, but at the same times, it usually starts to merge middle and little groups from different cameras before finishing to search for true positives to regroup. In table 5.28, we can extract the values of

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	19,529	0.9650	0.9120	0.9378	0.9921	0.8910	72
D1	0.004	18,279	0.8920	0.9240	0.9077	0.9868	0.8331	48
D1	0.003	11,988	0.5790	0.8800	0.6985	0.9397	0.4823	34
D2	0.005	22,013	0.9990	0.9230	0.9595	0.9948	0.9347	83
D2	0.004	16,012	0.9160	0.9350	0.9254	0.9830	0.8148	64
D2	0.003	12,574	0.6110	0.9010	0.7282	0.9425	0.5351	38
D3	0.005	48,356	0.9690	0.8280	0.8930	0.9945	0.8373	155
D3	0.004	42,100	0.4220	0.8170	0.5565	0.9510	0.3470	51
D3	0.003	27,433	0.5540	0.7420	0.6344	0.9679	0.4054	71
D4	0.005	54,925	0.5840	0.8070	0.6776	0.9634	0.4495	86
D4	0.004	45,597	0.8991	0.8292	0.8626	0.9902	0.7629	83
D4	0.003	28,021	0.5750	0.7150	0.6374	0.9683	0.4313	62

Table 5.28: FICFO-A Fair Hierarchical Agglomerative

precision for each dataset: 0.9650, 0.9990, 0.9690, 0.5840.
The big difference in the last two clusters depends by the fact that the algorithm is more cautious over D3, producing a partition of 155 clusters, while in D4 it obtains G= 86. For similar numbers of groups, as in case of the relaxed case, D3 and D4 perform very similar, as expected.
As the algorithm is implemented now, we cannot come up with any threshold able to work at the perfect height, because of the wrong order of attractions it usually follows.
As a check, we plot every diagram of our tables, computed with the Average-Linkage Criterion, and we notice that all of them are legit dendrograms, where the group distance vector is monotonically increasing with the height, as in the example figure 2.10. $R > 0.91$ $ARI > 0.99$ for D1 and D2 and $R \approx 0.83$ $ARI \approx 0.84$ for D3. The values drop very much for lower thresholds and on the last dataset.

5.4.14 RCIC-A Fair Hierarchical Agglomerative

All the considerations about the equivalent FICFO version are valid as well for RCIC. Starting from RCIC we obtain an efficient attraction stage in terms of computational complexity: $\mathcal{O}\left(G^2 log(G)\right)$ originated from G groups. From their tables 5.29-5.32 we can notice a decrements in the performances respect to the complete Full Matrix clustering, but at the same time a relaxation of the number $NCC <$ 73,500. We observe in average G= 35 to 151 clusters, $R \approx 0.80$ $ARI >$ 0.64 for D1 and D2 and $R > 0.70$ $ARI > 0.58$ for D3 and D4.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	31,919	0.8074	0.8012	0.8035	0.9737	0.6674	76
$Q_2(data)$	0.005	31,893	0.8145	0.7920	0.8078	0.9738	0.6749	76
$\Delta(data)$	0.005	3,442	0.0483	0.0317	0.0230	0.0061	0.0505	3
$E(data)$	0.004	24,968	0.7448	0.8336	0.7853	0.9663	0.6236	50
$Q_2(data)$	0.004	24,786	0.7410	0.8465	0.7927	0.9684	0.6331	50
$\Delta(data)$	0.004	2,518	0.0760	0.0375	0.0412	0.0085	0.0617	4
$E(data)$	0.003	17,819	0.6053	0.7340	0.6625	0.9505	0.4679	36
$Q_2(data)$	0.003	17,574	0.6115	0.7405	0.6593	0.9508	0.4614	36
$\Delta(data)$	0.003	1,719	0.0508	0.0415	0.0301	0.0087	0.0340	4

Table 5.29: RCIC-A Fair Hierarchical Agglomerative- D1

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	32,173	0.8024	0.7980	0.7997	0.9671	0.6468	81
$Q_2(data)$	0.005	31,498	0.7948	0.7950	0.8038	0.9687	0.6563	81
$\Delta(data)$	0.005	4,769	0.0060	0.0270	0.0203	0.0078	0.0681	3
$E(data)$	0.004	23,885	0.7472	0.8308	0.7859	0.9628	0.6262	55
$Q_2(data)$	0.004	23,464	0.7287	0.8380	0.7796	0.9645	0.6386	54
$\Delta(data)$	0.004	1,968	0.0783	0.0180	0.0343	0.0122	0.0689	8
$E(data)$	0.003	15,611	0.5806	0.7102	0.6382	0.9260	0.3824	35
$Q_2(data)$	0.003	15,215	0.5916	0.7030	0.6554	0.9340	0.4298	36
$\Delta(data)$	0.003	1,422	0.0726	0.0330	0.0443	0.0298	0.1218	2

Table 5.30: RCIC-A Fair Hierarchical Agglomerative- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	66,735	0.7744	0.7116	0.7413	0.9841	0.5810	140
$Q_2(data)$	0.005	66,727	0.7718	0.7140	0.7442	0.9847	0.5916	138
$\Delta(data)$	0.005	5,390	0.0110	0.0080	0.0049	0.0023	0.0294	12
$E(data)$	0.004	52,656	0.6994	0.7272	0.7125	0.9801	0.5334	95
$Q_2(data)$	0.004	52,099	0.6937	0.7190	0.7180	0.9795	0.5331	93
$\Delta(data)$	0.004	3,210	0.0653	0.0295	0.0311	0.0031	0.0429	10
$E(data)$	0.003	30,103	0.5108	0.6126	0.5570	0.9679	0.3372	65
$Q_2(data)$	0.003	30,555	0.5125	0.6100	0.5576	0.9675	0.3397	66
$\Delta(data)$	0.003	1,385	0.0058	0.0265	0.0140	0.0023	0.0172	3

Table 5.31: RCIC-A Fair Hierarchical Agglomerative- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	73,452	0.7825	0.7074	0.7429	0.9828	0.5985	151
$Q_2(data)$	0.005	73,449	0.7828	0.7060	0.7485	0.9835	0.6006	149
$\Delta(data)$	0.005	5,935	0.0540	0.0100	0.0200	0.0031	0.0490	4
$E(data)$	0.004	58,136	0.7233	0.7224	0.7223	0.9774	0.5352	107
$Q_2(data)$	0.004	57,531	0.7528	0.7220	0.7364	0.9815	0.5796	109
$\Delta(data)$	0.004	3,613	0.0784	0.0070	0.0364	0.0106	0.1149	11
$E(data)$	0.003	33,515	0.5142	0.6149	0.5595	0.9641	0.3465	64
$Q_2(data)$	0.003	34,008	0.5140	0.6195	0.5618	0.9637	0.3563	64
$\Delta(data)$	0.003	1,872	0.0455	0.0263	0.0370	0.0046	0.0541	4

Table 5.32: RCIC-A Fair Hierarchical Agglomerative- D4

5.4.15 FICFO-A Fair Static Pure

The purification step, in both its forms, static or dynamic, introduces a huge complexity (up to 250,000) during the attraction phase, in the order the NCC of Bloy. The groups in input are usually quite precise as they are computed by FICFO and RCIC, especially in their Fair version.
With the values of cost and performances obtained in tables 5.33-5.42, we decided not to consider those partitions as a pre-attraction step for a further attraction.
FICFO-A Fair Static Pure and RCIC-A Fair Static Pure have to check the full set of fingerprints and they split the partition to a huge amount of groups, very hard to work with. In table 5.33, we observe in average G= 48 to 195 clusters, $R > 0.78$ $ARI > 0.81$ for D1 and D2 and $R \approx 0.70$ $ARI > 0.73$ for D3 and D4.
The values drop very much for lower thresholds.

5.4.16 RCIC-A Fair Static Pure

RCIC-A Fair Static Pure suffers from the same problems of the method applied to FICFO. RCIC generates more errors due to the random order of regrouping. Indeed, in tables 5.34-5.37, we observe G= 66 to 260 clusters, $R \approx 0.90$ $ARI >$ 0.91 for D1 and D2 and $R \approx 0.80$ $ARI > 0.83$ for D3 and D4.
The values drop very much for lower thresholds.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	84,084	0.9970	0.9050	0.9488	0.9941	0.9155	78
D1	0.004	78,172	0.9890	0.9100	0.9479	0.9938	0.9115	60
D1	0.003	71,907	0.7510	0.7800	0.7652	0.9680	0.6078	48
D2	0.005	112,832	0.9990	0.9130	0.9541	0.9941	0.9252	92
D2	0.004	85,882	0.9260	0.9060	0.9159	0.9846	0.8216	70
D2	0.003	67,478	0.7810	0.8200	0.8000	0.9703	0.6681	55
D3	0.005	220,966	0.9870	0.8070	0.8880	0.9946	0.8351	174
D3	0.004	188,781	0.9560	0.8120	0.8781	0.9937	0.8134	147
D3	0.003	130,225	0.7090	0.6550	0.6809	0.9813	0.4993	103
D4	0.005	249,478	0.9950	0.8020	0.8881	0.9943	0.8439	195
D4	0.004	205,241	0.9530	0.7820	0.8591	0.9918	0.7782	161
D4	0.003	127,821	0.7030	0.6320	0.6656	0.9795	0.4916	100

Table 5.33: FICFO-A Fair Static pure

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	139,200	0.9982	0.8040	0.8906	0.9886	0.8247	139
$Q_2(data)$	0.005	140,000	0.9980	0.8090	0.8944	0.9889	0.8290	140
$\Delta(data)$	0.005	4,159	0.0012	0.0210	0.0129	0.0006	0.0111	4
$E(data)$	0.004	93,333	0.9880	0.8540	0.9161	0.9906	0.8607	93
$Q_2(data)$	0.004	92,000	0.9890	0.8520	0.9154	0.9904	0.8575	92
$\Delta(data)$	0.004	4,200	0.0063	0.0112	0.0092	0.0008	0.0121	4
$E(data)$	0.003	66,333	0.7660	0.7290	0.7469	0.9685	0.5851	66
$Q_2(data)$	0.003	67,000	0.7850	0.7280	0.7554	0.9688	0.5859	67
$\Delta(data)$	0.003	4,200	0.0399	0.0077	0.0232	0.0032	0.0286	4

Table 5.34: RCIC-A Fair Static pure- D1

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	152,600	0.9926	0.7874	0.8781	0.9867	0.8166	153
$Q_2(data)$	0.005	154,000	0.9960	0.7950	0.8846	0.9868	0.8172	154
$\Delta(data)$	0.005	5,914	0.0020	0.0240	0.0146	0.0008	0.0135	6
$E(data)$	0.004	104,667	0.9810	0.8260	0.8968	0.9883	0.8438	105
$Q_2(data)$	0.004	102,000	0.9790	0.8320	0.8995	0.9883	0.8463	102
$\Delta(data)$	0.004	9,800	0.0056	0.0140	0.0060	0.0011	0.0185	10
$E(data)$	0.003	66,333	0.7957	0.7120	0.7514	0.9671	0.5970	66
$Q_2(data)$	0.003	67,000	0.8000	0.7130	0.7514	0.9683	0.6074	67
$\Delta(data)$	0.003	4,200	0.0245	0.0161	0.0115	0.0036	0.0389	4

Table 5.35: RCIC-A Fair Static pure- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	240,200	0.9920	0.7092	0.8271	0.9922	0.7390	240
$Q_2(data)$	0.005	240,000	0.9910	0.7110	0.8283	0.9923	0.7417	240
$\Delta(data)$	0.005	6,154	0.0010	0.0100	0.0065	0.0003	0.0127	6
$E(data)$	0.004	176,000	0.9263	0.7213	0.8110	0.9908	0.7119	176
$Q_2(data)$	0.004	174,000	0.9310	0.7160	0.8095	0.9911	0.7164	174
$\Delta(data)$	0.004	4,200	0.0196	0.0168	0.0145	0.0008	0.0227	4
$E(data)$	0.003	115,000	0.6610	0.6057	0.6319	0.9788	0.4363	115
$Q_2(data)$	0.003	117,000	0.6500	0.5980	0.6343	0.9787	0.4318	117
$\Delta(data)$	0.003	5,600	0.0273	0.0175	0.0116	0.0004	0.0099	6

Table 5.36: RCIC-A Fair Static pure- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	260,400	0.9924	0.7022	0.8224	0.9915	0.7520	260
$Q_2(data)$	0.005	261,000	0.9930	0.6980	0.8198	0.9916	0.7551	261
$\Delta(data)$	0.005	4,102	0.0050	0.0130	0.0089	0.0003	0.0128	4
$E(data)$	0.004	195,333	0.9357	0.7197	0.8135	0.9900	0.7201	195
$Q_2(data)$	0.004	197,000	0.9420	0.7200	0.8162	0.9904	0.7309	197
$\Delta(data)$	0.004	10,500	0.0315	0.0049	0.0151	0.0013	0.0295	11
$E(data)$	0.003	118,000	0.6850	0.6073	0.6438	0.9781	0.4636	118
$Q_2(data)$	0.003	121,000	0.6850	0.5950	0.6368	0.9780	0.4523	121
$\Delta(data)$	0.003	9,100	0.0476	0.0511	0.0497	0.0019	0.0522	9

Table 5.37: RCIC-A Fair Static pure- D4

5.4.17 FICFO-A Fair Dynamic Pure

FICFO-A Fair Dynamic Pure was created with the aim of updating the centroid while moving the fingerprint, for a more robust comparison and a correct re-clustering. It also avoids splitting any group. On the contrary, it should clean the error from the current group and trash the empty clusters (after relocating all the item previously present in).

This algorithm has the power to merge little groups and unclustered images, while finding errors and correcting them. The more the algorithm runs, the more it is sure about the decisions it takes. We can even think of running the code more than once, taking the first output as a partition still to further check, because after the first run we should have a better understanding of the position of the centroids. The complexity is very similar to the Static approach, due to the need of checking every fingerprint and comparing to every other group.

In our experiments, we changed the sequence in which groups are considered from the same ordering as the FICFO Fair output to the list sorted by decreasing order of cluster dimension.
It is logical to think that if we clear the errors from the big groups at the beginning, we are in presence of a great solid starting point to merge other images in those big pure groups and to reject false positives.
Experiments shows that this consideration is not true on our datasets, conversely, the performances get worse. We tried also to sort groups in increasing order of dimensions. Of the three options, this performs way worse, as expected.

The problem is the imprecision of the algorithm to re-assign items. Too many fingerprints are moved, usually to wrong clusters. This makes us often retrieve a worse cluster partition respect to the one we started from. Again, the huge complexity (up to NCC= 255,617) of one or more runs of this procedure make it unfeasible for a pre-attraction step. In table 5.38, we observe G= 17 to 63 clusters, way less than the static approach, we notice some performances drops, as $R \approx 0.61$ and $ARI \approx 0.44$ in D2.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	79,682	0.9430	0.8840	0.9125	0.9912	0.8760	32
D1	0.004	59,324	0.9440	0.8980	0.9204	0.9910	0.8756	30
D1	0.003	67,799	0.4140	0.5590	0.4757	0.9303	0.2916	17
D2	0.005	129,775	0.5680	0.6110	0.5887	0.9513	0.4361	23
D2	0.004	93,912	0.5720	0.6440	0.6059	0.9516	0.4694	23
D2	0.003	79,421	0.4580	0.4990	0.4776	0.9432	0.3234	24
D3	0.005	189,094	0.8060	0.7730	0.7892	0.9899	0.7179	60
D3	0.004	154,996	0.8210	0.7690	0.7941	0.9903	0.7261	63
D3	0.003	133,291	0.3860	0.5440	0.4516	0.9625	0.2762	30
D4	0.005	255,617	0.8300	0.7850	0.8069	0.9907	0.7646	62
D4	0.004	201,201	0.5940	0.6080	0.6009	0.9778	0.4644	49
D4	0.003	129,595	0.4000	0.5070	0.4472	0.9628	0.2648	33

Table 5.38: FICFO-A Fair Dynamic pure

5.4.18 RCIC-A Fair Dynamic Pure

As for many other codes, also in RCIC-A Fair Dynamic Pure we can clearly see the parallelism with its FICFO version. The algorithm performs the same operations and output similar wrong conclusions. Indeed, in tables 5.39-5.42, we observe in average G= 29 to 160 clusters, $R \approx 0.81$ $ARI > 0.81$ for D1 and D2 and $R > 0.71$ $ARI < 0.75$ for D3 and D4.
The values drop very much for lower thresholds.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	141,984	0.9158	0.8162	0.8631	0.9877	0.8181	39
$Q_2(data)$	0.005	142,800	0.9156	0.8213	0.8667	0.9879	0.8224	40
$\Delta(data)$	0.005	4,080	0.0022	0.0213	0.0120	0.0006	0.0110	4
$E(data)$	0.004	95,200	0.9064	0.8670	0.8863	0.9896	0.8538	41
$Q_2(data)$	0.004	93,840	0.9073	0.8650	0.8857	0.9894	0.8506	40
$\Delta(data)$	0.004	4,284	0.0058	0.0114	0.0087	0.0008	0.0120	2
$E(data)$	0.003	67,660	0.7028	0.7401	0.7208	0.9675	0.5804	29
$Q_2(data)$	0.003	68,340	0.7202	0.7391	0.7295	0.9678	0.5812	29
$\Delta(data)$	0.003	4,284	0.0366	0.0078	0.0232	0.0032	0.0283	2

Table 5.39: RCIC-A Fair Dynamic pure- D1

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	152,796	0.9145	0.8047	0.8560	0.9863	0.8145	50
$Q_2(data)$	0.005	151,980	0.9147	0.8071	0.8575	0.9862	0.8142	49
$\Delta(data)$	0.005	6,120	0.0018	0.0091	0.0039	0.0010	0.0065	6
$E(data)$	0.004	119,340	0.8991	0.8228	0.8590	0.9866	0.8258	48
$Q_2(data)$	0.004	109,650	0.8977	0.8350	0.8669	0.9869	0.8311	47
$\Delta(data)$	0.004	7,854	0.0045	0.0135	0.0052	0.0011	0.0183	3
$E(data)$	0.003	75,990	0.7718	0.7536	0.7624	0.9715	0.6541	32
$Q_2(data)$	0.003	69,360	0.7390	0.7289	0.7305	0.9677	0.6086	30
$\Delta(data)$	0.003	4,284	0.0225	0.0092	0.0064	0.0036	0.0301	2

Table 5.40: RCIC-A Fair Dynamic pure- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	245,004	0.9101	0.7200	0.8039	0.9912	0.7330	140
$Q_2(data)$	0.005	244,800	0.9092	0.7218	0.8051	0.9913	0.7358	140
$\Delta(data)$	0.005	6,120	0.0009	0.0102	0.0059	0.0003	0.0126	6
$E(data)$	0.004	179,520	0.8498	0.7323	0.7867	0.9898	0.7062	77
$Q_2(data)$	0.004	177,480	0.8541	0.7269	0.7857	0.9901	0.7107	76
$\Delta(data)$	0.004	4,284	0.0180	0.0171	0.0140	0.0008	0.0225	2
$E(data)$	0.003	117,300	0.6064	0.6149	0.6104	0.9778	0.4328	50
$Q_2(data)$	0.003	119,340	0.5963	0.6071	0.6119	0.9777	0.4283	51
$\Delta(data)$	0.003	5,712	0.0250	0.0178	0.0118	0.0004	0.0099	2

Table 5.41: RCIC-A Fair Dynamic pure- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	265,608	0.9105	0.7129	0.7996	0.9905	0.7459	160
$Q_2(data)$	0.005	266,220	0.9110	0.7086	0.7972	0.9906	0.7491	161
$\Delta(data)$	0.005	4,080	0.0046	0.0132	0.0082	0.0003	0.0127	4
$E(data)$	0.004	199,240	0.8584	0.7306	0.7893	0.9890	0.7143	85
$Q_2(data)$	0.004	200,940	0.8642	0.7310	0.7920	0.9894	0.7251	86
$\Delta(data)$	0.004	10,710	0.0289	0.0050	0.0152	0.0013	0.0293	5
$E(data)$	0.003	120,360	0.6284	0.6166	0.6224	0.9771	0.4599	51
$Q_2(data)$	0.003	123,420	0.6284	0.6041	0.6160	0.9770	0.4487	53
$\Delta(data)$	0.003	9,282	0.0437	0.0519	0.0478	0.0019	0.0518	4

Table 5.42: RCIC-A Fair Dynamic pure- D4

5.4.19 FICFO-A Fair Khan attraction

We tested the Khan attraction on our FICFO Fair cluster partition. We see in table 5.43 that Khan attraction tries to re-order the groups, merging some of them and splitting others.

In our experiments, we always obtained the same number of groups G (between 48 and 195) that we started from, by chance. On every occasion, Khan attraction is able to enhance the recall value, $R>$ 0.91 in the easy datasets and $R>$ 0.82 in D3 and D4, but it drops all the other metrics. In particular, P and ARI suffer a lot (D2 presents $ARI=$ 0.4846 for $Th=$ 0.0050) for the big new groups present in the output (even two times bigger than what expected in the ground truth).
We can conclude that FICFO-A Fair Khan produces quite low values of NCC for the easy datasets but, for sure, it does not represent a suitable alternative.

dataset	Th	NCC	P	R	F	RI	ARI	G
D1	0.005	25,387	0.8981	0.9190	0.9084	0.9658	0.5618	78
D1	0.004	21,712	0.7871	0.9300	0.8526	0.9566	0.4479	60
D1	0.003	14,163	0.3487	0.9130	0.5047	0.9218	0.1452	48
D2	0.005	30,204	0.9589	0.9310	0.9447	0.9535	0.4846	92
D2	0.004	20,712	0.7446	0.9380	0.8302	0.9314	0.3703	70
D2	0.003	15,448	0.4541	0.9190	0.6078	0.9223	0.1889	55
D3	0.005	78,068	0.9241	0.8350	0.8773	0.9850	0.5693	174
D3	0.004	63,243	0.7662	0.8370	0.8000	0.9797	0.4703	147
D3	0.003	37,731	0.3660	0.7820	0.4986	0.9649	0.1441	103
D4	0.005	147,786	0.9543	0.8290	0.8873	0.9875	0.6564	195
D4	0.004	71,001	0.7914	0.8450	0.8173	0.9724	0.3900	161
D4	0.003	37,721	0.4073	0.7620	0.5309	0.9642	0.1614	100

Table 5.43: FICFO-A Fair Khan attraction

5.4.20 RCIC-A Fair Khan attraction

RCIC-A Fair Khan has the same attitude of the FICFO version for cleaning the groups, joining some of them and splitting other clusters.
As the tables 5.44-5.47 present, this time Khan is able to increase every metrics, at least for the strict threshold of $Th = 0.005$, because of the higher number of groups it starts from. It merges little pure clusters, without correcting the errors

but at least without introducing new false positives. P and ARI receive a strong boost from the good work done in reducing the number of groups. In D4 P reaches 0.9762 and ARI= 0.8253.
Khan attraction is a very cheap algorithm, in terms of NCC computations. It introduces no more than 6,000 new cross-correlations in easy datasets and less than 21,000 in hard datasets. Therefore, on RCIC-A Fair it deserves to be compared with the best algorithms.

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	35,147	0.9980	0.9020	0.9476	0.9926	0.9039	92
$Q_2(data)$	0.005	35,275	0.9965	0.9160	0.9546	0.9931	0.9008	92
$\Delta(data)$	0.005	4,770	0.0253	0.0095	0.0138	0.0014	0.0187	9
$E(data)$	0.004	23,739	0.9210	0.9103	0.9153	0.9882	0.8458	58
$Q_2(data)$	0.004	22,925	0.9320	0.9130	0.9173	0.9889	0.8515	58
$\Delta(data)$	0.004	3,265	0.0790	0.0120	0.0405	0.0066	0.0744	2
$E(data)$	0.003	16,142	0.6060	0.8063	0.6906	0.9422	0.4615	38
$Q_2(data)$	0.003	15,912	0.6120	0.8090	0.6968	0.9404	0.4525	39
$\Delta(data)$	0.003	1,701	0.1360	0.0080	0.0921	0.0133	0.0631	4

Table 5.44: RCIC-A Fair Khan attraction- D1

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	35,672	0.9990	0.9000	0.9469	0.9922	0.9189	99
$Q_2(data)$	0.005	35,275	0.9955	0.8980	0.9442	0.9924	0.9135	101
$\Delta(data)$	0.005	4,770	0.0253	0.0095	0.0119	0.0024	0.0270	6
$E(data)$	0.004	25,957	0.9300	0.8980	0.9137	0.9867	0.8434	69
$Q_2(data)$	0.004	25,957	0.9240	0.9030	0.9134	0.9858	0.8354	72
$\Delta(data)$	0.004	5,967	0.0380	0.0310	0.0343	0.0070	0.0752	15
$E(data)$	0.003	16,031	0.5553	0.8010	0.6555	0.9251	0.4198	35
$Q_2(data)$	0.003	15,653	0.5380	0.8020	0.6541	0.9269	0.4195	36
$\Delta(data)$	0.003	2,810	0.0520	0.0670	0.0475	0.0109	0.0051	3

Table 5.45: RCIC-A Fair Khan attraction- D2

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	79,554	0.9810	0.8060	0.8849	0.9940	0.8260	173
$Q_2(data)$	0.005	79,832	0.9825	0.8020	0.8831	0.9941	0.8231	175
$\Delta(data)$	0.005	4,475	0.0193	0.0143	0.0106	0.0005	0.0156	10
$E(data)$	0.004	57,848	0.8353	0.7977	0.8160	0.9878	0.6826	131
$Q_2(data)$	0.004	58,599	0.8430	0.7990	0.8225	0.9879	0.6864	130
$\Delta(data)$	0.004	4,284	0.0430	0.0120	0.0247	0.0020	0.0387	10
$E(data)$	0.003	31,984	0.4423	0.6913	0.5384	0.9511	0.2848	67
$Q_2(data)$	0.003	33,019	0.4630	0.6830	0.5493	0.9536	0.2873	67
$\Delta(data)$	0.003	3,819	0.0620	0.0410	0.0378	0.0142	0.0474	8

Table 5.46: RCIC-A Fair Khan attraction- D3

value	Th	NCC	P	R	F	RI	ARI	G
$E(data)$	0.005	90,374	0.9762	0.7972	0.8776	0.9935	0.8253	191
$Q_2(data)$	0.005	89,990	0.9770	0.7920	0.8742	0.9936	0.8256	192
$\Delta(data)$	0.005	4,541	0.0090	0.0130	0.0133	0.0005	0.0184	5
$E(data)$	0.004	63,315	0.8163	0.8013	0.8087	0.9851	0.6683	134
$Q_2(data)$	0.004	63,469	0.8210	0.8000	0.8104	0.9858	0.6765	132
$\Delta(data)$	0.004	9,143	0.0480	0.0080	0.0277	0.0056	0.0904	6
$E(data)$	0.003	32,369	0.5163	0.7017	0.5940	0.9521	0.3267	67
$Q_2(data)$	0.003	32,145	0.4870	0.7010	0.5747	0.9473	0.2842	68
$\Delta(data)$	0.003	1,505	0.1360	0.0600	0.1106	0.0197	0.1302	4

Table 5.47: RCIC-A Fair Khan attraction- D4

5.5 Comparison with State of the art

The best versions of the algorithms, with and without attraction (Full Matrix, FICFO Fair, RCIC Fair, FICFO-A Fair Full Matrix and RCIC-A Fair Khan), are compared with the State of the art methods described in chapter 3.
All results in tables 5.48-5.51 are obtained for $Th = 0.0050$, the strict threshold that often provides good clustering performances. The attraction phase raises the threshold to $Th\_att = 0.0063$, with a static empirically increase of ρ= 1.25. The parameter γ is set to 0.10 when needed.

Our experiments reveal (also in images 5.1-5.4) that some instances of FICFO-A Fair and RCIC-A Fair may be used as fast, low complexity codes for image clustering. This is particularly true in a scenario in which we have access to a huge number of big sized fingerprints with a good ordering factor RI. Once we decrease the crop size or we reach a hard $NC \gg SC$ problem, it starts to appear a non-negligible presence of false positives in our groups. Errors are always likely to drag many others with them and to drop the algorithm performances.
Working with many cameras of the same brand and model, it usually happens a small portion of their fingerprints is attracted by a wrong cluster, so little pure clusters hypothesis is not always true. This fact decreases the recall and affect a bit also the precision of the first phase. Therefore, it increases the presence of some false positives in our clusters (leading to low precision) or it creates many more groups than what the ground truth suggests (decreasing the recall).

Comparing different clustering algorithms, we can show that both working with RCIC and FICFO, the Fixed Reference clustering performs always worse respect to the other codes for the clustering phase. Even in FICFO, where the first fingerprint has the highest value of reliability, it is not wise to fix the first attractor as the centroid of the cluster. Instead, it works better to compute a mean of all the fingerprints we merged in the cluster and obtain the centroid from all of them, to average the error and possibly remove the noise components.

The Weighted version gives quite the same performances of the average, but it is based on the assumption of some very high-quality image present in the dataset, therefore we prefer not to use it as a base for attraction, also because it is not compatible with the RCIC algorithm.
Further investigations on FICFO-A Weighted shows that we never reach better performances than the corresponding FICFO-A Fair alternative.

The results on Full Matrix and FICFO Fair, with and without attraction, are very promising. In particular, FICFO Fair seems to be the winning choice. It combines a very strong value of ARI while maintaining a very low complexity approach for all the datasets.

method	NCC	P	R	F	RI	ARI
Full Matrix (FM)	499,500	1,0000	0.9300	0.9637	0.9959	0.9421
Bloy	72,671	1,0000	0.9050	0.9501	0.9975	0.9656
FICFO base	28,818	0.9990	0.8220	0.9019	0.9891	0.8333
RCIC base	27,406	0.9910	0.5490	0.7066	0.9765	0.5610
FICFO Fair	19,381	0.9970	0.9050	0.9488	0.9941	0.9155
RCIC Fair	30,090	0.9980	0.8005	0.8888	0.9888	0.8266
FICFO-A Fair FM	22,384	0.9747	0.9130	0.9428	0.9921	0.8912
RCIC-A Fair Khan	35,147	0.9980	0.9020	0.9347	0.9934	0.9039

Table 5.48: Dataset D1: Comparison with State of the art

method	NCC	P	R	F	RI	ARI
Full Matrix (FM)	499,500	0.9990	0.9430	0.9702	0.9964	0.9555
Bloy	41,790	0.9957	0.8910	0.9405	0.9978	0.9738
FICFO base	29,951	0.9980	0.8320	0.9075	0.9890	0.8517
RCIC base	27,893	0.9940	0.6000	0.7483	0.9756	0.6089
FICFO Fair	21,832	0.9990	0.9140	0.9546	0.9941	0.9258
RCIC Fair	29,681	0.9960	0.8025	0.8893	0.9877	0.8323
FICFO-A Fair FM	26,018	0.9767	0.9280	0.9515	0.9926	0.9093
RCIC-A Fair Khan	35,672	0.9990	0.9000	0.9469	0.9936	0.9189

Table 5.49: Dataset D2: Comparison with State of the art

The output metric values of FICFO Fair are always a bit lower than the respective Bloy or Full Matrix, but we must consider that a greedy algorithm as FICFO Fair can handle just approximations of the ground truth. Without a full knowledge of all the fingerprints at once, it is impossible to compute an optimal partition.

If we relax the problem and we do not account for the computational complexity, Full Matrix offers an extremely accurate solution for easy datasets, while Bloy seems even a bit better in facing the challenge of clustering on hard datasets. The scarcity of information about the cameras has a strong impact on the Full matrix. It should provide the theoretical optimal solution, but it is very dependent on the correlation values it pre-computes. So, the less statistics it can extract from the fingerprints, the less of a precise centroid it can assign for the groups.

The use of the bin_vect rule, as a binary mask of length, appears to have benefits over all the datasets in the base version of the algorithms, while it does

not seem to give a clear advantage as an attraction after FICFO.

method	NCC	P	R	F	RI	ARI
Full Matrix (FM)	499,500	0.9790	0.8370	0.9024	0.9952	0.8567
Bloy	232,105	0.9845	0.8150	0.8918	0.9970	0.9180
FICFO base	53,121	0.9880	0.7730	0.8674	0.9935	0.7954
RCIC base	55,535	0.9800	0.5720	0.7224	0.9890	0.5940
FICFO Fair	47,966	0.9870	0.8070	0.8880	0.9946	0.8351
RCIC Fair	63,024	0.9915	0.7140	0.8304	0.9924	0.7498
FICFO-A Fair FM	63,017	0.9736	0.8320	0.8973	0.9943	0.8333
RCIC-A Fair Khan	79,554	0.9810	0.8060	0.8849	0.9943	0.8260

Table 5.50: Dataset D3: Comparison with State of the art

method	NCC	P	R	F	RI	ARI
Full Matrix (FM)	499,500	0.9920	0.8330	0.9056	0.9951	0.8696
Bloy	305,894	0.9910	0.8180	0.8962	0.9980	0.9530
FICFO base	61,676	0.9910	0.7560	0.8577	0.9924	0.7828
RCIC base	61,934	0.9770	0.5430	0.6980	0.9869	0.5615
FICFO Fair	54,478	0.9950	0.8020	0.8881	0.9943	0.8439
RCIC Fair	69,326	0.9910	0.7020	0.8226	0.9911	0.7353
FICFO-A Fair FM	69,529	0.9654	0.8350	0.8955	0.9912	0.7827
RCIC-A Fair Khan	89,990	0.9770	0.7920	0.8742	0.9936	0.8256

Table 5.51: Dataset D4: Comparison with State of the art

RCIC appears to be a good fast alternative respect to the other algorithms, nevertheless is has always worse initial performances respect to FICFO and it is not very suitable for an attraction phase. Indeed, the number of computations increases a lot and overall, we have a significant degradation of the performances. We recommend working with RCIC Fair, as an improvement from RCIC Khan algorithm, but not to rely on the attraction phase performed by the RCIC-A Fair versions.
For the same dataset and imposed parameters, RCIC-A performs always worse respect to the simpler FICFO Fair, plus we must take into account the stochastic behavior of RCIC that make us always question about the real accuracy of our results. If we need to execute each RCIC code many times to obtain a stable

average of the clusters, we must multiply the computational complexity times the number of runs we tested, receiving a big disadvantage respect to the deterministic algorithm.
The purification step, in both its forms, static or dynamic, introduces a non-negligible complexity to the attraction phase. The inputs are usually quite precise as they are computed by FICFO and RCIC, especially in their Fair version. In conclusion, the pure algorithms perform bad both as an attraction phase and as a pre-attraction step. They have to check the full set of fingerprints and they re-assign too many of them, usually to wrong clusters or they split the partition to a huge amount of groups, very hard to work with.

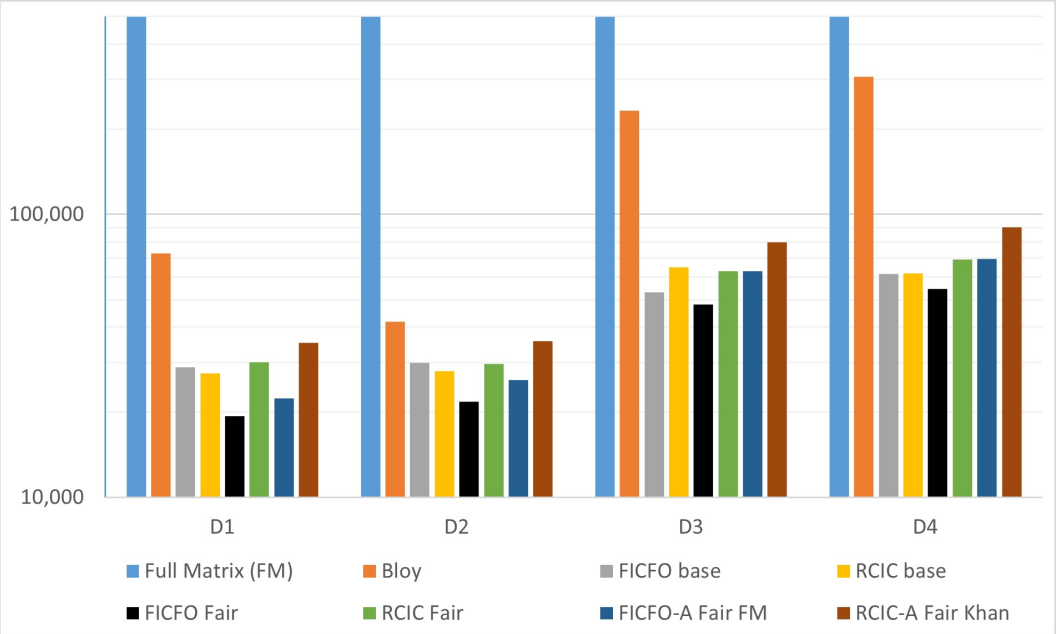

Figure 5.1: Comparison plot NCC

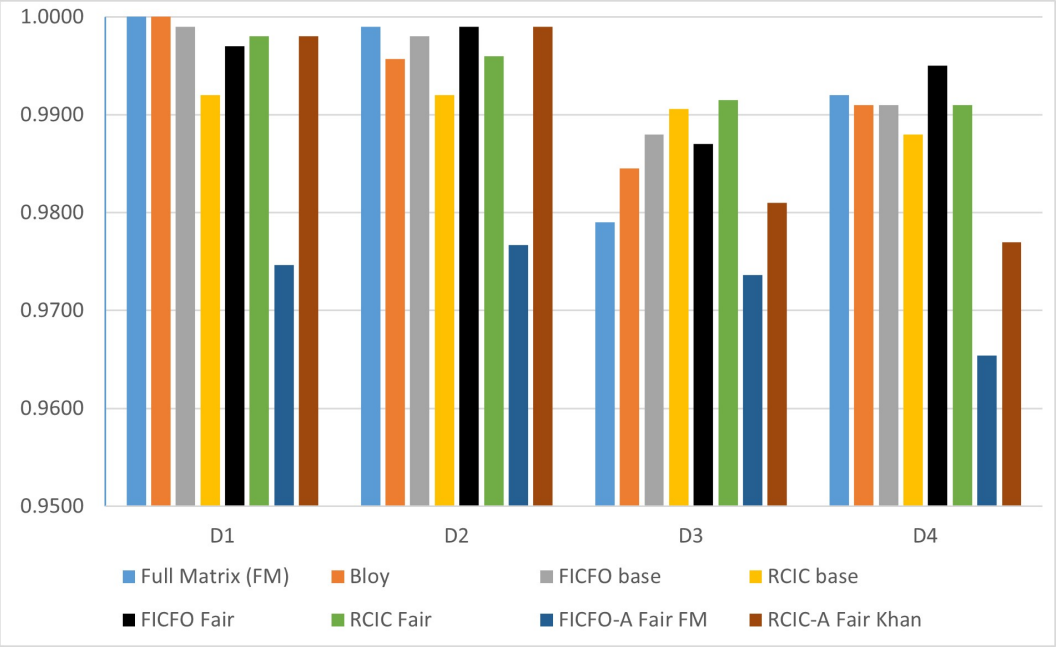

Figure 5.2: Comparison plot P

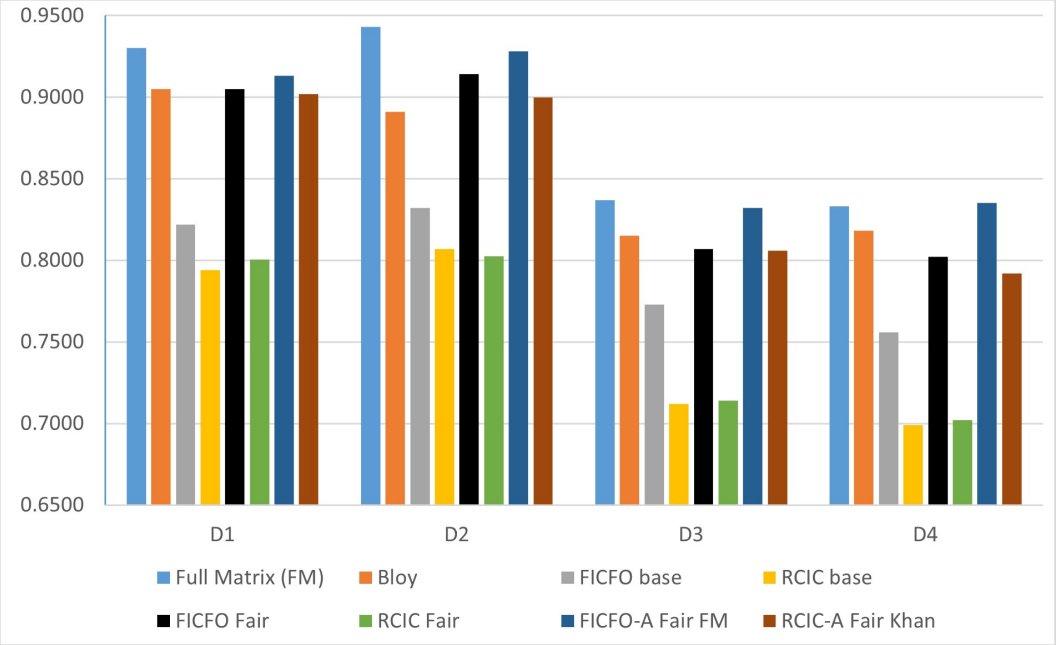

Figure 5.3: Comparison plot R

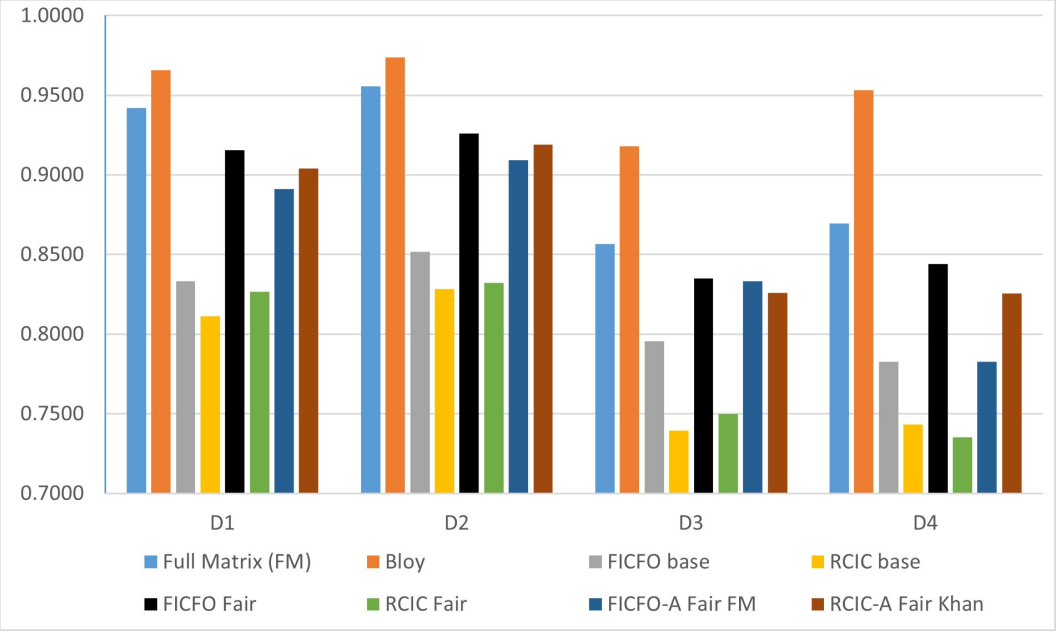

Figure 5.4: Comparison plot ARI

Chapter 6

Conclusions

In this thesis we experimented with low complexity image clustering algorithms to group images based on their camera fingerprints. The proposed improvements in the clustering and attraction phases are designed to work on top of FICFO [1] and RCIC [2] algorithms.
FICFO computes a ranking index RI as a metric of the quality of each estimated fingerprint and it stores all the fingerprints sorted in descending order of RI. FICFO produces a partition that is usually an overclustering of little pure (or quite pure) groups, following a fast and efficient procedure.
RCIC instead, saves the fingerprints in a random order and it has to regroup without any knowledge of the goodness of the attractors, producing stochastic outputs usually in a slightly less efficient way.
The proposed algorithms are suitable for large datasets since computational complexity per image decreases as the size of the image dataset increases. At the same time, are also robust when the size of clusters is small compared to the number of cameras, which is a typical problem in image clustering.

The results on different clustering techniques highlight FICFO Fair as the best of our solutions. It combines a very strong value of ARI while maintaining a very low complexity approach for all the datasets.
In a real scenario, we cannot expect an optimal partition computed by a greedy algorithm as FICFO Fair but just an approximation of the ground truth.

If we relax the problem and we do not account for the computational complexity, Full Matrix offers an extremely accurate solution for easy datasets; although, on hard datasets, the scarcity of information about the cameras has a strong impact on the Full matrix. It should provide the theoretical optimal solution, but it is very dependent on the correlation values it pre-computes. So, the less statistics it can extract from the fingerprints, the less of a precise centroid it can assign for the

groups.

After the clustering step, we are free to consider the obtained partition already as the final image clustering we were searching for, or we can apply an optional attraction phase, in order to obtain less groups and better clustering effectiveness.

RCIC appear to be a good fast alternative respect to the other algorithms, nevertheless is has always worse initial performances respect to FICFO and it is not very suitable for an attraction phase. Indeed, the number of computations increases a lot and overall, we have a significant degradation of the performances. We recommend working with RCIC Fair, as an improvement from RCIC Khan algorithm, but not to rely on the attraction phase performed by the RCIC-A Fair versions.
For the same dataset and imposed parameters, RCIC-A performs always worse respect to the simpler FICFO Fair, plus we must take into account the stochastic behavior of RCIC that make us always question about the real accuracy of our results. If we need to execute each RCIC code many times to obtain a stable average of the clusters, we must multiply the computational complexity times the number of runs we tested, receiving a big disadvantage respect to the deterministic algorithm.

The purification step, in both its forms, static or dynamic, introduces a non-negligible complexity to the attraction phase. The inputs are usually quite precise as they are computed by FICFO and RCIC, especially in their Fair version. In conclusion, the pure algorithms perform bad both as an attraction phase and as a pre-attraction step. They have to check the full set of fingerprints and they re-assign too many of them, usually to wrong clusters or they split the partition to a huge amount of groups, very hard to work with.

6.1 Future work, ways of improving

A good strategy for decreasing the complexity may be working with reduced fingerprints. For clustering digital images, we can suppose only a subset of pixels is relevant in our analysis. In our experiments, we always rely on the full fingerprint reference, hence we ensure more precise outputs, caused by the curse of dimensionality (Figure 2.9). This strategy has been already discussed in the literature and it is presented in the section: Sparse representation 2.2.3

Each of the proposed algorithms can be also extended to the case of large-scale databases in a divide-and-conquer strategy. We can load in memory only parts of our dataset and proceed by grouping sub-clusters based on the correlation matrix.

Each sub-cluster can be represented by its centroid, computed by averaging all the fingerprints in that sub-cluster and successively merged with the others. These techniques would allow us also to parallel computing, to reduce the computational time. For more information and for a fast overview we remind to the section: Divide and conquer split 2.4.3

We can come up with a better Weighted clustering technique that treats the ordering factor in a wiser way or it make use of another metric as weight. It would be great to have a strong knowledge extracted by the images. The ideal would be to distinguish few very good images (max weight parameter ~ 1), more or less one for each camera, and work by regrouping around them. The centroids should be able to move around the original position but just for a little amount, in order not to lose the optimal component they started from.
Such an algorithm would always require some flat image uniformly bright and not saturated, to fully achieve its goals. If we are just in presence of average-quality images, this trust makes us cluster too bound to our strong assumption.

A way of improving our code can be the introduction of an efficient low-complexity check phase that, starting from the considerations we did about the static and the dynamic purification step, is able to precisely locate the errors (without checking every fingerprint) and it can move them to the right cluster.
This operation would surely improve the precision at the output and, in case an attraction phase is used after, it can also improve the recall of the new groups, because the quite absence of errors would make the centroids more precise, allowing an easy join of groups under the threshold value.

Appendix A

Matlab code

A.1 Normalized cross correlation matrix

```
%% full cross corr matrix computations:
if exist('corr') % simply load the cross correlation
else %load the references of each fingerprint and create the corr
    matrix
     Ref=zeros(G,dim_sqr);
     for jj=1:G
          pathnoise=sprintf('%s/',srcFiles(jj).folder);
          filename= strcat(pathnoise,srcFiles(jj).name);
          Imm=load(filename);
          Ref(jj,:)=zscore(Imm.full);
          %Read full camera fingerprint and standardize to zero mean
     end

     corr=zeros(G);
     for ii =1:G
          for jj =1:ii-1 %corr(ii,ii)= 0; %don't consider ii-ii as a
    good pair
          corr(ii,jj)=(Ref(ii,:)*Ref(jj,:)')/dim_sqr;
          %normalized cross_correlation= 1-pdist([v1;v2],'cosine')
          corr(jj,ii)=corr(ii,jj); % the matrix is symmetric
          end
     end
end
ncc=n*(n-1)/2; % NCC counting to compute the computational complexity
```

A.2 Full Matrix without bin_vect

```
n=G; % n= number of fingerprints ,G= initial number of groups
len_Gr=ones(G,1);
% research of the max in the matrix and its indices:
C=max(max(corr));
[ind,~]=find(corr==C); ii=ind(1); jj=ind(2);

while(C>Th) % while there still exist a norm(C) value greater than Th
    corr(:,ii)=(sqrt(len_Gr(ii))*corr(:,ii) + sqrt(len_Gr(jj))*corr
    (:,jj)) /sqrt(len_Gr(ii)+len_Gr(jj));
    Group(ii)= [Group(ii);Group(jj)];
    len_Gr(ii)=len_Gr(ii)+len_Gr(jj);
    corr(ii,:)=corr(:,ii); corr(ii,ii)=-2;
    %value out of bounds, never considered again
    corr(jj,:)= -2*ones(n,1); corr(:,jj)=corr(jj,:);
    clear Group(jj);
    G=G-1;
    len_matrix=(len_Gr*len_Gr').^gamma;
    C=max(max(corr./len_matrix)); % normalization of corr
    [ind,~]=find(corr./len_matrix==C); ii=ind(1); jj=ind(2);
end

list= who ('Group*');
for ii=1:length(list) %shift groups indices:
FGroup(ii)=char(list(ii))
end
```

A.3 Full Matrix with bin_vect

```
n=G; %n= number of fingerprints ,G= initial number of groups
len_Gr=ones(G,1);
%research of the max in the matrix and its indices:
C=max(max(corr));
[ind,~]=find(corr==C); ii=ind(1); jj=ind(2);

while(C>Th) % while there still exist a norm(C) value greater than Th
    %correction coefficient between the two fingerprints:
    corr_coeff=C*sqrt(w_corr(ii))*sqrt(w_corr(jj));
    corr(:,ii)=(sqrt(w_corr(ii))*corr(:,ii) + sqrt(w_corr(jj))*corr
    (:,jj)) /sqrt(w_corr(ii)+w_corr(jj)+2*corr_coeff);
    w_corr(ii)=w_corr(ii)+w_corr(jj)+2*corr_coeff;
    Group(ii)= [Group(ii);Group(jj)];
    len_Gr(ii)=len_Gr(ii)+len_Gr(jj);

    corr(ii,:)=corr(:,ii); corr(ii,ii)=-2;
```

```
    %value out of bounds, never considered again
    corr(jj,:)= -2*ones(n,1); corr(:,jj)=corr(jj,:);
    clear Group(jj);
    G=G-1;
    len_matrix=(len_Gr*len_Gr').^gamma;
    C=max(max(corr./len_matrix)); % normalization of corr
    [ind,~]=find(corr./len_matrix==C); ii=ind(1); jj=ind(2);
end

list= who ('Group*');
for ii=1:length(list) %shift groups indices:
FGroup(ii)=char(list(ii))
end
```

A.4 First reference clustering

```
...
C=(Ref*Ixj')/length(Ixj); %Computing NCC
if(C>=Th)
    v1=zscore(Imi.full) %v1 = reference of the first Imm
    for jj=1:len_file
        Group(G)=[Group(G);src_file(jj)];
    end
    Ref(G)=v1; %Merging Fingerprints for the Attraction stage
```

A.5 Fair clustering

```
...
C=(Ref*Ixj')/length(Ixj); %Computing NCC
if(C>=Th)
    v2=zscore(Imi.full) %v2 = reference of the first Imm
    for jj=1:len_file
        Group(G)=[Group(G);src_file(jj)];
        %v2 = reference, updated with the new fingerprint Imm:
        v2=zscore( ((Ref*Av_fact)+Imm)/(Av_fact+1) );
        Av_fact=Av_fact+1;
    end
    Ref(G)=v2; %Merging Fingerprints for the Attraction stage
```

A.6 Weighted clustering

```
...
C=(Ref*Ixj')/length(Ixj); %Computing NCC
if(C>=Th)
    v2=zscore(Imi.full) %v2 = reference of the first Imm
    for jj=1:len_file
        Group(G)=[Group(G);src_file(jj)];
        curr_w=1-srcFiles(j).Ord; %current weight
        %v2 = reference, updated with the new fingerprint Imm:
        v2=zscore( ((Ref*Av)+Imm*curr_w)/(Av+curr_w) );
        Av=Av+curr_w;
    end
    Ref(G)=v2; %Merging Fingerprints for the Attraction stage
```

A.7 Hierarchical Agglomerative clustering

```
threshold= 1-Th; %it works with 1- the distance
corr_vect=squareform(corr); %vectorialized form of corr
Z = linkage(1-corr_vect,'average'); %average cosine distance.
T= cluster(Z,'cutoff',threshold,'Criterion','distance');
figure(1)
plot= dendrogram(Z,G,'ColorThreshold',threshold);%G= # of leaves
set(plot,'LineWidth', 2), hold on % increases the thickness
yline(threshold,'-.',sprintf('Threshold= %.2d',threshold)), hold off;
%yline shows the threshold
title('Cluster Assignments of FICFO Fair Clustering data', ...
Th=%.2d, #G= %d',threshold,max(T)),
xlabel('data'),ylabel('height');

%% merge Ggroup with the label =T(ii) in the same Fgroup
for ii=1:max(T)
    FGroup(ii)=[];
end
for ii=1:G
FGroup(T(ii))=[FGroup(T(ii));Group(ii)];
end
G=max(T); %update # of groups
```

A.8 Static pure clustering

```
load (srcFiles);
len_file= length(srcFiles);

for ii=1:len_file % ii= old group index
    v1=srcFiles(ii);
    corr_old=zeros(1,dim_sqr);
    for jj=1:G  %G=num of groups, jj= new group index
        v2= Refav(jj);
        corr_new= (v1*v2')/dim_sqr;

        if( corr_new > corr_old)
            corr_old=corr_new;
            index=jj; %better group found
            len_index=len_j; %store the length of Group_jj
        end
    end
    %add reference(ii) in the group:
    Group(index)= [Group(index);srcFiles(ii)];
end
```

A.9 Dynamic pure clustering

```
for ii=1:G %G=num of groups, ii= old group index
    kk=0; % fingerprint index
    while (kk<len_i &len_i>0)
        kk=kk+1;
        v1=load(Imm.full(kk))
        v2= Refav(ii);
        %compute the reference of the group(ii)- v1:
        v2=zscore( ((v2*len_i)-v1)/(len_i-1) );
        corr_old= (v1*v2')/dim_sqr;
        corr_old(isnan(corr_old))=0; %if it comes from a  div0 (Nan)

        for jj=1:G  %G=num of groups, jj= new group index
            v3= Refav(jj);
            corr_new= (v1*v3')/dim_sqr;

            if( corr_new > corr_old)
                corr_old=corr_new;
                index=jj; %better group found
                len_index=len_j; %store the length of Group(jj)
            end
            run shift_fingerprint % v1 is reassigned to Group(index)
    end
```

```
end

%% shift_fingerprint:
if( index ~= ii) %inside the while loop
    fprintf('Group(%d) Ref_%d->Group(%d)\n',ii,kk,index); %debug
    if( len_i>1) %remove 1 fingerprint in a big group
        %add kk in Group(index):
        Group(index)= [Group(index);Group(ii,kk)];
        %cut kk from Group(ii):
        Group(ii)=[Group(ii,1:kk-1);Group(ii,kk+1.len_i)];
        Refav(ii)= v2; %overwrite Refav_ii
        kk=kk-1; %next fingerprint has the same index, removed one
        len_i=len_i-1;
    else %attract 1 unclustered fingerprint in another group
        Group(index)= [Group(index);Group(ii)];%add kk in Group_jj
        Group(ii)= []; %empty Group(ii)
        len_i=0;
    end
    %recompute  Refav(jj):
    Refav(jj)= zscore((v1+(v3*len_index))/(len_index+1));
end
```

A.10 RI computation

```
TT= nchoosek2(sum(sum(Mtr)),2); %binomial coefficient (N,2)!
A= sum(sum(Mtr(:,:).^2));
B= sum(sum(Mtr(:,:),1).^2);
C= sum(sum(Mtr(:,:),2).^2);
RI= (TT + A - B/2 - C/2)/TT;
```

Bibliography

[1] S. Khan. «Efficient Image Clustering based on Camera Fingerprints». PHD thesis. Politecnico di Torino, 2020, pp. 21–107 (cit. on pp. ii, iv, 3, 17, 21, 28, 32, 92).

[2] S. Khan and T. Bianchi. «Fast image clustering based on compressed camera fingerprints». In: *Signal Processing: Image Communication* 91 (2021), p. 116070. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2020.116070. URL: https://www.sciencedirect.com/science/article/pii/S0923596520302010 (cit. on pp. ii, 3, 15, 25, 31, 92).

[3] G. J. Bloy. «Blind Camera Fingerprinting and Image Clustering». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.3 (2008), pp. 532–534. DOI: 10.1109/TPAMI.2007.1183 (cit. on pp. ii, 2, 3, 16, 26).

[4] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti. «Fast image clustering of unknown source images». In: *2010 IEEE International Workshop on Information Forensics and Security*. 2010, pp. 1–5. DOI: 10.1109/WIFS.2010.5711454 (cit. on pp. ii, iii, 3, 4, 22, 23, 35, 36).

[5] Q. Phan, G. Boato, and F. G. B. De Natale. «Image Clustering by Source Camera via Sparse Representation». In: *Proceedings of the 2nd International Workshop on Multimedia Forensics and Security* (2017), pp. 1–20 (cit. on pp. iv, 12).

[6] G. Chierchia, S. Parrilli, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. «PRNU-based detection of small size image forgeries». In: (July 2011). DOI: 10.1109/ICDSP.2011.6004957 (cit. on p. 2).

[7] R. Bakhshi S. Georgievska and B. van Werkhoven. «Clustering image noise patterns by embedding and visualization for common source camera detection». In: *Digital Investigation* 23 (2017), pp. 22–30. ISSN: 1742-2876. DOI: https://doi.org/10.1016/j.diin.2017.08.005. URL: https://www.sciencedirect.com/science/article/pii/S1742287617301706 (cit. on p. 2).

[8] C. Li. «Unsupervised classification of digital images using enhanced sensor pattern noise». In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010, pp. 3429–3432. DOI: 10.1109/ISCAS.2010.5537850 (cit. on p. 2).

[9] L. Bondi, P. Bestagini, F. Pérez-González, and S. Tubaro. «Improving PRNU Compression Through Preprocessing, Quantization, and Coding». In: *IEEE Transactions on Information Forensics and Security* 14.3 (2019), pp. 608–620. DOI: 10.1109/TIFS.2018.2859587 (cit. on p. 5).

[10] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva. «Blind PRNU-Based Image Clustering for Source Identification». In: *IEEE Transactions on Information Forensics and Security* 12.9 (2017), pp. 2197–2211. DOI: 10.1109/TIFS.2017.2701335 (cit. on pp. 5, 18).

[11] X. Lin and C. Li. «Large-Scale Image Clustering Based on Camera Fingerprints». In: *IEEE Transactions on Information Forensics and Security* 12.4 (2017), pp. 793–808. DOI: 10.1109/TIFS.2016.2636086 (cit. on pp. 8, 13).

[12] Q. Phan, G. Boato, and F. G. B. De Natale. «Accurate and Scalable Image Clustering Based on Sparse Representation of Camera Fingerprint». In: *IEEE Transactions on Information Forensics and Security* 14.7 (2019), pp. 1902–1916. DOI: 10.1109/TIFS.2018.2886929 (cit. on pp. 9, 12, 20).

[13] C. Hermeson, R Zampolo, D. Carmo, A. Castro, and E. Santos. «On the practical aspects of applying the PRNU approach to device identification tasks». In: Sept. 2012, pp. 1–7 (cit. on p. 11).

[14] Fernando Martin-Rodriguez. «Testing Robustness of Camera Fingerprint (PRNU) Detectors». In: *arXiv e-prints* (Feb. 2021), arXiv:2102.09444. eprint: 2102.09444 (cit. on p. 11).

[15] M.Goljan. «Digital Watermarking». In: ed. by Hyoung-Joong Kim, Katzenbeisser, and Ho. Springer Berlin Heidelberg, 2009, pp. 454–468. ISBN: 978-3-642-04438-0 (cit. on p. 11).

[16] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli. «Compressed Fingerprint Matching and Camera Identification via Random Projections». In: *IEEE Transactions on Information Forensics and Security* 10.7 (2015), pp. 1472–1485. DOI: 10.1109/TIFS.2015.2415461 (cit. on p. 13).

[17] M. Chen, J. Fridrich, M. Goljan, and J. Lukas. «Determining Image Origin and Integrity Using Sensor Noise». In: *IEEE Transactions on Information Forensics and Security* 3.1 (2008), pp. 74–90. DOI: 10.1109/TIFS.2007.916285 (cit. on pp. 14, 16, 17, 31).

[18] Daniel G. Costa and Luiz Affonso Guedes. «A Discrete Wavelet Transform (DWT)-Based Energy-Efficient Selective Retransmission Mechanism for Wireless Image Sensor Networks». In: *Journal of Sensor and Actuator Networks* 1.1 (2012), pp. 3–35. ISSN: 2224-2708. DOI: 10.3390/jsan1010003. URL: https://www.mdpi.com/2224-2708/1/1/3 (cit. on p. 15).

[19] M. Kivanc Mihcak, I. Kozintsev, and K. Ramchandran. «Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising». In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*. Vol. 6. 1999, pp. 3253–3256. DOI: 10.1109/ICASSP.1999.757535 (cit. on p. 15).

[20] Olli Virmajoki. «Pairwise Nearest Neighbor Method Revisited». In: (Dec. 2004), pp. 1–10 (cit. on pp. 17, 26).

[21] R. P. Adams. «Hierarchical clustering». In: *[online] Available:* (). URL: https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/hierarchical-clustering.pdf (cit. on pp. 19, 45).

[22] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. «An efficient k-means clustering algorithm: analysis and implementation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 881–892. DOI: 10.1109/TPAMI.2002.1017616 (cit. on p. 21).

[23] I.S. Dhillon, Y. Guan, and B.Kulis. «Weighted graph cuts without eigenvectors a multilevel approach». In: *IEEE transactions on pattern analysis and machine intelligence* (2007), pp. 1944–1957 (cit. on p. 24).

[24] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva. «Correlation clustering for PRNU-based blind image source identification». In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. 2016, pp. 1–6. DOI: 10.1109/WIFS.2016.7823910 (cit. on p. 41).

[25] T. Gloe and R. Böhme. «The 'Dresden Image Database' for Benchmarking Digital Image Forensics». In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. Association for Computing Machinery, 2010, pp. 1584–1590. ISBN: 9781605586397. URL: https://doi.org/10.1145/1774088.1774427 (cit. on pp. 50, 67).