


Doctoral Dissertation
Doctoral Program in Electronics Engineering (32.th cycle)

Efficient Image Clustering based on Camera Fingerprints

Sahib Khan
* * * * * *

Supervisor
Prof. Tiziano Bianchi

Doctoral Examination Committee:
Prof. Alessandro Piva, Referee, Università degli Studi di Firenze
Prof. Simone Milani, Referee, Università degli Studi di Padova
Prof. Paolo Bestagini, Politecnico di Milano
Prof. Lorenzo Galleani, Politecnico di Torino
Prof. Enrico Magli, Politecnico di Torino

Politecnico di Torino
2020

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Sahib Khan

Turin, 2020

Summary

This work presents a set of techniques concerning image forensics, more specifically, the core of the thesis deals with image clustering based on camera fingerprints. Image clustering using camera fingerprints is a blind problem, and clustering is performed in the absence of any prior information. The clustering process faces some serious issues of high computational cost, input-output (I/O) cost and $NC \gg SC$ problem, i.e., when the number of cameras NC is much larger than the average number of images per camera SC. Reducing the computational cost and finding a solution to the $NC \gg SC$ problem are the main objective of this thesis. Several algorithms have been proposed and presented, including reduced complexity image clustering (RCIC), fast image clustering algorithm based on fingerprints ordering (FICFO), canopy based image clustering (CIC) and compressed fingerprints based image clustering (CFIC) algorithms. The RCIC and FICFO are also implemented with an optional stage of attraction. The attraction process helps to improve the quality of the clusters. The RCIC and FICFO with attraction are represented as RCIC-A and FICFO-A, respectively.

For every algorithm, an estimate of the camera fingerprint for each available image is computed using standard techniques, then the algorithm tries to cluster those fingerprints according to the source camera. RCIC randomly selects a camera fingerprints from a set of un-clustered fingerprints as a reference and uses it as an attractor to construct a cluster. The remaining fingerprints are processed in the same manner by randomly choosing the next reference fingerprint out of the set of currently un-clustered fingerprints, and building another cluster. The RCIC is applied until all fingerprints are assigned to a cluster. The clusters are refined using the attraction process in the RCIC-A algorithm. The RCIC and RCIC-A clusters images with significantly lower computational cost in comparison with existing clustering algorithms, while maintaining similar or even better performance.

Moreover, these algorithms are robust to the $NC \gg SC$ problem. The clustering process is further simplified and made faster in FICFO by sorting the camera fingerprints using the inherent information of images. A factor of goodness called ranking index $\Re I$ is computed for each fingerprint, using the gray level, saturation, and textures level of the respective image. The higher the $\Re I$, the lower is the estimation error on the fingerprint and vice versa. Therefore, all the fingerprints

are arranged in the descending order of the $\Re I$ and the best fingerprint i.e., the fingerprint on the top, among the un-clustered fingerprints, is selected as reference fingerprint to attract other fingerprints of the same camera and construct a cluster. The results confirm that the FICFO and FICFO-A result in a cluster of high quality and at lower computational complexity. The sorting of fingerprints helps to reduce the computational cost with respect to RCIC and RCIC-A. The method efficiently handles the problem of $NC \gg SC$. The results show that these algorithms are more suitable for large scale clustering.

The CIC algorithm is another approach to cluster images using the camera fingerprints. The CIC algorithm uses the sorted camera fingerprints and operates in two stages. In the first stage, a relaxed threshold is set to construct fewer raw clusters of large sizes. The raw clusters are further clustered using a strict threshold. This results in a large number of pure clusters; most of them are singleton clusters. The clusters are refined using the attraction stage, which helps to reduce the number of clusters and improve the quality of clusters. The results show that the CIC algorithm does not suffer from the $NC \gg SC$ problem and has a significantly lower computational cost. However, even though the computational complexity of the CIC algorithm is lower than FICFO and RCIC, sometimes the performance of the CIC algorithm is degraded with respect to RCIC-A and FICFO-A. Furthermore, the lower computational complexity of the CIC algorithm for large image datasets makes it suitable for large scale clustering.

To reduce the computational cost further, reduced and full camera fingerprints are employed in the CFIC algorithm. The initial clustering is done using reduced fingerprints to construct clusters. The clustering performed on reduced fingerprints has a lower computational cost than that of using full fingerprints. The full fingerprints of each initially created cluster are merged by taking the average of them and standardizing the result to zero mean and unit norm. The merged fingerprints are used to refine the cluster and construct fine clusters. The CFIC algorithm, before clustering, computes $\Re I$ for each fingerprint, and arranges all fingerprints i.e., full and reduced fingerprints, in the descending order of $\Re I$. The CFIC method results in high quality clusters, comparable to the state-of-the-art techniques, at a significantly lower computational cost. The results show that the CFIC algorithm is suitable for large scale clustering and does not suffer from the $NC \gg SC$ problem.

Acknowledgements

First and foremost, I would like to thank Allah Almighty for all the blessings bequeath upon me. Without Allah's will, I would not have to get along this far. It is Allah's blessing that I can complete my research project. It was He who provided me enough courage and strength to take on this challenge, accomplish it, and make every hurdle easy for me.

I attribute myself lucky to have such a wise and kind supervisor and mentor, Prof. Tiziano Bianchi, who was available to me in every hour of consultation and need. I am thankful to him for the endless support throughout my research and Ph.D. studies.

Besides, I would also like to thank all the members and researchers of image processing and learning (IPL), especially Diego Valsesia, Giolio Coluccia, and Matteo Testa. The technical support, they extended to me has always been highly rewarding, and I learned a lot from them.

Finally, my deep and sincere gratitude to my family for their continuous and unparalleled love, help, and support. I am forever indebted to my parents for their prayers, love, opportunities, and experiences that have made me who I am.

Finally, I am thankful to the government of Pakistan and HEC Pakistan for the financial support under HRDI-UESTPs/UETs program. This is an excellent initiative by the government of Pakistan to improve the academic strength of the universities of the country and create quality researchers. This initiative is indeed a step to the educated and prosperous Pakistan.

Dedicate to prayers, love and smile; the love of parents, wife and siblings and the smile of a son

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In its initial era, photography was a chemical process and images were acquired on a photographic film contrived of silver halide emulsion, a light-sensitive material, coated on a flexible base. With the advancement of technology, analog photography has been replaced with digital photography [52, 76]. The coated films and chemicals have been substituted with arrays of photo-sensors and computer software. Nowadays, digital imaging has become an integral part of our life, together with other technologies [75]. The essential life events can be captured, and memories can be stored in digital pictorial documents. The photos can be posted on the internet and shared with family and friends on social media [94]. It becomes possible to use images in several computer applications e.g., to create presentations, documentation, pamphlet, journals, magazines, and much more. The images captured with cameras can play a role of evidence on crimes in the court of law and can help to solve cases such as bank fraud, child smut, street crime, and violence, etc. [7, 5].

The universally accessible nature of digital cameras, in the modern world, makes all these applications of digital images very easy. But, along with these advantages, digital imaging has brought many new challenges, including modification of digital images, origin identification of digital photos, grouping images based on a common source, and so on. Consequently, the forensic analysis of images becomes imperative, especially in crime inquiries [66, 100]. The different forensic tasks can be classified in the categories of source classification, device identification, device linking, recovery of processing history, integrity verification and anomaly inspection and analysis [4].

For the investigation of criminal cases, in order to reach a possible solution with the help of images, it is often essential to know the source of the images. For example, analyzing the source of images can help in linking an anonymous account to a known account by comparing the respective image galleries. Reliable and inexpensive camera identification techniques are beneficial to recognize the device which has been used to catch a particular image or video [15, 106]. The most common and straightforward approach is to obtain such information by looking

for clues inside the digital image file. The information related to camera type and the condition under which the image has been taken is present in the header file and the metadata, e.g., the Exif header [96, 65]. However, this information is not present in all images of all camera models, and even if the information is available, it can be easily altered [32]. The embedding of a watermark, visible or invisible [63, 113], carrying the camera-related information is another solution for camera identification and device linking. The use of watermarking is highly appreciated in different forensics tasks, but it depends on the fact that the camera vendor should embed it in the firmware and it can be removed with different attacks [35, 108]. In general, it can not be adopted as a source of camera identification and device linking.

Therefore, intrinsic, non-removable, stable, and unique features of images are needed to get source information of the camera. For this purpose, it is reported in the literature that pixels of the camera sensors contribute some noise to the image taken by the camera [1, 123]. These include shot noise or random noise, readout noise, and pattern noise. The shot noise is caused by the random fluctuation in the density of photons, striking the pixels of the sensor [11, 55]. The shot noise varies from image to image depending on environmental conditions; therefore, it can not be used as a unique camera fingerprint for source identification and device linking. The readout noise is introduced by reading the data from the sensor. However, this varies with light intensity i.e., the brightness of images and can not be treated as a unique fingerprint. Pattern noise is deterministic and remains stable [16, 105]. Every image that is captured with a camera sensor has a specific pattern noise. By averaging multiple images of the same camera, random noise components can be reduced, improving the relative strength of pattern noise and making it more readable.

The pattern noise is a combination of fixed pattern noise (FPN) and photo response non uniformity (PRNU) noise, [86, 14]. The FPN is additive by nature and is caused by the dark current [10, 24]. The problem with FPN is that it can be eliminated from the image by simply subtracting the dark frame of the camera from the image. The PRNU is added in an image due to the pixel non uniformity. It is the preeminent component of the pattern noise. The non uniformity occurs during manufacturing [14, 61]. This makes the PRNU a unique and stable noise component that remains unaffected by temperature or humidity [86, 51]. Due to these characteristics, PRNU can be adopted as a unique camera fingerprint and used for camera identification, device linking, and other forensics tasks [78, 25]. The PRNU can be used to identify the source of an image, finding the brand and model of a camera used for taking an image, or group together the images coming from a common source camera [25, 89]. The camera fingerprint is usually estimated from an image by subtracting the de-noised image from the original image [86, 14].

Figure 1.1: Camera fingerprint.

1.1 Problem Statement

The main focus of this work is to cluster digital images using camera fingerprints. The grouping becomes very easy if either source cameras or appropriate candidate images for the cameras are available [104]. The fingerprints are estimated from images taken by available cameras, or from the candidate images. Which are used to cluster the set of unknown images. However, in real scenarios, the source cameras and candidate images of the source cameras are not available. Usually, only a bunch of images are available to forensic analysts without any prior knowledge of source camera or candidate images. But, it is still essential to cluster images, so that each cluster is composed of images from the same camera [101, 102].

The clustering of images using the camera fingerprints, without any other information available, by itself is a significant research problem for forensics experts. However, the unique and stable PRNU makes it possible to group images from the same sources. Several image clustering algorithms using camera fingerprints have been proposed in the literature [8, 21, 77]. Usually, a correlation mechanism is employed, such as the peak correlation energy (PCE) or normalized cross correlation (NCC). Both the similarity measure i.e., PCE and NCC, can be used for the clustering of images based on camera fingerprints. The PCE is a much stable test statistic than NCC, but the PCE is computationally expensive and the NCC is a key factor for computing PCE [35, 110]. Due to the less computational cost, we prefer to use NCC instead of PCE. Therefore, in our proposed work, the clustering

is done by computing the NCC among the PRNU patterns, onward called camera fingerprints, and using the NCC as a similarity measure. Pair of fingerprints with NCC above a predefined threshold value are grouped together and considered to come from the same device. However, calculating the NCC between camera fingerprints is computationally very expensive, especially when the number and size of images become very large.

Several existing algorithms [77, 83, 125, 12] compute the NCC between each possible pair of camera fingerprints. These algorithms develop a full cross-correlation matrix among n fingerprints, at the cost of $n(n-1)/2$ correlations. This large number of correlations make these algorithms unsuitable for clustering large image datasets. The large number and large size of camera fingerprints also increase the memory requirements. Along with the computational cost and memory requirements, the input-output (I/O) cost is another important problem related to the clustering of images using camera fingerprints.

In the most challenging setting, clustering is performed in a complete blind environment, and we do not have any information regarding the number of cameras and number images contributed by each camera in the dataset. However, for clustering images using camera fingerprints, efficiently, it is also vital to have a reasonable number of images from each camera, contributing to the dataset. The absence of a reasonable number of candidate images from each camera affects the reliability of estimated fingerprints. Usually, a fingerprint estimated from a single image has a large estimation error. While the presence of a suitable number of images reduces the estimation error and increases the reliability by merging the camera fingerprints. The merging is done by taking the average of the estimated fingerprints, which helps to suppress the disturbances like the shot noise, the readout noise, high frequency contents of the image, and other noise sources [27]. The problem is called $NC \gg SC$ problem, and it arises when the number of cameras NC is much larger than the average number of images taken per camera SC [81]. Many current algorithms [77, 83, 12, 116, 40] face this problem.

These severe limitations of high computational cost, I/O cost, large memory requirements, $NC \gg SC$, sensitivity to outliers, and the need for prior information, make the clustering problem very difficult. The solution to these problems is the prime purpose of this research work.

1.2 Contributions

With the increasing role and effect of digital images in human life, it is essential to authenticate the source and the contents of the images. The forged and modified contents can mislead us. Several techniques have been developed for the authenticity of images. These techniques are broadly classified into two categories, i.e., active techniques and passive techniques [30, 84]. The active authentication

techniques need prior information. The data hiding is used to embed a code in the image at the time of acquisition [126]. The passive authentication techniques do not need prior information and need the image contents itself [99]. The main focus of the thesis has been on passive techniques that exploit camera fingerprints to link a photo to the device that acquired it.

In this thesis, the focus of the research is to cluster digital images using their camera fingerprint without exploiting any other source of information. The image clustering faces some serious problems, as discussed in Section 1.1. The main issues that are addressed in this thesis are computational complexity, cluster quality, and $NC \gg SC$ problem. Many solutions are devised to solve these problems.

The first contribution towards the image clustering is reduced complexity. The algorithm clusters images based on their camera fingerprints. The camera fingerprint is estimated for each image of an unknown source. A set of un-clustered fingerprints is processed, and a fingerprint is randomly selected as a reference fingerprint. The reference fingerprint is used as an attractor to construct a cluster. NCC is computed between the reference fingerprint and each of the un-clustered fingerprints one by one. The fingerprints that satisfy threshold criteria are considered from the same camera as the reference fingerprint. The algorithm is explained in [70]. The algorithm results in high quality clusters at significantly reduced computational cost. Moreover, the algorithm is robust to the $NC \gg SC$ problem.

Our second contribution is the clustering of sorted fingerprints. The fingerprints are arranged in the descending order of a factor called ranking index $\Re I$. The ranking index is computed using the inherent gray level, saturation, and texture information of the images. Flat images that are neither too dark nor too saturated results in good quality camera fingerprints that can be used more reliable as attractors. Therefore, $\Re I$ is computed for camera fingerprints using their own images. The camera fingerprints are sorted in the descending order of the values of $\Re I$. The algorithm clusters images using these sorted camera fingerprints. The clustering algorithm selects the best fingerprint, from the set of sorted and un-clustered fingerprints, as reference fingerprint and uses it as an attractor to construct a cluster. The sorting of fingerprints helped to reduce the computational cost further. The clustering based on the sorted camera fingerprint is presented in [69]. The algorithm results in high quality clusters at much reduced computational cost. The algorithm also provides a solution to the $NC \gg SC$ problem.

The Canopy based image clustering is the third contribution of this thesis. The sorted camera fingerprints are first grouped in large clusters using relaxed threshold criteria with a higher probability of false alarm (PFA). This process builds fewer clusters of large size. The large clusters are called Canopies, which are further clustered using a hard threshold with a very low PFA. The clusters are refined to get final clusters. The algorithm constructs quality clusters at a lower computational cost.

The compressed fingerprints based image clustering is another contribution to

the thesis. The full fingerprints are estimated from the images under the test. The full fingerprints are compressed to get reduced camera fingerprints. This framework performs clustering on compressed camera fingerprints. The compressed fingerprints based clustering has a significantly lower computational cost and results in high-quality clusters. The full fingerprints of each constructed cluster are merged and used for fine clustering to refine the clusters. The compressed fingerprints based clustering has the lowest computational cost out of the proposed techniques. The computational cost is also quite lower than in state-of-the-art clustering algorithms. The $NC \gg SC$ problem is also solved by this framework.

We also investigated an active technique based on data hiding. Ant colony optimization (ACO) based data hiding has been devised to hide information in the complex region of cover images. The ACO-based data hiding in a complex area develops a pheromone matrix that identifies the complex region. The pixels that belong to the edges are then subjected to the LSB substitution technique. The LSB of the pixels of the complex region is substituted with secret message bits and hide a secret message in these pixels. The results show that the ACO based data hiding in the complex region result in quality stego images, and the hidden information is undetectable for the human visual system (HVS).

1.3 Organization of the thesis

The thesis is organized in the following chapters:

- Chapter 2 presents the camera sensor output model, camera fingerprints estimation, and matching procedure. It also presents a detailed literature review of image clustering algorithms based on camera fingerprints, discussing the advantages and the problems these algorithms face.

- Chapter 3 presents the proposed image clustering algorithms. The steps involved and the detail implementation of the algorithms are discussed in this chapter.

- Chapter 4 presents a detailed discussion on the experimental setup and the evaluation metrics. It then shows extensive experimental results to analyze the clustering algorithm for large and small scale clustering. The robustness of the algorithms against the $NC \gg SC$ problem is also examined. The comparison of the proposed algorithm among themselves and with state-of-the-art techniques is also presented in this chapter.

- Chapter 5 presents a data hiding technique in the image complex region based on ant colony optimization. The proposed technique is implemented on different cover images, and the algorithm is evaluated based on the quality of

stego images and hiding capacity achieved. The performance of the proposed algorithm is also compared with some of the previous data hiding techniques.

- Chapter 6 concludes the work. It also states the possible future work in this field.

Chapter 2

Background on Image Clustering based on Camera Fingerprints

Digital cameras usually use charge-coupled device (CCD) and complementary metal-oxide semiconductor (CMOS) as imaging sensors [82, 92, 91]. The sensor is composed of several photo detectors, called pixels in imaging terminology. These pixels capture the light incident on them and transform the photons into electrons in compliance with the well-known photoelectric effect. The number of generated electrons depends on the intensity of incident light and the dimension of the light sensitive area of the pixel. The dimension of the sensor's active area and the contribution of impurities in the semiconductor material vary from sensor to sensor due to the non-ideal manufacturing environment and tools. These factors, along with many other factors, introduce both systematic and random variations in the images. These fluctuations are independent of the scene and do not vary with the image capturing conditions. These play a vital part in the forensic examination of the digital images acquired with these sensors.

The sensor imperfections contribute several types of noise to the image captured with the camera. These include shot noise or random noise, readout noise, and pattern noise. The random noise is caused by the random fluctuation in the density of photons, striking the pixels of the sensor [11, 55]. The pattern noise is composed of two major components [86, 14]. The first component is the fixed pattern noise (FPN), which is additive by nature and is caused by the dark current [10, 24]. The problem with FPN is that it can be eliminated from the image by simply subtracting the dark frame of the camera from the image. The other component of pattern noise is known as photo response non-uniformity (PRNU) in literature [27, 74]. PRNU is added in an imaged due to the pixel non uniformity. It is the preeminent component of the pattern noise. The PRNU is unique, stable, and multiplicative in nature. Therefore, PRNU uniquely represents the camera acquisition device and is used as a camera fingerprint. The PRNU is used to accomplish several forensic tasks e.g., camera identification, device linking, and tampering detection, etc. [106, 27].

To exploit the use of PRNU for the mentioned task, it is essential to estimate the PRNU from the images. For estimating the PRNU, it is useful to introduce a sensor output model.

2.1 Sensor Output Model

The image acquisition is a complex process and varies with different camera models. But, the basic concept, elements, and functions are shared by most camera models. Each camera has a batch of sensors, and the light is directed onto the array of the image sensor, also called pixels. The incident light generates current due to the photoelectric effect. The current is enhanced by amplification and quantized. The pixels use a color filter that allows only a single color (red, blue, or green) light incident on a pixel. This array of filters is called the color filter array (CFA). The signal is interpolated or demosaicked to get a color image [27, 85]. To correctly display an image on a screen, the colors are adjusted using color correction and gamma correction. Some camera models also use denoising or sharpening techniques. The processed image is stored in an appropriate format e.g., JPEG.

Let us consider the quantized signal $I[i]$ acquired at pixel i, $i = 1, ..., w \times h$, before demosaicking. Where $w \times h$ is the dimension of the image. Let us denote as $Y[i]$ the intensity of the light incident on pixel i. For a clearer and readable representation of the camera model, the pixel indices are dropped. According to Fridrich [27], the output of the camera can be modeled as given in Eq. 2.1.

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

In Eq. 2.1, the factors g and γ represent the color gain and gamma correction, respectively. The term K is noise signal i.e., PRNU, of zero-mean. Ω represents a mix of other types of noises, i.e., shot noise, the dark current, and readout noise [54, 42]; Q represents quantization and compression distortion.

The scene light is the major factor in an image that is not dark and is represented by the terms in square brackets in Eq. 2.1. Applying the Taylor expansion [60] and keeping the first two terms, we remain with the terms expressed in Eq. 2.2 and Eq. 2.3.

$$I = (gY)^{\gamma}.(1 + \gamma K + \gamma\Omega/Y) + Q \tag{2.2}$$

$$I = I^{(0)} + I^{(0)}K + \Theta. \tag{2.3}$$

2.2 PRNU Fingerprint

In the absence of all other noise sources, when the light of intensity Y strikes on the imaging sensor, the sensor registers a $I^{(0)}+I^{(0)}K$ signal. The signal $I^{(0)}+I^{(0)}K$, is a combination of two components, scene $I^{(0)}$ and the noise pattern K. The term K represents the PRNU and is used as a camera fingerprint.

The noise components are much smaller than the image contents. To improve the signal to noise ratio (SNR) between the signal of interest (pattern noise) and the image contents I, the image contents are suppressed. PRNU noise residual W is obtained by subtracting the denoised signal $D(I)$ of a sensor from the original signal I [14, 54].

$$W = I - D(I) \tag{2.4}$$

$$W = IK + \Xi \tag{2.5}$$

The Ξ represents the sum of Θ and other noise introduced by the denoising filter. Ξ is present in the content $I - D(I)$ due to the inability of the denoising filter to separate content from noise. The $D(.)$ represents the Mihcak denoising filter [86]. The wavelet coefficients based denoising filter is modeled by treating the high wavelet coefficient of the noisy image as a concoction of stationary i.i.d., signal of zero mean and stationary white Gaussian noise $N(0, \sigma_0^2)$.

Let us consider a set of n images I_1, I_2,..., I_n. We estimate noise residual W_1, W_2,...,W_n and model the Ξ_1, Ξ_2,...,Ξ_n as white Gaussian noise (WGN) with variance σ^2. The Ξ term is technically not independent from the PRNU signal of IK. But, the energy of Ξ is smaller than IK; therefore, these terms are assumed to be independent of each other.

For the noise residual of each image I the Eq. 2.5, can be written as

$$\frac{W_k}{I_k} = K + \frac{\Xi_k}{I_k} \tag{2.6}$$

The maximum likelihood estimator is used to estimate the fingerprint $\hat{K}$ shown in Eq. 2.7.

$$\hat{K} = \frac{\sum_{k=1}^{n} I_k W_k}{\sum_{k=1}^{n} I_k^2} \tag{2.7}$$

The $\hat{K}$ for a camera sensor is estimated from the n images taken with the same camera. This can be an effective way of computing $\hat{K}$ when a suitable number of known candidate images are available. Usually, it is preferred to use several candidate images to estimate camera fingerprints. As the optimal estimator is a bit more complex, therefore, the fingerprint for a sensor is obtained by taking an average of camera fingerprints estimated from individual images of the same camera sensor. The averaging helps to suppress the disturbances like the shot noise, the

readout noise, high frequency contents of the image, and other noise sources [27] and also improve the PRNU pattern noise.

But, in the clustering problem, it is not possible to take an average of multiple fingerprints, because the source of images, the number of cameras and the images taken with a camera are not known. Therefore, the clustering is done using PRNU estimated from every single picture.

The fingerprint of a single image can be given as

$$K = \frac{IW}{I^2} \tag{2.8}$$

We suppose that the PRNU is the representative of the camera, which has captured it.

The term K is mainly responsible for the camera fingerprint. According to the Cramer-Rao Lower Bound (CRLB) [64], the bright and unsaturated images with small σ^2, i.e., smooth contents, are most suitable for fingerprints estimation. The fingerprints estimated from unsaturated, uniformly illuminated, and low textured images are more reliable to represent the source camera [14, 26]. Moreover, the PRNU signal varies among cameras of different models. It changes from camera to camera of the same model. That's why it is unique for each camera and is considered as a unique fingerprint for the camera.

2.2.1 Fingerprints Post-processing

The estimated PRNU fingerprint is processed to remove non-unique artifacts (NUA), the artifact shared by different cameras. The fingerprint estimate K contains several artifacts, systematically present in images. These artifacts occur due to color interpolation, JPEG compression, on-sensor signal transfer [29], and sensor design. These non-unique artifacts (NUA) are shared among the cameras of the same model or sensor design, while the PRNU is unique for each sensor. In the presence of these artifacts, PRNU of two different cameras may be slightly correlated and may result in false identification. Most of these artifacts are, fortunately, due to demosaicking algorithms that depend on the CFA and are periodic in nature. These can be eliminated by zero-meaning the rows and columns of the PRNU separately for each pixel type, as defined by the CFA. The fingerprint is further processed using a Wiener filter with noise variance σ^2 in the frequency domain, to suppress any remaining NUAs, such as non-periodic artifacts [13].

2.3 Device Identification and Fingerprint Matching

After the estimation of fingerprints, this section demonstrates how the fingerprints can be used to identify whether a specific image is acquired by a particular device or not. In such cases, the device and the unknown images are available. Let us consider an unknown image I of size d, and a fingerprint K_i is estimated from the image, following the previously presented procedure. The fingerprint $\hat{K}$ is computed for the device using the images captured with it. Then the NCC ρ between the fingerprint of the device and the fingerprint K_i estimated from the image I is computed as given in Eq. 2.9

$$\rho = \frac{1}{d} \sum_{x=1}^{d} \hat{K}[x] K_i[x] \tag{2.9}$$

If the NCC ρ, between the $\hat{K}$ and K_i, is higher than a predefined threshold value, the image is considered to be captured with the device under test.

However, when the device is not available, we have only images without any information about their sources. We are still interested to know whether two images are acquired with the same device or not, which is the usual case in blind clustering. To determine whether a group of images is taken with a specific common source camera or not, PRNU, i.e., fingerprints are used. For the decision, whether two images share a common source or not, the normalized cross-correlation (NCC) ρ between their respective, estimated fingerprints are computed. Let us consider two fingerprints K_1 and K_2 estimated from I_1 and I_2, respectively. The NCC ρ between a couple of fingerprints is calculated as given in Eq. 2.9.

If the NCC $\rho \geq Th$, the K_1 and K_2 are considered of the same sensor, and the corresponding images are considered to be captured with the same camera. The threshold value is calculated according to Eq. 2.10.

$$Th = \sqrt{2 \times \frac{1}{d}}\, erfc^{-1}(2 \times PFA) \tag{2.10}$$

Where $erfc^{-1}(.)$ is the inverse of the complementary error function, and PFA represents the desired probability of false alarm.

According to the Central Limit Theorem (CLT) [43], if we have two normalized fingerprints X and Y of size equal to d and belonging to two distinct cameras, the NCC ρ between them is approximately distributed as a normal distribution with zero mean and $1/d$ variance, i.e., $\rho(X, Y) \sim N(0,1/d)$ [89]. Hence, the probability of declaring two fingerprints from the same camera when they, in fact, comes from different cameras is bounded by PFA.

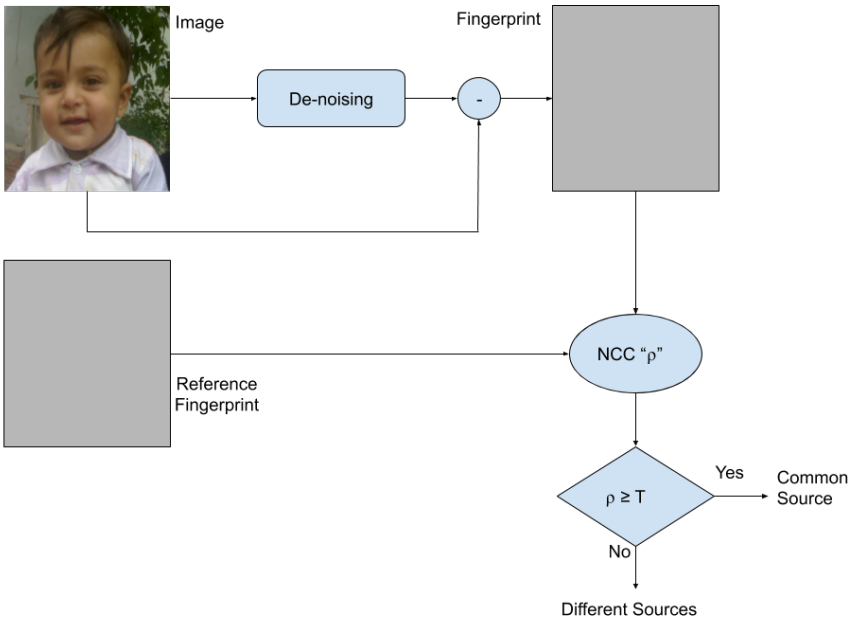


Figure 2.1: Fingerprint matching.

2.4 Literature review

The source identification and image clustering based on camera fingerprints is an area of great interest to forensic experts. The image clustering, which we focus on, deals with a set of images from unknown sources. The images are grouped without any prior information about the source camera, the number of source cameras and the number of images contributed by individual cameras. Therefore image clustering problem can be seen as a blind process. Many techniques related to image clustering have been reported in the literature. The camera fingerprinting and image clustering (BCFIC) [8] and large scale image clustering (LSIC) [81] are the two most prominent algorithms. The performance of our proposed algorithms is compared with these two algorithms. Therefore, the BCFIC and LSIC algorithms are presented in detail in this section. We also present a summary of some other selected image clustering techniques in the following sections. The details of the techniques can be found in the respective papers.

2.4.1 Blind Camera Fingerprinting and Image Clustering (BCFIC)

Bloy was the first who proposed a blind image clustering (BCFIC) algorithm [8]. The algorithm was based on the pairwise nearest neighbor (PNN) technique

reported in [21]. The PNN technique deals with each element as a single point cluster and calculates the pairwise distances between all single point clusters. The closest point clusters are merged to get a new cluster with its own centroid. The process ends when a stopping condition is met. Bloy in [8] used the modified version of the PNN algorithm to reduce complexity and accelerate the clustering process.

The BCFIC, before the clustering process, estimates the camera fingerprints from a set of images. The fingerprints enable the source camera identification of each image without any prior information of the source. The BCFIC does not rely on any training set and needs only the pre-calculated threshold, which plays a decisive role in the clustering process.

Bloy's algorithm constructs a cluster using the following process.

- Randomly selects a pair of fingerprints and computes the correlation between them. If the correlation value is higher than or equal to the threshold value, the process of searching for a pair of fingerprints stops and the algorithm moves to the next step. Otherwise, another pair of fingerprints is selected, and the process is repeated. The process continues until a pair with a correlation value greater than the threshold is obtained or a certain number of attempts are made. If no couple meets the criteria, the clustering process ends and the fingerprints are declared unclustered.

- As the pair is found, the pair of fingerprints are merged by averaging them. The averaged fingerprint is used as a reference fingerprint for constructing clusters, from remaining un-clustered fingerprints. The correlation between the reference fingerprint and the remaining fingerprint is calculated one by one. The fingerprint with correlation value greater than the threshold, are assigned to the same cluster as that of reference fingerprint. The process continues until the size of the cluster reaches 50, or all un-clustered fingerprints are processed.

- The fingerprints of the constructed cluster are merged by taking an average of them. The unclustered fingerprints are again processed by computing the correlation between the average reference fingerprint and each un-clustered fingerprint. If the threshold criterion is met, the fingerprint is assigned to the same cluster; otherwise, it is left as un-clustered. The process ends when all the un-clustered fingerprints are processed.

- Repeat the process until all fingerprints are assigned to a cluster or enough number pairs have been tried without success in the first step.

The algorithm results in different clusters and each cluster is composed of images acquired with the same camera. The images not assigned to a cluster are labeled as un-clustered image. However, the algorithm suffers from the $NC \gg SC$ problem.

For efficient clustering, it is essential to have a suitable number of images in the dataset for each camera, which is not the usual case in real life.

2.4.2 Large Scale Image Clustering Algorithm (LSIC)

Lin and Li in [81], presented a large scale image clustering (LSIC). LSIC is an attempt to address to clustering issue of large datasets and also $NC \gg SC$ problem. The algorithm split the large dataset into small sub-datasets, which can be efficiently loaded on RAM. The computational cost is reduced by using reduced fingerprints, obtained by using sparse random projection [80, 2]. The LSIC algorithm consists of the following steps.

- **Preparation:** The dark or saturated images are eliminated from the set of images. Then full camera fingerprint is estimated and standardized to zero mean unit norm, from each non-eliminated image. The dimension of the fingerprints is reduced using the sparse random projections [80]. The full and reduced fingerprints are stored and used for clustering the images.

- **Coarse Clustering:** The set of the reduced fingerprints is split into small batches so that it can be easily loaded on RAM. Initially, two batches are loaded on RAM and processed to construct a graph L. Each node L_i of the constructed graph L contains the list of the indices of the vertices adjacent to the i^{th} fingerprint. The remaining batches are loaded one by one, and the graph L is constructed. The Graclus [17] graph partitioning algorithm is used to create coarse clusters.

- **Fine Clustering:** After the coarse clustering, the correlation matrix is calculated and binarized for each coarse cluster using the full fingerprints. The cluster is split into sub-clusters based on the binary correlation matrix. Each sub-cluster is represented by its centroid, computed by averaging all the fingerprints in that sub-cluster. Using the centroid, the full correlation matrix is calculated and binarized using a threshold of τ. Larger clusters are constructed by merging the sub-clusters satisfying the threshold criteria.

- **Attraction:** Then centroids are computed for the merged clusters and are equalized to reduce the NUAs, which are further used as attractors to attract the unclustered fingerprints.

- **Post-processing:** The fingerprints of the cluster smaller than a threshold size are assigned to the unclustered set of fingerprints, and the remaining clusters are declared as final clusters. The algorithm ends when no more notable clusters are obtained. The unclustered fingerprints and the fingerprints of clusters with a size smaller than the threshold size are declared unclustered.

LSIC algorithm generates quality clusters specifically for large databases and also solves $NC \gg SC$ problem. Although the computational complexity of clustering has been reduced, but still significantly high and needs to be reduced further.

2.4.3 Other Image Clustering Algorithms

In [77], Li used enhanced fingerprints for clustering. The fingerprints are considered as random variables and repeatedly clustered using a Markov random field (MRF). A bunch of images is randomly picked, and fingerprints are extracted and enhanced. A pairwise similarity matrix is generated and using the matrix, a similarity criterion and membership committee are determined. The fingerprints are used to calculate the likelihood probability of belonging to a class. The fingerprints are assigned to a class on the bases of the highest likelihood probability in the membership committee. This procedure continues when the class label changes in consecutive iterations. If no change in the class label is observed after two consecutive iterations, the process stops. In the end, the fingerprints not included in the training set are assigned to the closest clusters constructed in the training process. This algorithm performs significantly well and is very efficient for clustering small datasets. The algorithm needs the likelihood probability of each class and each fingerprint in the class, which is computationally expensive. The time complexity is about $O(n^3)$ for clustering n fingerprints in the first iteration. The high computational cost makes this algorithm inadequate for clustering large datasets. Along with high computational cost, the algorithm also faces a $NC \gg SC$ problem. The $NC \gg SC$ problem arises when the cameras NC are much larger in number than the average number of images taken per camera SC [81].

In [83], camera fingerprint clustering was treated as a graph partitioning and considered it a weighted unidirectional graph. The fingerprints are regarded as the vertexes of the graph. The similarity between the fingerprints is used as the weight of the edge that links the fingerprints. The k-nearest graph is built by randomly selecting a vertex as an initial center, and the edge weights with the rest of vertices are computed. Then $(K+1)^{th}$ closest vertex to the center is chosen as the new center. The edge weights of the new center are calculated with the remaining vertices, excluding the already selected center. K is the sparsity controlling factor of the graph. The selection of center vertices and calculating edges' weights terminates when the number of vertices not chosen as the center becomes less than K. Afterward, the partition of the vertices of the k-nearest graph is done using a multi-class spectral clustering algorithm [125]. The algorithm in [83] has high I/O cost as well as high computational cost. Because during computing similarity of fingerprints with selected centers, the fingerprints are read multiple times from memory, while the algorithm in [125] has a computational cost of $O(n^{\frac{3}{2}}m + nm^2)$. The m represents the total number of partitions. If the total of fingerprints n is much larger than m, i.e., $n \gg m$ the algorithm performs better than Li's algorithm

in [77]. But, the algorithm needs information about the number of partitions m in advance, which is not possible in practical scenarios. However, to reach the best value of m, the clustering is repeated for various values of m, until a single element cluster is achieved. This process makes the algorithm computationally expensive and makes this unsuitable for clustering large image datasets.

A hierarchical image clustering is proposed in [12]. The fingerprints are enhanced using the technique given in [77]. Instead of using the complete dataset for clustering, a set of the randomly selected training set is used for clustering. Each member of the training is handled by an individual cluster, and a pairwise similarity matrix for the full training set is calculated. The clusters with maximum similarity are merged, and the matrix is updated. Following the updating, the overall measure of the aptness of the existing partition is obtained by calculating the silhouette coefficient [23] of each fingerprint and averaging the silhouette coefficients. The partition having the highest aptness is assumed to be the best partition. The algorithm in [12] is faster than the algorithm expressed in [77], with acceptable accuracy. However, the computational cost is very high to cluster large datasets. In [31], instead of original full fingerprints, compressed fingerprints are used. At the start, each fingerprint is considered as a cluster, and the similarity matrix is calculated. The clustering is performed iteratively. In each iteration, a couple of clusters having the highest correlation are selected and merged if the correlation between a couple of clusters is higher than a predefined threshold; otherwise, the clustering stops. The work presented in [23] further improved the clustering results by using Hu's moment vector mechanism for refinement. Smart-phone image clustering is presented in [116]. The algorithm proposed in [12] has been used for clustering with some modifications. The smart-phone image clustering algorithm calculates the silhouette coefficient for each cluster rather than for each fingerprint. The algorithm produces a cluster with good quality but is computationally expensive.

In [79], a fast source-oriented image clustering technique is presented. The algorithm works in a complete blind environment, deals camera fingerprints as random variables, and calculates the pairwise correlation between camera fingerprints. The MRF technique is used to assign a class label to each fingerprint. The class label is assigned using a cost function. The function is formulated with a different voting power of neighbors of the fingerprints, depending on their similarity.

Following the belief of sparse subspace clustering (SSC) [20] that a data point can be represented by the linear combination of other points in the same subspace, Phan et al., in [101] presented as new clustering technique based on the SSC. The algorithm obtains the sparse representation of each camera fingerprint, using l1-regularized least squares and estimated suitable parameters in a data-driven manner. The clustering is done using a divide and conquers approach, that enables the proposed algorithm to cluster large datasets efficiently.

Several classical clustering algorithms exist in the literature. But they are not suitable for use in the clustering of images based on camera fingerprints due to

large size of datasets and individual images, absence of prior knowledge about the sources and the number of sources of images. The K-means [53] and CLARANS [98], clustering algorithms need the number of clusters in advance, which is not possible in practical scenarios. Besides, the algorithm performs several iterations to process the whole database, which does not fit well to cluster large-scale camera fingerprint clustering. The DBSCAN [22], is another efficient classical clustering algorithm that processes the whole database at a time, and a large memory is required along with a substantial I/O cost [40]. The DBSCAN is also sensitive to its parameters, and the noise-like nature of camera fingerprints can generate clusters of different sizes. The hierarchical clustering techniques like [40] and [41], solve the memory requirement issue by reducing the input size, especially for large databases, but will face the $NC \gg SC$ problem. The BIRCH [127] and CHAMELEON [62], are devised for large databases. However, these techniques are considered unsuitable for camera fingerprints databases because these are very sensitive to outliers and have high I/O cost while building the K-nearest neighbor graph.

In [90], McCallum et al. presented an algorithm to cluster large datasets. The algorithm is composed of two main stages. In the first stage, the algorithm divides large datasets into several overlapping subsets, called canopies, using inexpensive and approximate distance measures. In the second stage, the elements present in the same canopies are further clustered using exact distance measurement. The canopies based approach convert large dataset clustering problem to several small dataset clustering problems and make large dataset clustering practical. The algorithm is computationally very economical without losing the clustering efficiency significantly. Canopies can have many applications in different domains and can be used in a collection of clustering methodologies, e.g., including Greedy Agglomerative Clustering [97], K-means [59], and Expectation-Maximization [56].

Chapter 3

Low Complexity Clustering Algorithms

This chapter presents the proposed image clustering algorithms. All the algorithms cluster images in a blind manner and need no prior information regarding the candidate image, the source cameras, the number of images contributed by each camera and the number of cameras. The first algorithm, which is used as a baseline for the other algorithms, has been named reduced complexity image clustering (RCIC) algorithm. The RCIC algorithm clusters images based on camera fingerprints. An additional and optional stage of attraction is also used to refine the clustering quality. Therefore, the RCIC algorithm can be implemented without and with attraction. The RCIC algorithm without attraction is abbreviated as RCIC, whereas the version with attraction is represented with RCIC-A. RCIC and RCIC-A are simple but very efficient clustering algorithms with a reduced computational cost and provide a solution to the $NC \gg SC$ problem.

The fast image clustering based on the fingerprint ordering (FICFO) algorithm further reduces computational complexity. Ordering depends on the quality of the fingerprints. The ordering factor called ranking index $\Re I$ is computed for each fingerprint using the image features that usually correlate with fingerprint quality and all fingerprints are arranged in descending order of $\Re I$. The FICFO can also be implemented with attraction, in which case it is denoted as FICFO-A. The FICFO and FICFO-A have a reduced computational cost with respect to RCIC and RCIC-A algorithms, respectively. These algorithms are also robust to the $NC \gg SC$ problem.

The canopy based image clustering (CIC) algorithm uses two step clustering approach i.e., raw clustering and fine clustering, using two different threshold criteria. The CIC algorithm is composed of fingerprints sorting, raw clustering, fine clustering, and attraction stages. The raw clustering generates few large clusters using relaxed threshold criteria. The raw clusters are further processed using a stricter threshold in the fine clustering stage. The attraction process is applied to

refine the clustering quality. The CIC algorithm also has reduced computational cost as compared to the state-of-the-art.

The compressed fingerprints based image clustering (CFIC) algorithm is based on reduced and full camera fingerprints. The initial clustering is performed on reduced fingerprints, and full fingerprints are used for refining the clustering. The CFIC algorithm is composed of full and compressed fingerprints estimation, sorting of fingerprints, initial clustering, and fine clustering.

The proposed algorithms achieve a performance comparable to state-of-the-art algorithms, with a significantly lower computational cost. Large datasets are clustered with a considerably lower computational cost. These methods are suitable for large scale clustering and are robust to $NC \gg SC$ problem.

The above algorithms are discussed in detail in the following sections.

3.1 RCIC Algorithm

The RCIC technique explores the image dataset, without any prior information about source camera, number of cameras, number of images captured by a camera and candidate images. Since images come from different cameras with different dimensions, the size of all images is equalized by center cropping them. Then camera fingerprints, i.e., PRNU patterns, are extracted from all resized images. Each image is de-noised using Mihcak filter operation [86] and subtracted from the original image to get the noise residual W [14, 54] as given in Eq. 2.4 and Eq. 2.5. Then, the noise residual of each image is further processed to estimate PRNU fingerprint as given in Eq. 2.8. The PRNU is further processed using zero-meaning and Wiener filter to remove the suppress the periodic and non-periodic artifacts. The denoising filter i.e., Mihacak filter is implemented with a 4 level Wavelet decomposition using the Daubechies 8 tap Wavelet filter. The same process is adopted in all proposed clustering algorithms [86]. The resulting PRNU is used as camera fingerprint. For details refer to Section 2.2. A set of, initially un-clustered, camera fingerprints M is obtained from the dataset of images I, where each fingerprint is standardized to zero mean and unit norm as given in Eq. 3.1 and Eq. 3.2.

$$K_i = \{\frac{X_i W_i}{X_i^2} \wedge 1 \leqslant i \leqslant n, X_i \in I\} \tag{3.1}$$

$$M = \{F_i | F_i = \Phi\left(\boldsymbol{W}\left(K_i\right)\right) \wedge 1 \leqslant i \leqslant n\} \tag{3.2}$$

Where $\boldsymbol{W}$ is the Wiener filter operator, $\Phi(.)$ is the standardization function, n is the number of images in dataset, X_i is the i^{th} image in dataset and F_i is the camera fingerprint obtained from X_i.

The clustering algorithm is applied to the set of un-clustered fingerprints M_k, to construct k^{th} cluster C_k. Initially M_k is equal to M. All extracted fingerprints in

the dataset are in random order. The total number of fingerprints in the dataset is equal to $|M|$. To start clustering, an empty cluster C_k is initiated, and a fingerprint F_i is randomly selected as reference fingerprint RF_k and assigned to cluster C_k. The clustering is done by calculating the normalized cross-correlation (NCC) ρ between all other fingerprints F_i and reference fingerprint RF_k, as given by Eq. 2.9.

If the fingerprint F_i has an NCC ρ value greater than a threshold value Th, it is assigned to the cluster C_k; otherwise, the fingerprint F_i is left un-clustered. The threshold Th is calculated as given in Eq. 2.10. The probability of assigning to cluster C_k a fingerprint from a different camera is bounded by PFA.

After processing all fingerprints a total of $|M_k|-1$ correlation operations are performed to construct cluster C_k. The fingerprints grouped in cluster C_k are removed from the dataset M_k and we are left with $|M_k| - |C_k|$ un-clustered fingerprints.

To cluster the remaining fingerprints, new cluster C_{k+1} is initiated and a fingerprint F_i is randomly selected from M_{k+1} as reference fingerprint RF_{k+1}. The un-clustered fingerprints are processed by repeating the same procedure used for constructing the first cluster C_k. The algorithm stops when all fingerprints are assigned to a cluster. The implementation of the RCIC algorithm is presented in Algorithm 1. The implementation of RCIC algorithm is also explained in Figure 3.1, in detail.

Algorithm 1 RCIC Algorithm

Input: M : Set of Fingerprints, K : Cluster index, PFA : Probability of false alarm, d dimension of fingerprint

Output: $C_k : k^{th}$ Cluster

Initialization : $k = 1$, $M_k = randomize(M)$

$Th = \sqrt{2 \times \frac{1}{d}}\ erfc^{-1}(2 \times PFA)$
$N = |M_k|$
while $(N \neq 0)$
$UC_k = \emptyset$
$C_k = \emptyset$
$RF_k = F_i$
$C_K \leftarrow F_i$
for $j = 2$ *to* N **do**
  $\rho(j) = \frac{1}{d} \sum_{x=1}^{d} RF_K[x] F_j[x]$
  if $(\rho(j) \geq Th)$ **then**
    $C_k \leftarrow F_j$
  else
    $UC_k \leftarrow F_j$
  end if
end for
$k = k + 1$
$M_K = randomize(UC_k)$
$N = |M_k|$
endwhile

The constructed clusters are further refined by using an attraction stage. In attraction, an average reference fingerprint ARF_k is calculated by averaging all fingerprints in cluster C_k and standardizing it to zero mean and unit variance. The set A of average reference fingerprints is obtained as expressed in Eq. 3.3.

$$A = \{ARF_i | ARF_i = \Phi\left(\frac{\sum_{y=1}^{|C_i|} C_i[y]}{|C_i|}\right) \wedge 1 \leqslant i \leqslant k\} \tag{3.3}$$

Each of the average reference fingerprints is treated as a single fingerprint, and the previously adopted procedure is repeated. The clusters whose average reference fingerprints have an NCC ρ greater than threshold Th are merged; otherwise, the clusters are left unaffected. The pseudo code of the attraction process is expressed in Algorithm 2.

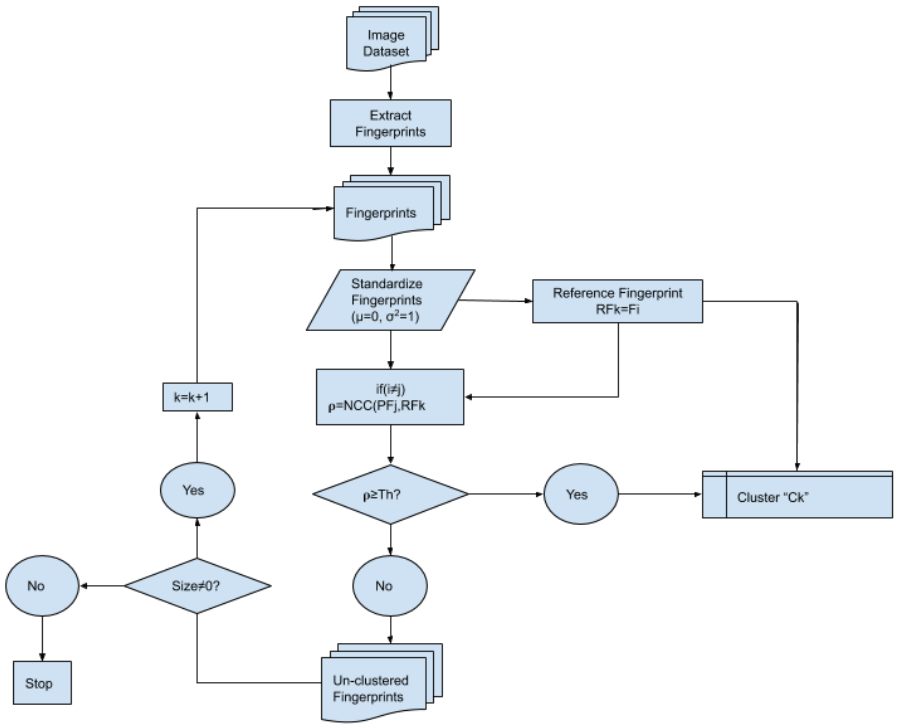


Figure 3.1: Implementation of RCIC algorithm without attraction.

Algorithm 2 Attraction

Input: A, PFA_h, d
Output: C_i, $Cost_{Att}$

Initialization : $Cost_{Att} = 0$
$Th_h = \sqrt{2 \times \frac{1}{d}}\ erfc^{-1}(2 \times PFA_h)$
$N = |A|$
while $(N \neq 0)$
$RF_i = ARF_i$
$A = A - ARF_i$
for $j = 1$ *to* $(N - 1)$ **do**
  $\rho(j) = \frac{1}{d} \sum_{y=1}^{d} RF_I[y] ARF_j[y]$
  $Cost_{Att} = Cost_{Att} + 1$
  if $(\rho(j) \geq Th_h)$ **then**
    $C_i = merge(C_i, C_j)$
    $A = A - ARF_j$
  else
    $C_i = C_i$
  end if
end for
$N = |A|$
endwhile

The attraction is an optional stage, and the proposed technique can be implemented with attraction as well as without attraction.

The attraction stage is applied to the clusters constructed with the RCIC algorithm. The RCIC and attraction stage are collectively called the RCIC-A algorithm. The complete implementation and clustering process of the RCIC-A algorithm is shown in Figure 3.2.

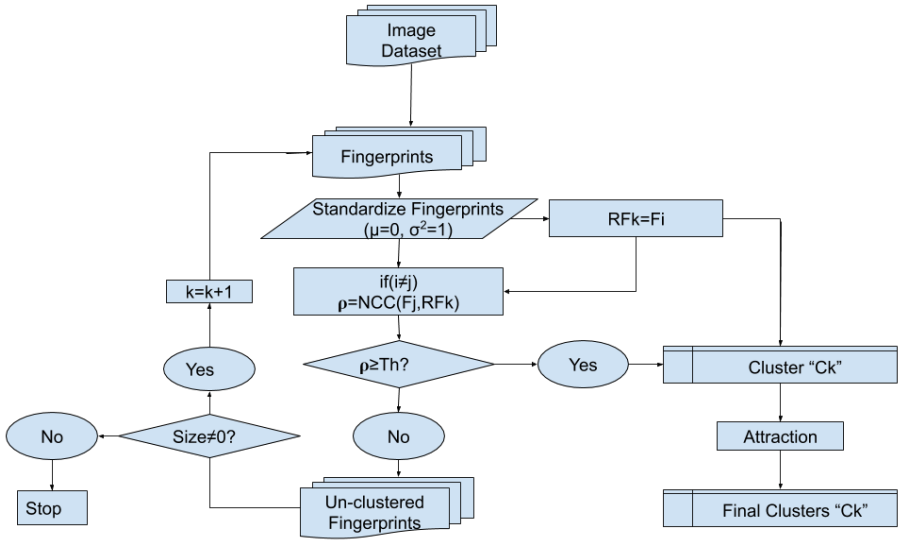


Figure 3.2: Implementation of RCIC-A algorithm.

3.1.1 Complexity, I/O Cost and RAM Requirement of RCIC and RCIC-A

The total complexity of the proposed clustering algorithm is measured in terms of the total number of NCC operations performed during the clustering. The total complexity of RCIC $T_{c_{(RCIC)}}$ and RCIC-A $T_{c_{(RCIC-A)}}$ are given by Eq. 3.4 and Eq. 3.5, respectively.

$$T_{c_{(RCIC)}} = \sum_{k=1}^{nc} (|M_k| - 1) \tag{3.4}$$

$$T_{c_{(RCIC-A)}} = \sum_{k=1}^{nc} (|M_k| - 1) + Cost_{Att} \tag{3.5}$$

Where, $|M_k| = |M_{k-1}| - |C_{k-1}|, \forall\ k \geq 2$ and $M_1 = M$, nc is the number of clusters constructed by the algorithm before attraction and $Cost_{Att}$ is the complexity of attraction stage and is evaluated experimentally.

Along with less computational complexity, the RCIC and RCIC-A algorithms have very low RAM requirements. The RCIC has a reference fingerprint RF and

one full fingerprint F loaded on RAM, at a time. As the reference fingerprint and full fingerprint have the same size, therefore we can say that two full fingerprints are present in RAM at a time. The maximum RAM requirement of the RCIC algorithm RAM_{RCIC} is given by Eq. 3.6.

$$RAM_{RCIC} = 64 \times (|RF| + |F|) = 64 \times 2 \times |F| \; bits \tag{3.6}$$

While the maximum RAM required by RCIC-A is higher than the RCIC as it also has the averaged reference fingerprints ARF loaded in the RAM. The maximum RAM usage of RCIC-A i.e., RAM_{RCIC-A}, is given by Eq. 3.7

$$RAM_{RCIC-A} = 64 \times nc \times |F| \; bits \tag{3.7}$$

The I/O cost of RCIC and RCIC-A is the same and is given by the Eq. 3.8 and Eq. 3.9, respectively.

$$I/O_{RCIC} = 64 \times \sum_{k=1}^{nc} |M_k| \times |F| \; bits \tag{3.8}$$

$$I/O_{RCIC-A} = 64 \times \left(\sum_{k=1}^{nc} |M_k| + n \right) \times |F| \; bits \tag{3.9}$$

Eq. 3.9 shows that the I/O cost of the RCIC-A algorithm is higher than the RCIC, because all the n fingerprints are read to estimate average reference fingerprints for attraction. The RCIC and RCIC-A have a high I/O cost. Which can be reduced by loading more than two fingerprints in RAM. However, it will increase the maximum RAM requirement. Therefore, a compromise can be made between the RAM requirement and the I/O cost.

3.2 FICFO Algorithm

Image clustering using the camera fingerprints would be more straightforward if the centroids of clusters, to attract other fingerprints of the same camera, were known in advance. But, in real life situations, such centroids are not available or known and, usually, we establish the centroids from the estimated fingerprints of the available images. It is assumed that fingerprints with lower estimation errors are closer to their centroid. The FICFO algorithm uses the same assumption and tries to sort camera fingerprints using inherent information of the respective images to predict fingerprint estimation quality.

According to the Cramer-Rao Lower Bound on the variance of the fingerprint [14, 78], the fingerprints estimated from dark or textured images are not reliable and do not represent the source cameras appropriately. Therefore, a better estimation of a fingerprint is made from a flat image that is uniformly bright and not saturated.

The fingerprints estimated from flat, uniformly vivid and unsaturated images have lower estimation errors and are more suitable to be used as centroid to attract other fingerprints of the same cameras.

Therefore, instead of selecting a random reference fingerprint e.g., in RCIC and RCIC-A, it is more suitable to select the best fingerprint with lower estimation error as reference fingerprints. The best reference fingerprints are deemed closer to the respective centroid, so as to help the attraction of fingerprints belonging to the same camera.

The detailed implementation of the algorithm is presented in the following subsections.

3.2.1 Fingerprints Ordering

All estimated fingerprints are sorted using a factor of goodness. This factor is termed as ranking index $\Re I$. The $\Re I$ is computed for each fingerprint using the inherent gray level, saturation and texture level of the corresponding image. We assume that fingerprints estimated from images with high $\Re I$ have lower estimation errors.

The average and normalized gray-level G and saturation S for an image X_i are calculated as in Eq. 3.10 and Eq. 3.11, respectively.

$$G_i = \frac{\sum_{j=1}^{d} X_i(j)}{255 \times d} \tag{3.10}$$

$$S_i = \frac{\sum_{j=1}^{d} (X_i(j) == 255)}{d} \tag{3.11}$$

Where d is the size of image X_i.

To calculate the texture T of an image X_i, we use a Laplacian filter [107] that highlights the regions of rapid gray-level change. The Laplacian L_i of an image X_i is computed as given in Eq. 3.12.

$$L_i = imfilter(X_i, A) \tag{3.12}$$

Where, $imfilter(.)$ denotes 2D filtering and A is a kernel that approximates the second order derivative [117].

$$A = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.13}$$

The texture level T_i is calculated as given in Eq. 3.14.

$$T_i = \frac{\sum |L_i|^2}{\sum |X_i|^2} \tag{3.14}$$

Finally, the $\Re I_i$ for image X_i is obtained by combining the values of G_i, S_i, and T_i according to Eq. 3.15.

$$\Re I_i = G_i^{\frac{1}{\alpha}} \times (1 - S_i)^{\frac{1}{\beta}} \times (1 - T_i)^{\frac{1}{\gamma}} \tag{3.15}$$

Where, α, β and γ are factors defining the contribution of G_i, S_i and T_i in RI_i, respectively. The $\alpha > 1$, $\beta > 0$ and $\gamma > 0$, the $\Re I$ increases with increase in α and γ, while decrease with increase in β. The changes are very fast for small values of these factors and after certain values a steady state is reached. As, these factor control the contribution of different features of image in $\Re I$, therefore, it is important to chose suitable values of these factor to compute $\Re I$.

Eq. 3.15 shows that flat images having a reasonable gray level but not saturated, result in high $\Re I$ and vice versa. The fingerprints estimated from images having high $\Re I$ have lower estimation errors.

Then, the images are processed for fingerprint estimation. A set of camera fingerprints M, standardized to zero mean and unit variance, is obtained from the images in dataset I as given in Eq. 3.2. The estimated fingerprints are arranged in decreasing order of $\Re I$, to get a set of sorted fingerprints M_S. These fingerprints are then used for clustering.

3.2.2 Fingerprint Clustering

At a generic clustering step denoted by index k, a cluster $C_k = \emptyset$ and a set of un-clustered fingerprints UC_k are considered. When we start clustering, i.e., $k = 1$, all un-clustered fingerprints are assigned to UC_k i.e., $UC_1 = M_S$. To construct C_k, the k^{th} cluster, the proposed algorithm always selects as reference fingerprint RF_k the first fingerprint from the set of sorted and un-clustered fingerprints UC_k and assigns it to cluster C_k. If the ranking index is consistent, RF_k will be the best estimated fingerprint among all the un-clustered fingerprints UC_k and the best representative of the respective cluster C_k. The NCC ρ between all other fingerprints F_i and reference fingerprint RF_k is calculated according to Eq. 2.9.

If the NCC ρ between fingerprint F_i and reference fingerprint RF_k has a value greater than a threshold value Th, F_i is assigned to the cluster C_k; otherwise the fingerprint F_i is attached to the set of un-clustered fingerprints UC_{k+1}. The threshold Th value is computed as given in Eq. 2.10.

While constructing the cluster C_k, a total of $|UC_k| - 1$ correlation operations are performed, and a total of $|UC_{k+1}| = |UC_k| - |C_k|$ fingerprints are left un-clustered.

To cluster the remaining fingerprints, if any, the cluster index k is incremented by 1, i.e., $k = k+1$ and the un-clustered UC_k fingerprints, are processed to construct a new cluster C_k by repeating the same procedure. The process continues till all fingerprints are assigned to a cluster and UC_{k+1} remains empty. The FICFO algorithm is presented in Algorithm 3. The implementation of the FICFO algorithm is also explained in Figure 3.3 with details.

Algorithm 3 FICFO Algorithm

Input: M : Fingerprints, $\Re I$: Ranking Indexes, k : Cluster index, PFA : Probability of false alarm, d dimension of fingerprint

Output: C_k : Cluster K , RF_k : Reference Fingerprint

Initialization : $k = 1$, $UC_k = M_S = sort(M, \Re I)$

$Th = \sqrt{2 \times \frac{1}{d}}\ erfc^{-1}(2 \times PFA)$
$N = |UC_k|$
while $(N \neq 0)$
$UC_{k+1} = \emptyset$
$C_k = \emptyset$
$RF_k = F_1$
$C_k \leftarrow F_1$
for $i = 2$ *to* N **do**
  $\rho(j) = \frac{1}{d} \sum x = 1_R^d F_k[x] F_i[x]$
  if $(\rho(j) \geq Th)$ **then**
    $C_k \leftarrow F_i$
  else
    $UC_{k+1} \leftarrow F_i$
  end if
end for
$k = k + 1$
$N = |UC_k|$
endwhile

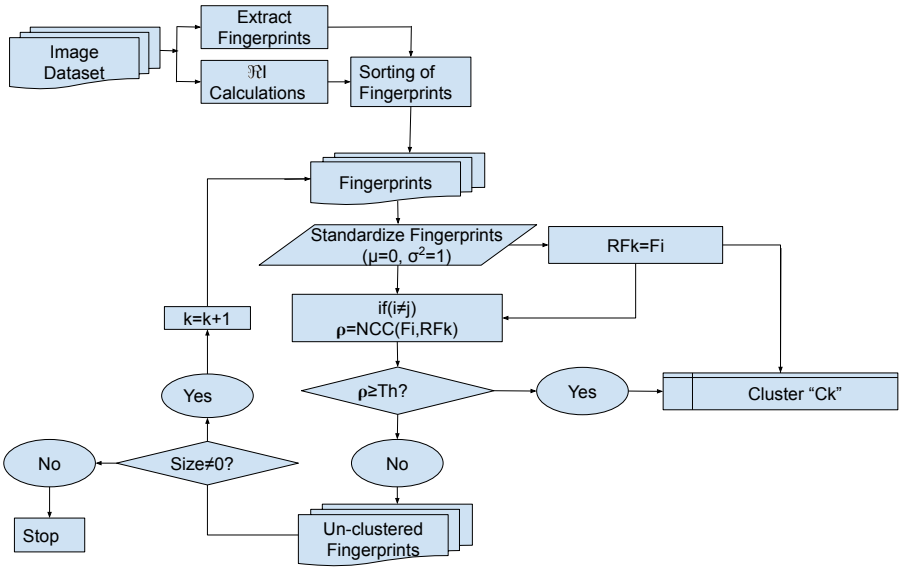


Figure 3.3: Implementation of FICFO algorithm without attraction.

The constructed clusters are further refined by using an attraction stage. In attraction, an average reference fingerprint ARF_k is calculated by averaging all fingerprints in cluster C_k and standardizing it to zero mean and unit variance. The set A of average reference fingerprints is obtained as expressed in 3.3. Each of the average reference fingerprints is treated as a single fingerprint. As each cluster is constructed by selecting the best fingerprints among the un-clustered fingerprints as a reference, therefore, we can assume that the constructed clusters are present in the descending order of their goodness. That is, the ranking used by FIFCO also propagates to the attraction process. Therefore, in the attraction process, the clusters are scanned according to their goodness and the average fingerprint of the first cluster among all clusters is selected as reference fingerprint and the NCC ρ between the reference fingerprint and all other averaged fingerprints is computed one by one. If the NCC ρ is greater than the threshold, the corresponding clusters are merged; otherwise, the clusters are left unaffected. This process continues until all average fingerprints are treated as reference fingerprint or their corresponding clusters are merged with some other cluster. The attraction process is explained in Algorithm 2.

To implement the FICFO-A technique of image clustering, the fingerprints estimation, sorting of fingerprints, clustering of fingerprints, and attraction process are used. The implementation of the FICFO-A technique is explained with the help of a diagram in Figure 3.4.

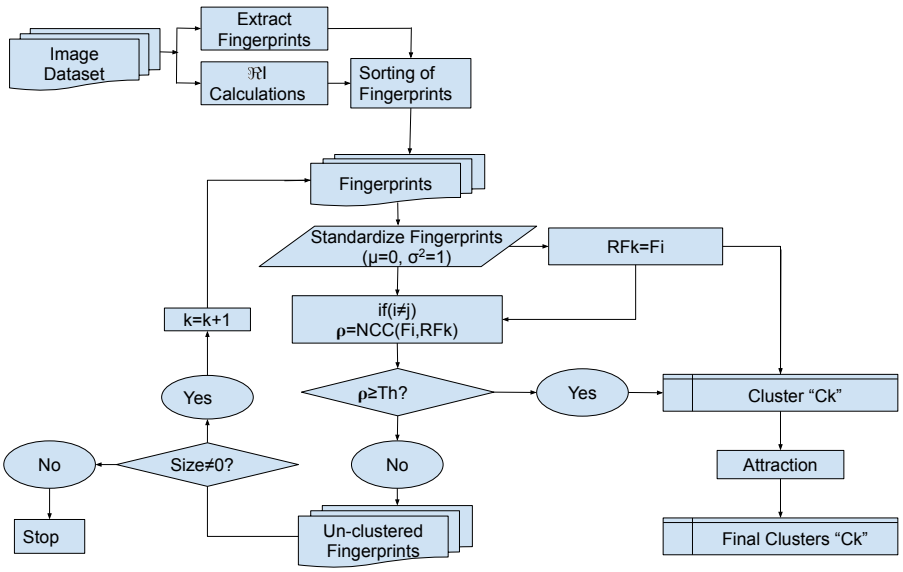


Figure 3.4: Implementation of FICFO-A algorithm.

3.2.3 Complexity, I/O Cost and RAM Requirement of FICFO and FICFO-A

The total complexity of the FICFO $T_{c_{(FICFO)}}$ and FICFO-A $T_{c_{(FICFO-A)}}$ are measured as given in Eq. 3.16 and Eq. 3.17, respectively.

$$T_{c_{(FICFO)}} = \sum_{K=1}^{nc} \left(|UC_K| - 1\right). \tag{3.16}$$

$$T_{c_{(FICFO-A)}} = \sum_{K=1}^{nc} \left(|UC_K| - 1\right) + cost_{att} \tag{3.17}$$

Where $cost_{att}$ is the complexity added by the attraction process and is evaluated experimentally. While, the total complexity of FICFO is same as FICFO-A except the attraction cost $cost_{att}$.

The FICFO and FICFO-A do the computation in calculating G, S, T, and $\Re I$ and sorting process. However, the computational cost of calculating G, S, T and RI is negligible concerning the estimation cost of fingerprints and complexity of sorting fingerprints is far less than computing correlation of very long vectors. Therefore, these are neglected in calculating the total complexity of t_c.

Similar to RCIC, the FICFO algorithm has very low RAM requirement and it always has only two fingerprints, i.e., reference fingerprint and one other full fingerprints, loaded in RAM. The maximum RAM occupied by FICFO algorithm is given in Eq. 3.18.

$$RAM_{FICFO} = 64 \times 2 \times |F| \tag{3.18}$$

While compared to RCIC and FICFO, the FICFO-A has large RAM requirements. The FICFO-A, similar to RCIC-A, have the averaged reference fingerprints ARF loaded in the RAM. The RAM occupancy reaches its peak when the clustering process end and the average fingerprints are computed for each cluster. All the average fingerprints remain in the RAM in bits. As the attraction process starts and clusters start merging, the load on RAM reduces gradually. The maximum RAM occupied by the FICFO-A is given by Eq. 3.19

$$RAM_{FICFO-A} = 64 \times nc \times |F| \; bits \tag{3.19}$$

The I/O costs of FICFO and FICFO-A are equal and is given by the Eq. 3.20 and Eq. 3.21, respectively.

$$I/O_{FIFCO} = 64 \times \sum_{K=1}^{nc} |UC_K| \times |F| \; bits \tag{3.20}$$

$$I/O_{FIFCO-A} = 64 \; times \left(\sum_{K=1}^{nc} |UC_K| + n\right) \times |F| \; bits \tag{3.21}$$

The Eq. 3.20 and Eq. 3.21 show that the FICFO has lesser I/O cost than the FICFO-A.

3.3 CIC Algorithm

The CIC algorithm takes its inspiration from the work of McCallum et al. [90]. McCallum et al. presented a canopy based technique to cluster large datasets. The main idea of the canopy based approach is to divide the large datasets into overlapping subsets using a cheap, approximate distance measure. These subsets are called canopies. These canopies are then considered as individual self-contained datasets. The canopies are then clustered using the exact measuring distances among the elements of a common canopy. The canopies approach convert the large scale clustering problem into a number of small scale clustering problem and make the large clustering problems possible that were formerly impossible. Using a suitable cheap distance metric reduces the computational complexity and without any loss in clustering accuracy. McCallum used the canopy based approach for clustering article citations into sets containing citations to the same article. We used the canopy based concept for image clustering based on camera fingerprints.

However, the CIC algorithm, instead of two different distance measures, uses a single distance measure in both stages. In the first stage, a relaxed threshold value is used to construct a few, non-overlapping large raw clusters i.e., canopies. The raw clusters are further clustered using a hard threshold value. In the end, refined clusters are obtained using the attraction process. The main difference between the McCallum et al. work and the CIC techniques is that McCallum et al. uses two different distance measures and creates overlapping subsets while the CIC technique uses normalized correlation as a single distance measure but with two different threshold values and creates non-overlapping subsets. Dealing with non-overlapping subsets makes the algorithm computationally more efficient. When using overlapping subsets, some candidates are present in more than one subset and are processed multiple times when constructing final clusters, while in the CIC algorithm, each candidate exists only in one subset so, it will be processed for clustering only in one subset.

The essential steps involved in the CIC techniques are camera fingerprints estimation, ranking index $\Re I$ calculation for each fingerprint, sorting of the fingerprints based on the ranking index, raw clusters i.e., canopies construction, fine clustering, and attraction.

3.3.1 Fingerprint Estimation and Sorting

As a preliminary step, camera fingerprints, i.e., PRNU patterns, are extracted from the images and standardized to zero mean and unit variance. Let's have a

dataset of randomly ordered images I, a set of camera fingerprints M, standardized to zero mean and unit norm, estimated from the images in dataset I using the mathematical expression as in Eq. 3.2.

The fingerprints are arranged in the descending order of $\Re I$. The $\Re I$ of each fingerprint is computed from the inherent information of the respective images. The $\Re I$ is computed as given in Eq. 3.15. The set of ordered fingerprints M_S is used for clustering.

3.3.2 Raw Clustering

After the estimation of camera fingerprints and computation of $\Re I$ for each image in the image dataset to be clustered, the camera fingerprints M are arranged in the descending order of $\Re I$ to get a set of sorted camera fingerprints M_S. The arranged fingerprints have the best fingerprint on top while the worst fingerprint at the bottom. Then fingerprints are clustered to construct raw clusters. The raw clustering is an iterative process and is repeated until each fingerprint is assigned to a raw cluster. For the construction of a cluster, the best fingerprint F_i among the un-clustered fingerprints is selected as reference fingerprint RF_{Kr} and is assigned to a raw cluster RC_{Kr}, where Kr represents the index of a raw cluster and iteration number. Then normalized correlation ρ between the reference fingerprint and all un-clustered fingerprints is calculated, one by one, as given by Eq. 2.9.

The fingerprints having ρ value equal to or greater than a threshold value are assigned to the same raw cluster RC_{Kr} as that of the reference fingerprint; otherwise, the fingerprints are assigned to a set of un-clustered fingerprints UC_{Kr}. In this way, one raw cluster RC_{Kr} is constructed in one complete iteration.

In order to create raw clusters, we use the same criterion based on NCC as for the previous algorithms; however, a larger value of PFA is used which, results in a relaxed threshold.

The set of un-clustered fingerprints is then processed repeating, the same procedure to construct another raw cluster, and so on. The cluster index Kr is incremented i.e., $Kr = Kr + 1$, and in each iteration, a raw cluster RC_{Kr} is constructed. Here, it is important to mention that in each iteration, the best fingerprint with the highest value of $\Re I$ is selected as reference fingerprint RF_{Kr}, from the set of un-clustered fingerprints. The process ends when all fingerprints are assigned to a raw cluster.

Due to the relaxed threshold, the fingerprints are easily attracted. That's why the raw clustering results in a few large non-overlapping clusters. Each large cluster may have fingerprints of different cameras in it. However, these clusters are constructed with a lesser computational cost. The raw clustering can be view as the division of a large dataset of un-clustered fingerprints into small datasets. Therefore, these raw clusters further processed to get fine clusters in the fine clustering stage, by considering each raw cluster as a dataset of un-clustered fingerprints.

Algorithm 4 Raw clustering algorithm

Input: M, Kr, PFA_r and d
Output: RC_{Kr} :
Initialization : $Kr = 1$, $UC_{Kr-1} = M_S$
$Th_r = \sqrt{2 \times \frac{1}{d}}\, erfc^{-1}(2 \times PFA_r)$
$N = |UC_{Kr-1}|$
while $(N \neq 0)$
$UC_{Kr} = \emptyset$
$RC_{Kr} = \emptyset$
$RF_{Kr} = F_i$ and $RC_{Kr} \leftarrow F_i$
for $j = 1$ *to* $(N-1)$ **do**
  $\rho(j) = \frac{1}{d} \sum_{x=1}^{d} RF_{Kr}[x] F_j[x]$
  if $(\rho(j) \geq Th_r)$ **then**
    $RC_{Kr} \leftarrow F_j$
  else
    $UC_{Kr} \leftarrow F_j$
  end if
end for
$Kr = Kr + 1$
$N = |UC_{Kr}|$
endwhile

3.3.3 Fine Clustering

The raw clusters have images from different source cameras and that need to be processed further to get fine clusters. The fine clusters are constructed by setting strict threshold criteria with a significantly low probability of false alarm PFA. The clusters are supposed to have images from the same camera. The fine clustering processes each raw cluster RC_{Kr} and initially considers it as a set of un-clustered fingerprints UC^{kr}_{Kf-1}. Where Kf is the clustering index of a fine cluster contracted, and Kr is the index of a raw cluster being processed. The fingerprints are already present in the order of decreasing $\Re I$ and the best fingerprint among the un-clustered ones is selected as reference fingerprint RF^{Kr}_{Kf} and is assigned to a fine cluster FC^{Kr}_{Kf}.

The NCC ρ between the reference fingerprint RF^{Kr}_{Kf} and an un-clustered fingerprint F^{Kr}_i of raw cluster RC_{Kr} is calculated as given in Eq. 2.9. The NCC ρ between the reference fingerprint RF^{Kr}_{Kf} and the each of the un-clustered fingerprints is computed one by one. The fingerprint having ρ value equal to or greater than the threshold, called hard threshold, is assigned to the same fine cluster FC^{Kr}_{Kf}. Otherwise, the fingerprint is attached to a set of un-clustered fingerprints UC^{kr}_{Kf}. The hard threshold Th_h value is computed as follows.

$$Th_h = \sqrt{2 \times \frac{1}{d}}\, erfc^{-1}\left(2 \times PFA_h\right) \tag{3.22}$$

Where PFA_h is the desired probability of false alarm for the strict threshold.

The un-clustered fingerprints are then processed using the same procedure, and fine clusters are constructed repeatedly. The process of fine clustering ends when all fingerprints in all raw clusters are assigned to a fine cluster. Here, it is essential to mention that the clustering index Kf is set to 1 at the start of the clustering of each raw cluster RC_{Kr} and is incremented with the construction of each new fine cluster from the same raw cluster RC_{Kr}. The fine clustering process ends when all fingerprints of all raw clusters are assigned to a fine cluster FC_{Kf}^{Kr}.

Algorithm 5 Fine clustering

Input: RC_{Kr}, Kf, PFA_h, d
Output: FC_{Kf}^{Kr}

Initialization : $Kf = 1$, $UC_0^{Kr} = \{\}$
for $Kr = 1$ *to* N_{RC} **do**
  $UC_{Kf-1}^{Kr} = RC_{Kr}$
  $Th_h = \sqrt{2 \times \frac{1}{d}}\, erfc^{-1}(2 \times PFA_h)$
  $N = |UC_{Kf-1}^{Kr}|$
  while $(N \neq 0)$
  $UC_{Kf}^{Kr} = \emptyset$
  $FC_{Kf}^{Kr} = \emptyset$
  $RF_{Kf}^{Kr} = F_i^{Kr}$
  $FC_{Kf}^{Kr} \leftarrow F_i^{Kr}$
  for $j = 1$ *to* $(N-1)$ **do**
    $\rho(j) = \frac{1}{d}\sum_{x=1}^{d} RF_{Kf}^{Kr}[x] F_j^{Kr}[x]$
    if $(\rho(j) \geq Th_h)$ **then**
      $FC_{Kf}^{Kr} \leftarrow F_j^{Kr}$
    else
      $UC_{Kf}^{Kr} \leftarrow F_j^{Kr}$
    end if
  end for
  $N = |UC_{Kf}^{Kr}|$
  $Kf = Kf + 1$
  endwhile
end for

The fine clustering creates various purified non-overlapping clusters, most of them are singleton clusters. These clusters are of good quality, and the chances of placing two fingerprints from two different cameras, in the same cluster are

negligible. However, a large number of singleton clusters is an issue. We can not ignore these singleton clusters because some cameras may possibly have only a single candidate image in the dataset and the singleton clusters represent the respective cameras. At the same time, it is also possible that the singleton clusters are generated because the fingerprints are not correctly classified. Therefore, the quality of the clusters is further refined using the attraction process.

3.3.4 Attraction

The attraction process is used to refine the clustering quality and merge the clusters having fingerprints from the same camera. To start attraction, we have a set of N_{fc} fine clusters. To avoid confusion and make the symbols simple, let's consider a set Z of fine clusters, each represented by FC_i.

$$Z = \{FC_1, FC_2, FC_3, FC_4, ...FC_{N_{fc}-1}, FC_{N_{fc}}\} \tag{3.23}$$

Then average fingerprint AF_i is computed for each fine cluster FC_i by averaging all member fingerprints of the fine cluster FC_i and the standardizing to zero mean unit variance. Hence, we get a set of average fingerprints A, with each AF_i representing the corresponding fine cluster FC_i. textcolorblueThe set A is obtained as expressed in Eq. 3.3.

The average reference fingerprints are present in the same order as that of the reference fingerprints used in fine clustering, i.e., the ranking propagates also to the attraction process. Then the best average fingerprint AF_i is selected as reference fingerprint RF_{Ka} for attraction process and NCC ρ between the reference fingerprint RF_{Ka} and all other average fingerprints AF_i is calculated as given in Eq. 2.9.

If the NCC ρ between average fingerprint AF_i and the reference fingerprint RF_{Ka} has a value higher than a threshold value Th_h as given in Eq. 3.22, the corresponding fine clusters are merged. Otherwise, the fine cluster is left as it is. The process stops when each of the average fingerprints is either used as reference fingerprint for attraction or the respective fine cluster is merged with other fine clusters. The implementation of the attraction process is explained in Algorithm 2. The computational cost of the attraction process $Cost_{Att}$ is evaluated experimentally. The attraction process improves clusters quality which is reflected in the improvement of recall R and $F - measure$ but, along with this, it contributes to computational complexity.

The implementation of CIC algorithm has been demonstrated here with the help of the tree diagram in Figure 3.5.

Figure 3.5 shows the initial the fingerprints are clustered using the raw clustering process and a small number of canopies or raw clusters RC are obtained, which are further processed using fine clustering and each cluster is divided into several fine clusters FC. This process builds several high quality clusters.

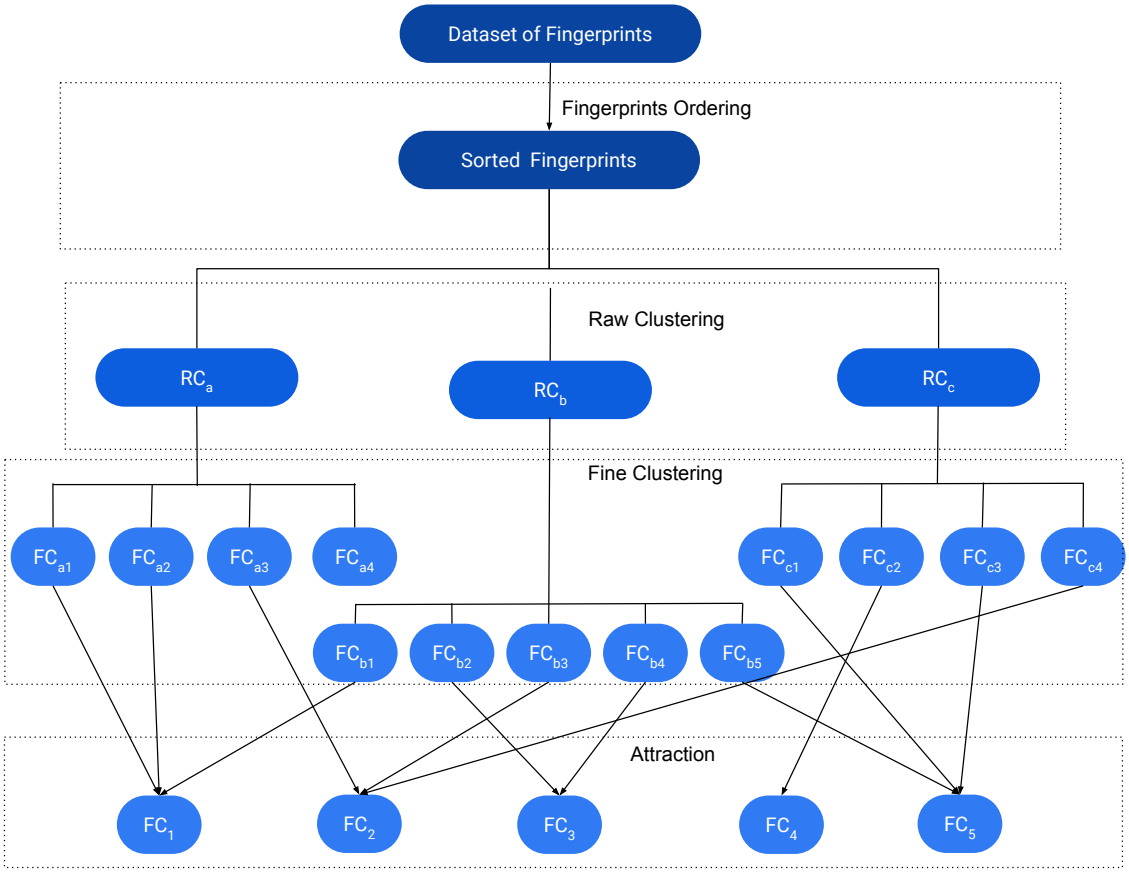


Figure 3.5: Block diagram of the CIC algorithm.

3.3.5 Complexity, I/O Cost and RAM Requirement of CIC

The CIC algorithm consists of raw clustering, fine clustering and attraction and each of the processes has its own computational complexity. The raw clustering, while constructing a raw cluster RC_Kr, in an iteration Kr, performs $|UC_{Kr-1}| - 1$ correlation operations. If a total of N_{RC} raw clusters are constructed and the total computational cost of the raw cluster t_r, is given by Eq. 3.24.

$$t_r = \sum_{Kr=1}^{N_{RC}} (|UC_{Kr-1}| - 1) \tag{3.24}$$

The fine clustering process computes $|UC_{Kf}^{Kr}| - 1$ correlations while constructing a fine cluster FC_{Kf}^{Kr}. If a total of Nf_{Kr} fine clusters are constructed from a raw cluster RC_{Kr}, then the total computation cost t_f of fine clustering is given by Eq. 3.25.

$$t_f = \sum_{Kr=1}^{N_{RC}} \sum_{Kf=1}^{Nf_{Kr}} \left(|UC_{Kf-1}^{Kr}| - 1 \right) \tag{3.25}$$

Where N_{RC} is the total number of raw clusters and Nf_{Kr} is the number of fine clusters generated from Kr^{th} raw cluster. The total number of fine clusters N_{fc} is given as

$$N_{fc} = \sum_{r=1}^{N_{RC}} Nf_{Kr}. \tag{3.26}$$

Adding the computational cost of the attraction process, the total computational cost T_c of the proposed algorithm is equal to the sum of the computational cost of raw clustering T_r, cost of fine clustering T_f and cost of attraction $Cost_{Att}$.

$$T_c = T_r + T_f + Cost_{Att} \tag{3.27}$$

Now, to discuss the RAM requirements of the CIC algorithm, the CIC algorithm always has a maximum of two fingerprints loaded on RAM during the raw clustering and fine clustering process. However, at the start of the attraction, the RAM load reaches its maximum. Because the average reference fingerprint is computed for each fine cluster and the average reference fingerprints remain in the RAM. Therefore, the maxim RAM requirement of the CIC algorithm in bits in given by Eq. 3.28.

$$RAM_{CIC} = 64 \times N_{fc} \times |F| \; bits \tag{3.28}$$

The maximum RAM requirement of the CIC algorithm is higher than that of RCIC, RCIC-A, FICFO and FICFO-A as the total number of fine clusters N_{fc} before the attraction process is larger than the number of clusters nc of RCIC-A and FICFO-A before attraction.

The I/O cost of the CIC algorithm is also higher than the RCIC, RCIC-A, FICFO and FICFO-A and is given by Eq. 3.29.

$$I/O_{CIC} = 64 \times \left(\sum_{Kr=1}^{N_{RC}} \sum_{Kf=1}^{Nf_{Kr}} |UC_{Kf-1}^{Kr}| + \sum_{Kr=1}^{N_{RC}} |UC_{Kr-1}| + n \right) \times |F| \; bits \tag{3.29}$$

3.4 CFIC Algorithm

The CFIC algorithm uses both reduced and full fingerprints for image clustering. The fingerprints are sorted in descending order of $\Re I$. The primary clustering is done with the reduced fingerprint. The clusters are refined using full fingerprints. The use of reduced fingerprints contributes to a significant reduction in

computational complexity. Here it is worth noting that, differently from [101, 102], the CFIC and other proposed algorithms, we have proposed, do not exclude any image from the clustering process based on darkness, saturation, and texture level. The removal of saturated and dark images improves the clustering quality of [101, 102] and enhances the precision of clusters. While the proposed algorithms do not exclude any image based on saturation and darkness. However, the removal of saturated and dark images will also help to improve the quality of the proposed algorithms. The implementation of the CFIC algorithm consists of the following processes.

- Fingerprint estimation
- Fingerprint compression
- Sorting of fingerprints
- Initial clustering
- Fine clustering

3.4.1 Fingerprints Estimation

The camera fingerprints are estimated using the procedure explained in Section 2.2.

3.4.2 Fingerprint Compression

The compression of camera fingerprints is one of the key processes in the implementation of the proposed algorithm. The compression process reduces the size of the camera fingerprint, which in turn helps to reduce the computational cost and memory requirement of the clustering process. The compression is done in several ways, and several techniques have been proposed to compress the camera fingerprints. These include trimming and cropping [87], digest[87], Gaussian random projections [115] and binarization [6]. Trimming unwrap camera fingerprint column-wise and trim the fingerprint by preserving only the first P_r samples. While cropping preserves only the center portion of the camera fingerprint and unwrap the cropped camera fingerprint. The fingerprint digest technique builds a digest by keeping the P_r highest energy components and their positions, from the camera fingerprints. The digest technique relies on the assumption that the most prominent peaks of the extracted camera fingerprints can be used as a suitable camera attribute. The Gaussian random projections based compression technique was introduced by Valsesia et al. The basic idea is to project the one-dimensional unwrapped camera fingerprint from a vector space of large dimension to a subspace

of reduced dimension P_r. This technique is a very effective way of camera fingerprint compression. The binarization technique is a very effective way of reducing the bit-rate. This can be used even after the earlier mentioned camera fingerprints compression techniques. The binarization technique transforms camera fingerprint from a real number to a binarized version by performing an element-wise quantization operation.

In the proposed algorithm, fingerprint compression is done in two different manners. The first way to get a reduced fingerprint is to use decimation, random projection computation, and dead-zone quantization process. The random projections process determines the total number of elements of the reduced fingerprint. The reduced fingerprints obtained by this method are approximate representations of their respective full camera fingerprints. The second way of computing a reduced fingerprint is by applying the dead-zone quantization directly on the decimated camera fingerprint. The reduced fingerprint obtained has the same number of elements as in decimated fingerprint, but the size is significantly reduced by the dead-zone quantization. Each process, i.e., decimation, random projection, and dead-zone quantization, introduces some sort of noise to final reduced fingerprints. But, despite these noises, the reduced fingerprints are good enough to be used for clustering, instead of using full camera fingerprints.

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

Decimation

It is known from the literature [37], that when a camera fingerprint is estimated from the lossy compressed image, the statistical properties of the detection change. For example, the JPEG compression increases the variance of cross-correlation values between the noise residual. In [9], Bondi et al. considered the estimated fingerprints extracted from flat-field and natural images to analyze the effect of JPEG compression on the power spectral density (PSD) of noise residual for different quality factors. The analysis reveals that increasing compression lowers the power of the residue at high spatial frequencies. While, the residual, i.e., the camera fingerprint, contributions in high-frequency bins are combined with residuals of blockiness artifacts from JPEG compression that cannot be removed entirely by the residue extraction process [9].

The previously mentioned observations are kept in mind, and a straightforward approach of decimation is adopted. The decimation reduces the dimensionality of the camera fingerprint F. The decimation process attenuates the high-frequency components by decimating F by a factor $df > 1$ along rows and columns. The cubic kernel $H_c(z)$ [67] is used to decimate the fingerprint via interpolation as given in Eq. 3.30.

$$h_c(z) = \begin{cases} 1.5|z|^3 - 2.5|z|^2 + 1 & \text{if } |z| \geq 1 \\ -0.5|z|^3 + 2.5|z|^2 - 4|z| + 2 & \text{if } 1 < |z| \leq 2 \\ 0 & \text{otherwise} \end{cases}. \tag{3.30}$$

Given a vector y of length L_y, if the vector y is decimated by a factor df then the i^{th} element of the decimated vector y_d is obatined as given in Eq. 3.31[9].

$$y_d(i) = \sum_{j=0}^{L_y-1} h_c(j - i.df).y(j), \ \forall i \in \{0, ..., \lfloor L_y/df \rfloor\}. \tag{3.31}$$

After applying a decimation process on a camera fingerprint F, we get a decimated camera fingerprint F_d of reduced size $|F|/df^2$. The decimation process results in a loss in performance. Therefore, the decimation factor df is chosen in such a way that the detection performance of the reduced fingerprint F_d is not affected severely.

Random Projection

After decimation, the next step is to generate random projections for the given decimated camera fingerprint F_d. The random projections are obtained using the Gaussian sensing matrix [115], which has proven to be an effective way of compressing camera fingerprints. The sensing matrix Ψ of dimension $|F_d \times P_r|$ is generated with sample being extracted from a i.i.d zero-mean Gaussian distribution. The decimated camera fingerprint F_d is column-wise unwrapped with P_r random projections. The resulting projection RP of the decimated camera fingerprint F_d is obtained by taking a matrix product between the sensing matrix Ψ and the decimated camera fingerprint F_d.

$$RP = \Psi \odot F_d \tag{3.32}$$

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

Dead-Zone Quantization

Binarization of random projections is an effective technique to preserve good performance in terms of detection [9]. In order to generalize binarization, here we adopt the dead-zone quantization approach, as suggested in [9]. The dead-zone quantizer processes the random projections RP and generates reduced fingerprint Fr. Given σ, the standard deviation of RP, the i^{th} element of Fr for $i = 1, ..., P_r$ is obtained as

$$Fr\left(i\right) = \begin{cases} +1 & \text{if } RP\left(i\right) > \delta\sigma \\ 0 & \text{if } -\delta\sigma \leq RP\left(i\right) \leq \delta\sigma \\ -1 & \text{if } RP\left(i\right) < -\delta\sigma \end{cases}. \tag{3.33}$$

Where the δ is the quantization factor. Higher values of δ will increase the probability of zero samples in the F_r. Therefore, the value of δ is carefully chosen.

The dead-zone quantization has two advantages; first, it preserves the peaks, which are very important in terms of cross-correlation. Secondly, the variable threshold of the quantization process allows reducing the bit-rate of Fr via entropy coding by increasing δ while keeping P_r fixed. The dead-zone quantization, as given in Eq. 3.33 results in Fr with three intensity levels i.e., $+1$, 0 and -1. With proper entropy codes, we can use fewer bits to represent the intensity levels of F_r; however, a simpler two-bit encoding facilitates the computation of similarity scores between quantized fingerprints. Hence, the size of the Fr in terms of bits will be $2 \times P_r$ bits.

Algorithm 6 Method 1: Camera Fingerprint Compression

Input: F : Camera Fingerprint, df : Decimation Factor, P_r : Number of Random Projections, Ψ : Sensing Matrix, δ : Zero sample controlling factor

1: $F_d = Decimate(F, df)$

2: $RP = \Psi \odot F_d$

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

Algorithm 7 Method 2: Camera Fingerprint Compression

Input: F : Camera Fingerprint, df : Decimation Factor, P_r : Number of Random Projections, δ : Zero sample controlling factor

1: $F_d = Decimate(F, df)$

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

3.4.3 Ranking Index Computation and Sorting of Fingerprints

The $\Re I$ for each image is calculated as explained in FICFO using Eq. 3.15. The reduced and full fingerprints i.e., F_r and F are arranged in the descending order of

$\Re I$ to get a set of sorted fingerprints M_O. Which are further used for clustering.

3.4.4 Initial Clustering

The processes of camera fingerprint estimation, computation of compressed camera fingerprints, $\Re I$ computation, and sorting of camera fingerprints based on $\Re I$, are followed by clustering. The clustering is performed in two steps. The first step of clustering is named as Initial clustering and 2^{nd} stage is called fine clustering. The initial clustering is performed using decimated and quantized camera fingerprints, also called reduced camera fingerprints Fr. The clusters are refined in the fine clustering step using full camera fingerprints, hereafter named as full fingerprints F. The clustering process works iteratively, following different rounds and each round results in a cluster. The clustering round is generally denoted by cluster index K. At the start of each round, an empty cluster $C_K = \{\}$, is initiated. The set of un-clustered fingerprints is represented by UC_K.

During the initial stage, the proposed algorithm uses the reduced camera fingerprints Fr to cluster images. At the start of clustering, the cluster index K is set to one, i.e., $K = 1$, all sorted camera fingerprints are assigned to the set of un-clustered fingerprints UC_K i.e., $UC_1 = M_O$. The K^{th} cluster C_K^r, is constructed by selecting a reference fingerprint RF_K^r. The CFIC algorithm selects the best-reduced fingerprint F_r from the set of sorted and un-clustered fingerprints UC_K as reference fingerprint. The full fingerprint F corresponding to F_r is assigned to cluster C_K^r. If the ranking index is consistent, RF_K^r will be the best-estimated fingerprint among all the un-clustered fingerprints UC_K^r and the best representative of the respective cluster C_K^r. The normalized cross-correlation (NCC) ρ between all other fingerprints Fr_i and reference fingerprint RF_K^r is used to decide whether the given F_{r_i} belongs to the same camera as that of RF_K^r, or not.

The reduced camera fingerprints are quantized in the dead zone quantization process, and each entry of these has only three possible values of $+1$, 0 and -1. If the probability of zeros entries is P_o, then the probability of $+1$ assumed equal to the probability of -1, is $(1 - P_o)/2$. Hence, the variance of reduced fingerprints is $1/\sqrt{P_r(1 - P_o)}$. The NCC ρ between Fr_i and RF_K^r is given by in Eq. 3.34.

$$\rho\,(i) = \frac{1}{\sqrt{P_r\,(1 - P_o)}} \sum_{x=1}^{P_r} RF_K^r[x] Fr_i[x] \tag{3.34}$$

where P_r is the dimension of the reduced fingerprint FR_i.

If the NCC ρ between the reduced fingerprint F_i^r and reference fingerprint RF_K^r has a value greater than or equal to a threshold value Th, F_i is assigned to the cluster C_K^r. Otherwise, the reduced fingerprint F_i^r and corresponding full fingerprint F_i are attached to the set of un-clustered fingerprints UC_{K+1}. The threshold value is computed as expressed in Eq. 3.35.

$$Th = \sqrt{2 \times \frac{1}{P_r}}\, erfc^{-1}\left(2 \times PFA\right) \tag{3.35}$$

According to the Central Limit Theorem (CLT), the NCC ρ between two d-dimensional normalized fingerprints, X and Y, from different cameras approximately follows a normal distribution with zero mean and $1/|X|$ variance, i.e., $\rho(X,Y) \sim N(0,1/|X|)$ [89]. The same supposition can used for reduced fingerprints. Because if we have two normalized reduced fingerprints X_r and Y_r, of size $|X_r|$, then the NCC will have a variance of $\frac{1}{|X_r|}$. Hence, it can be said that the normalized reduced fingerprints also follow normalized distribution with zero mean and $\frac{1}{|X_r|}$ variance.

While constructing the cluster C_K^r, a total of $|UC_K| - 1$ correlation operations are performed, and a total of $|UC_{K+1}| = |UC_K| - |C_K^r|$ fingerprints are left un-clustered.

To cluster the remaining fingerprints, if any, the cluster index K is incremented by 1, i.e., $K = K + 1$ and the un-clustered UC_K fingerprints, are processed to construct a new cluster C_K^r by repeating the same procedure. The process continues till all fingerprints are assigned to a cluster and UC_{K+1} gets empty.

At the end of each round K a cluster C_K^r is constructed, which is composed of only full fingerprints F. The full fingerprints F in each cluster C_K^r are merged by taking the average of them, to compute a full reference fingerprint RF_K^f for each cluster C_K^r. The full reference fingerprints RF_K^f is then used in the fine clustering stage to attract other clusters.

The clustering based on reduced camera fingerprints is a convenient way of grouping the fingerprints because cross-correlation is performed between the reduced and quantized camera fingerprints. The clustering creates quality clusters. But it also generates small clusters, most of which are singleton clusters. Moreover, the process requires very little memory.

The fingerprint clustering algorithm is explained here as:

Algorithm 8 Algorithm of Initial Clustering

Input: M_O, K, PFA, P_r
Output: C_K^r,RF_K^f

Initialization : $K = 1$, $UC_K = |M_O|$
$Th = \sqrt{2 \times \frac{1}{P_r}}\ erfc^{-1}(2 \times PFA)$
while $(|UC_K| \neq 0)$
$UC_{K+1} = \emptyset$
$C_K^r = \emptyset$
$RF_K^r = Fr_1$
$C_K^r \leftarrow F_1$
for $j = 2$ *to* $|UC_K|$ **do**
  $\rho(j) = \frac{1}{\sqrt{P_r(1-P_o)}} \sum_{x=1}^{P_r} RF_K^r[x] Fr_j[x]$
  if $(\rho(j) \geq Th)$ **then**
    $C_K^r \leftarrow F_j$
  else
    $UC_{K+1} \leftarrow Fr_j$
    $UC_{K+1} \leftarrow F_j$
  end if
end for
$RF_K^f = \frac{\sum_{i=1}^{|C_K^r|} F_i}{|C_K^r|}$ where $F_i \in C_K^r$
$K = K + 1$
endwhile

3.4.5 Fine Clustering

The fingerprint clustering in the previous stage generates several clusters; some of them are singleton clusters. These clusters may be less accurate than those obtained using full fingerprints because they are created by using the reduced camera fingerprints, which have been obtained by quantizing the Gaussian random projection of the decimated version of full camera fingerprints. These reduced camera fingerprints are the approximate fingerprints, not the exact fingerprints. There is a possibility of improving the clusters' quality. Therefore, the clusters are further processed using fine clustering.

The fine clustering process uses the full reference fingerprints RF_K^f for the possible attraction of clusters C_K^r. Initially, we have a set of full reference fingerprints M_{RF^f}. All the reference fingerprints are treated as un-clustered fingerprints. The fine clustering index H is initiated and set to one i.e., $H = 1$ and all reference fingerprints are assigned to the set of un-clustered fingerprints UC_H i.e., $UC_1 = M_{RF^f}$. To construct A fine C_H^F, the H^{th} cluster, the proposed algorithm always selects

as reference fingerprint RF_H^f the first full reference fingerprint RF_1^f from the un-clustered full fingerprints UC_H and all the fingerprints in the corresponding cluster C_H^r are assigned to the fine cluster C_H^F, i.e., $C_H^F \leftarrow C_H^r$. The NCC ρ between the full reference fingerprint RF_H^f all other full reference fingerprints RF_i^f is calculated one by one as given in Eq. 2.9.

If the NCC ρ, between the reference fingerprints, is greater than or equal to a threshold value Th, all the fingerprints in cluster C_i^r, are assigned to the cluster C_H^f and the reference fingerprints are merged; otherwise, the reference fingerprint RF_i^f is assigned to the set of un-clustered fingerprints UC_{H+1} and the corresponding C_i^r is left unaffected. The threshold value of Th is computed as given in Eq. 2.9.

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

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

The fine clustering uses the average of multiple full fingerprints F as reference fingerprint RF_H^f in round H and the average reference fingerprints are more stable and reliable [86]. Therefore, the clusters constructed are more reliable. The fine clustering results in good quality clusters and lesser in number than that constructed in the initial clustering.

The fine clustering algorithm is explained here as:

Algorithm 9 Algorithm of Fine clustering

Input: M_{RF^f}, H, RF_H^f, PFA
Output: C_H^f

Initialization : $H = 1$, $UC_K = |M_{RF^f}|$

$T = \sqrt{2 \times \frac{1}{|F|}}\, erfc^{-1}(2 \times PFA)$
while $(|UC_H| \neq 0)$
$UC_{H+1} = \emptyset$
$C_H^f = \emptyset$
$RF_H^f = RF_1^f$
$C_H^f \leftarrow C_H^r$
for $j = 2$ *to* $|UC_H|$ **do**
  $\rho(j) = \frac{1}{|F|} \sum_{x=1}^{|F|} RF_H^f[x] RF_j^f[x]$
  if $(\rho(j) \geq T)$ **then**
    $C_H^f \leftarrow C_j^r$
    $RF_H^f = \frac{(RF_H^f + RF_j^f)}{2}$
  else
    $UC_{H+1} \leftarrow RF_j^f$
    C_j^r un-affected
  end if
end for
$H = H + 1$
endwhile

With the completion of fine clustering, we reach the end of the clustering process, and we have fine clusters in hand. The implementation of the CFIC algorithm is also explained in the block diagram shown in Figure 3.6.

3.4.6 Complexity, I/O Cost and RAM Requirements of CFIC

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

Computation Complexity

The CFIC algorithm performs initial clustering on reduced fingerprints and the clusters are then refined using average full fingerprints during the fine clustering process. The two stages of clustering contribute to computational cost. The computation cost is important in clustering, since this is directly related to the execution

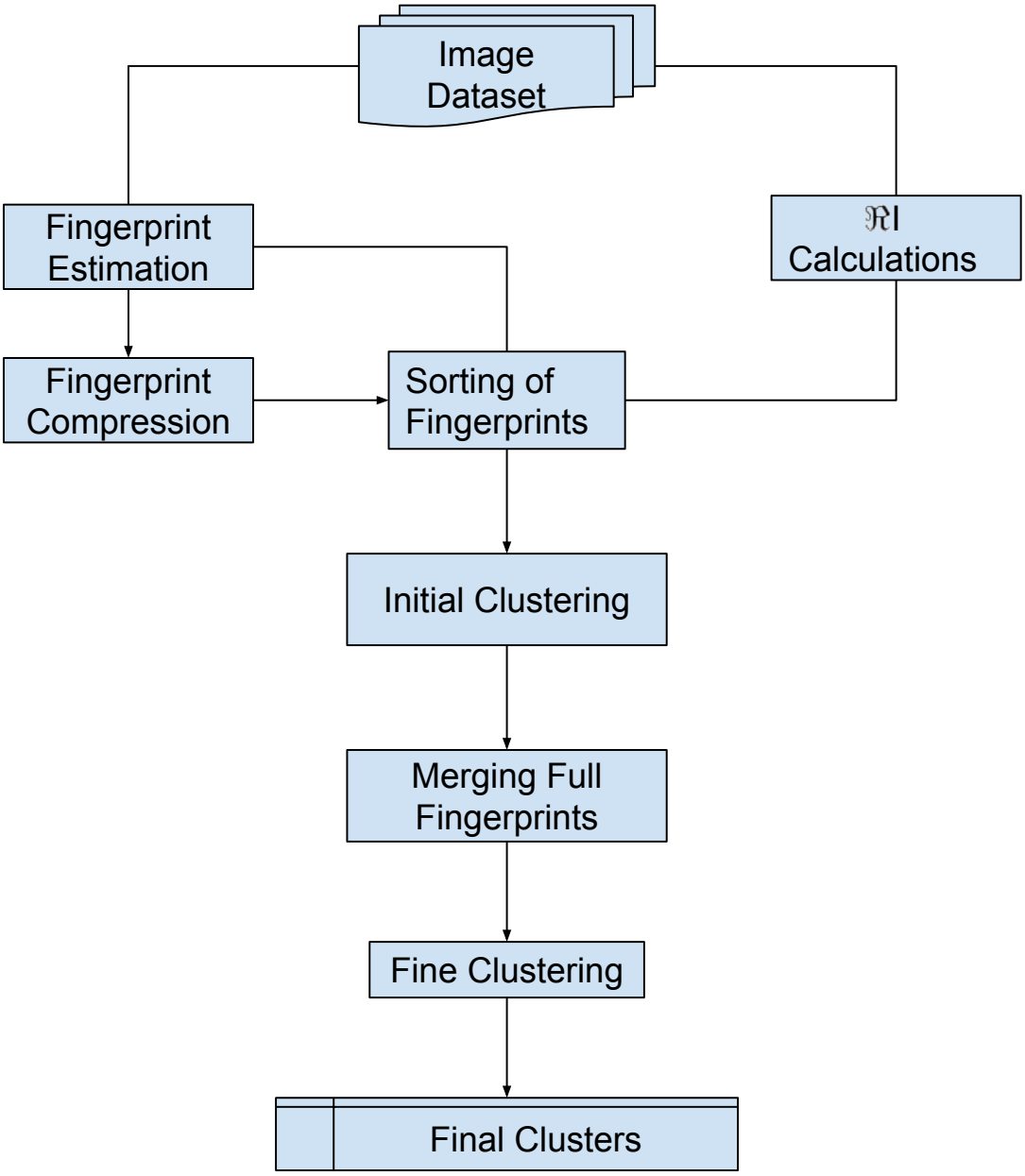


Figure 3.6: Block diagram of the CFIC algorithm.

time needed to cluster images. The computational cost T_c^r of the initial clustering stage is given by Eq. 3.36.

$$T_c^r = \zeta \times \left(\sum_{i=1}^{NCr} |UC_i| - NCr \right) \tag{3.36}$$

Where NCr, is the total number of clusters constructed in initial clustering and ζ is the ratio between the sizes (in bits) of the reduced and full fingerprints. As the compressed fingerprints are made of ternary symbols $\{-1\ ,\ 0,\ 1\}$, therefore the correlation can be computed using simple sums and avoiding multiplications. For example the correlation of the compressed fingerprints is equivalent to the Opposite Absolute Distance (OAD) [9], which uses only sums to compute the OAD. This implementation can help to reduce run time.

The computational cost of the fine clustering is measured only in terms of the total number of correlations performed. The computational cost T_c^f of the fine clustering stage is given by Eq. 3.37.

$$T_c^f = \sum_{i=1}^{NCf} |UC_i| - NCf \tag{3.37}$$

Where, NCf is the total number of clusters obtained after fine clustering.

The total computational cost T_c^t of the proposed algorithm is the sum of the computational cost of fine clustering T_c^f, scaled computational cost of initial clustering stage T_c^r and the cost of merging fingerprints $Cost_{merging}$, given by Eq. 3.38.

$$T_c^t = T_c^f + T_c^r + Cost_{merging} \tag{3.38}$$

$$T_c^t = \left(\sum_{i=1}^{NCf} |UC_i| - NCf\right) + \zeta \times \left(\sum_{i=1}^{NCr} |UC_i| - NCr\right) + Cost_{merging} \tag{3.39}$$

The merging of fingerprints requires only additions equal to the number of full fingerprints in the dataset while the correlations are larger in number and require multiplications. The cost of merging is negligible as compared to the combined cost of the correlations of both initial clustering and fine clustering. Therefore the cost of merging is neglected from the total cost of CFIC, which is significantly less than the reference complexity i.e., $n(n-1)/2$, when the number of images per camera SC is greater than or equal to 2 i.e., $SC \geq 2$, while for $SC = 1$ the computation cost is comparable to the reference complexity. For example, if we have a dataset of n fingerprints with $SC = 1$ and the fingerprints are correctly clustered without any false positive, then the CFIC will perform $\zeta(n(n-1)/2)$ and $n(n-1)/2$ correlations in initial and fine clustering respectively. And the total cost of clustering will be $(\zeta+1)(n(n-1)/2)$ which is $(\zeta+1)$ times larger than the reference complexity i.e., $n(n-1)/2$. However, if $SC \geq 2$, the computational complexity of the proposed algorithm, as compared to the reference complexity, decreases as the size of the dataset increases. This makes the algorithm suitable for large scale clustering.

RAM requirement

The CFIC algorithm, during the initial clustering has one reduced fingerprint Fr_j, and a reduced reference fingerprint RF_K^r is RAM. The maximum RAM occupied during the initial clustering always remains constant is equal to $2 \times P_r$ bits. The RAM occupancy reaches its peak when the full fingerprints of each constructed cluster are merged together by averaging them. The average full fingerprints remain in the RAM and are used in the fine clustering stage. The RAM load decreases with the possible merging of clusters during the fine clustering. The maximum RAM RAM_{CFIC} required by CFIC algorithm is given by Eq. 3.40.

$$RAM_{CFIC} = 64 \times \left(\sum_{i=1}^{NCr} NCr \times |F| \right) \ bits \tag{3.40}$$

I/O Cost

Here it is also important to discuss the I/O cost of the CFIC algorithm. The I/O cost of the algorithm depends on the number of clusters constructed in the initial clustering. The I/O cost I/O_{CFIC} of the CFIC algorithm is given in Eq. 3.41.

$$I/O_{CFIC} = 2 \times \left(P_r \times \sum_{i=1}^{NCr} |UC_i| + 32 \times |F| \times n \right) \ bits \tag{3.41}$$

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

Chapter 4

Experimental Results

In this section, we provide experimental validation of the proposed clustering algorithms. We validate the performance of our algorithms under different settings, on datasets exhibiting different size and distribution of the clusters. The performance of algorithms is also analyzed for the $NC \gg SC$ scenario.

4.1 Datasets

The proposed clustering algorithms have been evaluated on the Dresden image database [33, 34]. The algorithms are evaluated for both small scale and large scale clustering. It is very challenging to cluster images of different cameras of the same model based on camera fingerprints. This is due to the shared in-camera processing of images by different devices of the same model. Therefore, based on the fact mentioned, the datasets are classified as easy and hard datasets. The easy datasets include images taken by cameras of different distinct models, while the hard datasets are composed of images from different devices of similar models. Along with this, in real scenarios, the images taken by the cameras vary in number, which is responsible for different contributions of cameras in a dataset. According to different distributions, the datasets are classified into symmetric and asymmetric ones. In symmetric datasets, all cameras contribute equally to the dataset while, in asymmetric, the contribution is not equal.

We have up to 53 cameras in the Dresden dataset [33, 34]. If we fix the number of cameras, the proposed algorithms can be analyzed in the $NC \gg SC$ scenario, by varying the average number of images per camera SC. Hence, to have a meaningful analysis of the behaviour of the proposed algorithms under the $NC \gg SC$ problem, we need a larger number of cameras. The dataset with a large number of cameras is built using the images of the existing cameras. The image of an existing camera is divided into different non-overlapping patches of size 1023×1023 which are considered as separate images. The patches do not share any part of the camera

sensor; therefore, each created image will have unique PRNU. The part of a sensor array which has captured the particular patch is considered as a single camera. By using this procedure, a dataset of images coming from 295 "virtual" cameras and a contribution of 20 images by each camera is generated. The dataset can be classified as symmetric hard as it has equal contributions from different cameras of different brands and models.

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

- $D0$: Small dataset. It consists of 600 images taken by 15 cameras, each equally contributing 40 images. The 15 cameras are of different models and cover 8 popular camera brands.
- $D1$: Easy symmetric dataset. It consists of 1,000 images taken by 25 cameras, each equally contributing 40 images.The 25 cameras are of different models and covering 8 popular camera brands, such as Canon, Nikon, Olympus, Pentax, Samsung, and Sony.
- $D2$: Easy asymmetric dataset. The dataset is also composed of 1,000 images taken by the same 25 cameras as in $D1$. These camera alternatively contribute 20, 30, 40, 50 and 60 images.
- $D3$: Hard symmetric dataset. It consists of 1,000 images taken by 50 cameras, each contributing 20 images. The 50 cameras only cover 12 popular models, so some of them are of the same model.
- $D4$: Hard asymmetric dataset. The same 50 cameras as in $D3$ are part of this dataset, alternatively contributing 10, 15, 20, 25 and 30 images.
- $D5$: Large Scale dataset. It consists of 5900 images from 295 cameras, each contributing 20 images.
- $D6$: Dresden dataset. The dataset is composed of 10960 images from 53 cameras of 18 different models and 10 different brands.

The $D0$ dataset is used to investigate the parameters of the algorithms e.g., finding the suitable sigma δ for computing the compressed fingerprint in the CFIC algorithm and α β and γ in FICFO. The other four datasets, i.e., $D1$, $D2$, $D3$, and $D4$, are used for examining the performance of the algorithms on different types of small and medium datasets. These datasets are also used for comparing the proposed algorithms among them and with state-of-the-art algorithms. The Dresden database is used for large scale clustering analysis and investigating the $NC \gg SC$ problem. While $D5$ dataset is used for the extensive analysis of the proposed algorithms under the $NC \gg SC$ problem.

As datasets have images from different cameras of different models and different sizes, they result in images and fingerprints of different sizes too. Therefore to avoid

fingerprints with different sizes and cluster images based on camera fingerprints, the images used for clustering are center cropped to 1023×1023 pixels. Camera fingerprints are extracted from the images using the process explained in Section 2.2 [86, 14].

4.2 Evaluation Metrics

A measure of agreement is essential for the evaluation and comparison of the clustering algorithms with state-of-the-art techniques. Therefore, evaluation and comparison of clustering algorithms are done using a suitable evaluation metric. Some of the metrics, used for assessment of clustering algorithms, are based on the matching of sets, e.g., Precision P, Recall R and F-measure, on information theory, e.g., mutual information MI and normalized mutual information NMI, and pair of objects counting e.g., rand index RI and adjusted rand index ARI [109, 118]. However, the clustering solutions with more clusters results in higher values of NMI when compared with ground truth classes [3]. This may be misleading while comparing different clustering algorithms yielding different numbers of clusters. For this reason, MI and NMI are not used in this thesis. RI and ARI count the pair of objects, either in the same cluster or different clusters. The RI is equal to 1 if two groups agree entirely. The RI faces a problem that the expected value of the RI of two random partitions does not take a constant value (say zero). The problem is corrected by the ARI that assumes the generalized hyper-geometric distribution as the model of randomness. The ARI has the maximum value of 1, and its expected value is 0 in the case of random clusters. Hence, there is a wider range of values that the ARI can take on, thus increasing the sensitivity of the index. A larger ARI means a higher agreement between two partitions. Therefore, ARI is recommended for measuring agreement even when the partitions compared have different numbers of clusters [95].

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

$$\Omega = \{\omega_1, \omega_2, \omega_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\omega_{NC}\} \tag{4.1}$$

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

$$C = \{c_1, c_2, c_3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots c_y\} \tag{4.2}$$

Where each c denotes a set of fingerprints assigned to a cluster.

The precision P and recall R are calculated from the classes and clusters as given in the following equations.

$$P = \frac{\sum_k (max_j |c_k \cap \omega_j|)}{\sum_k |c_k|} \tag{4.3}$$

$$R = \frac{\sum_j (max_k |c_k \cap \omega_j|)}{\sum_j |\omega_j|} \tag{4.4}$$

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

The $F - measure$ is calculated using P and R as

$$F - measure = 2 \times \frac{(P \times R)}{(P + R)}. \tag{4.5}$$

All the metrics have values between 0 and 1, 0 being the worst and 1 being the best.

The RI and ARI are computed using Eq. 4.6 and Eq. 4.7, respectively [103, 48, 124].

$$RI = \frac{(a + d)}{a + b + c + d} = \frac{(a + d)}{\binom{n}{2}}. \tag{4.6}$$

Where, a is the number of pairs of fingerprints which are in the same set in Ω and in the same set in C, b is the number of pairs of fingerprints which are in the same set in Ω and in different sets in C, c is the number of pairs of fingerprints which are in different sets in Ω and in the same set in C and d is the number of pairs of fingerprints which are in different sets in Ω and in different sets in C [49].

$$ARI = \frac{RI - \mathbf{E}[RI]}{1 - \mathbf{E}[RI]}. \tag{4.7}$$

Where $\mathbf{E}[RI]$ is the expected value of RI.

Along with the quality of clusters of an algorithm, it is also essential to know how fast an algorithm is and for this purpose and a metric called complexity reduction Cr [70, 69] is introduced. For Cr, we consider as reference complexity the computation of all pairwise correlations in a set of n fingerprints and the value t_c estimates the number of pairwise correlations performed by the algorithm under test. The Cr is computed as given by Eq. 4.8 and it gives the relative complexity of an algorithm with respect to the reference complexity $n(n-1)/2$.

$$Cr = \frac{n \times (n - 1)}{2 \times t_c}. \tag{4.8}$$

Where, t_c is the total computational cost of the clustering and n is the size of the dataset. The total complexity t_c of BCFIC, RCIC, RCIC-A, FICFO, FICFO-A, and CIC is computed in the same manner as all of these algorithms use full camera fingerprints of the same size. However, the total complexity of the LSIC and CFIC algorithms is computed in a bit different way. The two algorithms use two different types of fingerprints of different sizes, called reduced and full fingerprints. But the reduced fingerprints in the case of LSIC are computed in a different way than the CFIC algorithm. The size of the reduced fingerprint used in the CFIC algorithm is much less than that of LSIC, thanks to dead-zone quantization. Therefore, the number of correlation operations performed on reduced and full fingerprints are weighed differently. In case of LSIC, the total complexity t_c is calculated as $t_c = ncf + (r/d) \times ncr$, where, ncf and ncr are the number of correlation among full and reduced fingerprints respectively. While, in case of CFIC, total computational cost is computed as given $t_c = ncf + (\zeta \times ncr)$, where $\zeta = \frac{P_r}{32 \times |F|}$. While P_r and $|F|$ are the dimensions of the reduced fingerprints and full fingerprints, respectively. Here it is important to mention that the FICFO, FICFO-A, CIC and CFIC algorithms perform some computation while calculating G, S, T and $\Re I$ and also in sorting fingerprints. However, the cost of calculating G, S, T, and $\Re I$ is negligible with respect to the estimation of fingerprints. The cost of sorting fingerprints is also far less than computing correlations of very long vectors. Therefore, the cost of computing $\Re I$ and sorting fingerprints is neglected, while computing the total computational complexity t_c and Cr.

4.3 Performance of RCIC and RCIC-A

The RCIC and RCIC-A algorithms have been evaluated on the Dresden image database [33, 34], and sub datasets, as discussed in 4.1. The PFA is set to 10^{-6}, to compute threshold Th as given by Eq. 2.10 and is used in all the subsequent experiments. The same setup is used throughout experimentation.

4.3.1 Effect of Randomization

As the clustering algorithm randomly selects reference fingerprints to construct clusters, each experiment is repeated a different number of times to obtain an average performance metric. The RCIC algorithm is applied to the $D0$ dataset, as given in Section 4.1. The clustering process is repeated a different number of times i.e., 25, 20, 15, and 10 times. The precision P, recall R, and $F-measure$ are computed in each of these experiments. The variation in the parameters in different experiments is checked by calculating the variance of the evaluation parameters, i.e., $\sigma^2(P)$, $\sigma^2(R)$ and $\sigma^2(F - measure)$. The results obtained are listed in Table 4.1. The experimental results show that the values of $\sigma^2(P)$, $\sigma^2(R)$ and $\sigma^2(F-measure)$

vary very little, and the variation can be ignored. The small values of $\sigma^2(P)$, $\sigma^2(R)$ and $\sigma^2(F-measure)$ show that the RCIC algorithm is very stable.

Table 4.1: Variance of evaluation measures of RCIC for different No. of experiments.

No. of Experiments	$\sigma^2(P)$	$\sigma^2(R)$	$\sigma^2(F-measure)$
10	4.468×10^{-6}	1.118×10^{-4}	4.137×10^{-5}
15	1.737×10^{-6}	1.349×10^{-4}	5.110×10^{-5}
20	1.982×10^{-5}	1.773×10^{-4}	7.504×10^{-5}
25	1.276×10^{-6}	1.510×10^{-4}	5.955×10^{-5}

Similarly, the RCIC-A algorithm is applied to cluster the images in the dataset $D0$, and the experiments are repeated the same 25, 20, 15 and 10 times. The precision P, recall R and F-measure $F-measure$ are computed for each experiment and the variance of the evaluation parameters i.e., $\sigma^2(P)$, $\sigma^2(R)$ and $\sigma^2(F-measure)$ is computed. The experimental results are listed in Table 4.2. The experimental results show that the values of $\sigma^2(P)$, $\sigma^2(R)$ and $\sigma^2(F-measure)$ do not vary significantly and the variances recorded are even much lower than that of RCIC algorithm. The small values of $\sigma^2(P)$, $\sigma^2(R)$ and $\sigma^2(F-measure)$ in the case of both RCIC and RCIC-A algorithm show that these algorithms are very stable. Hereafter, clustering is repeated 15 times for each experiment and the average values of evaluation metrics are reported for both RCIC and RCIC-A.

Table 4.2: Variance of evaluation measures of RCIC-A for different No. of experiments.

No. of Experiments	$\sigma^2(P)$	$\sigma^2(R)$	$\sigma^2(F-measure)$
10	5.054×10^{-8}	1.851×10^{-6}	4.893×10^{-7}
15	3.001×10^{-8}	1.879×10^{-6}	5.750×10^{-7}
20	2.855×10^{-8}	2.231×10^{-6}	7.980×10^{-7}
25	1.957×10^{-8}	1.988×10^{-6}	6.149×10^{-7}

4.3.2 Small Scale Clustering

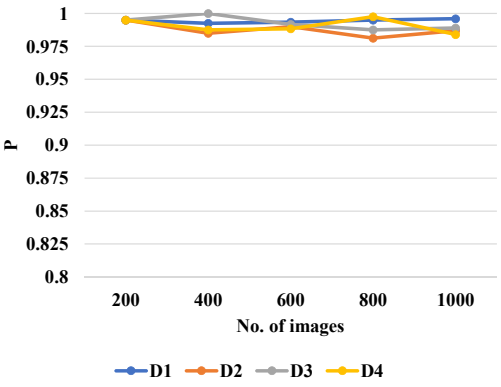
The small scale and large scale clustering are equally crucial for forensic experts. As discussed in Section 4.1, the small datasets can have the same or different contributions from different cameras and the number of cameras may also vary. Therefore, We are investigating the performance of the RCIC and RCIC-A algorithms on all the four different small datasets, i.e., $D1$, $D2$, $D3$ and $D4$. The experiments are performed on these datasets considering the different average number of images SC per camera while keeping the number of cameras NC fixed. The number of

cameras in symmetric easy and asymmetric easy datasets is 25, while in symmetric hard and asymmetric hard datasets, the number of contributing cameras NC is 50.

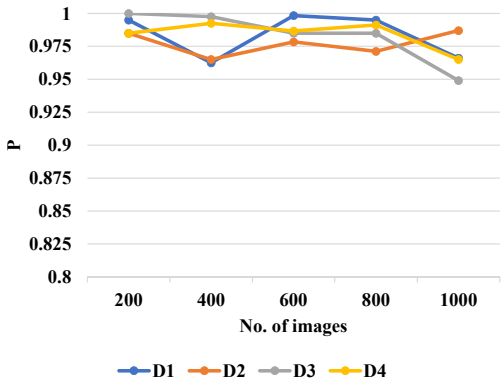
A number of 200, 400, 600, 800 and 1000 fingerprints are selected in each experiment. The number of images is selected in such a way that only the number of images per camera SC changes and the number of cameras NC do not change for each dataset. Therefore, the 200, 400, 600, 800, and 1000 images correspond to a SC of 8, 16, 24, 32 and 40 respectively in case of easy datasets i.e., $D1$ and $D2$. While, in the case of hard datasets i.e., $D3$ and $D4$, the mentioned number of images related to a SC of 4, 8, 12, 16 and 20 respectively. The fingerprints are clustered using the RCIC and RCIC-A algorithms. Each experiment is repeated 15 times and P, R, $F-measure$, RI, ARI and Cr are computed in each experiment. The average values of P, R, $F-measure$, RI, ARI, and Cr are used for performance analysis of the RCIC and RCIC-A algorithm.

The experimental results show that the RCIC and RCIC-A algorithms, when applied to $D1$, $D2$, $D3$, and $D4$ with the different number of images per source camera, generate good quality clusters. The evaluation metrics measured are shown in Figure 4.1. The experimental results show that both RCIC and RCIC-A result in a high P for all datasets and the different numbers of images, as shown in Figure 4.1a and Figure 4.1b. It can be observed that P of the RCIC algorithm remains constant while small fluctuations occur in the P of RCIC-A. This fluctuation can be due to some false merging of clusters during the attraction stage. Closely observing the values of P, obtained from experiments, a slight reduction in P occurs with an increase in SC.

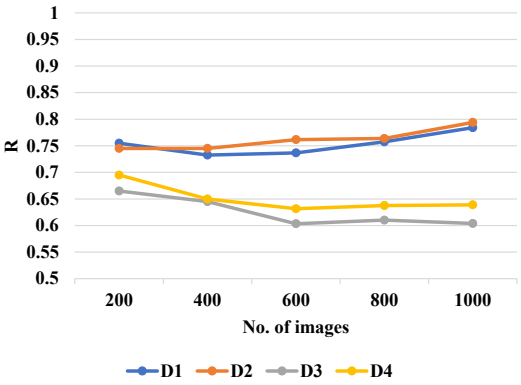
Considering the metrics R and $F-measure$ of RCIC and RCIC-A algorithm, it can be observed that both versions of the algorithm perform better on easy datasets as compared to hard datasets, as shown in Figures 4.1c-f. The results also reveal that the performance of the RCIC algorithm on the asymmetric dataset is relatively better than on symmetric datasets. However, due to the attraction stage, the R and $F-measure$ of RCIC-A algorithm is higher than the RCIC algorithm. The RCIC results in some singleton clusters which are attracted and merged with the other closest clusters, in the attraction stage in the RCIC-A algorithm. Hence, the RCIC-A gives a clustering result that is closer to the ground truth and results in higher values of R and $F-measure$.



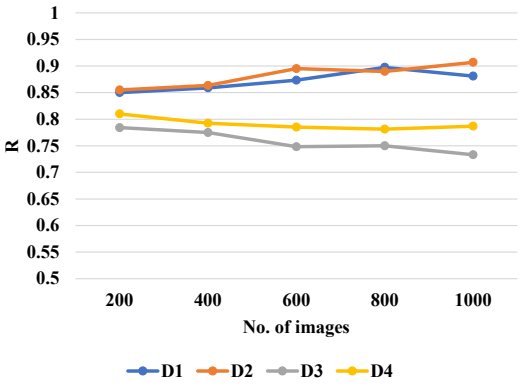
(a) P of RCIC



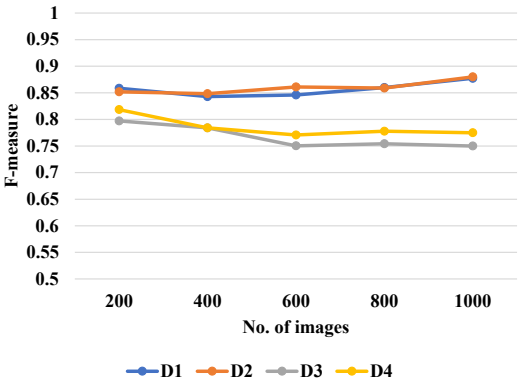
(b) P of RCIC-A



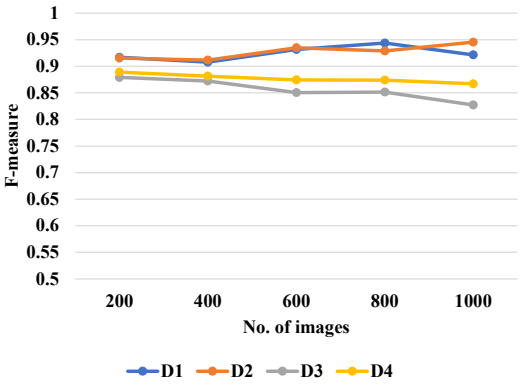
(c) R of RCIC



(d) R of RCIC-A

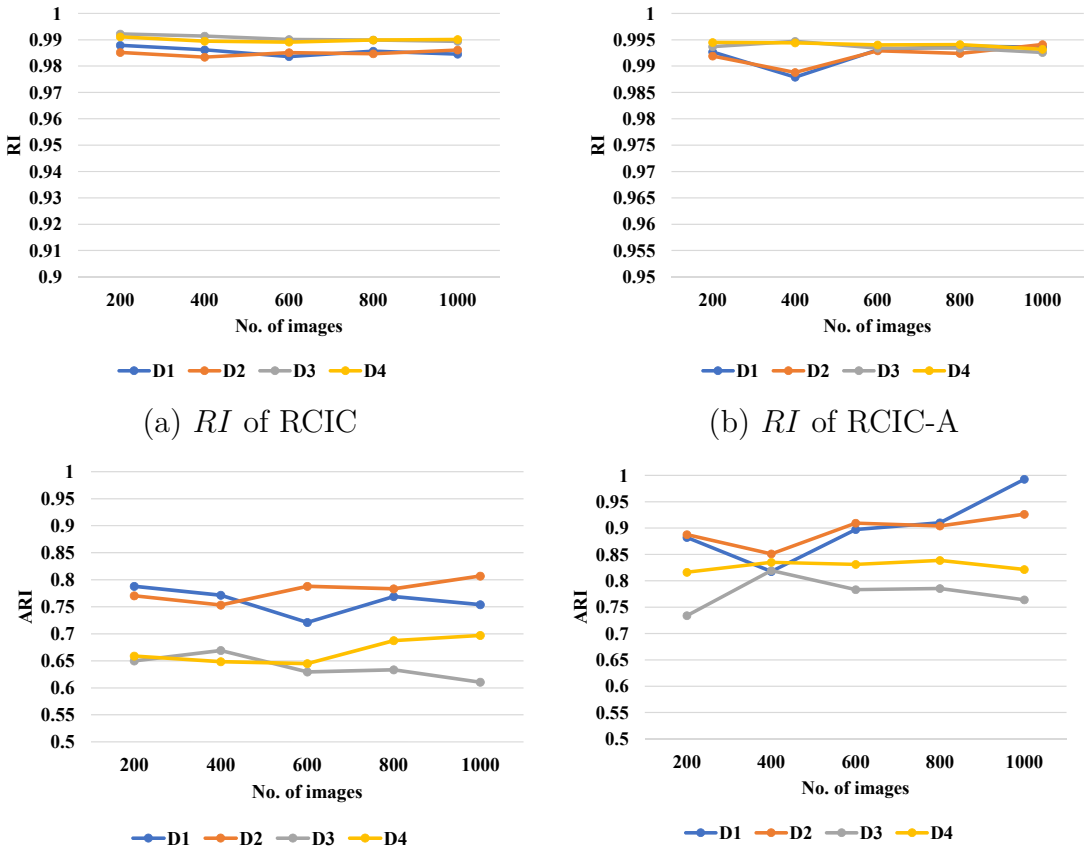


(e) $F-measure$ of RCIC



(f) $F-measure$ of RCIC-A

Figure 4.1: P, R and $F-measure$ of the RCIC and RCIC-A algorithm on small datasets.



(a) *RI* of RCIC

(b) *RI* of RCIC-A

(c) *ARI* of RCIC

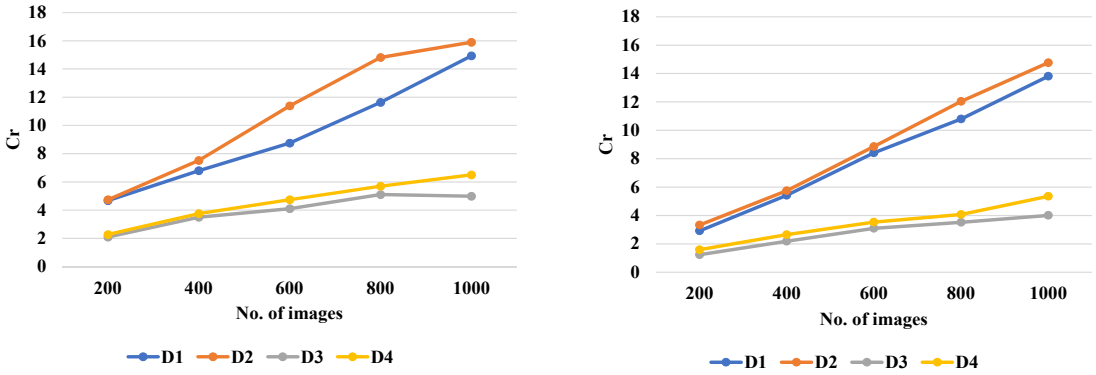
(d) *ARI* of RCIC-A

Figure 4.2: *RI* and *ARI* of the RCIC and RCIC-A algorithm on small datasets.

The P, R, and $F-measure$ are highly dependent on the number of constructed clusters. A few singleton clusters can affect these evaluation measures. Therefore, to evaluate the strength of the RCIC and RCIC-A algorithms, the evaluation metrics of *RI* and *ARI* are used. The two parameters are independent of the number of clusters and depend highly on the quality of clusters. The experimentally obtained results of *RI* and *ARI* are shown in 4.2. The results show that both RCIC and RCIC-A algorithms result in high values of these evaluation metrics. The *RI* for RCIC and RCIC-A is high and remains constant and does not vary with the change in the number of images of different types of datasets. A little drop in *RI* is observed for easy datasets at 400 images. But, this is not much significant. The results obtained show that RCIC and RCIC-A result in higher values of *RI* on hard datasets with respect to easy datasets. The *ARI* is also good for the different numbers of images of different datasets. The *ARI* of the RCIC algorithm does not vary significantly; however, for RCIC-A, the *ARI* remains stable for hard datasets and increases for the easy dataset. The increase in the *ARI* is due to the attraction

process. The performance of RCIC and RCIC-A algorithms, in terms of ARI is higher on the easy dataset as compared to hard datasets.

The construction of a high quality of clusters is essential while analyzing a clustering algorithm. But, at the same time, the computational cost is also crucial. It is highly desired to construct clusters with the least possible computational cost. The complexity reduction of Cr is a parameter used in this work, to describe the computational cost. The Cr of the RCIC and RCIC-A algorithm computed in the different experiments is shown in Figure 4.3. The experimental results show that the Cr of both the RCIC and RCIC-A algorithm increases with an increase in the number of images used in clustering, and hence the relative computational cost decreases. The results also show that RCIC-A has less Cr than RCIC due to the additional computational cost of the attraction process. The RCIC and RCIC-A have higher Cr on easy datasets than hard dataset and on asymmetric datasets than symmetric datasets.



(a) RCIC (b) RCIC-A

Figure 4.3: Cr of the RCIC and RCIC-A algorithm on small datasets.

4.3.3 Medium and Large Scale Clustering

After the analysis of RCIC and RCIC-A algorithms on different small datasets, it is important to study the performance of these clustering algorithms on medium and large datasets. Therefore, in this section, the RCIC and RCIC-A algorithms are applied to different subsets of images selected from Dresden [33, 34]. Different numbers of images are considered from the Dresden datasets and are clustered with RCIC and RCIC-A algorithm. The images are selected in such a manner that cameras NC is set fixed, i.e., $NC = 53$, and the average number of images from each camera SC is varied. For number of images, the evaluation metrics i.e., P, R, $F - measure$, RI and ARI are computed. Along with the evaluation measures, the complexity reduction, Cr is also computed for each experiment.

The experimental results of P, R and $F-measure$ for both RCIC and RCIC-A are shown in Figure 4.4. The results demonstrate that both techniques perform well for different sizes of datasets and different SC. The results show that as the number of images increases, P of the RCIC algorithm is almost constant, and R and $F-measure$ also do not change significantly. While, in the case of RCIC-A, the R and $F-measure$ are stable, but P decreases due to the attraction of some wrong clusters, when the number of images and SC grows. The results obtained reveal that for smaller values of SC, the RCIC, and RCIC-A algorithm performs very well. This shows that the RCIC and RCIC-A algorithms do not suffer from the $NC \gg SC$ problem.

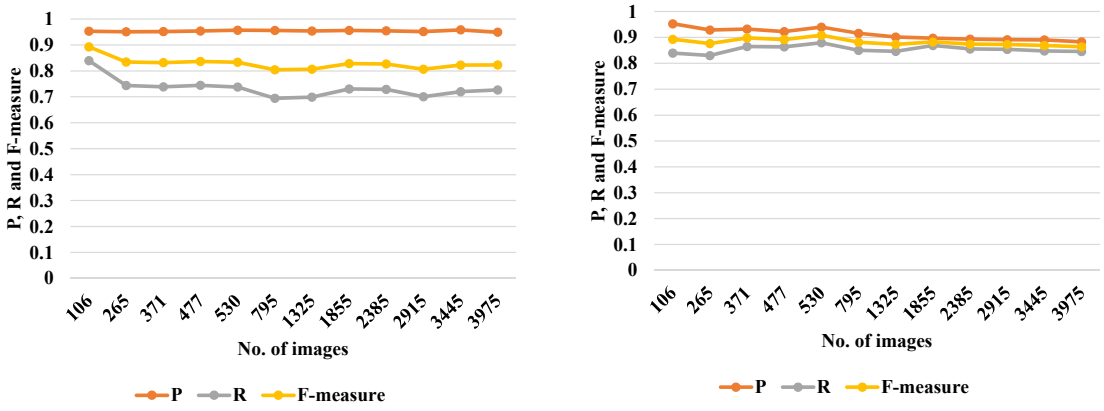


Figure 4.4: P, R and $F-measure$ for increasing SC and fixed $NC = 53$, (a) RCIC (b) RCIC-A.

The experimental values RI and ARI for different number of images are shown in Figure 4.5. The results demonstrate that the RI is very high and remains constant in the case of RCIC, while in the case of the RCIC-A algorithm, the RI is very high but slightly decreases when the number of images gets much larger. The ARI is also good for both the techniques. The ARI obtained for the RCIC algorithm fluctuates a bit when the number of images changes, which can be due to the variation in the nature of images. The ARI in the case of the RCIC-A algorithm first increases with an increase in SC and the number of images and then decreases. The increase in ARI is due to the attraction process and as the SC increases, the number of images in the major clusters before attraction increases, which results in average reference fingerprints of good quality. Which attracts more and more fingerprints in the minor clusters of the same camera towards its cluster. However, the decrease in ARI at larger SC and larger number of images may be due to the presence of higher variations in the images and also due to false attraction.

It can be observed that as SC increases, the size of the dataset increases and

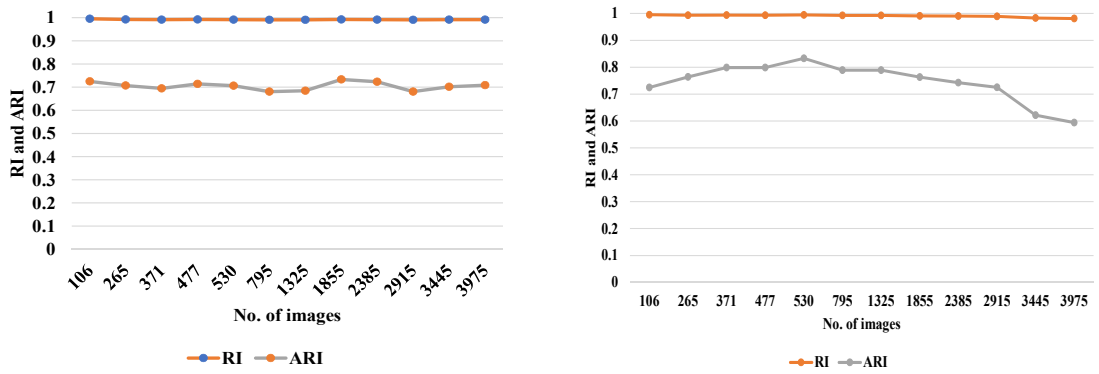


Figure 4.5: *RI* and *ARI* for increasing *SC* and fixed $NC = 53$, (a) RCIC (b) RCIC-A.

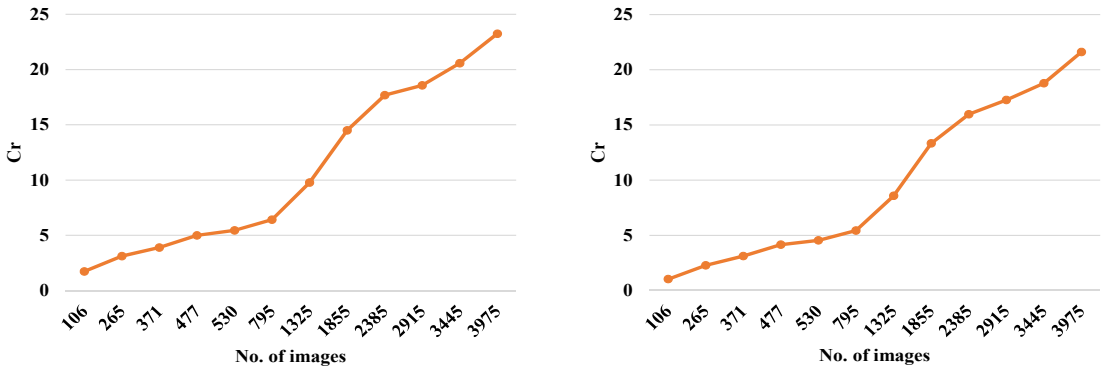


Figure 4.6: *Cr* for increasing *SC* and fixed $NC = 53$, (a) RCIC (b) RCIC-A.

the evaluation measures decrease, especially in the case of the RCIC-A algorithm. However, this decrease in evaluation measures is compensated by the significant reduction in computational complexity. The results in Figure 4.6, show that as the size of the dataset increases, the complexity of both versions of the proposed algorithm decreases with respect to the complexity i.e., $n(n-1)/2$ and hence the complexity reduction *Cr* factor increases. The decrease in computational complexity proves the strength of the RCIC and RCIC-A algorithm to cluster large datasets. Thus, the above techniques are considered suitable for large scale clustering.

4.3.4 $NC \gg SC$ analysis

The $NC \gg SC$ problem has been discussed in the previous sections of small scale and large scale clustering. But in Section 4.3.2, the datasets used are small and have a limited number of cameras while in Section 4.3.3, the number of images is large enough but again has limited NC. To use the Dresden dataset [33, 34] for $NC \gg SC$ problem we have only 53 cameras in it. In this section, we are evaluating the robustness of RCIC and RCIC-A algorithms under the $NC \gg SC$ scenario on $D5$ dataset, which has a significantly larger NC.

Firstly, the experiments are performed for different SC and fixed $NC = 295$. The SC equal to 2, 3, 5, 10, 15 and 20 is used for experimentation. The P, R, $F-measure$, RI, ARI and Cr are computed. The results obtained at different values of SC are shown in Figure 4.7, Figure 4.8 and Figure 4.9.

The results in Figure 4.7 show that the P of the RCIC algorithm is significantly good and constant for different SC, while the P of RCIC-A is also quite good and decreases a little when the SC increases. The change in the P may be due to the possible false attraction. Looking at the values of R and $F-measure$, both the parameters are higher for smaller values of SC and decrease with an increase in SC. It can also be observed that as expected the RCIC-A results in higher R and $F-measure$ than that of RCIC, due to the attraction process. The better values of P, R and $F-measure$ for smaller values of SC, prove that RCIC and RCIC-A can withstand the $NC \gg SC$ problem.

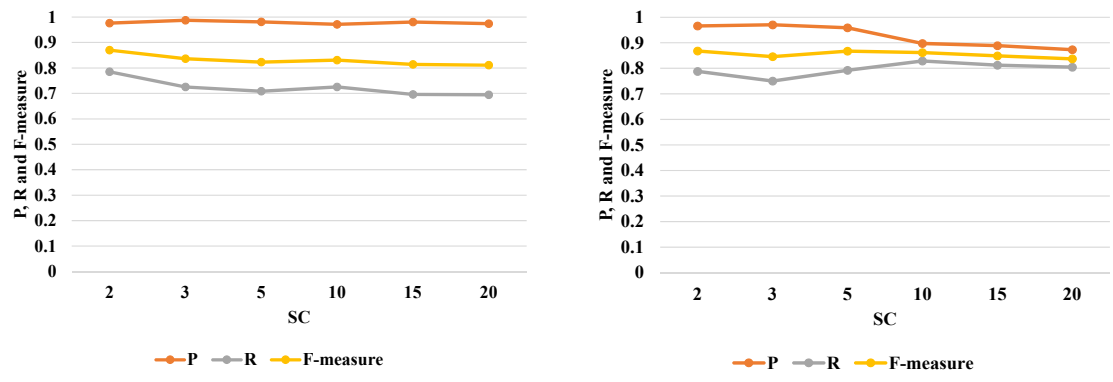
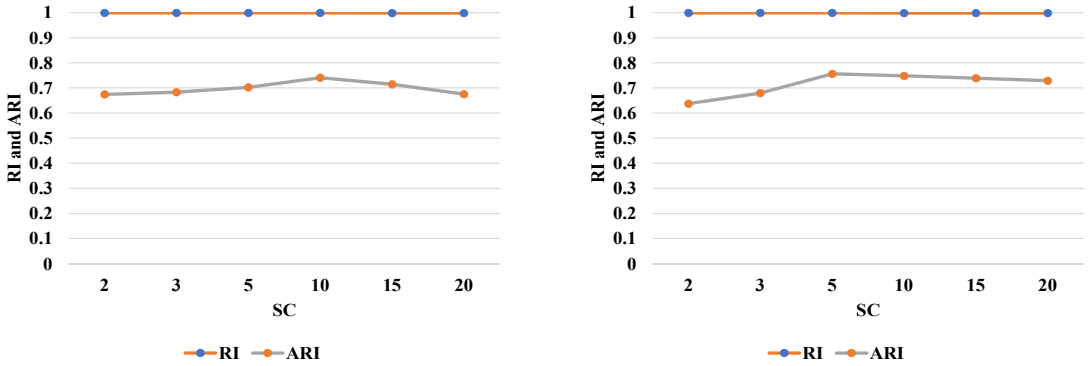


Figure 4.7: P, R and $F-measure$ for increasing SC and fixed $NC = 295$, (a) RCIC (b) RCIC-A.

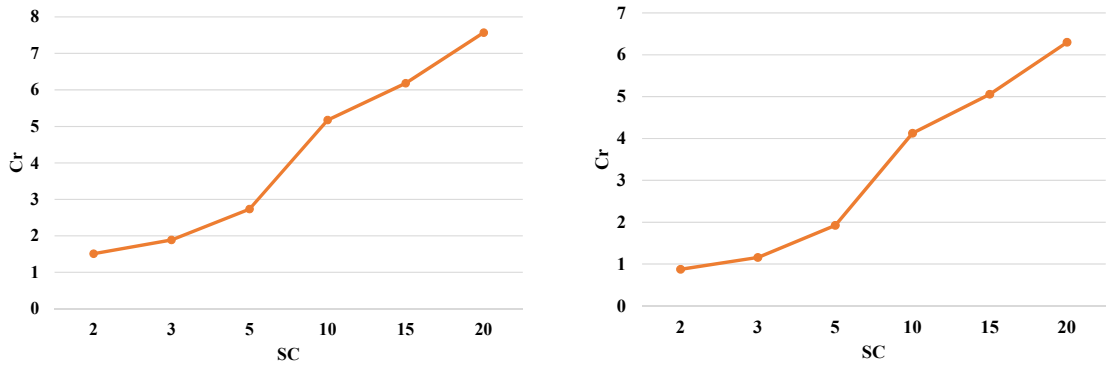
The resulting values of RI and ARI for different values of SC using both RCIC and RCIC-A techniques are shown in Figure 4.8. The results show that both methods have high values of RI and ARI. The results further confirm that RI does not vary with SC. However, the ARI increases with an increase of SC.



(a) (b)

Figure 4.8: RI and ARI for increasing SC and fixed $NC = 295$, (a) RCIC (b) RCIC-A.

Now, analyzing the complexity, as expected the Cr increases with an increase in SC for both RCIC and RCIC-A. The increase in Cr with SC is due to an increase in the size of the dataset n. This is due to the fact that the increase in complexity in our algorithms is subquadratic in n, so when we compare it with $n(n-1)/2$, this reduction is larger as n grows. As observed in Figure 4.8, the Cr for RCIC and RCIC-A increases with an increase in the size of the dataset. The Cr of RCIC-A is less than that of the RCIC algorithm, due to the additional computational cost of the attraction in RCIC-A. Hence, the increase in Cr with the increase in the size of the dataset proves that the RCIC and RCIC-A techniques are suitable for large scale clustering.



(a) (b)

Figure 4.9: Cr for increasing SC and fixed $NC = 295$, (a) RCIC (b) RCIC-A.

After the experimental results obtained for different SC at fixed $NC = 295$, the $NC \gg SC$ is further analyzed by varying NC and keeping SC fixed at 20.

The RCIC and RCIC-A are applied to different datasets with NC equal to 50, 75, 100, 125, 150, 175, 200, 250 and 295 at $SC = 20$. The constructed clusters are evaluated using P, R, $F-measure$, RI and ARI. While the computational complexity relative to the upper bound on complexity is measured in terms of Cr. The computed P, R and $F-measure$ are shown in Figure 4.10. The results in Figure 4.10a show that as the NC gets larger and larger relative to SC, the P, R and $F-measure$ remain stable. The little fluctuation in the values of R is due to some images that are not attracted by the correct cluster. The P obtained for RCIC-A is less than that of RCIC, as given in Figure 4.10b. However, the R and $F-measure$ of RCIC-A is relatively higher than that of RCIC. The decrease in P is due to the attraction of some wrong singleton clusters. Similarly, the attraction is also responsible for an increase in R and consequently, in $F-measure$. The results show that RCIC and RCIC-A have significantly good P, R and $F-measure$ even when the $NC \gg SC$ problem gets worst and worst. Hence, it can be said that both RCIC and RCIC-A are robust to $NC \gg SC$.

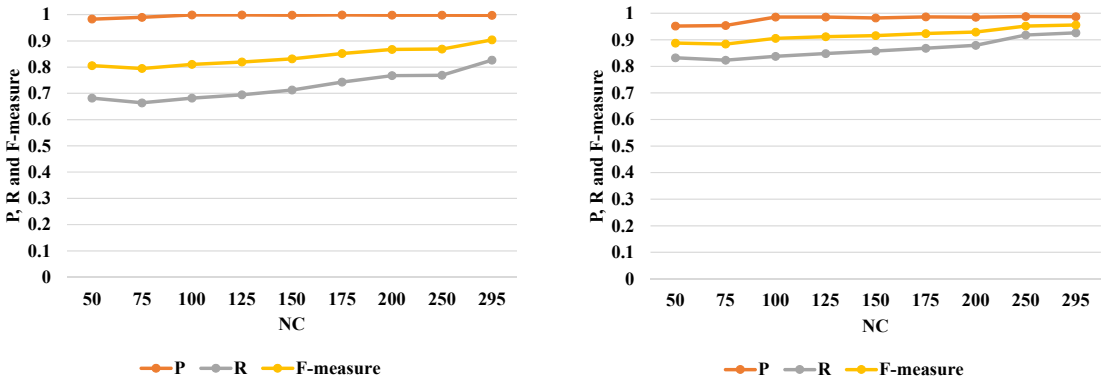
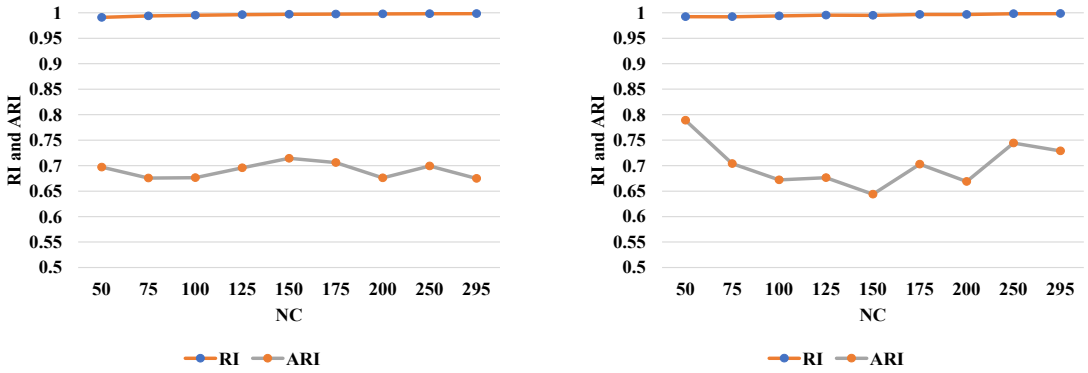


Figure 4.10: P, R and $F-measure$ for increasing NC and fixed $SC = 20$, (a) RCIC (b) RCIC-A.

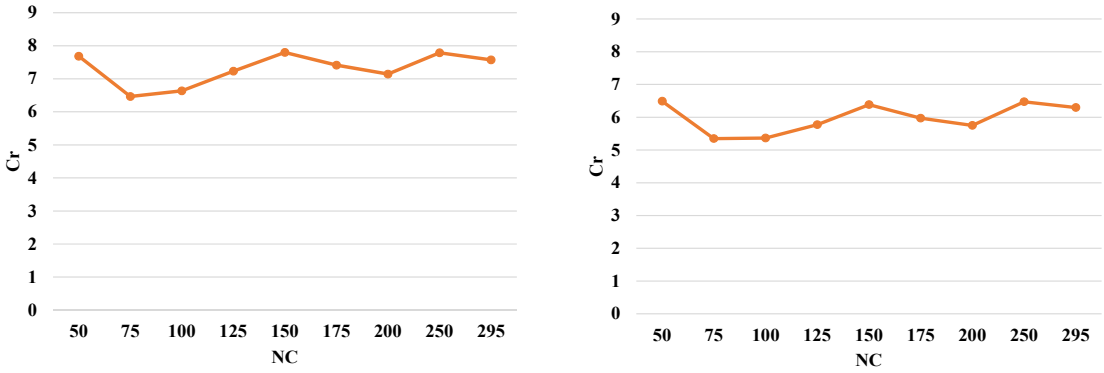
The RCIC and RCIC-A are also examined using RI and ARI. The results obtained for different NC and fixed $SC = 20$ are displayed in Figure 4.11a-b. While evaluating the RCIC and RCIC-A using the RI and ARI, the results show that RI remains stable with increasing NC and fixed $SC = 20$ but, the ARI decreases with increase in the NC and then start improving when NC get much larger at fixed $SC = 20$. The higher values of RI and ARI for higher NC with respect to SC prove that the RCIC and RCIC-A do not suffer from $NC \gg SC$ problem. Hence, the quality of clusters constructed with RCIC and RCIC-A are not affected by the increase in NC with respect to SC and RCIC and RCIC-A are considered robust to $NC \gg SC$ problem.



(a) (b)

Figure 4.11: RI and ARI for increasing NC and fixed $SC = 20$, (a) RCIC (b) RCIC-A.

Along with the quality of the constructed clusters, the computational cost of clustering is also analyzed for different NC and fixed $SC = 20$. The results also show that as the NC increases relative to SC, the Cr fluctuates, and no increasing or decreasing trend is followed by Cr. The results are shown in Figure 4.12a-b.



(a) (b)

Figure 4.12: Cr for increasing NC and fixed $NC = 20$, (a) RCIC (b) RCIC-A.

The increase in NC increases the size of the dataset. Which tries to increase the Cr, as discussed in the large scale analysis of RCIC and RCIC-A. At the time, the increase in NC makes the dataset harder and as the hardness increases the Cr decreases, as shown in the small scale analysis. Both the effect equalizes each other, and the Cr does not show an increasing or decreasing trend and fluctuates with increasing NC and fixed $SC = 20$. The fluctuations in Cr are probably due to the large variance in the quality of the images. The results in Figure 4.12b show

that the Cr of RCIC-A is less than that of RCIC. Because the attraction stage adds extra computational cost.

4.4 Performance of FICFO and FICFO-A

Similar to the RCIC and RCIC-A algorithms, the FICFO and FICFO-A techniques are also examined on small and large datasets. The robustness of the techniques against the $NC \gg SC$ problem is also evaluated experimentally. The effectiveness of the $\Re I$ is also evaluated. The values of PFA is set to 10^{-6} for computing the Th and is followed in all subsequent experiments.

The evaluation metrics of P, R, $F-measure$, RI and ARI are used to judge the clusters quality while Cr is computed to measure the computational complexity relative to the upper bound $n(n-1)/2$. The evaluation metrics and Cr are defined and discussed in Section 4.2.

4.4.1 Analysis of $\Re I$ and NCC ρ

The ranking index $\Re I$ computed form the average gray level, saturation, and texture level of an image, should represent how close an estimated fingerprint is to the ideal noiseless fingerprint. To study the contribution, we design an experiment that shows the effectiveness of $\Re I$ to fingerprint based image clustering and its contribution to the reduction in computational complexity. For this test, natural and flat images of the Dresden dataset [33, 34] with different camera brands and models are considered. The reference fingerprints for each camera are obtained by averaging the estimated fingerprints from flat field images of the cameras. The averaged reference fingerprints are standardized to zero mean unit variance. Then, the camera fingerprint is estimated fro each image among the natural images. The camera fingerprints are standardized, and $\Re I$ is computed for each of these camera fingerprints using the respective images Eq. 3.15.

To find the relationship between the $\Re I$ of a fingerprint and the NCC ρ, the NCC ρ between each fingerprint of every camera model and average reference fingerprint of the corresponding camera model is calculated. The values $\Re I$ of images are plotted against the resulting values of ρ, as displayed in Figure 4.13.

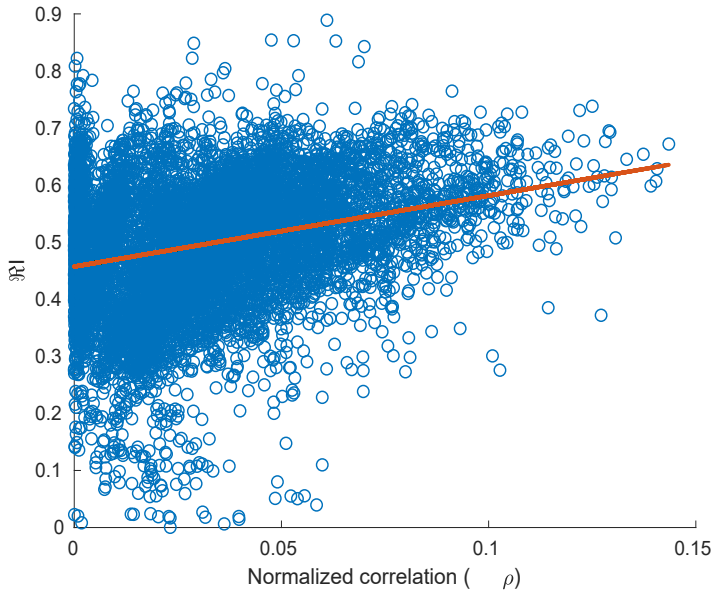


Figure 4.13: Analysis of $\Re I$ vs NCC ρ.

The scatter plot shows the relationship between the $\Re I$ and NCC ρ, to have a much clear view of the trend, a regression line is fitted to the data using a linear regression model. From experiments, it has been observed that the slope of the linear model varies with α, β and γ. In the following, we will use $\Re I$ obtained when $\alpha = 2$, $\beta = 0.5$ and $\gamma = 2$.

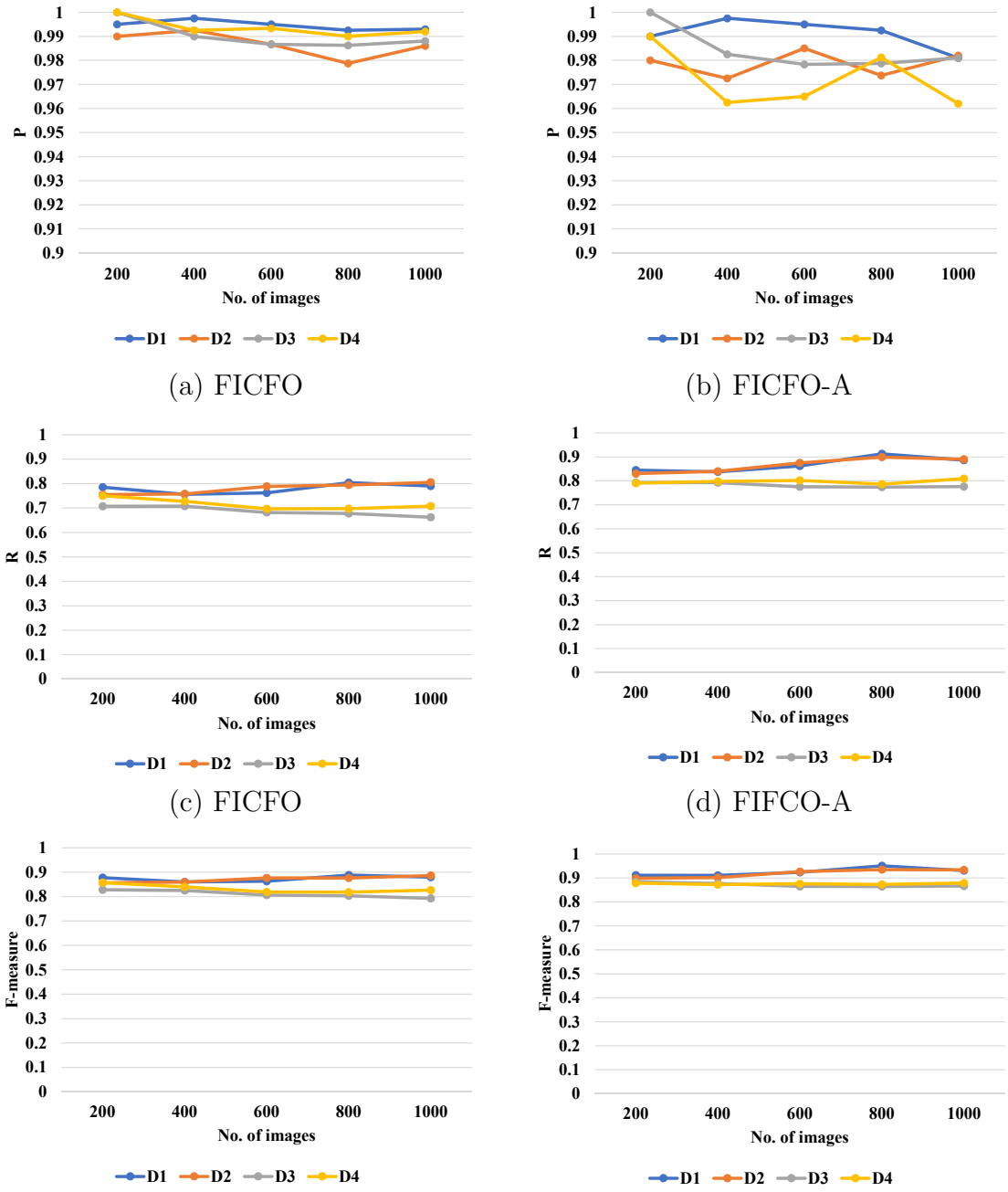
The values of α, β and γ are obtained experimentally using flat field and natural images of different camera models available in Dresden dataset. The average camera fingerprints estimated flat field and natural images. Then average reference fingerprints are obtained from the camera fingerprints of flat field images. The $\Re I$ for each natural image is computed and camera fingerprints of natural images from each camera are sorted in the descending order of $\Re I$. The NCC between the average reference fingerprints of a camera and the fingerprints estimated from natural images the same camera is computed. The $\Re I$ are plotted against the NCC computed. The process is repeated for different values of α, β and γ. It has been found from experiments that $\alpha = 2$, $\beta = 0.5$ and $\gamma = 2$ best describes the relation between the NCC and $\Re I$.

The results show that the fingerprints with a high value of $\Re I$ result in high NCC ρ and $\Re I$ can be used to predict the correlation between a fingerprint and the corresponding reference fingerprint. From this, it can be concluded that fingerprints with a high value of $\Re I$ are the best choice to be used as reference fingerprints during clustering. In all subsequent experiments, the fingerprints are sorted using $\Re I$ with the same values of parameters. The FICFO and FICFO-A techniques are evaluated for different types of small datasets, large scale clustering, and $NC \gg SC$ problem.

4.4.2 Small Scale Clustering

The performance of FICFO and FICFO-A on small datasets is examined using four different types of small datasets, i.e., $D1$, $D2$, $D3$ and $D4$. The details of the datasets are given in Section 4.1. The experiments are performed on 200, 400, 600, 800, and 1000 images of each dataset. The number of images is selected in such a way that only the number of images per camera SC changes and the number of cameras NC do not change for each dataset. This experimental setup helps to evaluate the techniques not only on different types and different size of small datasets but also to study the $NC \gg SC$ problem for different types of small datasets.

The FICFO and FICFO-A, when applied to the above mentioned datasets of different types, generate high quality clusters. The evaluation measures computed during the various experiments on small datasets are shown in Figure 4.14. Figure 4.14a demonstrates the P obtained in the case of FICFO, remains constant for the different number of images of each small dataset. On the other hand, the P obtained for FICFO-A, shown in Figure 4.14b, shows a little inconsistency and reduction with the increasing number of images. The reason for the fluctuations is some false merging of small clusters during the attraction process, and the minor modification may be due to the presence of some bad images. As mentioned above, the size of the dataset is increased by increasing the SC with fixed NC. It can be seen that the P is not affected by varying SC and result in significantly high P even for very small SC. Hence, the FICFO and FICFO-A are robust to the $NC \gg SC$ problem on datasets of different configurations.



(a) FICFO (b) FICFO-A

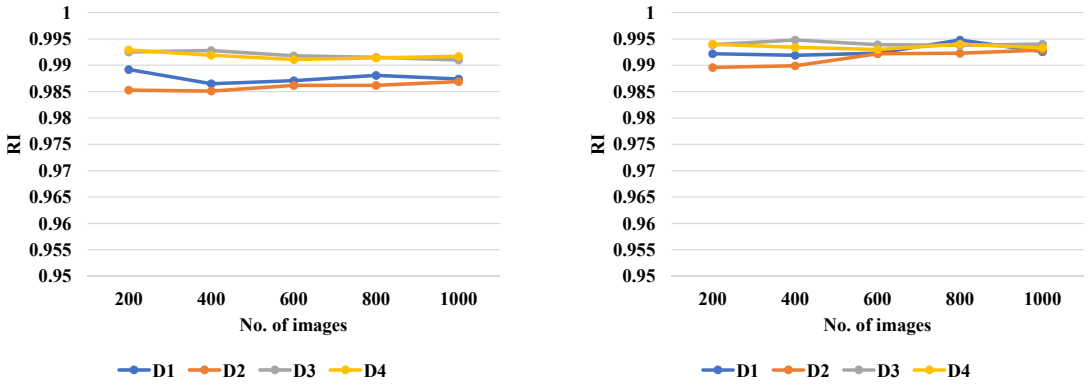
(c) FICFO (d) FIFCO-A

(e) FICFO (f) FICFO-A

Figure 4.14: P, R and $F-measure$ of the FICFO and FICFO-A algorithm on small datasets.

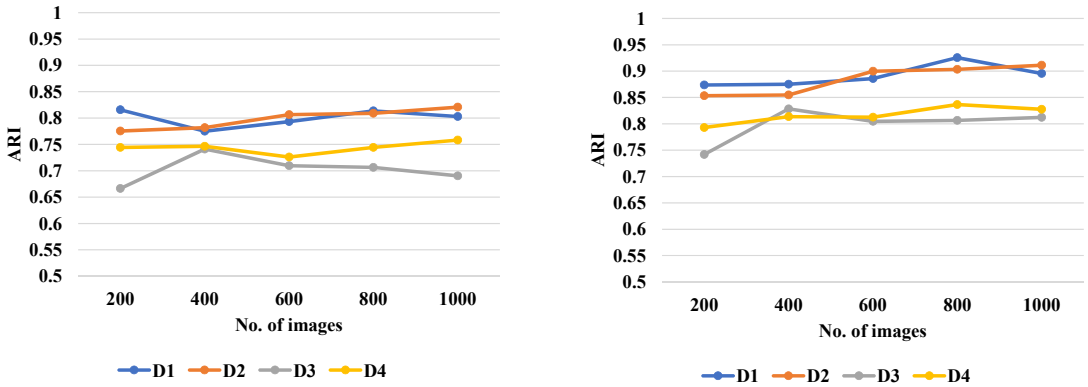
Along with P, the R and $F-measure$ are also crucial for the analysis of clustering algorithms. The R and $F-measure$ obtained from clustering the different number of images of different types of small datasets using FICFO and FICFO-A

are displayed in Figure 4.14c-f. The results reveal that both FICFO and FICFO-A perform well on all small datasets, but the performance is better on easy datasets relatively to hard datasets, as shown in Figures 4.14c-f. It can be observed that due to the attraction process, FIFCO-A results in a higher R and $F-measure$ than the FICFO algorithm. The FIFCO-A attracts the singleton clusters and merges them with the closest cluster, based on the similarity criteria. Hence, the FICFO-A results in higher values of R and $F-measure$.



(a) FICFO

(b) FICFO-A



(c) FICFO

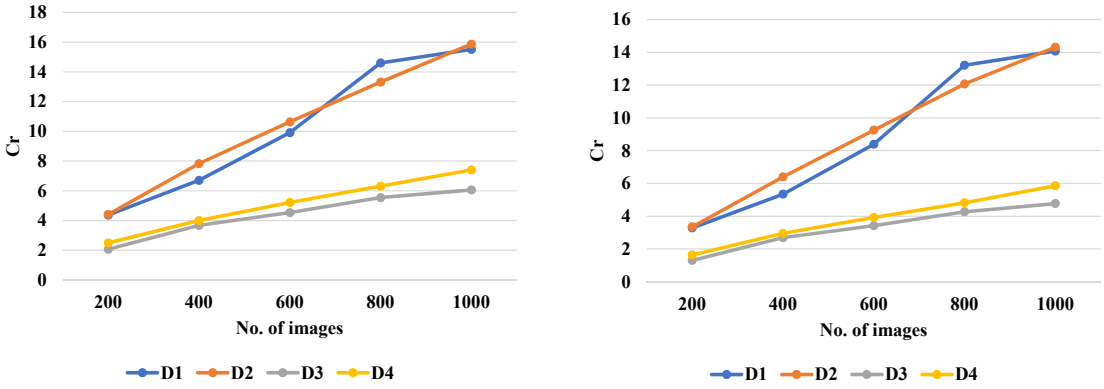
(d) FICFO-A

Figure 4.15: RI and ARI of the FICFO and FICFO-A algorithm on small datasets.

The number of constructed clusters has a great influence on R and $F-measure$ and the generation of few singleton clusters can affect these evaluation metrics. Therefore, the FICFO and FICFO-A are also evaluated using another couple of related evaluation metrics of RI and ARI. The two parameters are independent of the number of clusters and depend on the quality of clusters only. The experimentally obtained results of RI and ARI are shown in 4.15. The results show that both FICFO and FICFO-A result in high and constant RI for the different number of

images of different types of small datasets. While the value of ARI is also high for both the algorithms. The algorithms result in higher values of RI on hard datasets with respect to easy datasets. The ARI values are higher on the easy dataset as compared to hard datasets for both FICFO and FICFO-A.

Along with the quality of clusters, the computation cost of clustering is also a critical factor in analyzing a clustering algorithm. The high computational cost means the need for more resources and time. Hence, the lesser the computational cost faster the algorithm and lower the time needed. Time is gold, and therefore, it is highly desired to make the clustering process faster and with the least possible computational cost. The complexity reduction of Cr is a parameter used in this work, to describe the computational cost. The Cr of FICFO and FICFO-A is computed while clustering the different number of images of the various small datasets, and the results obtained are shown in Figure 4.16. The experimental results show that the Cr of both the FICFO and FICFO-A increases as the number of images increases, and hence the computational cost decreases. The results also show that the Cr of FICFO-A is less than the Cr the FICFO, due to the additional computational cost of the attraction process. The Cr is higher for easy than hard dataset and on asymmetric than symmetric datasets for both FICFO and FICFO-A.



(a) FICFO (b) FICFO-A

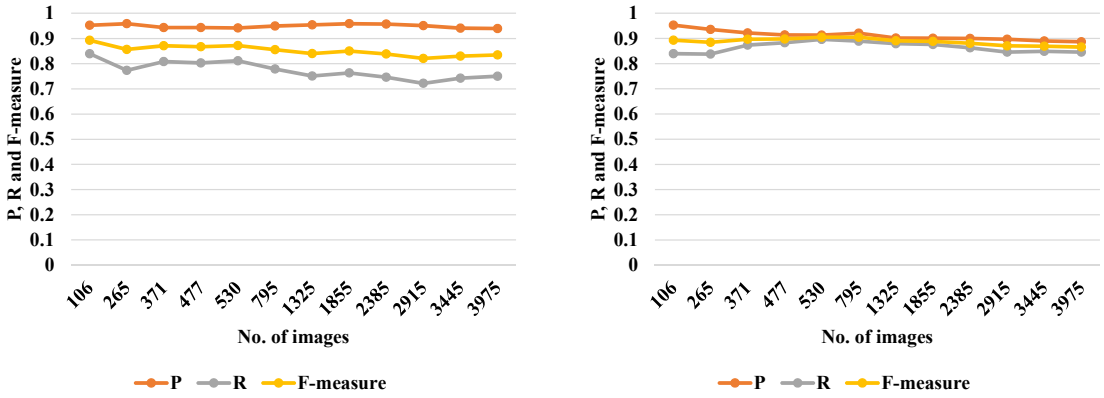
Figure 4.16: Cr of the FICFO and FICFO-A algorithm on small datasets.

4.4.3 Medium and Large Scale Clustering

The analysis of the FICFO and FICFO-A using small datasets is presented in the previous section. But, it is also important to evaluate the performance of the techniques on medium and large datasets. A different number of images are selected from Dresden [33, 34]. The subsets with the different numbers of images are constituted by varying Sc and keeping the NC fixed, i.e., $NC = 53$. Therefore each subset has $NC \times SC$ images. The SC is set to 2, 5, 7, 9, 10, 15, 25, 35,

45, 55, 65 and 75 to check the performance of FICFO and FICFO on different size of datasets from small to large and also to evaluate the robustness under the $NC \gg SC$ scenario.

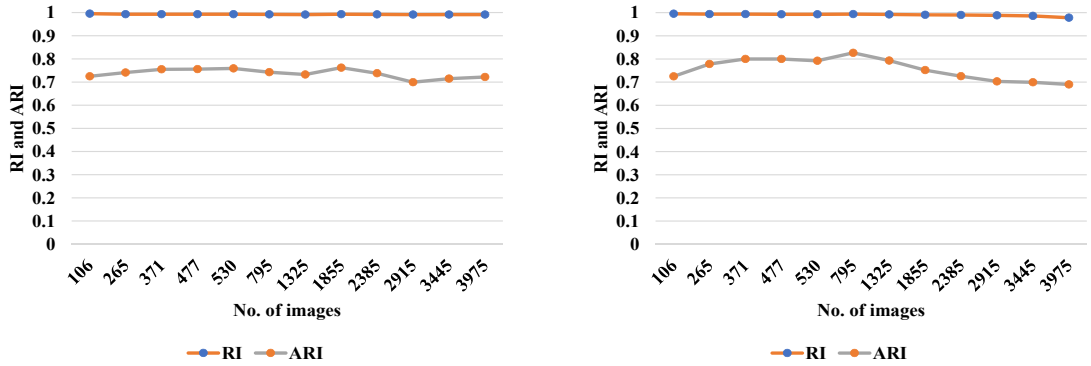
The evaluation metrics i.e., P, R, $F-measure$, RI and ARI as well as the complexity reduction Cr are computed in each experiment. The results obtained in terms of P, R, $F-measure$ are shown in Figure 4.17. The results in Figure 4.17a, show that the FICFO performs well for datasets of different sizes. The evaluation metrics P, R, and $F-measure$ show a constant and stable trend of FICFO with the increasing SC and the size of the dataset. The experimental results obtained using FICFO-A are shown in Figure. 4.17b. The displayed results show that R remains stable with an increase in the number of images and SC, however, P and $F-measure$, decrease with the increase in the size of the dataset. The decrease in P can be due to the attraction of some wrong clusters, and consequently, the $F-measure$ also decreases. The better response of FICFO and FICFO-A on a larger number of images supports the suitability of both techniques for large scale clustering, at the same time, remarkable performance on the datasets with smaller SC prove their robustness against $NC \gg SC$ problem.



(a) (b)

Figure 4.17: P, R and $F-measure$ for increasing SC and fixed $NC = 53$, (a) FICFO (b) FICFO-A.

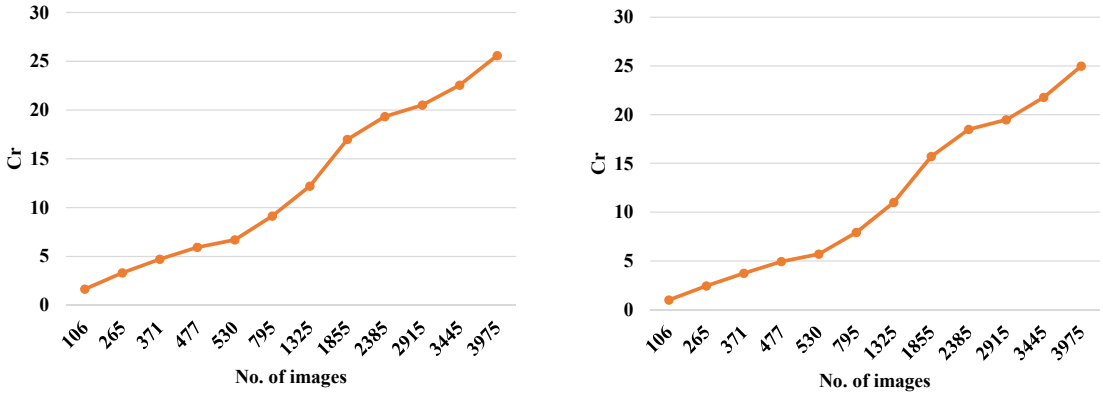
The experimental results obtained in terms of RI and ARI for the different numbers of images are shown in Figure 4.18. The results in Figure 4.18a show that RI and ARI computed for FIFCO are high and remain stable at different values of SC and the number of images. The results displayed in Figure 4.18b, reveal that RI for FICFO-A is also high, but slightly decrease with the increasing number of images and SC. While, the ARI for FICFO-A initially increases with an increase in the number of images to be clustered and then decreases. This shows that some wrong cluster, singleton clusters, are attracted at the attraction stage in FICFO-A, which result in a decrease in RI and ARI.



(a) (b)

Figure 4.18: *RI* and *ARI* for increasing *SC* and fixed $NC = 53$, (a) FICFO (b) FICFO-A.

Figure 4.19 shows the *Cr* of both FICFO and FICFO-A followed an increasing trend with the increase in the size of the dataset. The growing trend of *Cr* proves that the computational cost of FICFO and FICFO-A decreases with respect to the upper bound on complexity i.e., $n(n-1)/2$ on the same number of images n. The results prove the suitability of FICFO and FICFO-A for large scale clustering. Hence it is claimed that FICFO and FICFO-A can cluster large datasets faster and can solve the $NC \gg SC$ problem efficiently.



(a) (b)

Figure 4.19: *Cr* for increasing *SC* and fixed $NC = 53$, (a) FICFO (b) FICFO-A.

4.4.4 $NC \gg SC$ analysis

The analysis of FICFO and FICFO-A algorithm on small and large datasets is done in the previous two sections. Along with the small and large scale clustering

analysis, the techniques are also analyzed for the $NC \gg SC$ problem in Section 4.4.2 and Section 4.4.3, respectively. To check the performance of the FICFO and FICFO-A in scenarios, the $NC \gg SC$ problem is more evident; the algorithms are evaluated using $D5$ dataset. The experiments are performed on images with SC equal to 2, 3, 5, 10, 15 and 20 and while keeping the $NC = 295$ fixed.

The results shown in Figure 4.20a show that the FICFO results in high P, R and $F-measure$ and the evaluation measures remain constant for different SC at fixed $NC = 295$. While the P of FICFO-A is significantly high at smaller SC and decreases with an increase in SC. The decrease in the P may be due to some false merging of minor clusters during the attraction process. But, due to the attraction process, the recall increases with an increase in the SC. Due to the attraction, the FICFO-A results in a higher R and $F-measure$ than that of FICFO. The better values of P, R and $F-measure$ for smaller values of SC, prove that FICFO and FICFO-A can withstand the $NC \gg SC$ problem.

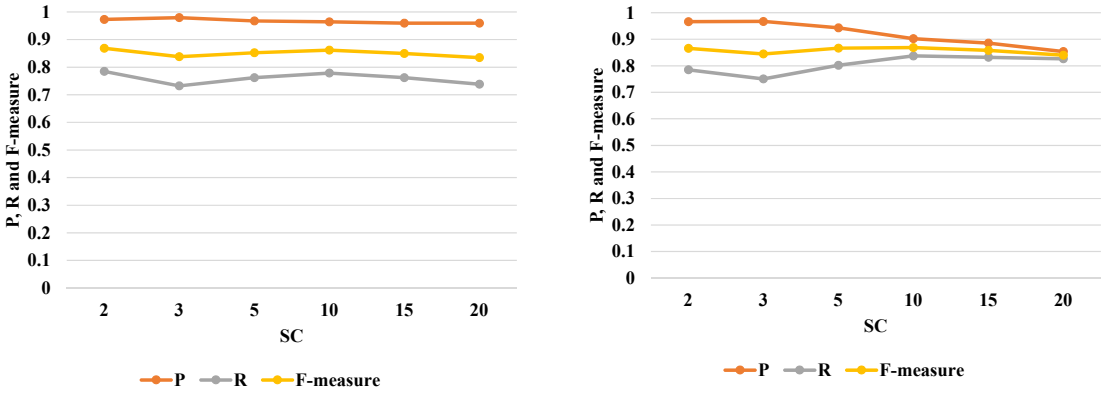
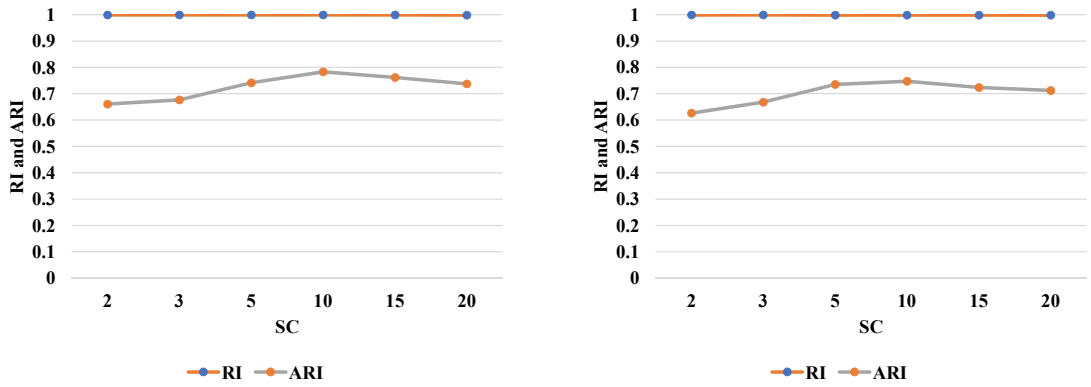


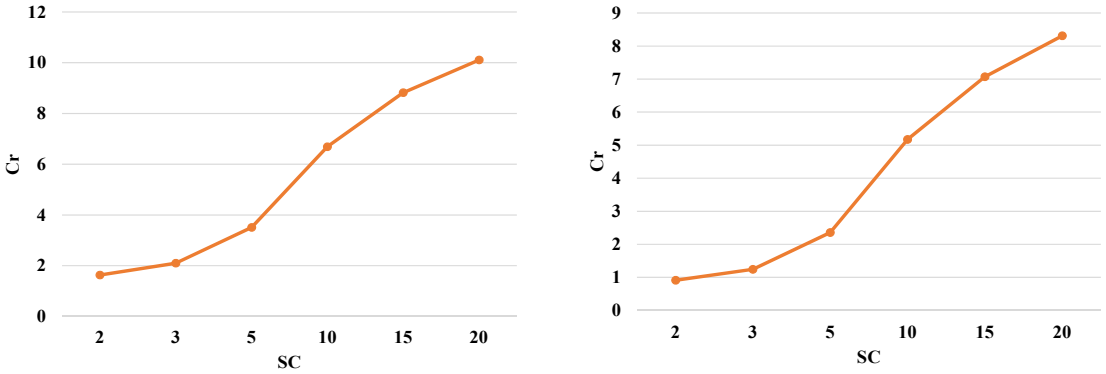
Figure 4.20: P, R and $F-measure$ for increasing SC and fixed $NC = 295$, (a) FICFO (b) FICFO-A.

The P, R, and $F-measure$ depend on the number of constructed clusters and fluctuate with small changes in the number of clusters. Therefore, RI, ARI are computed for each experiment for both the techniques to get clear observations. The resulting values of RI and ARI for different values of SC are shown in Figure 4.21. The results show that both FICFO and FICFO-A results in a higher and constant RI, while the ARI increases with an increase in SC. The results obtained in terms of RI and ARI prove that FICFO and FICFO-A are robust to the $NC \gg SC$ problem.



(a) (b)

Figure 4.21: RI and ARI for increasing SC and fixed $NC = 295$, (a) FICFO (b) FICFO-A

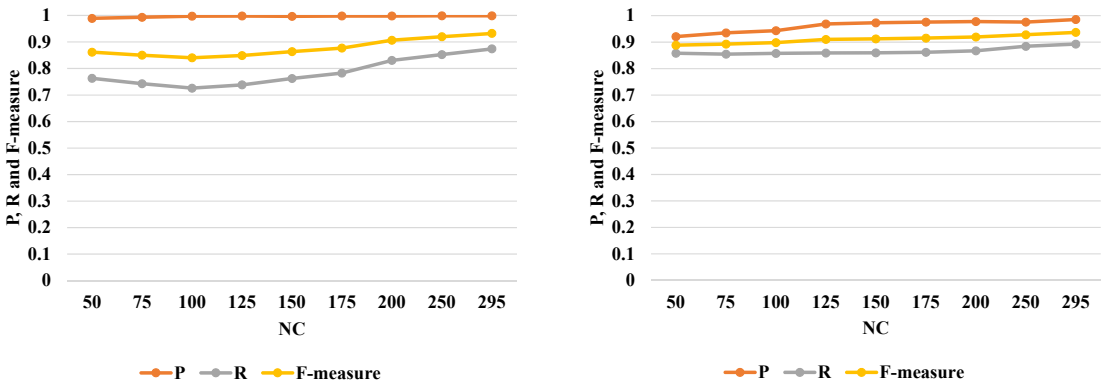


(a) (b)

Figure 4.22: Cr for increasing SC and fixed $NC = 295$, (a) FICFO (b) FICFO-A.

Now, analyzing the Cr, the results in Figure 4.22, show that the Cr of both FICFO and FICFO-A increases with an increase in SC. This is because with the rise in SC at fixed $NC = 295$, the number of images and size of dataset increases and the Cr increases with increasing size of the dataset. This trend of Cr has been observed in the clustering of small and large datasets. However, the attraction performed in FICFO-A has some computation cost, which results in a reduction of Cr for FIFCO-A as compared to FICFO. The whole experimentation and results prove that the computational cost of FICFO and FICFO-A relative to the reference computational complexity i.e., $n(n-1)/2$, decreases with the increase in the size of the dataset and hence prove the suitability of FICFO and FICFO-A for large scale clustering.

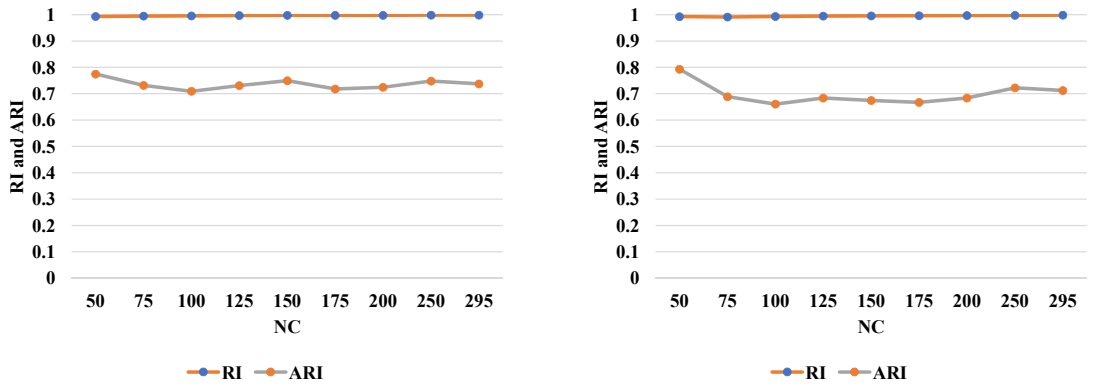
Now, the FICFO and FICFO-A are examined on datasets with different NC i.e., 50, 75, 100, 125, 150, 175, 200, 250 and 295 and fixed $SC = 20$. The constructed clusters are evaluated using P, R, $F-measure$, RI and ARI. While the computational complexity relative to the upper bound on complexity is measured in terms of Cr. The computed P, R and $F-measure$ are shown in Figure 4.23. The results in Figure 4.23a show that as the NC gets larger and larger relative to SC, the P, R and $F-measure$ of FICFO algorithm remain stable. The minor fluctuation in the values of R is due to the addition of some bad images. The results in Figure 4.23b show that the P obtained for FICFO-A is less than that of RCIC. However, the R and $F-measure$ of FICFO-A is relatively higher than that of FICFO. The decrease in P is due to the false attraction of some singleton. Similarly, the attraction is also responsible for an increase in R and consequently, in $F-measure$. The results show that FICFO and FICFO-A give higher P, R and $F-measure$ even when the $NC \gg SC$ problem gets worst and worst. Hence, it can be said that both FICFO and FICFO-A are robust to $NC \gg SC$.



(a) (b)

Figure 4.23: P, R and $F-measure$ for increasing NC and fixed $SC = 20$, (a) FICFO (b) FICFO-A.

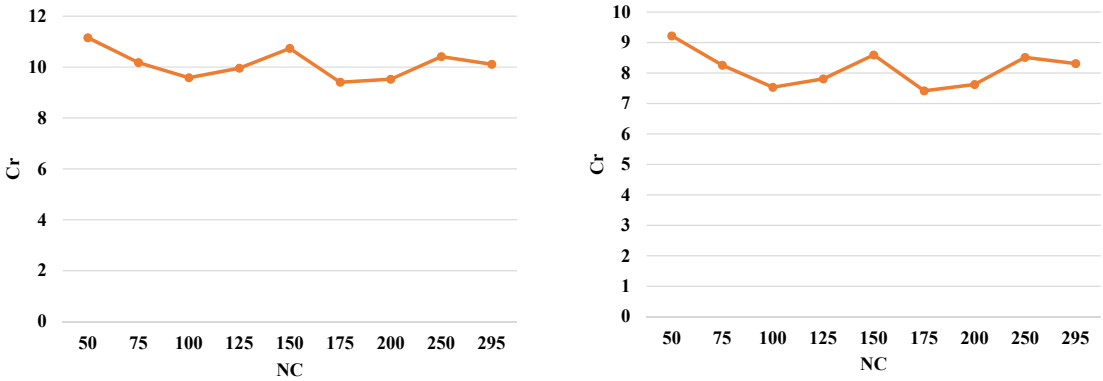
The FICFO and FICFO-A are also examined using RI and ARI, and the results obtained for different NC and fixed $SC = 20$ are displayed in Figure 4.24a-b. While evaluating the FICFO and FICFO-A using the RI and ARI, the results show that RI remains stable with increasing NC and fixed $SC = 20$. While the ARI decreases with an increase in NC, initially, and then increases at larger NC. The higher values of RI and ARI for higher NC concerning SC prove that the FICFO and FICFO-A do not suffer from $NC \gg SC$ problem. Hence, the quality of clusters constructed with FICFO and FICFO-A are not affected by the increase in NC with respect to SC, and FICFO and FICFO-A are considered robust to $NC \gg SC$ problem.



(a) (b)

Figure 4.24: RI and ARI for increasing NC and fixed $SC = 20$, (a) FICFO (b) FICFO-A.

The quality of clusters and computational complexity are important to analyze a clustering algorithm. Therefore, the FICFO and FICFO-A are evaluated based on the computational cost using various NC and fixed $SC = 20$. The results also show that as the NC increases relative to SC, the Cr fluctuates, and no increasing or decreasing trend is followed by Cr. The results are shown in Figure 4.25a-b.



(a) (b)

Figure 4.25: Cr for increasing NC and fixed $SC = 20$, (a) FICFO (b) FICFO-A.

The increase in NC increases the size of the dataset. The increase in the dimension of the dataset tries to raise the Cr, as shown in the large scale analysis of FICFO and FICFO-A. At the same time, the increase in NC makes the dataset harder and as the hardness increases, the Cr decreases, as shown in the small scale analysis. Both the effects equalize each other, and the Cr does not show an increasing or decreasing trend and fluctuates with increasing NC and fixed

$SC = 20$. The fluctuation in Cr is probably due to the addition of different quality images in the datasets with an increase of NC. The Cr improves when good images are added and decreases when bad images are added to the dataset. The results in Figure 4.12b show that the Cr of FICFO-A is less than that of FICFO. Because the attraction stage adds extra computational cost.

The FICFO and FICFO-A algorithm result in high quality clusters. The performance of FICFO and FICFO-A on small, medium and large datasets is comparable to RCIC and RCIC-A, respectively, in terms of evaluation metrics. But, Cr of FICFO and FICFO-A higher than the RCIC and RCIC-A algorithms, respectively. The reduction in the computational complexity is due to the fingerprints ordering based on $\Re I$ and the use of best fingerprints as a reference fingerprints among the unclustered ones.

4.5 Performance of CIC

The CIC algorithm is initially analyzed to find the best values of probability of false alarm for the relaxed and hard threshold. After finding the best values of PFA_r and PFA_h, the values of the respective relaxed and hard threshold are used to examine the algorithm on different types of small datasets, in the large scale clustering scenario and in the presence of the $NC \gg SC$ problem. The PFA analysis is done on $D0$ dataset, while the Dresden image database [33, 34] and sub datasets i.e., $D1$, $D2$, $D3$, $D4$ and $D5$ are used for the other experiments. The details of these datasets are given in Section 4.1.

4.5.1 PFA Analysis

The CIC algorithm first constructs a small number of large non-overlapping clusters using a relaxed threshold; then, the initial clusters are further clustered using a hard threshold. Both of these thresholds depend on the PFA values: for a relaxed threshold we set larger PFA whereas, for a hard threshold, a smaller PFA is used.

To find appropriate values of PFA_r and PFA_h, the CIC algorithm is applied to cluster dataset $D0$, as explained in Section 4.1. The camera fingerprints are estimated from the images and center cropped to a size of 1023×1023. Then the camera fingerprints are standardized to zero mean and unit variance. The $\Re I$ for each camera fingerprint is computed using the corresponding resized image. The $\Re I$ obtained at $\alpha = 2$, $\beta = 0.5$ and $\gamma = 2$. The fingerprints are arranged in the descending order of $\Re I$. Then the CIC algorithm is applied to cluster the images using the estimated camera fingerprints. Clustering using the CIC algorithm is repeated using the different values of PFA_r and PFA_h to set the relaxed and hard threshold, respectively. Complexity reduction Cr and the evaluation metrics P, R

and $F - meaure$ are computed for each experiment.

P and R can not be considered alone for the evaluation of the best PFA value: a very small PFA value will yield a very high P, at the cost of lowering the R; conversely, smaller PFA values will tend to increase the R, but will decrease the P. F which is the combination of both P and R, is the best suitable metric for the evaluation of the CIC algorithm for different PFAs. The F metric is used along with P and Cr to evaluate the CIC algorithm for different values of PFA. The experimental results are shown here in Figure. 4.26.

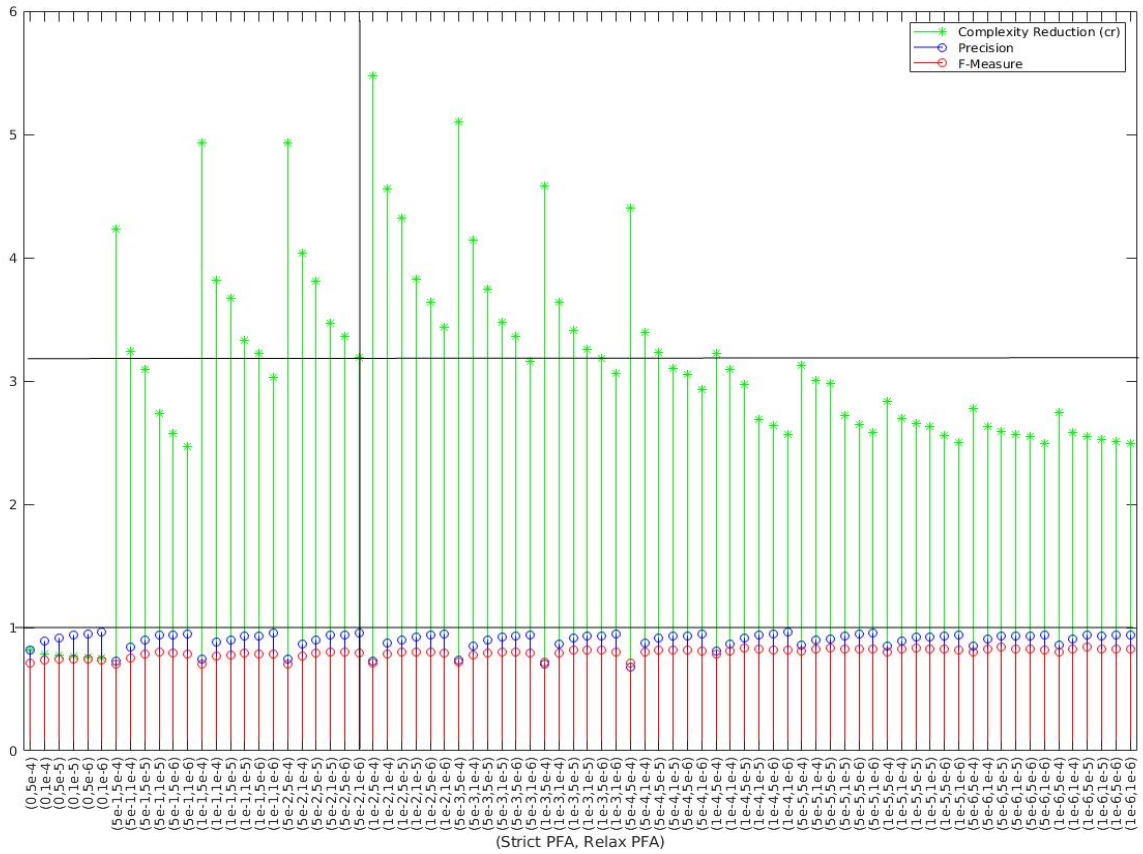


Figure 4.26: Analysis of $F - Measure$ and Cr for different values of relaxed and hard threshold PFAs.

The experimental results show that for a fixed value of PFA_r, the complexity reduction Cr decreases with an increase in PFA_h. However, precision P and $F - measure$ increases. While increasing PFA_r, the complexity reduction of Cr first increases and then decreases. A highest complexity reduction Cr is observed for PFA_r of 1×10^{-2} and PFA_h of 5×10^{-4}. But, the precision P and F-measure F, the two important factors describing the quality of the clusters, are very low for these

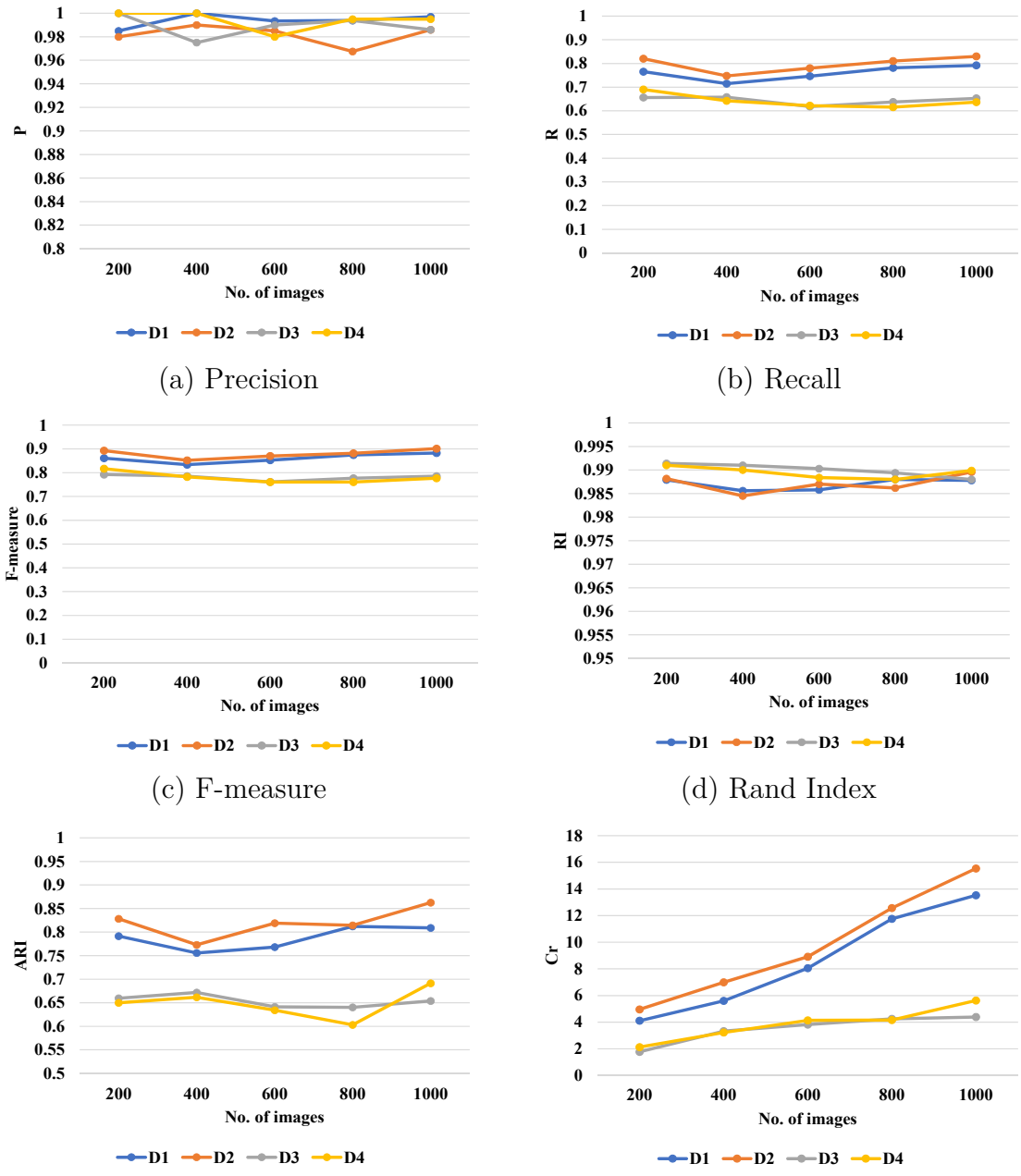
values of PFA_r and PFA_h. Closely observing the experimental results for different pairs of values of PFA_r and PFA_h, it can be seen that best value of precision P, along significantly good values of $F-measure$ and complexity reduction Cr can be obtained with PFA_r and PFA_h equal to 5×10^{-2} and 1×10^{-6}, respectively. Therefore, these values are selected and used in all subsequent experiments.

4.5.2 Small Scale Clustering

The small datasets i.e., $D1$, $D2$, $D3$ and $D4$, are used to examine the performance of the CIC algorithm on small scale clusters. The experiments are performed by selecting 200, 400, 600, 800, and 1000 images. The change in SC results in a change in the number of images in a dataset. While the NC remains and equal to the total number of cameras in the dataset. This setup is used to evaluate the CIC algorithm not only on different types but also on datasets of different sizes with different SC. The results are used to check the robustness of the CIC algorithm against the $NC \gg SC$ problem for different types of small datasets.

Following the experimental setup, the CIC algorithm is applied to cluster each set of images from each dataset, and the evaluation measure P, R, $F-measure$, RI and ARI are computed in the experiment. Along with cluster quality measure the computational complexity, relative to the upper bound on complexity i.e., $n(n-1)/2$, is also measured in terms of Cr. The results obtained in these experiments are shown in Figure 4.27. The experimental results show that the CIC algorithm gives a high P for each set of images, as shown in Figure 4.27a. The results showed that P slightly fluctuates with an increasing number of images and SC, which can be due to the addition of some bad images, which are wrongly attracted and clustered in the attraction stage. The results confirm that the P of the CIC algorithm is not affected when $NC \gg SC$.

The Figure 4.27b, shows that CIC algorithm results in a high value of R. The R obtained on easy datasets is higher than hard datasets, which show that the CIC algorithm performs better on the easy datasets while compared to hard datasets. The result obtained in terms of $F-measure$ also shows the same trend as R and is shown in Figure 4.27c.



(a) Precision

(b) Recall

(c) F-measure

(d) Rand Index

(e) Adjusted Rand Index

(f) Complexity Reduction

Figure 4.27: Small scale analysis of CIC algorithm.

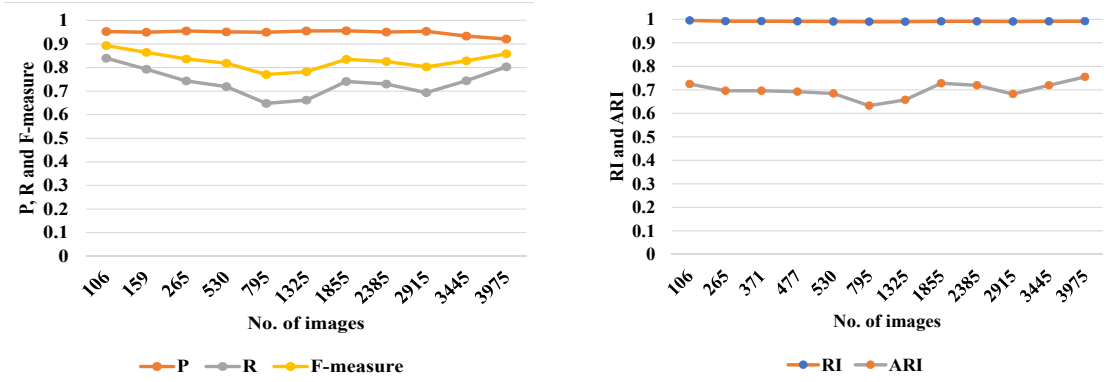
The *RI* and *ARI* as given in Figure 4.27d and Figure 4.27e, respectively, show that the CIC algorithm results in very high values of *RI*. Similar to the previous algorithms, the CIC algorithm performs better on hard datasets with respect to easy datasets, in terms of *RI*. The *ARI* obtained for the different number of images is high. The *ARI* computed for the CIC algorithm has higher values of

easy datasets as compared to the hard datasets.

The resulting Cr on a different set of images shows that the Cr increases with the increase in the number of images as displayed in Figure 4.27f. Along with the increasing trend of Cr with the number of images, it can be seen that the CIC algorithm has less computational complexity on easy datasets than the hard dataset, which is apparent from the higher values of Cr on easy datasets.

4.5.3 Medium and Large Scale Clustering

The CIC algorithm is applied to different subsets of images chosen from Dresden [33, 34] using the same and fixed NC, i.e., $NC = 53$, and varying the SC. The number of images to be clustered increases with increasing SC. The experimental results are shown in Figure 4.28a, and demonstrate that the proposed technique performs well for different sizes of datasets and different SC. The results show that as the average SC changes with respect to NC, P is almost constant but R and $F - measure$, decrease and then increase with the increasing number of images and SC due to attraction of some wrong clusters. While the other two clustering quality measuring parameters i.e., RI and ARI are shown in Figure 4.28b. The results show that the RI does not change with an increase in the number of images under test. While, the ARI shows some fluctuations in its values with the change in the number of images. The fluctuations may be due to the variations in the nature of images.



(a) P, R and $F - measure$ (b) RI and ARI

Figure 4.28: Evaluation measure of CIC algorithm vs increasing SC and fixed $NC = 53$.

The higher values of the evaluation metrics on large number images make the CIC algorithm suitable for large scale clustering. Moreover, high P, R, $F - measure$, RI and ARI for small number images per camera SC show that this algorithm does not suffer from $NC \gg SC$ problem.

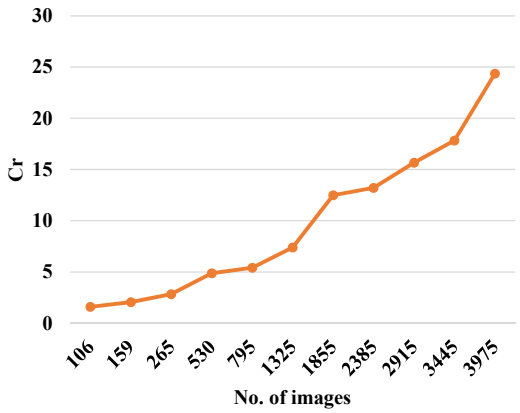


Figure 4.29: Cr of CIC algorithm vs increasing SC and fixed $NC = 53$.

Figure 4.29 shows the complexity reduction Cr obtained in the different cases. It has been observed that the complexity, with respect to the reference complexity i.e., $n(n-1)/2$, of the CIC algorithm decreases with increasing size of dataset i.e., the number of images. Hence the complexity reduction Cr factor increases. The less computational cost and higher Cr at a large number of images prove that the CIC algorithm can be a good choice for large scale clustering.

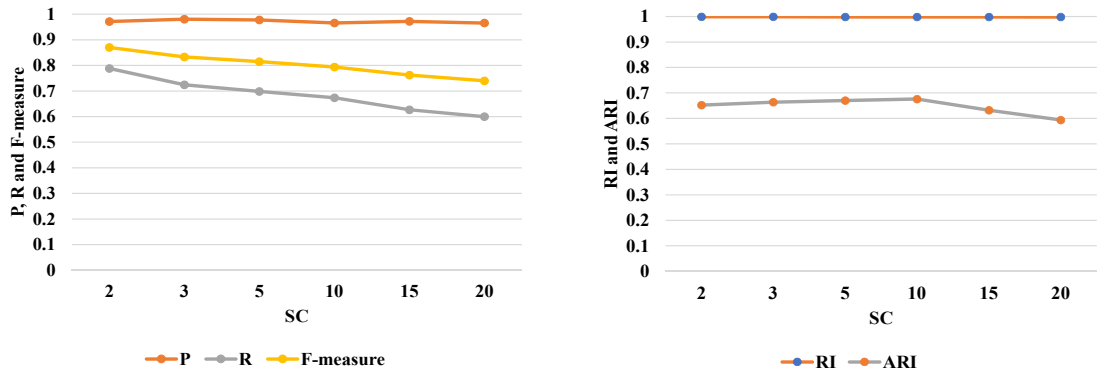
4.5.4 $NC \gg SC$ analysis

Like the RCIC, RCIC-A, FICFO, and FICFO-A algorithms, the CIC algorithm is also analyzed for the $NC \gg SC$ problem using the $D5$ dataset. The robustness of the CIC algorithm against the $NC \gg SC$ problem is also discussed in both Section 4.5.2 and Section 4.5.3 on small and large datasets respectively. However, to examine the CIC algorithm on harder scenarios for what concerns the $NC \gg SC$ problem, the $D5$ dataset with 295 different cameras is used. The camera fingerprints are estimated from each image of each camera in the dataset. The experiments are performed on images with SC equal to 2, 3, 5, 10, 15 and 20 and while keeping the $NC = 295$ fixed.

The results shown in Figure 4.30a show that the CIC algorithm results in high and constant P, while the R and $F - measure$ decreases with increase in SC at fixed $NC = 295$. The decrease in the R and $F - measure$ can be due to the presence of some bad images. The fingerprints of the bad images are not clustered properly and result in several singleton clusters. The singleton or small clusters have image/s from a single camera. That's why the P remains high and the R and $F - measure$ drop. The results show that the CIC algorithm has higher values of P, R, and $F - measure$, which prove that the CIC algorithm is robust to the $NC \gg SC$ problem.

The R and $F - measure$ are highly dependent on the number of constructed

cluster and a generation a single singleton cluster affect these metrics. Therefore, we examine the CIC algorithm using the parameters of RI and ARI. And the results obtained, shown in Figure 4.30b for different SC at fixed $NC = 295$, show that the CIC algorithm has a high and constant RI. The ARI obtained for different SC increases initially and then decreases. The decrease in the values of ARI at higher SC is due to the false attraction during the attraction process.



(a) P, R and $F - measure$ (b) RI and ARI

Figure 4.30: Evaluation measure of CIC algorithm vs increasing SC and fixed $NC = 295$.

Now, analyzing the Cr, as expected, the Cr of the CIC algorithm increases with an increase in SC. The results shown in Figure 4.31. This is because with the increase in SC at fixed $NC = 295$, the number of images and size of the dataset increases and consequently, the Cr increases with increasing size of the dataset. This trend of Cr has been observed in the clustering of small and large datasets too. The results prove that the computational cost of the CIC algorithm relative to the reference computational complexity i.e., $n(n-1)/2$, decreases with the increase in the size of the dataset and hence endorse the suitability of the CIC algorithm for large scale clustering.

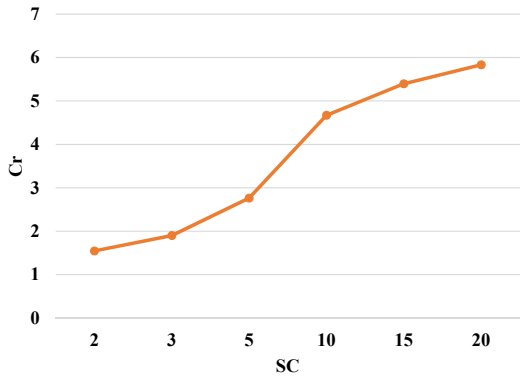
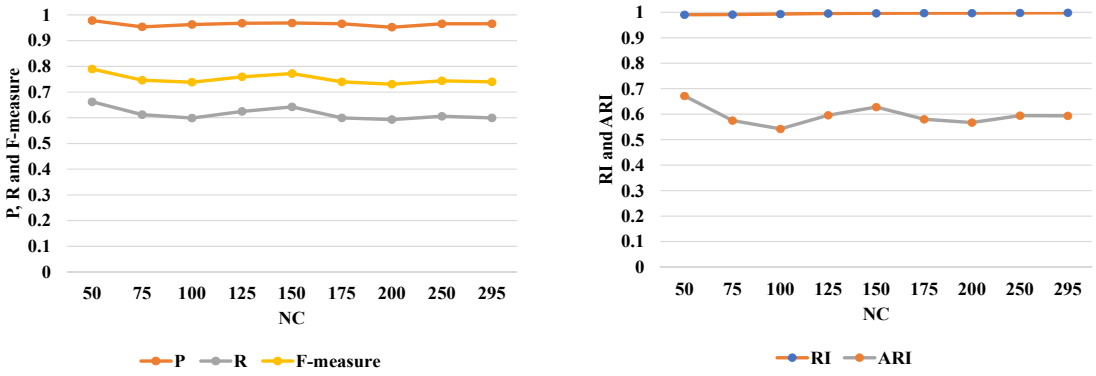


Figure 4.31: Cr of CIC algorithm vs increasing SC and fixed $NC = 295$.

To further analyze the CIC algorithm for $NC \gg SC$ problem, the NC is changed to 50, 75, 100, 125, 150, 175, 200, 250 and 295 while the SC is kept fixed at 20. The datsets with different NC are clustered using the CIC algorithm and the evaluation measure i.e., P, R, $F-measure$, RI and ARI are computed. The resulting evaluation measures are shown in Figure 4.32. While the computational complexity relative to the upper bound on complexity is measured in terms of Cr. The results obtained for Cr are displayed in Figure 4.33.

The results in Figure 4.32a show that the P for the CIC algorithm is very high and stable for the datasets with different NC. While R and $F-measure$ are high but, minor fluctuations occur with the change in NC. The changes in the value of R and $F-measure$ is due to the possible presence of some bad images. Similarly, the resulting RI and ARI are very high and remain stable with the change in NC, as displayed in Figure 4.32b. The results show that the CIC algorithm give higher P, R, $F-measure$, RI and ARI for $NC \gg SC$. Hence, the quality of clusters constructed with the CIC is not affected by the increase in NC with respect to SC, and the CIC algorithm is considered robust to the $NC \gg SC$ problem.



(a) P, R and $F-measure$ (b) RI and ARI

Figure 4.32: Evaluation measure of CIC algorithm vs increasing NC and fixed $SC = 20$.

Besides the analysis of the quality of the clusters constructed by the CIC algorithm, the algorithm is also analyzed based on the computational cost of clustering for different NC and fixed $SC = 20$. The results also show that as the NC increases relative to SC, the Cr fluctuates, and no increasing or decreasing trend is followed by Cr. The results are shown in Figure 4.33.

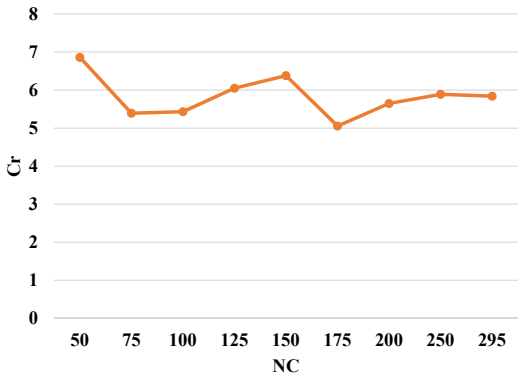


Figure 4.33: Cr of CIC algorithm vs increasing NC and fixed the $SC = 20$.

The CIC algorithm results in high quality clusters. The performance of the CIC algorithm on small, medium and large datasets is quite good. But, as compared to the RCIC, RCIC-A, FICFO and FICFO-A algorithms, the computational cost is higher. While the quality of clusters is comparable with the RCIC, RCIC-A, FICFO and FICFO-A algorithms. Like, the previously mentioned algorithms, the CIC algorithm also does not suffer from the $NC \gg SC$ problem.

4.6 Performance of CFIC

The CFIC algorithm is examined for the best way of computing the reduced fingerprints, and the best value of quantization level δ while applying the dead-zone quantization. After that, the CFIC algorithm is analyzed for small, medium and large scale clustering for different scenarios of $NC \gg SC$. The datasets presented in Section 4.1 are used in different experiments.

4.6.1 The performance of CFIC algorithm using different reduced fingerprints.

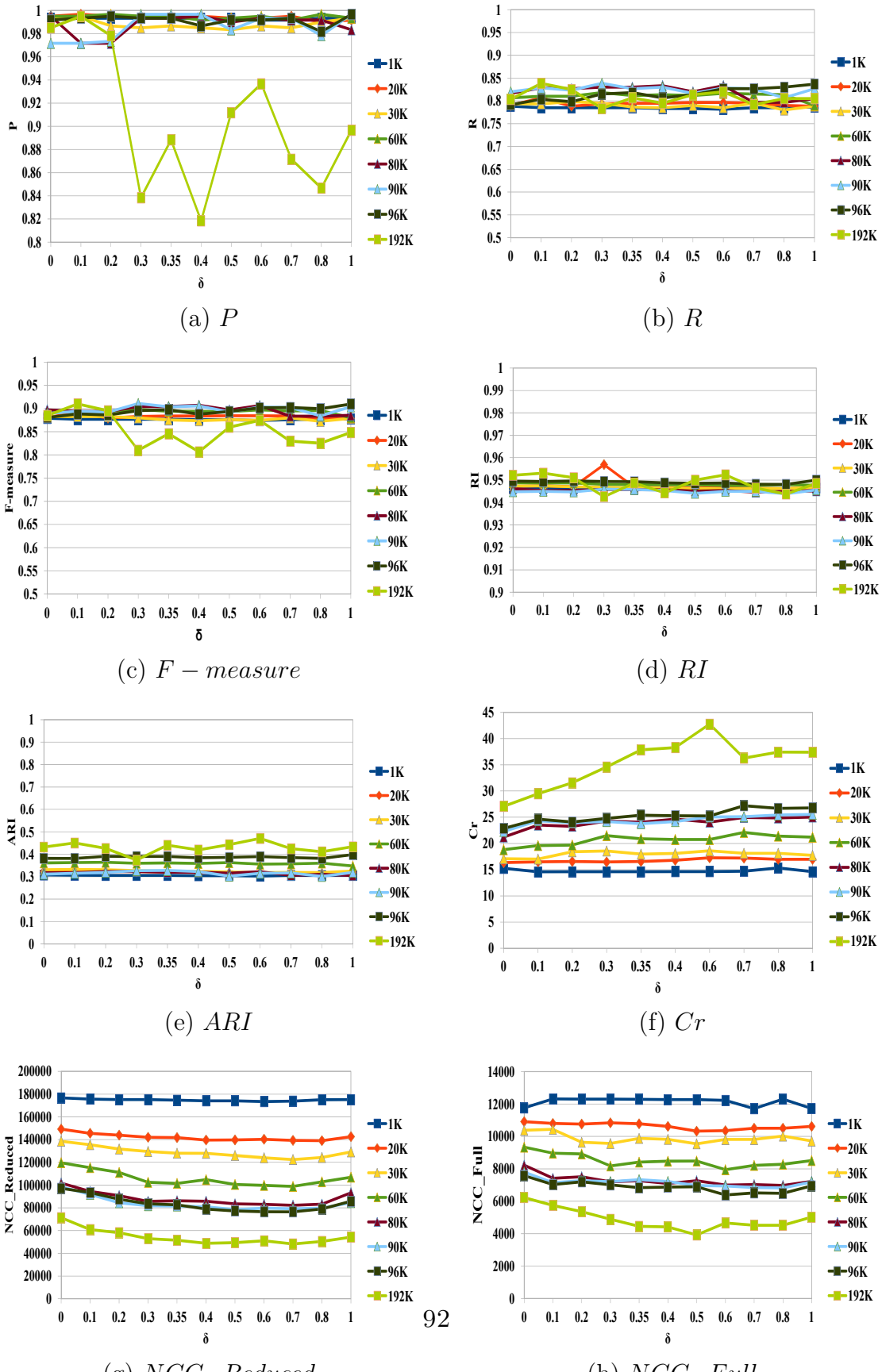
To cluster images based on camera fingerprints, the proposed algorithm uses reduced fingerprints Fr and full fingerprints F for clustering and refining the clusters, respectively. The full camera fingerprints are estimated directly from the images. However, to obtained reduced camera fingerprints, we have various possibilities. We can use decimation, random projections, and dead-zone quantization or we can apply dead-zone quantization directly to full fingerprint after decimation or without decimation. Along with these, we may use a different number of projections and values of quantization factor δ to get the reduced fingerprints. Therefore, it is important to investigate which of the above methods is the most suitable for clustering. It is also important to find the appropriate value of the number of projection, quantization factor δ, and decimation factor. For this analysis, the image dataset $D0$ is used in the following subsections.

Reduced Fingerprints with decimation, random projections, and quantization.

In a first experiment, the reduced fingerprints Fr are obtained by decimating the estimated full camera fingerprint F, computing the random projections using the Gaussian sensing matrix and quantizing the random projections using dead-zone quantization. The full camera fingerprints F, each of the size 1023×1023, are decimated by a factor of 2. The random projections of length P_r equal to 1000, 20000, 30000, 60000, 80000, 90000, 96000 and 192000 are generated using Gaussian sensing matrix of appropriate sizes for each decimated camera fingerprint. The random projections of each camera fingerprint are quantized at different levels of δ i.e., 0, 0.1, 0.2, 0.3, 0.35, 0.4, 0.5, 0.6, 0.7, 0.8 and 1, using dead-zone quantization. The fingerprints are clustered using the reduced camera fingerprints Fr and average full camera fingerprints, at initial clustering and fine clustering stages, respectively. The performance of the clustering algorithm is analyzed based on the measured evaluation metrics.

The evaluation metrics obtained for reduced fingerprints Fr of different sizes, i.e., P_r, are plotted against the different values of quantization factor δ. The results

are shown in Figure 4.34.



(a) P (b) R

(c) $F-measure$ (d) RI

(e) ARI (f) Cr

(g) $NCC\_Reduced$ (h) NCC_Full

Figure 4.34: Performance of CFIC algorithm using reduced fingerprints Fr of different size P_r at different values of δ.

The results shown in Figure 4.34a, show that the proposed algorithm results in a reasonably good values of precision almost above 0.9 for different sizes of reduced fingerprints P_r and quantization factor δ, except $P_r = 192000$, which results in a low precision at most values of qunatization factor δ. The recall computed for different parameters varies from a minimum of 0.78 to maximum 0.8387. Recall greater than 0.8 is obtained for the majority of the quantization factor δ values at Pr equal to 60,000, 80,000, 90,000 and 96,000, as shown in Figure 4.34b. Now, from the results shown in Figure 4.34c it is apparent that the $F-measure$ greater or equal to 0.9 is obtained for very few combination of P_r and δ. A maximum value of 0.91087 for $F-measure$ is obtained at $P_r = 90{,}000$ and $\delta = 0.3$. Similarly, if we look at the resulting RI and ARI, as shown in Figure 4.34d and Figure 4.34e respectively, no significant variations have been observed at different values of P_r and δ. The clustering algorithm results in RI of 0.945 and above for experiments. While, the ARI, obtained in these experiments varies between 0.28 to 0.49. The higher values of ARI are recorded for $P_r = 192K$.

Along with the evaluation of the quality of resulting clusters, it is also important to analyze how fast clustering is performed. The clustering process will be fast if fewer computations are performed. The parameter we are using, i.e., the complexity reduction Cr that depends on the number of NCC computed among the reduced fingerprints, full fingerprints and the total number of fingerprints. The results shown in Figure 4.34f, Figure 4.34g and Figure 4.34h, show that the number of NCC computed among the reduced fingerprints NCC_R in initial clustering and NCC computed among the full fingerprints NCC_F at fine clustering, decreases as the size of reduced camera fingerprints P_r decreases and as a result the Cr also increases. Similarly, the NCC_R and NCC_F decreases with an increase in δ up to $\delta = 0.5$ and then starts increasing and hence, the Cr behaves in reverse. The results shows that high complexity reduction is achieved for $P_r = 192{,}000$.

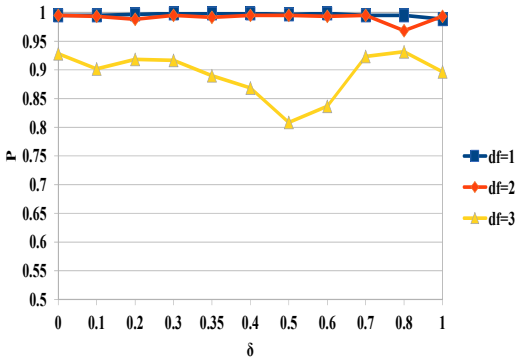
As our priority is to have good quality clusters, we give more importance to Precision, recall, and $F-measure$ when selecting the best combination of P_r and δ. By observing all the results in Figure 4.34, we reach to the conclusion that $P_r = 90{,}000$ and $\delta = 0.3$ is the best combination for estimating reduced camera fingerprints. The proposed algorithm results in highest values of 0.9967, 0.8387 and 0.91087 for Precision, recall and F-measure, respectively and a Cr of 24.78 is achieved at these values of P_r and δ.

Reduced Fingerprints with decimation and quantization.

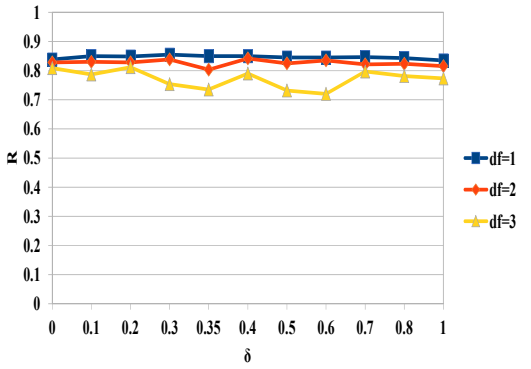
Now, we check the performance of the proposed algorithm, using the reduced fingerprints obtained by decimating the estimated full camera fingerprint and dead-zone quantization. The full camera fingerprints F, each of the size 1023×1023, are decimated by different decimated factors. The decimation factors of $df = 1$, $df = 2$ and $df = 3$ are used to reduce the size of fingerprints and result in decimated

camera fingerprints of size $\frac{|F|}{df^2}$. Here is important to mention that $df = 1$ means no decimation. Then, the decimated fingerprints F_d are quantized at different levels of sigma δ i.e., 0, 0.1, 0.2, 0.3, 0.35, 0.4, 0.5, 0.6, 0.7, 0.8 and 1, using dead-zone quantization, to get final reduced camera fingerprints. The proposed algorithm is used to cluster the fingerprints. The reduced camera fingerprints and average full camera fingerprints are used at initial clustering and fine clustering stages, respectively. The performance of the clustering algorithm is analyzed based on the measured evaluation metrics.

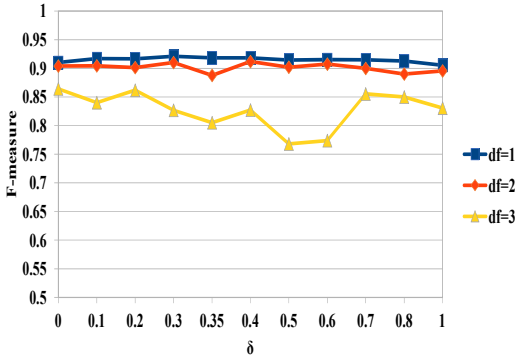
The evaluation metrics obtained for the cluster using reduced fingerprints estimated at different decimation factors, i.e., df and quantization factor δ. The results are shown in Figure 4.35.



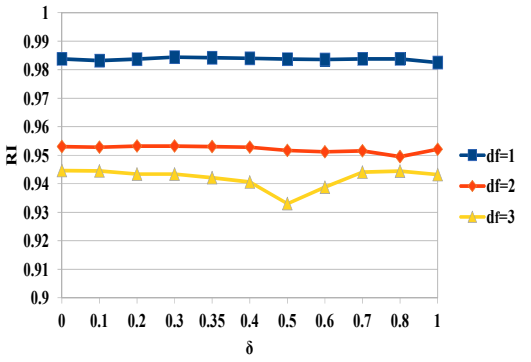
(a) P



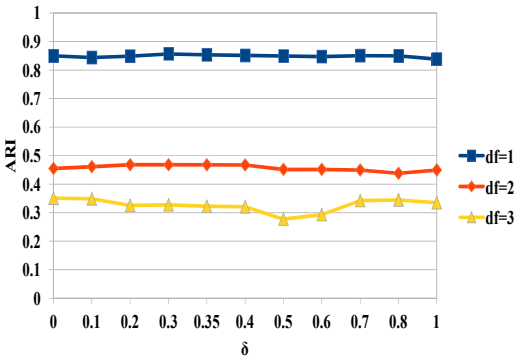
(b) R



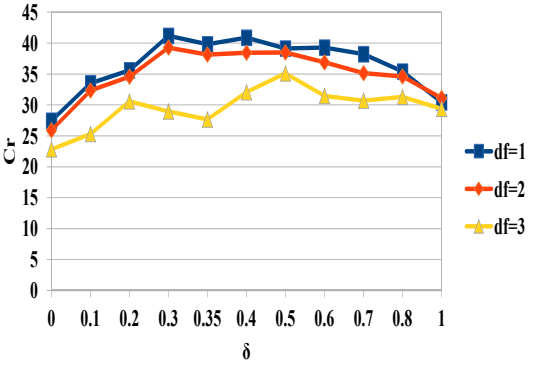
(c) $F-measure$



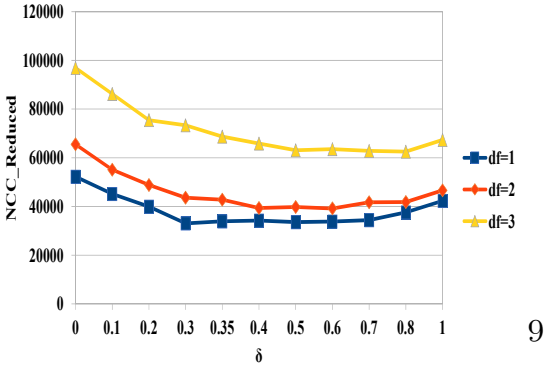
(d) RI



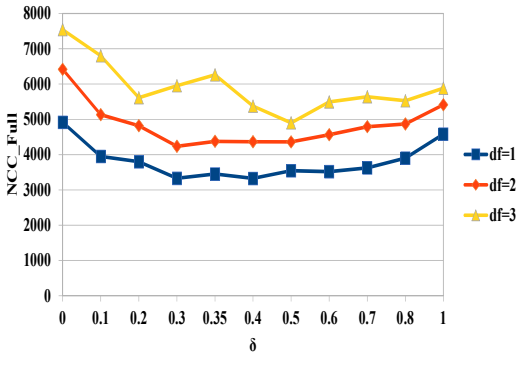
(e) ARI



(f) Cr



(g) $NCC\_Reduced$



(h) NCC_Full

Figure 4.35: Performance of CFIC algorithm using reduced fingerprint with different decimation factor d at different values of δ.

The experimental results obtained shows that the P, R and $F-measure$ decrease with increase in decimation factor, as shown in Figure 4.35a, 4.35b and 4.35c. The RI and ARI also slightly decrease with increase in decimation factor. The resultant RI and ARI are shown in Figure 4.35d and 4.35e, respectively. The results show that the proposed algorithm result in high RI and ARI at $df = 1$, means in case of no decimation.

The resulting Cr, NCC_R and NCC_F are shown in Figure 4.35f, 4.35g and 4.35h, respectively. The results show that the complexity reduction Cr initially increases as the δ increases from 0 to 0.4 and then a decrease in Cr for $df = 1$ and $df = 2$. Similarly, the NCC_R and NCC_F are lesser at the mid values of δ and larger at lower and higher values of δ. However, it has been observed that at $df = 3$ the clustering algorithm behaves differently and irregularities are observed in the values of Cr, NCC_R and NCC_F for different values of δ.

Therefore, the results in Figure 4.35 clearly shows that the reduced fingerprint obtained from full fingerprints with decimation factor $df = 1$ is a better choice to use instead of $df = 2$, $df = 3$ and higher values of decimation factor.

The Selection of Appropriate Reduced Fingerprint.

In the previous discussion, we have two ways to estimate reduced fingerprint. One way is to perform decimation, computing random projection and then apply dead-zone quantization. The direct decimation of the full fingerprint and dead-zone quantization can be used as a second method. To differentiate between the two types of reduced fingerprints, the reduced fingerprints estimated using the first method are represented by $F^r_{P_r}$ and that computed with the second method is represented by F^r_{df}. It has been observed that using the reduced fingerprints $F^r_{P_r}$, the proposed algorithm results in high values of evaluation metrics at $P_r =$ 90,000. While in terms of complexity reduction of Cr, the algorithm has the best performance at $P_r =$ 192,000. Similarly, using the reduced fingerprints F^r_{df}, the proposed algorithm results in high values of evaluation metrics and complexity reduction Cr at $df = 1$.

Now the best candidates of the two types of reduced fingerprints, i.e., $F^r_{P_r}$ and F^r_{df} are compared, to decide which types and size of reduced fingerprints result in a better quality of clusters and least computational complexity, i.e., high complexity reduction Cr.

Table 4.3: Precision P.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$Fr_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	0.972	0.992	0.985	0.995	0.995
0.1	0.972	0.993	0.995	0.995	0.993
0.2	0.973	0.995	0.978	0.997	**0.988**
0.3	0.997	0.993	0.838	**0.998**	0.995
0.35	0.997	0.993	0.888	**0.998**	0.992
0.4	0.9967	0.987	0.818	**0.998**	0.995
0.5	0.983	0.992	0.912	0.997	0.995
0.6	0.993	0.992	0.937	**0.998**	0.993
0.7	0.993	0.993	0.872	0.995	0.995
0.8	0.978	0.982	0.847	0.995	0.968
1	0.997	0.997	0.897	0.988	0.993

Table 4.4: Recall R.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	0.82	0.792	0.803	0.838	0.828
0.1	0.828	0.803	0.838	0.85	0.83
0.2	0.823	0.798	0.825	0.848	0.8283
0.3	0.827	0.815	0.783	**0.855**	0.838
0.35	0.827	0.818	0.807	0.85	0.803
0.4	0.83	0.807	0.795	0.85	0.842
0.5	0.817	0.813	0.813	0.845	0.825
0.6	0.828	0.827	0.82	0.845	0.835
0.7	0.827	0.827	0.792	0.847	0.822
0.8	0.807	0.83	0.805	0.843	0.823
1	0.827	0.837	0.805	0.835	0.815

Table 4.5: $F - measure$.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	0.889	0.880	0.8851	0.910	0.904
0.1	0.894	0.888	0.910	0.917	0.904
0.2	0.892	0.886	0.895	0.916	0.901
0.3	0.911	0.895	0.810	**0.921**	0.910
0.35	0.904	0.897	0.846	0.918	0.888
0.4	0.906	0.888	0.806	0.918	0.912
0.5	0.892	0.894	0.860	0.914	0.902
0.6	0.903	0.902	0.874	0.915	0.907
0.7	0.902	0.902	0.830	0.915	0.900
0.8	0.884	0.899	0.825	0.913	0.890
1	0.904	0.910	0.848	0.905	0.895

The results obtained for reduced fingerprint $F^r_{P_r}$ at P_r = 90,000, P_r = 96,000 and P_r = 192,000 and reduced fingerprint Fr_{df} at $df = 1$ and $df = 2$ compared with each other. The results listed in Table 4.3 shows that the F^r_{df} at $df = 1$ results in cluster with highest precision of 0.998 for δ equal to 0.3, 0.35, 0.4 and 0.6.

Table 4.6: Rand Index RI.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	0.9448	0.9495	0.9522	0.9838	0.953
0.1	0.945	0.9494	0.9531	0.9832	0.9528
0.2	0.9447	0.9496	0.9511	0.9837	0.9532
0.3	0.946	0.9494	0.9426	**0.9844**	0.9532
0.35	0.9458	0.9493	0.9487	0.9842	0.953
0.4	0.9454	0.9488	0.9443	0.984	0.9528
0.5	0.9441	0.9486	0.95	0.9837	0.9517
0.6	0.945	0.9487	0.9524	0.9836	0.9512
0.7	0.945	0.9482	0.9465	0.9838	0.9515
0.8	0.9439	0.9481	0.9438	0.9838	0.9495
1	0.9456	0.95	0.9486	0.9825	0.9521

Table 4.7: Adjusted rand index ARI.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	0.3113	0.3815	0.4323	0.8498	0.4552
0.1	0.3154	0.3813	0.4508	0.8442	0.4607
0.2	0.3192	0.3912	0.4265	0.8489	0.4681
0.3	0.327	0.3907	0.3742	**0.8565**	0.4681
0.35	0.3278	0.3904	0.4405	0.8534	0.4675
0.4	0.3226	0.3841	0.4195	0.8517	0.4673
0.5	0.3024	0.3864	0.4434	0.8494	0.4515
0.6	0.3151	0.3898	0.4707	0.8475	0.4513
0.7	0.3164	0.3852	0.4248	0.8504	0.4498
0.8	0.3012	0.3819	0.4116	0.85	0.4379
1	0.3192	0.3985	0.4337	0.8383	0.4498

Along with the precision a highest value 0.855 of recall and highest value 0.921 of $F-measure$ are achieved by F^r_{df} at $df = 1$ all for δ equal to 0.3, as shown in Table 4.4 and Table 4.5, respectively. Similarly, if we compare the different methods in terms of RI and ARI, the proposed algorithm using reduced fingerprint F^r_{df} at $df = 1$ and $\delta = 0.3$, results in highest value of RI equal to 0.9844, as listed in Table 4.6. While, in terms of ARI, a highest values of 0.8565 is obtained using $F^r_{df=1}$ at $\delta = 0.3$ as given in Table 4.7.

Table 4.8: Number of NCC on reduced fingerprints $NCC\_Reduced$.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	98477	97100	71323	52244	65506
0.1	91742	93358	60883	45186	55063
0.2	84489	87551	58117	39954	48796
0.3	81805	83428	52867	**33072**	43606
0.35	81585	82667	51434	33952	42803
0.4	80963	78810	48744	34216	39387
0.5	78386	77256	49348	33587	39781
0.6	79705	76489	51014	33813	39178
0.7	79120	76492	48212	34448	41735
0.8	79628	79058	50364	37583	41821
1	84584	85718	54174	42305	46592

Table 4.9: No. of NCC on full fingerprints NCC_Full.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	7801	7581	6226	4914	6420
0.1	7142	7028	5750	3949	5131
0.2	7278	7203	5359	3799	4817
0.3	7207	7012	4893	**3331**	4236
0.35	7336	6840	4453	3453	4378
0.4	7215	6881	4414	3326	4365
0.5	6969	6892	3919	3544	4359
0.6	6942	6388	4661	3519	4568
0.7	6852	6514	4523	3626	4790
0.8	6823	6484	4517	3899	4867
1	7189	6951	5019	4582	5412

Table 4.10: Complexity reduction Cr.

sig	$F^r_{P_r=90k}$	$F^r_{P_r=96k}$	$F^r_{P_r=192k}$	$F^r_{df=1}$	$F^r_{df=2}$
0	22.28	22.86	27.084	27.45	25.92
0.1	24.32	24.63	29.46	33.52	32.31
0.2	23.94	24.11	31.57	35.60	34.56
0.3	24.20	24.78	34.58	**41.17**	39.26
0.35	23.78	25.39	37.85	39.81	38.13
0.4	24.18	25.28	38.29	40.88	38.45
0.5	25.03	25.26	42.77	39.12	38.48
0.6	25.11	27.20	36.28	39.27	36.86
0.7	25.44	26.69	37.44	38.21	35.12
0.8	25.543	26.78	37.39	35.42	34.60
1	24.23	24.97	33.72	30.44	31.11

When comparing the performance of the proposed algorithm in terms of complexity reduction Cr, NCC_R and NCC_F, using the different reduced fingerprints, it has been observed that the proposed algorithm achieves the least number of NCC_R and NCC_F of 33072 and 3331 at $df = 1$ and $\delta = 0.3$ respectively; therefore, it performs clustering with the least computational cost and result in highest Cr at $dfD = 1$ and $\delta = 0.3$. The results of Cr for different reduced fingerprints at different values of δ are listed in Table 4.10.

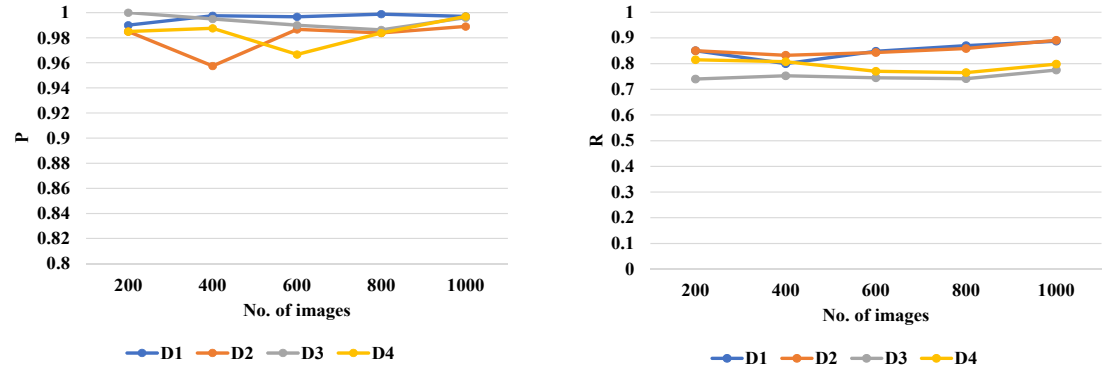
In short, the proposed algorithm generates high-quality clusters with the least computational cost while using reduced fingerprints, estimated directly from full fingerprints by applying dead-zone quantization, without any decimation i.e., $df = 1$. This is obvious because the reduced fingerprints have decimation and random

projection noises and have only quantization noise. Therefore, these fingerprint with $df = 1$ is a better approximation of the actual full fingerprints and results in high quality of clustering. From, the results it has also been observed that the reduced fingerprints estimated with quantization factor $\delta = 0.3$, is the most suitable choice for clustering images using the proposed algorithm.

From now on, with reduced fingerprints, we denote only the fingerprints estimated from full fingerprints without any decimation and using dead-zone quantization at $\delta = 0.3$. The reduced fingerprints are represented by Fr, hereafter in the text.

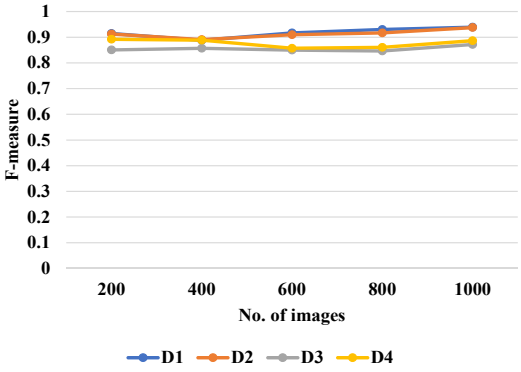
4.6.2 Small Scale Clustering

In this section, we are investigating the performance of the proposed algorithm on $D1$, $D2$, $D3$, and $D4$ datasets, varying the number of images per cluster SC while keeping the number of cameras fixed. The number of cameras in $D1$ and $D2$ datasets is 25, while in the other two datasets the number of contributing cameras NC is 50. Full camera fingerprints F and reduced camera fingerprints F^r is estimated for each image of each dataset. A number of 200, 400, 600, 800 and 1000 images are selected at SC equal to 8, 16, 24, 32 and 40 while $NC = 25$ for $D1$ and $D2$. The proposed algorithm is applied to cluster the selected images. Similarly, a number of 200, 400, 600, 800 and 1000 images are selected at SC equal to 4, 8, 12, 16 and 20 while $NC = 50$ for $D3$ and $D4$. The fingerprints are clustered using the proposed algorithm.

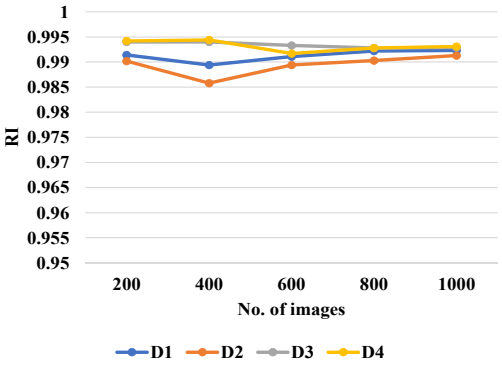


(a) P

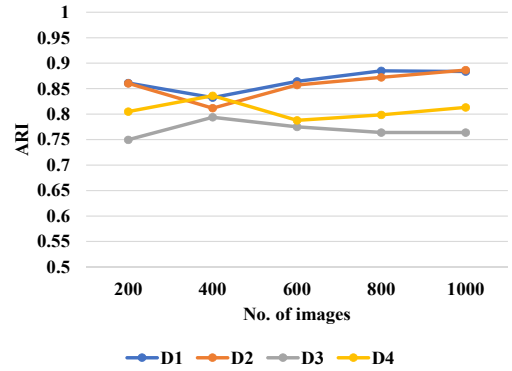
(b) R



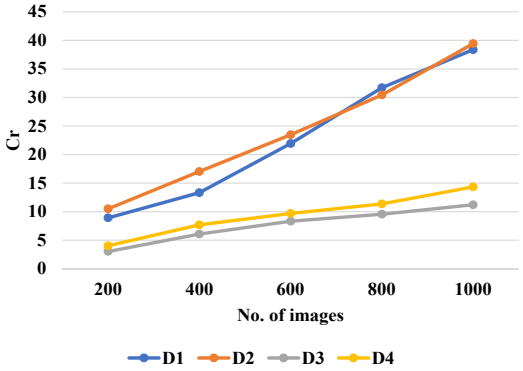
(c) $F-measure$



(d) RI



(e) ARI



(f) Cr

Figure 4.36: Performance of CFIC algorithm small datasets i.e., $D1$, $D2$, $D3$ and $D4$.

The experimental results show that the proposed algorithm when is applied to symmetric easy $D1$, asymmetric easy $D2$, symmetric hard $D3$ and asymmetric hard $D4$ with the different average number of images per camera SC, generate good quality clusters. The evaluation metrics measured are shown in Figure 4.36.

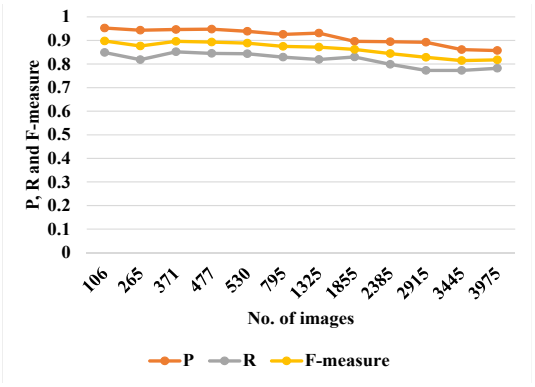
The results in Figure 4.36a show that the CFIC algorithm results in a higher P for different experiments. A small reduction is P has been observed at 400 and 600 number of images for $D2$ and $D4$ datasets, respectively. The reduction in P can be due to the false attraction of fingerprints during the fine clustering. The results obtained for R, $F-measure$, RI and ARI shown in Figure 4.36b-e, show that the CFIC results in higher values of these parameters. While in terms of complexity reduction, the proposed algorithm gives good results on $D1$ and $D2$, as compared to that of $D3$ and $D4$. The results in Figure 4.36(f) show that the proposed algorithm cluster the easy datasets faster than the hard datasets. The results also reveal that as the size of the dataset and the average number of images per camera SC, increase the complexity reduction also increases.

4.6.3 Medium and Large Scale Clustering

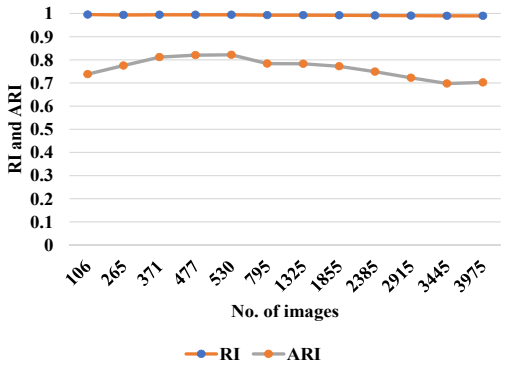
After evaluating the proposed algorithm on small datasets, it is also important to know the behavior of the algorithm on medium and large datasets. Therefore, for large scale analysis the proposed algorithm is applied to different subsets of images selected from Dresden [33, 34] using the same number of cameras, i.e., $NC = 53$, and varying the average number of images from each camera SC. Figure 4.37 shows the performance evaluation metrics, i.e., precision P, recall R, $F-measure$, RI and ARI obtained in the different cases. The results show that the proposed technique performs well for different sizes of datasets and the different number of images per camera SC. The experimental results shown in Figure. 4.37a show that the proposed algorithm results in good values of P, R and F-measure. However, the values of P, R and F-measure decrease when SC increases. This reduction in P, R, and F-measure can be due to the construction of more singleton clusters in the clustering stage and no attraction of some singleton clusters in fine clustering. Figure 4.37b shows that the values of RI are higher and remain stable while the values of ARI are also good and decrease with an increase in the number of images. The results also show that the clustering algorithm does not suffer from $NC \gg SC$ problem and results in a higher quality cluster when SC gets smaller with respect to NC.

The results shown in Figure 4.37c and 4.37d show that the complexity reduction of the proposed algorithm increases as the SC increases. It can be observed that NCC_R and NCC_F increase with an increase of SC. This increase is since the number of cameras NC is fixed and the average number of images per camera SC increase, which increases the total number of images $NC \times SC$ and overall size of the dataset. Therefore, the clustering of a large number of images will perform a large number of NCC, both NCC_R and NCC_F. However, the NCC_R and NCC_F grow slower than the $n(n-1)/2$ as n increases. Hence, the overall complexity reduction of Cr, which compares the total computational cost with the reference complexity $n(n-1)/2$, decreases. As shown in Figure 4.37c, the

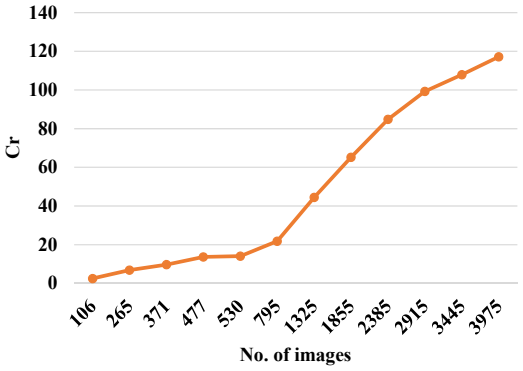
complexity reduction of Cr increases with an increase in the size of the dataset. Therefore, the clustering algorithm can be much suitable for clustering large scale datasets.



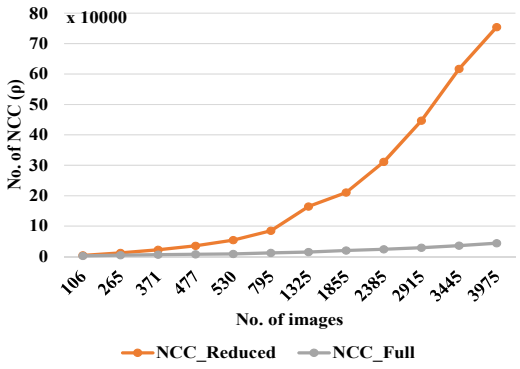
(a) P, R and $F-measure$



(b) RI and ARI



(c) Cr



(d) $NCC\_Reduced$

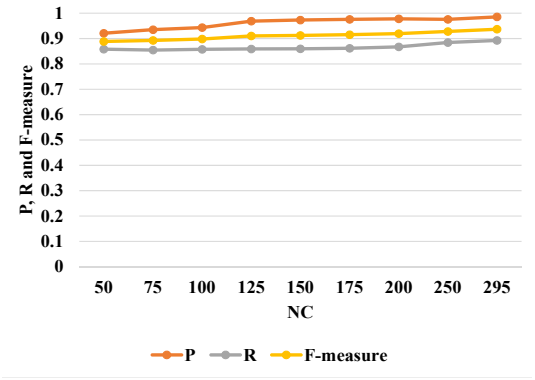
Figure 4.37: Large scale clustering analysis of CFIC algorithm and $NC \gg SC$ problem.

4.6.4 $NC \gg SC$ analysis

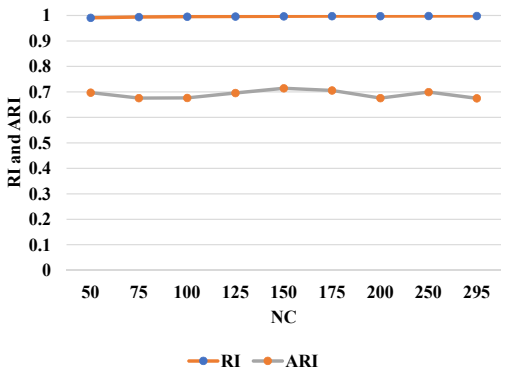
In this section, we are evaluating the proposed algorithm for the $NC \gg SC$ problem. The $NC \gg SC$ problem is somehow discussed in Section 4.6.3. To have a detailed analysis of the $NC \gg SC$ problem in a more challenging scenario, we use $D5$ dataset.

Full and reduced camera fingerprints are estimated from each image of each camera in the dataset. The CFIC algorithm is applied to cluster the images based on the fingerprints. The experiments are performed for fixed $SC = 20$ and varying the number of cameras NC. The experimental results obtained for NC equal to 50, 75, 100, 125, 150, 175, 200, 250 and 295 with fixed $SC = 20$, are shown in

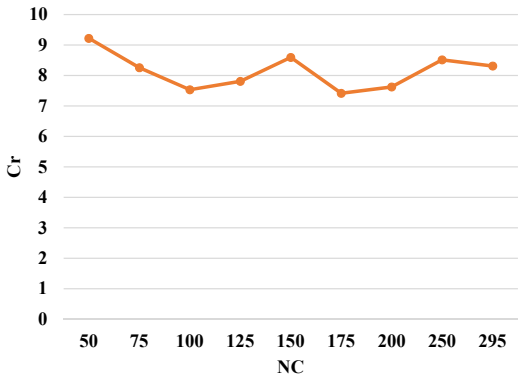
Figure 4.38. The results show that as the NC gets larger and larger than SC, the evaluation metrics of P, R and $F-measure$ get better and better. The resulting RI is high for different experiments with different NC. While, the values of ARI vary a little with the change in the number of images under test. The results obtained in terms of P, R, $F-measure$ and RI prove that the CFIC algorithm does not suffer from $NC \gg SC$.



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



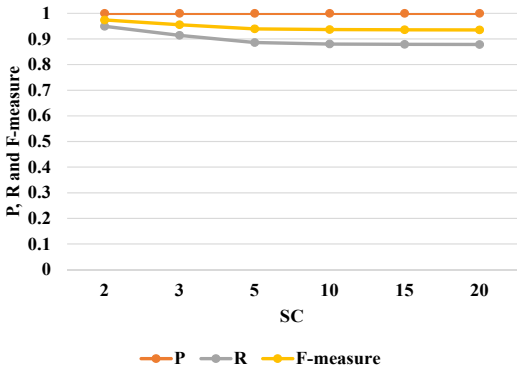
(b) RI and ARI



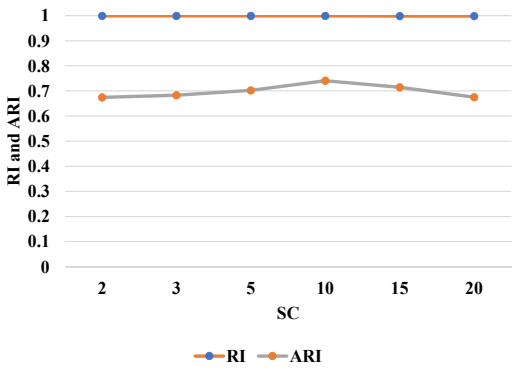
(c) Cr

Figure 4.38: The robustness of CFIC algorithm to $NC \gg SC$ problem, for various values of NC and fixed $SC = 20$.

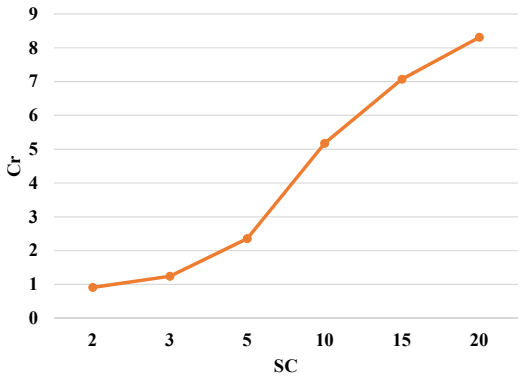
The results also show that as the NC varies the Cr fluctuates. The increase in the size of datasets tries to increase Cr but at the same time, the increase in hardness of dataset with increasing NC, attempts to reduce the Cr. Therefore, both the increasing size and increasing hardness balance each other effect. The addition of bad quality images results in a reduction of Cr. So, the fluctuations in Cr occur because of images of different qualities.



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



(b) RI and ARI



(c) Cr

Figure 4.39: The robustness of CFIC algorithm to $NC \gg SC$ problem, for various values of SC and fixed $NC = 295$.

Further experiments are performed for fixed $NC = 295$ and varying the SC. The experimental results obtained for SC equal to 2, 3, 5, 10, 15 and $SC = 20$, are shown in Figure 4.39. The results in Figure 4.39a, show that the evaluation metrics of P, R and $F-measure$ slightly decrease with increase in SC with fixed $NC = 295$. The results in Figure 4.39b, show that the CFIC algorithm results in high and stable RI for different SC. However, the ARI varies a bit with the change in SC. The higher values of P, R, $F-measure$ and RI at lower SC prove the robustness of the CFIC algorithm against $NC \gg SC$ problem.

While observing the resulting Cr for different SC, Figure 4.39c shows that as the SC increases the Cr increases. This increase in Cr is due to the increase in the number of images with an increase of SC. This also confirms the suitability of the algorithm for large scale clustering.

The performance of the CFIC algorithm on small, medium and large datasets shows that the CFIC is a very powerful and efficient clustering algorithm. The CFIC performs very well in different scenarios of $NC \gg SC$. The algorithm has a

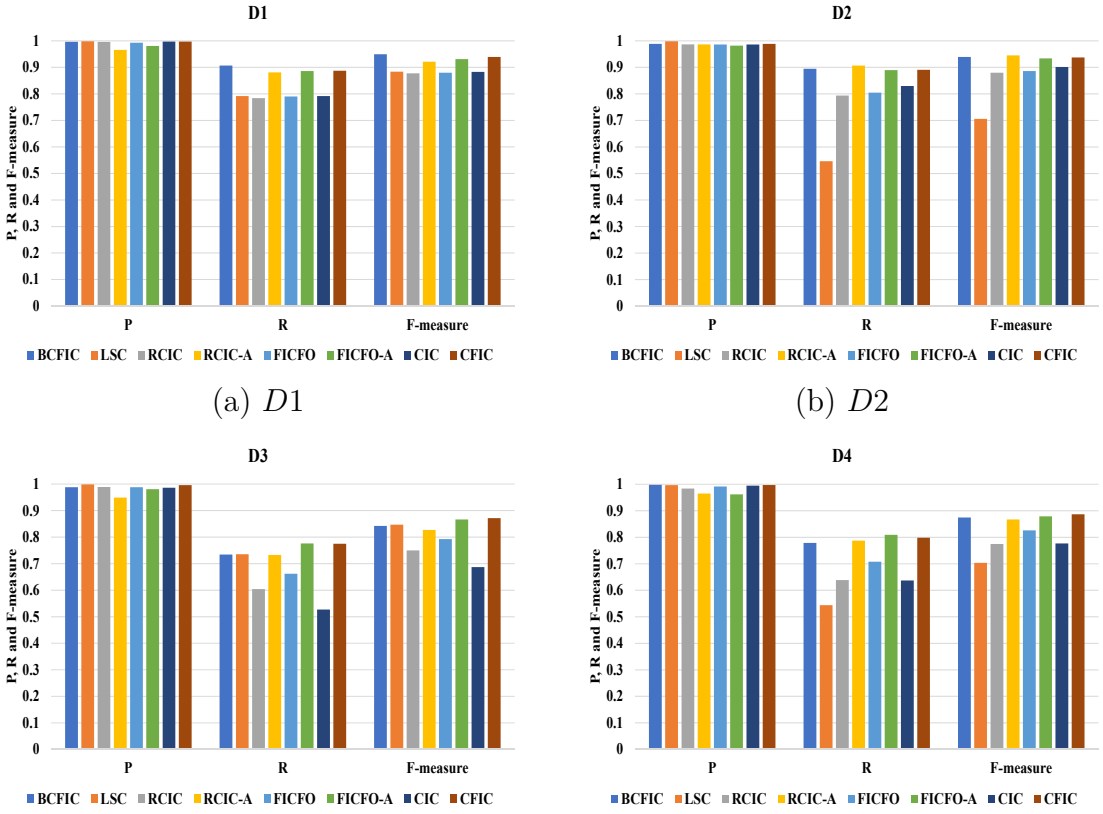
higher Cr and the lowest computational cost among all our proposed algorithms, i.e., RCIC, RCIC-A, FICFO, FICFO-A and RCIC algorithm. Along with the lower computational complexity, the quality of clusters is also comparable with the other proposed algorithms. The performance of the proposed algorithms is compared with each other and with the state-of-the-art algorithm in the following section.

4.7 Comparison of Image Clustering Algorithms

In this section, we present the comparison of the proposed image clustering algorithm among themselves and with state-of-the-art algorithms. The main aim of this work is to reduce the computational cost of image clustering. That's why the proposed algorithms, i.e., RICIC, RCIC-A [70], FICFO, FICFO-A [69], CIC, and CFIC are focused on the reduction of computational complexity per image while preserving the quality of the constructed clusters. The proposed algorithms are compared among themselves as well as with state-of-the-art algorithms. The evaluation metrics of P, R, $F-measure$, RI and ARI along with the complexity reduction Cr are used for comparison.

The proposed algorithms are compared with state-of-the-art algorithms, namely blind camera fingerprinting image clustering (BCFIC) [8] and large scale image clustering (LSIC) [81]. As discussed in Section 3.4, we don't include [101, 102] in the comparison since this method is not directly comparable to our methods. All comparison are made using the four different small datasets i.e., $D1$, $D2$, $D3$ and $D4$ [81]. Which are presented in detail in Section. 4.1.

The evaluation metrics i.e., P, R, $F-measure$, RI and ARI and the Cr are computed for each algorithm using each dataset. The results obtained in terms of P, R and $F-measure$ are shown in Figure 4.40. The results in Figure 4.40a, using the $D1$ dataset, show that RCIC, FICFO, CIC, and CFIC algorithm have a comparable P as that of BCFIC and LSIC algorithms. While the RCIC-A and FICFO-A have a slightly reduced P, with respect to state-of-the-art algorithms. The reduction in P of RCIC-A and FICFO-A is due to the false attraction of some bad fingerprints at the attraction stage. The attraction process results in an increase of R for the RCIC-A and FICFO-A algorithms. The results show that using $D1$, the RCIC-A, FICFO-A, and CFIC result in R comparable to BCFIC and higher than LSIC, RCIC, FICFO, and CIC algorithms. The results obtained in terms of $F-measure$, which show the combined effect of P and R, show that all the proposed algorithms result in $F-measure$ higher than the LSIC algorithm. The CFIC algorithm performs better than all the algorithms except BCFIC, which has a bit higher value of $F-measure$ than the CFIC algorithm. It can also be observed that the RCIC-A and FICFO-A also have significantly good values of $F-measure$.



(a) $D1$ (b) $D2$

(c) $D3$ (d) $D4$

Figure 4.40: Comparison of the proposed algorithms with other techniques on the basis of P, R and $F-measure$.

The results in Figure 4.40b, using the $D2$ dataset, show that all the algorithms result in quality clusters with significantly high values of P. However, looking to results obtained in terms of R and $F-measure$ show that using the $D2$ dataset, the RCIC-A, FICFO-A and CFIC algorithms perform equivalently to the BCFIC algorithm and better than the rest of the presented algorithms. The results also show that R and $F-measure$ of RCIC, FICFO, and CIC algorithms are better than the state-of-the-art LSIC algorithm.

The results in Figure 4.40c, using the $D3$ dataset, show that all the algorithms, except RCIC-A, results in quality clusters with significantly high values of P. However, CFIC and LSIC perform better than all the presented techniques. The results show that using $D3$, the CFIC and FICFO-A have higher R and $F-measure$ as compared to the rest of the clustering algorithm. Therefore it can be observed that all the proposed algorithms result in a higher $F-measure$ than LSIC algorithm.

The results in Figure 4.40d, using the $D4$ dataset, show that FICFO, CIC, and CFIC algorithms have comparable P as that of state-of-the-art LSIC and BCFIC

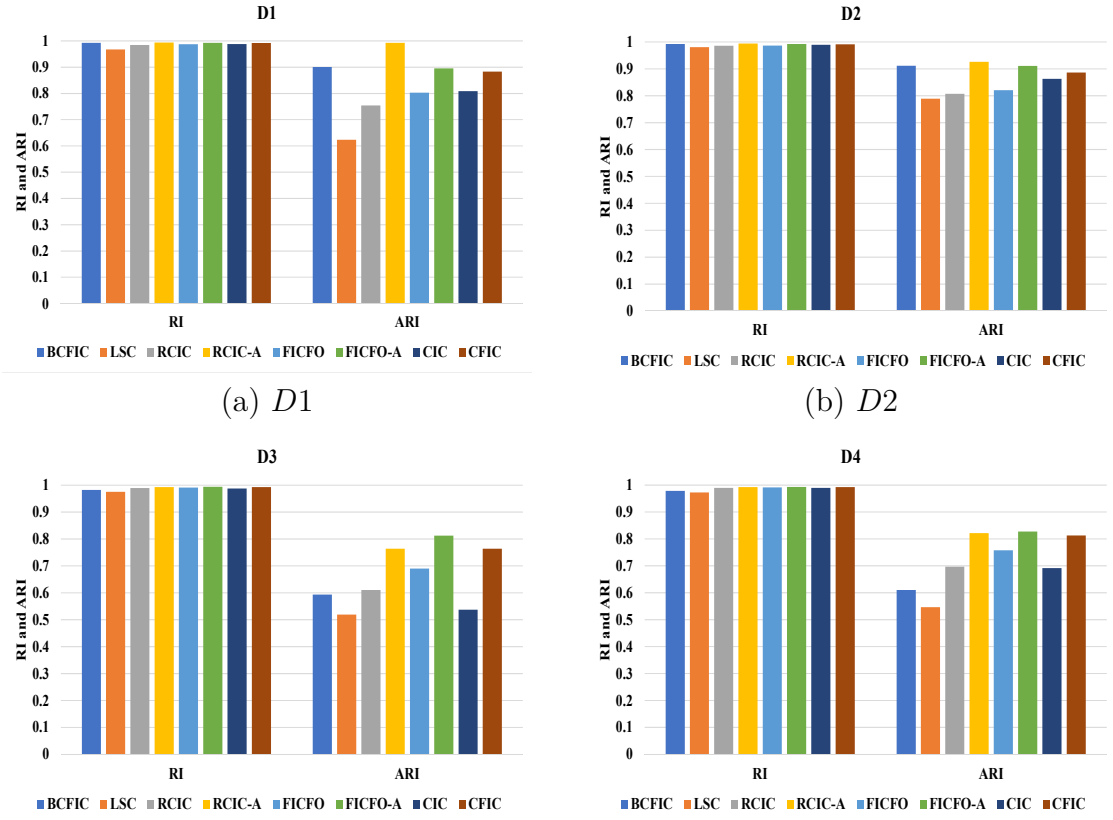
algorithms. While looking at the results of R, it can be observed that RCIC-A, FICFO-A, and CFIC perform better than the state-of-the-art and other proposed techniques. The computed $F - measure$ for each algorithm using the $D4$ dataset shows that the CFIC algorithm performs better than all the algorithms. While, the performance of RCIC-A and FICFO-A, in terms pf $F - measure$ is better than LSIC, RCIC, FICFO, and CIC algorithm and comparable with BCFIC algorithm.

From the results shown in Figure 4.40, it can be concluded that the CFIC algorithm performs better than all the presented algorithms on all the datasets, i.e., $D1$, $D2$, $D3$ and $D4$.

After making a comparison using P, R and $F - measure$, now the proposed algorithms are compared among themselves and with the state-of-the-art algorithm using the RI and ARI parameters. The results obtained on the different datasets are presented in Figure 4.41.

The results show that the proposed algorithms perform better than the LSIC algorithm on all the datasets. While the performance is equivalent to that of the BCFIC algorithm. The lower values of RI and ARI of the LSIC algorithm are due to the un-clustered fingerprints. The LSIC algorithm, while clustering images using camera fingerprints, declares all the fingerprints that belong to clusters having a size smaller than a predefined dimension, as unclustered. The results show that the proposed RCIC, RCIC-A, FICFO, FICFO-A, and CIC algorithms perform slightly lower than the BCFIC and CFIC algorithms, in case of easy datasets, i.e., $D1$ and $D2$. But, their performance on hard datasets, i.e., $D3$ and $D4$ is comparable with the performance of the BCFIC and CFIC algorithms. The analysis of RI and ARI show that the CFIC algorithm performs better than all the algorithms presented in Figure 4.41.

The overall comparative analysis on the basis of evaluation metrics i.e., P, R, $F - measure$, RI and ARI, show that the all proposed algorithms provide reasonably high quality clusters while CFIC performs comparable or even better than the rest of the proposed algorithms as well as the state-of-the-art algorithms, i.e., BCFIC and LSIC.

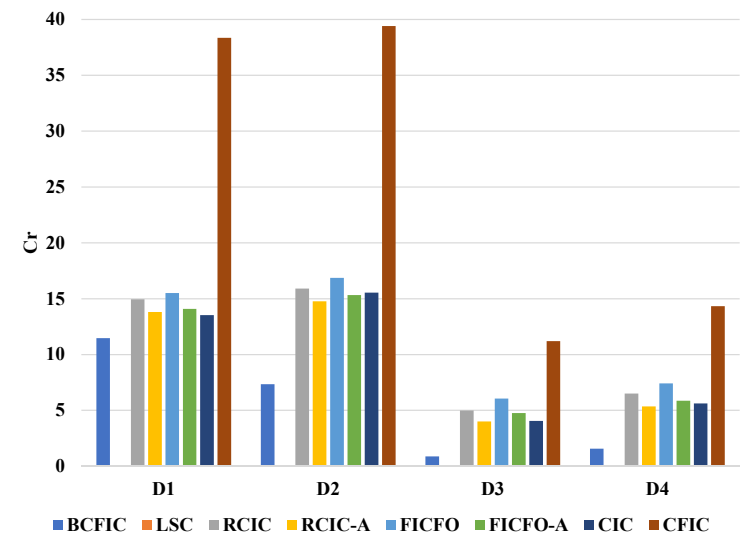


(a) $D1$ (b) $D2$

(c) $D3$ (d) $D4$

Figure 4.41: Comparison of the proposed algorithms with other techniques on the basis of RI and ARI.

To compare the algorithms among themselves and with the state-of-the-art algorithms of BCFIC and LSIC using the computational complexity the complexity reduction obtained by all algorithms is calculated. The results are presented in Figure 4.42. The total complexity t_c of BCFIC, LSIC, RCIC, RCIC-A, FICFO, FICFO-A, CIC and CFIC algorithms are computed in different manners. Because the BCFIC, RCIC, RCIC-A, FICFO, FICFO-A and CIC algorithms use only the full camera fingerprints while the LSIC and CFIC algorithms use the reduced and full fingerprints for clustering. The calculation of t_c and Cr is explained in Section 4.2 with detail.



Cr

Figure 4.42: Comparison of the proposed algorithms with other techniques on the basis of Cr.

The results show that all the proposed algorithms have higher Cr and lower computational complexity than the state-of-the-art algorithms. The Cr computed for RCIC and FICFO shows that the sorting of fingerprints helps to reduce the computational complexity and increases Cr. While the attraction stage decreases the Cr, that's why RCIC-A and FICFO-A have lesser Cr than RCIC and FICFO, respectively. The CIC algorithm also has a higher Cr and reduced complexity in comparison with the BCFIC and LSIC algorithms. However, its complexity is higher than the rest of the proposed techniques on all the datasets except $D2$, on which the CIC shows better performance than RCIC-A and FIFO-A algorithms. The Cr of the CFIC algorithm is significantly higher than all the presented algorithms, and hence it has the least computational cost per camera fingerprints. The reduction in the computational cost is due to the use of sorted reduced fingerprints for clustering.

BCFIC algorithm has a high computation cost because it performs three rounds to construct a single cluster. These rounds are repeated for each cluster. Conversely, the proposed algorithms select a fingerprint either at random or the best fingerprint among all un-clustered, as a reference, to construct a cluster. While the CFIC picks the best reduced fingerprint to build a cluster and performs very little number correlation on full fingerprints in the fine clustering. The LSIC complexity is quite evident due to coarse clustering, fine clustering, and attraction.

From the analysis of the clustering algorithms, it can be concluded that the CFIC is so far the best image clustering algorithm. The CFIC constructs high quality clusters with higher values of P, R, $F-measure$, RI and ARI on different

types of small datasets at a significantly lower computational cost, which is apparent from the high Cr in Figure 4.42.

Chapter 5

ACO based Data Hiding in the Complex Region Pixels

This chapter presents a data hiding technique that hides secret information in the least significant bits (LSB) of the pixels of a complex region in a cover image. Ant colony optimization (ACO) is used to identify edges of the cover image, and the LSB substitution technique is used to embed the secret information in the pixels that belong to the complex region. The ACO-based data hiding in a complex area develops a pheromone matrix that identifies the complex region. The local variations in the pixels' values are detected using the ant's movement. The local variations are used to construct a pheromone matrix. The LSBs of the pixels of edges are substituted with secret message bits in order to hide a secret message in these pixels.

5.1 Introduction

Data hiding refers to methodologies used to transmit information using existing media as a cover. It can be used for different purposes: steganography uses data hiding to transmit a message secretly. Differently from cryptography, this does not protect the confidentiality of messages, but hides the fact that a message is transmitted; watermarking is used to place a mark on the media [57, 112]. Data hiding, in general, uses digital audio, video, and text as a cover. However, images are preferred and got more attention due to their level of redundancy. A very simple data hiding technique is based on LSBs substitution and embeds secret information bits in the LSBs of cover image pixels. However, even if this simple embedding technique may be undetectable by human visual system (HVS), it fails under a simple statistical test like histogram inspection. In order to be used for steganography, more advanced LSB replacement techniques should be used [93]. LSB substitution is not a robust watermarking technique, since even a slight modification of the cover

destroys the hidden message. However, LSB substitution can be used for fragile watermarking, e.g., a sort of mark that should not be visible in order not to degrade the image and that can be used to check image integrity since any modification, e.g., compression, will likely erase it.

Several researchers explored the field of data hiding, and some of them proposed many useful data hiding techniques to improve security and ensure the existence of hidden information protected from unintended parties. A couple of initial and famous data hiding techniques were presented by Honsinger et al. and Fridrich et al., exploiting the digital image pixel directly and hiding secret information in the pixels [28, 46]. The technique uses a fixed number of LSB of pixels to embed information. The information hidden, using the fixed LSB approach, can be recovered easily if its presence is suspected. Therefore, it is better not to use a fixed number of LSB for data hiding. The key-based approach can be a good solution for hiding information at random positions of the cover media using a specific key, and the key is shared with the intended party only. However, in the case of hiding information in edges, the position of information hiding is decided based on edges not on key. The other approach is to hide a different number of LBS in different pixels of cover images, which make the retrieval of hidden information difficult for an unauthorized party.

Using this idea, Sahib et al. devised a variable least significant bits (VLSB) data hiding technique. He implemented the VLSB data hiding using modular distance technique (MDT) [71], decreasing distance decreasing bits algorithm (DDDBA) [50], varying index varying bits substitution (VIVBS) algorithm [68]. To enhance the security of hidden information, Sahib et al. presented a block chaining based data hiding approach. He got the inspiration from chipper block chaining (CBC) encryption and proposed stego block chaining (SBC) and enhanced stego block chaining (ESBC) technique for improved and more secure hiding of information [73].

Data hiding aims to hide information in a cover medium in such a manner that the existence of the hidden information remains undetectable to the HVS and no noticeable variation is created. It has been reported in the literature that LSB substitution has some HVS limitations. The HVS can detect any changes introduced in the smooth area of the cover image. While the changes created in complex regions are relatively difficult to identify with the naked eye. This makes the complex region of the cover image more suitable for data hiding, and therefore, secret information is embedded in the LSB of the pixels of edges [122, 39]. Data hiding techniques using edges for hiding information keep the smooth region of the cover medium unaffected or least affected. In other hiding techniques, larger information hiding is done in the pixels of the complex region than that of smooth region pixels, and give good quality stego images. The Data hiding techniques using the edges for information hiding include LSB methods [38], PVD methods, and side-match methods [38, 44], and much detail are available in [47, 111]. The

hiding capacity of such techniques is minimal and small [44, 45, 114]. Jung et al. [58] improved the hiding capacity by presenting a method that used both smooth and complex region pixels for data hiding. The resulting stego images are more distorted. Therefore, due to low hiding capacity and changes introduced in the smooth region, these techniques do not satisfy the rules of data hiding in edges, in the real sense.

The ACO based data hiding in edges technique is one such technique that hides secret information in complex region pixels of the cover image. The hidden information does not create significant distortion to attract human consideration, and the information remains imperceptible to HVS. Our technique uses the ACO algorithm [18, 19] for edges detection [121], and secret information are embedded in LSB of the complex region pixels [36, 88]. But as discussed earlier, the hiding of information in the LSBs may not be robust to simple statistical steganalysis, e.g., histogram inspection. However, the ACO based data hiding technique can be used as a fragile watermarking scheme. The detailed implementation of the technique is presented in Section 5.2, with experimental results in Section 5.3 and discussion ends up with comparison in Section 5.4.

5.2 Proposed Technique

The initial step, to hide secret information in edges, is to detect the pixels that belong to the complex region of a cover image. The complex region pixels are used for information hiding while the smooth region pixels are left unaffected. Numerous ways, e.g., canny edge detection, Deriche, differential, Sobel, Prewitt, Roberts cross, and other, exist in the literature that can be used for this purpose. These methods are used to detect complex region in digital images, but, the performance of some of these techniques do not meet the needs completely. These techniques detect weak and disconnected edges, and the corresponding pixels are considered as part of the real complex region. The edges based information hiding using these techniques also subject some of those pixels that do not belong to edges to the information hiding process. This makes such an information hiding technique more sensitive to noise. The ACO based method is used to detect edges in the cover image [119, 72], and using a 4LSB substitution mechanism, the information bits are hidden in the pixels of the detected edges.

The edges are detected in two phases; initially, edges are detected directly using the cover image, using ACO based edge detection technique. Then secret message bits are hidden in the LSBs of pixels that belong to edges. The hiding process may possibly affect the edges in the stego images. Therefore, the resulting stego image is then subjected to edge detection. The edges detected in the original cover image and stego image are compared and the pixels that are common in the complex region of both detected edges are considered as final edges. This process helps in

avoiding weak edges.

Although the final edges are relatively strong with respect to primary edges, however, embedding the message bit in the cover images based on the final edges, we will obtain a different image with respect to the first embedding. Likely, ACO will mark as edges some pixels that do not belong to strong edges when running on this image. Therefore, a way is needed to transmit the side information regarding the position of edges used for hiding information. The 1_{st} LSB of the cover image is used for this purpose. The 1_{st} LSB is set to 0 when the pixel belongs to the complex region and message bits are hidden it otherwise, 1_{st} LSB is set to 1. The value of 1_{st} LSB is used for the retrieval of hidden information.

The detection based on ACO, buildups a pheromone matrix, utilizing several ants and moving the ants across the pixels of a 2-D image. The local variation in the pixel values plays the role of guiding factor for the movement of the ants. The values of the matrix contain the edge information at the position of each pixel of the cover image. The ACO based technique is repeated N times to build a pheromone matrix. The process performs both construction and update steps iteratively. After the successful buildup of the matrix, the decision process is used to assign the pixels either to the complex region or smooth region. The steps involved to accomplish the goal and implement our technique are presented and explained in the subsequent sections.

5.2.1 ACO based Edge Detection

The cover image is initially subjected to ACO based edge detection to distinguish pixels of the complex region from the smooth region's pixels. The edge detection process involves the following steps.

Initialization

As we know that a digital image is a 2-D array of pixels with a specific intensity level; let say I. Let consider a gray-scale image of size $M_1 \times M_2$, as cover to hide secret information. A total of ants K are randomly deployed on the cover image Ć. Each pixel of the cover image is treated as a node. The process of edge detection is initiated, and the initial value of each pheromone matrix's component $\tau^{(0)}$ is set to a constant $\tau_{initial}$.

Construction

The construction mechanism is composed of several steps and is done iteratively. At the n_{th} construction-step, one ant, from a total of K ant, is randomly chosen. The chosen ant can move across the pixels of the cover image for N_{mov} movement

steps. The movement of the ant from initial node (x, y) to its neighbor node (i, j) is done according to the transition probability $P^{(n)}_{(x,y)(i,j)}$ as given by Eq. 5.1.

$$P^{(n)}_{(x,y)(i,j)} = \frac{\left(\tau^{(n-1)}_{i,j}\right)^{\alpha} (\eta_{i,j})^{\beta}}{\sum_{(i,j)\in\Omega_{(i,j)}} \left(\tau^{(n-1)}_{i,j}\right)^{\alpha} (\eta_{i,j})^{\beta}} \tag{5.1}$$

Where, $\tau^{(n-1)}_{i,j}$ is the pheromone value at node (i, j), $\Omega_{(i,j)}$ is the neighbourhood (4 or 8-connected) node of the node (x, y), $\eta_{i,j}$, is the heuristic information at node (i, j), α is the influence of the pheromone matrix and β is the influence of heuristic matrix.

The heuristic information at any node (i, j) is calculated using Eq. 5.2.

$$\eta_{i,j} = \frac{V_c\left(\acute{\mathrm{C}}_{i,j}\right)}{\acute{\mathrm{Z}}} \tag{5.2}$$

Where $\acute{\mathrm{Z}}$ is the normalization factor and given by Eq. 5.3.

$$\acute{\mathrm{Z}} = \sum_{i=1:M_1} \sum_{j=1:M_2} V_c\left(\acute{\mathrm{C}}_{i,j}\right) \tag{5.3}$$

Where, $\acute{\mathrm{C}}_{i,j}$ is the intensity level of pixel (i, j) of image $\acute{\mathrm{C}}$.

The $V_c(\acute{\mathrm{C}}_{i,j})$ depends on the variations in the gray level intensities of pixels in the clique c and is represented as given in Eq. 5.4.

$$\begin{aligned} V_c\left(\acute{\mathrm{C}}_{i,j}\right) = f\Big(&|\acute{\mathrm{C}}_{i-2,j-1} - \acute{\mathrm{C}}_{i+2,j+1}| + |\acute{\mathrm{C}}_{i-2,j+1} - \acute{\mathrm{C}}_{i+2,j-1}| + |\acute{\mathrm{C}}_{i-1,j-2} - \acute{\mathrm{C}}_{i+1,j+2}| \\ &+ |\acute{\mathrm{C}}_{i-1,j-1} - \acute{\mathrm{C}}_{i+1,j+1}| + |\acute{\mathrm{C}}_{i-1,j} - \acute{\mathrm{C}}_{i+1,j}| + |\acute{\mathrm{C}}_{i-1,j+1} - \acute{\mathrm{C}}_{i-1,j-1}| \\ &+ |\acute{\mathrm{C}}_{i-1,j+2} - \acute{\mathrm{C}}_{i-1,j-2}| + |\acute{\mathrm{C}}_{i,j-1} - \acute{\mathrm{C}}_{i,j+1}|\Big) \end{aligned} \tag{5.4}$$

The $f(.)$ has four different options. The function can be Flat, Gaussian, Sine and Wave and all of these four functions are considered to implement the hiding technique. The Flat, Gaussian, Sine and Wave fictions expressed in Eq. 5.5, Eq. 5.6, Eq. 5.7 and Eq. 5.8, respectively.

$$f(x) = \lambda x \quad \forall x \geq 0 \tag{5.5}$$

$$f(x) = \lambda x^2 \quad \forall x \geq 0 \tag{5.6}$$

$$f(x) = \begin{cases} sin\left(\frac{\pi x}{2\lambda}\right) & 0 \leq x \leq \lambda \\ 0 & \text{else} \end{cases} \tag{5.7}$$

$$f(x) = \begin{cases} \pi x \frac{sin\left(\frac{\pi x}{\lambda}\right)}{\lambda} & 0 \leq x \leq \lambda \\ 0 & \text{else} \end{cases} \tag{5.8}$$

Where λ is the shape control parameter of the functions.

Updating Stage

The updating of the pheromone matrix is a two steps process. The first updating is done at each individual construction step, with the movement of each individual ant, as expressed in Eq. 5.9.

$$\tau_{i,j}^{(n-1)} = \begin{cases} (1-\rho)\,\tau_{i,j}^{(n-1)} + \rho\Delta_{i,j}^{(K)} & \text{if } (i,j) \text{ is visited by } K \text{ ant} \\ \tau_{i,j}^{(n-1)} & \text{otherwise} \end{cases} \tag{5.9}$$

Where, ρ is the evaporation rates and $\Delta_{i,j}^{(K)}$ is equal $\eta_{i,j}$ and decided by heuristic matrix.

The next updating is done on the movement completion of the ant, in the construction step. The updating is mathematically expressed in Eq. 5.10.

$$\tau^{(n)} = (1-\psi)\,\tau^{(n)} + \psi\tau^{(0)} \tag{5.10}$$

Where ψ is the pheromone decay coefficient.

Decision Stage

The decision process is the step to decide whether a pixel is part of a complex region or a smooth region. This is the final process and results in a binary image. A threshold value is computed and is applied to the values pheromone matrix τ^N. The threshold is decided based on the criterion described in [121].

The average value of the pheromone matrix is chosen as the initial threshold of $T^{(0)}$. Then the elements of pheromone are classified into two groups. One group includes the values smaller than the $T^{(0)}$, while the second has the value higher than the threshold $T^{(0)}$. After that, the average value of each of the two groups is calculated. The average of both means is considered as a new threshold. The threshold computation process continues until the threshold approaches a steady value in terms of tolerance ϵ.

The decision for each pixel at (i,j) is done on the basis of the pheromone value $\tau_{i,j}^{(N)}$ at position (i,j) and the value is compared with the final threshold value $T^{(l)}$ as expressed in Eq. 5.11.

$$Ep_{i,j} = \begin{cases} 0 & \text{if } \tau_{i,j}^{(N)} \geq T^{(l)} \\ 1 & \text{otherwise} \end{cases} \tag{5.11}$$

Where Ep is the initial binary image.

The pixel at position (i, j) is declared as part of the complex region if the pheromone value at (i, j) is higher than the threshold. Otherwise, the pixels are considered as part of the smooth region.

5.2.2 Primary Data Hiding

The LSB of the pixels that belong to the complex region is used for information hiding. The cover image Ć and the binary image obtained after edges detection are considered, and the cover images are processed pixel by pixel. Each pixel $\acute{C}_{i,j}$ is checked, whether it is part of a complex region or a smooth region. If the pixel is found a part of the smooth region, it is not subjected to data hiding and is left unaffected. If the pixel is found a part of the complex region, then the pixel is subjected to LSB bits substitution, and the LSB is substituted with the bits of the secret information. The process stops when all pixels of the cover images are processed.

A pixel $\acute{C}_{i,j}$ is declared as a pixel of the complex region if its respective pixel of the binary image $Ep_{i,j} = 0$ and is processed as a pixel of the smooth region if $Ep_{i,j} = 1$. Let's consider a secret message m to be hidden in the complex region and Sp is the primary stego image obtained at the end of information hiding. The stego image Sp is given by Eq. 5.12.

$$Sp_{i,j} = \begin{cases} \acute{C}_{i,j} * m_n & \text{if } Ep_{i,j} = 0 \\ \acute{C}_{i,j} & \text{if } Ep_{i,j} = 1 \end{cases} \tag{5.12}$$

Where, $*$ is LSB substitution operator and m_n represent the n bits of message. The n may have different values from 1 to 4.

5.2.3 Final Edge Detection

The stego image is processed to detect the pixels belonging to the complex region. For this purpose, the ACO based edge detection technique, as given in Section 5.2.1, is applied to the primary stego image Sp. The edges detection results in secondary binary image Es, representing the edges of the stego images. The final binary image is obtained by considering the common pixels that belong to the complex region in both binary images, i.e., Ep and Es.

$$E_{i,j} = \begin{cases} 0 & \text{if } Ep_{i,j} = 0 \; and \; Es_{i,j} = 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.13}$$

The E is the final binary image representing the final edges. This used for final data hiding in the complex region of the cover image.

5.2.4 Final Data Hiding

The final binary image E is used to check the pixel of cover image $\acute{C}_{i,j}$, whether it is part of a complex region or smooth region. If the pixel belongs to complex region the 1_{st} LSB of the pixel is set to 0 and n bits of secret message m are hidden in the n LSBs, excluding the 1_{st} LSB, of the pixel; otherwise the 1_{st} LSB of the pixel is set to 1. The process stops when all pixels of the cover image are processed. A pixel $\acute{C}_{i,j}$ is declared as a pixel of the complex region if its respective pixel of the binary image $E_{i,j} = 0$, and is processed as a pixel of the smooth region if $E_{i,j} = 1$. At the end of the hiding process final stego image S is obtained. The stego image S is given by Eq. 5.14.

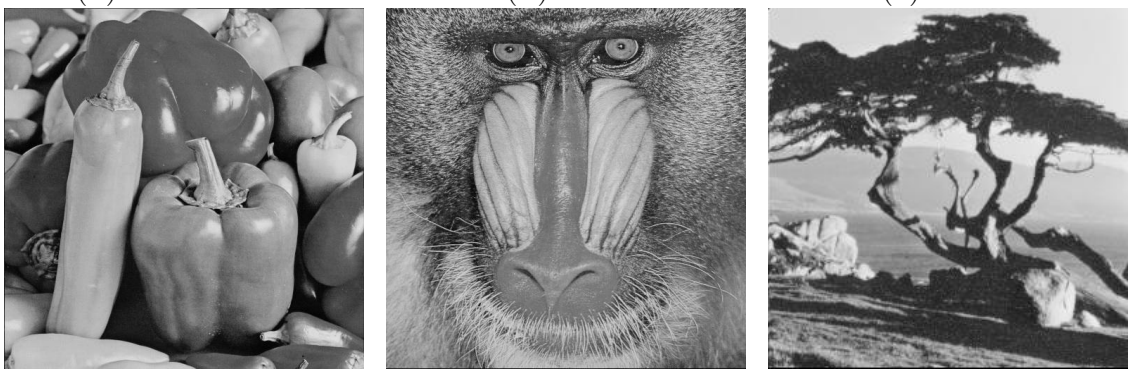
$$S_{i,j} = \begin{cases} \acute{C}_{i,j} * m_n & \text{if } E_{i,j} = 0 \\ \acute{C}_{i,j} & \text{if } E_{i,j} = 1 \end{cases} \tag{5.14}$$

Ensuring the recovery of hidden information without errors is essential for a data hiding technique. For this purpose, the 1_{st} LSB of each pixel of the stego image is used. The 1_{st} LSB of the all the pixels has the information about the pixels belonging to the complex region with hidden information. Retrieving the secret information, the 1_{st} LSB of each pixel of the stego image is checked if the 1_{st} LSB is equal to 0 the hidden message bits are retrieved from the other LSBs of the pixel and if 1_{st} LSB is equal to 1 then the pixel is ignored. The hidden message bits are retrieved by processing the whole stego image pixel by pixel, in the same way.

5.3 Results and Analysis

The secret information bits are hidden in the complex region of the cover image pixels using ACO based data hiding technique. The quantitative and qualitative results are obtained using different images as cover. The images used as cover include the images of Cameraman, Lena, House, Pepper, Mandrill, and Tree. The images are shown in Figure 5.1a-f. The cover images are of different sizes and are converted to grayscale images. The cover images are processed for edges detection using ACO based complex region detection technique, and further are subjected to information hiding. The complex region can be detected using ACO using four distinct functions, i.e., Flat, Gaussian, Sine, and Wave, as given by Eq. 5.5 to Eq. 5.8. After the pixels classification of complex and smooth regions, LSB substitution methodology is adopted to hide information in the LSBs of the pixels of a complex region and the 1_{st} LSB of each pixel is modified as discussed in the previous section.

(**a**) Cameraman (**b**) Lena (**c**) House



(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.1: Cover Images.

The cover images are processed for edge detection using ACO based technique. To detect edges the parameters of λ, α, β, ρ and ψ are set to 10, 1, 0.1, 0.1 and 0.05, respectively, for ACO to operate. The determination of the above parameters is critical to the performance of the proposed approach. Therefore, we have used the values reported in [114] for the ACO to operate. Then message bits are hidden in the LSBs of pixels that belong to the complex region. Here it is important to mention that 1-bit, i.e., 1_{st} LSB, is used as a key deciding whether the pixel belongs to complex region and contains the secret message bits or not and the other 3-bits, i.e., LSBs, per pixel are used to hide secret message bits in the complex region. After the hiding process, we get primary stego images. The primary stego images are again processed for secondary edge detection using ACO based technique. The primary and secondary detected edges are processed to find the complex region common in both detected edges. The common complex region is considered the final complex region. This has less number of pixels than that of primary and secondary complex regions.

The final stego image is obtained by setting the 1_{st} LSB of each pixel belonging to the complex region and containing hidden information, to 0 and other 3-bits are used for information hiding and the 1_{st} LSB of all other pixels is set to 1.

The hiding technique is quantitatively analyzed using the hiding capacity HC, and peak signal to noise ratio $PSNR$ and structural similarity $SSIM$. The HC and $PSNR$ are expressed as given in Eq. 5.15 and Eq. 5.16 respectively [121].

$$HC = \frac{No.ofbitshidden}{TotalbitsofCoverimage} \times 100 = \frac{m}{r \times c \times 8} \times 100 \tag{5.15}$$

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \tag{5.16}$$

Where MSE is the mean square error and is given as

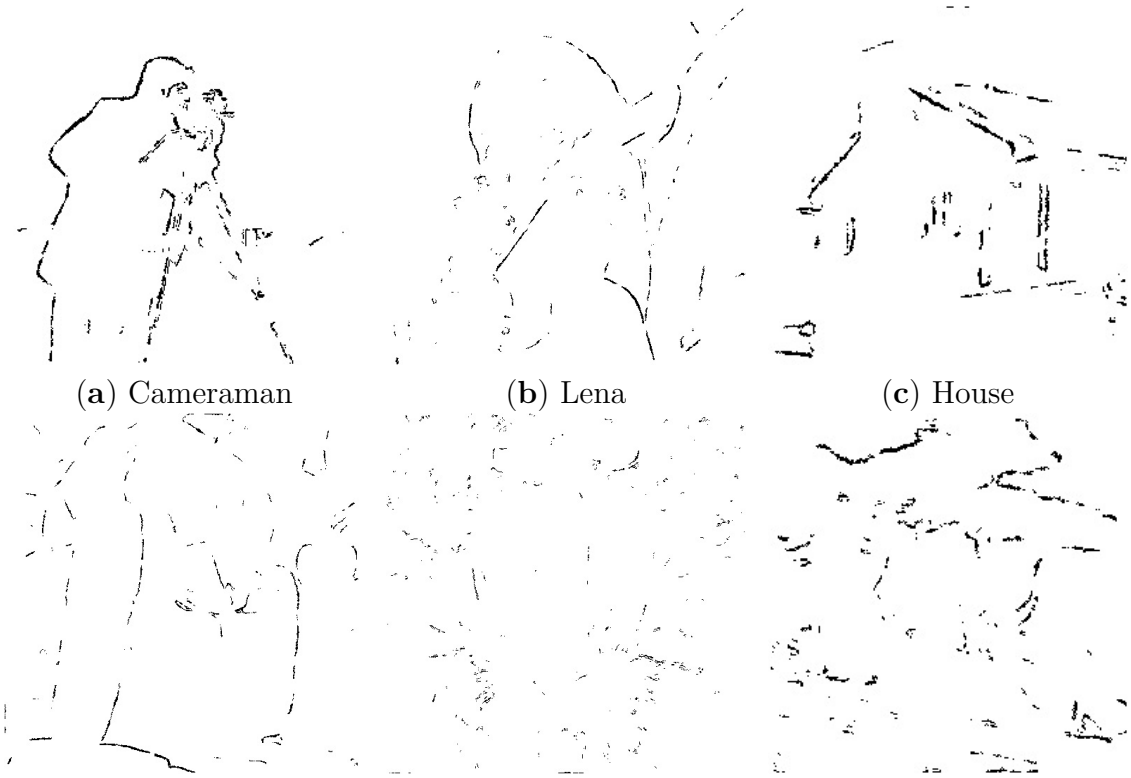
$$MSE = \frac{\sum_{i=1}^{r} \sum_{j=1}^{c} \left(C\left(i,j\right) - S\left(i,j\right)\right)^2}{r \times c} \tag{5.17}$$

The $SSIM$ [120], a perceptual metric that quantifies image quality degradation, is expressed as

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + C_1\right)\left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{5.18}$$

Where μ_x is the mean of x, μ_y is the mean of y, σ_x^2 is the variance of x, σ_y^2 is the variance of y, σ_{xy} is the covariance of x and y, C_1 and C_2 are the factors used to stabilize the division with weak denominator.

We have four different functions to detect edges and hide secret information using the ACO technique. In the first experiment, Flat function, as explained in Eq. 5.5 is used to identify the pixels of the primary and secondary complex regions of cover images shown in Figure 5.1a-f, based on the ACO technique. The primary and secondary edges are further processed to get the final complex region. The final complex regions are shown in Figure 5.2.

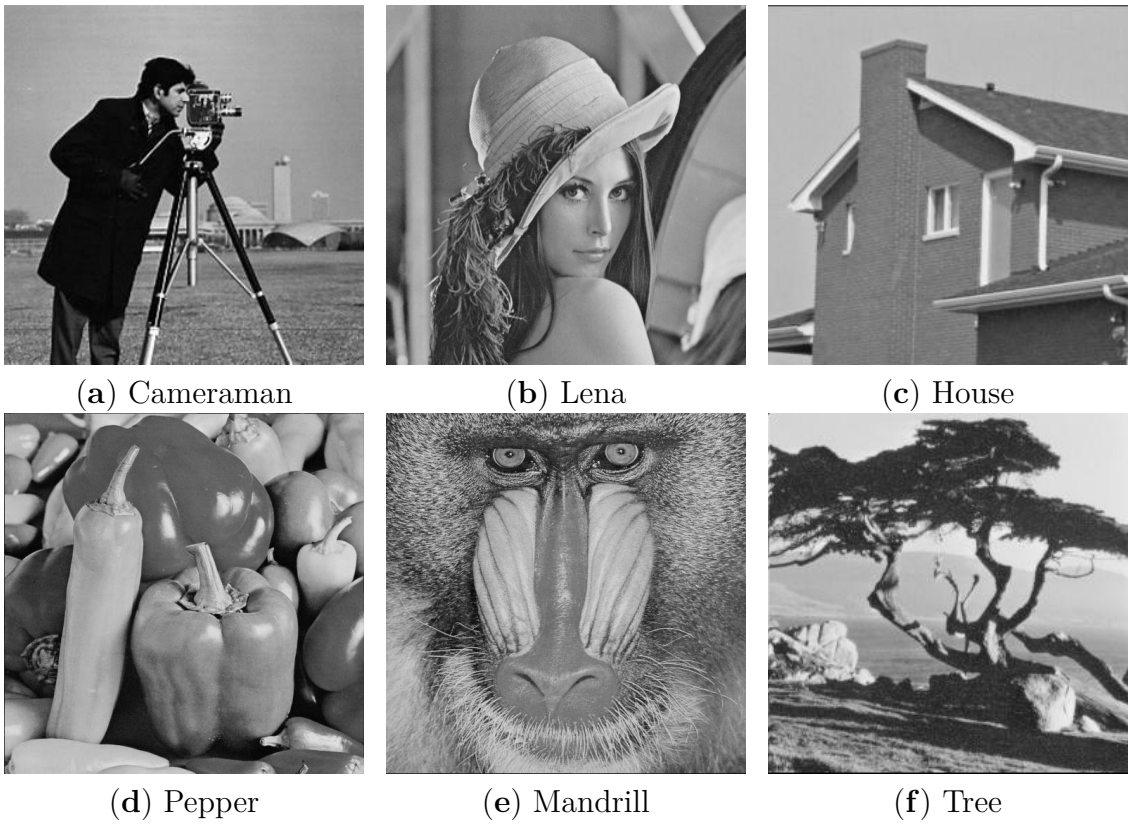


(**a**) Cameraman (**b**) Lena (**c**) House

(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.2: Final complex region detected using Flat function.

Each of the cover images presented in Figure 5.1a-f is subjected to the hiding process, and the secret information is hidden in the pixels of the final complex region, 3-bits per pixels, of cover images. While, the 1_{st} LSB of each pixel of the cover images is set to either 0 or 1 based on the criteria of information hiding and no hiding, respectively. The 1_{st} LSB of all the pixels of stego images contain the information helpful for information retrieval. The resulting stego images are shown in Figure 5.3a-f. The qualitative analysis shows that the presented hiding technique results in good quality stego images. The images do not have any significant distortion, and hence the existence of hidden secret information is undetectable to HVS.



(**a**) Cameraman (**b**) Lena (**c**) House

(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.3: Stego Images of ACO based data hiding in edges using Flat function ref to Eq. 5.5.

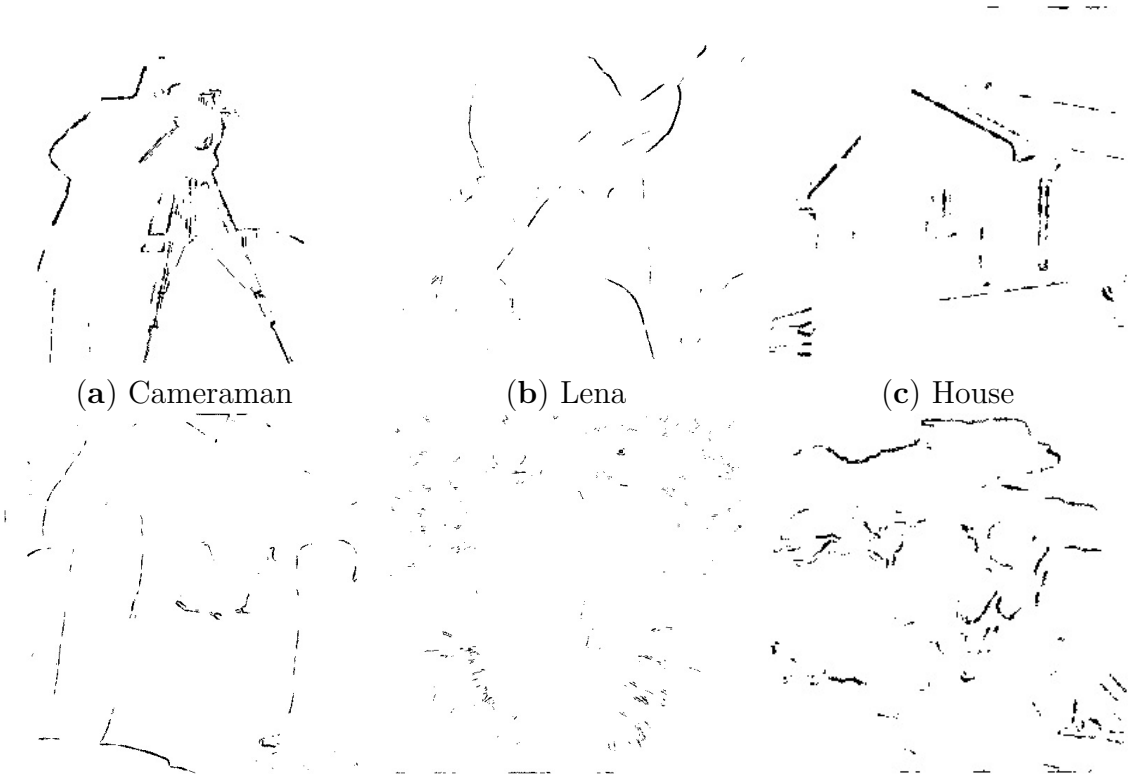
Quantitatively analyzing the hiding technique, the hiding capacity HC, $PSNR$ and $SSIM$ are calculated in the experiment using each of the cover images. The resulting values are shown in Table 5.1. The results show that for all previously mentioned cover images, the resulting stego images have a $PSNR$ of $37.6371dB$ and greater, which is higher than $30dB$, the minimum acceptable values for a stego image. The $SSIM$ obtained for different stego images is significantly high and results in value equal to 0.9912 and above for different cover images under test. While the HC remains greater than or equal to 0.2632%. The highest HC of 0.9109% is obtained using Tree as a cover image.

Table 5.1: $SSIM$, $PSNR$ and HC with Flat function.

Cover Image	$SSIM$	$PSNR(dB)$	$HC(\%)$
Cameraman	0.9974	44.1994	0.8021
Lena	0.9976	42.9513	0.3872
House	0.9976	42.4384	0.8011
Pepper	0.9976	42.7215	0.4118
Mandrill	0.9912	37.6371	0.2632
Tree	0.9985	39.186	0.9109

After the Flat function in Eq. 5.5, the Gaussian function in Eq. 5.6 is used to detect the final complex region based on the ACO technique. Each of the cover images presented in Figure 5.1 is processed to cover secret information in the 4-LSBs of the complex region's pixels of cover images. Out of the 4-bits, the 1_{st} LSB is set to 0 and is used as side information representing the pixels with hidden information and other 3-bits contain the hidden secret message bits. While the 1_{st} LSB of each pixel belonging to the smooth region is set to 1 and play its role in information retrieval.

The binary images showing the final complex region using the Gaussian function, are given in Figure 5.4. While the final stego images are shown in Figure 5.5.

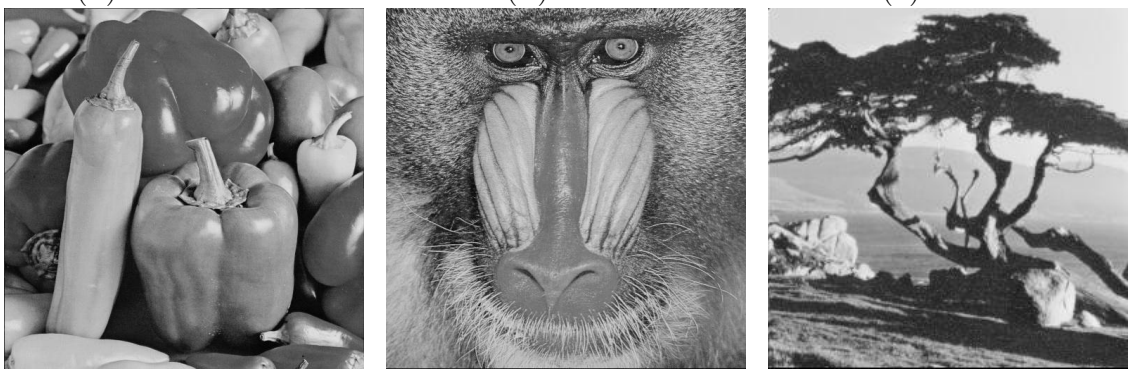


(**a**) Cameraman (**b**) Lena (**c**) House

(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.4: Final complex region detected using Gaussian function.

(**a**) Cameraman (**b**) Lena (**c**) House



(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.5: Stego Images of ACO based data hiding in edges using Gaussian function ref to Eq. 5.6.

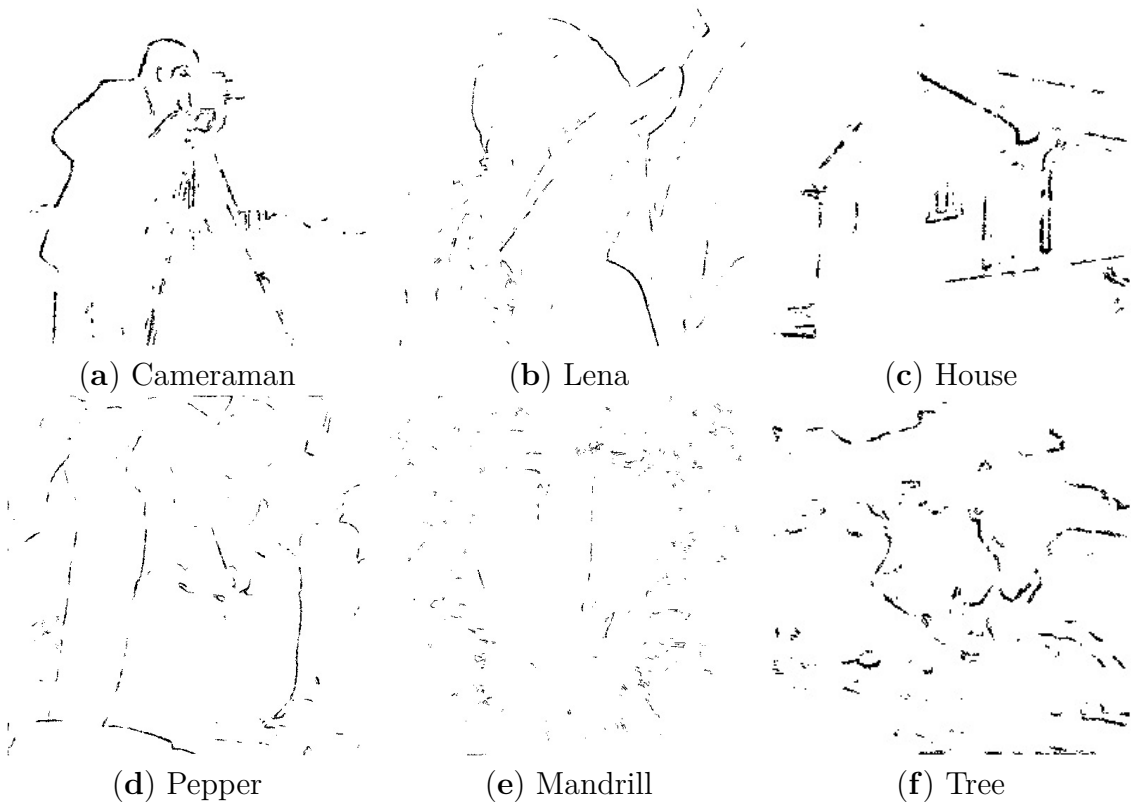
The resulting stego images are shown in Figure 5.5a-f. The qualitative analysis shows that the presented hiding technique gives high quality stego images as output. The stego images do not attract the attention of eavesdroppers. The hidden information does not add high distortion in stego images, and hence the existence of hidden secret data is undetectable to HVS.

Quantitatively examining the performance of the hiding technique, the HC, $PSNR$ and $SSIM$ are computed for individual images exposed to data hiding. The resulting statistics of HC, $PSNR$ and $SSIM$ are shown in Table 5.2. The results show that for all previously mentioned cover images, the resulting stego images have a $PSNR$ of $37.7077dB$ and greater, which is higher than $30dB$, the minimum acceptable values for a stego image. The $SSIM$ obtained for different stego images is significantly high and remains greater than or equal to 0.9915. The maximum HC, i.e., 0.8451%, is obtained for the image Tree. The results show that $PSNR$ and $SSIM$ obtained for stego images with Gaussian function is better than the Flat function. However, HC with Gaussian function is less than that of Flat function.

Table 5.2: $SSIM$, $PSNR$ and HC with Gaussian function.

Cover Image	$SSIM$	$PSNR(dB)$	$HC(\%)$
Cameraman	0.9975	44.393	0.6842
Lena	0.9978	43.1443	0.2332
House	0.9978	42.716	0.6071
Pepper	0.9977	42.946	0.2812
Mandrill	0.9915	37.7077	0.2011
Tree	0.9986	39.1908	0.8451

Similarly, all cover images used in the previous experiments, are exposed to edge detection using the Sine function in Eq. 5.7. After the detection of primary and secondary edges, final edges are obtained, as discussed in Section 5.2.3. The binary images obtained for each cover image, with Sine function using ACO based edge detection, are shown in Figure 5.6. Then the LSB replacement process is applied to cover the secure data in the 3-LSBs, excluding 1_{st} LSB, of the pixels of the final complex region of the cover image. Edges' information is embedded in the 1_{st} LSB of each pixel of the cover image. The 1_{st} LSB is set to 0 in the case of complex region's pixel and data hiding; otherwise, the 1_{st} LSB is set to 1. The final stego images are displayed in Figure 5.7a-f. The obtained stego images are of good quality and have no visibly significant distortion.

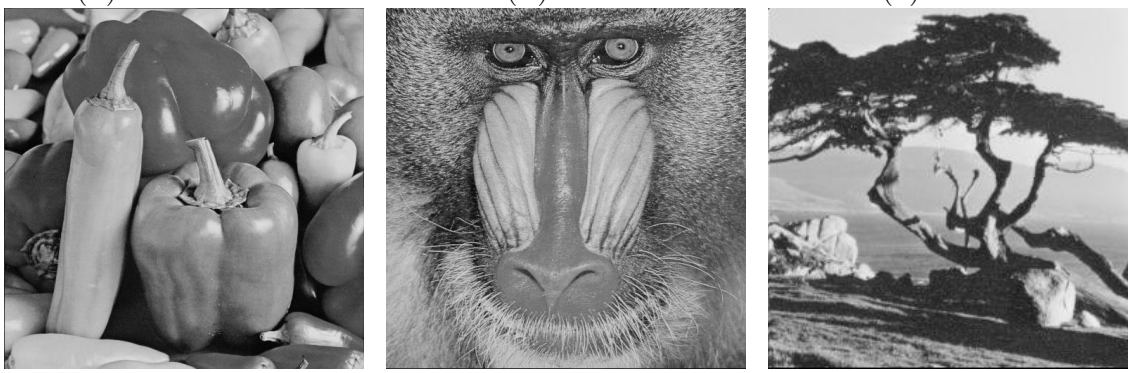


(**a**) Cameraman (**b**) Lena (**c**) House

(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.6: Final complex region detected using Sine function.

(**a**) Cameraman (**b**) Lena (**c**) House



(**d**) Pepper (**e**) Mandrill (**f**) Tree

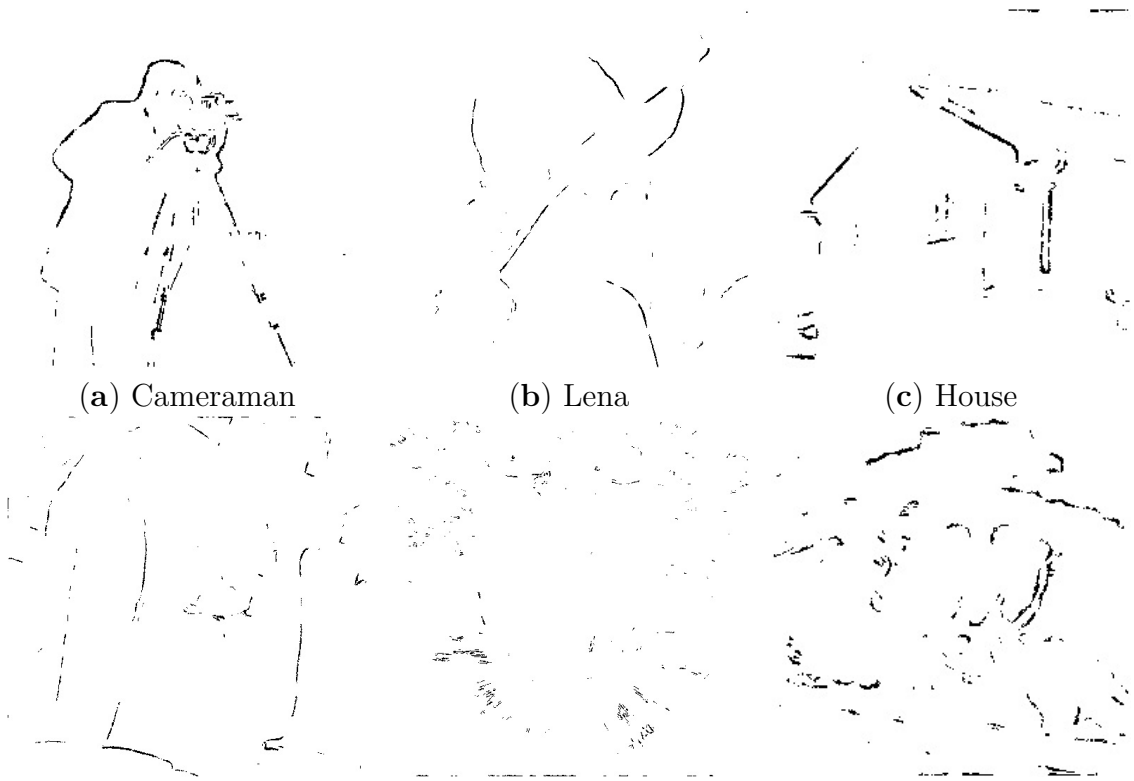
Figure 5.7: Stego Images of ACO based data hiding in edges using Sine function ref to Eq. 5.7.

The quantitative results of HC, $PSNR$, and $SSIM$ are computed in each experiment using each of the cover images and are reported in Table 5.3. The results show that for all previously mentioned cover images, the resulting stego images have a $PSNR$ of $37.6478dB$ and greater, which is much higher than $30dB$, the minimum acceptable values for a stego image. The $SSIM$ obtained for different stego images is significantly high and remains greater than or equal to 0.9913. The minimum HC, i.e., 0.2439%, is obtained for the image of Mandrill. While the maximum HC of 0.9665% is obtained for Tree image.

Table 5.3: $SSIM$, $PSNR$ and HC with Sine function.

Cover Image	$SSIM$	$PSNR(dB)$	$HC(\%)$
Cameraman	0.9974	44.2723	0.6862
Lena	0.9976	42.9346	0.3847
House	0.9976	42.5307	0.7742
Pepper	0.9976	42.7762	0.4319
Mandrill	0.9913	37.6478	0.2439
Tree	0.9985	39.1	0.9665

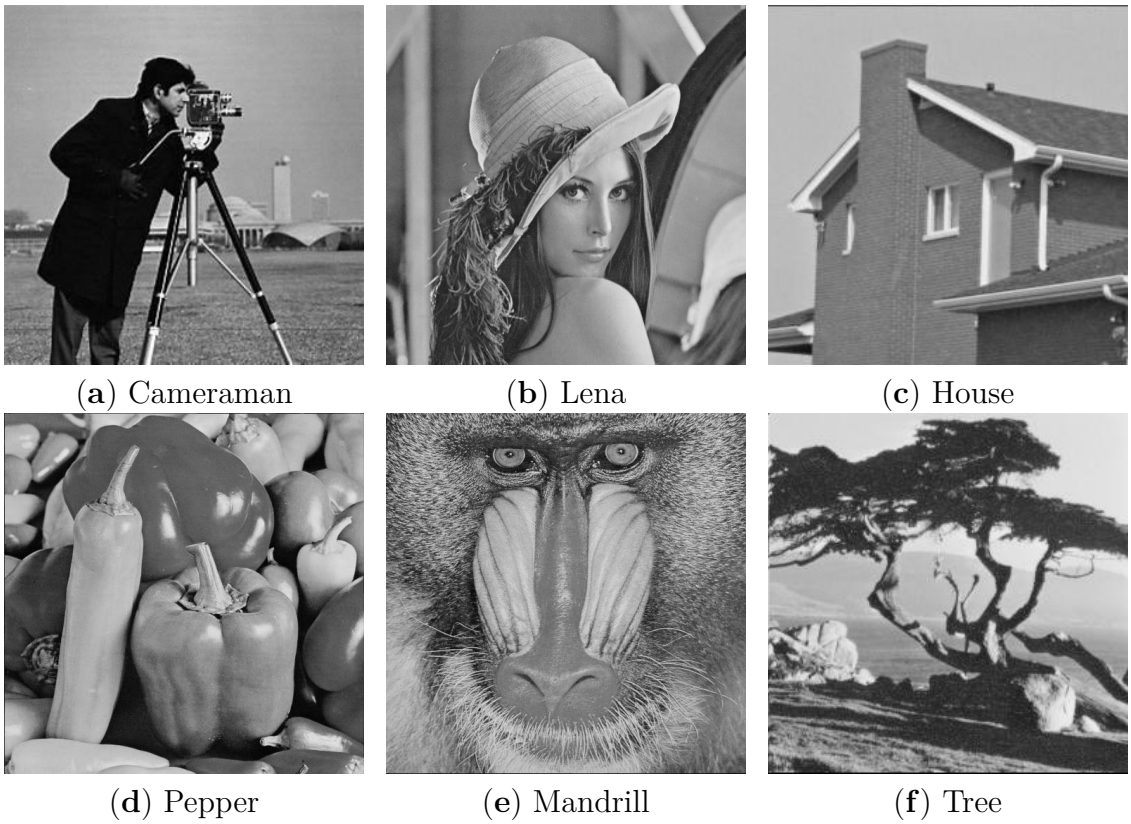
Like the Flat, Gaussian, and Sin functions, the Wave function as expressed in Eq. 5.8 is used in ACO for final edge detection, and the detected pixels of the complex region is subjected to the hiding process. The final complex regions are shown in Figure 5.8. The cover images displayed in Figure 5.3 are used one by one for hiding the secret information in the final edges. The secret message bits are hidden, 3-bits per pixel, in the pixels belonging to the complex region 1_{st} LSB of each of these pixels is set to 0. The 1_{st} LSB all the pixels that do not belong to the complex region is set to 1. The obtained stego images are shown in Figure 5.9a-f. The quality of the stego image is quite high, and no visible distortion is added by the hiding process. The stego images do not attract the attention of the unauthorized person. Hence the existence of hidden secret information is imperceptible to HVS.



(**a**) Cameraman (**b**) Lena (**c**) House

(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.8: Final complex region detected using Wave function.



(**a**) Cameraman (**b**) Lena (**c**) House

(**d**) Pepper (**e**) Mandrill (**f**) Tree

Figure 5.9: Stego Images of ACO based data hiding in edges using Wave function ref to Eq. 5.8.

To quantitatively analyze the technique using the Wave function, the HC, $PSNR$, and $SSIM$ are calculated in each experiment using each of the cover images one by one. The resulting values of HC, $SSIM$ and $PSNR$ are shown in Table 5.4.

Table 5.4: $SSIM$, $PSNR$ and HC with Wave function.

Cover Image	$SSIM$	$PSNR(dB)$	$HC(\%)$
Cameraman	0.9975	44.5692	0.6279
Lena	0.9977	43.147	0.2396
House	0.9978	42.7577	0.5556
Pepper	0.9977	42.9357	0.2955
Mandrill	0.9915	37.6869	0.2472
Tree	0.9985	39.0712	0.8657

The results show that for all previously mentioned cover images, the resulting stego images have a $PSNR$ of $37.6869dB$ and higher, which is much higher than

$30dB$, the minimum acceptable values for a stego image. The $SSIM$ obtained for different stego images is significant and results in value 0.9915 and greater for different cover images. The maximum HC, i.e., 0.8657%, is obtained for the image Tree.

The results show that all the ACO based data hiding in the complex region results in significantly higher quality stego images. However, Flat function in Eq. 5.5 and Sine function in Eq. 5.7, are very efficient both in terms of hiding capacity and stego image quality.

5.4 Comparison

The experimental results presented in Section 5.3, show the ACO based data hiding in complex region pixels results in significantly high quality stego images with $PSNR$ higher than $30dB$. The hiding technique uses the four functions to detect complex region in the cover image, hide information in the edges. The proposed technique hides information efficiently. The hidden information does not create any detectable or perceivable distortion in the stego images. Along with high quality results, it is important to compare the technique with other data hiding techniques, here, this section is dedicated for this purpose. The ACO based data hiding in complex region technique is compared with the techniques of Honsinger et al. [46], Macq and Dewey [88], Fridrich et al. [28], Goljan et al. [36], Vleeschouwer et al. [119], and Khan et al. [72]. All the techniques are used to hide the same message in cover images of Lena and Mandrill. The quality measuring parameter $PSNR$ and hiding capacity HC are computed for each the algorithms are listed in Table 5.5. Here, it is important to mention that the results of ACO based data hiding in the complex region are obtained by using the Flat function as expressed in Eq. 5.5.

Table 5.5: The Comparison of proposed technique with other data hiding techniques.

Technique	Lena			Mandrill		
	$HC(bpp)$	$PSNR$	$SSIM$	$HC(bpp)$	$PSNR$	$SSIM$
Honsinger et al.	<0.0156	-	-	<0.0156	-	-
Macq and Dewey	0.0325	48.75	0.9754	0.12	48.34	0.9963
Fridrich et al.	0.0156	-	-	0.0156	-	-
Goljan et al.	0.36	39.00	0.9915	0.44	39.00	0.9871
Vleeschouwer et al.	0.0156	30.00	0.8662	0.0156	29.00	0.8469
Khan et al.	0.33	46.23	0.8771	0.669	44.12	0.9508
Proposed Work	0.0310	42.95	0.9976	0.0211	37.64	0.9912

The results obtained show that the $PSNR$ of the technique presented in this

chapter is higher than the $PSNR$ of other data hiding algorithms, except the methods of Goljan et al. and Khan et al. The Goljan et al. has lower $PSNR$ than the proposed technique in case of Lena image, while for Mandrill image the $PSNR$ of the proposed algorithm is lower than the Goljan et al. method. Khan et al. has higher $PSNR$ than the proposed algorithm fro both Lena and Mandrill cover images. The comparison made based on $SSIM$ shows that the proposed technique performs better than the rest of the techniques on Lena's image. While using Mandrill image as cover, the proposed algorithm results in higher $SSIM$ than all the techniques except Macq and Dewey's technique, which has higher $SSIM$ than the proposed algorithm. The HC of the presented method on both images is higher than the methods of Hosinger et al., Fridrich et al. and Vleeschouwer et al. While, using Lena as cover image, the proposed technique has a HC comparable to the Macq and Dewey method and less than Goljan et al. and Khan et al. methods. Similarly, using Mandrill's image as cover, the HC of the proposed technique is less than the HC of the Macq and Dewey, Goljan et al. and Khan et al. methods.

Chapter 6

Conclusions and Future Work

In this thesis we have proposed a collection of algorithms to perform image clustering using the camera fingerprints trying to address the problem of high computational cost and addressing the scenario in which the number of cameras is much larger than the number of images available for a single camera.

The first algorithm i.e., RCIC, we proposed, is a simple but efficient way of clustering images according to randomly selected reference fingerprints. The clustering process is further improved in FICFO, with the help of fingerprints ordering. The sorting of camera fingerprint made the clustering process faster and reduced the computational complexity. The RCIC and FICFO are implemented with the additional but optional step of attraction, which helps to improve the quality of clustering with a little additional computational cost. The clustering process is further modified in the CIC algorithm, and clustering is performed in two stages of raw clustering and fine clustering. The two stage process helped to reduce the computational cost even further. The final attempt to reduce the computational is made in the CFIC algorithm. The CFIC uses reduced and full fingerprints for clustering. The use of reduced fingerprints helped in the significant reduction of computational cost. The results obtained on the Dresden image database illustrate that the algorithms have a computational cost quite lower than state-of-the-art algorithms, with a comparable or even better performance, in some cases. The computational cost decreases gradually from the RCIC to the CFIC algorithm. The computational complexity per image, of the RCIC, RCIC-A, FICFO, FICFO-A, CIC and CFIC algorithms, decreases as the size of the image dataset increases, which proves the effectiveness of these algorithms for large scale clustering, especially the CFIC which has a significantly lower computational cost the state-of-the-art algorithms as well as the other algorithms proposed in this thesis. The presented clustering algorithms are also robust to the $NC \gg SC$ problem.

Hence, it can be concluded that the presented image clustering algorithms are very efficient to cluster small and large datasets, easy and hard datasets, symmetric and asymmetric datasets and datasets with $NC \gg SC$.

We tried to reduce the computational cost as lower as possible and also to solve the $NC \gg SC$ problem. But, nothing is full and final, there is always an open window for improvement. The computational cost can be improved further by using different fingerprints compression techniques without affecting the quality of camera fingerprints. New efficient ways can be devised to arrange fingerprints, which will help to reduce computational complexity. A step ahead, the techniques can be devised to increase the efficiency of camera fingerprints based video clustering and reducing the computation complexity of video clustering. A technique can be developed to detect the group of pictures (GOP) of images, detect I-frames, and ranked the I-frames based on goodness. The use of good I-frames for estimating camera fingerprint may possibly contribute significantly to improve the quality of clusters and reduce the computational complexity.

Nomenclature

Acronyms / Abbreviations

ACO Ant Colony Optimization

ARI Adjusted Rand Index

BCFIC Blind Camera Fingerprinting and Image Clustering

CCD Charge-Coupled Device

CFA Color Filter Array

CFIC Compressed Fingerprints based Image Clustering

CIC Canopy based Image Clustering

CLT Central Limit Theorem

CMOS Complementary Metal-Oxide Semiconductor

CRLB Cramer-Rao Lower Bound

FICFO Fast Image Clustering based on Fingerprints Ordering

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

FPN Fixed Pattern Noise

GAC Greedy Agglomerative Clustering

JPEG Joint Photographic Experts Group

LSB Least Significant Bits

LSIC Large Scale Image Clustering

MRF Markov Random Field

MSE Mean Square Error

NCC Normalized Cross Correlation

NUA Non-Unique Artifacts

PFA Probability of False Alarm

PNN Pairwise Nearest Neighbour

$PRNU$ Photo Response non Uniformity

PSD Power Spectral Sensity

$PSNR$ Peak Signal to Noise Ratio

$RCIC$ Reduced Complexity Image Clustering

$RCIC - A$ Reduced Complexity Image Clustering with Attraction

RI Rand Index

RLE Run Length Ecoding

SPN Sensor Pattern Noise

SSC Sparse Subspace Clustering

$SSIM$ Structural Similarity

$VLSB$ Variable Least Significant Bits

WGN White Gaussian Noise

GOP Group Of Pictures

Bibliography

[1] E. Abreu et al. “A new efficient approach for the removal of impulse noise from highly corrupted images”. In: *IEEE transactions on image processing* 5.6 (1996), pp. 1012–1025.

[2] D. Achlioptas. “Database-friendly random projections: Johnson Lindenstrauss with binary coins”. In: *Journal of computer and System Sciences*. 66.4 (2003), pp. 671–687.

[3] A. Amelio and C. Pizzuti. “Is Normalized Mutual Information a Fair Measure for Comparing Community Detection Methods?” In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. Paris, France. ACM, 2015, pp. 1584–1585.

[4] I. Amerini et al. “A sift-based forensic method for copy–move attack detection and transformation recovery”. In: *IEEE Transactions on Information Forensics and Security*. 6.3 (2011), pp. 1099–1110.

[5] A. Bartow. “Pornography, coercion, and copyright law 2.0”. In: *Vand. J. Ent. & Tech. L.* 10 (2007), p. 799.

[6] S. Bayram, H. T. Sencar, and N. Memon. “Efficient sensor fingerprint matching through fingerprint binarization”. In: *IEEE Trans. Inf. Forensics Security* 7.4 (2012), pp. 1404–1413.

[7] K. Biber. *Captive images: Race, crime, photography*. Routledge-Cavendish, 2007.

[8] G.J. Bloy. “Blind camera fingerprinting and image clustering”. In: *IEEE Transaction on Pattern Analysis and Machine Intelligence.* 30.3 (2008), pp. 532–534.

[9] L. Bondi et al. “Improving PRNU Compression Through Preprocessing, Quantization, and Coding”. In: *IEEE Trans. Inf. Forensics Security* 14.3 (2019), pp. 608–620.

[10] A. Bovik and T. Goodall. *Measurement of non-uniformity noise*. US Patent App. 15/758,631. Feb. 2019.

[11] Holst. Gerald C. “CCD arrays, camera, and displays”. In: *IEEE Trans. Inf. Forensics Security* (1998).

[12] R. Caldelli et al. “Fast image clustering of unknown source images”. In: *Proceedings of IEEE International Workshop on Information Forensics and Security.* IEEE. 2010, pp. 1–5.

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

[14] M. Chen et al. “Determining image origin and integrity using sensor noise”. In: *IEEE Trans. Inf. Forensics Security.* 3.1 (2008), pp. 74–90.

[15] K. Cohen. “Digital still camera forensics”. In: *Small scale digital device forensics Journal* 1.1 (2007), pp. 1–8.

[16] D. Cozzolino and L. Verdoliva. “Noiseprint: a CNN-based camera model fingerprint”. In: *IEEE Transactions on Information Forensics and Security* (2019).

[17] 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* 29.11 (2007), pp. 1944–1957.

[18] M. Dorigo and S. Thomas. *Ant Colony Optimization.* Cambridge: MIT Press, 2004.

[19] H.B. Duan. “Ant colony algorithms: theory and applications”. In: *Chinese Science* (2005).

[20] E. Elhamifar and R. Vidal. “Sparse subspace clustering”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE. 2009, pp. 2790–2797.

[21] W.H. Equitz. “A new vector quantization clustering algorithm”. In: *IEEE Transaction on Acoustics, Speech, and Signal Processing.* 37.10 (1989), pp. 1568–1575.

[22] M. Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *Kdd.* Vol. 96. 34. 1996, pp. 226–231.

[23] O.M. Fahmy. “An efficient clustering technique for cameras identification using sensor pattern noise”. In: *Proceedings of International Conference on Systems, Signals and Image Process.* IEEE. 2015, pp. 249–252.

[24] F.D. Fernandez et al. “Implementation of a non-linear CMOS and CCD focal plane array model in ASSET”. In: *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XXX.* Vol. 11001. International Society for Optics and Photonics. 2019, p. 110010C.

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

[26] J. Fridrich. "Digital Image Forensics Using Sensor Noise". In: *Signal Processing Magazine* 26.2 (2009), pp. 11–62.

[27] J. Fridrich. "Sensor defects in digital image forensic". In: *Digital Image Forensics*. Springer, 2013, pp. 179–218.

[28] J. Fridrich, M. Goljan, and R. Du. "Invertible authentication". In: *Security and Watermarking of Multimedia contents III*. Vol. 4314. International Society for Optics and Photonics. 2001, pp. 197–208.

[29] A. El Gamal et al. "Modeling and estimation of FPN components in CMOS image sensors". In: *Solid State Sensor Arrays: Development and Applications II*. Vol. 3301. International Society for Optics and Photonics. 1998, pp. 168–177.

[30] N.K. Gill, R. Garg, and E.A. Doegar. "A review paper on digital image forgery detection techniques". In: *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE. 2017, pp. 1–7.

[31] F. Gisolf et al. "Common source identification of images in large databases". In: *Forensic Science International.* 44 (2014), pp. 222–230.

[32] T. Gloe. "Forensic analysis of ordered data structures on the example of JPEG files". In: *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2012, pp. 139–144.

[33] T. Gloe and R. Böhme. "The 'Dresden image database' for benchmarking digital image forensics". In: *J. Digit. Forensic Pract.* 3.2-4 (2010), pp. 1584–1590.

[34] T. Gloe, S. Pfennig, and M. Kirchner. "Unexpected artefacts in PRNU based camera identification: A 'Dresden image database' case-study". In: *Proc. ACM Workshop Multi. Secur.* ACM. 2012, pp. 109–114.

[35] M. Goljan. "Digital camera identification from images-estimating false acceptance probability". In: *International workshop on digital watermarking*. Springer. 2008, pp. 454–468.

[36] M. Goljan, J.J. Fridrich, and R. Du. "Distortion-free data embedding for images". In: *International Workshop on Information Hiding*. Springer. 2001, pp. 27–41.

[37] M. Goljan et al. "Effect of compression on sensor-fingerprint based camera identification". In: *Electron. Imag.* 2016.8 (2016), pp. 1–10.

[38] N. Guan et al. "NeNMF: An optimal gradient method for nonnegative matrix factorization". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898.

[39] N. Guan et al. "Online nonnegative matrix factorization with robust stochastic approximation". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.7 (2012), pp. 1087–1099.

[40] S. Guha, R. Rastogi, and K. Shim. "CURE: an efficient clustering algorithm for large databases". In: *ACM Sigmod Record*. Vol. 27. 2. ACM. 1998, pp. 73–84.

[41] S. Guha, R. Rastogi, and K. Shim. "ROCK: A robust clustering algorithm for categorical attributes". In: *Information systems* 25.5 (2000), pp. 345–366.

[42] G.E. Healey and R. Kondepudy. "Radiometric CCD camera calibration and noise estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.3 (1994), pp. 267–276.

[43] W. Hoeffding, H. Robbins, et al. "The central limit theorem for dependent random variables". In: *Duke Mathematical Journal* 15.3 (1948), pp. 773–780.

[44] W. Hong and T.S. Chen. "A novel data embedding method using adaptive pixel pair matching". In: *IEEE transactions on information forensics and security* 7.1 (2011), pp. 176–184.

[45] W. Hong, T.S. Chen, and C.W. Shiu. "Reversible data hiding for high quality images using modification of prediction errors". In: *Journal of Systems and Software* 82.11 (2009), pp. 1833–1842.

[46] C.W. Honsinger et al. "Lossless recovery of an original image containing embedded data". In: (Aug. 2001).

[47] C.S. Hsu and S.F. Tu. "Probability-based tampering detection scheme for digital images". In: *Optics Communications* 283.9 (2010), pp. 1737–1743.

[48] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of classification.* 2 (1985), pp. 193–218.

[49] M. Huffman, D. Steinley, and M.J. Brusco. "A note on using the adjusted Rand index for link prediction in networks". In: *Social networks.* 42 (2015), pp. 72–79.

[50] M.A. Irfan, N. Ahmad, and S. Khan. "Analysis of Varying Least Significant Bits DCT and Spatial Domain Stegnography". In: *Sindh University Research Journal-SURJ (Science Series)* 46.3 (2014), pp. 301–306.

[51] V.I. Ivanov and J.S. Baras. "Authentication of area fingerprint scanners". In: *Pattern Recognition* 94 (2019), pp. 230–249.

[52] R. Jacobson et al. *Manual of Photography.* Routledge, 2013.

[53] A. K. Jain and R. C. Dubes. "Algorithms for clustering data". In: (1988).

[54] J.R. Janesick et al. *Scientific charge-coupled devices*. Vol. 117. SPIE press Bellingham, 2001.

[55] J.T. Janesick. "CCD arrays, camera, and displays". In: *IEEE Trans. Inf. Forensics Security* 117 (2001).

[56] X. Jin and J. Han. "Expectation maximization clustering". In: *Encyclopedia of Machine Learning*. Springer, 2011, pp. 382–383.

[57] N.F. Johnson and S. Jajodia. "Exploring steganography Seeing the unseen". In: *Computer* 31.2 (1998), pp. 26–34.

[58] K.H. Jung and K.Y Yoo. "Data hiding using edge detector for scalable images". In: *Multimedia tools and applications* 71.3 (2014), pp. 1455–1468.

[59] T. Kanungo et al. "An efficient k-means clustering algorithm: Analysis and implementation". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (2002), pp. 881–892.

[60] R.P. Kanwal and K.C. Liu. "A Taylor expansion approach for solving integral equations". In: *International Journal of Mathematical Education in Science and Technology* 20.3 (1989), pp. 411–414.

[61] A. Karaküçük and A.E. Dirik. "PRNU based source camera attribution for image sets anonymized with patch-match algorithm". In: *Digital Investigation* (2019).

[62] G. Karypis, E. H. Han, and V. Kumar. "Chameleon: Hierarchical clustering using dynamic modeling". In: *Computer* 32.8 (1999), pp. 68–75.

[63] S. Katzenbeisser and F.A.P. Petitcolas. "Digital Watermarking". In: *Artech House, London* (2000).

[64] S.M. Kay. *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[65] E. Kee, M.K. Johnson, and H. Farid. "Digital image authentication from JPEG headers". In: *IEEE transactions on information forensics and security* 6.3 (2011), pp. 1066–1075.

[66] D. Kennedy. "Editorial retraction". In: *Science.* 211.5759 (2006), pp. 335–335.

[67] R. Keys. "Cubic convolution interpolation for digital image processing". In: *IEEE Trans. Acoust., Speech, Signal Process.* 6.3 (9), pp. 1153–1160.

[68] S. Khan, N. Ahmad, and M. Wahid. "Varying index varying bits substitution algorithm for the implementation of VLSB steganography". In: *Journal of the Chinese Institute of Engineers* 39.1 (2016), pp. 101–109.

[69] S. Khan and T. Bianchi. "Fast Image Clustering Based on Camera Fingerprint Ordering". In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2019, pp. 766–771.

[70] S. Khan and T. Bianchi. "Reduced Complexity Image Clustering Based on Camera Fingerprints". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 2682–2688.

[71] S. Khan and M.H. Yousaf. "Implementation of VLSB Stegnography Using Modular Distance Technique". In: *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*. Springer, 2013, pp. 511–525.

[72] S. Khan et al. "A secure true edge based 4 least significant bits steganography". In: *2015 International Conference on Emerging Technologies (ICET)*. IEEE. 2015, pp. 1–4.

[73] S. Khan et al. "Enhanced stego block chaining (ESBC) for low bandwidth channels". In: *Security and Communication Networks* 9.18 (2016), pp. 6239–6247.

[74] M. Kirchner and T. Gloe. "Forensic camera model identification". In: *Handbook of Digital Forensics of Multimedia Data and Devices* (2015), pp. 329–374.

[75] E.Y. Lam. "Image restoration in digital photography". In: *IEEE Transactions on Consumer Electronics* 49.2 (2003), pp. 269–274.

[76] M. Langford. *Basic photography*. Routledge, 2013.

[77] C.T. Li. "Unsupervised classification of digital images using enhanced sensor pattern noise". In: *Proceedings of IEEE International Symposium on Circuits and Systems*. IEEE. 2010, pp. 3429–3432.

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

[79] C.T. Li and X. Lin. "A fast source-oriented image clustering method for digital forensics". In: *EURASIP Journal on Image and Video Processing*. 1 (2017), p. 69.

[80] P. Li, T. J. Hastie, and K. W. Church. "Very sparse random projections". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2006, pp. 287–296.

[81] X. Lin and C.T. Li. "Large-scale image clustering based on camera fingerprints". In: *IEEE Transactions on Information Forensics and Security*. 12.4 (2017), pp. 793–808.

[82] D. Litwiller. "Ccd vs. cmos". In: *Photonics spectra* 35.1 (2001), pp. 154–158.

[83] B.B. Liu et al. "On classification of source cameras: A graph based approach". In: *IEEE International Workshop on Information Forensics and Security*. IEEE. 2013, pp. 1–5.

[84] C.S. Lu and H.M. Liao. "Structural digital signature for image authentication: an incidental distortion resistant scheme". In: *IEEE transactions on multimedia* 5.2 (2003), pp. 161–173.

[85] W. Lu and Y.P. Tan. "Color filter array demosaicking: new method and performance measures". In: *IEEE transactions on image processing* 12.10 (2003), pp. 1194–1210.

[86] J Lukás, J Fridrich, and M Goljan. "Digital camera identification from sensor pattern noise". In: *IEEE Trans. Inf. Forensics Security* 1.2 (2006), pp. 205–214.

[87] Goljan M., J. Fridrich, and T. Filler. "Managing a large database of camera fingerprints". In: *Proc. SPIE* 7541 (2010), p. 754108.

[88] B. Macq and F. Dewey. "Trusted headers for medical images". In: *DFG VIII-D II Watermarking Workshop*. Vol. 10. Citeseer. 1999.

[89] F. Marra et al. "Blind PRNU-based Image Clustering for Source Identification". In: *IEEE Transactions on Information Forensics and Security* 12.9 (2017), pp. 2197–2211.

[90] A. McCallum, K. Nigam, and L. H. Ungar. "Efficient clustering of high-dimensional data sets with application to reference matching". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2000, pp. 169–178.

[91] S. Mendis, S.E. Kemeny, and E.R. Fossum. "CMOS active pixel image sensor". In: *IEEE transactions on Electron Devices* 41.3 (1994), pp. 452–453.

[92] G. Meynants, B. Dierickx, and D. Scheffer. "CMOS active pixel image sensor with CCD performance". In: *Advanced Focal Plane Arrays and Electronic Cameras II*. Vol. 3410. International Society for Optics and Photonics. 1998, pp. 68–76.

[93] J. Mielikainen. "LSB matching revisited". In: *IEEE signal processing letters* 13.5 (2006), pp. 285–287.

[94] A.D. Miller and W.K. Edwards. "Give and take: a study of consumer photo-sharing culture and practice". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2007, pp. 347–356.

[95] G.M. Milligan and M.C. Cooper. "A study of the comparability of external criteria for hierarchical cluster analysis". In: *Multivariate behavioral research* 21.4 (1986), pp. 441–458.

[96] K.M. Mohamad and M.M. Deris. "Visualization of JPEG metadata". In: *International Visual Informatics Conference*. Springer. 2009, pp. 543–550.

[97] F. Murtagh and P.Contreras. "Algorithms for hierarchical clustering: an overview". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), pp. 86–97.

[98] R. T. Ng and J. Han. "CLARANS: A method for clustering objects for spatial data mining". In: *IEEE transactions on knowledge and data engineering.* 14.5 (2002), pp. 1003–1016.

[99] T.T. Ng et al. "Passive-blind image forensics". In: *Multimedia security technologies for digital rights management*. Elsevier, 2006, pp. 383–412.

[100] H. Pearson. "Image manipulation: CSI: Cell biology". In: *Nature.* 434 (2005), pp. 952–953.

[101] Q.T. Phan, G. Boato, and F.G. De Natale. "Image Clustering by Source Camera via Sparse Representation". In: *Proceedings of the 2nd International Workshop on Multimedia Forensics and Security.* ACM. 2017, pp. 1–5.

[102] Q.T. 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 (2018), pp. 1902–1916.

[103] W.M. Rand. "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association.* 66.336 (1971), pp. 846–850.

[104] R. Rouhi et al. "Hybrid Clustering of Shared Images on Social Networks for Digital Forensics". In: *IEEE Access* 7 (2019), pp. 87288–87302.

[105] T. Salunke, P. Bhavsar, and S. Bodhe. "Source Camera Identification Using SPN with PRNU". In: *International Journal Of Emerging Technology and Computer Science* 4.2 (2019), pp. 61–65.

[106] H.T. Sencar and N. Memon. "Digital image forensics". In: *Counter-forensics: attacking image forensics* (2013), pp. 327–366.

[107] M. Sharifi, M. Fathy, and M.T. Mahmoudi. "A classified and comparative study of edge detection algorithms". In: *Proceedings. International conference on information technology: Coding and computing*. IEEE. 2002, pp. 117–120.

[108] P. Singh and R.S. Chadha. "A survey of digital watermarking techniques, applications and attacks". In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.9 (2013), pp. 165–175.

[109] M.C.P. de Souto et al. "A Comparison of External Clustering Evaluation Indices in the Context of Imbalanced Data Sets". In: *2012 Brazilian Symposium on Neural Networks*. Bellingham, Washington USA. 2012, pp. 49–54.

[110] C. Su and H. Kikuchi. *Information Security Practice and Experience: 14th International Conference, ISPEC 2018, Tokyo, Japan, September 25-27, 2018, Proceedings*. Vol. 11125. Springer, 2018.

[111] M.S. Subhedar and V.H. Mankar. "Current status and key issues in image steganography: A survey". In: *Computer science review* 13 (2014), pp. 95–113.

[112] M.D. Swanson, M. Kobayashi, and A.H. Tewfik. "Multimedia data-embedding and watermarking technologies". In: *Proceedings of the IEEE* 86.6 (1998), pp. 1064–1087.

[113] M.D. Swanson, B. Zhu, and A.H. Tewfik. "Transparent robust image watermarking". In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Vol. 3. IEEE. 1996, pp. 211–214.

[114] J. Tian, W. Yu, and S. Xie. "An ant colony optimization algorithm for image edge detection". In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 751–756.

[115] D. Valsesia et al. "Compressed fingerprint matching and camera identification via random projections". In: *IEEE Trans. Inf. Forensics Security* 10.7 (2015), pp. 1472–1485.

[116] L.G. Villalba, A.S. Orozco, and J.R. Corripio. "Smartphone image clustering". In: *Expert Systems with Applications*. 42 (2015), pp. 1927–1940.

[117] O.R. Vincent et al. "A descriptive algorithm for sobel image edge detection". In: *Proceedings of Informing Science & IT Education Conference (InSITE)*. Vol. 40. Informing Science Institute California. 2009, pp. 97–107.

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

[119] C.D. Vleeschouwer, J.E. Delaigle, and B. Macq. "Circular interpretation of histogram for reversible watermarking". In: *2001 IEEE Fourth Workshop on Multimedia Signal Processing (Cat. No. 01TH8564)*. IEEE. 2001, pp. 345–350.

[120] Z. Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[121] Z. Wang and A.C. Bovik. "A universal image quality index". In: *IEEE signal processing letters* 9.3 (2002), pp. 81–84.

[122] G. Xuan et al. "Distortionless data hiding based on integer wavelet transform". In: *Electronics Letters* 38.25 (2002), pp. 1646–1648.

[123] A. Yamashita et al. "Removal of adherent noises from images of dynamic scenes by using a pan-tilt camera". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566).* Vol. 1. IEEE. 2004, pp. 437–442.

[124] K.Y. Yeung and W.L. Ruzzu. "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data". In: *Bioinformatics.* 17.9 (2001), pp. 763–774.

[125] S.X. Yu and J. Shi. "Multiclass spectral clustering". In: *IEEE International Conference on Computer Vision.* IEEE. 2003, pp. 313–319.

[126] H.B. Zhang, C. Yang, and X.M. Quan. "Image authentication based on digital signature and semi-fragile watermarking". In: *Journal of Computer Science and Technology* 19.6 (2004), pp. 752–759.

[127] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: an efficient data clustering method for very large databases". In: *ACM Sigmod Record.* Vol. 25. 2. ACM. 1996, pp. 103–114.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.