h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Hochschule Darmstadt

Fachbereich Informatik

Efficient Biometric Identification in Large-Scale Palm Vein Databases

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M.Sc.)

vorgelegt von

Benedikt-Alexander Mokroß

725215

Referent:	Prof. Dr. Christoph Busch
Korreferent:	Prof. Dr. Christoph Wentzel
Korreferent:	MSc. Pawel Drozdowski

Ausgabedatum:	01.06.2017
Abgabedatum:	01.12.2017

Erkärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 01.12.2017 Benedikt-Alexander Mokroß

Abstract (English)

With the increasing number and size of biometric systems around the world, biometric databases are rapidly growing. Biometric systems use measurable and distinctive human characteristics for the purpose of automatised recognition or identity verification tasks. This thesis aims to contribute to the research in the area of biometric workload reduction in open-set identification scenarios in large-scale palm vein databases. The palm vein as a biometric characteristic is very difficult to obtain a copy from without notice of the individual and it records a steady growth in biometric market share. Therefore, it has been selected as the biometric characteristic for this project.

The main research in this thesis was carried out using a biometric indexing approach based on Bloom filters and binary search trees, which has already been successfully applied for the iris characteristic. To transform the extracted palm vein patterns in a Bloom filter compatible representation, an approach based on Fourier transformations has been chosen. This representation was introduced for the fingerprint characteristic using defined reference points as an input. A so-called feature extraction pipeline is proposed that extracts such reference points and yields them in a usable form for further processing. This system has shown an acceptable biometric performance and a high workload reduction, as well as showing a drastic speed improvement compared to conventional pattern affinity approaches.
In addition to the Bloom filter-indexing approach, a less complex indexing approach used was presented, merely employing the Spectral Minutiae Representation used, utilising binary search trees. The presented system adopts optimisation approaches presented for the Bloom filter binary search trees and achieves a higher biometric performance than the Bloom filter-indexing approach while sacrificing a small amount of workload reduction. To the best of the authors knowledge, this is the first study of such an indexing approach using the selected representation.

It was discovered that the high fuzziness of palm vein imaging impairs the biometric performance of the base representation and thus also impairs the biometric performance of the indexing approaches. Facing the acceptable but not excellent biometric performance, promising further research topics are presented, whereby their recorded biometric performance results show promise and open new possible avenues of research in this area.

Abstract (German)

Mit der international steigenden Anzahl an biometrischen Systemen, deren immer größeren ausmaßen und deren zunehmenden Einsatzbereichen, resultierend aus zunehmender Akzeptanz und zunehmenden Vertrauen in die Biometrie, steigt auch die Anzahl an immer größeren, biometrischen Datenbanken. Biometrische Systeme nutzen mess- und bestimmbare Eigenschaften bzw. Charakteristiken des menschlichen Körpers um dessen Identität zu bestimmen oder zu verifizieren. Der Betrieb solcher enormen biometrischen Systeme benötigt außerordentlich starke Rechenkapazitäten um eine für die Benutzer akzeptable Zugriffszeit zu gewährleisten. Deswegen zielt diese Thesis darauf ab, einen Teil zu der Forschung im Gebiet der Effizienzsteigerung für biometrische Systeme im Identifikationsbetrieb beizutragen. Als zentrale biometrisches Charakteristikum wurde die Handvene ausgewählt, da diese unter anderem sehr schwer unbeobachtet zu erfassen ist und der Marktanteil an Handvenensystemen stetig wächst.

Die Kernthemen der Forschung wurden mittels eines biometrischen Indizierungsverfahrens auf Basis von Bloom Filtern und binären Suchbäumen analysiert, welches bereits erfolgreich für die Iris getestet wurde. Um die Handvenenstruktur in eine Bloom Filter kompatible Darstellung zu überführen, wurde eine für Fingerabdrücke entworfene Darstellung auf Basis von Fouriertransformationen ausgewählt. Für diesen Schritt wird eine eigens entworfene Bildverarbeitungspipeline vorgestellt, die die Handvenenstruktur in ein für die Fouriertransformationen geeigente Darstellung konvertiert. Die erziehlte Erkennungsleistung in den Ergebnissen ist akzeptabel und das System weist eine hohe Effizienz auf.
Neben der Indizierung mittels Bloom Filtern wurde eine weniger komplexe Indizierungsmethode vorgestellt, welche sich nur auf die Darstellung nach den Fouriertransformationen und binären Suchbäumen stützt. Die vorgestellte Methode nutzt Optimierungen im Suchvorgang, wie sie im Bloom Filter System eingesetzt werden, und erziehlt dabei eine bessere Erkennungsleistung bei minimal niedrigerer Effizienz als das Bloom Filter System. Diese Methode ist nach besten Wissen und Gewissen erstmalig in dieser Thesis vorgestellt und erprobt worden.

Es stellte sich heraus, dass die Fouriertransformationen mit der Unschärfe der Handvenen nicht ideal funktioniert, was die Erkennungsleistung des Systems reduziert. Im Hinblick auf die akzeptable, jedoch nicht ideale, Erkennungsleistung wurden vielversprechende weiterführende Forschungsansätze vorgestellt. Die erfolgversprechenste Methode wurde dargestellt.

Acknowledgements

I would like to thank my supervisors Professor Christoph Busch and Professor Christoph Wentzel for making this project possible and for the excellent communication, guidance and advice throughout the entire project period. I am especially grateful to M.Sc. Pawel Drozdowski for the day-to-day supervision and feedback. I also thank Doctor Alexander W. Lenhardt for supplying me with hardware, office space, time and everything I needed during the entire project. I owe my gratitude to Myriam Hilfenhaus for her moral support and her effort to provide stability and harmony for me.

Portions of the research in this paper use the CASIA-MS-PalmprintV1 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA) as well as The Hong Kong Polytechnic University (PolyU) Multispectral Palmprint Database and the PUT Vein Database by the Poland Institute of Control and Information Engineering. The sources of the re-used images are attributed directly in the text. The data processing and visualisation in this project was done using the OpenCV software library.

Contents

Chapter 1

Introduction

The steadily growing interest in biometrics for two decades lies in its promising application in a vast range of disciplines. At the time of writing, biometrics are already reliably used as an alternative or extension to traditional knowledge and token-based access control systems, identity documents and forensic sciences. While the biometric market value in 2015 was estimated at approximately 14 billion USD [Tra14], more recent studies predict market values from approximately 35 billion USD by 2022 [Cre16] up to 70 billion USD by 2025 [Tra17]. One of many catalyst for the rapid market value increase is government-driven, large-scale biomteric deployments like the Indian AADHAAR project [Ind], which aims to enroll the entire Indian population of 1.3 billion individuals and already has enrolled over 1 billion subjects, as well as several immigration programmes. The operation of such large deployments yields immense computational load. Up-scaling the hardware in terms of computing power quickly reaches certain limits. Therefore, the underlying systems software needs to implement efficient strategies to reduce its computational load. Traditional indexing or classification solutions are ill-suited: the fuzziness of the biometric data does not allow for naïve hashing or comparison methods. This matter is the key motivation and the main focus of this thesis.

One emerging biometric characteristic, that steadily increases market share[1] and popularity is the vascular (blood vessels) pattern in several human body parts. Wrist and back-of-hand vessels hold the most interest, since they are intuitive to use for users and feature several advantageous properties. While several biometric modalities like the fingerprint [CFM11] and iris [DRB17] are already covered by workload reduction research, the (palm-) vein characteristic lacks approaches for such computational load reduction. Wrist, finger and back-of-hand vascular biometrics are already commonly covered in several research topics, whereas the palm vein is often passed or ignored and replaced by palm print biometrics. Therefore, this thesis shall address the palm vein characteristic with a focus on the biometric identification scenario.

[1] 2014[Tra14], 2016 [Cre16], 2017[Tra17]

1.1 Thesis Contribution

This thesis contributes to current palm vein and minutiae-based biometric research by the following means:

- A state-of-the-art survey of existing approaches within minutiae-based biometric workload reduction.
- A proposal of a palm vein signal processing subsystem for minutiae-based vascular biometric systems based on well-known and optimized algorithms of various image processing disciplines.
- A proposal on how to extract quality information for vascular data.
- Implementation and open-set evaluation of the proposed palm vein signal processing subsystem.
- Re-implementation and open-set evaluation of the Bloom filter and binary tree-based biometric workload reduction.
- Proposal, implementation and open-set evaluation of a less complex (compared to the Bloom filter-indexing approach) biometric workload reduction approach using binary trees for all minutiae-based biometric systems.

1.2 Thesis Organisation

This document is organised as follows:

- Chapters 2 and 3 aim to introduce the fundamentals of biometric systems with extension to the vascular biometric characteristic and relevant related work.
- Chapter 4 proposes a palm vein signal processing subsystem capable of a robust feature extraction, even in palm-print environments with a high level of noise, yielding the Spectral Minutia Representation (SMR) templates outlined in chapter 5.
- Chapters 6 and 7 describe two biometric indexing approaches used in this project.
- Chapters 8 and 9 present the experimental setup and results.
- Finally, chapter 10 concludes the thesis with a discussion of the results with proposals for further research topics, followed by a summary of the achievements in this project in chapter 11.

Chapter 2

Biometric System Fundamentals

This chapter provides a general introduction to biometric systems with a focus on the vascular network (*rete venosum*), especially the vascular network of palm veins (*rete venosum dorsal palma manus*) as the biometric characteristic of choice.

2.1 Veins as a Biometric Characteristic

The *rete venosum* is the network of (connected) veins internal to the individual's body. Their main responsibility is to transfer deoxygenated blood from the tissues back to the heart. By contrast, arteries carry oxygenated blood away from the heart to the tissues. Figure 2.1 shows the typical layout and layers of veins and arteries below the skin (*epidermis*) with labels on the different types.

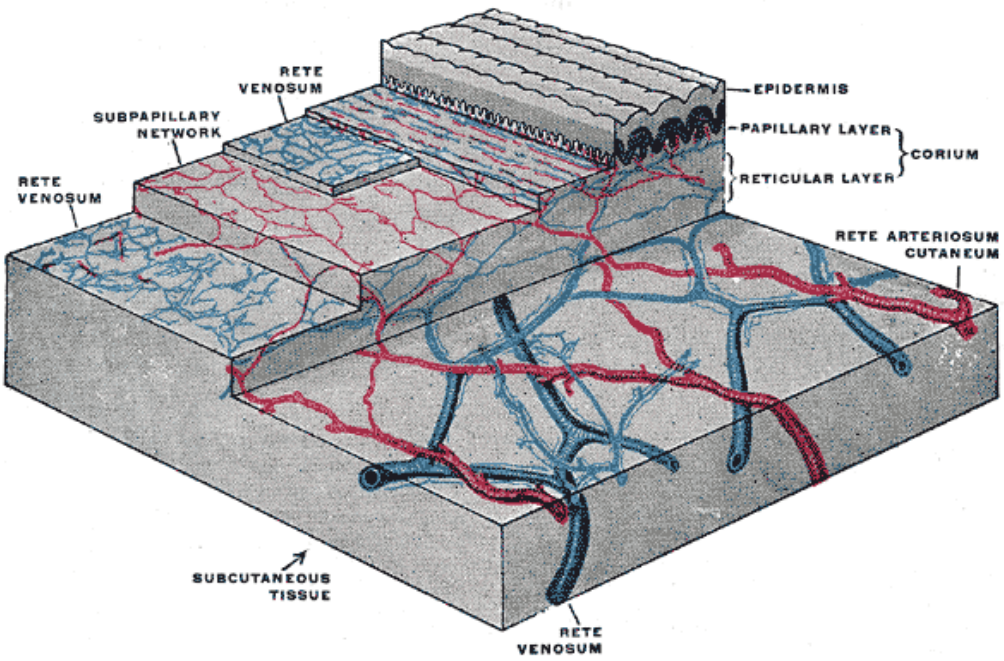
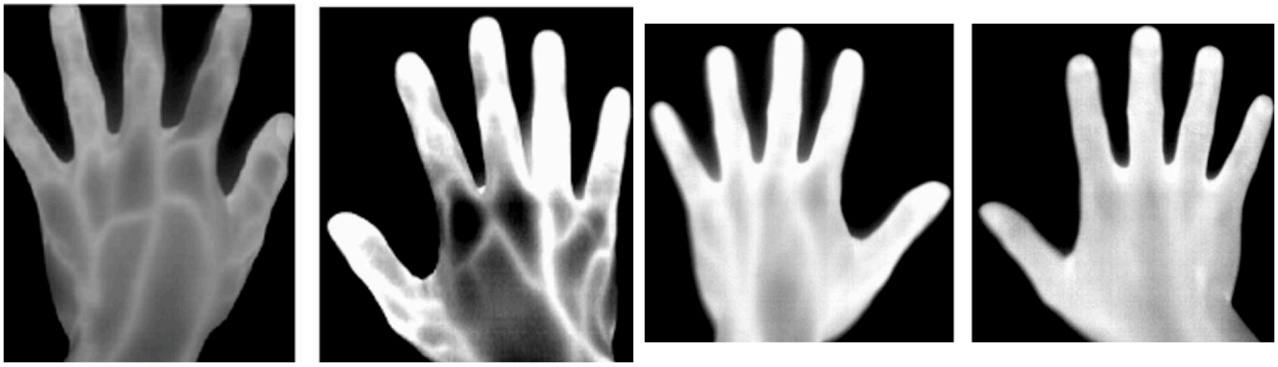


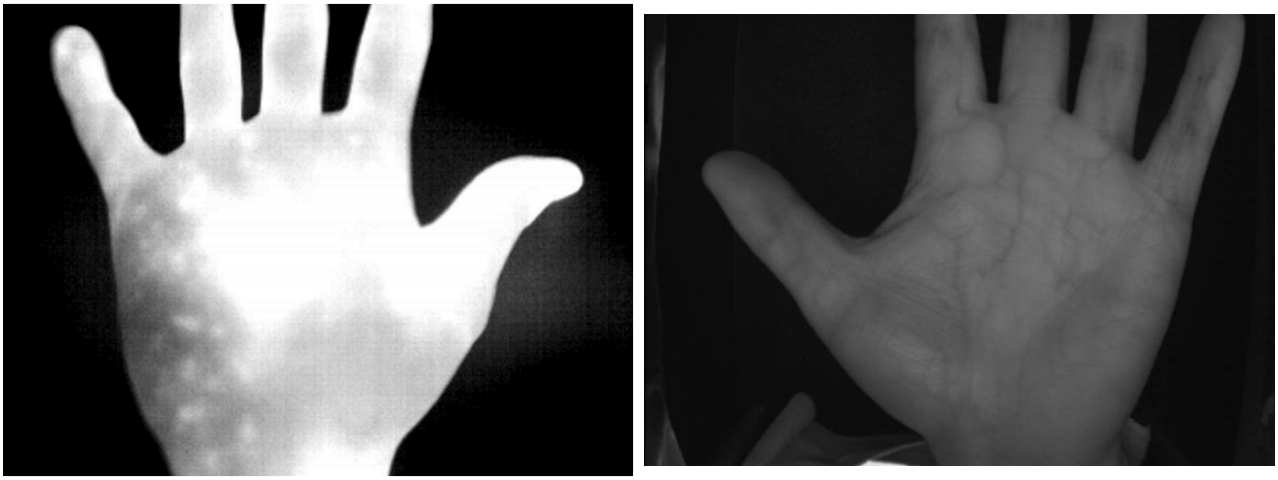
Figure 2.1: Distribution of the vessels in the skin (source: [GL18], figure 942).



(a) Office environment

(b) Outdoor environment

Figure 2.2: Example FIR images of the back of hand in different environments (taken from [WL06]).



(a) FIR

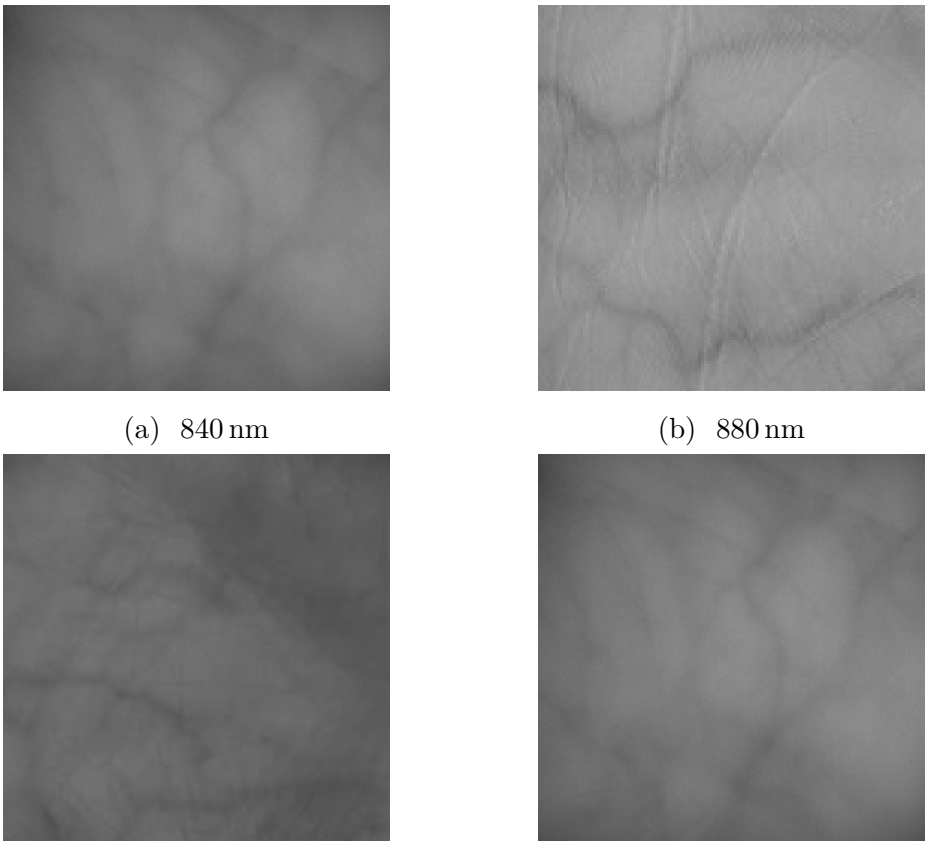
(b) NIR

Figure 2.3: Palm images using NIR and FIR imaging ((a) taken from [WL06]; (b) from [cas]).

The idea of using blood vessels for biometric authentication dates back when Joseph Rice submitted his patent for vascular technology authentication in September 1985 [Ric85]. Over time, many approaches in vascular imaging and authentication algorithms have been introduced, mainly using Far-Infrared (FIR) imaging in wavelengths 50 000 nm to 1 000 000 nm or Near-Infrared (NIR) imaging in wavelength 750 nm and 2000 nm.

FIR imaging of blood vessels uses heat radiation emitted from the vessels and thus it does not need external illumination. A downside of the FIR imaging approach is the high environmental impact, as shown in figure 2.2, which yields a very unstable image of the vascular pattern due to strongly varying contrast [Har12]. Furthermore, using FIR for palm vein imaging can lead to several problems, i.e. different skin thickness that result in different contrast levels in a single image, renders FIR imaging a much less trivial approach compared to NIR (compare figures 2.3a and 2.3b) imaging due to different heat levels. Therefore, this thesis will focus on approaches and algorithms using NIR images.

Current NIR vessels imaging devices capture the superficial blood vessels using an NIR-enabled



(a) 840 nm (b) 880 nm

(c) 940 nm (d) 940 nm

Figure 2.4: Example images from palms captured with some NIR enabled devices with illumination around 840 nm to 940 nm.

capturing device (e.g. CCD/CMOS sensor without an IR filter) while illuminating the Region of Interest (ROI) with light in a spectral band between 750 nm and 950 nm. These capturing approaches base on Lunnen's advice for medical photographers [Lun61]. He also mentions the absorption of various NIR wavelenghts by blood based on the level of oxygen in the haemoglobin. Due to their size, location and oxygen level, the veins of the superficial blood vessels are visible using the mentioned approach. These veins are visible as blurred[1] dark lines in the images.

To look upon the feasibility of using blood vessels, especially the palm veins, as a biometric characteristic the properties proposed in [BP98] will be analysed.

Universality - Is every individual able to present the biometric characteristic? Excluding congenital malformations, diseases or amputations, every individual has hands that contain blood vessels. There are no relevant statistics stating the percentage of individuals in ownership of at least one hand supplied with blood in the entire human population. Nevertheless, it is safe to consider the palm veins as a universal biometric characteristic.

[1]The NIR radiation scatters while penetrating deeper into the skin, which leads to blurred edges.

Distinctiveness - Is it likely, that two individuals are not distinguishable? Even though the development of blood vessels is not truly random, the difference between individuals is great. There are no observable identical patterns between siblings, ancestors or even the left and right hand of one individual [Nad07, Bad06]. In all three databases used in this thesis (see section 8.1) there were also not any similarities found. Therefore, it is a valid assumption that palm veins are a distinctive biometric characteristic.

Performance - How accurate is the biometric identification? Watanabe et al. (Fujitsu) state a False Acceptance Rate (FAR) of 0.01% and False Rejection Rate (FRR) of 0.00008% (at maximum) in [WESS05] for their PalmSecure biometric system in a verification scenario. However, these reports cannot be verified: the database of 140 000 palms and their approaches are not public. Further, a more transparent evaluation of the Fujitsu PalmSecure biometric system in [The06] reports a False Non Match Rate (FNMR) of 4.23% at $\approx 0.01\%$ False Match Rate (FMR). Other vascular modalities perform at a similar level. In [Nad07], Nadort presents an overview of several vascular modalities and their performance stated as FAR and FRR. Summarising, vascular biometrics perform at a very high level. Palm vein biometrics should be no exception.

Permanence - Does the characteristic alter in time? Like other body parts, the palm vein pattern stops developing, and thus changing, on average by the age of 20. Having implied no diseases, wounds or surgical interventions alter the vascular pattern, the only expected changes are shrinking and expansion of the vessels due to temperature and blood pressure. However, several publications report damages of the vascular cells of smoking individuals [Pit00, PGS+09].

Collectability - Is it easy to obtain a sample of the biometric characteristic? Acquiring a vascular image is fairly simple with cheap components using NIR-Imaging. However, there are several pitfalls in acquiring a high-quality vascular image for a high performance biometric system. A brief elaboration of recommendations for acquiring high-quality vascular images is outlined in section 4.1.

Acceptability - Do users accept the biometric system? The rising acceptance and trust in biometric systems also benefits vascular biometric systems, even thought vascular biometric systems are not as well-known as iris or fingerprint systems. Vascular imaging can be made contact-less, and thus hygienic, which also helps the acceptance. Several case studies by Fujitsu Limited report a high acceptance in high-security applications like ATMs [csp13b] and access control [csp13a, csp14].

Circumvention - How hard is it for a attacker to acquire a sample of the biometric characteristic or to get accepted by the biometric system? Allthought it is very difficult to receive a high-quality vascular image of an individual without their knowledge, it is still possible.

With some knowledge, it is possible to produce artefacts (i.e. persentation attack instruments) from low-quality vascular images or random artefacts for presentation attacks. Countermeasures for presentation attacks are liveness detection with e.g. the pulse.

Collectability and acceptability are two important properties, why the palm veins are of high interest. Other modalities using vascular patterns as characteristic show several drawbacks in those properties. One example of such modality are the facial veins. The facial veins are thinner, therefore require a image capturing system with a higher resolution to detect these thin veins. Further a individual may rather lift his hand in front of a sensor then his face.

Summarized, the palm vein is a promising biometric characteristic that mostly suffers from its lack of widespread deployment and lack of publicy-available high-quality databases (see section 4.1 and 8.1) for research tasks.

2.2 Generic Biometric System

The following subsections introduce a generic biometric system and its basic operational details, which can be used to describe a vast majority of biometric systems.

2.2.1 Workflow

ISO/IEC describe a basic workflow for a generalised biometric system in their standard on biometric performance testing and reporting [iso06]. This workflow, as shown in figure 2.5, can be applied to every biometric modality.

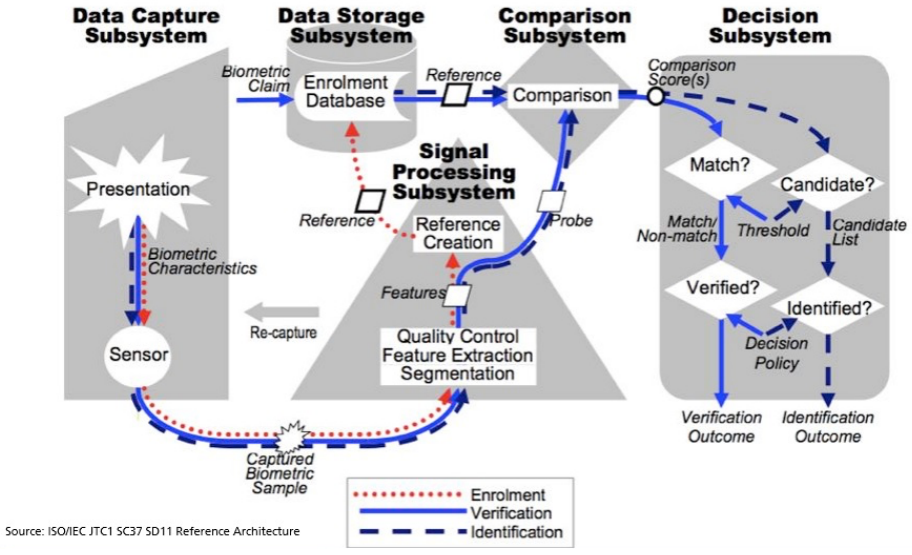


Figure 2.5: Generic biometric system (source: [iso06])

The five biometric subsystems described by ISO/IEC are briefly outlined below regarding to the palm veins as biometric characteristic.

Data Capture Subsystem An NIR-enabled imaging device (typically monochromatic), various NIR light sources and its hardware to capture the necessary information needed for the following subsystems (see section 2.1).

Signal Processing Subsystem A pipeline of algorithms is needed to extract the biometric data (feature set) and process it to an understandable model, the template, for the subsequent subsystems. This subsystem can further be detailed to the following processing steps:

Segmentation/ROI detection The image retrieved from the Data Capture Subsystem has to be segmented to the foreground and background. Subsequently, the ROI has to be found. This step is crucial for further processing steps, since it is necessary to always process the nearly exact portion of the palm to extract a stable biometric feature set.

Feature extraction After the ROI has been successfully found, the biometric feature set has to be extracted. For vascular biometric systems, this can be e.g. the whole vein pattern or specific features like bifurcations or endpoins, so-called minutiae, of the pattern. The requirement for the biometric feature set is to have a low intra-class variation (low variance in samples from the same individual) while maintaining a high inter-class variation (high variance in samples from different individuals).

Quality Control It is possible that the feature extraction fails to extract sufficient features due to poor image quality or a misslocated ROI. Automated or manual quality assessment can reject low-quality feature sets to maintain an overall good performance of the biometric system. In vascular biometric systems, it is possible to count the number of veins or minutiae found, or in an early step check for sufficient contrast in the ROI to reliably extract the vein pattern.

At the end of this pipeline, the Signal Processing subsystem returns a template created from the extracted biometric feature set. The template needs to address various privacy concerns, e.g. it should not be possible to translate the template back to an image of the original vessels [FZ05].

Data Storage Subsystem Once the data is processed into a template, it has to be stored so that the comparison subsystem can access it on demand. The Data Storage Subsystem is subject to several privacy concearns outlined in section 2.2.4.

Comparison Subsystem If a biometric system is queried with a request, it has to compare the biometric probe processed by the signal processing subsystem with the templates stored by the data storage subsystem. Various algorithms have emerged over time using all kind of different approaches and trade-offs. Most algorithms result in a probability, rated by a similarity or dissimilarity score.

Decision Subsystem Once all candidate templates are compared with the biometric probe, the decision subsystem has to evaluate whether a template and the probe can be mated and if there is a match in the candidate list or the biometric probe is non-matched.

The workflow also describes three paths in the biometric system: **(i)** Enrolment; **(ii)** Verification; and **(iii)** Identification;

Path (i) describes the process of creating a reference template, will be stored in the enrolment database. This reference template is stored inside the Data Storage Subsystem and later on used as a reference in comparison to a query. Paths (ii) and (iii) are subject to section 2.2.3.

2.2.2 Subsystems in this thesis

Through this thesis, implementations and approaches for the different subsystems described in 2.2.1, targeting the palm veins, are presented. To help the reader's overview of the different subsystems, the graph in figure 2.6 is used. Every node in this graph is expanded in its corres-

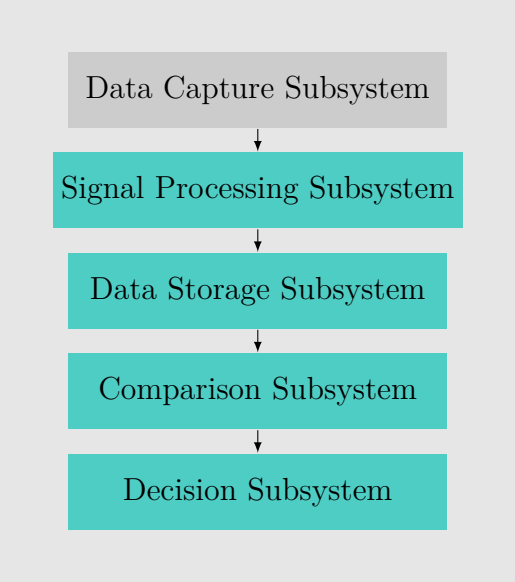


Figure 2.6: Collapsed subsystem-graph, based on [Har12].

ponding chapter to guide through the details (in the further course referenced as a pipeline). A concrete introduction in the Data Capture Subsystem is beyond the scope of this thesis and will be skipped. Refer to section 2.1 and 4.1 for a brief overview or [Har12] for a deeper introduction of the Data Capture process in (palm-)vein modalities.

2.2.3 Operation Modes

As already introduced in section 2.2, two more paths through the biometric system exist. The two leftover paths (ii) and (iii) are called operation modes. They describe the flow, selection and processing of information in the system and determine how the Decision Subsystem takes

a decision. In an abstract sense, they describe the difference between an $1:1$ ($O(1)$) and $1:\mathcal{S}$ ($O(\mathcal{S})$) biometric system.

Verification In verification mode, a biometric system confirms the identity claim of an individual. The individual provides a claim to an identity, e.g. with an username, RFID card or another attribute only assigned to him. With this claim, the biometric system only needs to compare the individual's query with the reference templates stored for the claimed identity. For palm vein biometric systems, this corresponds to an $O(1)$ (only one palm enrolled) or $O(2)$ (two palms enrolled per individual) comparison.

Identification In identification mode, a biometric system has no identity claim by an individual. The individual only presents his biometric characteristic and queries the biometric system with this probe. Now the biometric system has to confirm that the individual is enrolled and find his identity or deny the probe if the individual is not enrolled in the biometric system. For this purpose, in a naïve approach the biometric system has to compare the query template with all reference templates of $\mathcal{S}$ individuals. In the worst case, $O(\mathcal{S})$ (only one palm enrolled) or $O(2\mathcal{S})$ (two palms enrolled per individual) comparisons are needed to reach a decision.

Without rotation compensation (like pre-alignment of the probe or rotation invariant representations) of the biometric probes and references, the complexity rises by a factor of the number of rotations that have to be tested for both operation modes. While computational cost is a non-issue for biometric systems in verification mode, these costs are intolerable in large-scale identification mode biometric systems. The naïve approach also introduces another issue presented by [Dau00]. Daugman demonstrates the probability P_S of having at least one False Match (FM) in an identification scenario with equation (2.1). S denotes the number of enrolled subjects and P_1 the probability of a false match in a single identification transaction.

$$P_S = 1 - (1 - P_1)^S \tag{2.1}$$

Figure 2.7 shows the plot for equation (2.1) with a FMR of 1%, 0.1% and 0.01%.

Looking upon a biometric system with $S = 800$ enrollees, the probability of at least one false match at a FMR of 1% is $P_{800} \approx 99.9\%$. Even with a FMR of 0.1% the probability is still very high ($P_{800} \approx 55\%$).

It is clearly visible, that the probability of at least one false match quickly rises with the number of enrolled subjects when choosing the nïve approach. Without an reduction of template comparisons, large biometric systems quickly reach a point where they won't behave like expected: the system could fail to identify the correct user or, even worse, allows access to an unauthorized individual. While this is less an issue for very small biometric systems ($\mathcal{S} < 10$), larger systems need to reduce the number of template comparisons in a identification or Duplicate Enrolment Check (DEC) scenario to tackle computational workload and false positive

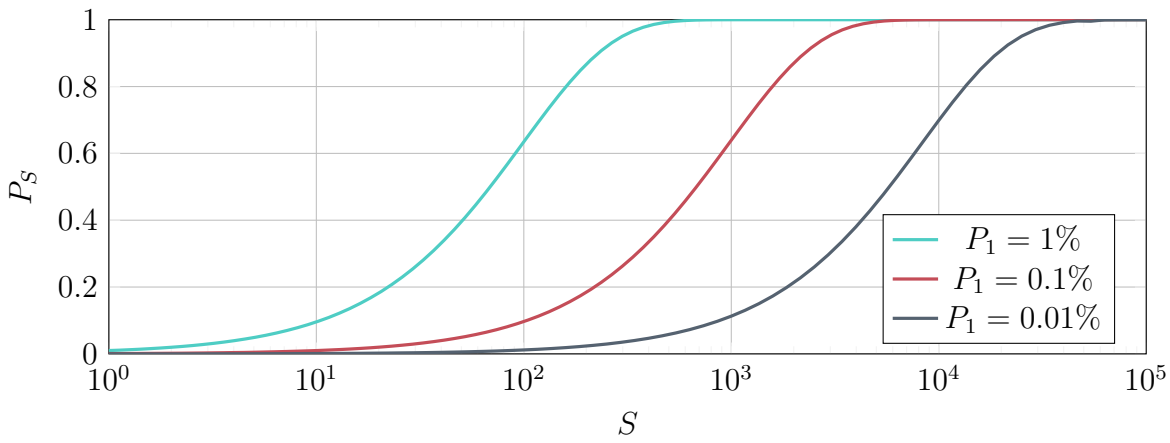


Figure 2.7: Plot of equation (2.1).

occurrences.

2.2.4 Template protection requirements

Biometric templates carry sensitive information from individuals, and thus they are subject to several privacy concerns, including the following:

Impersonation Biometric templates can be stolen or forged like passwords, thus enabling an attacker to generate an artefact (i.e. replica of the biometric characteristic) and several attack scenarios from a template.

Irrevocably Biometric characteristics cannot be changed.

Tracking With a stolen template, a attacker can track people's behaviour by cross-comparison of templates across several databases.

The ISO/IEC 24745:2011 (draft: [Bus10]) describes several requirements to protect templates to prevent these privacy concerns.

Renewability In the event of a compromised template, it should be possible to generate a new and different template from the same biometric characteristic.

Unlinkability The templates stored in different biometric databases and applications should not be linkable, e.g. to prevent tracking.

Irreversibility A template needs to be transformed by an asymmetric transformation in a different and comparable representation to prevent a reconstruction of the original biometric sample from a template while maintaining the ability to compare different templates.

Refer to [RU11] for a overview of some existing template protection schemes.

2.3 Chapter Conclusion

The vascular pattern of the humans palm is a promising biometric characteristic in the means of the seven properties universality, distinctiveness, performance, permanence, collectability, acceptability and circumvention. Biometric probes are collectable with NIR imaging when illuminating the palm in wavelengths of 750 nm to 950 nm. The vascular network appears as dark lines in NIR imaging.

Every biometric system can be generalised to the five subsystems data capturing, signal processing, data storage, comparison subsystem and decision subsystem operating in verification or identification mode. A biometric system has to address several privacy concerns; templates shouldn't be traceable across different databases, templates shouldn't be reversible to its biometric sample and it should be possible to renew templates in a way, that a renewed template isn't linkable to a old template of the same instance.

Section 2.2.3 introduced the at-least-one-false-match problem in identification scenarios, which also applies to the palm vein modality. Thus, it is required to employ a strategy to reduce the number of necessary template comparisons (workload reduction) for the palm vein modality. Workload reduction for palm vein modalities remains an insufficiently-researched topic. In the following chapter, selected state-of-the-art approaches for workload reduction and state-of-the-art representations of (palm) vein patterns will be presented, combined and expanded to find new strategies for workload reduction in palm vein environments.

Chapter 3

Related Work

This chapter describes current state-of-the-art and newly -introduced workload reduction approaches in palm vein identification scenarios, preceded by a brief discourse in palm vein feature representations. The chapter is organized in three sections: first, a generalized list of state-of-the-art types of palm vein representations is presented in section 3.1 and discussed. Second, a brief overview of types of workload reduction is presented in section 3.2, followed by a short literature survey for selected types in the corresponding subsections.

3.1 Vascular Biometric Systems

At the time of writing, there are two common features found in palm vein patterns.

Minutiae Points Vein minutiae points are nearly identical to fingerprint minutiae points: they mark endpoints, bifurcations and trifurcations of the vein patterns. Minutiae points can be described by their location and rotation.

Vein Strands The veins themselves can be used as features. Descriptors to veins are start- and endpoints, direction, curvature, angles or abstract figure.

Due to their almost identical characteristics to fingerprint minutiae, the vein minutiae are commonly used for vascular biometric systems. Substantial research effort has been devoted to fingerprint minutiae and a broad segment of the introduced approaches are applicable, but not necessarily feasible, to palm vein minutiae: recalling section 2.1, compared to fingerprint minutiae, the palm vein minutiae suffers from a higher noise in the capturing process due to the blurred edges of the veins and low contrast. Therefore, most fingerprint minutiae approaches need to be re-evaluated for the more fuzzy palm vein minutiae. A brief survey of different fingerprint minutiae approaches is presented in [Zae11].
Using minutiae as feature descriptor isn't as commonly used as using the full vascular pattern as feature descriptor. For example, patents [Wat14, JHF14, SKS+14] by Fujitsu Limited hint at the usage of the full palm vein vascular pattern as a biometric feature set in their comercial palm vein biometric system.

3.2 Workload Reduction

As already mentioned in 2.3, it is crucial to reduce the number of necessary template comparisons in a biometric system. The different approaches in workload reduction are categorized in three different types by [DRB17].

Serial combination of algorithms Instead of comparing every template with the query using the selected comparison algorithm, a less complex, respectively faster, and approximate algorithm is used to find a shortlist of most likely template matches. subsequently, only the templates in the shortlist are compared using a complex and more accurate algorithm.

Classification, Clustering and Binning Another two-step approach is to split the enrollee database into multiple subsets based on a easy-to-determine feature characteristic. On a query, the probe is classified and only the enrolled templates matching the class of the probe are compared with the probe using the actual comparison algorithm. Trivial class descriptors in palm vein templates are e.g. the race or left/right hand.

Indexing Workload reduction utilizing probabilistic or hierarchical data structures (e.g. trees). These approaches reduce the system load in terms of the big-O notation.

It has to be noted that even if the serial combination of algorithms and classification types appear to be the same (both types run a pre-selection step to select a shortlist of qualified templates), there is one crucial difference: while the serial combination of algorithms generates a tailored shortlist for each query from the entire database, the classification/binning approaches predefine shortlists of the database and select the right list for a query.

For the completeness sake, [DRB17] also lists hardware acceleration and parallelism as a category: disjoint parts of the enrollee database can be processed simultaneously on multiple CPUs/threads by using multi-core processors or GPUs. However, it is noted that the workload is not reduced but simply distributed so the problem of reducing the number of templates is not solved by these approaches. Further, [DRB17] divides the approaches in image encoding depended and independent approaches.
Biometric systems can use a combination of the presented categories. For example, preceding a paralleled indexing approach with an classification approach may further reduce workload in large-scale scenarios.
Although it appears obvious, it has to be noted that a workload reduction approach must not (significantly) impair the naïve biometric performance to be viable.
The following subsections provide a short list of examples for the different types of workload reduction and list some possible advantages and disadvantages. Note that most examples are for other modalities that use either veins as a biometric characteristic or use minutiae-points as

feature descriptor, since approaches explicitly targeting palm veins are not broadly researched or publicly available.

3.2.1 Serial combination of algorithms

All surveyed approaches that are stated as a serial combination of algorithms by the authors are in fact indexing or classification approaches, followed by a state-of-the-art biometric recognition approach.

3.2.2 Classification, Clustering and Binning

The advantage of the classification, binning and clustering approaches compared to the serial combination of algorithms is trivial: while a serial combination of algorithms has to inspect the whole database just-in-time, the predefined classes of a classification/clustering or the computed bins of a binning approach are generated off-line during the database construction. This leads to increased computational overhead while creating the database, although the overhead for each query is comparatively small. Frequently, classification and clustering are used as synonyms for each other. However, in data mining, classification (supervised) and clustering (unsupervised) are two different learning approaches. Classification approaches are trained with determined outcomes (i.e. pre-defined classes), while in clustering the algorithm has to determine the classes by itself.

A fairly new approach in workload reduction in terms of binning is presented by [ZLF+14]. By extracting an orientation matrix based on the Gaussian Radon transform using the Modified Finite Radon Transform (MFRAT) [JHZ08, ZK11], the principal direction features of a palm vein image is determined. Based on the angle of the principal direction, the template is classified in six bins (Bin_α), corresponding to the major directions 0°, 30°, 60°, 90°, 120° and 150°. Therefore, a probe that is classified as $Bin_{30°}$ only needs to be compared to the references that are classified as $Bin_{30°}$.
Another classification approach presented in [SRB15] uses clustering (K-Means [Llo82]) to form groups of similar finger-vein patterns. Thus a query first is compared to the centre points of each cluster and second the query is only compared to the templates stored in the most fitting cluster.

Disadvantages of a classification workload reduction approach include

the limited number of classes for each class descriptor: differing between left or right hand only results in two classes; using the skin colour may result in two to four classes, whereby even the introduced approach using the principal direction of the palm veins only features five classes. Thus, the expected workload reduction for classification is comparatively small. Furthermore, most classification approaches expect a evenly distributed number of

templates in every class. In most real-world scenarios, this is not the case, and thus the actual workload is much higher when one query is classified to a large bin.

the similar two-step problem like in serial combination of algorithms approaches: if a short list does not contain the correct reference template, the following steps would yield a non-match, thus lowering the genuine acceptance rate.

3.2.3 Indexing

Indexing on biometric data requires different approaches than common indexing algorithms due to the fuzziness: instead of querying with the same key, a similar key is queried and thus normal hashing approaches are not feasible. The retrieval performance of an indexing scheme most depends on its hashing functions, which need to fulfil the conditions that the hashed representation of two similar templates are also similar as well without losing too much inter-class variance. A good introduction[1] and a possible approach to the problem is presented by [HDZ08]. Most noteworthy disadvantages and trade-offs are the high offline (preprocessing) and additional storage costs.

An novel minutiae-based indexing approach is presented in [CFM11] using the Minutia Cylinder-Code (MCC) (presented in [CFM10]) minutiae representation. Initially targeting fingerprint minutiae, the design of the MCC offers promising characteristics for palm vein minutiae. The MCC is a fixed-radius local minutiae approach represented in a 3D cylindrical data structure. These cylinders are built from invariant distances and angles in a neighbourhood of each minutiae, without the type and quality.

The authors state, amongst others, three advantages that - projected to palm vein minutiae - deal with some of the negative characteristics of palm vein feature detection:

- The fixed-radius approach tolerates missing and spurious minutiae.
- Small feature extraction errors are tolerated due to the adoption of smoothing.
- Noisy regions with many spurious minutiae are dealt with a limiting function.

Indexing the MCC is achieved by linearising the cells of a cylinder corresponding to a given minutiae, thus receiving a vector of linearised cylinders. Each linearised cylinder is hashed by three different hashing functions. For each hash, the value (bucket) is stored. To generate a short list of candidate templates, the same procedure is applied to the query and the database templates with the highest number of bit collisions are selected.

[1] Another good publication for the awareness of the fuzzy data problem can be found in [Pro13].

3.3 Chapter conclusions

This chapter has introduced the different types of workload reduction and presented some current state-of-the-art biometric workload reduction approaches for minutiae- and vascular-based modalities. The three introduced types are serial combination of algorithms, classification/binning and indexing. For each presented approach, the basics were briefly outlined. Some presented approaches base on special representations. This is most likely found for indexing and serial combinations of algorithm types. Most classification and binning approaches do not rely on a special representation, and thus can be efficiently employed without additional storage requirements. This renders them a good choice as a first level for multi-level workload reduction environments.

Article	Type stated	Real type	Performance	Workload	Non-intrusive*
[CFM11]	Indexing	Indexing	—	—	No
[ZLF+14]	—	Classification	—	16.6%	Yes
[SRB15]	—	Classification (Clustering)	—	—	Yes

* Does not requires a special representation, thus easily combinable with other approaches.

Table 3.1: Summary of surveyed workload reduction approaches.

It has to be noted that many approaches labelled as a serial combination of algorithms turned out to be binning approaches, hence the relabelling in table 3.1.
Before presenting the two indexing workload reduction approaches used for the practical work of this thesis, the feature extraction pipeline and the feature representation will be presented in the next two chapters.

Chapter 4

Palm Vein Feature Detection

As shown in figure 4.1, the pipeline of a common Signal Processing Subsystem can be further separated into four different stages: quality control, pre-processing, feature extraction and post-processing. All approaches in this chapter belong to the pre-processing or feature extraction stage. The Signal Processing Subsystem in palm vein modalities is much more complex and

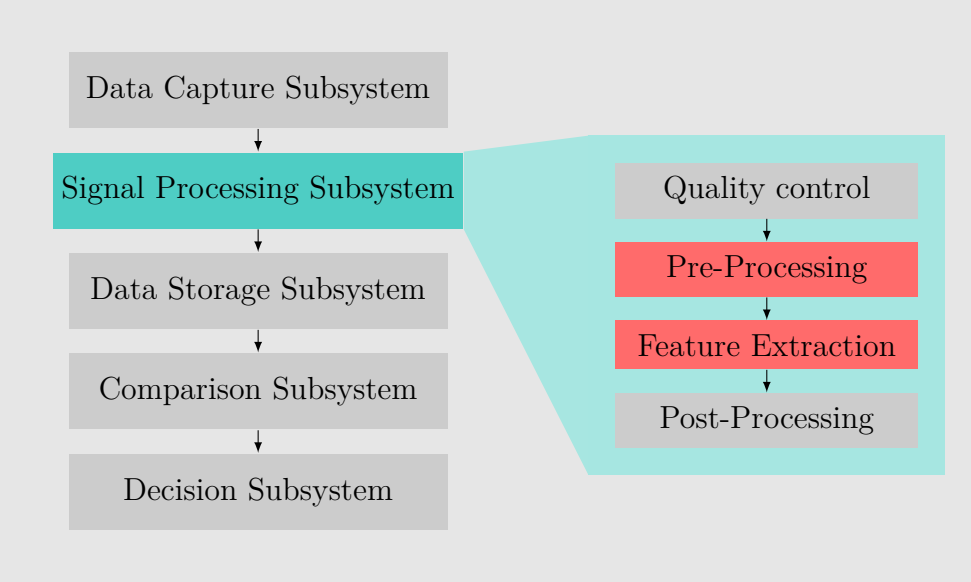
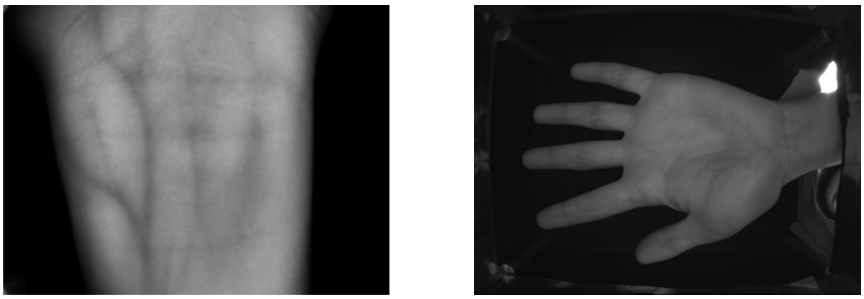


Figure 4.1: Signal Processing Subsystem

error-prone compared to other biometric modalities. Even compared to other vein modalities, there are some additional difficulties to address. Compare figures 4.2a and 4.2b. Note the inhomogeneous illumination of the ROI due to the geometry of the palm. Moreover, note the creases and strong skin valleys resulting in large amount of noise that can not be found in the skin texture of the wrist or the back of the hand. The lack of public available large-scale, high-quality palm vein databases requires to use databases built for other modalities like palm print (refer to section 8.1 for details) in experiments. Palm print databases purposely feature a high amount of skin details, and thus feature more noise than purpose-built palm vein databases, which aggravates the palm vein feature extraction (recall, figure 4.3a sample of a purpose-build



(a) (b)

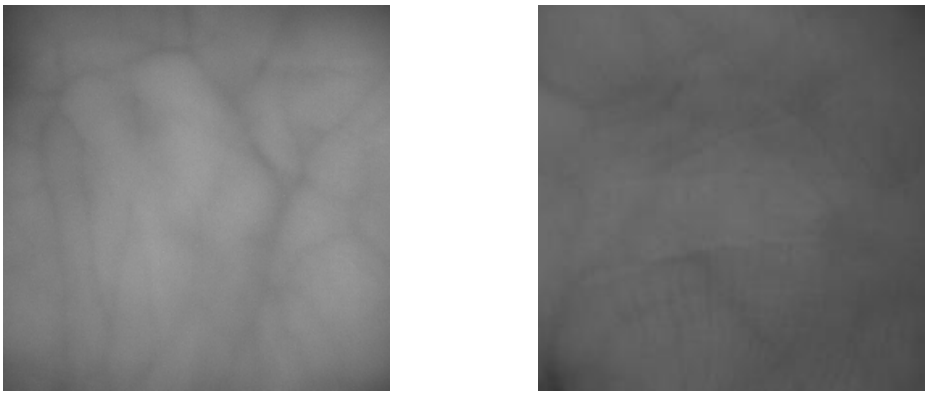
Figure 4.2: Raw samples of different vein-modalities found in public databases: (a) wrist [PUASRL10] and (b) palm [cas].

palm vein imaging device; figure 4.3b sample of a purpose-build palm-print imaging device).
A discussion of different approaches for low-quality vein images in [KK10] presupposes high contrast and little, homogeneous noise, and thus is not feasible for the fuzzy-noise environment of palm-print images. Further related work by [HOXB11, HOX+12] has already briefly discussed several approaches for pre-processing and feature extraction of other vein modalities with less noise. However, the lack of documentation details on these approaches exacerbates the reimplementation and replicability for further research.
Since the pre-processing and feature extraction steps are not trivial for a high-noise environment and important for the biometric performance of following subsystems, and given that most approaches are taken from other modalities, it is necessary to discuss the adaptations and the combination of these approaches in the following sections. The first following section provides a brief elaboration of recommendations for NIR imaging to increase the overall input quality.

4.1 Recommendations for high-quality vascular imaging

For high-quality images of the superficial blood vessels, a NIR-pass filter that blocks visible light is recommended to filter noise like reflections and skin details. Another recommendation for noiseless images is to illuminate the ROI with a diffuse, rather than specular, illumination source. Further, a very small back-focus (in respect to the skin surface) of the lens helps to further reduce noise. Figure 4.3 shows the difference in cleanliness of the captured veins when using diffuse (4.3a) and specular (4.3b) sources.

Observe the noise in figure 4.3b: The image features parts of the skin texture (mostly shadows in the texture valleys) and suffers from inhomogeneous illumination distribution due to specular light, which can lead to false detection of veins by the feature detection subsystem.



(a) (b)

Figure 4.3: Difference between diffuse and specular lightning of the palm ROI using reflection capturing: (a) diffuse NIR illumination (captured with a proprietary device); (b) specular NIR illumination (from [cas])

4.2 ROI Detection

In real-world contact- or guidance-less palm vein applications, a robust and accurate ROI determination is a crucial step in the pre-processing stage.
The single requirement of the introduced approach to the user is that he should spread[1] his fingers. It can safely presupposed that in high-security applications, individuals keep a high level of cooperation and are aware of requirements in interacting with the biometric system.
Starting with a prioritized watershed-flood filling - described in [BLM14] - for image segmentation, the palm and fingers are separated from the background and other noise using a simple mask shown in figure 4.4. Once the palm and fingers are filled (visualized in 4.5adg), the Border

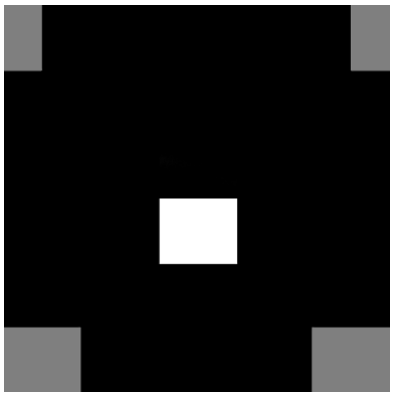


Figure 4.4: Mask used for watershedding, rectified (white = highest priority; black = lowest priority).

Following algorithm by Suzuki from [S+85] is used to retrieve the contour of the hand. Next, the convex hull of the hand is retrieved using the approach from [Skl82][2]. With the convex hull and

[1] Overspreading should be avoided to keep the palm geometry undistorted as possible.

[2] Note that [GY83] describes a more efficient implementation for finding the convex hull that could be used to replace the algorithm used.

the real contour of the hand, so-called convexity defects can be calculated. Convexity defects are gaps, thus the error, between the convex hull and the contour. They can now be used to find the finger gaps. In figure 4.5, the convex hull (purple) and the convexity defects (green) are visualized for three different samples in the second column. Finding the three finger gaps in a vector of convexity defects can be implemented according to the following assumptions: **(i)** the convexity defects of fingers are comparatively deep (large error); **(ii)** the convexity defects of interest have a narrow start and end point (start/end of error); **(iii)** convexity defects of fingers lay very close together; **(iv)** the center convexity defect (between the middle and ring finger) has the smallest distance to the other two finger convexity defects; and **(v)** when spanning a line between the two outer gaps, the middle gap has to be above the line, and thus the orientation of the hand can be determined. If the assumptions are implemented in this order, the three finger gaps found can easily be mapped to the left, middle and right finger gap. The reference points (shown coloured blue and cyan in the third column of figure 4.5) can now be used to span the ROI-rectangle: left and right mark the ROI's size with rotation and the middle reference point helps to decide where to move the rectangle. Now the ROI-center can be calculated: Let $P_{center} = (P_{left} + P_{right}) * 0.5$, Δ_P the distance between P_{left} and P_{right}, and α the angle of the axis spanned by P_{left} and P_{right} in respect to the X-axis, then $P_{ROIxy} = (P_{center_x} + \Delta_P * 0.8 * cos(\alpha), P_{center_y} + \Delta_P * 0.8 * sin(\alpha))$. The scalar 0.8 is used to reduce the translation, otherwise the ROI-center would translate too far away from the center of the palm. Around P_{center}, a rectangle with the edge size of 120% of Δ_P is created and rotated by α. These calculations result in the blue rectangles shown in the third column of figure 4.5. Note that there are two blue rectangles per palm: the enclosed area between the two rectangles is used to determine whether the calculated ROI is placed too close to the palms boundaries.

The introduced ROI determination approach proved robust and accurate on multiple samples and correctly framed[3] the ROI of 99.2% ($\sim$ 1190/1200) of the 940 nm samples contained in the CASIA-MS[cas] database. Another feature is its rotation invariance: the ROI is correctly determined independently of the image and the palm rotation.

4.3 Image Enhancement

The second stage in the pre-processing step involves various image enhancement operations. As already mentioned in the fundamentals chapter (see section 2.1), vascular imaging is very noisy and blurred. To best eliminate noise and receive a most clean and stable vascular image, the following combination of the Non-Local Means (NL-means) and the Non-Linear Diffusion (NL-diffusion) algorithms are used in the Signal Processing Subsystem.

[3]ROI centered in the palm, no background inside the ROI and at least 90% overlap of ground-truth and determined ROI.

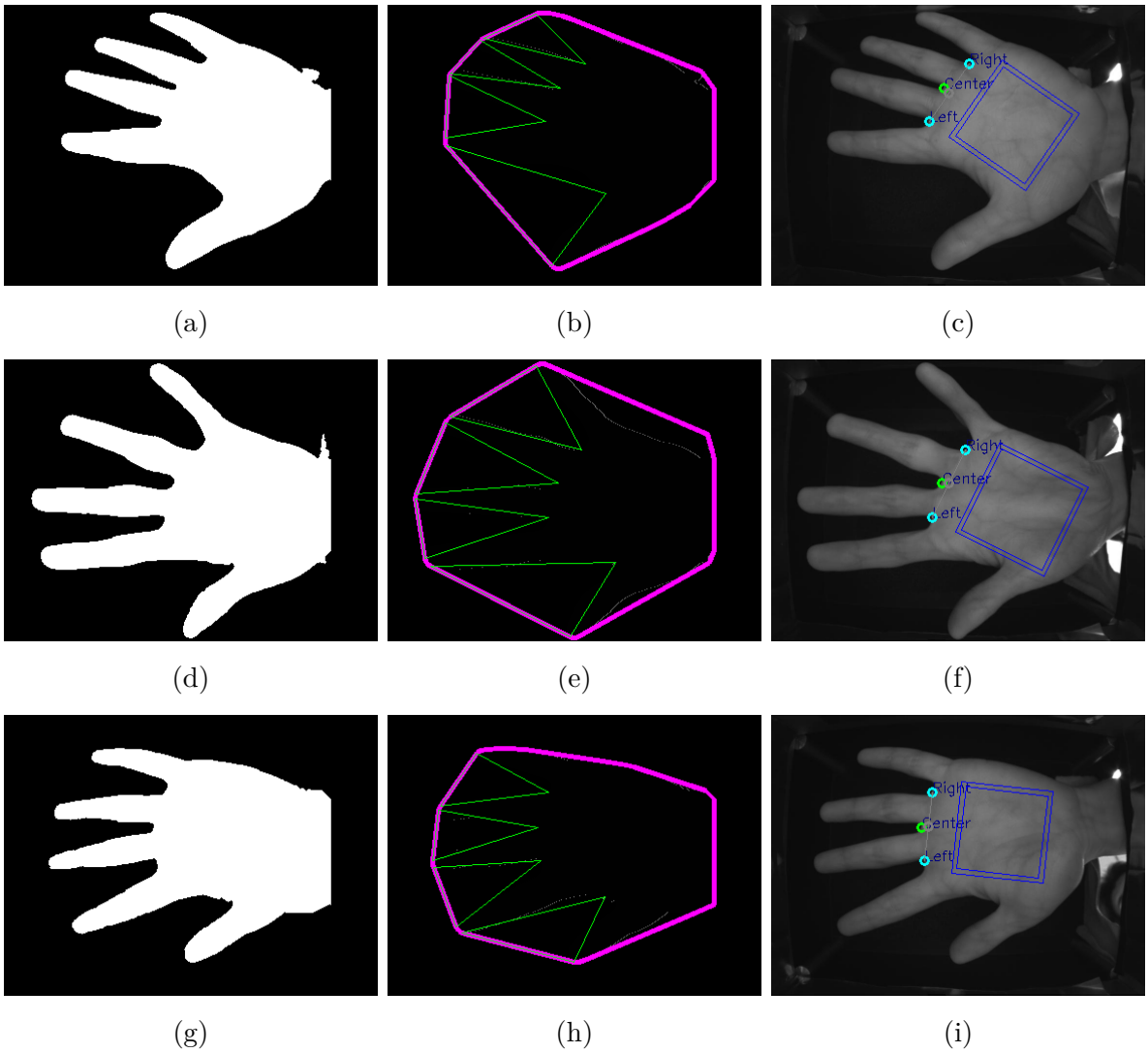


Figure 4.5: Visualisation of different intermediate data in the ROI finding process for samples from CASIA-MS[cas]: (a,d,g) hand seperated from background using priority-watershedding with mask shown in figure 4.4; (b,e,h) convex hull (purple) and convexity defects (green); (c,f,i) original image with projected ROI-rectangle (blue) and reference points (green, cyan, grey).

4.3.1 Non-Local Means

The first used algorithm is used to create an illumination invariant texture of the ROI by non-local smoothing. Initially presented for face recognition by [ŠP09], it was successfully used by [HOXB11, HOX+12] for vascular imaging. For ROI-sizes of about $128px$, a template window size of 7×7 and a search window size of 21×21 proved feasible in all experiments carried out in the curse of this project.

Figure 4.6b shows the output of the NL-means when a worst-case sample in terms of illumination variance shown in figure 4.6a is used as a input. Note the shadows in the input sample resulting from inhomogeneous illumination and a folded palm that are successfully removed.

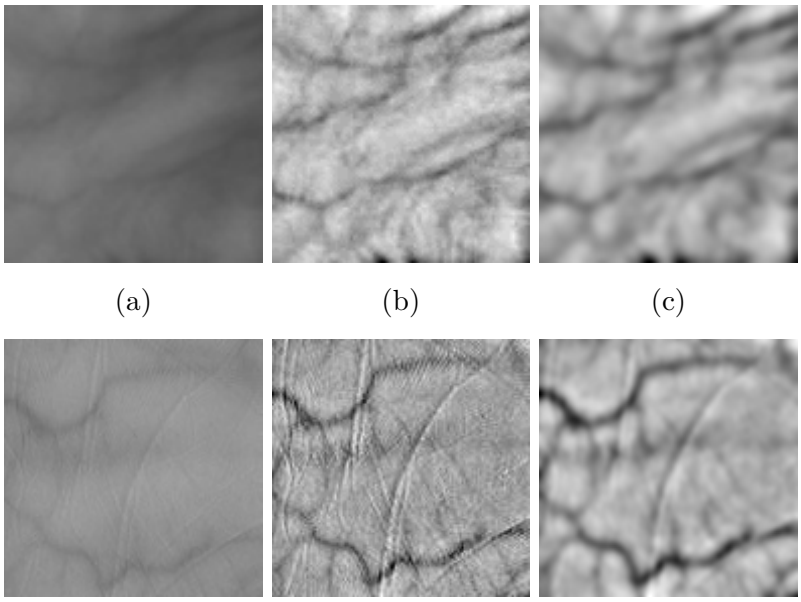


Figure 4.6: Output from single stages of the image enhancement process: (a) raw input (ROI from figure 4.5f, see section 4.2); (d) raw input (NIR from [pol]); (b,e) output of NL-means (inverted); (c,f) denoised image output of NL-diffusion after inversion.

Another property of the presented algorithm used is its ability to increase contrast on edges while removing illumination contrast, with a high decay parameter h. The decay parameter is used to control the decay of the Gaussian weighted Euclidian distance that serves as a similarity value between local neighbourhoods. Refer to section 3.1 of [ŠP09] for a in-depth description of the decay parameter.
The introduced contrast enhancement by this algorithm with $h = 30$ is sufficient for further steps. Thus, an additional contrast enhancement is not necessary. Furthermore, well-known contrast enhancement algorithms like CLAHE tend to increase contrast in non-vein areas, thus again introducing additional noise that can lead to falsely-detected veins by the following steps.

A disadvantage of this algorithm is its high computational complexity. However, the advantage of skipping further contrast enhancement with computational complex methods puts this disadvantage in perspective.

4.3.2 Non-Linear Diffusion

Following the NL-means, the NL-diffusion agorithm is used to remove high-frequency and some mid-frequency noise introduced by the skin texture. Noise removal is a well-researched topic. The selected approach introduced by [Wei01] features two feasible advantages for this application: **(i)** reduces high- and mid-frequency noise and **(ii)** preserves blurred edges. Advantage (i) is expected for a good noise-reduction algorithm while advantage (ii) can be considered an

additional feature.

For demonstration purposes, a very noisy sample was chosen in figure 4.6d. Observe how NL-diffusion removed the high-frequency and some mid-frequency noise in 4.6f after applying NL-means in figure 4.6e, while the veins are still clearly visible due to the ability to keep blurred edges. The NL-diffusion is configured with by the paramenter k, which describes the amount of diffusion iterations done.

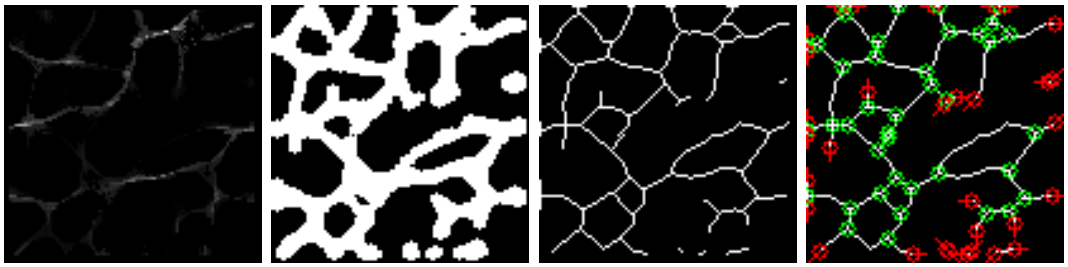
4.4 Vein Detection

After completing the pre-processing stages of finding the ROI and enhancing the cut ROI, the features can be extracted. Recall section 2.1, the features of interest (the veins) are the dark lines. The task to solve can be simplified to line tracking. While line tracking is a well researched topic, due to its application in a vast number of disciplines most approaches are not feasible for vein extraction: many approaches require pre-defined starting points, directions or patterns and depend on high-contrast edges. Neither of these requirements are fulfilled in palm vein patterns. With the introduced image enhancement pipeline, the contrast isn't high enough on edges of small veins for traditional line tracking approaches to extract a stable representation of the vascular pattern. Even with a high enough contrast, gaps, noise and the determination of starting-points are still a problem for line tracking approaches.

In [MNM04], a approach using simple line-tracking with random starting-points, trimmed for vascular images, is presented. However, the approach fails to reliably follow veins in low contrast areas and thick veins. The same authors present in [MNM07] a more robust vein-pattern extraction approach using maximum-curvature points. Simplified, the algorithm examines the ROI-image horizontally, vertically and diagonally line by line as a intensity graph[4] and converts it in a curvature representation. The curvature representations is now searched for local maxima. Every local maxima is scored with the probability that it is positioned on the centre of a vein, and thus a map of scores is created.

Using the enhanced image from figure 4.6c as an input for the maximum-curvature algorithm, the score map visualized in figure 4.7a is received. With a threshold function, the score map can be converted to a binary representation of the palm vein pattern. Even though the maximum-curvature algorithm tries to connect the vein centres inside the score map, experiments have shown that using a dilate-erode-dilate step helps to connect single gaps or closing small holes (see figure 4.7b).

[4]The very basic idea of [MNM07] is again picked up (three years later) by [KK10]. Neither [KK10] nor the following publication [KK11b] mention the approach by [MNM07].



(a) (b) (c) (d)

Figure 4.7: Output from single stages of the feature detection process with input from figure 4.6c: (a) score map retrieved with the maximum-curvature algorithm; (b) binarised and dilated score map; (c) thinning with algorithm of Guo and Hall; (d) detected minutiae (bifurcations/trifurcations green, endpoints red);

The response of the maximum-curvature algorithm can be trimmed with its σ parameter that controls the kernel size of the algorithm. Reducing σ decreases the size of the window in which local maxima are searched, thus increasing the sensitivity. In the original publications a $\sigma = 8$ is used. Experiments showed an increased Genuine Acceptance Rate (GAR) for all used databases with $\sigma = 7$.

Another application of the maximum-curvature algorithm response is extracting quality information about the vascular pattern. Refer to section 5.7.1 for a detailed description of the quality extraction process.

4.5 Minutiae

With the palm vein pattern received from binarising the maximum-curvature score map, the minutiae are easily extractable. Instead of complex pattern searching, a much faster and more robust approach based on the skeletarised representation of the binarised score map can be used. The approach presented in [OHBL11] extracts the minutiae type and rotation in one image iteration with a single convolution kernel, where other approaches need multiple iterations and kernels.

To receive the skeleton (see fig. 4.7c) of the vein pattern (see fig. 4.7b), the algorithm introduced by [GH89] is used. The thinning algorithm was chosen because it best preserves the initial structure[Koc13] and is highly parallelisable.

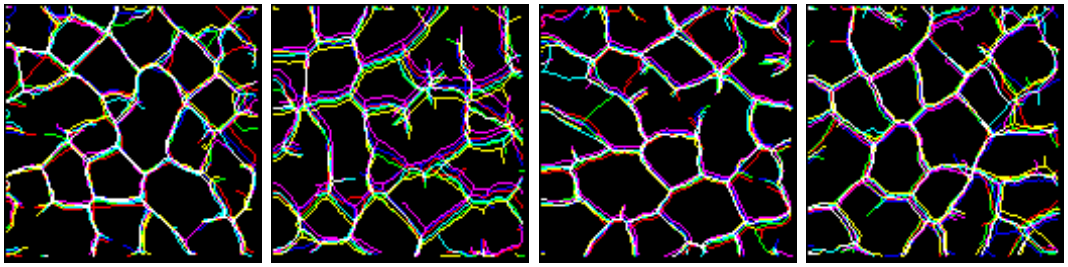
The kernel by [OHBL11] (see fig. 4.8) represents a 2D bitmask, thus applied to the binary skeleton image, every pixel is assigned with a value describing its neighbourhood. Minutiae (skeleton endpoints, bifurcations and trifurcations) and their rotations can be determined by special values in the filters response: e.g. Endpoints at rotation $0°, 45°, 90°, 135°, 180°, 225°, 270°$ and $315°$ (ccw, starting top-centre) are assigned a value of $258, 257, 384, 320, 288, 272, 264$ and

1	2	4
128	256	8
64	32	16

Figure 4.8: 2D-Kernel used by [OHBL11] to receive minutiae types and rotation in one iteration.

260. Figure 4.7d shows the minutiae found. Bifurcations and trifurcations are marked green, while endpoints are marked red with an additional marker following the direction of the endpoint.

4.6 Chapter Conclusions



(a) (b) (c) (d)

Figure 4.9: Pipeline applied on six samples of four palm instances selected from [pol] and rendered as one image. Each colour represents one sample. Colours: Red, green, blue, cyan, magenta, yellow.

In this chapter, the pipeline for the pre-processing and feature extraction stages of the Signal Processing Subsystem was presented. The presented pipeline is a robust basis for further algorithms: Figure 4.9 shows the combined output of six samples from four randomly-selected palm instances. Note the robust vein extraction even under the noisy conditions of a palm-print database. Most errors appear to be a result of different blood circulation due to heartbeat. Only some minor translation errors are observable.

Chapter 5

Spectral Minutiae Representation

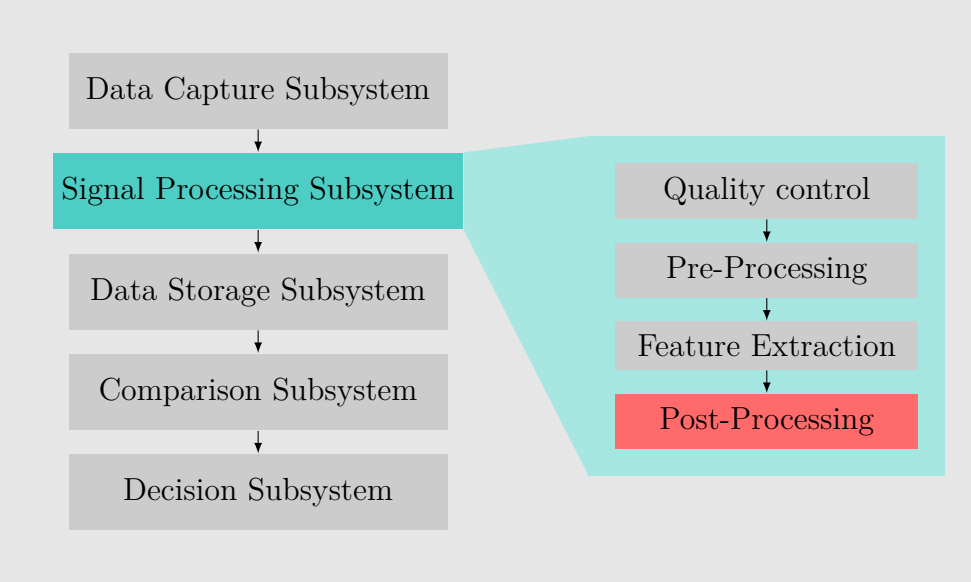


Figure 5.1: Signal Processing Subsystem

By now, the pre-processing and feature extraction stages have successfully extracted the palm vein minutiae for further processing. The following Comparison Subsystem and Data Storage Subsystem can use the raw minutiae vector as a template. However, this introduces several problems, starting with privacy concerns in terms of storing the raw biometric features and ending with computational drawbacks due to differently-sized feature-vector[1] sizes.
To address these issues, a post-processing stage can convert the raw minutiae to a fixed feature-size representation that should not be reversible to the minutiae. This chapter describes the approach that was investigated in this thesis. Readers already familiar with the SMR approach may only read section 5.7 and skip the rest of this chapter.

[1]Even probes of the same subject differ in their number of features.

5.1 Spectral Minutiae

Inspired by the Fourier-Mellin transform [CP76] used to obtain an translation, rotation and scaling invariant descriptor of an image, the SMR [XV10c] transforms a variable-sized minutiae feature vector in a fixed-length translation, and implicit rotation and scaling invariant spectral domain. To prevent the re-sampling and interpolation introduced by the Fourier transform and the polar-logarithmic mapping, the authors introduce a so-called *analytical* representation of the minutiae set and an so-called *analytical* expression of a continuous Fourier transform that can be evaluated on polar-logarithmic coordinates. According to the authors, the SMR meets the requirements for template protection and allows faster biometric comparisons.

To represent a minutiae in its analytical form, it has to be converted into a Dirac pulse to the spatial domain. Each Dirac pulse is described by the function $m_i(x,y) = \omega(x-x_i, y-y_i), i = 1,\ldots,Z$ where (x_i, y_i) represents the location of the i-th minutiae in the palm vein image. Now the Fourier transform of $m_i(x,y)$ is given by:

$$\mathscr{F}\{m_i(x,y)\} = exp(-\mathrm{j}(w_x x_i + w_y y_i)) \tag{5.1}$$

Based on this analytical representation, the authors introduced several types of spectral representations and improvements for their initial approach. The following four sections will cover the three main types and their sampling introduced in their corresponding publications. Subsequently, the introduced feature reduction approaches and the binarisation approach, followed by the comparison methods, are presented.

5.1.1 Spectral Minutiae Location Representation

The initial approach introduced in [XVK+08] is called the Spectral Minutia Location Representation (SML). It only uses the minutiaes location for the spectral representation:

$$\mathcal{M}(w_x, w_y) = \left| \sum_{i=1}^{Z} exp(-\mathrm{j}(w_x x_i + w_y y_i)) \right| \tag{5.2}$$

To compensate small errors in the minutiae location, a Gaussian low-pass filter is introduced by the authors. A 2-D Gaussian filter in the space domain is represented by

$$g(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{5.3}$$

and its Fourier transform

$$G(w_x, w_y) = exp\left(-\frac{w_x^2 + w_y^2}{2\sigma^{-2}}\right). \tag{5.4}$$

Thus, the magnitude of the smoothed SML with a fixed σ is defined as

$$\mathcal{M}(w_x, w_y; \sigma^2) = \left| exp\left(-\frac{w_x^2 + w_y^2}{2\sigma^{-2}}\right) \sum_{i=1}^{Z} exp(-\mathrm{j}(w_x x_i + w_y y_i)) \right| \tag{5.5}$$

in its analytical representation. By taking the magnitude, the translation invariant spectrum is received.

5.1.2 Spectral Minutiae Orientation Representation

Additional to the location, the orientation θ of a minutiae can be incorporated as an additional source of information to better describe a minutiae. This second Spectral Minutiae type was introduced in [XVB+09]. For the Spectral Minutia Orientation Representation (SMO), to every minutiae the function $m_i(x, y, \theta)$ is assigned, such that

$$\mathscr{F}\{m_i(x, y, \theta)\} = \mathrm{j}(w_x cos\theta_i + w_y sin\theta_i) * \exp(-\mathrm{j}(w_x x_i + w_y y_i)). \tag{5.6}$$

Applying the Gaussian smoothening and taking the magnitude yields

$$\mathcal{M}(w_x, w_y; \sigma^2) = \left| exp\left(-\frac{w_x^2 + w_y^2}{2\sigma^{-2}}\right) \sum_{i=1}^{Z} \mathrm{j}(w_x cos\theta_i + w_y sin\theta_i) * exp(-\mathrm{j}(w_x x_i + w_y y_i)) \right|. \tag{5.7}$$

However, the SMO did not show better results in experiments than the SML: the orientation is included as a derivative of the delta function, thus amplifying the noise in the higher frequencies. To compensate the higher noise, a higher σ is needed for the Gaussian kernel, therefore losing precision.

5.1.3 Spectral Minutiae Complex Representation

To tackle the disadvantages of the SMO approach, the Spectral Minutia Complex Representation (SMC) approach was introduced in [XV10b]. Instead of incorporating the orientation as derivative of the delta function, the orientation is assigned by a complex amplitude $e^{\mathrm{j}\theta_i}$. Thus, the magnitude of the Gaussian smoothed Fourier spectrum yields

$$\mathcal{M}(w_x, w_y; \sigma^2) = \left| exp\left(-\frac{w_x^2 + w_y^2}{2\sigma^{-2}}\right) \sum_{i=1}^{Z} exp(-\mathrm{j}(w_x x_i + w_y y_i) + \mathrm{j}\theta_i) \right|. \tag{5.8}$$

The authors achieve an overall better performance using the SMC compared to the SML even for the lesser-quality dataset.

5.2 Spectral Minutiae Sampling

As already mentioned in section 5.1, the different types of the SMR can be evaluated on a polar-logarithmic grid. When sampling the SMR on a polar-logarithmic or polar-linear grid, rotation of the minutiae become horizontal circular shifts. For this purpose, sampling of the continuous spectra 5.5(SML) or 5.7(SMO) is proposed by the authors using $M = 128$ in the radial direction λ logarithmically distributed between $\lambda_{min} = 0.1$ and $\lambda_{max} = 0.6$. The angular direction β for SML and SMO is proposed between $\beta = 0$ and $\beta = \pi$ in $N = 256$ uniformly distributed samples. A sampling between $\beta = 0$ and $\beta = \pi$ is sufficient due to the symmetry of the Fourier transform for real-valued functions.
The SMC features much more information in the higher frequencies. Instead of sampling λ of SMC on a logarithmically distributed grid, sampling on a linear grid is proposed. This provides

more samples in the higher frequency part. Furthermore, due to the additional complexity, the SMC is proposed to be sampled between $\beta = 0$ and $\beta = 2\pi$ in $N = 256$ uniformly distributed samples.

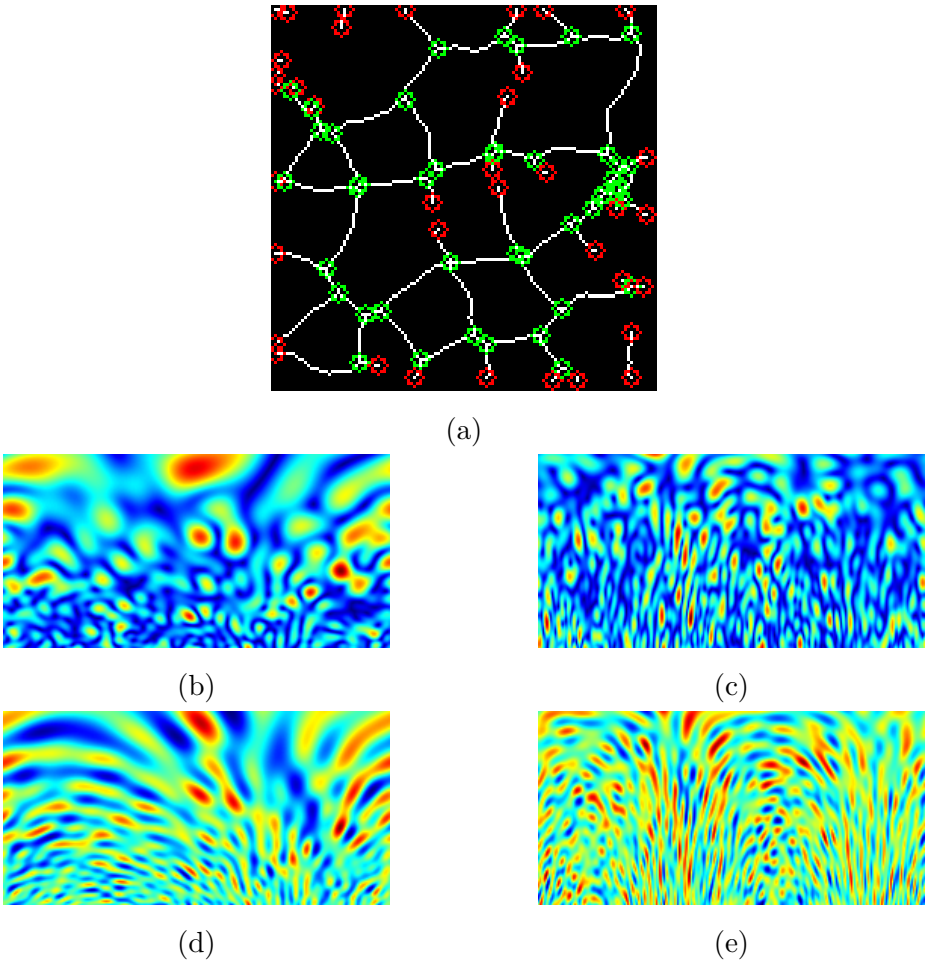


Figure 5.2: Illustration of the Spectral minutiae approach: (a) input minutiae visualised on the extracted vein pattern; (b) complex-modulus SML Fourier spectrum sampled on a polar-logarithmic grid; (c) complex-modulus SMC Fourier spectrum sampled on a polar-linear grid. (d) real-valued SML Fourier spectrum sampled on a polar-logarithmic grid; (e) real-valued SMC Fourier spectrum sampled on a polar-linear grid.

Refer to figure 5.2 for examples of sampling the minutiae input visualized in 5.2a with the different approaches: The magnitude of the polar-logarithmic SML is shown in figure 5.2b while the magnitude of the polar-linear SMC is shown in figure 5.2c. Figures 5.2d and 5.2e show the real-valued grids of the same SML and SMC. Real-valued SMR lose their translation invariance compared to the absolute-valued SMR, but are more stable on low-quality data with various spurious minutiae.

Further, it is possible to tune the SMR sampling to more stable, but less detailed configuration by altering λ_{min} and λ_{max} [XVK+08]. λ_{min} correspondents to the lower frequencies

of a SMR and λ_{max} to the higher frequencies. To receive a more stable SMR in a high noise environment, the λ_{max} should be lowered to values between 0.45 and 0.55. Altering λ_{min} should be avoided since all experiments that used a λ_{min} $/0.1$ achieved a notoriously bad biometric performance.

5.3 Normalisation

Since the SMR yields spectra with different energies depending on the number of minutiae per sample, each spectrum has to be normalised to reach zero mean and unit energy. To normalise the SMR, the following equation is used.

$$X = \frac{\mathcal{M} - \overline{\mathcal{M}}}{\sigma(\mathcal{M})^2} \tag{5.9}$$

By subtracting the mean ($\overline{\mathcal{M}}$) of the magnitude[2] ($\mathcal{M}$) and then dividing it by the standard deviation ($\sigma(\mathcal{M})^2$) of it, the resulting spectrum $\mathcal{X}$ is normalised to zero mean and unit energy.

5.4 Feature Reduction

Sampling the spectra on a $N = 256$ and $M = 128$ grid yields a $N \times M = 32768$ sized feature vector. This large-scale feature vector introduces two drawbacks:

Storage Considering $N \times M = 32768$ double-precision float (64bit) values, each template would take $2097152bit = 256kB$ RAM or data storage.

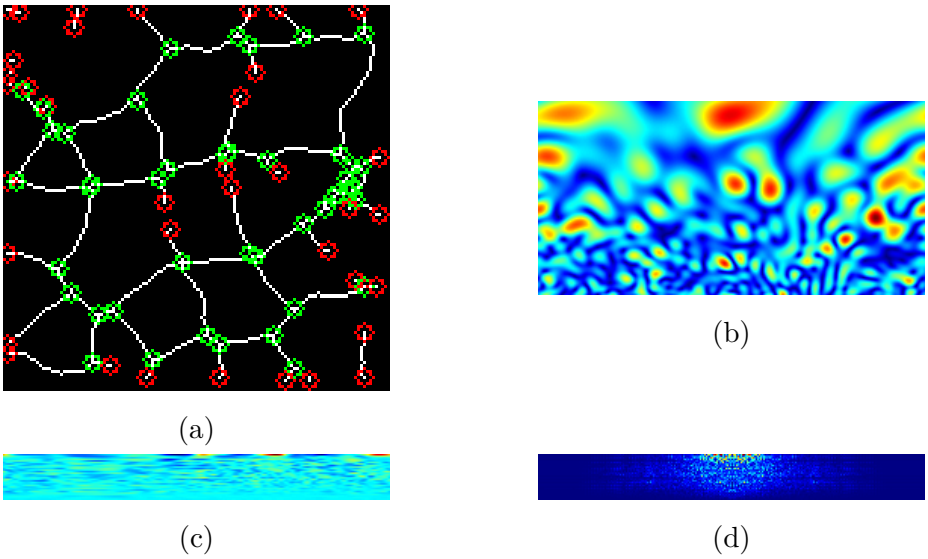
Comparison Complexity Processing a $N \times M = 32768$ sized feature vector is a large computational task and limits comparison speeds, especially with large-scale databases in biometric identification scenarios.

To address these issues, the same authors of the SMR approaches introduced two feature-reduction approaches in [XVKA09]. Both base on well-known algorithms and are explained in the following subsections.

5.4.1 Column Principal Component Analysis

Based on the idea of the well-known Principal Component Analysis (PCA) originally presented in [Pea01], the CPCA is introduced as one feature reduction approach. The authors present two problems by applying the original PCA to the SMR. First mentioned is the small sample size problem introduced in [RJ+91]: to receive a reliable PCA feature reduction, it is necessary to train the PCA with a large training set. It is unlikely possible to acquire a large training set in real-world scenarios. The second mentioned problem is the rotation invariance of the PCA:

[2] Or real valued part.



(a) (b) (c) (d)

Figure 5.3: Illustration of the feature reduction approaches: (a) input minutiae visualised on the extracted vein pattern; (b) complex-modulus SML Fourier spectrum sampled on a polar-logarithmic grid; (c) feature reduction with Column Principal Component Analsysis (CPCA); (d) feature reduction with Line Discrete Fourier Transformation (LDFT) applied on top (c).

when applying the PCA to a SMR, the resulting feature vector loses its rotation invariance and every probe has to be pre-rotated to a fixed position for all references and probes.
To avoid both problems, the PCA is only applied column-wise, resulting in the CPCA. Naïvely spoken, the CPCA scrutinizes the input SMR and applies the Singular Value Decomposition (SVD) column-wise.
The first problem is thus addressed by the number of columns (M) per SMR. Let $\mathcal{X}$ be the $M \times N$ sized 2-D feature vector received by the SMR, every column is translated in a new feature vector $\overrightarrow{z} = (z_1, \ldots, z_M)^T$, thus every spectrum consists of N feature vectors $\mathcal{X} = (\overrightarrow{z}_1, \ldots, \overrightarrow{z}_N)$. When L samples of $\mathcal{X}$ $(\mathcal{X}_1, \ldots, \mathcal{X}_L)$ are present in the training set, $L_N = L \times N$ trainings vectors $Z = [\overrightarrow{z}_1, \ldots, \overrightarrow{z}_{L_N}]$ are received. In other words, the spectra get appended on the X-axis to another and every column is analysed.
The second problem of keeping the rotation operator (shifts on the horizontal axis) is avoided by only reducing features in the means of rows. Thus the information of every column is still in the same column after the CPCA transformation and rotations can still be compensated by horizontal shifts.
Since the features are concentrated in the upper rows after applying the CPCA, the lower rows can be removed, resulting in a $N \times M_{CPCA}$ sized feature vector. According to the authors, the achieved reduction is stated up to 90% for SMO and up to 80% for SML with the CPCA approach while maintaining the biometric performance. Figure 5.3c shows the CPCA of the SML with proposed settings in figure 5.3b with a small $L = 4$. The SMR with applied CPCA-reduction will further be referenced as SMR reduced with the CPCA feature-reduction (SMR-CPCA) respective SML reduced with the CPCA feature-reduction (SML-CPCA) and SMC-

CPCA.

5.4.2 Line Discrete Fourier Transform

The second introduced feature reduction approach reduces the features in the horizontal direction. It bases on the periodic characteristic of the horizontal axis of the spectrum $\mathcal{X}$. Thus, it is possible to apply it on top of the CPCA reduction.
In the LDFT feature reduction, every row of $\mathcal{X}$ is translated in a line vector $\overrightarrow{y} = (y_1, \ldots, y_M)$, therefore $\mathcal{X} = (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_M)^T$. With the discrete Fourier transform [OWN96],
$\mathcal{X}_{LDFT} = (Y_1(k), \ldots, Y_M(k))^T$, an exact representation of $\mathcal{X}$, is received. Only the Fourier components with a certain percentage of energy, the authors propose 80% (refer to [XVKA09] for details), are retained, and thus the feature reduction is achieved. The circular shift rotation is then expressed as

$$T(m, n - n_{cs}) = exp\left(-\mathrm{j}\frac{2\pi}{N} k n_{cs}\right) \times \mathcal{X}_{LDFT};\ k = 0, 1, \ldots, N-1;\ m = 1, \ldots, M \tag{5.10}$$

with n_{cs} expressing the columns to shift.
The authors state a reduction of up to 70% for the stand-alone LDFT and up to 95% for LDFT after CPCA while maintaining the biometric performance.

5.5 Binarisation

Since $\sum_{i=1}^{Z} exp(-\mathrm{j}(w_x x_i + w_y y_i)) \in \mathbb{C}$, thus $\mathcal{M}(w_x, w_y) \in \mathbb{R}$, every element in $\mathcal{X}$ is defined as a 32bit or 64bit floating-point real-valued number. Comparisons or calculations (especially divisions) with single or double precision floating-points is a relatively complex task compared to integer operations. In the proposed SMR scenario, where a direct comparison (see section 5.6) is defined as one multiplication, one division and one addition per feature, $N \times M \times OP_{cnt} = 256 \times 128 \times 3 = 98304$ floating-point operations[3] are needed for one comparison. To address this computational complexity and comply with other template-protection or indexing approaches where a binary feature-vector is required, the SMR can be converted to a binary feature vector as presented in [XV10a].
The authors introduce two binarisation approaches, each using a different data model but both resulting in the same representation.

The binarisation approaches yield two binary vectors: a so-called *sign-bit* vector and a so-called *mask-bit* vector:

sign-bit The sign-bit vector contains the actual features of the SMR in a binary representation. Each bit is set according to one of the two binarisation approaches.

[3] A small additional overhead for preprocessing optimised values is ignored.

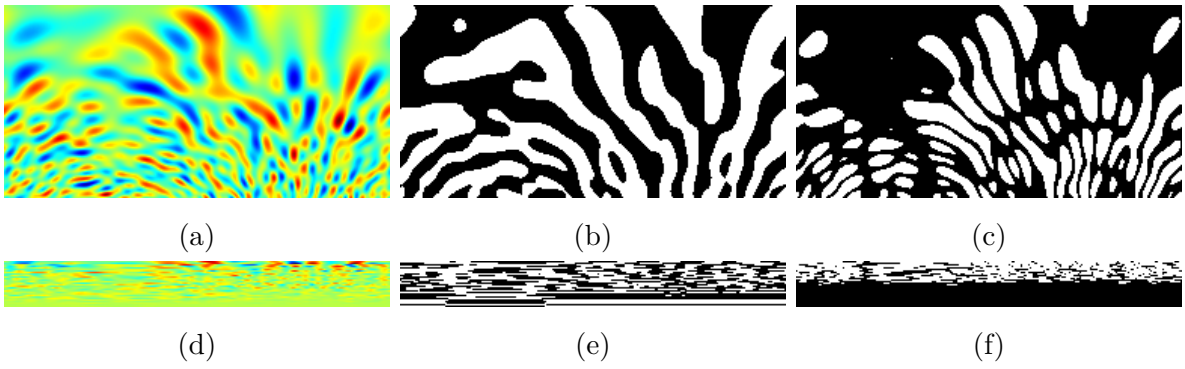


Figure 5.4: Input and result of the SMR-binarisation for SML and SML-CPCA: (a) real-valued SML input; (d) real-valued SML-CPCA input; (b) the spectral sign-bit of (a); (e) the spectral sign-bit of (d); (c) the spectral mask-bit of (a); (f) the spectral mask-bit of (d).

mask-bit Since binary representations suffer from bit-flips on edges in fuzzy environments, a second vector is introduced. This vector masks the likely-to-be-stable, called reliable, sign-bits.

The mask contained in the mask-bit vector is not applied to the sign-bit. Instead, it is kept as auxilary data and is applied during the comparison step[4].
The authors describe two approaches of receiving both vectors:

Spectral-Bits The Spectral-Bits are the result of the first, naïve approach. Let $\mathcal{X}_{\mathbb{R}}$ be the SMR feature vector, $\mathcal{X}_{\mathbb{B}Ssign}$ the sign-bit vector and $\mathcal{X}_{\mathbb{B}Smask}$ the mask-bit vector of the Spectral-Bits. Then $B_{\mathcal{X}_{sign}}$ is defined as

$$\mathcal{X}_{\mathbb{B}Ssign}(x,y) = \begin{cases} 1, & \text{if } \mathcal{X}_{\mathbb{R}}(x,y) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.11}$$

and B_{mask} as

$$\mathcal{X}_{\mathbb{B}Smask}(x,y) = \begin{cases} 1, & \text{if } |\mathcal{X}_{\mathbb{R}}(x,y)| > MT \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

Using the proposed threshold of 0.8 as MT for the mask-bit ensures masking only the stable bits of $\mathcal{X}_{\mathbb{B}Ssign}$.

Phase-Bits While the Spectral-Bits only evaluate the real-valued SMR, the Phase-Bits also use the imaginary part of the SMR spectrum $\mathcal{X}_{\mathbb{C}}$. A third dimension is added to the sign-bit and mask-bit vectors for the Phase-Bits. The subsequent steps equal the Spectral-Bits approach but with an additional dimension. Let $\mathcal{X}_{\mathbb{C}}$ be described as $\mathcal{X}_{\mathbb{C}}(x,y) = (real + i * imag)$ and again $\mathcal{X}_{\mathbb{B}Psign}$ the sign-bit vector and $\mathcal{X}_{\mathbb{B}Pmask}$ the mask-bit vector

[4]This approach equals the masking procedure in iris recognition (see [Dau03, Dau04]).

of the Phase-Bits, then

$$\mathcal{X}_{\mathbb{B}Psign}(x,y) = \begin{cases} [0,0], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} \leq 0 \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} \leq 0 \\ [0,1], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} \leq 0 \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} > 0 \\ [1,0], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} > 0 \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} \leq 0 \\ [1,1], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} > 0 \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} > 0 \end{cases} \tag{5.13}$$

and

$$\mathcal{X}_{\mathbb{B}Pmask}(x,y) = \begin{cases} [0,0], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} \leq MT \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} \leq MT \\ [0,1], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} \leq MT \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} > MT \\ [1,0], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} > MT \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} \leq MT \\ [1,1], & \text{if } \mathcal{X}_{\mathbb{C}}(x,y)_{real} > MT \text{ and } \mathcal{X}_{\mathbb{C}}(x,y)_{imag} > MT \end{cases} \tag{5.14}$$

For the mask-bit vector of the Phase-Bits, a threshold of 1.2 is proposed.

5.6 Comparison

In the initial publication [XVK+08], two comparison methods for the SMR are introduced: one direct comparison approach and one weighted comparison approach. Both of them showed almost identical results, and thus the weighted comparison approach was dropped in later publications because the slightly better results did not justify additional training and computational overhead. Later on in [XVB+09], a third comparison method that tries to compensate rotation and scaling variances by using an additional Fourier transform is introduced. However, this approach shows a significantly lower performance in terms of FMR, FNMR and Equal Error Rate (EER) and was also dropped in later publications. Therefore, the next sections will only cover the direct comparison methods, that are used later in this thesis.

5.6.1 Spectral Minutiae

The most proven performance in SMR comparison is reached with the so-called direct comparison[5]. It yields the most reliable comparison scores while keeping a minimal computational complexity.

Let $R(m,n)$ be the spectrum of the reference template and $P(m,n)$ the spectrum of the probe template, both sampled on the polar-logarithmic grid and normalised to zero mean and unit energy (see section 5.3). Then, the similarity-score $E_{DM}^{(R,P)}$ is defined as

$$E_{DM}^{(R,P)} = \frac{1}{MN} \sum R(m,n)P(m,n) \tag{5.15}$$

The score is thus defined by correlation, which is a common approach in image processing.

[5] In [XVK+08], the comparison method is named *direct matching*, where *matching* is used as wrong term for *comparison*.

5.6.2 Binary Spectral Minutiae

For comparing two binary SMR or SMR-CPCA, a different approach is introduced in [XV10a], which is also used in the iris modality [Dau03, Dau04].
After converting $R(m,n)$ and $P(m,n)$ in their individual mask-bit and sign-bit, yielding $\{maskR, signR\}$ and $\{maskP, signP\}$, the fractional Hamming Distance (FHD) can be applied on those binary representations.

$$FHD^{(R,P)} = \frac{\|(signR \otimes signP) \cap maskR \cap maskP\|}{\|maskR \cap maskR\|} \tag{5.16}$$

The inclusion of masks in the Hamming Distance masks out any likely-to-flip bits and only compares the parts of the sign-bit vector where the mask-bit vectors overlap. Therefore, only the likely-to-be-stable, so-called reliable, areas are compared. This typically improves the recognition performance.

5.7 Extending the Spectral Minutiae Representation with quality data

As an enhancement for the SMR, a approach of using quality information about minutiae is presented in [XV09]. The authors differ between two types of quality:

Location Accuracy (q_L) Under certain circumstances, the position of a minutiae cannot be accurately determined. For example, when wide veins need to be thinned to their skeleton, a displacement of the actual vein centre could happen. The Location Accuracy describes the expected certainty that the minutiae has not been misplaced in the feature-extraction process.

Minutiae Reliability (q_M) It is possible for feature-extraction pipelines to falsely extract minutiae. Some pipelines are able to determine a genuine certainty for each minutiae that describes the certainty that the extracted reference point is a genuine minutiae and not a spurious minutiae. The following steps can decide whether they use a minutiae or how to weight them.

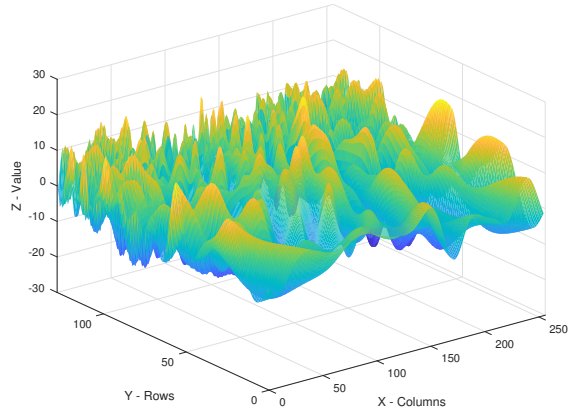
Both types of quality can be incorporated independent of one another in all previously-presented SMR, yieling the quality data enhanced Spectral Minutia Location Representation (QSML) and the quality data enhanced Spectral Minutia Complex Representation (QSMC).
Recall section 5.1.1, where equation 5.5 defines the Gaussian smoothed SML. The Gaussian filter helps to reduce small location errors for all minutiae. Now, if the accuracy of minutiae locations is known, instead of using a fixed σ Gaussian kernel for all minutiae, σ is linear depending on the minutiae quality individually, thus yielding

$$\mathcal{M}(w_x, w_y) = \left| \sum_{i=1}^{Z} \left(exp\left(-\frac{w_x^2 + w_y^2}{2\sigma_i^{-2}} \right) * exp(-\mathrm{j}(w_x x_i + w_y y_i)) \right) \right| \tag{5.17}$$

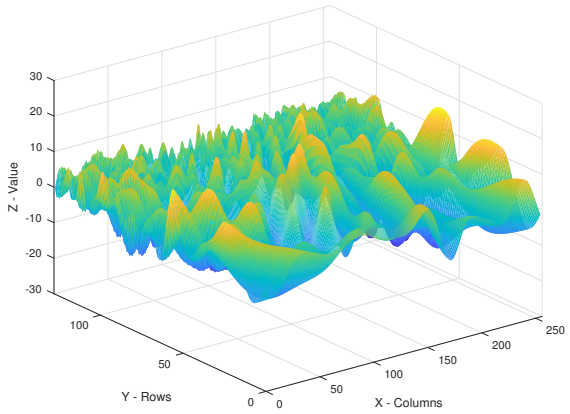
as a new equation for the SML with location accuracy quality.
When the minutiae reliability is known, the Dirac pulse (eq. 5.1) of each minutiae can be weighted linearly:

$$\mathcal{M}(w_x, w_y; \sigma^2) = \left| exp\left(-\frac{w_x^2 + w_y^2}{2\sigma^{-2}}\right) \sum_{i=1}^{Z} w_i exp(-\mathrm{j}(w_x x_i + w_y y_i)) \right| \tag{5.18}$$

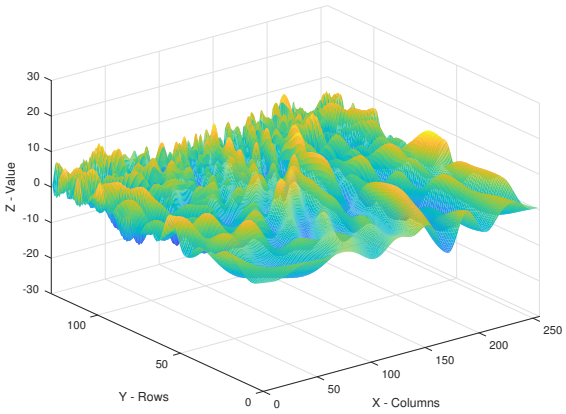
A higher reliability correspondents to a higher weight w_i for minutiae $m_i(x, y, q_M)$.



(a) Raw SMR



(b) Location accuracy



(c) Minutiae reliability

Figure 5.5: Effects of the quality data on the (not normalised) SML

Compare figure 5.5a and 5.5b: in 5.5b the location accuracy quality data is used with an average accuracy of 50%. The location accuracy smooths the high frequency parts of the SMR that are sampled in the higher Y-axis regions ($Y > 64$), and thus the impact[6] of a minutia with a low accuracy is reduced in high frequencies, while minutia with a high location accuracy keep their impact on the high frequencies. With increasing frequency, a minutiae's location is

[6]Increase of the SMR value (Z-axis).

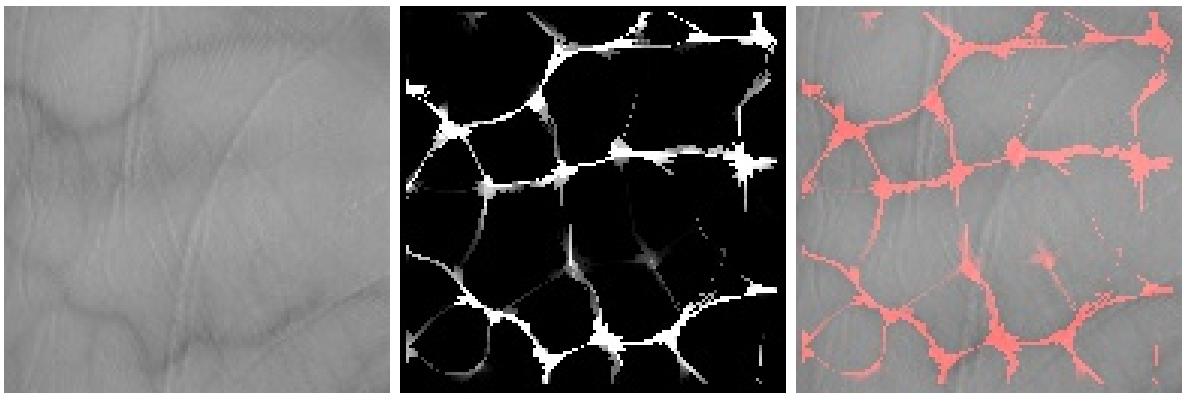
narrowed down: the higher the frequency, the more precise the location information.
By contrast, figure 5.5c shows the effect of the minutiae reliability on the SMR, compared to figure 5.5a. The minutiae reliability reduces the impact on all frequencies of the SMR. Therefore, minutiae with a low reliability will leave a smaller footprint in the SMR than minutiae with a high reliability. Observe the lower frequencies along the beginning and the end of the X-axis, whereby the value of the SMR is reduced since unreliable minutiae have a lesser impact.
Note: the QSML shown in figure 5.5 are not normalised to $[-1, 1]$. The normalisation process accentuates the visualisation of the quality data impact since it evenly distributes the Z-axis.

5.7.1 Retrieving and applying minutiae-reliability data in vascular modalities

Applying minutiae-reliability quality data promises increasing performance in terms of GAR. In chapter 4, a feature-extraction pipeline was introduced which is already able to report minutiae-reliabilty data.

One approach for extracting quality data could be to use the contrast of the veins. However, this introduces at least one concern: image contrast is relative. To use the contrast as quality data, a reference value is needed. Using the average image brightness as reference could yield small contrast gradations. Second, it is very hard to normalise the output of the contrast quality data approach. Therefore, another quality data extraction approach is proposed.

Recall section 4.4, whereby the maximum-curvature approach yields a score map describing the probability that a local maxima actually represents the centre of a vein. Observe figure

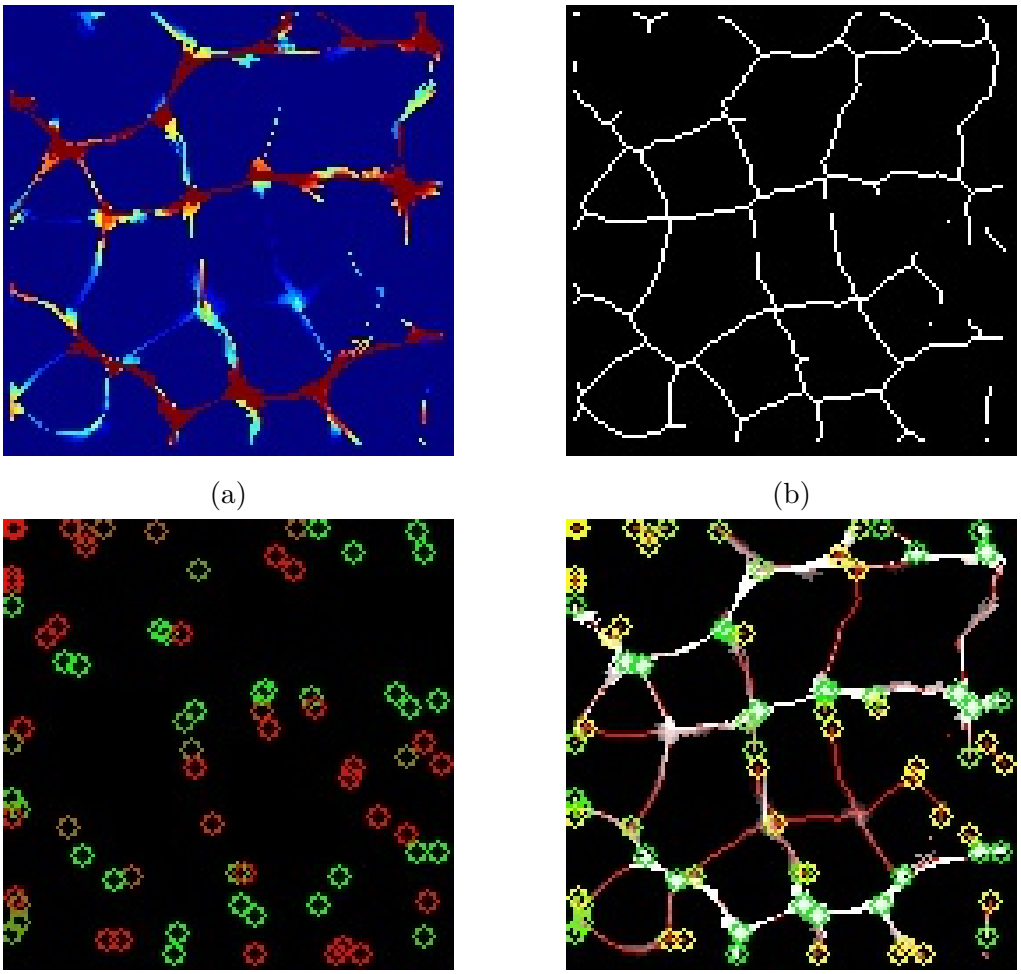


(a) (b) (c)

Figure 5.6: Feature detection with the maximum-curvature approach: (a) input probe; (b) response of the maximum-curvature algorithm; (c) probe (a) with layover response (b).

5.6, it is visible, that the response also corresponds to the darkness and contrast of the vein: clearly-visible veins result in a high response by the maximum-curvature algorithm, while less

noticeable veins result in a low response. Since these small, less noticeable veins are more likely to be affected by short-term influences like temperature or blood pressure, it is desirable to weight their minutiae less than those of the larger veins. As described in section 4.5, the used thinning algorithm best keeps the initial shape and best hits the actual centre of the input. Thus, it is safe to assume that the minutiae found in the skeleton will be in the centre of their maximum-curvature extracted veins. Following the assumption, the value at the minutiae



(a) (b) (c) (d)

Figure 5.7: Extending the minutiae with reliability data taken from the response of the maximum-curvature algorithm: (a) response of the maximum-curvature algorithm of the probe showed in figure 5.6a (jet colour-map); (b) skeleton of the vein pattern; (c) scored minutiae (red = lowest, green = highest score); (d) fusion of the previous figures (maximum-curvature without jet colour-map; minutiae yellow = lowest, green = highest score).

location in the maximum-curvature response corresponds to the minutiae reliability.

A drawback of this approach is its dependency on overall contrast: probes with low contrast, even after the image enhancement, will automatically result in a overall lower reliability. To address this issue, the maximum-curvature response has to be normalised to $[0, 1]$. Let $\overrightarrow{MC}$ be the maximum-curvature response as one continuous vector, then the normalised maximum-

curvature vector $\widehat{MC}$ is given by

$$\widehat{MC}(x,y) = \min\left(g\frac{MC(x,y)}{\max \overrightarrow{MC}}, 1\right) \tag{5.19}$$

where g is a gain factor used to adjust the minimal needed response for 100% ($\widehat{MC}(x,y) = 1$) minutiae reliability. After applying the normalisation to the maximum-curvature response, the values are contrast-invariant and can safely be used as reliability descriptors.
For further confidence, it is possible to include the minutiae N4- or N8-neighbourhood of the normalised maximum-curvature response in the calculation and take the mean or weighted mean as minutiae reliability. Thus scores of minutiae detected from single pixel width maximum-curvature responses will be reduced while the score of reliable thick responses will barely be altered.

5.7.2 Retrieving and applying location-accuracy data in vascular modalities

Even if the thinning algorithm used (see section 4.5) promises the best skeleton accuracy, it is possible for the skeleton to be shifted by a few pixels. Therefore, it might benefit the GAR to retrieve and use the location-accuracy data to reduce errors introduced by shifted minutiae.
Again using the maximum-curvature response, the location-accuracy can be approximated for individual minutiae points. Instead of using the normalised maximum-curvature response, a binary representation will be used. The location accuracy is then approximated by the width of the vein. A naïve approach is to test the vein width by taking the minimum span of set bits of the binary maximum-curvature response MC_B in each direction as proposed in listing 1. The returned pixel span can now be used to approximate the location accuracy relatively to the probes size: minutiae where their returned $SpanDir_0$ differs from $SpanDir_1$ by more than a few pixel are likely to be shifted in other probes. Thus, the difference of $SpanDir_0$ and $SpanDir_1$ ($\Delta_{SpanDir}$) corresponds linearly[7] to the location accuracy.

5.8 Minutiae pre-selection

Besides the quality extensions presented in the previous section, the SMR can further be expanded by a simple minutiae pre-selection. Since the feature extraction pipeline extracts various spurious minutiae, it holds interest whether the biometric performance can be increased by automatically ignoring minutiae, based on simple rules:

Type A naïve selection approach is to either select only endpoints or select only bifurcations for the SMR generation. However, it is expected that only selecting endpoints results

[7] Note that the difference has to be put in perspective to the probe's size: a difference of $5px$ is much less of an issue in a 512×512 sized probe than in a 128×128 sized probe.

Algorithm 1 Naïve vein span algorithm for minutiae $m(x, y)$.

function VEINSPAN(MC_B, x, y)
  $sHeight_0 \leftarrow 1$, $sHeight_1 \leftarrow 1$, $sWidth_0 \leftarrow 1$, $sWidth_1 \leftarrow 1$, $sDiagA_0 \leftarrow 1$, $sDiagA_1 \leftarrow 1$, $sDiagB_0 \leftarrow 1$, $sDiagB_1 \leftarrow 1$
  $grow \leftarrow true$, $growFactor \leftarrow 1$
  while $grow = true$ **do**
    $heightGrow \leftarrow 0$ ▷ Height-Span
    if $MC_B(x, y + growFactor) = 1$ **then**
      $sHeight_0 \leftarrow sHeight_0 + 1, heightGrow \leftarrow heightGrow + 1$
    end if
    if $MC_B(x, y - growFactor) = 1$ **then**
      $sHeight_1 \leftarrow sHeight_1 + 1, heightGrow \leftarrow heightGrow + 1$
    end if
    $widthGrow \leftarrow 0$ ▷ Width-Span
    if $MC_B(x + growFactor, y) = 1$ **then**
      $sWidth_0 \leftarrow sWidth_0 + 1, widthGrow \leftarrow widthGrow + 1$
    end if
    if $MC_B(x - growFactor, y) = 1$ **then**
      $sWidth_1 \leftarrow sWidth_1 + 1, widthGrow \leftarrow widthGrow + 1$
    end if
    $diagAGrow \leftarrow 0$ ▷ Diagonal-Span (/)
    if $MC_B(x + growFactor, y - growFactor) = 1$ **then**
      $sDiagA_0 \leftarrow sDiagA_0 + 1, diagAGrow \leftarrow diagAGrow + 1$
    end if
    if $MC_B(x - growFactor, y + growFactor) = 1$ **then**
      $sDiagA_1 \leftarrow sDiagA_1 + 1, diagAGrow \leftarrow diagAGrow + 1$
    end if
    $diagBGrow \leftarrow 0$ ▷ Diagonal-Span (\)
    if $MC_B(x - growFactor, y - growFactor) = 1$ **then**
      $sDiagB_0 \leftarrow sDiagB_0 + 1, diagBGrow \leftarrow diagBGrow + 1$
    end if
    if $MC_B(x + growFactor, y + growFactor) = 1$ **then**
      $sDiagB_1 \leftarrow sDiagB_1 + 1, diagBGrow \leftarrow diagBGrow + 1$
    end if
    if $heightGrow = 0$ **or** $widthGrow = 0$ **or** $diagAGrow = 0$ **or** $diagBGrow = 0$ **then**
      $grow \leftarrow false$
    end if
    $growFactor \leftarrow growFactor + 1$
  end while
  return $\min((sHeight_0, sHeight_1), (sWidth_0, sWidth_1), (sDiagA_0, sDiagA_1), (sDiagB_0, sDiagB_1)$
end function

in very poor biometric performances since *only selecting endpoints corresponds to only selecting noise*.

Reliability Based on the minutiae-reliability extraction presented in section 5.7.1, only minutiae with a reliability score that exceeds a certain threshold will be selected for SMR generation.

Neighbourhood The neighbourhood selection (neighbourhood cleaning) extends the approach to only select bifurcations: all endpoints are excluded from the minutiae list. Further, the bifurcation minutiae whose closest neighbour (in the means of Euclidean distance) is a endpoint minutiae are also excluded.

The reliability criterium can be combined with the other two, resulting in five minutiae selection approaches. Applying one or a combination of the minutiae pre-selection to the SMR yields the Spectral Minutia Representation with minutiae pre-selection (PSMR), respective the PSMR sub-types Spectral Minutia Location Representation with minutiae pre-selection (PSML), Spectral Minutia Complex Representation with minutiae pre-selection (PSMC), PQSML and PQSMC.

5.9 Chapter Conclusions

In the first section of the chapter, the SMR has been outlined based on the original publications and the collection in [XV10c]. With the SMR, Xu et al. introduced a promising representation for templates of fingerprints. The SMR features comparable characteristics for fingerprint templates like the Iris-Code (see [Dau04]) for iris recognition: yielding a fixed-length feature vector with template-protection properties and fast comparison methods for the complex-valued as well as the binary-valued representation.
Since the SMR is based upon minutiae data, it should be feasible to apply the SMR to other minutiae-based modalities. Hartung et al. analysed the feasibility of the SMR for finger veins, wrist veins and veins in the back of the hand in [HOXB11] and [HOX+12]. The authors concluded the feasibility of the SML variant for those vein modalities. Continuing on the work of Hartung et al., the experiments in section 9.2 analyse the feasibility of the SMR for the palm vein characteristic and state recommendations and hints about applying the SMR in the palm vein modality.
The final two sections outlined the quality enhancements for the SMR by [XV09] with the SML as a example with a new contribution: a proposal to retrieve and apply minutiae-reliability as well as minutiae-accuracy data based on the maximum-curvature approach. Enhancing the SML with quality data yieldes the QSML, and enhancing the SMC yields the QSMC representation. The approaches are expected to raise the GAR for vascular minutiae and are subject to experiments in section 9.2.
Thus far, no approaches for workload reduction to address the issue stated in section 2.2.3 have

been introduced. The next chapters introduce two promising indexing approachs for workload reduction that are used in the experiments.

Chapter 6

Bloom filter-based indexing

In section 2.2.3, the challenges associated with large-scale biometric identification databases are described. To tackle those challenges, workload reduction strategies have to be implemented. This chapter outlines the first workload reduction approach chosen in this thesis using the Bloom filter.

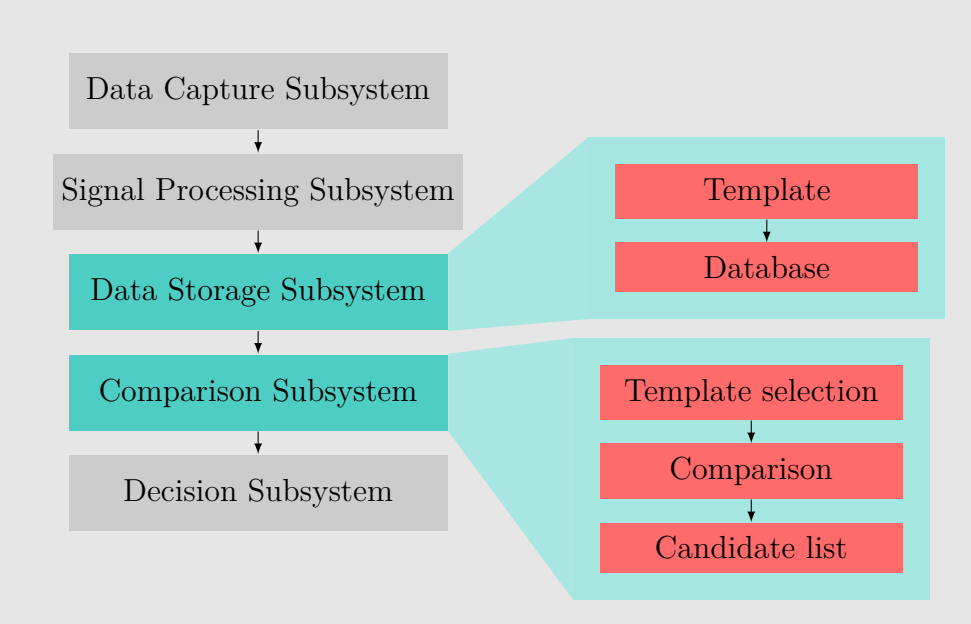


Figure 6.1: Bloom filter-based indexing

6.1 Bloom Filter

Introduced as early as 1970 in [Blo70], the Bloom filter is still used today in a broad scope in computer sciences (e.g. Big Data [CDG+08], CDN-caching [MS15], URL-matching [Yak10, Hes12] and computer forensics). The purpose of the probabilistic data structure is an efficient membership query. A Bloom filter (denoted $\mathcal{B}$) is organized as a vector of l bits. In the initial (empty) Bloom filter, every bit is set to zero.
Upon adding or for querying, an element has to be converted to its Bloom filter representation.

Hash-set creation The new element or query is hashed by a fixed number (k) of independent hashing functions $H_1, \ldots, H_k$ that produce hash values in the range of $[0, l]$ resulting in the hashes $h_1, \ldots, h_k$. These hash values represent indices for the Bloom filter.

Now the hashes $h_1, \ldots, h_k$ can either be added to the Bloom filter or a membership query can be run:

Adding to the Bloom filter An element is added to the Bloom filter by taking the hash values h_i as indices for the Bloom filter and setting the corresponding bits to one as by $\mathcal{B}_N[h_i] = 1 \in 0 \ldots k$. Thus an element is added to $\mathcal{B}$ by $\mathcal{B} = \mathcal{B} \vee \mathcal{B}_N$.

Membership query As well as in adding to the Bloom filter, the queried element is converted to its Bloom filter representation by $\mathcal{B}_Q[h_i] = 1 \in 0 \ldots k$. Now the query is performed by comparing the Bloom filter with the query by $|\mathcal{B} \wedge \mathcal{B}_Q|$. Only if $|\mathcal{B} \wedge \mathcal{B}_Q| = |\mathcal{B}_Q|$ is fulfilled is the query's element contained in the Bloom filter. Thus, if any of the set bits in $\mathcal{B}_Q$ is not set in $\mathcal{B}$, the element is definitely not part of the Bloom filter, and therefore no false negatives occur. However, it is possible for false positives to occur due to bit collision. Therefore, a Bloom filter is only able to tell if the element is perhaps or definitely not part of the Bloom filter.

Bit collisions cannot be prevented, but can be delayed by increasing l. A Bloom filter is full if all bits are set and thus all queries will yield a positive result. It is desirable to keep a Bloom filter as a sparse matrix to minimise the probability of bit collisions. The event of exceeding the threshold of set bits for a Bloom filter is further called overflow in this thesis.

In its basic definition, the Bloom filter offers very successful and efficient membership queries for scenarios where the probe and the reference are identical. However, in biometric systems, the Bloom filters can be enabled to work with fuzzy data, allowing its usage in verification and naïve identification scenarios if an appropriate comparison function (e.g. the hamming distance) is used.
The initial objective of this work was to apply the Bloom filter on the SMR to achieve a significant workload reduction for identification scenarios. In [RBB13] and [RBBB15], the Bloom filter approach is already successfully applied to Iris-Codes. Since Iris-Codes are fixed-length, binary structures, the approach is tailored to templates representing thsee characteristics. As presented in section 5.5, the SMR is also transferable in a fixed-length binary representation of biometric references. Therefore, applying the same concept to the binary SMR seems to be a reasonable approach. In the next sections, the basics of this concept are outlined.

6.2 System Basics

In this section, operational details of the Bloom filter-based scheme presented in [RBBB15] and extended by [DRB17] will be presented. Both publications provide the basis and the baseline for

the workload reduction approach, which is benchmarked against a new contribution presented later in this thesis.

6.2.1 Template Transformation

To transform a SMR template to its Bloom filter representation, the procedure explained below is used.

1. The SMR has to be converted to its binary form (see section 5.5). Since the basic form does not support auxiliary-data schemes, i.e. the $sign, mask$ pair, the mask-bit has to be applied to the sign-bit so that $matrix = sign \wedge mask$.

2. The resulting binary matrix has to be segmented into equally-sized blocks of width $\mathcal{W}$ and height $\mathcal{H}$.

3. In contrast to the original Bloom filter approach, where multiple hash function are applied to the data, a simple mapping scheme is used. Each block is mapped to a Bloom filter ($\mathcal{B}$) by interpreting the bits of each column $(c_1, \ldots, c_\mathcal{W})$ as unsigned integer $(h_1, \ldots, h_\mathcal{H})$, which are then used as indices for setting the bits in the Bloom filter. This represents an inverted hashing strategy: the original Bloom filter uses multiple hashing functions on a single element, whereas here multiple elements (the columns) are hashed by a single hashing function. Both strategies yield the same end result of $h_1, \ldots, h_\mathcal{W}$ hash values subsequently used as indices.

4. Repeat step 3 for every block: each block yields a Bloom filter $\mathcal{B}_i$ resulting in a sequence of Bloom filters ($\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{BC}\}$) of length $BC = \frac{N}{\mathcal{W}} \times \frac{M}{\mathcal{H}}$.

The transformation features rotation-compensating properties for the binary SMR matrix. In most width and height configurations, fewer shifts for rotation compensation have to be done, due to the indices scheme.
The size (τ) of a Bloom filter sequence template with block size $\mathcal{W} \times \mathcal{H}$ for a $N \times M$ binary SMR matrix is calculated by equation 6.1.

$$\tau = 2^\mathcal{H} * \frac{N}{\mathcal{W}} * \frac{M}{\mathcal{H}} \tag{6.1}$$

With the described template transformation, it is possible to use the Bloom filter in a verification and a naïve identification scenario. A sequence of Bloom filter for a $N = 256 \times M = 128$ binary SMR with a block size of $\mathcal{W} = 8 \times \mathcal{H} = 4$ reduces the bit comparisons by 50%.

However, in identification scenarios the overhead for large-scale databases is dwindling relative to the overall computation time. To use the modified Bloom filter for a more efficient identification than a naïve approach, the methods described in the following two sections are used.

6.2.2 Tree Construction

Storing the Bloom filter-based templates in a binary search tree enables a more efficient identification by reducing the number of comparisons needed. The process of building the tree is described below and conceptually illustrated in figure 6.2.

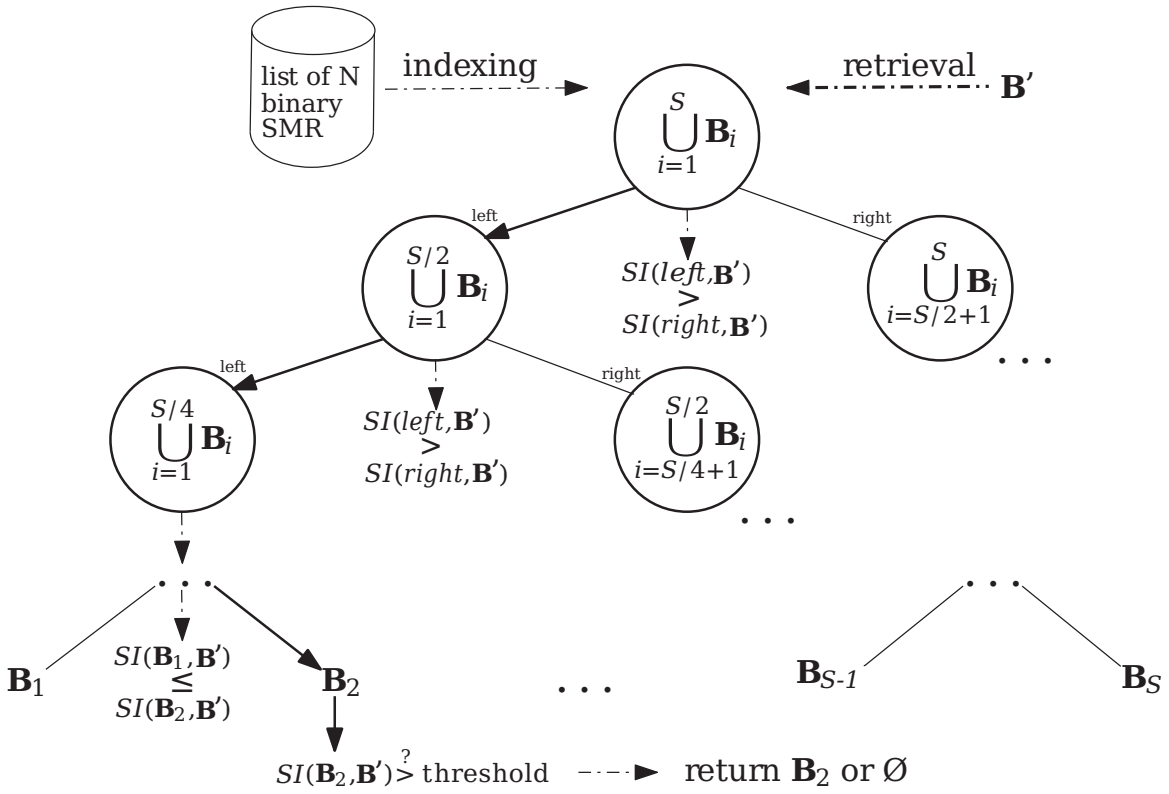


Figure 6.2: Bloom filter tree construction scheme and tree traversal for query $\mathbf{B}'$ (taken from [DRB17], original in [RBBB15]).

1. The tree root is created as the element-wise union of all templates ($\mathcal{S}$) such that $\Psi_0 = \mathbf{B}_1 \cup \mathbf{B}_2 \cup \cdots \cup \mathbf{B}_\mathcal{S} = \bigcup_{i=1}^{\mathcal{S}} \mathbf{B}_i$.
2. Child nodes are created by splitting the template list of their parents in two parts and taking the element-wise union of one of the two sub-sets (i.e. $\Psi_1 = \mathbf{B}_1 \cup \cdots \cup \mathbf{B}_{\mathcal{S}/2} = \bigcup_{i=1}^{\mathcal{S}/2}$, $\Psi_2 = \mathbf{B}_{(\mathcal{S}/2)+1} \cup \cdots \cup \mathbf{B}_\mathcal{S} = \bigcup_{i=\mathcal{S}/2+1}^{\mathcal{S}}$).
3. Repeating step 2 until only the leafs are left, which then are the templates ($\mathbf{B}_1, \ldots, \mathbf{B}_\mathcal{S}$) themselves (e.g. $\Psi_{2\mathcal{S}-2} = \mathbf{B}_\mathcal{S}$, if $\mathcal{S}$ is a power of two).

Inserting a new template into an already-built tree is trivial and can be achieved in $O(log\mathcal{S})$ steps: The new template is inserted as a new leaf and is added to the element-wise union of its parents and their predecessors. If no node has an empty leaf, a leaf is converted to a node and

receives its template as well as the new template as children (respectively leafs). Removing one or more templates requires fully rebuilding[1] the tree, which is done in $O(\mathcal{S}log\mathcal{S})$ steps.

6.2.3 Tree Traversal

Upon receiving a membership query, the query is transformed in its Bloom filter representation ($\mathbf{B}'$). Subsequently, the tree traversal for a single tree starts at the root node ($\Psi_{current} = \Psi_0$) by computing the similarity score for the two children ($\Psi_{current+1}, \Psi_{current+2}$) of the root node. Basing on the two received scores, the direction of the first descend is determined, thus $\Psi_{current} = \Psi_{current+1}$ if the similarity score of $\Psi_{current+1}$ and $\mathbf{B}'$ is higher than the score of $\Psi_{current+2}$ and $\mathbf{B}'$, respectively $\Psi_{current} = \Psi_{current+2}$ vice versa. These direction decisions are repeated until a leaf is reached. Finally, the similarity score of the leaf and $\mathbf{B}'$ is compared to a threshold, which has been previously trained with a set disjoint from the enrolled templates or manually set by an operator. The lookup process is conceptually illustrated in figure 6.2 (bold lines) and formally stated in algorithm 2.
The similarity score of two Bloom filter sequences ($\mathbf{B}$, $\mathbf{B}'$) is calculated by a score-level fusion of the pair-wise similarities between the $\mathcal{B}_i \in \mathbf{B}$ and $\mathcal{B}'_i \in \mathbf{B}'$ as shown in function 6.2. By taking the amount of matching bits and the population count of the set of matching bits of booth $\mathcal{B}$ and $\mathcal{B}'$, the similarity is calculated. Because the population counts of $\mathcal{B}$ and $\mathcal{B}'$ varies, it is necessary to normalise the score by subtracting the average of both population counts. Equation 6.3 shows the final similarity calculation for $\mathcal{B}$ and $\mathcal{B}'$.

$$SI(\mathbf{B}, \mathbf{B}') = \frac{1}{BC} \sum_{i=1}^{BC} SI(\mathcal{B}_i, \mathcal{B}'_i) \tag{6.2}$$

$$SI(\mathcal{B}, \mathcal{B}') = \frac{|\mathcal{B} \wedge \mathcal{B}'|}{\frac{1}{2}(|\mathcal{B}| + |\mathcal{B}'|)} \tag{6.3}$$

Because a decreasing similarity score per level is expected for impostor queries, an additional workload reduction, for impostor queries and queries for templates stored in another tree, can be achieved by prematurely stopping the tree traversal for the current tree, when the similarity score of the different levels is required to be ordered (i.e. $SI_{level_0} < SI_{level_1} < SI_{level_2} < \dots$). As a side effect, it is more difficult for impostors to be accepted, thus potentially lowering the False Positive Identification Rate (FPIR). However, some genuine queries may also be rejected, potentially increasing the False Negative Identification Rate (FNIR). Therefore, requiring a increasing similarity per level is considered as optional and is only recommended for biometric subsystems with a very robust feature-extraction subsystem.

By increasing the number of stored templates ($\mathcal{S}$) in a Bloom filter tree, the population count[2] of the root nodes $\mathbf{B}(\Psi_0)$ increases. While this is less an issue for the root node itself, an

[1] A derivative of the Bloom filter, the so-called counting Bloom filter (see [RB13]) addresses this issue on the cost of the efficiency of performing bitwise operations.

[2] In other words, the sparsity of the root node decreases.

excessive population count in the lower nodes affects the capability of making correct traversal direction decisions. A wrong traversal direction decision at the upper levels resulting from too many bit collisions is not correctable in lower levels, thus resulting in a False Negative Identification (FNI) event or worse in a False Positve Identification (FPI) event[3]. Therefore, it is necessary to build multiple trees with fewer subjects respectively biometric instances per tree. The lookup process for multiple trees employs the same strategy for traversing a tree as for single tree environments. Merely a trivial loop-logic has to be added, as shown formally in algorithm 3.

Algorithm 2 Tree traversal for a single Bloom filter tree.

```
1:  function TraverseTree(Node, Query, Threshold, LastScore)
2:      if Node[left] ≠ nil and Node[right] ≠ nil then
3:          scoreLeft ← similarity(Node[left], Query)
4:          scoreRight ← similarity(Node[right], Query)
5:          if scoreLeft > scoreRight then
6:              if scoreLeft < LastScore then
7:                  return nil                                  ▷ Optional
8:              end if
9:              return TraverseTree(Node[left], Query, Threshold, scoreLeft)
10:         else
11:             if scoreRight < LastScore then
12:                 return nil                                  ▷ Optional
13:             end if
14:             return TraverseTree(Node[right], Query, Threshold, scoreRight)
15:         end if
16:     end if
17:     score ← similarity(Node[this], Query)
18:     if score > Threshold and score > LastScore then   ▷ Second condition optional
19:         return Node[this]
20:     end if
21:     return nil
22: end function
```

6.2.4 Configuration

By now, three variables are to be considered for the basic Bloom filter tree system: Bloom filter width ($\mathcal{W}$), Bloom filter height ($\mathcal{H}$) and tree count ($\mathcal{T}$). Each of them can be adjusted considering the following limitations.

$\mathcal{W}$ Increasing the block width for the Bloom filter increases the number of bit collisions when transforming the binary SMR-matrix to the Bloom filter representation. Reducing the

[3]This behaviour was observed in the experiments: queries that would normally have the highest similarity score with their corresponding reference template in a naïve approach do not find their correct reference in the Bloom filter since the high amount of bit collisions direct the query to a different reference template where the similarity score unfortunately exceeds the decision threshold.

Algorithm 3 Additional loop-logic for multiple Bloom filter trees.

```
1: function LOOKUP(Trees, Query, Threshold)
2:     candidates ← empty list
3:     for all Trees as root do
4:         candidate ← TraverseTree(root, Query, Threshold)
5:         if candidate ≠ nil then
6:             candidates ← candidates + candidate
7:         end if
8:     end for
9:     return candidates
10: end function
```

block width increases the total template size, thus requiring more RAM, storage and bit comparisons. Further, it diminishes the rotation invariance properties of the templates.

$\mathcal{H}$ Increasing the block height for the Bloom filter increases the total template size. Reducing the block height increases the number of bit collisions and increases the overall population count of the Bloom filter.

$\mathcal{T}$ Employing additional trees for the Bloom filter reduces the number of templates per tree, therefore reducing the bit collisions and increasing the hit rate, thus decreasing the FNIR. However, additional trees require more template comparisons, thus increasing the workload. Refer to figure 6.3, with $\mathcal{S} = 512$ using two trees nearly doubles the number of template comparisons needed where using 64 trees hardly yields a workload reduction.

6.3 State-Of-The-Art Bloom filter approach

As already mentioned in section 6.2.2, wrong traversal decisions can occur if the child nodes are too populated[4], thus it is required to build multiple trees with fewer subjects. However, doing so introduces a new issue: multiple trees cause higher workload. For a single tree environment, the number of template comparisons needed per tree is given by $2log_2(\mathcal{S}) - 1$ (two comparisons per level). Splitting the same number of subjects to multiple trees $\mathcal{T}$, the number of template comparisons needed is

$$\mathcal{C} = \mathcal{T} * \left(2 * log_2\left(\frac{\mathcal{S}}{\mathcal{T}}\right)\right). \tag{6.4}$$

Note figure 6.3, for an example with $\mathcal{S} = 512$, 18 comparisons are needed in a single-tree environment, while using two trees nearly doubles the number of comparisons needed and using 64 trees nearly invalidates the Bloom filter tree approach. In the basic setup, every tree is traversed and there is no approach to skip trees that unlikely contain the searched template.

[4]It isn't possible to generally state a threshold when a child node is too populated since it depends on the configuration of the Bloom filter size and the number of bit errors in the raw binary templates.

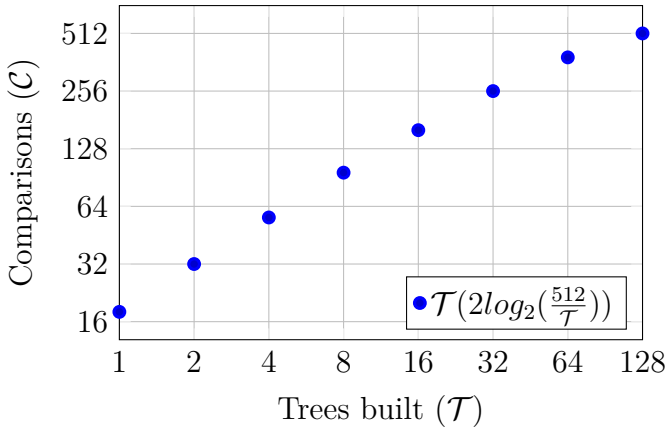


Figure 6.3: Number of needed comparisons ($\mathcal{C}$) per lookup for $\mathcal{S} = 512$ with $1, \ldots, \mathcal{T}$ trees.

6.3.1 Tree selection

Recall that the additional workload introduced by employing multiple trees reduces the workload-reducing effect of the Bloom filter-indexing. The estimated number of template comparisons ($\mathcal{C}$) for single- and multi-tree environments is summarized in equation 6.5.

$$\mathcal{C} = \begin{cases} 2 * log_2(\mathcal{S}) - 1 & \text{if } \mathcal{T} = 1 \\ \mathcal{T} * \left(2 * \left(log_2\left(\frac{\mathcal{S}}{\mathcal{T}}\right) - 1\right)\right) & \text{if } 1 < \mathcal{T} < \frac{\mathcal{S}}{2} \\ \mathcal{S} & \text{if } \mathcal{T} \geq \frac{\mathcal{S}}{2} \end{cases} \tag{6.5}$$

A naïve approach to reduce the impact of additional trees would be to stop traversing the trees, in a one-to-first search manner, if an appropriate candidate has been found. However, this naïve approach only reduces the number of template comparisons needed by factor 2 on average[5]

$$\mathcal{C} = \frac{\mathcal{T}}{2}\left(2 * \left(log_2\left(\frac{\mathcal{S}}{\mathcal{T}}\right) - 1\right)\right). \tag{6.6}$$

A more efficient strategy is presented in [DRB17]. The idea is to add a pre-selection step, selecting the t most promising trees with $t \ll \mathcal{T}$. To determine the most promising trees, the query template is compared with the root node of each tree. Only the t trees with the highest similarity between the root node and query template are then fully traversed. This concept is formally described in listing 4.

Note equation 6.7, using this strategy greatly reduces the additional workload introduced by the additional trees.

$$\mathcal{C} = \begin{cases} 2 * (log_2(\mathcal{S}) - 1) & \text{if } \mathcal{T} = 1 \\ \mathcal{T} + t * \left(2 * \left(log_2\left(\frac{\mathcal{S}}{\mathcal{T}}\right) - 1\right)\right) & \text{if } 1 < \mathcal{T} < \frac{\mathcal{S}}{2} \end{cases} \tag{6.7}$$

However, as shown in figure 6.4, the pre-selection scheme loses its efficiency when the number of trees $\mathcal{T}$ approaches the number of templates $\mathcal{S}$: The generated overhead by comparing the

[5] I has to be noted, that this is only true for genuine transactions, for impostor attempts, the workload is further reduced.

Algorithm 4 Improved loop-logic for multiple Bloom filter trees.

```
1:  function LOOKUP(Trees, t, Query, Threshold)
2:      rootScored ← empty list
3:      for all Trees as root do
4:          rootScored ← similarity(root, Query) + rootScored
5:      end for
6:      SortAscending(rootScores)
7:      candidates ← empty list
8:      for i = 0 . . . t do
9:          candidates ← TraverseTree(rootScored[i], Query, Threshold)
10:     end for
11:     return candidates
12: end function
```

query template to the root nodes starts to overweight, thus reducing the overall workload reduction.

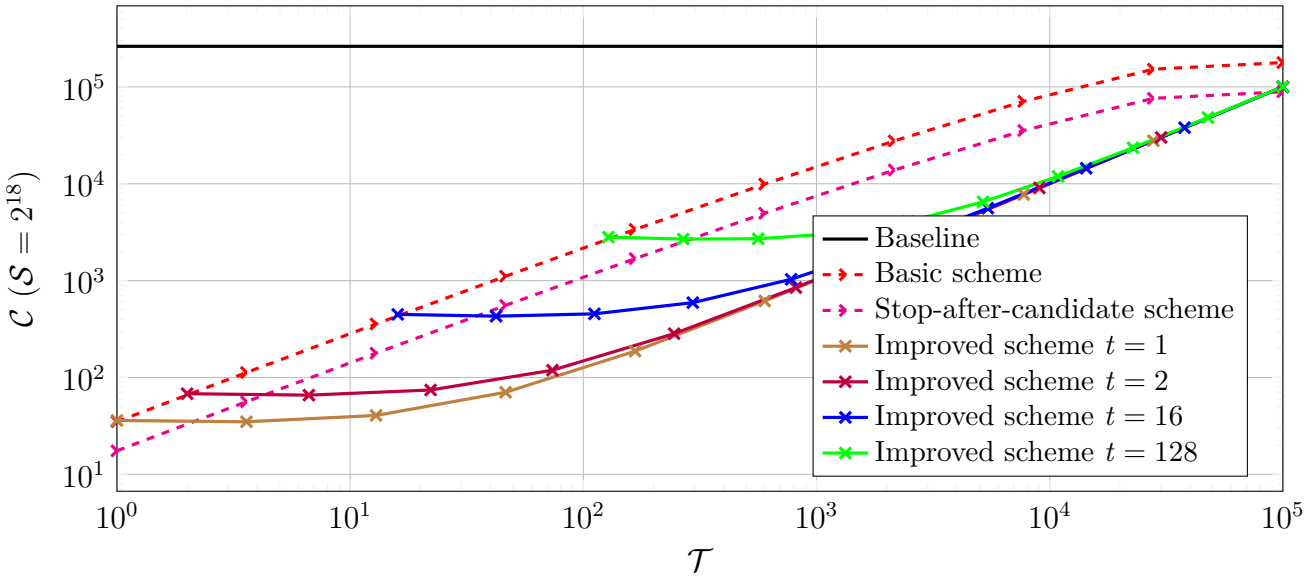


Figure 6.4: Number of needed template comparisons ($\mathcal{C}$) per lookup with the pre-selection scheme.

6.3.2 Quick traversal direction decision

Recall (section 6.2.3), in the basic system each traversal direction decision requires two comparisons, thus yielding $\mathcal{C} = 2 * log_2(\mathcal{S}) - 1$. With the usage of the assumption stated in section 6.2.3, the similarity scores increases per level (for queries with their associated template included in a tree) when traversing a tree, the number of template comparisons needed can be reduced to one per level, thus $\mathcal{C} = log_2(\mathcal{S}) - 1$. Instead of comparing both children per node, only one (e.g. the left/first) children is compared. When the similarity score has not increased compared to the previous, the other children is selected as next node; if the score has increased, the com-

pared node is selected. Thus, a second[6] comparison is only needed if the similarity score has not increased. Algorithm 5 formally shows the improved direction decision strategy. On average,

Algorithm 5 Tree traversal for a single Bloom filter tree with single comparison strategy.

```
1:  function TraverseTree(Node, Query, Threshold, LastScore)
2:      if Node[left] ≠ nil and Node[right] ≠ nil then
3:          scoreLeft ← similarity(Node[left], Query)
4:          if scoreLeft > LastScore then
5:              return TraverseTree(Node[left], Query, Threshold, scoreLeft)
6:          else
7:              scoreRight ← similarity(Node[right], Query)
8:              return TraverseTree(Node[right], Query, Threshold, scoreRight)
9:          end if
10:     end if
11:     score ← similarity(Node[this], Query)
12:     if score > Threshold and score > LastScore then        ▷ Second condition optional
13:         return Node[this]
14:     end if
15:     return nil
16: end function
```

the second comparison is needed for every second level, yielding $\mathcal{C} = \frac{3}{2} log_2(\mathcal{S}) - 1$.
Finally, it has to be noted that this improvement is not feasible if the assumption of increasing similarity scores (for queries with their associated template included in a tree) is not satisfied.

6.4 Tree root emulation and random template emulation

It is crucial to find a suiting tree-to-template ratio to receive an efficient workload reduction: using an insufficient number of trees increases the FNIR[7] and is difficult to detect at runtime. For this purpose, it is necessary to emulate the root node of a tree and random templates. In [DRB17], Drozdowski presented an approach to mathematically emulate random templates, thus emulating root nodes of trees. By calculating the overlap between a emulated random template and the emulated root node, the author is able to estimate the number of trees to build for a database with $\mathcal{S}$ templates. With a few adaptations, this approach should also be feasible for binary SMR Bloom templates, too.
Recall (section 6.2.1) that a Bloom filter $\mathcal{B}$ is created of a binary SMR block with width $\mathcal{W}$ and height $\mathcal{H}$. By assuming that all set bits in the block are *mutually independent* and are *uniformly distributed*, then:

$$\mathcal{B} = \{x \in \mathbb{N}_0 \mid 0 \leq x \leq 2^{\mathcal{H}}\},\ |\mathcal{B}| = 1 - \left(1 - \frac{1}{2^{\mathcal{H}}}\right)^{\mathcal{W}} \tag{6.8}$$

[6] Refer to appended section A.1 for a excursus why a second comparison is needed.

[7] It is possible to reach FNIR values that render the biometric system useless: With one or more levels overflown, finding the right template is more a fluke than a hit.

Figure 6.5: Spectral mask-bit (figure 5.4c) applied to spectral sign-bit (figure 5.4b) of the SML in figure 5.4a as introduced in section 6.2.1.

Observe figure 6.5, where the spectral mask-bit has been applied to the sign-bit for the Bloom filter transformation. It is clearly visible that for the SMR there is a very high dependency between individual columns and the bits set aren't distributed uniformly. To allow for this deviation, let ε denote the difference between the expected number of duplicate columns in a binary SMR and randomly-generated, mutually-independent columns:

$$\mathcal{B} = \{x \in \mathbb{N}_0 \mid 0 \leq x \leq 2^{\mathcal{H}}\},\ |\mathcal{B}| = 1 - \left(1 - \frac{1}{2^{\mathcal{H}}}\right)^{\mathcal{W}} - \varepsilon \tag{6.9}$$

There is no general value for ε since its value depends on parameters such as block size, the dataset itself, SMR configuration, SMR binarisation configuration and has small variations between different templates.
The concept can be extended for root nodes of the Bloom filter tree, too. Recall (section 6.2.2) that a Bloom filter tree root node (here $\mathcal{R}_\mathcal{K}$) is a union of its (here $\mathcal{K}$) child nodes, thus:

$$\mathcal{R}_\mathcal{K} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots \cup \mathcal{B}_\mathcal{K},\ |\mathcal{R}_\mathcal{K}| = \left(1 - \left(1 - \frac{1}{2^{\mathcal{H}}}\right)^{\mathcal{K}*(\mathcal{W}-\varepsilon)}\right) * 2^{\mathcal{H}} \tag{6.10}$$

At this point, it is possible to approximately emulate random templates, thus emulating random impostor templates and root nodes of Bloom filter trees. By estimating the overlap $\mathcal{O}$ of the tree root node (respective nodes) and an random impostor Bloom filter template (equation 6.11), it is possible to qualify a statement about the feasibility of the current configuration.

$$\mathcal{O} = |\mathcal{R}_\mathcal{K} \cap \mathcal{B}| \tag{6.11}$$

The expected overlap outcome follows a hypergeometric distribution with denoting the number of overlapping items in range $1 \ldots |\mathcal{B}|$:

$$P(\mathcal{O} = o) = \frac{\binom{|\mathcal{B}|}{o}\binom{2^{\mathcal{H}}-|\mathcal{B}|}{|\mathcal{R}_\mathcal{K}|-o}}{\binom{2^{\mathcal{H}}}{|\mathcal{R}_\mathcal{K}|}} \tag{6.12}$$

The mean of that distribution Θ (equation 5) can be used as a single number metric.

$$\Theta = |\mathcal{R}_\mathcal{K}| * \frac{|\mathcal{B}|}{2^{\mathcal{H}}} \tag{6.13}$$

Refer to [DRB17] for a explanation how to interpret the introduced Θ.

6.5 Summary

In the first two sections of this chapter, the fundamental details of a Bloom filter-based biometric system have been outlined, partially based on the original article in which this approach was proposed [RBBB15]. The third section outlines the state-of-the-art for the Bloom filter-indexing approach by Drozdowski [DRB17]. All sections include minor adoptions to enable the approach for the binary SMR.
How well the adoptions and the Bloom filter itself perform in identification scenarios is subjects to the experiments introduced in section 8.4 and evaluated in section 9.4.

Chapter 7

Spectral Minutiae CPCA-Tree indexing

In this chapter, a second workload reduction approach based on the SMR-CPCA is introduced. The system basics are adopted from chapter 6.

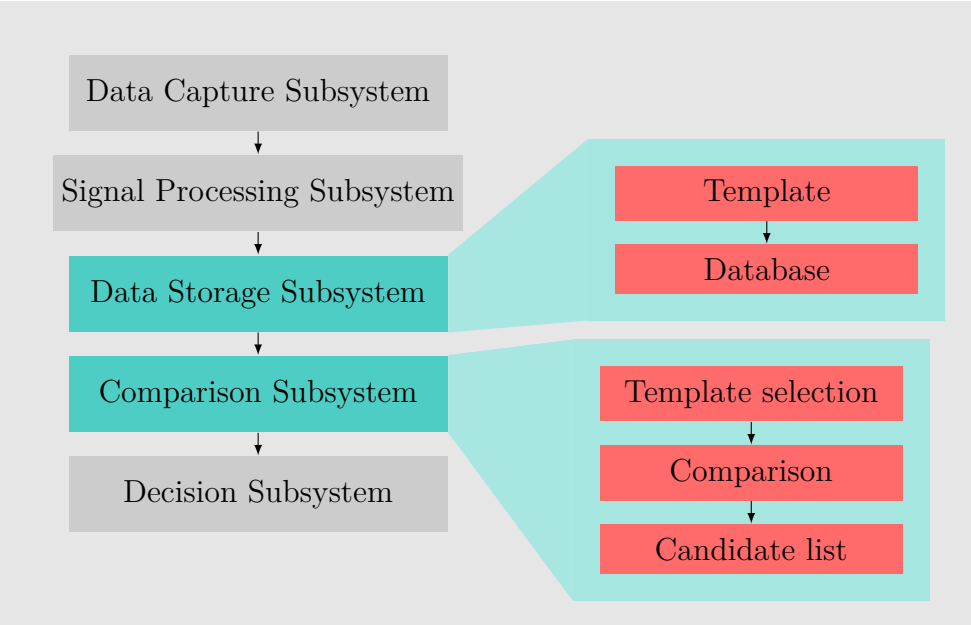


Figure 7.1: Spectral Minutiae CPCA-Tree indexing

7.1 System Basics

In contrast to the Bloom filter-based indexing approach, this author proposes a tree indexing scheme based on the SMR-specific CPCA. The CPCA already proved its feature-reduction, thus also workload-reduction, capabilities without impairing the recognition performance[1] [XVKA09]. The basics of the SMR-CPCA are already introduced in section 5.4.1. However, for this chapter, a closer observation of the SMR-CPCA representation is reasonable.

[1]This statement by Xu et al. will also be tested for palm vein (respective vein based) SMR.

7.1.1 Binary SMR-CPCA merging

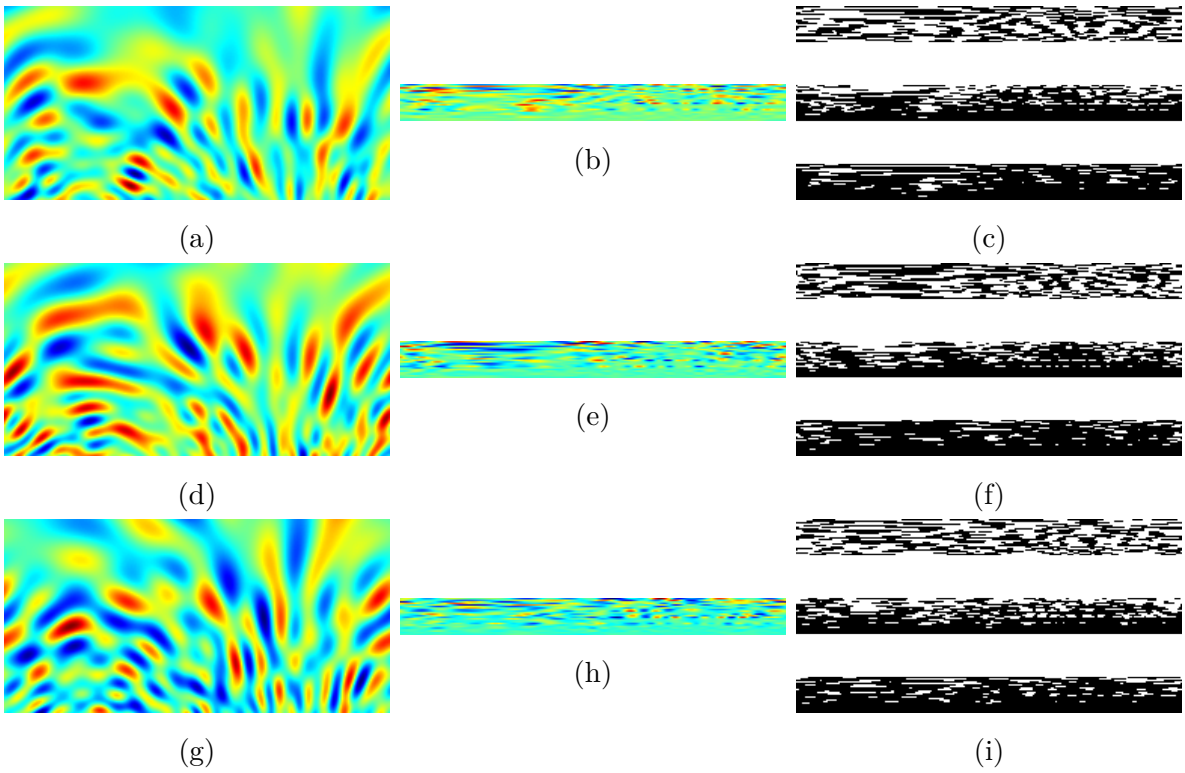
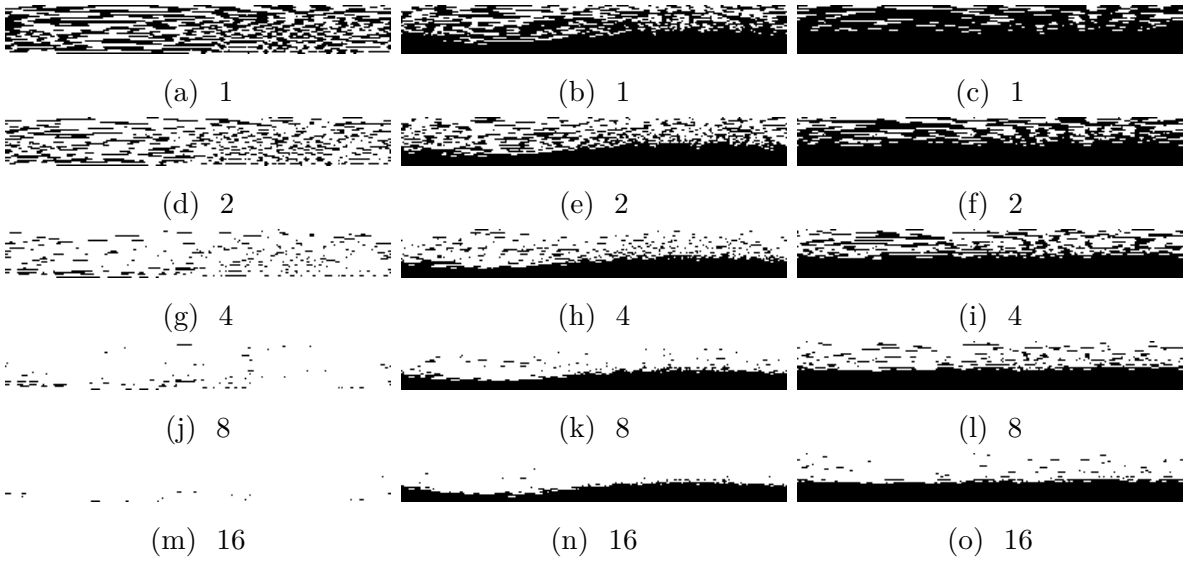


Figure 7.2: SML-CPCA creation from different subjects: (a,d,g) input SML; (b,e,h) CPCA representation; (c,f,i) binary CPCA (top to bottom: sign-bit, mask-bit, applied mask-bit to sign-bit)

Observe figure 7.2, where three different real-valued SMR (here SML) and their (binary) SMR-CPCA are illustrated: the binary SMR-CPCA features a high inter-class variance - all set bits in the binary SMR-CPCA matrices are differently distributed and there are few unanimous bits.

While a high inter-class variance is mandatory and desirable for a good recognition performance, it introduces some difficulties for naïve indexing schemes that rely on trivial stacking or merging. When merging SMR-CPCA, the population count rises quickly, thus diminishing the descriptive value. *In other words, the bits set in a binary SMR-CPCA have few bit collisions with SMR-CPCA from other subjects repectively from other biometric instances.* The problem can easily be explained visually: in figure 7.3, the bit-matrices for 1, 2, 4, 8 and 16 (randomly chosen) merged SMR-CPCA are shown and the percent of bits set are presented in table 7.1. Observe the quickly rising population count for the sign-bit (figure 7.3 left column) and mask-bit (figure 7.3 centre column). Already at 4 merged SMR-CPCA, the sign-bit and mask-bit have passed a population count where no descriptive value is left. If the mask-bit is applied to the sign-bit and the result (applied-bit) is merged (figure 7.3 right column), then about 8 SMR-CPCA can be merged without losing too much information, thus keeping the descriptive value. These conclusions will be further analysed in the experiments described in section 8.4.



(a) 1 (b) 1 (c) 1

(d) 2 (e) 2 (f) 2

(g) 4 (h) 4 (i) 4

(j) 8 (k) 8 (l) 8

(m) 16 (n) 16 (o) 16

Figure 7.3: SMR-CPCA (SML-CPCA) sign-bit (left), mask-bit (center) and applied mask-bit to sign-bit (right) for merged SMR-CPCA with 1, 2, 4, 8 and 16 SMR-CPCA (white = set, $\lambda_{min} = 0.1, \lambda_{max} = 0.6, M_{CPCA} = 24$).

No. merged SMR-CPCA	Bits set		
	sign-bit	mask-bit	applied-bit
1	50.3%	43.5%	21.4%
2	74.5%	67.9%	38.1%
4	93.3%	89.3%	61.4%
8	99.3%	98.8%	83.9%
16	99.9%	99.9%	96.6%

Table 7.1: Average rate of set bits in the SMR-CPCA (SML-CPCA) sign-bit, mask-bit and applied mask-bit to sign-bit for merged SMR-CPCA.

7.1.2 Masking out most common bits

Masking out bits that are set in a bulk of all SMR-CPCA templates (here SML-CPCA) is a common-known strategy used to lower the average similarity score of impostor comparison trials: the impostor comparison trials that gain their score from these commonly set bits are more likely to be rejected. However, since the CPCA transformation depends on the training dataset and the data itself, there is no common mask. As a rule of thumb, the upper bits are

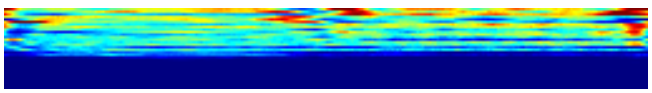


Figure 7.4: Example heat map of set bits generated with 467 SMR-CPCA (SML-CPCA) applied-bit (red = most, blue = least).

more likely to occur in SMR-CPCA than the lower bits. Figure 7.4 shows an example heat map

of 467 (trainig-set size 33) SMR-CPCA generated with the PolyU-Database ([pol], see section 8.1). Observe the upper rows of the heat map: these bits are set most often with a hot-spot in the top-centre and top-right. Most descriptive value is held in the middle rows: these bits are set less often and more likely to differ between multiple SMR-CPCA. In this example scenario, it is recommended to mask out the upper three rows and the red hot-spots in the top-centre and right. The experiments in 8.4 will further analyse the advantages and disadvantages of this strategy.

7.1.3 Template Transformation

For the proposed SMR-CPCA binary search tree (CPCA-Tree) approach, no additional template transformation is needed. The SMR is merely transferred in its binary CPCA representation and cut to M_{CPCA} rows. Refer to section 5.4.1 for details.

7.1.4 Tree Construction

The tree construction for the proposed CPCA-Tree approach follows the same scheme as the Bloom filter tree of section 6.2.2 with the Bloom filter templates replaced by SMR-CPCA templates. There are two different methods of representing the binary SMR-CPCA for the CPCA-Tree:

SMR-CPCA-Components (SMR-CPCA-C) Recall (section 5.5 and 5.4.1), there are two different strategies when transforming a SMR in its binary form. The spectral binarisation method yields two binary matrices: a sign-bit and a mask-bit. These two matrices represent the two components of the SMR-CPCA-Components (SMR-CPCA-C) representation.

SMR-CPCA-Applied (SMR-CPCA-A) Instead of keeping both binary components, the mask-bit is applied to the sign-bit, yielding the applied-bit (see section 7.1.1). The applied-bit matrix represents the single component representation SMR-CPCA-Applied (SMR-CPCA-A).

An advantage of the SMR-CPCA-A compared to the SMR-CPCA-C is its trivial adoption to the already-introduced Bloom filter tree: it follows the same comparison method (hamming distance), tree construction and traversal as the Bloom filter template. Most of the improvements by [DRB17] (see section 6.3) are also applicable. Therefore, refer to section 6.2.2 for further details on the tree construction with the SMR-CPCA-A. The SMR-CPCA-A, compared to the SMR-CPCA-C, loses the ability of merging the auxiliary data for the fractional hamming distance which limits the comparison to the stable bits set in booth templates. Thus the hamming distance is used for the comparison which compares all set bits, even if some of them are considered unstable in one of the two templates.

The SMR-CPCA-C requires a different comparison method (recall section 5.6.2; fractional hamming distance) and thus requires a tree with nodes that carry auxiliary data. Furthermore, a strategy is needed to handle the two matrices. In the Bloom filter approach, the root node is a union of its two child nodes, both of which are unions of their child nodes, etc. Again, observe figure 7.3 and table 7.1, by taking the union of e.g. the sign-bit of 4 random binary SMR-CPCA the nodes sign-bit matrix would be already set by 90%. The mask-bit for a union of 4 random binary SMR-CPCA is also highly populated. Therefore, trivially taking the union of both will not allow for identification trees containing more than 8 (4 per root children) templates without significantly losing recognition performance.
This problem with the SMR-CPCA-C leads to an artificial third method: replacing the sign-bit of the SMR-CPCA-C with the applied-bit of the SMR-CPCA-A (and keeping the mask-bit) could help to increase the number of templates for the SMR-CPCA-C-Tree and is further called SMR-CPCA-Mixed (SMR-CPCA-M).

7.1.5 Tree Traversal

The basic tree traversal logic of the CPCA-Tree also follows the same scheme as the Bloom filter tree. For the SMR-CPCA-A variant, the same comparison method and intelligent traversal strategies implemented in the Bloom filter tree can be used. In a SMR-CPCA-C- or SMR-CPCA-M-Tree, the comparison method has to be adapted to the sign/mask-scheme by using the FHD (see section 5.6.2). Since the FHD yields the same output as the original Bloom filter tree comparison method (a similarity value), all intelligent traversal strategies are also applicable.

7.1.6 Configuration

On top of the CPCA configuration, there are two variables to be considered for the CPCA-Tree system: SMR-CPCA height (M_{CPCA}) and tree-count ($\mathcal{T}$). Each of them can be adjusted considering the following limitations.

M_{CPCA} The height of the SMR-CPCA templates depends on the training data and the frequency (λ_{min} and λ_{max}, section 5.2) of both the training data and model data. Further, the M_{CPCA} is limited by $1 \leq M_{CPCA} \leq M$, since $M_{CPCA} = 0$ results in an zero-sized template and increasing the template ($M_{CPCA} > M$) is not possible. A large M_{CPCA} results in an increased template size (thus increased computational workload), and an insufficient M_{CPCA} drops some necessary information. It has to be noted that M_{CPCA} has an artificial threshold defined by the mask-bit of each SMR-CPCA. Again, observe the sign-bits shown in figure 7.3: in this figure, the $M_{CPCA} = 24$ could be further reduced, but increasing the M_{CPCA} would not increase the recognition performance since the mask-bits are not set[2] for rows beyond $M_{CPCA} = 24$.

[2]Compare figure 5.4f

$\mathcal{T}$ The behaviour of $\mathcal{T}$ for the CPCA-Tree equals $\mathcal{T}$ for the Bloom filter tree system and follows the same limitations: too many trees invalidates the workload reduction introduced by the system, while too few trees renders the system useless as a worst case.

The three different template types described in section 7.1.4 aren't considered a configuration property, instead they are implementation variants that are benchmarked in the experiments.

7.2 Template protection properties

The proposed system purely relies on the SMR-CPCA representation. Like mentioned in section 2.2.4, a biometric system and their templates should address several privacy protection requirements. This section inspects the three requirements unlinkability, renewability and irreversibility for the SMR-CPCA.

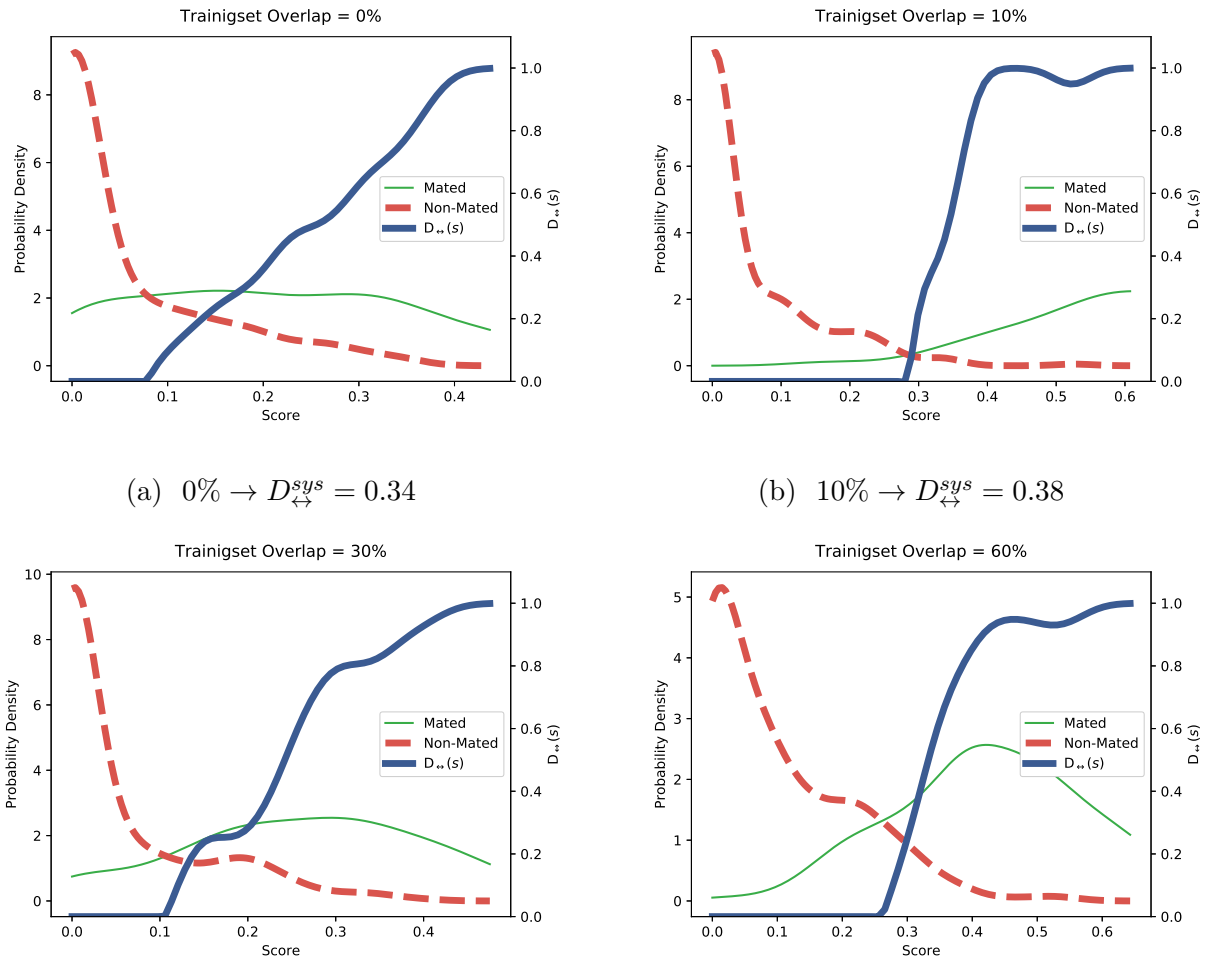
7.2.1 Unlinkability

The SMR-CPCA template depends on the training set of the PCA used for the CPCA. It can safely be stated that two different databases will use two different CPCA training sets.
To analyse the unlinkability properties of the CPCA approach, the metrics $D_{\leftrightarrow}^{sys}$ and $D_{\leftrightarrow}(s)$ proposed by Gomez-Barrero et al. [GBRG+16] are used. $D_{\leftrightarrow}^{sys}$ describes the overall system linkability and $D_{\leftrightarrow}(s)$ describes the linkability of a single similarity score both measures yield a value on a scale from 0 (fully unlinkable) to 1 (fully linkable). For the following, reasoning $D_{\leftrightarrow}^{sys}$ is rated as listed in equation 7.1.

$$\text{Linkability} = \begin{cases} \text{unlinkable} & \text{if } 0 \leq D_{\leftrightarrow}^{sys} < 0.15 \\ \text{semi-unlinkable} & \text{if } 0.15 \leq D_{\leftrightarrow}^{sys} < 0.3 \\ \text{semi-linkable} & \text{if } 0.3 \leq D_{\leftrightarrow}^{sys} < 0.6 \\ \text{linkable} & \text{if } 0.6 \leq D_{\leftrightarrow}^{sys} \leq 1 \end{cases} \tag{7.1}$$

Training two CPCA with disjoint training sets (for two different databases) yields two different transformations (referred to as CPCA_A and CPCA_B). For this unlinkability experiment, CPCA_A is used to generate SMR-CPCA_A of SMR_1 from instance I_1 and CPCA_B is used to generate SMR-CPCA_B of SMR_1 from instance I_1. *In other words: same instance, same palm vein, transformed using two differently trained CPCA*. As reference, non-mated templates are generated with CPCA_B to test whether the scores between the mated SMR-CPCA_A and SMR-CPCA_B are distinguishable from the scores of the non-mated templates. Two different scores are obtained during the experiment:

Mated scores: scores obtained from the comparison of two templates, extracted from samples obtained from a *single instance* I_1 and two *different transformations* CPCA_A and CPCA_B.



(a) $0\% \rightarrow D_{\leftrightarrow}^{sys} = 0.34$

(b) $10\% \rightarrow D_{\leftrightarrow}^{sys} = 0.38$

(c) $30\% \rightarrow D_{\leftrightarrow}^{sys} = 0.48$

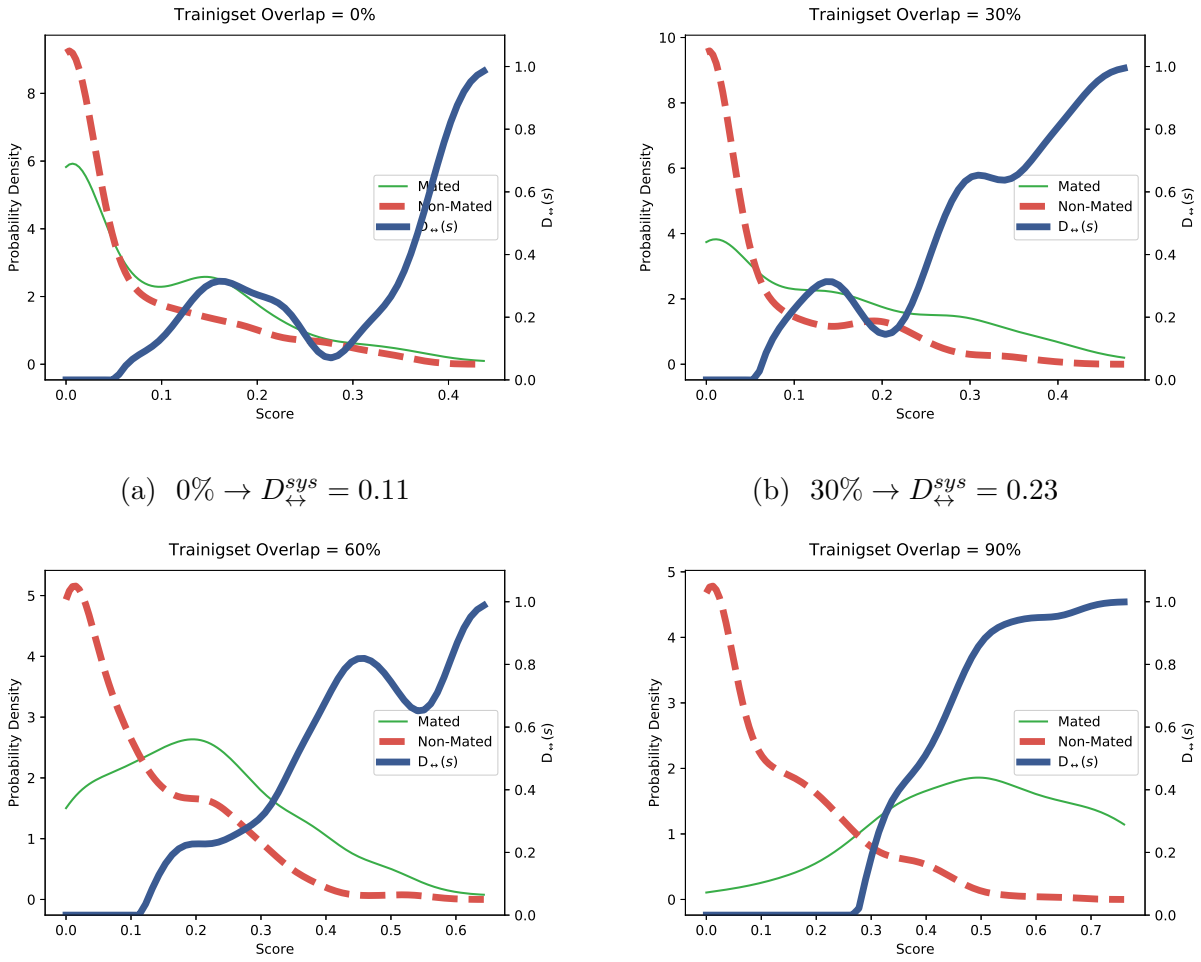
(d) $60\% \rightarrow D_{\leftrightarrow}^{sys} = 0.59$

Figure 7.5: Unlinkability $D_{\leftrightarrow}^{sys}$ plots for same images at (a) 0%, (b) 10%, (c) 30% and (d) 60% CPCA training set overlap.

Non-Mated scores: scores obtained from the comparison of two templates, extracted from samples obtained from two *different instances* I_1 and I_2 and two *different transformations* CPCA_A and CPCA_B

This simulates the scenario when an attacker received two protected templates enrolled in two applications (A and B), and tries to decide whether the were extracted from the same instance, e.g. when he tries to track the subject in different databases.
However, it is possible that even for different databases there are some small intersections in the training dataset. To cover this scenario figure 7.5 includes the plots ($\mathcal{S} = 400$) for a training set overlap of 0%, 10% and 30% with an additional bad-case overlap of 60%. For 0% (7.5a) and 10% (7.5b) overlap a $D_{\leftrightarrow}^{sys} < 0.4$ is achieved and is already considered semi-, but not fully-, linkable. As could be expected with a strong overlap in the training sets, the templates are more linkable. In particular, a $D_{\leftrightarrow}^{sys}$ above 0.5 (semi linkable, nearly linkable) is reached with an overlap of 60% (7.5d).

There is one additional hurdle for an attacker that is not covered in the previous analysis. Two different databases are usually generated by two different image capturing subsystems, feature extraction subsystems and different images of one subject. When analysing $D_{\leftrightarrow}^{sys}$ for



(a) $0\% \rightarrow D_{\leftrightarrow}^{sys} = 0.11$

(b) $30\% \rightarrow D_{\leftrightarrow}^{sys} = 0.23$

(c) $60\% \rightarrow D_{\leftrightarrow}^{sys} = 0.24$

(d) $90\% \rightarrow D_{\leftrightarrow}^{sys} = 0.57$

Figure 7.6: Unlinkability $D_{\leftrightarrow}^{sys}$ plots for different images of the same instance at (a) 0%, (b) 30%, (c) 60% and (d) 90% CPCA training set overlap.

different images but the same image capturing and feature extraction subsystems, the achieved values for 0% (7.6a) and 30% (7.6b) are both below 0.3, and thus are considered semi-unlinkable. Even for an overlap of 60% $D_{\leftrightarrow}^{sys} = 0.24$ (7.6c, semi unlinkable) and for an overlap up to 90% $D_{\leftrightarrow}^{sys} = 0.57$ (7.6d, semi linkable) is achieved. It is safe to assume that with different SMR configurations in other feature extraction subsystems, $D_{\leftrightarrow}^{sys}$ would further decrease and strengthen the unlinkability.

7.2.2 Renewability

The renewability of SMR-CPCA templates can be reasoned the same way as the unlinkability: the SMR-CPCA template depends on the training set of the PCA used for the CPCA. Therefore, it is possible to renew SMR-CPCA templates in certain boundaries.

Observe the similarity-score histogram in figure 7.7. Training two CPCA (refered to as old

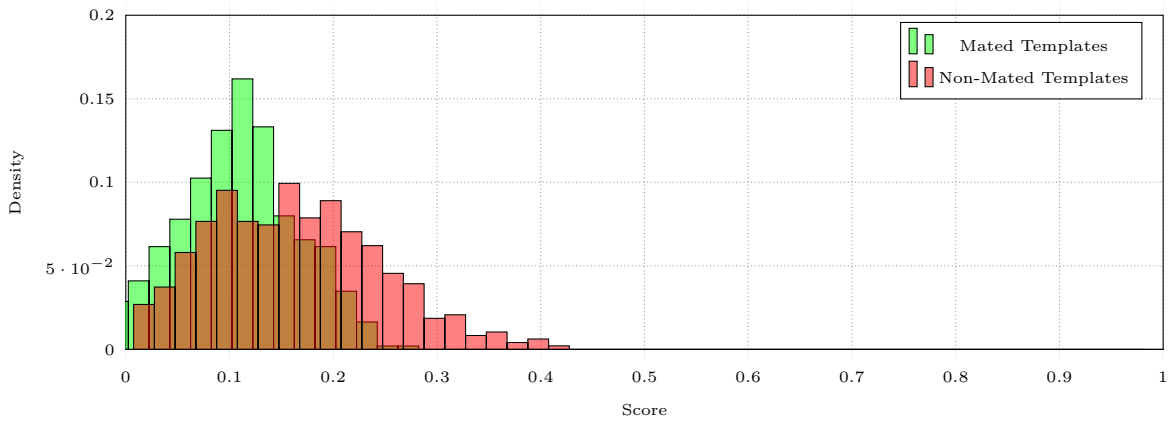


Figure 7.7: Histogram of similarity scores for mated old and renewed SMR-CPCA templates and non-mated old and renewed SMR-CPCA templates.

and renewed) with disjoint training sets yields two different templates: when comparing two SMR-CPCA of one instance generated with the old and the renewed CPCA, the score cannot be distinguished between scores generated by comparing the old SMR-CPCA with renewed SMR-CPCA of other instances.

Another hint at the renewability capability of the SMR-CPCA can be derived from the analysis of the unlinkability in the previous section. Observe the unlinkability for e.g. 30% and 60% training set overlap in figure 7.6. When acquiring new images and not using a fully disjoint training set for the new CPCA to renew the templates (e.g. after a data leakage), the unlinkability is still given to some extent. By contrast, if the new template is not linkable to the old template, the renewability is also given.

7.2.3 Irreversibility

The SMR itself can be stated irreversible[3] (it cannot be transferred back to its minutiae), although a discourse to prove the irreversibility of an SMR template is beyond the scope of this thesis. However, the SMR can be defined as a raw template, since the renewability is not given and the unlinkability is only given with e.g. permutations. Therefore, it is necessary to analyse the irreversibility properties of the CPCA transformation, to determine if the SMR-CPCA can be stated irreversible.

[3] [XV10c] hints at the irreversibility of the SMR but never actually states nor proves it.

In [XVKA09], the CPCA is given by

$$\mathcal{X}_{\mathrm{CPCA}} = \tilde{\mathbf{U}}_{\mathrm{Z}}^{\mathrm{T}} \mathcal{X} \tag{7.2}$$

where $\mathbf{U}_{\mathrm{Z}}$ is the transformation matrix used to project the SMR spectrum $\mathcal{X}$ to its SMR-CPCA $\mathcal{X}_{\mathrm{CPCA}}$. By only keeping the first M_{CPCA} columns of $\mathbf{U}_{\mathrm{Z}}$, $\tilde{\mathbf{U}}_{\mathrm{Z}}$ is received, thus yielding the $M_{CPCA} \times N$ sized SMR-CPCA. The CPCA can trivially reversed[4] by

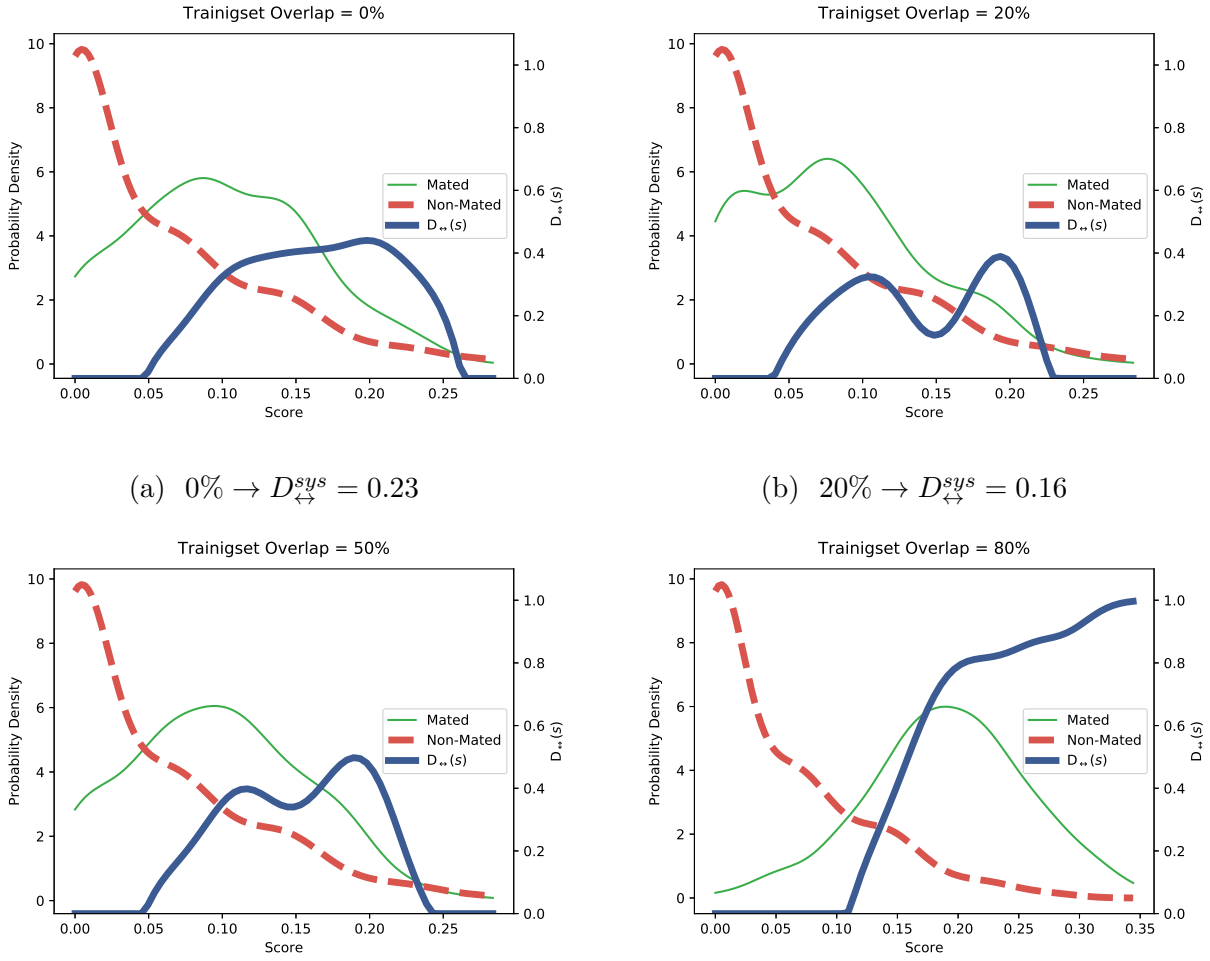
$$\mathcal{X}_{\mathrm{REV}} = \tilde{\mathbf{U}}_{\mathrm{Z}} \mathcal{X}_{\mathrm{CPCA}}. \tag{7.3}$$

This introduces a security risk since the reversibility renders both unlinkability and renewability useless. Therefore it is necessary to treat $\mathbf{U}_{\mathrm{Z}}$ (and $\tilde{\mathbf{U}}_{\mathrm{Z}}$) as classified key matrix for the CPCA system that should not be saved as plain data.

It has to be noted that the trivial reversibility is aggravated for the attacker in a real-world scenario, since he has to undertake additional steps to sufficiently reconstruct a SMR template. The SMR-CPCA templates should be stored in their normalised form. Since the normalisation is applied after the CPCA transformation, the dynamic range of the CPCA matrix multiplication (usually $[-3, 3]$) shrinks during the normalisation process (normalised to $[-1, 1]$). When reversing the matrix multiplication - which would normally transform the pre-normalised SMR-CPCA dynamic range to the normalised SMR - the resulting SMR is also reduced in its dynamic range. Comparing the reversed SMR with the original SMR using the direct comparison method therefore yields an average similarity score of $\sim$0.4. However, the original structure of the SMR is retained during the transformation process. Therefore, applying the normalisation on the reconstructed SMR yields SMR that achieve an average similarity score of $\sim$0.998 when compared to the original SMR. Reconstruction of normalised SMR-CPCA templates is not perfect but more than sufficient for tracking or other purposes.

Facing the general reversibility, a possible scenario is that an attacker was able to receive the SMR-CPCA template but could not receive $\mathbf{U}_{\mathrm{Z}}$ since it is strongly encrypted. The attacker can train an approximated $\mathbf{U}_{\mathrm{Z}}$ (referred to as $\mathbf{U}'_{\mathrm{Z}}$) by training a CPCA with random SMR templates that feature the same characteristics (M, N, SML/SMC) and approximate configurations (λ_{min}, λ_{min}, σ).
Again, the unlinkablity measure $D_{\leftrightarrow}^{sys}$ from [GBRG+16] is used to analyse the scenario. Observe figure 7.8, the linkability for the reconstructed SMR (here SML) while using $\mathbf{U}'_{\mathrm{Z}}$ by 0% CPCA training set overlap is with 0.23 still considered semi-unlinkable. In scenarios where the attacker was able to acquire templates used in the training set of the CPCA used to create $\mathbf{U}_{\mathrm{Z}}$, he is able to generate a $\mathbf{U}'_{\mathrm{Z}}$ with an overlap. For $\mathbf{U}'_{\mathrm{Z}}$ with 50% overlap, the SMR-CPCA

[4]The error introduced in double precision implementations is hardly detectable. For single precision implementations, the error is noticeable but influences a comparison between $\mathcal{X}_{\mathrm{REV}}$ and $\mathcal{X}$ by less than 0.5% in experiments.



(a) $0\% \rightarrow D_{\leftrightarrow}^{sys} = 0.23$

(b) $20\% \rightarrow D_{\leftrightarrow}^{sys} = 0.16$

(c) $50\% \rightarrow D_{\leftrightarrow}^{sys} = 0.24$

(d) $80\% \rightarrow D_{\leftrightarrow}^{sys} = 0.59$

Figure 7.8: Unlinkability $D_{\leftrightarrow}^{sys}$ plots between reconstructed SML and original SML at (a) 0%, (b) 20%, (c) 50% and (d) 80% CPCA training set overlap.

is still considered semi-unlinkable. When reaching up to 80% overlap, $D_{\leftrightarrow}^{sys} = 0.6$ is considered semi-linkable. Exceeding 80% overlap quickly rises $D_{\leftrightarrow}^{sys}$ and 100% overlap results in $D_{\leftrightarrow}^{sys} = 1$. Note the linkability of 0.16 for 20% overlap is smaller than the linkability of 0% overlap. This behaviour[5] was observable in all intervals of this experiment[6].

These linkability scores were generated using training sets and templates with equal SMR characteristics and configurations, and therefore they represent perfect conditions for an attacker since he is using a perfect copy of the system. Again, it is safe to assume that the linkability decreases when different characteristics and configurations are used in the training

[5]Between 0% and 50% overlap, one or two linkability scores peaked (where smaller) compared to linkability scores generated with less overlap.

[6]This author is not sure about the origin of this behaviour. One possible explanation would be that some SMR have a higher impact on the CPCA training, thus when these are included in both training sets, $\mathbf{U}'_Z$ and $\mathbf{U}_Z$ become more equal; when the overlap increases, the other SMR even the impact of the high impact SMR.

sets and templates.

7.3 Conclusions

The CPCA-Tree is a novel workload reduction approach exclusively for SMR-based biometric systems. It adopts the same strategy and mechanism of the already-proven Bloom filter-indexing approach [RBB13, RBBB15, DRB17].

In the first section of the chapter, the basics of the system were outlined. The template transformation uses the CPCA projection to reduce the full SMR feature vector to a on average 80% smaller representation. To construct a single CPCA-Tree, three different methods for representing a template are presented. Which method yields the best retrieval performance will be explored in the experiments. For the tree traversal, all state-of-the-art strategies from [DRB17] are also applicable. Merely the comparison method is adapted to the three different template methods. Finally, the CPCA-Tree features a configuration comparable to the Bloom filter-indexing configuration.

The second section analyses the template protection properties unlinkability, renewability and irreversibility. All three properties were evaluated using the unlinkablity measure $D_{\leftrightarrow}^{sys}$ from [GBRG+16]. While the unlinkability and renewability are given without prerequisites, the irreversability is only given if the transformation matrix $\mathbf{U}_Z$ of the CPCA approach is unreachable for an attacker (e.g. with strong encryption). When the unlinkability, renewability and irreversibility reached with the CPCA transformation are not sufficient, permutation approaches like the presented method in [GBRG+16] are also applicable.

The CPCA-Tree indexing approach will be used as a contender to the already-proven Bloom filter-indexing approach in the workload reduction experiments.

Chapter 8

Experimental Setup

The following sections describe the palm vein data used during this project for experiments, its preparation and a description of how the experiments were conducted.

8.1 Datasets

While there are many publicly-available vascular datasets, only few of them contain palm vein images. This situation is aggravated by the fact that most palm vein images are captured using a self-designed apparatus resulting in noisy and poor-quality images. However, for use during the conceptual and experimental phases of this project, three distinct datasets were selected. The datasets are briefly described below, example images are presented in figure 8.1 and all dataset values are summarized in table 8.1.

PolyU multispectral palmprint Database (PolyU) [pol] At the time of writing, the PolyU dataset is the largest publicly-available palm-print dataset containing NIR palm-print images usable for palm vein recognition known to the author. It comprises images from two sessions in an average of 9 days difference of 250 subjects with 6 images per hand and session, resulting in 6 000 images. The images have a pre-defined and stable ROI. All images have a very low-quality variance and are all equally illuminated. Since the PolyU dataset aims for palm-print recognitionm, it features a high amount of skin texture, which interferes with the vein detection and makes it a challenging dataset for the feature extraction pipeline.

CASIA-MS-PalmprintV1 (CASIA) [cas] Another palm-print targeting dataset featuring NIR palm-print images usable for palm vein recognition. The dataset is built in two sessions, with a difference of more then one month, from 100 subjects with 3 images per hand (i.e. biometric instance) and session, resulting in 1 200 images. All images picture the full geometry of differently-shaped hands without any guidance, thus makes it a prefect dataset for experiments for guidance-less ROI detection. Unfortunately, the image capturing device is a multispectral device, thus missing any visible light filters resulting

(again) in images with a high amount of skin texture. Furthermore, the illumination variance is very high, since the capturing device used specular LEDs. Therefore, small hand movements result in large illumination (shadows, hotspots) variance. Some hands were bent or crooked during the capturing process, aggravating the feature extraction or resulting in failure-to-acquire events. In summary, it is the most challenging dataset, which is expected to yield very low performance.

PUT Vein Database (PUT) [KK11a] Additional small dataset with three sessions, with a difference of at least one week, including 50 subjects with 4 images per hand and session, resulting in 1 200 images with a pre-defined ROI. The ROI suffers from a high translation variance but the images feature almost no skin texture or other noise[1].

In the PolyU dataset it is not possible to link the left hand instance and the right hand instance to one subject. The CASIA and the PUT datasets link left and right hand instances to subjects. Since it isn't possible to link left and right hand instances to subjects by just viewing at the palm vein images, it is assumed in the following sections and chapters that no two instaces are from one subject and therefore, every subject is represented by only one instance in the datasets.

Dataset	Instances	Images	Excluded Instances[§]	Excluded Images[§]	Resolution	ROI	Quality
PolyU	500	6 000	6	50	128×128 *px*	128×128 *px*	Moderate
CASIA	200	1 200	28	30	768×576 *px*	max. 128×128 *px*	Low
PUT	100	1 200	0	0	512×384 *px*[*]	272×176 *px*[†]	Moderate[‡]

[*] Downscaled off-line to reduce resource usage (disk and computation).
[†] Original 340×240 *px* downscaled by factor 0.8.
[‡] High image quality but no ROI alignment possible.
[§] Refer to section 8.3 for a detailed explanation of the exclusion criteria.

Table 8.1: Dataset overview.

(a) PolyU (b) CASIA (c) PUT

Figure 8.1: Example images from each dataset.

[1] It could be possible that the images are heavily pre-processed by the image-capturing device or some other subsystem in terms of contrast and colour gradients. All images are differently saturated and coloured, which hints at some pre-preprocessing steps.

8.2 Palm Vein Feature Detection Pipeline

The complete biometric processing chain (pipeline) has already been presented in chapter 4. Since the data of the selected datasets features completely different characteristics, the pipeline has to be adjusted to the three datasets.

CASIA The CASIA dataset uses the full featured pipeline presented in chapter 4.

PolyU Since the PolyU dataset already features a pre-defined ROI, the ROI detection step (section 4.2) is skipped. All following steps are used as presented.

PUT With the already pre-defined ROI of the PUT dataset, again no ROI detection step is needed. However, the images are further tailored to remove the background, thus removing noise. Refer to the appended figure A.1 for a visualisation of the ROI used.

8.3 Excluded Images

Unfortunately, not all images or subjects could be used for the experiments. This section outlines the reasoning behind the decisions for excluding images.

8.3.1 Capturing Errors

As described in section 2.1, capturing vascular images is not a trivial task. The four different failure-to-acquire types observed through the datasets are outlined below and example images[2] are presented in figure 8.2.

No visible veins For some instances, no veins are visible at all. Even with manual image manipulation, no (real) veins were extractable. This is mostly due to a poorly-chosen NIR wavelength and when the veins are too deep inside the skin. In all cases, this affects the whole subject, resulting in an exclusion of all images of that instance.

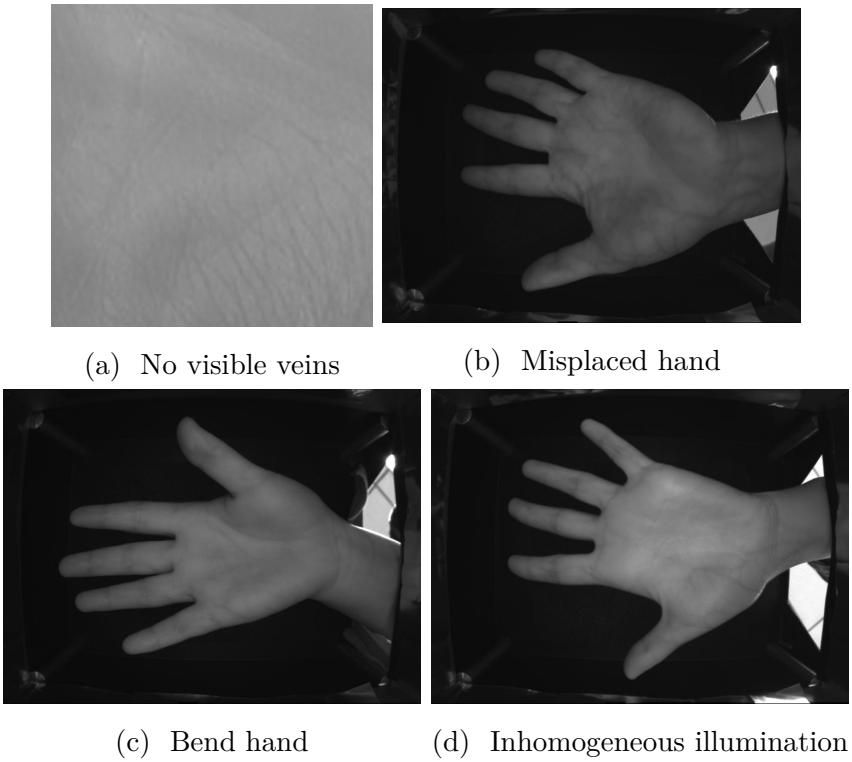
Missing/additional veins In some occasions, the pulses can make some veins visible, if the image was taken at the exact time of the heartbeat that are not visible between heartbeats. Several images were found where the minutiae count fluctuated more than 50% due to the pulses. To remove as few images as possible, the images with a high minutiae count (due to the heartbeat) were identified and removed as more images between heartbeats than images on a heartbeat are expected.

Misplaced hand Only affecting the CASIA dataset, some hands were shifted or tilted during during the image-capturing process. This heavily warps the venous network and results in false negative identifications.

[2] Missing/additional veins are difficult to illustrate using only one example image. The full illustration would exceed the scope of this section.

Bend hand Again only affecting the CASIA dataset, some hands were bent or crooked during image capturing. This mostly created skin folds and shadows that were detected as veins while covering the real veins. Consequently, bent hands were falsely mated rather than mated with their enrolled sample.

Inhomogeneous illumination Again only affecting the CASIA dataset, the inhomogeneous illumination created hotspots and shadows depending on the positioning of the hand. In some images, the shadows and hotspots displaced the real veins, resulting in falsely-mated templates.



(a) No visible veins (b) Misplaced hand

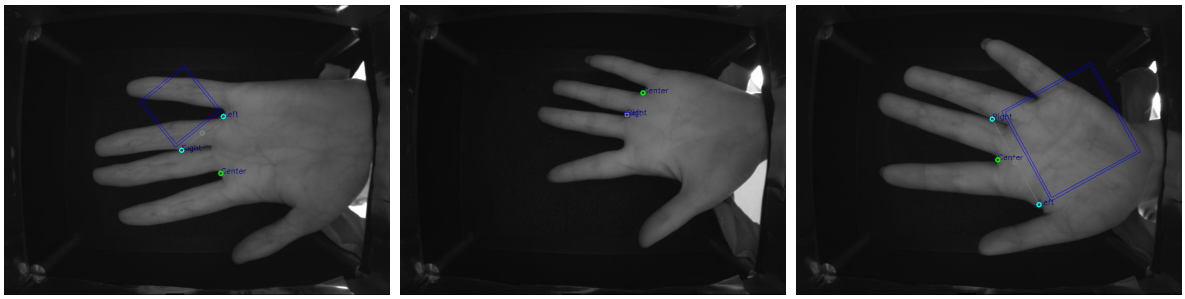
(c) Bend hand (d) Inhomogeneous illumination

Figure 8.2: Example images for the different failure-to-acquire event types.

Refer to table 8.1 for a summary ofhow many images are excluded for each dataset.
Note: no images of the CASIA dataset were removed due to quality issues. Therefore, the dataset contains all data from misplaced, bent or inhomogeneous illuminated hands. It is expected that this will significantly reduce the recognition performance on this dataset.

8.3.2 Preprocessing Failures

Preprocessing failures can occur as early as the ROI detection. The ROI is the most vulnerable and critical preprocessing step. A misplaced ROI propagates further down the processing chain, is difficult to automatically detect and impairs the overall system performance. Figure 8.3 illustrates the three common ROI detection errors and is explained below. The presented ROI detection approach in section 4.2 tries to determine the rotation of the hand and which



(a) (b) (c)

Figure 8.3: Example images for reasons of a misplaced ROI: (a) wrongly interpreted rotation; (b) one gap detected as two gaps; (c) wrongly associated gaps.

side of the hand is captured. In some cases, this step fails and returns a wrong association of the finger gaps, as illustrated in figure 8.3a. When the segmentation step fails to extract a clean contour, the finger gap finding step detects two gaps at one location. This usually yields an ROI smaller than $10px$, illustrated in figure 8.3b, that is misplaced as well. Lastly, the error illustrated in figure 8.3c is again a result of a falsely detected hand in terms of left and right hand. In this example, the subject has presented the left hand. The ROI detection falsely classifies it as a right hand and the gap association step associated the thumb-index-finger gap as the ring-little-finger gap, etc. Therefore, the determined ROI is too large and thus exceeds the hand perimeters.

Aside from the error illustrated in figure 8.3b, these errors are notoriously difficult to detect. Through a visual inspection, the worst and most obvious failures were found. From the CASIA dataset, 30 images (2.5%) had to be removed to avoid including templates from invalid ROIs in the experiments.
Again, refer to table 8.1 for a summary of how many images are excluded for each dataset.

8.4 Experiments

The dataset has been split into four groups: enrolled, genuine, impostor and training (for the CPCA feature reduction). Table 8.2 shows the number of templates in each group. The conducted experiments are listed below.

8.4.1 Conducted experiments

This section outlines the experiments conducted as practical work for this thesis.

MHD-Baseline The basic implementation, comparing palm vein probes based on minutiae with the modified Hausdorf Distance (MHD) [DJ94]. In an identification, scenario the

Dataset	Enrolled templates	Genuine samples	Impostor samples	Training samples
PolyU	256 (256)	2 816 (256)	2 487 (238)	100 (6)*
CASIA	99 (99)	476 (99)	468 (72)	100 (28)†
PUT	58 (58)	456 (58)	456 (38)	48 (4)

* With added images previously excluded.

† Only ∼4 images (2 per session) used per subject.

Table 8.2: Dataset partitioning overview, subjects in parentheses.

database is searched exhaustively, e.g. every query template (probe) is compared with every enrolled template (reference).

Verification Each template of every subject is compared with another template of the same subject and the score is checked against a threshold. Impostor scores are generated by comparing every template of one subject with all templates of another subjects. A false match is counted if a score of one non-mated comparison trial exceeds the decision threshold, whereas a true match is counted if a score of a mated comparison trial exceeds the decision threshold, and a false non match is counted, if a score of one mated comparison trial does not exceed the decision threshold.

Identification One reference template is enrolled for each subject. All remaining probe templates are compared against the enrolled references. A false positive identification is counted if the highest score of one probe exceeds the threshold on a reference template of a wrong subject (impostor score), whereas a true positive identification is counted if the highest score of one probe exceeds the threshold on an genuine reference template, and a false negative identification is counted if the highest score of one probe does not exceed the threshold.

SMR The same procedure as the baseline experiments, but with the SMR and direct SMR comparison. Since the main focus of this project is identification in large-scale databases, the identification experiments for the SMR are used to find the optimal representation and SMR settings for following experiments. These experiments represent the baseline for the workload-reduction approaches in the following experiments.

Identification As in the baseline identification experiment above, but with the SMR and direct SMR comparison. This setup is used to find the optimal representation and SMR settings.

Verification Supplemental experiment to the identification experiments. Procedure as described in the baseline verification experiments.

Workload reduction by feature reduction After determining of the best-performing SMR configuration as the performance and workload baseline, the first workload reduction

approaches for palm vein SMR are applied and evaluated in identification and verification scenarios as above.

CPCA Repetition of the above verification experiments, but with CPCA feature reduction.

Binary SMR The same experiments (for both original SMR and SMR-CPCA, verification and identification) as above are repeated with the binary representations of the SMR. It holds special interest whether the binary recognition performance benefits from other SMR settings than the (original) double-precision SMR.

Bloom filter The second workload-reduction experiments with the system described in chapter 6.

Identification and workload reduction - Binary SMR-CPCA With one template per subject (references), the Bloom filter trees are constructed. Genuine and impostor scores are obtained as described in the baseline implementation. All results achieved in the means of biometric performance are analysed with respect to the achieved workload reduction on top of the results achieved in the workload reduction by feature reduction section.

Identification and workload reduction - Binary SMR An additional experiment is run the same way as the above experiment to test whether the larger binary representation achieves a higher biometric performance under equal workload reduction achievements.

Verification Completing the Bloom filter experiments, the verification capabilities for both binary SMR-CPCA and binary SMR Bloom filter templates are tested.

CPCA-Tree The third workload-reduction experiments with the system described in chapter 7. These experiments follow the same procedure as the experiments for the Bloom filter, without the verification experiments since the templates are the same as in the workload reduction by feature reduction binary SMR-CPCA section.

Full vascular pattern SMR Finally, additional experiments are run to test whether the SMR can be enhanced with the full vascular pattern to increase the biometric performance.

8.4.2 Experiment vocabulary

Alongside plots and tables, many labels employing shortcuts are used. The following table 8.3 lists all labels and outlines their meaning.

Label	Reference	Description
MHD-Baseline	9.1	Biometric performance baseline yielded by the MHD identification and verification experiments
PSML-Baseline	9.2.4	Biometric performance baseline yielded by the naïve SMR identification and verification experiments for the PolyU dataset
PSML-CPCA	9.3.2	Biometric performance (or the experiment) yielded by applying the naïve CPCA transformation to the PSML-Baseline
Binary PSML	9.3.2	Biometric performance (or the experiment) yielded by binarisation of the naïve PSML-Baseline
Binary PSML-CPCA	9.3.2	Biometric performance (or the experiment) yielded by binarisation of the PSML reduced with the CPCA feature-reduction (PSML-CPCA)
SM* Tuned	9.2.1, 9.2.2, 9.2.3	SMR types tuned in the means of λ_{max}
Bf	9.4.2	Bloom filter-indexing experiment with binary PSML templates
Bf ABC	9.4.2	Bloom filter-indexing experiment with binary PSML templates and an additional binary PSML comparison
Bf CPCA	9.4.1	Bloom filter-indexing experiment with binary PSML-CPCA templates
Bf CPCA ABC	9.4.1	Bloom filter-indexing experiment with binary PSML-CPCA templates and an additional binary PSML-CPCA comparison (ABC)
Bf CPCA ARC	9.4.1	Bloom filter-indexing experiment with binary PSML-CPCA templates and an additional real-valued PSML-CPCA comparison (ARC)
PSML-CPCA-A	7.1.4, 9.5.1	SMR-CPCA-A (SCA) templates build from PSML-CPCA-A templates
PSML-CPCA-C	7.1.4, 9.5.1	SMR-CPCA-C (SCC) templates build from PSML-CPCA-C templates
PSML-CPCA-M	7.1.4, 9.5.1	SMR-CPCA-M (SCM) templates build from PSML-CPCA-M templates
C-T SCA	9.5.1	CPCA-Tree indexing experiment with PSML-CPCA-A templates
C-T SCC	9.5.1	CPCA-Tree indexing experiment with PSML-CPCA-C templates
C-T SCM	9.5.1	CPCA-Tree indexing experiment with PSML-CPCA-M templates
C-T SC* ARC	9.5.5	CPCA-Tree indexing experiment with PSML-CPCA-A/C/M templates and an additional real-valued PSML-CPCA comparison
C-T SC* Masking	9.5.4	CPCA-Tree indexing experiment with PSML-CPCA-A/C/M templates and masking out most common set bits
ω_{256}	8.5	Workload ω for $\mathcal{S} = 256$
F_B	8.5	Workload fraction F compared to the (naïve) PSML-Baseline
F_R	8.5	Workload fraction F compared to the (naïve) binary PSML-CPCA

Table 8.3: Labels used in plots and tables.

8.5 Workload metric

Aside from the performance reporting metrics specified by the ISO/IEC 19795-1 [iso06], the systems workload and system workload reduction metrics introduced by [DRB17] are used to compare the systems workload for the different approaches and expansion stages.
The authors of [DRB17] present six requirements for a full-featured workload and workload-reduction reporting:

R1 The baseline workload must be explicitly stated.

R2 The baseline biometric performance of a state-of-the-art algorithm on the dataset used must be explicitly stated in a manner described in the ISO/IEC standard.

R3 The workload of the proposed scheme is to be stated in the manner described in R1.

R4 The biometric performance of the proposed scheme must be reported according to the ISO/IEC standard.

R5 The additional costs and benefits of the proposed scheme should be listed.

R6 The total workload for both the baseline and the proposed system is to be computed using equation 8.1.

The workload itself is expressed as

$$\omega = \mathcal{S} * p * \tau \tag{8.1}$$

where $\mathcal{S}$ is the number of enrolled subjects, p the penetration rate (measure of the average number of pre-selected templates as a fraction of the total number of templates, as defined in [iso06]) and τ the cost of a single comparison (i.e. the number of compared bits). The measure ω is not feasible to describe the workload for the chosen baseline since the minutiae feature set yields a varying τ and the comparison cost of the modified hausdorf distance with varying reference points is difficult to describe comparable to bit comparisons. Therefore, the workload of a naïve SMR system is used as a baseline. The workload reduction is stated as a fraction (F) of the baseline workload as proposed by requirement 6.

8.6 Summary

This chapter has presented the details of the experimental setup for this project. In the first section, the datasets used and their properties are outlined. The following two sections described the feature detection pipeline and its errors. Next, the roadmap of the performed experiments was presented. The results of these experiments were presented and discussed in the following chapters. Finally, the workload metric used by [DRB17] was introduced.

The three chosen datasets do not feature optimal raw data. However, they were chosen due to the lack of better palm vein datasets. The PolyU dataset features the best characteristics, albeit it is not flawless (instances without visible veins). To test the ROI detection and receive a second test database, the CASIA dataset is used. Unfortunately, all contained palm vein images contain a high amount of skin texture (noise) and some subjects bent, misplaced or shifted their hands. This dataset is expected to yield the worst results. With the PUT dataset, the only purpose built palm vein dataset is introduced. While it features a high image quality, it is not possible to automatically align the ROI since there are no reference points of the hands visible, and therefore the translation invariance features of the SMR has to compensate for that.

Chapter 9

Results

The experimental results of this project are presented in this chapter.

9.1 Baseline

The baseline results are obtained using the MHD for the extracted palm vein minutiae. These results are necessary to evaluate the results of the subsequent experiments and were used to find the base settings of the feature extraction pipeline for each dataset. All experiments where run with inter-session data.

9.1.1 Score Distribution

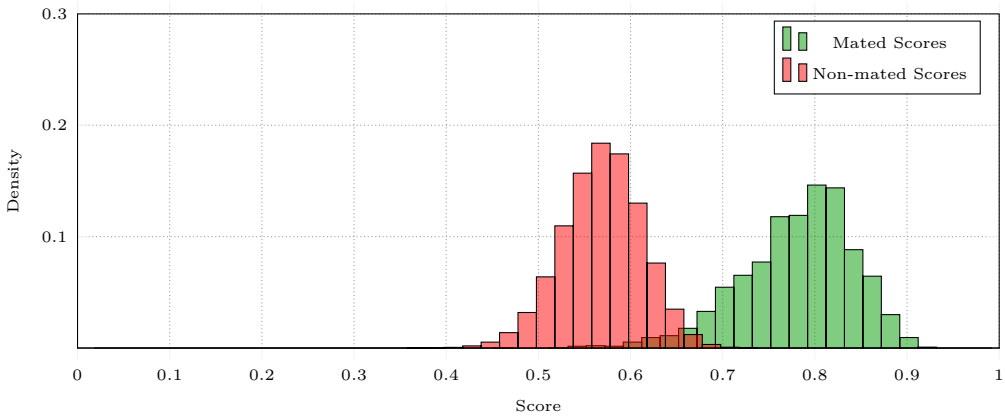
Looking upon the similarity score distribution of the datasets provides a first hindsight into the quality of the biometric data. Figure 9.1 presents the best-achieved MHD similarity score distributions for the three chosen datasets with feature extraction pipelines configured as stated in table 9.1. Observe h (NL-means) for all three datasets: the experiments have shown that

	max.-cur.	**NL-means**			**NL-diffusion**
Dataset	σ	h	$Size_{template}$ †	$Size_{search}$ †	k *
PolyU	8	7	7	23	1
CASIA	7	7	7	23	2
PUT	7	7	7	23	1

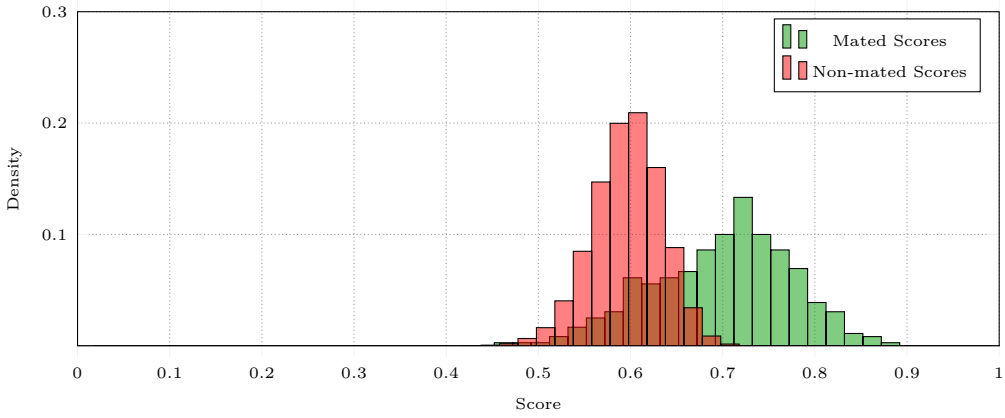
* Out of 2 non-linear-scale-space iterations with 2 sub-iterations.

† Same for all three datasets: ROI size is fixed to fit these sizes.

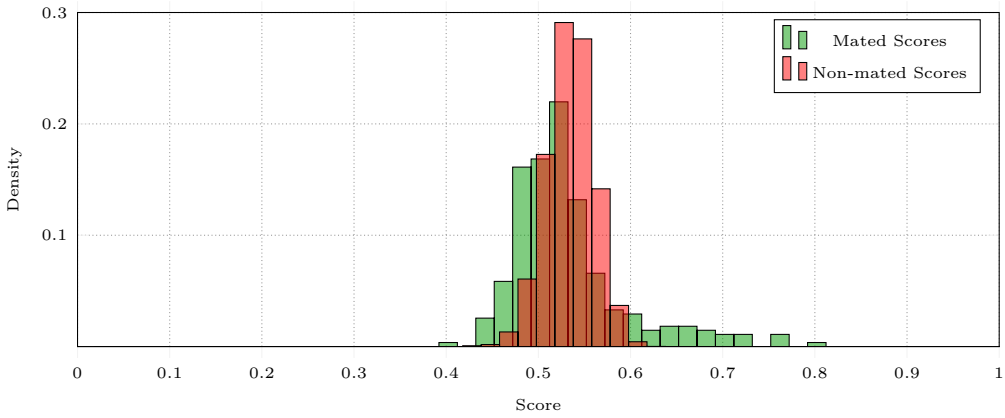
Table 9.1: Configuration of the minutiae feature extraction pipeline: σ - kernel size of the maximum curvature algorithm; h - decay parameter of the Euclidean distance in the NL-means; $Size$ - template window and search kernel size of the NL-means algorithm; k - amount of diffusion iterations in the NL-diffusion algorithm.



(a) PolyU



(b) CASIA



(c) PUT

Figure 9.1: MHD similarity score distribution (normalised to $x = 1.0$) for the three datasets.

$h > 7$ does not increase the recognition performance in any means[1]. In some cases, a $h > 7$ reduces the recognition performance because contrast resulting from noise (i.e. skin-texture) is also increased. Further observations showed that the maximum-curvature algorithm does not necessarily needs very high contrast to properly extract the veins, and thus reducing h to 7 does not impair the maximum-curvature performance.

[1]This contradicts the observation stated in section 4.3.1.

From examining figure 9.1, the following two observations can be stated:

- The score distribution for the PUT dataset is very bad. The nearly full overlap of impostor and genuine comparison trials yield no clear threshold, thus it is impossible to distinguish between impostor and genuine probes. This is expected for the PUT dataset since the dataset features many characteristics (rotations, translations, spurious minutia) that the MHD is not capable of compensating.
- The PolyU dataset impostor and genuine score overlap is very small. A threshold of around 0.65 is capable of distinguishing between impostor and genuine comparison trials with a very small error compared to the PUT dataset.
- As expected in section 8.3.1, the bad quality data contained in the CASIA dataset shifts the genuine scores to the left[2].

Basing on these observations and the three score distributions, it is expected that the PolyU dataset will yield the best biometric performance, followed by the CASIA dataset with a distant performance for the PUT dataset.

9.1.2 EER and ROC

The EER is a commonly-used metric for reporting the performance of a biometric verification system. It is defined by the [iso06] as the point at which the FMR and FNMR are equal. The

Dataset	EER
PolyU	5.4%
CASIA	30.5%
PUT	47.3%

Table 9.2: EER for the MHD comparison of the PolyU, CASIA and PUT datasets.

conclusions derived from figure 9.1 in the previous section can be validated using the EER presented by table 9.2. With an EER of 5.4%, the MHD achieves an moderate verification performance for the PolyU dataset. As expected, the performance of the CASIA dataset is below the PolyU dataset. However, the EER of 30.5% for the CASIA dataset confirms the MHD is not able to achieve distinguishable scores of impostor and genuine probes with the bad quality data of the dataset. Also confirming the expectations for the PUT dataset with an EER of 47.3%, the MHD fails to yield any distinguishable scores for heavily translated or rotated data.

[2]It can be expected that the CASIA dataset would perform similar to the PolyU dataset if the poor quality data were excluded.

The relevant metrics for biometric identification systems are the True Positive Identification Rate (TPIR) and FPIR, both defined by the [iso06] standard. By moving the decision threshold for the biometric identification system, these metrics can be plotted as a Receiver Operating Characteristic Curve (ROC) curve for fixed database sizes. Besides the ROC curve, three additional biometric performance points are used: $TP_{0.5}$, $TP_{0.1}$ and TP_0, which correspond to the TPIR at 0.5%, 0.1% and 0% FPIR, respectively. These will be used where single-value biometric performance indicators are feasible.

The ROC curves for the MHD comparison are plotted in figure 9.2 with $TP_{0.5}$, $TP_{0.1}$ and

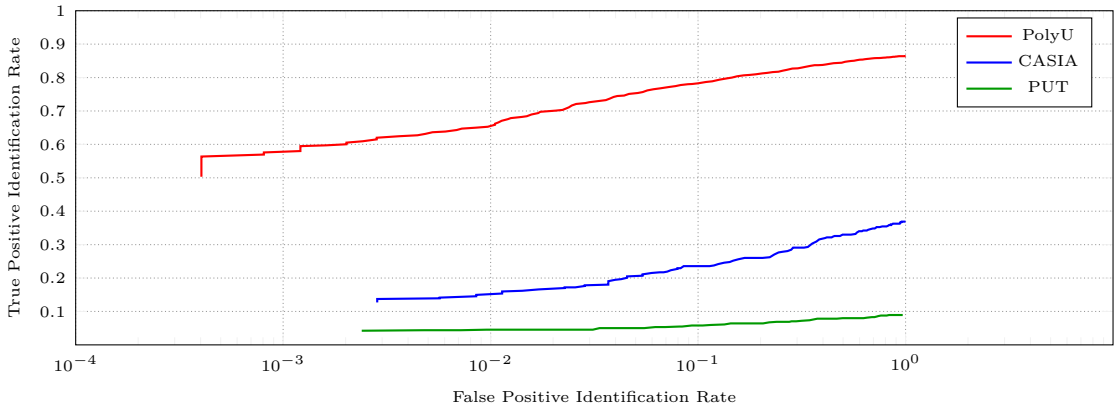


Figure 9.2: ROC curves for the MHD baseline system

Dataset	$TP_{0.5}$	$TP_{0.1}$	TP_0
PolyU	63.6%	59.7%	49.8%
CASIA	14.1%	-*	12.7%
PUT	4.4%	-*	4.1%

* Database to small to determine $TP_{0.1}$.

Table 9.3: $TP_{0.5}$, $TP_{0.1}$ and TP_0 for the MHD comparison of the PolyU, CASIA and PUT datasets.

TP_0 listed in table 9.3. As expected facing the results of the verification experiment, the MHD achieves the best biometric performance on the PolyU dataset, while CASIA and PUT are lagging far behind with an unacceptable TP_0. However, with a $TP_{0.1}$ of 59.7% and a TP_0 of 49.8%, the MHD yield results far from an acceptable performance.

9.2 Spectral Minutiae Representation

The initial SMR approach in [XVK+08] targeted fingerprints and their minutiae for biometric identification and verification. Image capturing of fingerprint samples is a much cleaner process then the image capturing of the vascular pattern. In fingerprint recognition the observed biometric characteristics are on the sufrace of the skin, and thus are easily collectable in various

ways. However, recall section 2.1, the palm veins are internal to the individual's body and only an blurred approximation of the vein pattern is represented in the captured vein images. Thus, even with a very stable feature extraction pipeline, more noise than in fingerprint feature extraction case must expected.
All three datasets show different properties. Therefore, it is necessary to run exhaustive experiments for all datasets since assumptions and results for one dataset are hardly applicable for the other datasets.

9.2.1 PolyU dataset

The PolyU dataset achieved the best biometric performance on the MHD baseline experiments. It features the best characteristics among all three datasets: homogeneous illumination, no translation, consistent noise levels and most poor-quality images have been removed. The input minutiae for the experiments are generated using the same feature extraction pipeline and settings as presented in table 9.1.

In section 5.1, three SMR types are presented. Further, every type can be compared absolute-valued or real-valued and additionally supplied with quality informations about single minutiae. Since the SMO does not yield a better performance than the SML (as stated by the authors in [XVB+09]), only SMC and SML will be used in the experiments. As shown in figure 9.3 for both

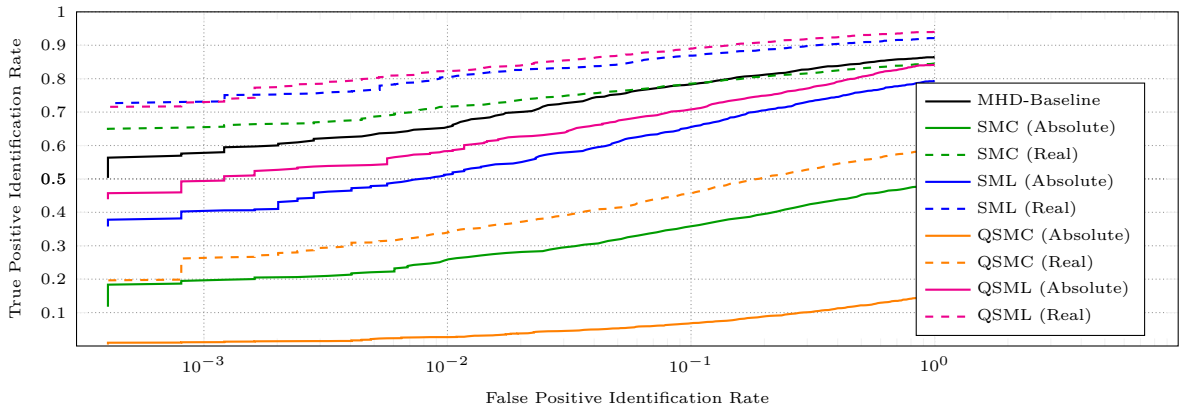


Figure 9.3: ROC curves of the SML, SMC, QSML and QSMC type for the PolyU dataset.

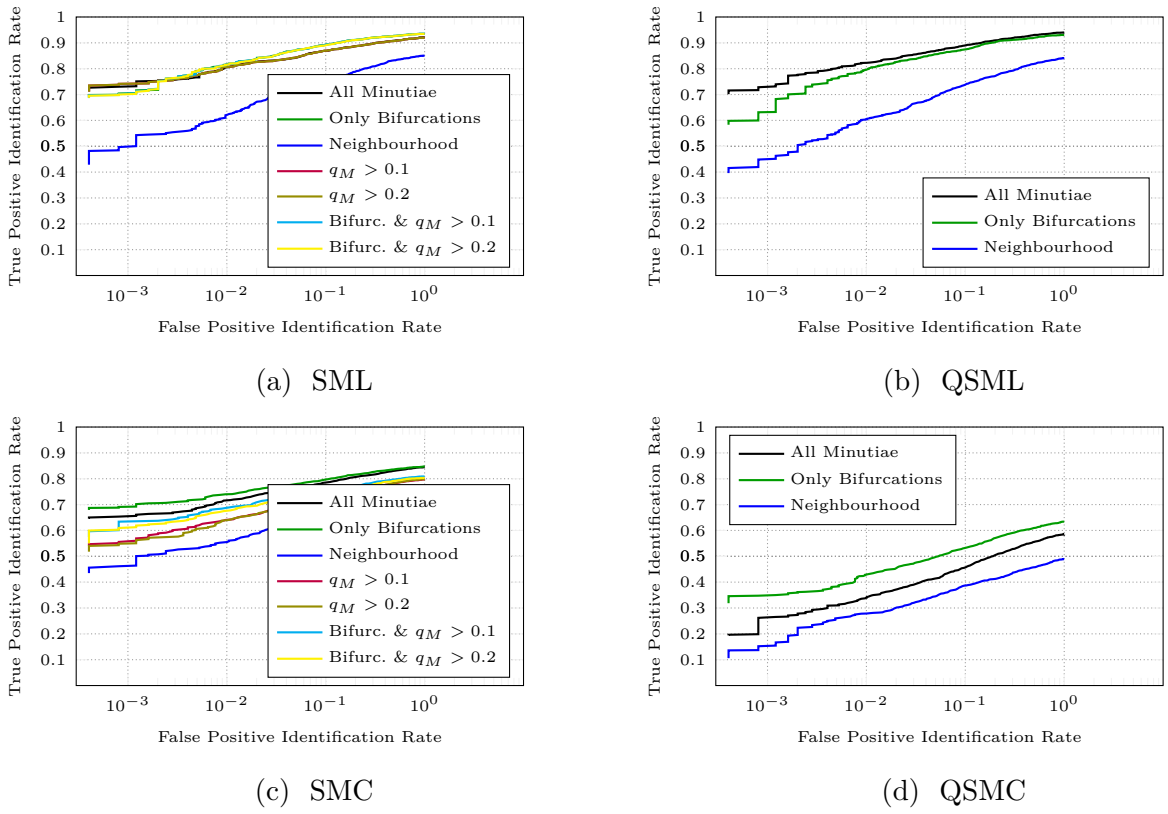
types the real-valued comparison performs much better than the absolute-valued comparison. The best performance for basic SMR identification approach is achieved by the real-valued SML with $TP_{0.1} = 75.1\%$ and $TP_0 = 72.4\%$ followed by the real-valued QSML with $TP_{0.1} = 77.3\%$ and $TP_0 = 70.1\%$. All six experiments were run with the settings recommended in [XVB+09]; $\sigma = 0.32$, $\lambda_{min} = 0.1$ and $\lambda_{max} = 0.6$. Note: the quality information extracted by the maximum-curvature algorithm for the quality data enhanced SMR (QSMR) is multiplied by 4 to achieve the plotted results. Refer to appended figure A.2 for a comprehensive plot of gain 2 to 8 for the QSML representation. Further, no q_L data were used for the QSMR experiments: the q_L

data made little to no difference in the pre-experiments. In theory, the QSML should perform better than the SML. Refer to the discussion (chapter 10) why the QSML is not able to achieve a higher performance in these experiments.

At this point, the real-valued SML, QSML and SMC representation achieve higher biometric performances than the basic MHD approach.

As mentioned in section 5.8, it is possible to use a minutiae pre-selection to exclude potential spurious minutiae. Figure 9.4 shows the ROC curves for the five minutiae selection approaches applied for the real-valued SML, SMC, QSML and QSMC[3].

Only the SMC and QSMC profit from a minutiae pre-selection, albeit insufficiently to reach



(a) SML (b) QSML

(c) SMC (d) QSMC

Figure 9.4: ROC curves of the real-valued SML, SMC, QSML and QSMC type for the PolyU dataset with applied minutiae pre-selection for determining a PSMR configuration.

the performance of the SML.

After determining the most promising SMR types, the final step is to tune the SMR sampling to fit the quality of the data (noise level) as mentioned in [XVB+09]. As for the SML, the tuning has not achieved a better biometric performance in terms of $TP_{0.1}$ and TP_0. However, in terms of the overall performance the tuned sampling does not achieve much better results.

The top three configurations in terms of identification biometric performance for the PolyU

[3]Since the QSML and QSMC already include quality information (see section 5.7), the reliability selection is skipped.

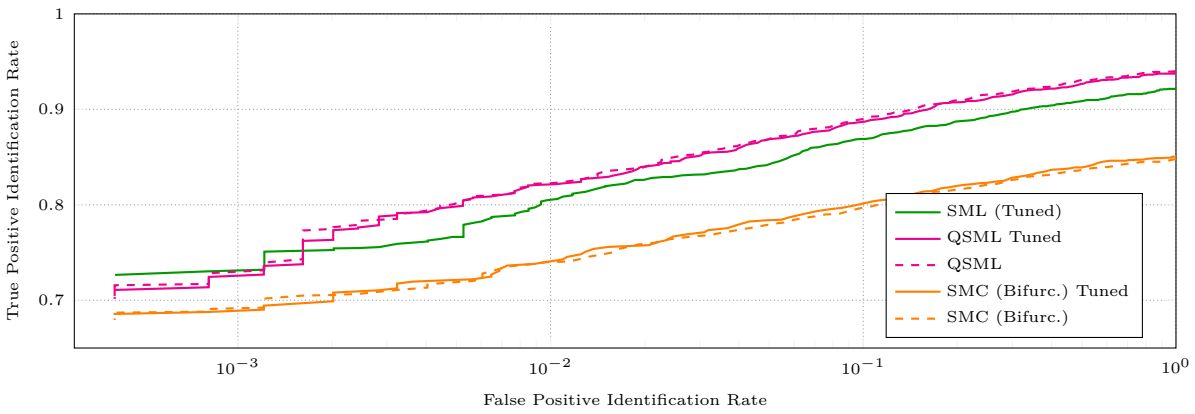


Figure 9.5: ROC curves of the most promising SMR types, tuned and untuned, for the PolyU dataset. Note: the tuned SML and the untuned SML are the same ROC since the SML achieves the best biometric performance at $\lambda_{max} = 0.6$.

dataset are summarized in table 9.4. For each of the top three configurations, a verification experiment has been run and the EER is also stated.

SMR				Ident.		Verif.	
Type	**Pre-Selection**	λ_{min}	λ_{max}	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
SML	-	0.1	0.6	78.1%	75.1%	72.4%	3.4%
QSML	-	0.1	0.64	80.8%	76.2%	70.5%	3.3%
PSMC	Bifurcations	0.1	0.57	72.2%	69.4%	68.0%	7.1%

Table 9.4: Top three configurations ($TP_{0.1}$), for a SMR biometric identification system using the PolyU dataset, with their corresponding verification EER, ordered by TP_0.

9.2.2 CASIA dataset

The CASIA achieved the second best, but worse, biometric performance on the MHD baseline experiments. It features the worst characteristics among all three datasets: inhomogeneous illumination, inconsistent noise levels, misplaced or tilted hands and no poor-quality image, except those with a falsely-detected ROI, have been removed. Again, the input minutiae for the experiments are generated using the same feature extraction pipeline and settings as presented in table 9.1.

Repeating the same experiments as for the PolyU dataset above, the initial SML, SMC, QSML and QSMC plots are presented in figure 9.6. The real-valued comparison again outperforms their absolute-valued counterpart. As expected from the MHD results, the results using SMR are not acceptable either. Unfortunately, the original SMR approaches even perform worse than the MHD approach.

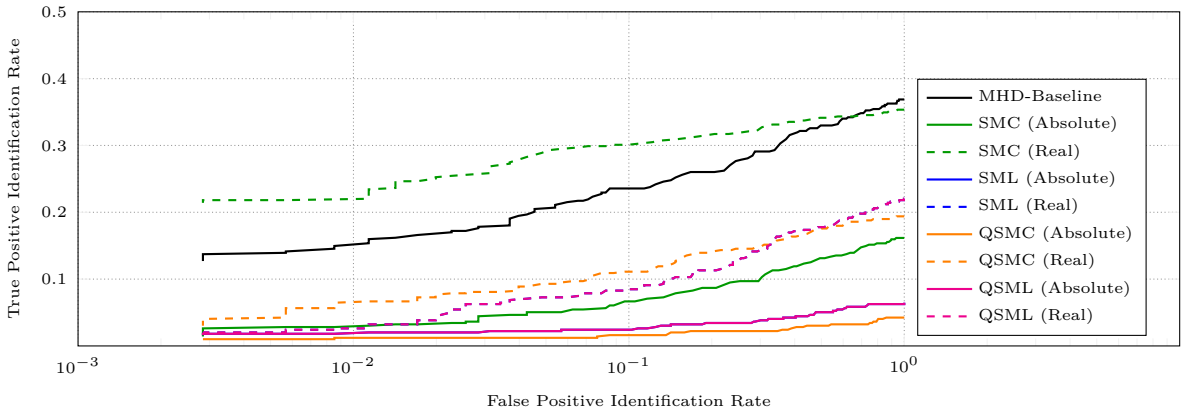


Figure 9.6: ROC curves of the SML, SMC, QSML and QSMC type for the CASIA dataset.

As already applied to the PolyU dataset, the CASIA dataset might profit from applying various minutiae pre-selection approaches. Since the real-valued SMR again yield a better basis, the very same minutiae pre-selection approaches as for the PolyU dataset are applied for the PolyU dataset on the real-valued SMR in figure 9.7. Unfortunately, even the various minutiae pre-selection approaches are unable to compensate the bad quality of the CASIA dataset. Only the performance of the SMC is improved by removing all endpoint minutiae.

The last outstanding experiment for the CASIA dataset is to tune the λ values for the different, most promising SMR. Since only the SMC and the only-bifurcation-minutiae-PSMC yield results that can be concerned, the plot in figure 9.8 only features the tuning of these types. Finally, table 9.5 summarizes the best configurations as for the CASIA dataset. The performance

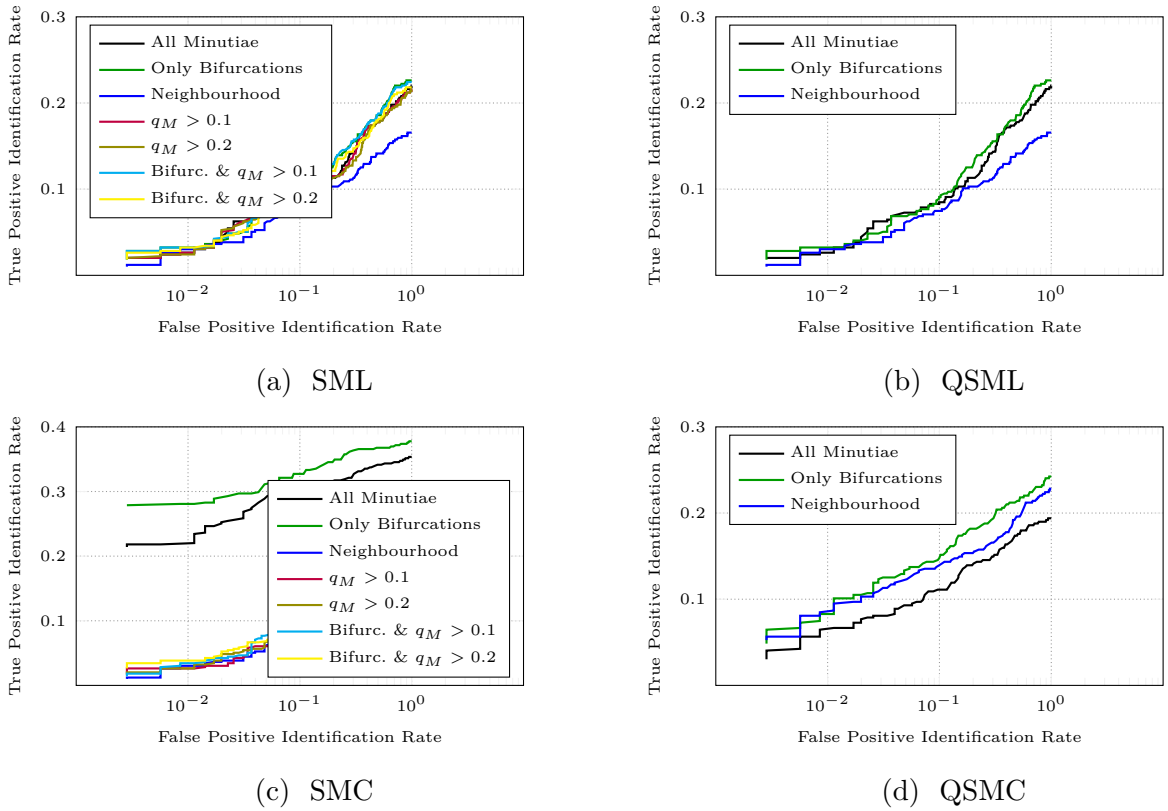
	SMR			**Ident.**		**Verif.**
Type	**Pre-Selection**	λ_{min}	λ_{max}	$TP_{0.5}$	TP_0	**EER**
PSMC	Bifurcations	0.1	0.60	27.9%	27.6%	16.4%
SMC	-	0.1	0.45	24.6%	22.8%	17.5%

Table 9.5: Top two configurations (TP_0), for a SMR biometric identification system using the CASIA dataset, with their corresponding verification EER.

on the CASIA dataset is astonishingly poor. Compared to the MHD approach, the SMR gains around 10% points in the means of TP_0 and $TP_{0.5}$. These poor performances (MHD and SMR) hint that there might be a problem with the feature extraction pipeline.

9.2.3 PUT dataset

Finally, the PUT achieved the worst biometric performance on the MHD baseline experiments. The results are dominated by two good and two bad properties: homogeneous illumination and



(a) SML (b) QSML

(c) SMC (d) QSMC

Figure 9.7: ROC curves of the real-valued SML, SMC, QSML and QSMC type for the CASIA dataset with applied minutiae pre-selection for determining a PSMR configuration.

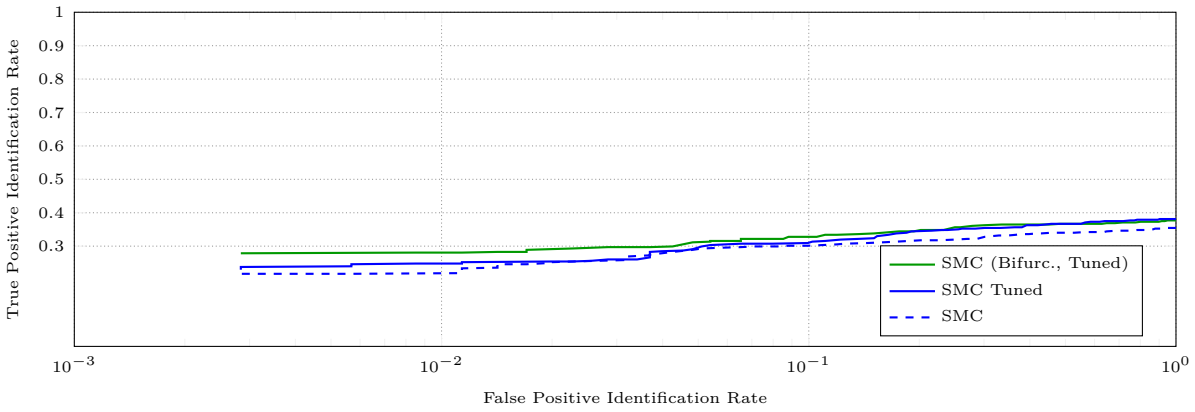


Figure 9.8: ROC curves of the most promising SMR types, tuned and untuned, for the CASIA dataset. Note: the tuned PSMC (SMC with only bifurcation minutiae pre-selection) and the untuned PSMC are the same ROC since the PSMC achieves the best biometric performance at $\lambda_{max} = 0.6$.

very good contrast but strong noise[4] and it is impossible to automatically compensate the ROI translation. Through the latter, it is expected that the absolute-valued SMR performs better than the real-valued ones. Again, the input minutiae for the experiments are generated using the same feature extraction pipeline and settings as presented in table 9.1.

[4]Recall (section 8.1), there are hints of some additional, manual image enhancements that also boost the noise.

Once again repeating the same experiments as for the PolyU and CASIA dataset above, the initial SML, SMC, QSML[5] and QSMC plots are presented in figure 9.9. As expected, the

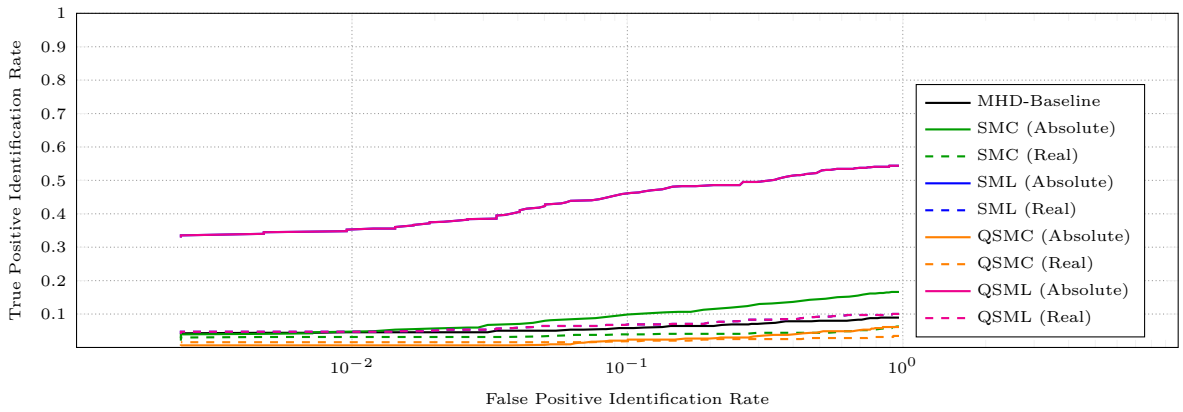


Figure 9.9: ROC curves of the SML, SMC, QSML and QSMC type for the PUT dataset.

absolute-valued SMR yield better biometric performance than their real-valued counterparts. Again, the SMR performs better than the MHD, but the SMR still does not yield a acceptable performance. The performance gains achieved by applying the minutiae pre-selection approaches to the PUT dataset, shown in figure 9.10, still do not yield desirable results. A deviation from the other datasets is the, in perspective, enormous performance gain of the PSMC (with applied nighbourhood cleaning; figure 9.10c) compared to the SMC.

To complete the experiments for the PUT dataset, the last experiment is run to tune λ for the SMR sampling as for the other datasets. Figure 9.11 shows the ROC of the tuned and untuned SMR for the best five configurations yielded by the previous experiment (figure 9.10).

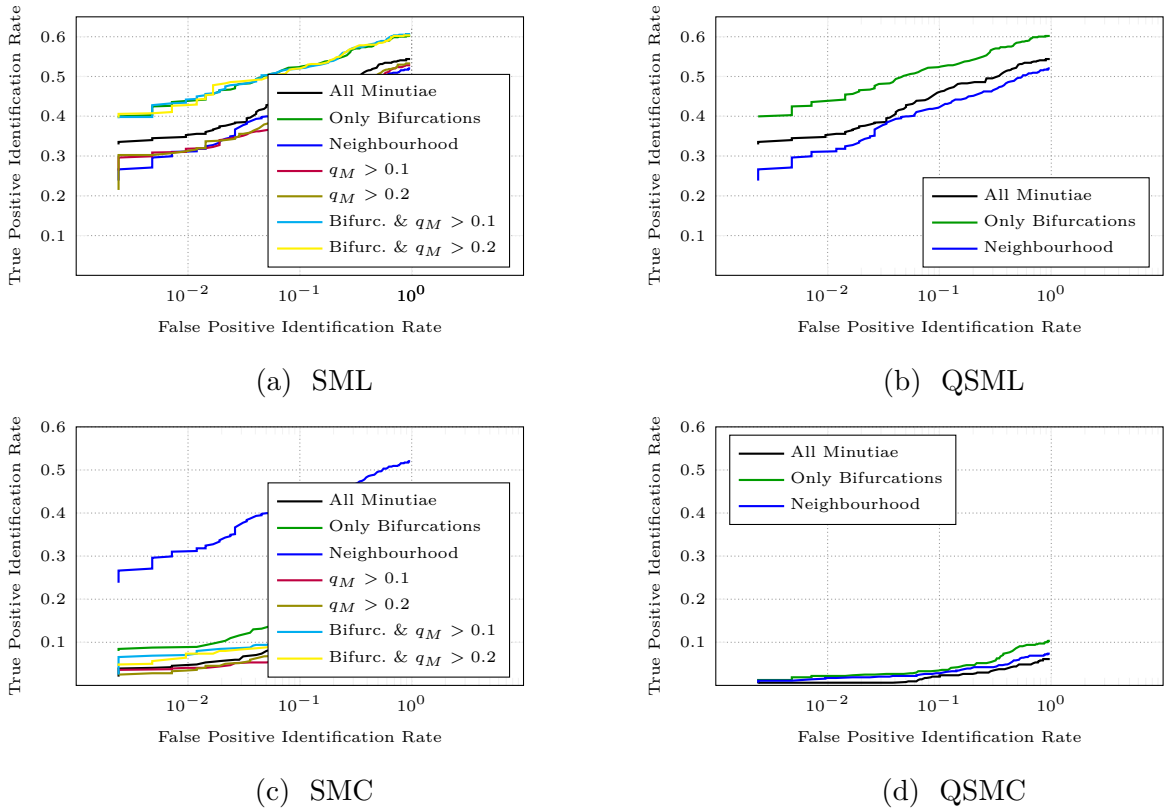
Finally, the top three configurations for the PUT dataset are summarized in table 9.6. Com-

	SMR			**Ident.**		**Verif.**
Type	**Pre-Selection**	λ_{min}	λ_{max}	$TP_{0.5}$	TP_0	**EER**
PSML	Bifurc. & $q_M > 0.2$	0.1	0.68	41.8%	41.1%	8.3%
PSML	Bifurc. & $q_M > 0.1$	0.1	0.47	41.4%	40.8%	8.5%
SML	-	0.1	0.68	43.3%	40.8%	8.5%

Table 9.6: Top three configurations, for a SMR biometric identification system using the PUT dataset, with their corresponding verification EER, ordered by TP_0.

pared to the MHD, the SMR performs much better, although the results are still unacceptable.

[5]Note: since the PUT features a much higher contrast than the other two datasets, the multiplier (gain) for the quality information extracted with the maximum curvature algorithm was redetermined using the same procedure as the initial gain determination.



(a) SML (b) QSML

(c) SMC (d) QSMC

Figure 9.10: ROC curves of the absolute-valued SML, SMC, QSML and QSMC type for the PUT dataset with applied minutiae pre-selection for determining a PSMR configuration.

Figure 9.11: ROC curves of the most promising SMR types, tuned and untuned, for the PUT dataset.

Again, these poor performances (especially MHD) hint that there might be a problem with the feature extraction pipeline.

9.2.4 Aging / Further experiments

All presented datasets suffer from aging. Figure 9.12 exemplary shows the ROC for the PolyU dataset analysing intra-session comparison trials (both probe and reference stem from the same session) versus inter-session comparison trials (probe and reference are not from the same

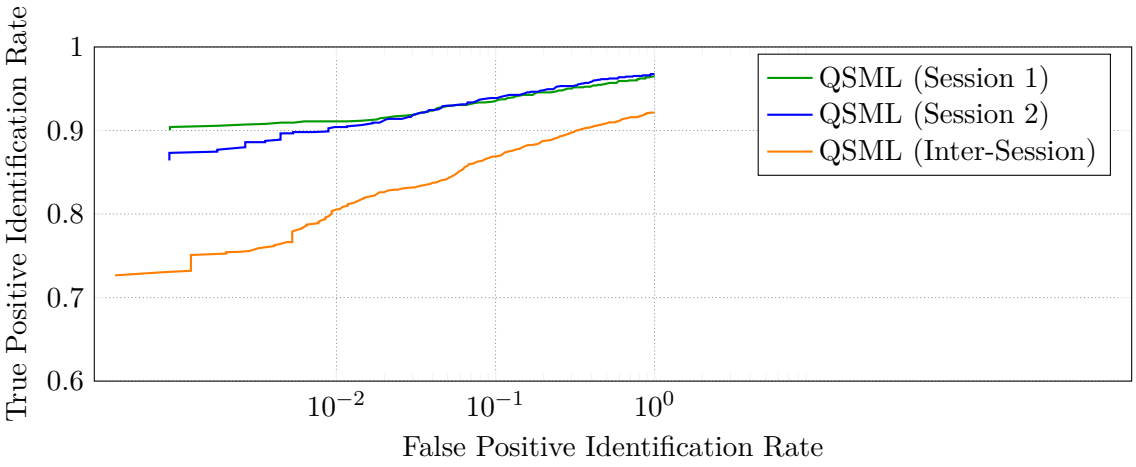


Figure 9.12: ROC for images of different sessions from the PolyU dataset.

session). Refer to the discussion in chapter 10 for an explanation for these results.
In order to continue the main experiments for this project on a reasonable basis, all following experiments will only be run with the first session of the PolyU dataset. However, all final experiments will be repeated twice for the PolyU dataset: once only using the second session, the second with an inter-session analysis.
Refer to table A.1 for the top three configurations for the first session. The best result[6] is yielded by the PSML with a minutiae pre-selection of $q_M > 0.2$ ($TP_0 = 90.4\%$).

9.3 Workload reduction by feature reduction

With the best-performing SMR configuration, PSML with minutiae pre-selection by $q_M > 0.2$ (further just called PSML-Baseline), determined, the first workload reduction experiments are executed. This section begins by determining the workload baseline, followed by the PSML-CPCA, binary PSML and the binary PSML-CPCA.

9.3.1 Workload baseline for the PSML-pipeline

Recall (section 8.5), to state the workload reduction, a baseline workload is needed. The workload is defined as $\omega = \mathcal{S} * p * \tau$.
For this experimental setup, 256 subjects are enrolled ($\mathcal{S} = 256$)[7]. The naïve biometric system compares all enrolled templates (penetration rate (p) = 1.0). Recall (section 5.2), one PSML (and SMR in general) template comprises $N \times M = 32768$ floating point numbers. There is no deterministic way to calculate the cost (e.g. CPU cycle count) of one floating point operation on a modern x86 (x86_64) since it depends too much on e.g. accessing data in cache, branch

[6] In terms of TP_0 and $TP_{0.1}$, the highest $TP_{0.5} = 92.0\%$ is achieved by the QSML.

[7] $\mathcal{S}$ is misleading in this context: since every user contributes two hands, 256 templates are enrolled from 128 subjects. In the PolyU it is not possible to link two biometric instances to one subject. Therefore it is assumed, that every subject presented one instance.

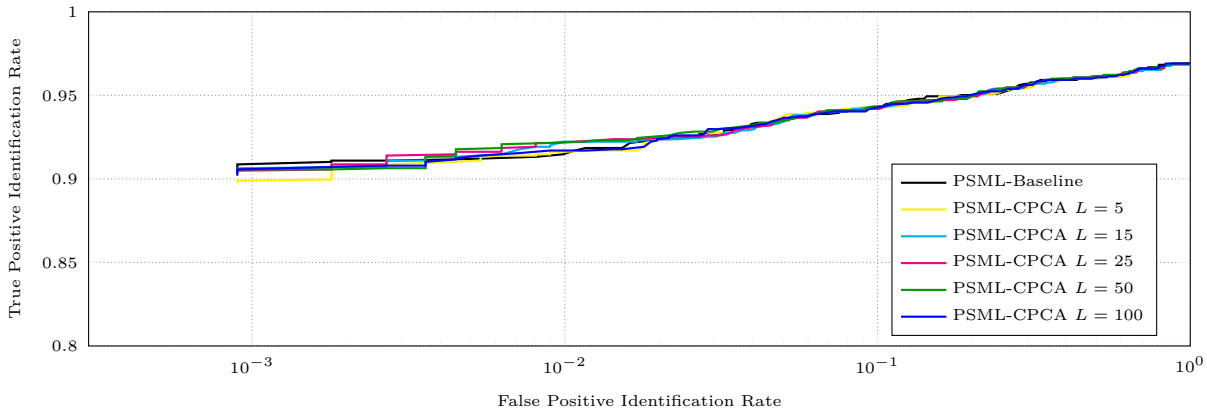


Figure 9.13: ROC for the CPCA feature reduction on the PolyU dataset with a training set of $L =$ 5, 15, 25, 50 and 100 PSML.

prediction and the instruction pipeline. Further, recall section 5.6.1: one SML comparison comprises the sum of $N \times M$ floating point multiplications with one division, thus two operations are actually executed per floating point: one multiplication and one addition.
Therefore, it is assumed that these two floating point operations equal 32 bit-operations[8]. This assumption yields $\tau = 1048576$.
The final baseline workload is therefore $\omega = 256 * 1 * 1048576 \approx 2.68 \times 10^8$.

9.3.2 CPCA

The first workload reduction approach uses the CPCA feature reduction. Applying the CPCA to the PSML-Baseline, yields little-to-no performance impairment, as shown in figure 9.13. No performance improvements could be achieved by using different SMR settings or CPCA training. Refer to table A.2 for a full listing of TP_0, $TP_{0.1}$, $TP_{0.5}$ and EER for different L, and to table 9.7. The CPCA applied to the PSML-Baseline is further called PSML-CPCA. Note: the

	SMR		**Ident.**			**Verif.**
Label	λ_{min}	λ_{max}	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
PSML-Baseline	0.1	0.51	91.2%	91.1%	90.4%	2.1%
PSML-CPCA $L = 100$	0.1	0.51	91.5%	90.6%	90.2%	2.1%

Table 9.7: TP_0, $TP_{0.1}$, $TP_{0.5}$ and EER for the CPCA feature reduction on the PSML-Baseline.

experiment was run with $L = 5, 15, 25, 50, 100$ PSML for the CPCA. There are no mentionable differences between the training set sizes[9]. Therefore, even small databases with a very small training set can apply the CPCA reduction for their SMR-based system and are able to train

[8] This follows the recommendation in [DRB17] to count the number of compared bits.

[9] In fact, the training set with $L = 15$ performs better than all others in the means of TP_0. The introduced error depends more on the quality (e.g. the significance) of the templates used.

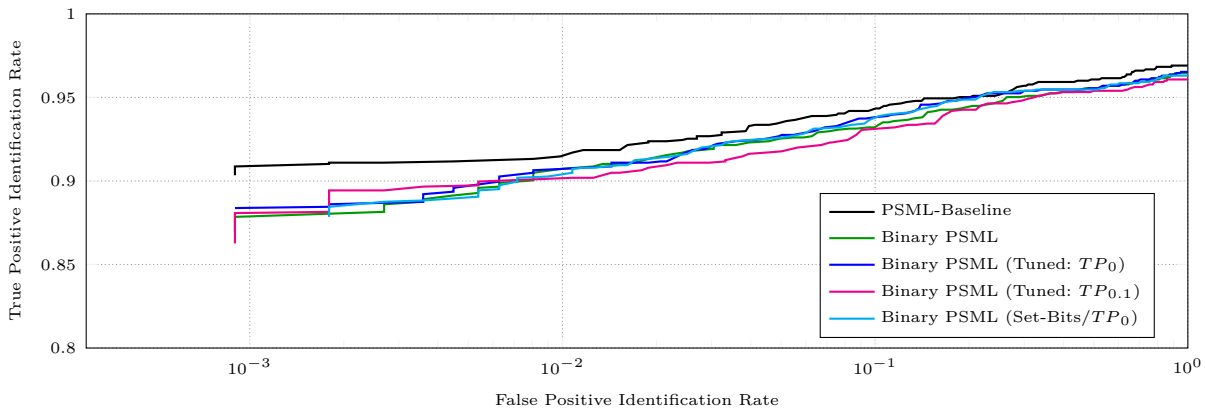


Figure 9.14: ROC for the binary PSML feature vector of the PSML-Baseline on the PolyU dataset.

a completely different CPCA, if necessary.

A additional minor SMR-CPCA configuration is the λ_{max}^{CPCA}, which states λ_{max} for the SMR that are used as CPCA training templates. Using a larger λ_{max}^{CPCA} then λ_{max} results in a larger SMR-CPCA projection necessary to keep the biometric performance at the level of the full-featured SMR. When using $\lambda_{max}^{CPCA} = \lambda_{max}$ with small λ_{max} the SMR-CPCA templates might be reduced too much, which (drastically) lowers the biometric performance. If not stated otherwise, all following SMR-CPCA-based experiments are run with $\lambda_{max}^{CPCA} = 0.6$.

After the CPCA feature reduction with $\lambda_{max}^{CPCA} = 0.6$, it is safe to crop the PSML-CPCA to $M_{CPCA} = 24$ rows, yielding $N \times M_{CPCA} = 6144$, thus $\omega = 256 * 1 * (6144 * 32) \approx 5.03 \times 10^7$. This alone results in a workload reduction to $F \approx \frac{5.03 \times 10^7}{2.68 \times 10^8} \approx 0.1875 = 18.75\%$.

9.3.3 Binary SMR

A further workload reduction approach is the binarisation of the floating point feature vector SMR. Reducing the feature vector from a $32bit$ to a $1bit$ feature vector results in a workload of $\omega = 256 * 1 * 32768 \approx 8.39 \times 10^6$, therefore achieving $F = 0.03125 = 3.125\%$. The binarisation experiment was run in two stages: first, the performance has been evaluated using the PSML-Baseline settings with the recommended MT for the binarisation; and second, an exhaustive experiment has been run to determine the best performing settings for the binary PSML in the means of λ_{max} and mask thresholds. In the second experiment, different mask thresholds for the enrolled templates and the query templates have been used. As seen in figure 9.14 and stated in table 9.8, the binarisation step does not drastically impair the performance of the original PSML-Baseline.

Label	*MT*		SMR		Ident.			Verif.
	Enrol	**Query**	λ_{min}	λ_{max}	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
Binary PSML	0.8	0.8	0.1	0.51	89.6%	87.9%	86.7%	2.4%
Binary PSML (Tuned: TP_0)	0.7	0.6	0.1	0.47	89.6%	88.6%	88.2%	2.2%
Binary PSML (Tuned: $TP_{0.1}$)	0.7	0.2	0.1	0.55	90.0%	89.4%	85.0%	2.3%
Binary PSML (Set-Bits/TP_0)	0.6	0.6	0.1	0.51	89.4%	88.5%	87.7%	2.2%

Table 9.8: TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for a binary PSML biometric identification system using the PolyU dataset, with their corresponding verification EER, ordered by TP_0.

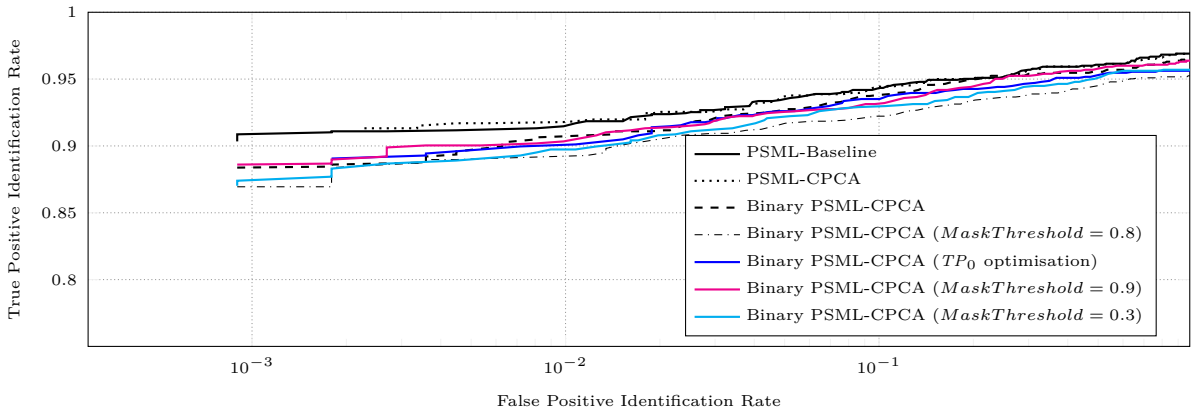


Figure 9.15: ROC for the binary PSML-CPCA feature vector on the PolyU dataset.

9.3.4 Binary SMR-CPCA

Combining the two feature reduction approaches above (CPCA and binarisation) results in an even greater workload reduction. Since M_{CPCA} can be reduced to 20 for the binary PSML-CPCA, the feature vector of size $N \times M_{CPCA} = 5120$ comprises $1bit$ elements, $\omega = 256{*}1{*}5120 \approx 1.31{\times}10^6$ is achieved, thus the workload reduction $\digamma \approx 0.004883 \approx 0.4883\%$. Under consideration of the performance-lossless feature compression achieved by the CPCA and the small amount of performance loss by the binarisation, the PSML-CPCA should achieve a comparable performance to the binary PSML.

The experiments confirm the previous assumption. Strictly speaking, the binary PSML-CPCA performs better than the binary PSML, as seen in figure 9.15 and stated in table 9.9. The experiments has been run as in the previous experiment: first, the performance has been evaluated using the best-performing PSML-CPCA settings with the recommended mask threshold for the binarisation ($MT = 0.8$); and second, an exhaustive experiment has been run to determine the best performing settings for the binary PSML-CPCA in the means of λ_{max} and mask thresholds. Additionally reported results feature a TP_0 optimised configuration, a low mask threshold configuration ($MT = 0.3$) and a high threshold, thus theoretically most stable on the edges, configuration ($MT = 0.9$).

Label	MT		SMR		Ident.			Verif.
	Enrol	**Query**	λ_{min}	λ_{max}	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
Binary PSML-CPCA TP_0	0.6	0.4	0.1	0.36	89.7%	89.1%	89.0%	2.2%
Binary PSML-CPCA $MT = 0.9$	0.9	0.9	0.1	0.45	90.0%	89.0%	88.4%	2.2%
Binary PSML-CPCA $MT = 0.3$	0.3	0.3	0.1	0.51	88.9%	88.3%	86.9%	2.3%
Binary PSML-CPCA $MT = 0.8$	0.8	0.8	0.1	0.51	90.0%	88.4%	86.7%	2.3%

Table 9.9: TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for a binary PSML-CPCA biometric identification system using the PolyU dataset, with their corresponding verification EER, ordered by TP_0.

9.3.5 Summary

With the feature reduction approaches, a significant workload reduction is already achieved. The smallest workload reduction $F \approx 0.4883\%$ is reached by the binary PSML-CPCA with a relative TP_0 recognition performance loss of only 1.4%-points compared to the PSML-Baseline.

Looking at the workload reduction achieved, the losses of the binary approaches are acceptable. Furthermore, both Bloom filter-indexing and CPCA-Tree-indexing require the binary representation. Therefore, the next experiments base on the results of this section. The next sections will explore the capabilities of two indexing approaches to reach a workload reduction by penetration rate reduction.

Approach	Workload		Ident.			Verif.	
	ω_{256}	F	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**	TP_0 **loss to PSML-Baseline**
PSML-Baseline	2.68×10^8	–	91.2%	91.1%	90.4%	2.1%	–
PSML-CPCA	5.03×10^7	18.748%	91.5%	90.6%	90.2%	2.1%	0.2%
Binary PSML	8.39×10^6	3.125%	89.6%	88.6%	88.2%	2.2%	2.2%
Binary PSML-CPCA	1.31×10^6	0.488%	89.7%	89.1%	89.0%	2.2%	1.4%

Table 9.10: Summary of achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for SMR biometric identification system using the PolyU dataset and their corresponding verification EER with applied feature reduction approaches, ordered by TP_0.

9.4 Bloom filter-indexing Approach

The Bloom filter-indexing approach applied in a iris recognition application achieved a workload reduction down to $F \approx 1\%$ in [DRB17]. However, the binary SMR shows different properties compared to the iris-code. This section presents the results achieved with the Bloom filter for the binary PSML-CPCA, starting with the main identification experiments, followed by a brief overview of the Bloom filter's verification capabilities.

9.4.1 Binary SMR-CPCA Bloom filter-indexing

The binary PSML-CPCA achieved better results than the binary PSML, in terms of both regarding biometric performance and workload reduction. Therefore, the binary PSML-CPCA is chosen as the first input for the Bloom filter-indexing.

First, the best-performing $\mathcal{H}$ and $\mathcal{W}$ need to be found. Since the PSML-CPCA is comparatively small, Bloom filter block sizes $4 \leq \mathcal{H} \leq 8$ respectively $4 \leq \mathcal{W} \leq 16$ are tested. For every $\mathcal{H}$ and $\mathcal{W}$ pair, the binarisation mask thresholds $0.1, 0.2, \ldots, 0.9$ are used to find the optimal mask-block size combination. To obtain the most clean results for finding $\mathcal{H}$ and $\mathcal{W}$ in identification scenarios, a Bloom filter tree configuration with no workload reduction is chosen; $\mathcal{T} = 64$, without quick traversal decision and without tree pre-selection.
The results of the first Bloom filter experiments are listed in table 9.11. Even with no workload

Bloom filter		**SMR**	**Workload**			**Recognition Performance**			
$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
4	5	0.1	$2.09 * 10^6$	0.7812%	160.0%	82.1%	80.9%	79.6%	10.8%
4	5	0.2	$2.09 * 10^6$	0.7812%	160.0%	82.1%	78.0%	76.6%	13.8%
5	4	0.5	$1.04 * 10^6$	0.3906%	80.0%	80.2%	78.0%	76.6%	13.8%
4	5	0.3	$2.09 * 10^6$	0.7812%	160.0%	79.1%	77.7%	76.6%	13.8%
4	4	0.4	$1.31 * 10^6$	0.4882%	100.0%	80.9%	76.8%	73.6%	16.8%
5	4	0.6	$1.04 * 10^6$	0.3906%	80.0%	77.1%	75.4%	73.5%	16.9%
6	4	0.7	$8.73 * 10^5$	0.3256%	66.7%	76.9%	75.9%	72.3%	18.1%
6	4	0.8	$8.73 * 10^5$	0.3256%	66.7%	74.5%	71.8%	68.5%	21.9%
7	4	0.9	$7.48 * 10^5$	0.2790%	57.1%	70.3%	69.2%	69.2%	21.2%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table 9.11: Best achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tested binarisation mask threshold with their corresponding $\mathcal{H}$ and $\mathcal{W}$ ($\mathcal{T} = 64$), ordered by $TP_0, TP_{0.1}$.

reduction by using lesser trees, the Bloom filter loses ∼10.8%-points of TP_0 in its best configuration for the binary PSML-CPCA. Furthermore, the best configuration reintroduces 60% workload compared to the naïve binary PSML-CPCA approach, due to the small Bloom filter block size. Using a larger Bloom filter block size drastically reduces the achieved performance.

The origin of the poor performance of the Bloom filter applied to binary PSML-CPCA templates is the high number of bit errors when comparing two mated binary PSML-CPCA, visualised in figure 9.16. Even at a very high mask threshold of 0.9, the bit-error rate is 12.4% with an error in more than 50% of columns, which is excessive for a reliable Bloom filter trans-

(a) Threshold 0.1 (b) Threshold 0.4 (c) Threshold 0.9

Figure 9.16: Bit-errors between two mated binary PSML-CPCA templates at three different binarisation mask thresholds (white = bit-error).

formation that needs stable column-bits. Several bit-error correction methods for more robust columns were tried to achieve a higher recognition performance for the PSML-CPCA Bloom filter: majority-voting (3 rows), majority-voting (5 rows), majority-voting (N4), majority-voting (N8) and a median filter. Every correction method was also tested with a transposed PSML-CPCA. Among all correction methods, the majority-voting over 3 rows achieved the highest performance, albeit much less than without bit-error correction. Appended table A.3 lists the results of the majority-voting, the transposed PSML-CPCA and the transposed PSML-CPCA with majority-voting.

While analysing the poor performance of the Bloom filter for the binary PSML-CPCA, it was noticeable that the pre-selection error rate (PER; rate of pre-selection error events) was small, but their genuine comparison scores overlapped too much with the scores of impostor queries. Therefore, another experiment was run to test whether an additional comparison between the original enrolled binary PSML-CPCA template of the retrieved candidate and the original query binary PSML-CPCA template could increase the biometric performance while sacrificing[10] a minimal amount of workload reduction.
As listed in table 9.12, employing an additional PSML-CPCA comparison increases the performance up to 15% points. In some configurations, the generated overhead of the additional comparison is compensated by the increased Bloom filter block size (used to achieve a lower PER) compared to the previous experiment. Unfortunately, in the best-performing configuration the workload is doubled compared to the naïve PSML-CPCA.

However, the Bloom filter in combination with the binary search tree approach is able to reduce the workload by reducing the number of trees built and using the tree pre-selection. The next experiment is a repetition of the previous experiment with halved and quartered tree count ($\mathcal{T} = 32, \mathcal{T} = 16$, no tree pre-selection, additional PSML-CPCA comparison). As listed in table 9.13, reducing $\mathcal{T}$ to $1/16$ of enrolled templates introduces a very high performance loss while hardly reducing the workload compared to $1/8$ of enrolled templates due to too many bit collisions and overly-populated tree nodes for a correct traversal direction decision. Note: one configuration $\mathcal{T} = 32$ ($\mathcal{W} = 5, \mathcal{H} = 4$, mask threshold = 0.7) yields a higher biometric performance than all $\mathcal{T} = 64$ results while achieving a much higher workload reduction. Comparing all

[10]Recall, the number of bit comparisons for a Bloom filter template equals its size: $2^{\mathcal{H}} * {}^{M_{CPCA}}/_{\mathcal{H}} * {}^{N}/_{\mathcal{W}}$; thus $\omega_{256} = 256 * 2^{\mathcal{H}} * {}^{M_{CPCA}}/_{\mathcal{H}} * {}^{N}/_{\mathcal{W}}$. With the additional binary PSML-CPCA comparison, the workload increases to $\omega_{256} = 256 * 2^{\mathcal{H}} * {}^{M_{CPCA}}/_{\mathcal{H}} * {}^{N}/_{\mathcal{W}} + M_{CPCA} * N$.

Bloom filter		SMR	Workload			Recognition Performance			
$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
5	6	0.4	$2.80 * 10^6$	1.044%	213.7%	87.3%	87.1%	86.2%	4.2%
6	5	0.9	$1.40 * 10^6$	0.523%	107.1%	86.9%	86.2%	85.8%	4.6%
6	6	0.6	$2.33 * 10^6$	0.870%	178.2%	87.3%	86.6%	85.6%	4.8%
4	7	0.2	$5.99 * 10^6$	2.234%	457.5%	86.6%	85.5%	85.2%	5.2%
6	6	0.3	$2.33 * 10^6$	0.870%	178.2%	87.1%	86.1%	84.9%	5.5%
5	8	0.5	$8.39 * 10^6$	3.127%	640.4%	83.9%	83.6%	83.0%	7.4%
7	7	0.7	$3.42 * 10^6$	1.277%	261.6%	83.7%	83.6%	83.2%	7.2%
5	6	0.8	$2.80 * 10^6$	1.044%	213.7%	85.8%	85.3%	83.7%	6.7%
4	7	0.1	$5.99 * 10^6$	2.234%	457.5%	85.9%	84.9%	83.7%	6.7%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table 9.12: Best achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML-CPCA comparison, for every tested binarisation mask threshold with their corresponding $\mathcal{H}$ and $\mathcal{W}$ ($\mathcal{T} = 64$), ordered by $TP_0, TP_{0.1}$.

Bloom filter			SMR	Workload			Recognition Performance			
$\mathcal{T}$	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
16	4	4	0.2	$6.60 * 10^5$	0.246%	50.4%	84.5%	83.8%	83.0%	7.4%
	4	6	0.4	$1.75 * 10^6$	0.653%	133.7%	83.0%	82.7%	82.5%	7.90%
	4	6	0.5	$1.75 * 10^6$	0.653%	133.7%	83.0%	82.5%	82.3%	8.1%
	...‡									
32	**5**	**4**	**0.7**	$\mathbf{7.91 * 10^5}$	**0.295%**	**60.4%**	**87.8%**	**87.6%**	**86.6%**	**3.8%**
	5	5	0.8	$1.26 * 10^6$	0.471%	96.4%	87.7%	87.3%	86.2%	4.2%
	8	5	0.4	$7.91 * 10^5$	0.295%	60.4%	87.0%	86.6%	86.1%	4.3%
	...‡									

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.4.

Table 9.13: Top three achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML-CPCA comparison using $\mathcal{T} = 32$ and $\mathcal{T} = 16$ trees.

other $\mathcal{T} = 32$ and $\mathcal{T} = 64$ configurations, this could be serendipity.

Since the biometric performance loss introduced by using $\mathcal{T}$ less than 1/8 of enrolled templates is too high, another $\mathcal{T} = 64$ and $\mathcal{T} = 32$ experiment has been run with enabled tree pre-selection. Tree pre-selection introduces additional workload by first comparing all tree root nodes with the query template, thus selecting more than 48 trees for $\mathcal{T} = 64$ and more than

26 trees for $\mathcal{T} = 32$ would actually increase the workload. Observe table 9.14, reducing the

Bloom filter			**SMR**	**Workload**			**Recognition Performance**			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	4	4	0.2	$3.53 * 10^5$	0.1316%	26.9%	78.2%	77.3%	77.1%	13.3%
2	4	4	0.3	$3.73 * 10^5$	0.1392%	28.5%	82.0%	81.4%	81.3%	9.1%
3	4	4	0.4	$3.94 * 10^5$	0.1469%	30.1%	84.7%	84.3%	83.4%	7.0%
...‡										
18	5	5	0.6	$8.96 * 10^5$	0.3339%	68.4%	89.2%	88.6%	87.7%	2.7%
19	**5**	**5**	**0.6**	$\mathbf{9.22 * 10^5}$	**0.3437%**	**70.4%**	**89.5%**	**88.9%**	**88.0%**	**2.4%**
20	5	5	0.6	$9.48 * 10^5$	0.3535%	72.4%	89.5%	88.9%	88.0%	2.4%
...‡										
46	5	6	0.4	$2.71 * 10^6$	1.0113%	207.1%	87.3%	87.1%	86.2%	4.2%
47	5	6	0.4	$2.75 * 10^6$	1.0278%	210.4%	87.3%	87.1%	86.2%	4.2%
48	5	6	0.4	$2.80 * 10^6$	1.0444%	213.7%	87.3%	87.1%	86.2%	4.2%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.5.

Table 9.14: Best achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

Bloom filter			**SMR**	**Workload**			**Recognition Performance**			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	4	5	0.1	$3.16 * 10^5$	0.1179%	24.1%	66.6%	65.9%	65.7%	24.7%
2	4	5	0.1	$3.65 * 10^5$	0.1362%	27.9%	74.2%	73.8%	73.2%	17.2%
3	4	5	0.3	$4.14 * 10^5$	0.1545%	31.6%	76.6%	76.2%	76.0%	14.4%
...‡										
16	4	4	0.5	$6.60 * 10^5$	0.2460%	50.4%	87.7%	86.7%	86.6%	3.8%
17	4	5	0.6	$1.10 * 10^6$	0.4108%	84.1%	88.1%	87.2%	86.7%	3.7%
18	4	5	0.6	$1.15 * 10^6$	0.4292%	87.9%	88.1%	87.2%	86.7%	3.7%
...‡										
26	5	4	0.7	$7.75 * 10^5$	0.2888%	59.1%	87.4%	87.2%	86.2%	4.2%
27	5	4	0.7	$7.99 * 10^5$	0.2979%	61.0%	87.5%	87.3%	86.3%	4.1%
28	5	4	0.7	$8.24 * 10^5$	0.3071%	62.9%	87.6%	87.3%	86.4%	4.0%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.6.

Table 9.15: Best achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 26$ trees at $\mathcal{T} = 32$.

number of pre-selected trees to 19 to 21 increased the performance while reducing the workload ∼30%. In its best performance, the Bloom filter with $\mathcal{T} = 32$ suffers a ∼1.3%-points performance loss compared to the $\mathcal{T} = 64$ setup. Because a smaller Bloom filter block size is needed to receive this performance for $\mathcal{T} = 32$ with tree pre-selection (table 9.15), the workload reduction is less[11] than the workload reduction with $\mathcal{T} = 64$ and tree pre-selection.

The binary PSML-CPCA Bloom filter achieved the best biometric performance of $TP_0 = 88\%$ with $\mathcal{T} = 64$, a tree pre-selection of $t \approx 5/16$ and an additional binary PSML-CPCA comparison. This is a TP_0 loss of 1%-point compared to the binary PSML-CPCA baseline while reaching a workload reduction of just 30%. One final experiment is run to test whether using the real-valued PSML-CPCA as the additional comparison is able to achieve a higher biometric performance compared to the binary PSML-CPCA baseline while receiving at least a small amount of workload reduction, even if the comparison cost for a real-valued PSML-CPCA is 32 times higher than the binary PSML-CPCA comparison. Since the three template protection requirements are proven in section 7.2, it is safe to keep these templates. As listed in table 9.16,

Bloom filter			**SMR**	**Workload**			**Recognition Performance**			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	4	4	0.2	$3.53 * 10^5$	0.1316%	26.95%	78.9%	78.4%	78.0%	12.4%
2	4	5	0.3	$5.94 * 10^5$	0.2216%	45.39%	83.6%	83.1%	82.7%	7.7%
3	4	4	0.4	$3.94 * 10^5$	0.1469%	30.08%	85.4%	84.7%	84.6%	5.8%
					...‡					
25	**6**	**4**	**0.5**	$\mathbf{5.64 * 10^5}$	**0.2104%**	**43.1%**	**89.8%**	**89.1%**	**89.0%**	**1.4%**
26	6	4	0.5	$5.78 * 10^5$	0.2155%	44.14%	89.9%	89.1%	89.0%	1.4%
27	6	4	0.5	$5.92 * 10^5$	0.2206%	45.18%	90.0%	89.1%	89.0%	1.4%
					...‡					
46	6	5	0.6	$1.35 * 10^6$	0.5065%	103.7%	90.2%	89.4%	88.8%	1.6%
47	4	5	0.6	$2.06 * 10^6$	0.7710%	157.9%	90.0%	89.4%	88.8%	1.6%
48	4	5	0.6	$2.10 * 10^6$	0.7832%	160.4%	90.0%	89.4%	88.8%	1.6%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.7.

Table 9.16: Best achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final real-valued PSML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

the additional real-valued PSML-CPCA comparison achieves a higher biometric performance, than the binary PSML-CPCA baseline.

[11] When sacrificing a another small amount of biometric performance, a workload reduction ∼50% is achievable with $\mathcal{T} = 32$.

This and the two previous experiments use the Bloom filter-indexing as a rank 1 pre-selection approach with one final real-valued PSML-CPCA, respective one final binary PSML-CPCA comparison. With a tree pre-selection of $t = 25$ trees at $\mathcal{T} = 64$ and with the comparison cost of the real-valued comparison calculated, an additional[12] workload reduction of ∼57% is reached at $TP_0 = 89\%$ - just ∼1.42%-points less than the heavy, naïve PSML approach, while achieving the same biometric performance as the naïve binary PSML-CPCA approach and outperforming all previous binary PSML-CPCA Bloom filter-indexing approaches in terms of both biometric performance and workload.

9.4.2 Binary SMR Bloom filter-indexing

Bloom filter-indexing with binary PSML-CPCA templates yielded in their best performing configuration a TP_0 of 88% with an additional workload reduction of ∼30%. Workload reductions of ∼50% are achieved with a TP_0 of 86%. The following experiments are run to test whether the binary PSML is able to achieve a higher biometric performance than the smaller binary PSML-CPCA, although the base performance is lower.

Again, the best-performing $\mathcal{H}$ and $\mathcal{W}$ need to be found. The experiment is run the same way as for the binary PSML-CPCA Bloom filter. In the most basic experiment, the binary PSML

Bloom filter		**SMR**	**Workload**			**Recognition Performance**			
$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
5	4	0.2	$6.71 * 10^6$	2.500%	512.0%	85.3%	83.6%	82.5%	7.9%
5	4	0.1	$6.71 * 10^6$	2.500%	512.0%	85.5%	82.7%	81.2%	9.2%
6	4	0.3	$5.59 * 10^6$	2.083%	426.9%	83.9%	82.7%	80.5%	9.9%
7	4	0.4	$4.79 * 10^6$	1.786%	364.8%	82.4%	80.5%	79.9%	10.5%
8	6	0.5	$1.11 * 10^7$	4.167%	853.2%	82.0%	80.5%	79.0%	11.4%
7	4	0.6	$4.79 * 10^6$	1.786%	365.4%	80.9%	77.0%	76.3%	14.1%
7	4	0.7	$4.79 * 10^6$	1.786%	365.4%	78.7%	76.1%	72.1%	18.3%
8	8	0.8	$3.35 * 10^7$	12.500%	2560.0%	71.2%	70.2%	66.2%	24.2%
8	8	0.9	$3.35 * 10^7$	12.500%	2560.0%	67.3%	61.9%	52.7%	37.7%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table 9.17: Best achieved binary PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tested binarisation mask threshold with their corresponding $\mathcal{H}$ and $\mathcal{W}$ ($\mathcal{T} = 64$), ordered by $TP_0, TP_{0.1}$.

is able to achieve higher TP_0 values (table 9.17) than the binary PSML-CPCA, although the

[12] Workload reduction on top of the workload-reduction-by-feature-reduction.

performance loss compared to the baseline is too high.

By inspecting the bit-errors between two mated binary PSML templates, it is clearly visible that the binary PSML also suffers from a high bit-error rate. Even at a high mask threshold of 0.9, a bit-error rate of ∼14% is recorded. Figure 9.17 exemplary shows the bit-error between mated templates for different mask thresholds.

(a) Threshold 0.1 (b) Threshold 0.4 (c) Threshold 0.9

Figure 9.17: Bit-errors between two mated binary PSML templates from one subject (i.e. one instance) at three different binarisation mask thresholds (white = bit-error).

Therefore, the same bit-error correction methods as in the binary PSML-CPCA experiments are tested for the binary PSML. Both transposing the PSML and using majority voting

Bit-error correction	**Bloom filter** $\mathcal{W}$	$\mathcal{H}$	**SMR** MT	**Workload** ω_{256}	F_B*	F_R†	**Recognition Performance** $TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
Transposed	4	5	0.1	$1.34 * 10^7$	5.000%	1024.0%	84.5%	84.1%	83.0%	7.4%
	4	4	0.2	$8.38 * 10^6$	3.125%	640.0%	84.4%	82.8%	82.1%	8.3%
	6	4	0.3	$5.59 * 10^6$	2.083%	426.9%	83.8%	81.9%	81.6%	8.8%
	...‡									
Majority Voting	5	4	0.1	$6.71 * 10^6$	2.500%	512.0%	84.7%	83.0%	81.6%	8.8%
	5	4	0.2	$6.71 * 10^6$	2.500%	512.0%	84.7%	84.0%	81.6%	8.8%
	6	4	0.3	$5.59 * 10^6$	2.083%	426.9%	84.1%	82.0%	80.9%	9.5%
	...‡									
Transposed & Majority Voting	4	5	0.1	$1.34 * 10^7$	5.000%	1024.0%	84.3%	83.4%	82.9%	7.5%
	4	4	0.2	$8.38 * 10^6$	3.125%	640.0%	84.8%	82.9%	82.3%	8.1%
	6	6	0.3	$1.49 * 10^7$	5.556%	1137.9%	82.3%	81.3%	79.7%	10.7%
	...‡									

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.8.

Table 9.18: Top three achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tested binarisation mask threshold with their corresponding $\mathcal{H}$ and $\mathcal{W}$ ($\mathcal{T} = 64$) using the binary PSML with several bit-error reduction strategies, ordered by mask threshold.

increased the biometric performance, but transposing and using majority voting fails to increase the biometric performance compared to only using the transposed PSML. Unfortunately, transposing the binary PSML introduces additional workload due to the Bloom filter transformation

process.

Since the error introduced by the Bloom filter is still too high with the bit-error correction methods, the additional comparison strategy as for the binary PSML-CPCA is tested for the normal, transposed and majority voting binary PSML. As listed in table 9.19, with enabled

Bit-error	**Bloom filter**		**SMR**	**Workload**			**Recognition Performance**			
correction	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
None	8	6	0.6	$1.12 * 10^7$	4.179%	855.6%	89.8%	89.5%	89.0%	1.4%
	7	7	0.7	$2.19 * 10^7$	8.175%	1674.2%	90.1%	89.1%	88.9%	1.5%
	9	**4**	**0.5**	$\mathbf{3.76 * 10^6}$	**1.401%**	**286.7%**	**89.2%**	**88.8%**	**88.8%**	**1.6%**
	...‡									
Transposed	6	6	0.5	$1.49 * 10^7$	5.568%	1140.4%	89.4%	88.9%	88.5%	1.9%
	5	5	0.6	$1.07 * 10^7$	4.012%	821.7%	89.8%	88.7%	88.0%	2.3%
	5	8	0.7	$5.37 * 10^7$	20.008%	4098.4%	89.8%	88.8%	87.9%	2.5%
	...‡									
Majority Voting	8	5	0.8	$6.74 * 10^6$	2.512%	514.5%	89.8%	89.5%	88.9%	1.5%
	7	4	0.7	$4.82 * 10^6$	1.798%	368.0%	90.2%	89.4%	88.6%	1.8%
	6	5	0.6	$8.98 * 10^6$	3.346%	685.4%	90.1%	89.1%	88.5%	1.9%
	...‡									

* $\mathcal{F}$ compared to the PSML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.9.

Table 9.19: Top three achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ using the binary PSML with and without bit-error reduction strategies, ordered by TP_0.

additional comparison and without bit-error reduction strategies, a biometric performance of up to $TP_0 = 89\%$ is reached (0.9%-points more than the binary PSML baseline, equals the binary PSML-CPCA baseline) but with an increase of ∼33% in workload. Sacrificing 0.2% points in TP_0, a workload reduction of ∼55% is reached. With $\mathcal{F} = 0.701\%$, the overall workload reduction is still smaller than the workload reduction of the binary PSML-CPCA Bloom filter ($\mathcal{F} = 0.172\%$, $TP_0 = 88\%$).

All previous binary SMR Bloom filter experiments have been run at $\mathcal{T} = 64$. The experiments were also run with $\mathcal{T} = 32$ and $\mathcal{T} = 16$, with their results listed in table 9.20. Again, using $\mathcal{T} = 16$ introduces too much error in the means of TP_0. Unfortunately, the best configuration with $\mathcal{T} = 32$ fails to achieve a much higher workload reduction then the third best $\mathcal{T} = 64$ configuration and loses ∼0.6%-points TP_0.

Finally, two last experiments are run at $\mathcal{T} = 32$ and $\mathcal{T} = 64$ with enabled tree pre-selection. Using the final binary PSML comparison with tree pre-selection at $\mathcal{T} = 64$ achieves a biometric performance of $TP_0 = 88.9\%$ with a pre-selection of $t = {}^5/_{16}$ trees (table 9.21). Compared to the naïve binary PSML approach, this is an TP_0 increase of 0.9% and an additional workload

Bloom filter			SMR	Workload			Recognition Performance			
$\mathcal{T}$	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
16	5	5	0.5	$5.40 * 10^6$	2.012%	412.1%	82.4%	81.8%	81.6%	8.8%
	4	5	0.6	$6.74 * 10^6$	2.512%	514.5%	84.1%	83.8%	83.3%	7.1%
	4	6	0.7	$1.12 * 10^7$	4.179%	855.6%	83.3%	82.5%	81.9%	8.5%
	...‡									
32	**7**	**4**	**0.7**	$\mathbf{3.62 * 10^6}$	**1.351%**	**277.1%**	**89.0%**	**88.1%**	**88.0%**	**2.3%**
	7	4	0.5	$3.62 * 10^6$	1.351%	277.1%	89.2%	88.4%	87.9%	2.5%
	5	4	0.6	$5.06 * 10^6$	1.887%	386.5%	89.1%	88.5%	87.8%	2.6%
	...‡									

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.10.

Table 9.20: Top three achieved binary PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML comparison using $\mathcal{T} = 32$ and $\mathcal{T} = 16$ trees.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	5	0.1	$3.59 * 10^6$	1.340%	274.5%	72.1%	72.0%	71.7%	18.7%
2	4	5	0.2	$3.80 * 10^6$	1.418%	290.5%	79.1%	78.7%	78.0%	12.4%
3	4	4	0.6	$2.52 * 10^6$	0.940%	192.6%	82.9%	82.1%	81.2%	9.2%
					...‡					
19	7	4	0.7	$2.65 * 10^6$	0.989%	202.2%	89.6%	88.8%	88.8%	1.6%
20	**7**	**4**	**0.7**	$\mathbf{2.72 * 10^6}$	**1.017%**	**208.0%**	**89.8%**	**89.0%**	**88.9%**	**1.5%**
21	4	7	0.6	$2.22 * 10^7$	8.271%	1694.0%	89.8%	89.4%	88.6%	1.8%
					...‡					
35	7	7	0.7	$1.74 * 10^7$	6.517%	1335.0%	90.1%	89.1%	88.9%	1.5%
					...‡					
44	8	6	0.6	$1.05 * 10^7$	3.918%	802.5%	89.8%	89.5%	89.0%	1.4%
					...‡					

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.11.

Table 9.21: Best achieved binary PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

reduction down to ∼70%. The configuration with $\mathcal{T} = 32$ (table 9.22) fails to yield equal TP_0 values at a comparable workload reduction.

However, the binary PSML Bloom filter does not accomplish the overall workload reduc-

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	5	0.4	$2.02 * 10^6$	0.754%	154.2%	52.3%	52.2%	51.9%	38.5%
2	5	5	0.8	$1.87 * 10^6$	0.699%	143.4%	62.7%	62.5%	62.5%	27.9%
3	5	5	0.8	$2.12 * 10^6$	0.793%	162.6%	70.0%	69.8%	69.8%	20.6%
...‡										
15	4	5	0.6	$6.42 * 10^6$	2.395%	490.2%	88.4%	87.7%	86.7%	3.7%
16	4	5	0.6	$6.74 * 10^6$	2.512%	514.5%	88.6%	88.0%	87.0%	3.4%
17	4	5	0.6	$7.05 * 10^6$	2.629%	538.2%	88.7%	88.0%	87.0%	3.4%
...‡										
24	7	4	0.7	$3.32 * 10^6$	1.240%	254.1%	88.8%	87.9%	87.8%	2.6%
25	7	4	0.7	$3.44 * 10^6$	1.282%	262.4%	88.8%	87.9%	87.8%	2.6%
26	**7**	**4**	**0.7**	$\mathbf{3.55 * 10^6}$	**1.324%**	**270.7%**	**88.8%**	**88.0%**	**87.9%**	**2.5%**

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.12.

Table 9.22: Best achieved binary PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML comparison and tree pre-selection of $t = 1, \ldots, 26$ trees at $\mathcal{T} = 32$.

tion[13] and biometric performance of the binary PSML-CPCA Bloom filter. Using an additional real-valued PSML comparison as in the binary PSML-CPCA Bloom filter experiments is not feasible: the real-valued PSML does not fulfil the unlinkability and renewability template protection requirements. Therefore, keeping real-valued PSML is a privacy issue, and thus is irresponsible for real-world scenarios.

For the sake of completeness, the appended table A.13 lists the results of such approach. With a biometric performance of $TP_0 = 90.2\%$[14] at a relative workload reduction of ~64% compared to the naïve binary PSML approach, this approach outperforms every other binary and Bloom filter approach in the means of biometric performance, but the workload is still approximately 5 times higher than the best binary PSML-CPCA Bloom filter and approximate 2 times higher than the naïve binary PSML-CPCA approach. Even an increase in TP_0 of 0.1% compared to the real-valued PSML-Baseline is achieved with $F_B = 6.653\%$. At least this result hints at a confirmation of the at-least-one-false-match problem from section 2.2.3: the biometric performance increased by reducing the workload.

9.4.3 Binary SMR-CPCA Bloom filter-verification

In [DRB17], the Bloom filter achieved the best verification performance with comparatively small block sizes. Looking at the biometric performance for the indexing approach with a small

[13] Workload reduction compared to the real-valued PSML-Baseline.

[14] When doubling the workload due to a bad Bloom filter block size, a TP_0 of 90.4% is reached.

Bloom filter block size, it has to be expected that the Bloom filter will not be feasible for verification purposes with binary PSML-CPCA templates.

As shown in table 9.23, the best block size for verification purposes is $\mathcal{W} = 4, \mathcal{H} = 2$, resulting

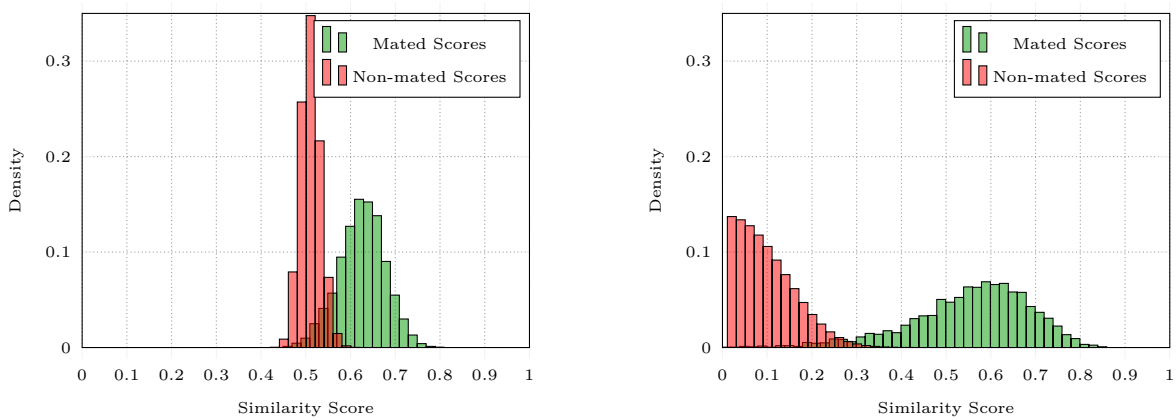
For every	Bloom filter $\mathcal{W}$	Bloom filter $\mathcal{H}$	SMR MT	Workload ω_1	Workload F_B*	Workload F_R†	Recognition Performance EER	Recognition Performance EER Loss to PSLM-Baseline
	4	**2**	**0.5**	$\mathbf{2.56 * 10^3}$	**0.244%**	**50.00%**	**2.8%**	**0.7%**
	4	3	0.5	$3.41 * 10^3$	0.326%	66.7%	3.2%	1.1%
	4	4	0.2	$5.12 * 10^3$	0.488%	100.0%	3.3%	1.2%
$\mathcal{H}$	5	5	0.2	$6.55 * 10^3$	0.625%	128.0%	3.2%	1.1%
	5	6	0.3	$1.09 * 10^4$	1.042%	213.3%	4.2%	2.1%
	4	7	0.2	$2.34 * 10^4$	2.232%	457.1%	4.9%	2.8%
	4	8	0.5	$4.09 * 10^4$	3.906%	800.0%	5.2%	3.1%
	2	2	0.5	$5.12 * 10^3$	0.488%	100.0%	3.0%	0.9%
	3	2	0.5	$3.41 * 10^3$	0.326%	66.7%	3.0%	0.9%
	4	**2**	**0.5**	$\mathbf{2.56 * 10^3}$	**0.244%**	**50.0%**	**2.8%**	**0.7%**
$\mathcal{W}$	5	2	0.5	$2.04 * 10^3$	0.195%	40.0%	3.0%	0.9%
	6	2	0.6	$1.70 * 10^3$	0.163%	33.3%	3.1%	1.0%
	7	2	0.6	$1.46 * 10^3$	0.140%	28.6%	3.1%	1.0%
	8	2	0.6	$1.28 * 10^3$	0.122%	25.0%	3.2%	1.1%

* F compared to the PSML ω_1.

† F compared to the binary PSML-CPCA ω_1.

Table 9.23: Best achieved EER by applying the Bloom filter to the binary PSML-CPCA, sorted by each $\mathcal{W}$ and $\mathcal{H}$.

in a workload reduction of 50% with an EER loss of 0.9%-points compared to the real-valued PSML-Baseline and real-valued PSML-CPCA. Looking upon the score distribution, for the in



(a) Bloom filter ($\mathcal{W} = 4,\ \mathcal{H} = 2$)

(b) binary PSML-CPCA

Figure 9.18: Verification score distribution (Y-axis normalised to 1) of the Bloom filter configuration achieving the best EER ($\mathcal{W} = 4, \mathcal{H} = 2$) for the binary PSML-CPCA in comparison to the score distribution of the binary PSML-CPCA.

table 9.23 highlighted configuration, in figure 9.18, the bad EER is easily explainable by the strong overlap of the genuine and impostor scores.

9.4.4 Binary SMR Bloom filter-verification

The binary PSML achieved a higher biometric identification performance in the basic Bloom filter approach. Therefore, it is expected that the binary PSML also results in a higher biometric verification performance. The results of the verification experiment, listed in table 9.24,

For	Bloom filter		SMR	Workload			Recognition Performance	
every	$\mathcal{W}$	$\mathcal{H}$	MT	ω_1	F_B*	F_R†	EER	EER Loss to PSLM-Baseline
	3	**2**	**0.1**	$\mathbf{2.18 * 10^4}$	**2.083%**	**426.9%**	**2.50%**	**0.40%**
	2	3	0.1	$4.37 * 10^4$	4.167%	853.1%	2.68%	0.60%
	3	4	0.1	$4.37 * 10^4$	4.167%	853.1%	2.73%	0.63%
$\mathcal{H}$	2	5	0.1	$1.05 * 10^5$	10.000%	2047.9%	2.76%	0.66%
	5	6	0.1	$6.99 * 10^4$	6.667%	1365.1%	2.94%	0.84%
	2	7	0.1	$3.00 * 10^5$	28.567%	5851.3%	2.89%	0.79%
	2	8	0.1	$5.24 * 10^5$	50.000%	10239.6%	3.13%	1.03%
	2	2	0.1	$3.28 * 10^4$	3.125%	640.0%	2.53%	0.43%
	3	**2**	**0.1**	$\mathbf{2.18 * 10^4}$	**2.083%**	**426.9%**	**2.50%**	**0.40%**
	4	2	0.1	$1.64 * 10^4$	1.562%	320.0%	2.61%	0.51%
$\mathcal{W}$	5	2	0.1	$1.31 * 10^4$	1.250%	256.0%	2.68%	0.58%
	6	2	0.1	$1.09 * 10^4$	1.042%	213.1%	2.80%	0.70%
	7	2	0.2	$9.36 * 10^3$	0.893%	183.0%	2.89%	0.79%
	8	3	0.3	$1.09 * 10^4$	1.042%	213.1%	3.03%	0.93%

* F compared to the PSML ω_1.

† F compared to the binary PSML-CPCA ω_1.

Table 9.24: Best achieved EER by applying the Bloom filter to the binary PSML, sorted by each $\mathcal{W}$ and $\mathcal{H}$.

confirm these expectations. Looking at the score distribution in figure 9.19 for the highlighted configuration, the genuine and impostor comparison scores are more separated than those for the binary PSML-CPCA Bloom filter, thus achieving the lower EER, but still featuring a high overlap. However, in consideration of the achieved workload reduction[15] and the EER, applying the Bloom filter on the binary SMR for verification purposes is not feasible at least for high noise environments.

9.5 CPCA-Tree-indexing Approach

Applying the Bloom filter to the binary PSML-CPCA with additional real-valued comparison outperformed the naïve binary PSML-CPCA approach in terms of both workload and biometric

[15]The workload ω does not consider the cost of transforming the binary input in the new Bloom filter representation, therefore the real workload is a fraction higher.

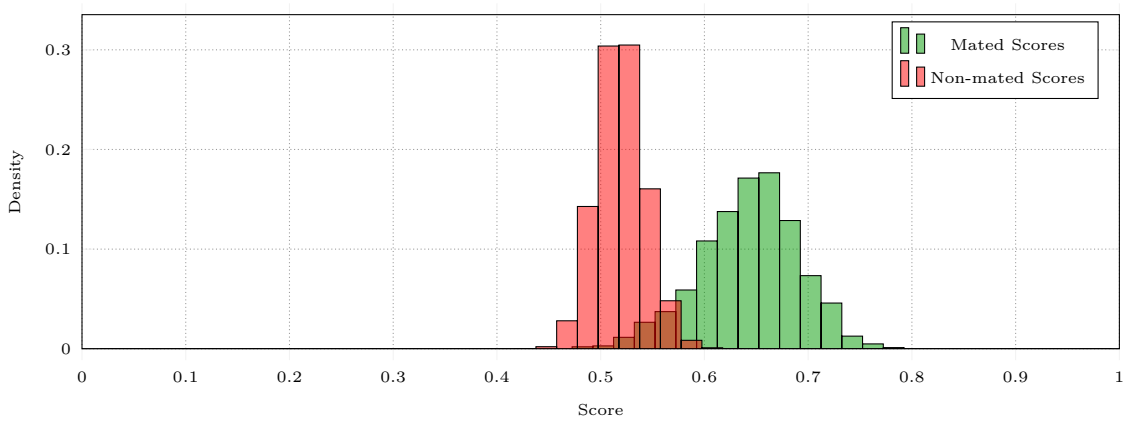


Figure 9.19: Verification score distribution of the Bloom filter configuration achieving the best EER ($\mathcal{W} = 4, \mathcal{H} = 2$) for the binary PSML.

performance. The next experiments are run to determine whether the introduced CPCA-Tree is able to outperform the Bloom filter approach in terms of workload reduction or biometric performance in one of its three types.

9.5.1 Basic implementation

In the basic configuration, the CPCA-Tree is built using the binary PSML-CPCA created with the high mask threshold configuration from section 9.3.4. As already discussed in section 7.1, employing more than 8 templates per tree introduces a very high population count in the (root) nodes, and thus is not feasible. Therefore, the experiments are run with $\mathcal{T} = 32$ and $\mathcal{T} = 64$.

Construction		SMR			Workload			Recognition Performance				
Type	$\mathcal{T}$	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	Γ_B*	Γ_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
PSML-CPCA-Applied (PSML-CPCA-A)	64	0.49	0.6	0.65	$1.31 * 10^6$	0.4883%	100.0%	4.141%	86.5%	85.4%	84.1%	6.3%
		0.48	0.6	0.65	$1.31 * 10^6$	0.4883%	100.0%	4.062%	84.7%	83.4%	83.4%	7.0%
		0.47	0.7	0.66	$1.31 * 10^6$	0.4883%	100.0%	4.531%	84.5%	83.3%	83.3%	7.1%
	32	0.48	0.7	0.65	$9.83 * 10^5$	0.3662%	75.0%	5.625%	85.2%	85.2%	84.8%	5.6%
		0.48	0.6	0.65	$9.83 * 10^5$	0.3662%	75.0%	5.625%	85.2%	85.2%	84.8%	5.6%
		0.48	0.6	0.65	$9.83 * 10^5$	0.3662%	75.0%	5.703%	85.5%	84.5%	83.8%	6.6%
PSML-CPCA-Components (PSML-CPCA-C)	64	**0.46**	**0.6**	**0.70**	$\mathbf{1.31 * 10^6}$	**0.4883%**	**100.0%**	**3.828%**	**90.1%**	**89.8%**	**89.8%**	**0.6%**
		0.46	0.6	0.70	$1.31 * 10^6$	0.4883%	100.0%	3.984%	90.6%	90.0%	89.7%	0.7%
		0.46	0.6	0.70	$1.31 * 10^6$	0.4883%	100.0%	3.828%	90.2%	89.9%	89.7%	0.7%
	32	0.49	0.7	0.65	$9.83 * 10^5$	0.3662%	75.0%	7.656%	87.0%	86.1%	85.9%	4.5%
		0.47	0.6	0.65	$9.83 * 10^5$	0.3662%	75.0%	7.266%	86.8%	86.6%	85.8%	4.6%
		0.47	0.6	0.65	$9.83 * 10^5$	0.3662%	75.0%	7.656%	86.6%	86.5%	85.8%	4.6%
PSML-CPCA-Mixed (PSML-CPCA-M)	64	0.46	0.6	0.65	$1.31 * 10^6$	0.4883%	100.0%	3.672%	90.0%	90.0%	89.8%	0.6%
		0.46	0.6	0.65	$1.31 * 10^6$	0.4883%	100.0%	3.750%	89.9%	89.8%	89.5%	0.9%
		0.46	0.6	0.65	$1.31 * 10^6$	0.4883%	100.0%	3.594%	90.2%	89.7%	89.4%	1.0%
	32	**0.46**	**0.6**	**0.65**	$\mathbf{9.83 * 10^5}$	**0.3662%**	**75.0%**	**5.234%**	**88.9%**	**88.9%**	**88.8%**	**1.6%**
		0.46	0.6	0.65	$9.83 * 10^5$	0.3662%	75.0%	5.391%	88.8%	88.8%	88.4%	2.0%
		0.46	0.6	0.65	$9.83 * 10^5$	0.3662%	75.0%	5.234%	89.0%	88.6%	88.3%	2.1%

* Γ compared to the PSML ω_{256}.
† Γ compared to the naïve binary PSML-CPCA ω_{256}.

Table 9.25: Top three achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for each of the three different template types with PSML-CPCA input at $\mathcal{T} = 64$ and $\mathcal{T} = 32$.

First, the performance of the three PSML-CPCA types are compared at $\mathcal{T} = 64$ and

$\mathcal{T} = 32$. The top three of every type and $\mathcal{T}$ combination is listed in table 9.25. Simply using the SMR-CPCA-A type of the PSML-CPCA, further called PSML-CPCA-A, fails to yield good TP_0, $TP_{0.1}$ and $TP_{0.5}$ due to a high overlap between genuine and impostor scores. Employing both components (SMR-CPCA-C) of the PSML-CPCA, further called PSML-CPCA-C, drastically increases the biometric performance of the CPCA-Tree and reduces the PER by a fraction. However, the hybrid type (SMR-CPCA-M) of the PSML-CPCA, further called PSML-CPCA-M, achieves a equivalent biometric performance as the PSML-CPCA-C for $\mathcal{T} = 64$ but outperforms it for $\mathcal{T} = 32$ due to the reduced population in the sign-bit.
Even in the most basic implementation, the CPCA-Tree is able to increase the biometric performance compared to the naïve binary PSML-CPCA approach without additional workload reduction ($\mathcal{T} = 64$; PSML-CPCA-M and PSML-CPCA-C), and nearly reaches the biometric performance of the naïve binary PSML-CPCA with a workload reduction of 25% ($\mathcal{T} = 32$; PSML-CPCA-M).

9.5.2 Tree pre-selection

By narrowing down the feasible types for $\mathcal{T} = 64$ to the PSML-CPCA-C and PSML-CPCA-M, for $\mathcal{T} = 32$ to PSML-CPCA-M, the next experiments are run to determine the workload reduction capabilities with tree pre-selection. As in the Bloom filter sections, the tree pre-selection experiments for $\mathcal{T} = 64$ are run with $t = 1, \ldots, 48$ trees and $t = 1, \ldots, 26$ trees for $\mathcal{T} = 32$ since selecting more trees would increase the workload.

Beginning with the PSML-CPCA-C, the tree pre-selection results are listed in table 9.26. The quick decay of the biometric performance is very conspicuous: already at a pre-selection of $t = 1/2$ trees, the biometric performance loses 0.5%-points in TP_0 while only achieving an additional workload reduction of ∼13% compared to the naïve PSML-CPCA baseline.

It can be observed from table 9.27 that the PSML-CPCA-M CPCA-Tree performs much better with enabled tree pre-selection. With as little TP_0 loss as 0.2%-points compared to the naïve binary SMR-CPCA approach, a workload reduction of ∼32% is achieved. A workload reduction of ∼50% is achievable with a performance gain of ∼1%-point compared to the naïve binary PSML-CPCA approach.
These better results are trivially explainable with figure 7.3 and table 7.1. $\mathcal{T} = 64$ yields 4 templates per tree. Comparing four merged (joined using a binary OR-operation) sign-bits (figure 7.3g) and four merged applied-bit[16] (figure 7.3i), the merged sign-bit has an average population rate of 93.5% while the applied-bit has an average population rate of 61.4%. Therefore, the root node of a CPCA-Tree using PSML-CPCA-C templates at $\mathcal{T} = 64$ is nearly set completely and thus does not contain any useful information. A tree pre-selection is therefore rendered serendipity, which may yield acceptable results with a high pre-selection rate, but drastically

[16] Which are identical to the sign-bit of the PSML-CPCA-M.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.47	0.64	0.65	$3.48 * 10^5$	0.1297%	26.6%	49.61%	49.3%	48.8%	48.6%	41.8%
2	0.49	0.63	0.65	$3.69 * 10^5$	0.1373%	28.1%	37.27%	61.0%	60.5%	60.4%	30.0%
3	0.49	0.63	0.65	$3.89 * 10^5$	0.1450%	29.7%	29.14%	68.9%	68.3%	68.1%	22.3%
					...‡						
31	0.46	0.62	0.70	$9.63 * 10^5$	0.3586%	73.4%	4.69%	90.1%	89.3%	89.3%	1.1%
32	0.46	0.62	0.70	$9.83 * 10^5$	0.3662%	75.0%	4.53%	90.2%	89.4%	89.4%	1.0%
33	0.46	0.62	0.65	$1.00 * 10^6$	0.3738%	76.6%	4.06%	89.8%	89.7%	89.5%	0.9%
					...‡						
39	0.46	0.62	0.70	$1.13 * 10^6$	0.4196%	85.9%	4.14%	89.9%	89.7%	89.7%	0.7%
40	**0.46**	**0.62**	**0.70**	$\mathbf{1.15 * 10^6}$	**0.4272%**	**87.5%**	**3.98%**	**90.0%**	**89.8%**	**89.8%**	**0.6%**
					...‡						
46	0.46	0.62	0.70	$1.27 * 10^6$	0.4730%	96.9%	3.83%	90.0%	89.8%	89.8%	0.6%
47	0.46	0.62	0.70	$1.29 * 10^6$	0.4807%	98.4%	3.75%	90.1%	89.8%	89.8%	0.6%
48	0.46	0.62	0.70	$1.31 * 10^6$	0.4883%	100.0%	3.75%	90.1%	89.8%	89.8%	0.6%

* $\mathcal{F}$ compared to the PSML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.14.

Table 9.26: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using PSML-CPCA-C templates at $\mathcal{T} = 64$.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.64	0.65	$3.48 * 10^5$	0.1297%	26.6%	28.75%	70.2%	69.5%	69.3%	21.1%
2	0.49	0.64	0.65	$3.69 * 10^5$	0.1373%	28.1%	18.44%	79.0%	78.4%	78.0%	12.4%
3	0.49	0.67	0.70	$3.89 * 10^5$	0.1450%	29.7%	14.22%	83.0%	82.7%	81.9%	8.5%
					...‡						
16	0.47	0.61	0.65	$6.55 * 10^5$	0.2441%	50.0%	5.08%	89.3%	89.2%	89.0%	1.4%
					...‡						
21	0.46	0.62	0.65	$7.58 * 10^5$	0.2823%	57.8%	4.22%	89.8%	89.8%	89.5%	0.9%
22	0.46	0.62	0.65	$7.78 * 10^5$	0.2899%	59.4%	4.14%	89.8%	89.8%	89.6%	0.8%
23	0.46	0.62	0.65	$7.99 * 10^5$	0.2975%	60.9%	3.91%	89.9%	89.9%	89.7%	0.7%
					...‡						
28	**0.46**	**0.62**	**0.65**	$\mathbf{9.00 * 10^5}$	**0.3357%**	**68.7%**	**3.75%**	**90.0%**	**90.0%**	**89.8%**	**0.6%**
					...‡						
48	0.46	0.62	0.65	$1.31 * 10^6$	0.4883%	100.0%	3.67%	90.0%	90.0%	89.8%	0.6%

* $\mathcal{F}$ compared to the PSML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.15.

Table 9.27: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using PSML-CPCA-M templates at $\mathcal{T} = 64$.

increases the PER when using small pre-selection rates.

Looking at the results for $\mathcal{T} = 32$ using the PSML-CPCA-C templates in table 9.28 a workload

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.60	0.70	$1.95 * 10^5$	0.0725%	14.84%	55.23%	43.9%	43.8%	43.7%	46.7%
2	0.49	0.60	0.75	$2.25 * 10^5$	0.0839%	17.19%	41.25%	57.4%	57.3%	57.0%	33.4%
3	0.49	0.60	0.75	$2.56 * 10^5$	0.0954%	19.53%	33.20%	65.2%	64.5%	64.5%	25.9%
					...‡						
20	0.46	0.62	0.65	$7.78 * 10^5$	0.2899%	59.38%	7.11%	87.7%	87.7%	87.6%	2.8%
21	0.46	0.62	0.65	$8.09 * 10^5$	0.3014%	61.72%	6.72%	88.1%	88.1%	88.0%	2.4%
22	0.46	0.62	0.65	$8.40 * 10^5$	0.3128%	64.06%	6.48%	88.3%	88.3%	88.1%	2.3%
23	0.46	0.62	0.65	$8.70 * 10^5$	0.3242%	66.41%	6.40%	88.4%	88.4%	88.2%	2.2%
24	0.46	0.62	0.66	$9.01 * 10^5$	0.3357%	68.75%	6.02%	88.7%	88.7%	88.5%	1.9%
25	0.46	0.62	0.66	$9.32 * 10^5$	0.3471%	71.09%	5.94%	88.7%	88.7%	88.5%	1.9%
26	**0.46**	**0.62**	**0.66**	$\mathbf{9.63 * 10^5}$	**0.3586%**	**73.44%**	**5.62%**	**88.8%**	**88.8%**	**88.7%**	**1.7%**

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.16.

Table 9.28: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 26$ trees using PSML-CPCA-M templates at $\mathcal{T} = 32$.

reduction using lesser trees and lesser trees with tree pre-selection is not feasible: using $\mathcal{T} = 64$ with a small tree pre-selection yields a much better workload reduction with respect to the biometric performance. The CPCA-Tree with $\mathcal{T} = 32$ using the PSML-CPCA-C templates suffers a high PER referable to the same problem as for the PSML-CPCA-C templates: a higher population rate in the root node of every tree aggravates a successful tree pre-selection. Therefore, it is advisable to keep the population rate in the tree roots smaller than ∼75%, desirable at ∼50% to ∼70%.

At this point, the PSML-CPCA-M results in an additional workload reduction of 50% compared to the naïve binary PSML-CPCA approach at the best biometric performance achieved with the binary PSML-CPCA Bloom filter, respectively with the biometric performance of the naïve binary PSML-CPCA approach. It is hardly expectable to achieve a higher workload reduction with the CPCA-Tree than with the binary PSML-CPCA Bloom filter. However, looking at the low PER of the CPCA-Tree it might be possible to increase the biometric performance of the binary PSML-CPCA without increasing the workload compared to the naïve approach. The next experiments are run to achieve this goal.

9.5.3 Additional binary comparison

As already conducted with the Bloom filter approaches, an additional binary PSML-CPCA template comparison could help to increase the biometric performance since impostor and genuine scores could be more separated. However, using the additional comparison approach is expected to only affect the PSML-CPCA-A templates since CPCA-Trees with PSML-CPCA-C

or PSML-CPCA-M templates already undertake these comparisons while traversing the trees. This expectation is proven in the appended table A.17 in comparison with table 9.27: using an additional binary PSML-CPCA template comparison for the PSML-CPCA-M does not yield any improved biometric performance but increases the workload by one comparison.

An experiment using the additional binary comparison for CPCA-Trees with PSML-CPCA-A templates is skipped. Due to the larger PER in CPCA-Trees with PSML-CPCA-A templates, it is not expected that an additional comparison will increase the biometric performance through the level of CPCA-Trees with PSML-CPCA-M templates.

9.5.4 Masking out most common bits

Masking out the most common bits in a binary PSML-CPCA template could increase the biometric performance since it is more difficult for impostor queries to be accepted while genuine queries are ideally unaffected. Since the PSML-CPCA-M templates achieve the highest biometric performance paired with the highest workload reduction, the PSML-CPCA-M templates with $\mathcal{T} = 64$ and $\mathcal{T} = 32$ are used in this experiment.

To create a mask with the most common bits, a heat map is used: the set bits of all enrolled templates are summed up and all sums are normalised to $[0, 1]$. The mask is created by selecting all normalised sums exceeding a given threshold[17].
Masking out the most common bits successfully reduced the PER and increased TP_0 by 0.1%-points for both $\mathcal{T} = 64$ (table 9.29) and $\mathcal{T} = 32$ (table 9.30). For $\mathcal{T} = 64$, a bit threshold of 0.6 to 0.7 seems feasible.

It has to be noted that the experiment can be further optimised. The heat map used for this experiment was created using a fixed binarisation threshold of 0.7. Better results could be achieved when creating the heat map with every tested mask threshold.

9.5.5 Additional real-valued comparison

Employing an additional real-valued PSML-CPCA comparison noteably increased the biometric performance for the Bloom filter approaches. Since the CPCA-Tree yield comperatively low PER, it could also gain in TP_0. The experiment is only run with $1, \ldots, 32$ pre-selected trees: selecting more trees would increase the workload due to the additional real-valued comparison and the tree pre-selection workload.
The TP_0 values in tables 9.31 and 9.32 are disappointing, while the $TP_{0.1}$ values indicate a much higher performance. When analysing the TP values for every threshold at $\mathcal{T} = 64$, it is noticeable that the bad TP_0 bases on one single impostor query that early exceeds the decision

[17] Called *bit threshold* in the tables.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.49	0.65	0.65	0.7	$3.53 * 10^5$	0.1316%	26.9%	28.91%	70.0%	69.7%	69.3%	21.1%
2	0.49	0.64	0.70	0.5	$3.74 * 10^5$	0.1392%	28.5%	18.67%	79.1%	78.4%	78.4%	12.0%
3	0.49	0.64	0.70	0.6	$3.94 * 10^5$	0.1469%	30.1%	14.22%	82.9%	82.1%	82.0%	8.5%
					...‡							
16	0.46	0.62	0.70	0.6	$6.60 * 10^5$	0.2460%	50.4%	4.844%	89.5%	89.1%	89.1%	1.3%
					...‡							
21	0.46	0.62	0.65	0.8	$7.63 * 10^5$	0.2842%	58.2%	4.219%	89.8%	89.8%	89.5%	0.9%
22	0.46	0.62	0.65	0.7	$7.83 * 10^5$	0.2918%	59.8%	4.219%	89.8%	89.8%	89.6%	0.8%
23	0.46	0.62	0.65	0.7	$8.04 * 10^5$	0.2995%	61.3%	3.984%	89.9%	89.9%	89.7%	0.7%
					...‡							
28	**0.46**	**0.62**	**0.70**	**0.6**	$\mathbf{9.06 * 10^5}$	**0.3376%**	**69.1%**	**3.828%**	**90.2%**	**89.9%**	**89.9%**	**0.5%**
					...‡							
46	0.46	0.62	0.70	0.5	$1.27 * 10^6$	0.4749%	97.3%	3.750%	90.2%	89.9%	89.9%	0.5%
47	0.46	0.62	0.70	0.6	$1.30 * 10^6$	0.4826%	98.8%	3.672%	90.2%	89.9%	89.9%	0.5%
48	0.46	0.62	0.70	0.6	$1.32 * 10^6$	0.4902%	100.4%	3.672%	90.2%	89.9%	89.9%	0.5%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.18.

Table 9.29: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using PSML-CPCA-M templates at $\mathcal{T} = 64$ with an additional binary PSML-CPCA comparison and masking out most common set bits.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.49	0.60	0.70	0.9	$2.00 * 10^5$	0.0744%	15.23%	55.23%	43.9%	43.8%	43.7%	46.7%
2	0.49	0.60	0.75	0.8	$2.30 * 10^5$	0.0858%	17.58%	41.25%	57.5%	57.4%	57.0%	33.4%
3	0.49	0.60	0.75	0.5	$2.61 * 10^5$	0.0973%	19.92%	32.97%	65.4%	64.7%	64.7%	25.7%
					...‡							
20	0.48	0.66	0.75	0.5	$7.83 * 10^5$	0.2918%	59.77%	6.95%	88.3%	88.2%	87.7%	2.7%
21	0.48	0.66	0.75	0.5	$8.14 * 10^5$	0.3033%	62.11%	6.64%	88.6%	88.5%	88.0%	2.4%
22	0.48	0.66	0.75	0.5	$8.45 * 10^5$	0.3147%	64.45%	6.02%	89.1%	88.7%	88.2%	2.2%
23	0.48	0.66	0.75	0.5	$8.76 * 10^5$	0.3262%	66.80%	5.86%	89.1%	88.8%	88.3%	2.1%
24	0.46	0.62	0.65	0.9	$9.06 * 10^5$	0.3376%	69.14%	6.02%	88.7%	88.7%	88.5%	1.9%
25	0.48	0.66	0.75	0.5	$9.37 * 10^5$	0.3490%	71.48%	5.86%	89.1%	89.1%	88.6%	1.8%
26	0.46	0.62	0.65	0.7	$9.68 * 10^5$	0.3605%	73.83%	5.70%	88.8%	88.8%	88.7%	1.7%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.19.

Table 9.30: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 26$ trees using PSML-CPCA-M templates at $\mathcal{T} = 32$ with an additional binary PSML-CPCA comparison and masking out most common set bits.

threshold. If this query is removed from the query set, $TP_0 = 90.8\%$[18] is achieved when pre-selecting 32 trees and $TP_0 = 90.3\%$ when pre-selecting 16 trees. However, in terms of TP_0, the

[18] ∼0.4%-points more than the naïve real-valued PSML approach; does not increase the workload compared to the naïve binary PSML-CPCA approach while achieving a much higher biometric performance.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.64	0.65	$6.76 * 10^5$	0.2518%	51.56%	28.67%	70.5%	70.2%	69.8%	20.6%
2	0.49	0.64	0.70	$6.96 * 10^5$	0.2594%	53.12%	17.81%	80.5%	80.1%	79.9%	10.5%
3	0.49	0.64	0.70	$7.17 * 10^5$	0.2670%	54.69%	13.98%	84.1%	83.8%	83.6%	6.8%
					...‡						
9	**0.47**	**0.65**	**0.70**	$\mathbf{8.40 * 10^5}$	**0.3128%**	**64.06%**	**6.094%**	**89.6%**	**89.4%**	**88.8%**	**1.7%**
					...‡						
14	0.48	0.66	0.70	$9.42 * 10^5$	0.3510%	71.88%	4.609%	90.6%	90.2%	88.0%	2.4%
15	0.48	0.65	0.70	$9.63 * 10^5$	0.3586%	73.44%	4.219%	90.9%	90.4%	88.1%	2.3%
16	0.48	0.65	0.75	$9.83 * 10^5$	0.3662%	75.00%	4.062%	90.8%	90.3%	88.1%	2.3%
					...‡						
30	0.48	0.64	0.65	$1.27 * 10^6$	0.4730%	96.88%	3.359%	91.2%	90.9%	88.5%	1.9%
31	0.47	0.64	0.70	$1.29 * 10^6$	0.4807%	98.44%	3.359%	91.2%	90.8%	88.5%	1.9%
32	0.47	0.64	0.70	$1.31 * 10^6$	0.4883%	100.00%	3.359%	91.2%	90.8%	88.5%	1.9%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.20.

Table 9.31: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 32$ trees using PSML-CPCA-M templates at $\mathcal{T} = 64$ with an additional real PSML-CPCA comparison.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.60	0.70	$5.22 * 10^5$	0.1945%	39.84%	55.23%	44.1%	43.8%	43.8%	46.7%
2	0.49	0.60	0.75	$5.53 * 10^5$	0.2060%	42.19%	41.25%	58.0%	57.7%	57.3%	33.1%
3	0.48	0.65	0.75	$5.84 * 10^5$	0.2174%	44.53%	32.66%	66.0%	65.9%	65.6%	24.8%
					...‡						
21	0.47	0.64	0.70	$1.14 * 10^6$	0.4234%	86.72%	5.78%	89.6%	89.1%	87%	3.5%
22	0.47	0.64	0.70	$1.17 * 10^6$	0.4349%	89.06%	5.62%	89.7%	89.2%	87%	3.5%
23	0.48	0.62	0.70	$1.20 * 10^6$	0.4463%	91.41%	5.31%	89.5%	89.2%	88.5%	1.9%
24	**0.49**	**0.65**	**0.65**	$\mathbf{1.23 * 10^6}$	**0.4578%**	**93.75%**	**5.08%**	**89.8%**	**89.5%**	**87.3%**	**1.1%**
25	0.49	0.65	0.65	$1.26 * 10^6$	0.4692%	96.09%	4.92%	90%	89.6%	87.5%	2.9%
26	0.48	0.62	0.65	$1.29 * 10^6$	0.4807%	98.44%	4.77%	90.2%	89.8%	87.5%	2.9%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.21.

Table 9.32: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 26$ trees using PSML-CPCA-M templates at $\mathcal{T} = 32$ with an additional real PSML-CPCA comparison.

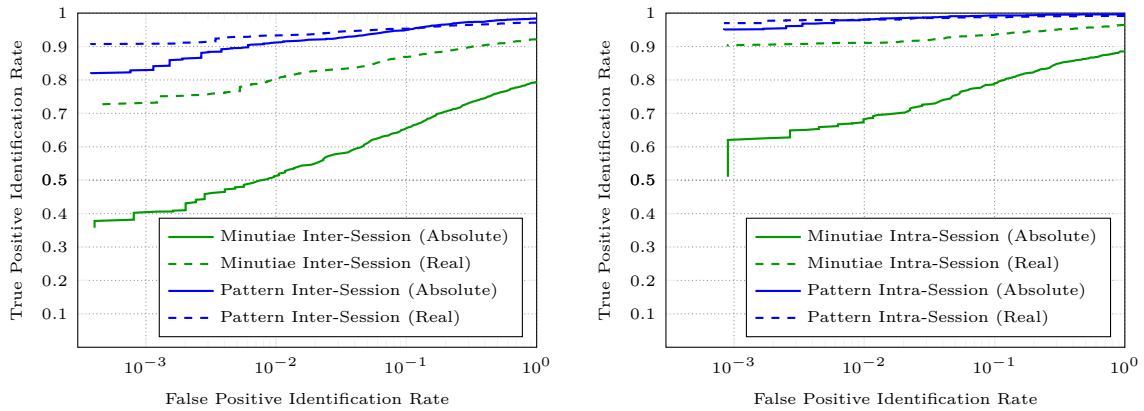
approach using an additional binary comparison yields a higher performance. It has to be noted that the configuration using a pre-selection of 15 trees in table 9.31 surpasses all naïve binary workload-reduction-by-feature-reduction approaches while introducing an additional workload reduction of ∼27% compared to the naïve binary PSML-CPCA approach. The highlighted row

in table 9.31 achieves the highest TP_0 with just 9 searched trees. Since this is the only configuration in this area and $TP_{0.1}$ and $TP_{0.5}$ are lower than the configuration with 10 searched trees, it could be serendipity.

Another experiment with most common bit masking and additional real comparison was run but did not achieve better results. Refer to appended table A.22 for the results.

9.6 Using the full vascular pattern in the SMR

Facing the overall acceptable baseline performance in intra-session experiments but poor baseline performance in the inter-session experiments, an additional experiment has been run to examine whether the SMR hits its limit in this fuzzy environment, or if employing more information achieves a higher biometric performance.
The average count[19] of minutiae in the PolyU dataset is ∼70, whilst the whole vascular pattern is represented by an average of ∼950 points. Therefore, ∼93% of the extracted information is not represented in the SMR. Using more information does not necessarily result in an increased biometric performance, although the chances are high that with a careful selection concerning which parts of the vein pattern to include as input for the SMR, at least the overlap between impostor and genuine attempt scores shrinks.
The results in figure 9.20 show a significant biometric performance improvement over the whole



(a) inter-session (b) intra-session

Figure 9.20: ROC of the PSML-Baseline (minutiae PSML) and full vein pattern SML.

ROC for inter-session and intra-session data. Table 9.33 lists the corresponding TP values for figure 9.20. Note how the full vascular pattern SML in the inter-session experiment achieves a similar or higher biometric performance than the intra-session experiments using minutiae PSML.

[19]Before the removal of spurious minutiae.

Experiment		Recognition Performance			
Session	**Input**	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
Inter	Pattern	92.9%	91.0%	90.5%	1.3%
Intra	Pattern	97.9%	97.7%	97.0%	0.28%
Inter	Minutiae	78.1%	75.1%	72.4%	3.4%
Intra	Minutiae	91.2%	91.1%	90.4%	2.1%

Table 9.33: Biometric performance comparison between minutiae PSML and vascular pattern SML.

Therefore, using the full vascular pattern for the SMR greatly increases the biometric performance without increasing the workload in any means: the full vascular pattern SMR is sampled in the same way and size as the minutiae SMR and the same workload reduction approaches can be utilised. Only the off-line cost of calculating the SMR is increased.

9.7 Summary

In summary, only five workload reduction methods, two using Bloom filter-indexing and three using CPCA-Tree-indexing, are of concern and listed in table 9.34.

	Workload		Recognition Performance			
Name	ω_{256}	F_B*	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
PSML-Baseline†	$2.68 * 10^8$	–	91.2%	91.1%	90.4%	–
Binary PSML-CPCA†	$1.31 * 10^6$	0.4880%	89.7%	89.1%	89.0%	1.4%
Bf CPCA ABC	$9.22 * 10^5$	0.3437%	89.5%	88.9%	88.0%	2.4%
Bf CPCA ARC	$5.23 * 10^5$	0.1952%	89.9%	89.1%	89.0%	1.4%
C-T SCM	$9.01 * 10^5$	0.3357%	90.0%	90.0%	89.8%	0.6%
C-T SCM Masking	$9.06 * 10^5$	0.3376%	90.2%	89.9%	89.9%	0.5%
C-T SCM ARC	$1.13 * 10^6$	0.4196%	91.2%	90.8%	88.5%	1.9%

* F compared to the PSML ω_{256}.

† Naïve approach.

Table 9.34: Summary of relevant experiments, respectively workload reduction methods, with their workload and biometric performance.

The configurations listed in the table are discussed below.

Bf CPCA ABC Using the Bloom filter with the binary PSML-CPCA and an additional binary comparison achieved, as expected due to the usage of the PSML-CPCA, a much better workload reduction, but fails to reach the biometric performance of the binary PSML Bloom filter with additional binary comparison. The best workload-to-biometric-

performance trade-off is achieved with a Bloom filter size of 5×5 with a MT of 0.6 at $t = 19, 20, 21$, yielding $\omega_{256} \approx 9.22 * 10^6$ ($F_B = 0.3437\%$) at $TP_0 = 88.0\%$.

Bf CPCA ARC To address the biometric performance loss of the Bloom filter with binary PSML-CPCA and an additional binary comparison, the additional binary comparison is replaced with an additional real-valued PSML-CPCA comparison (ARC). This method combines a low workload with a high biometric performance. Due to a more workload-reducing Bloom filter size of 6×4, the even smaller workload of $5.23 * 10^5$ ($F_B = 0.1952\%$) at a biometric performance of $TP_0 = 89.0\%$ with $t = 22$ is achieved. This is the recommended Bloom filter approach.

C-T SCM The CPCA-Tree-indexing (C-T) approach using the PSML-CPCA-M (SCM) templates, which proved to achieve the best results in all CPCA-Tree configurations, already achieves the highest biometric performance in the most basic implementation and is able to maintain the high biometric performance with a workload of ∼70% to the naïve binary PSML-CPCA. It has to be noted that the biometric performance is above this naïve approach. The highest TP_0 of 89.8% is reached at $t = 28$ with $F_B = 0.3357\%$, which is ∼30% workload less than the binary PSML-CPCA at a ∼0.8%-points higher TP_0. If the biometric performance of a naïve binary PSML-CPCA is desired, the workload can further be reduced to $F_B = 0.2441$ (half of the naïve approach) with $t = 16$.

C-T SCM Masking Masking out the most common bits set in a PSML-CPCA-M template for CPCA-Tree-indexing increased the maximum TP_0 to 89.9% for the sake of one additional binary comparison[20]. A TP_0 of 89.9% is reached at $t = 28$ with $F_B = 0.3376\%$ ($TP_0 = 89.8\%$ is achieved with $F_B = 0.3147\%$). Again, if the biometric performance of a naïve binary PSML-CPCA is desired, the workload can further be reduced to $F_B = 0.2384\%$ with $t = 15$.

C-T SCM ARC Employing an additional real comparison to the CPCA-Tree with PSML-CPCA-M templates increased both $TP_{0.1}$ and $TP_{0.5}$ to the highest values of all workload reduction experiments, but reduces TP_0. Therefore, this configuration is not recommended for high security scenarios. When selecting $t = 23$, the biometric performance of $TP_0 = 88.5\%$ and $TP_0 = 90.8\%$ is achieved with at $F_B = 0.4196\%$.

Even if the Bloom filter-indexing using the binary PSML achieved an acceptable biometric performance, it is not able to reach the workload reduction achieved with the indexing methods using the PSML-CPCA. Since the CPCA-Tree-indexing methods yield a similar or better biometric performance, paired with an much lower workload, the binary PSML Bloom filter-indexing is not considered necessary to highlight in this summary.

[20] When comparing equal TP_0 values for the two approaches, the workload of the masking approach is actually lower.

For a better understanding of the workload-to-biometric-performance trade-off, figure 9.21 shows the workload reduction achieved in relation to the achieved biometric performance. It is clearly visible that the CPCA-Tree-indexing with and without most common bits masking achieves a better biometric performance up to $F_B \approx 0.23\%$ in the means of TP_0. If a lesser workload is desired, the PSML-CPCA Bloom filter with an additional real-valued PSML-CPCA comparison passes the CPCA-Tree.

To deterministically benchmark the different indexing methods, the euclidean distance between an optimal operation point ($O_{TP} = 90.4\%$, $O_F = 0.1\%$) and each performance-to-workload sample from each experiment as shown in equation 9.1 can be used.

$$\delta(TP, F_B) = \sqrt{(TP - O_{TP})^2 + (F_B - O_F)^2} \tag{9.1}$$

The smallest $\delta(TP, F_B)$ for each experiment rates the corresponding indexing method. Choosing the baseline biometric performance (instead of $TP_0 = 100\%$) as optimal operation point moves the emphasis more to the workload reduction than the biometric performance, to even the scaling differences (TP_0 scales from ∼70% to ∼90% while F_B scales between ∼0.15% and ∼1.0%).

Name	$\delta(TP_0, F_B)$	$F_B{}^*$	TP_0
C-T SCM Masking	0.55	0.3376	89.9
C-T SCM	0.64	0.3357	89.8
Bf CPCA ARC	1.40	0.1952	89.0
Binary PSML-CPCA†	1.45	0.4880	89.0
C-T SCM ARC	1.61	0.3128	88.8
Bf CPCA ABC	2.41	0.3437	88.0

Name	$\delta(TP_{0.1}, F_B)$	$F_B{}^*$	$TP_{0.1}$
C-T SCM ARC	0.40	0.4425	90.9
C-T SCM	1.12	0.3357	90.0
C-T SCM Masking	1.22	0.2995	89.9
Bf CPCA ARC	1.31	0.2155	89.8
Binary PSML-CPCA†	2.04	0.4880	89.0
Bf CPCA ABC	2.21	0.3437	88.9

* F compared to the PSML ω_{256}.

† Naïve approach.

Table 9.35: Ranking for each relevant experiment, respectively workload reduction method, ordered by $\delta(TP, F_B)$ with the optimal operation point at PSML-Baseline biometric performance ($TP_0 = 90.4$) and $F = 0.1\%$.

Table 9.35 lists the minimum recorded $\delta(TP, F_B)$ for every indexing method and figure

9.22 plots their ROC. For both TP_0 and $TP_{0.1}$, the CPCA-Tree-indexing methods clearly yield the smallest $\delta(TP, F_B)$ since the CPCA-Tree achieves the highest TP_0 at an average workload reduction. When moving the optimal operation point to the biometric performance of the naïve binary PSML-CPCA approach and $F = 0.1\%$, the Bloom filter-indexing using binary PSML-CPCA templates with an additional real-valued PSML-CPCA comparison takes the lead as listed in appended table A.33.

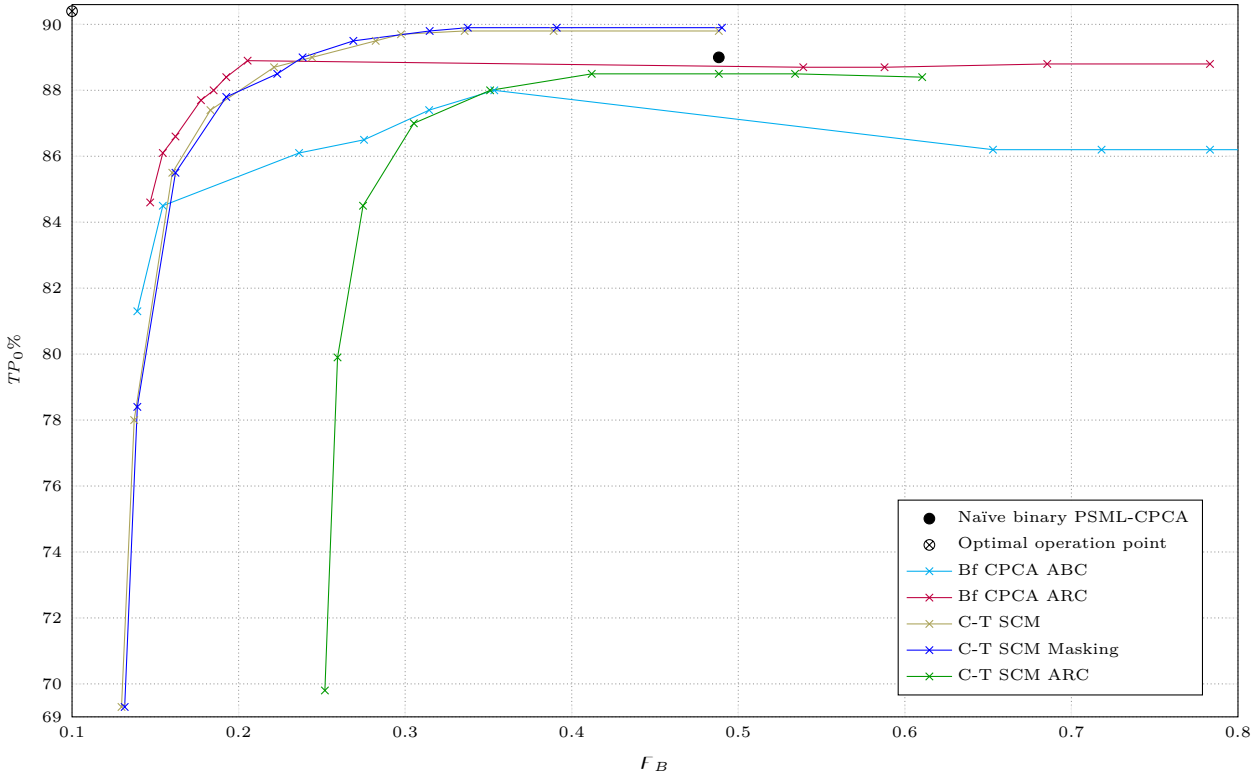
Finally observe the figures in 9.21, for nearly every indexing method, the achieved TP values rise with decreasing t until $t \approx 3/5$. The reduced number of template comparisons needed reduces the number of false matches as predicted by the at-least-one-false-match problem described in section 2.2.3.

The relevant experiments listed in this summary were repeated with inter-session data from the PolyU dataset and also repeated only for the second session of the dataset. They confirm most general statements by showing the same behaviour as the main experiments listed here run on the first session of the dataset. However, some differences between the intra-session and inter-session experiments are recorded and listed below.

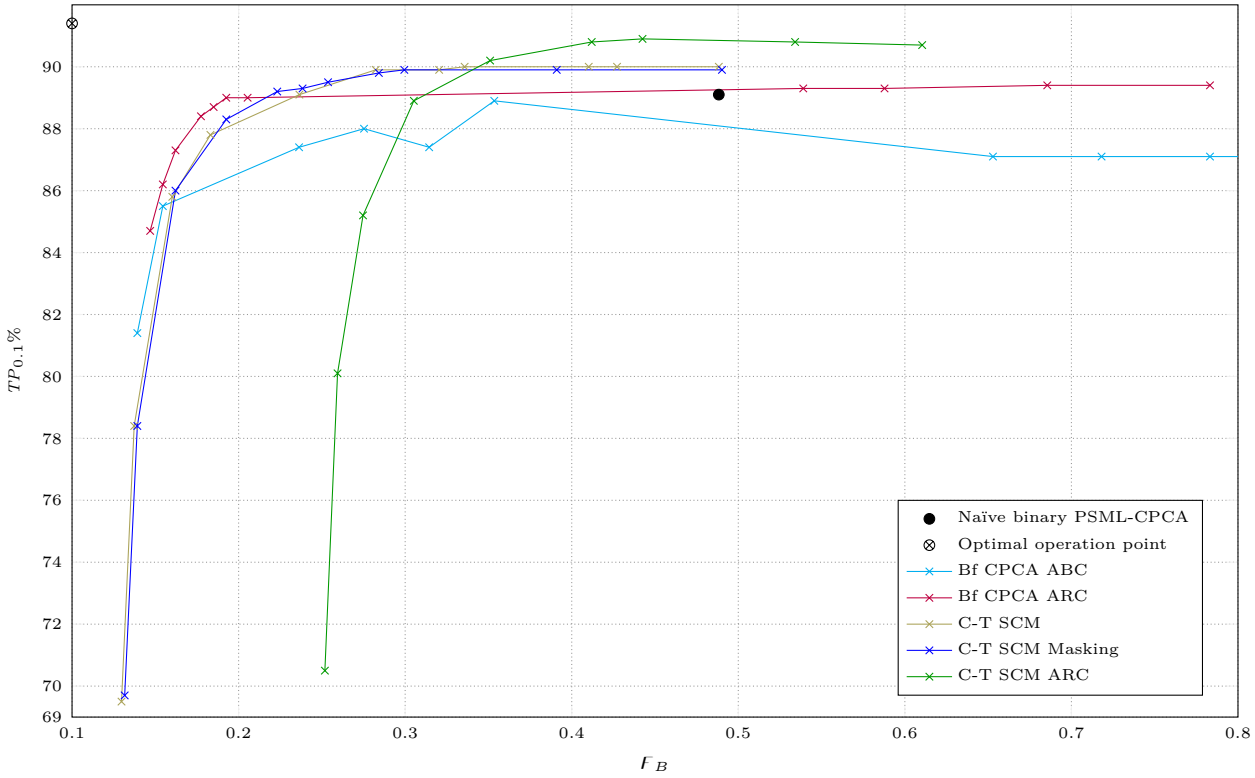
- The Bloom filter-indexing methods suffer in biometric performance from the increased number of bit-errors in the binary SMR and SMR-CPCA templates in inter-session experiments. While the Bloom filter-indexing with additional real-valued SMR-CPCA comparison yields a TP_0 performance loss of $\sim$1.5% for $F_B \approx 0.2\%$ in intra-session experiments, a TP_0 performance loss of $\sim$5% for $F_B \approx 0.2\%$ is recorded in inter-session data.

- For intra-session data, masking out the most common bits in CPCA-Tree-indexing yielded a TP_0 biometric performance gain of $\sim$0.1%, for inter-session data a TP_0 performance gain up to $\sim$1% is recorded.

- While in intra-session experiments using an additional real-valued SMR-CPCA comparison reduced the TP_0 performance for CPCA-Tree-indexing, using an additional real-valued SMR-CPCA comparison in inter-session experiments increased the TP_0 performance by $\sim$0.7%. However, masking out the most common bits yields a higher TP_0 performance.



(a) TP_0



(b) $TP_{0.1}$

Figure 9.21: TP_0 and $TP_{0.1}$ performance in respect to F_B values for relevant experiments at selected t. Used abbreviations: Bf (Bloom filter), C-T (CPCA-Tree), ABC (Additional Binary Comparison), ARC (Additional Real Comparison), SCM (PSML-CPCA-M).

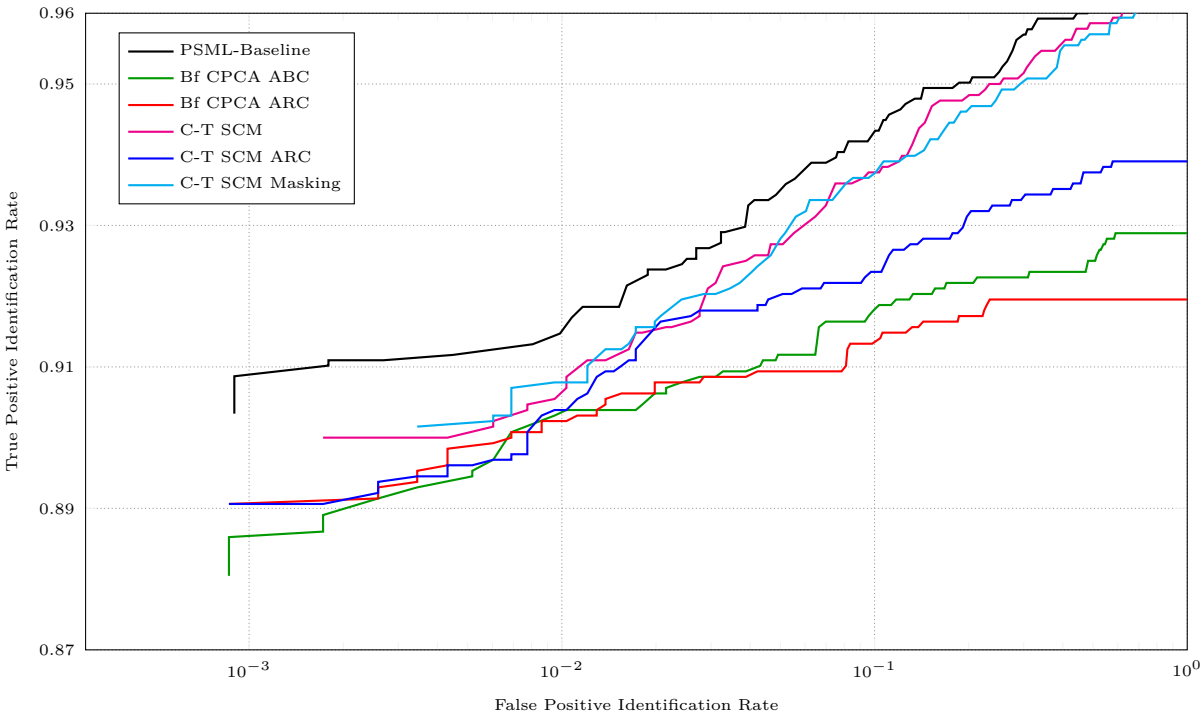


Figure 9.22: ROC for the with $\delta(TP, F_B)$ selected configurations.

9.8 Conclusions

The SMR is a feasible representation for a efficient biometric identification. 75 000 binary SMR comparisons per second, $\sim 0.013ms$ per comparison, are achieved in a single-threaded experiment on a modern Intel i7 CPU. Nearly 300 000 comparisons per second, $\sim 0.003ms$ per comparison, are achieved using the binary SMR-CPCA. With the workload reduction approaches used, the comparison speed is not necessarily increased[21] but the amount of compared templates is reduced. Using e.g. the CPCA-Tree-indexing with $t \approx 3/8$ reduces the amount of compared templates by around $\sim 40\%$ without a significant loss in biometric performance, thus in a $\mathcal{S} = 100\,000$ identification scenario, only $\sim 60\,000$ template comparisons are needed.

However, the baseline biometric performance achieved with the PSML only using extracted minutiae, especially in inter-session scenarios, is not feasible for a real-world scenario. The last experiment presented in section 9.6 shows that the biometric performance can be increased to real-world capable levels by employing more data in the SMR then only bifurcations or endpoints of the vascular pattern. With an increased biometric performance, it might be possible to achieve a lower workload.

[21] Only in some Bloom filter configurations an increased comparison speed is expected, since some configurations result in a smaller representation than the given input.

Chapter 10

Discussion

In this chapter, the achieved results of the experiments in the previous chapter are discussed. Additionally, several future research possibilities for palm vein indexing are presented.

10.1 Results

This section discusses issues that occurred during the course of this project and several observations that did not fit in the result chapter.

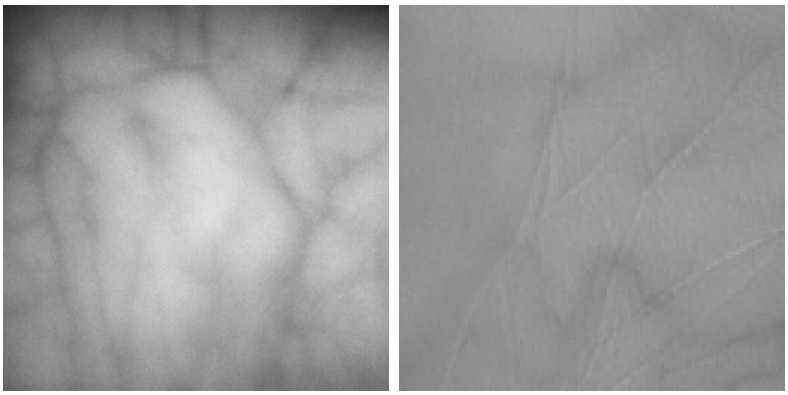
10.1.1 Poor performance on the CASIA and PUT datasets

The results obtained for the CASIA and PUT datasets are unusable. Already by comparing the results yielded by the MHD for the CASIA and PUT datasets with the results for the PolyU it is noticeable that there are issues with the feature extraction pipeline.
In the early stages of this thesis, during the implementation of the feature extraction pipeline only the PolyU dataset was available. Therefore, every algorithm was tested and tuned with the PolyU dataset. Tuning the first stages of the pipeline (e.g. the image enhancement, section 4.3) is a notoriously difficult task that requires much time[1]. Reconfiguration of the first stages also requires a reconfiguration of the following stages.
In order to avoid placing the main objective (workload reduction experiments) at risk, no further reconfiguration of the pipeline for the datasets has been undertaken.

10.1.2 Towards SMR on palm-print datasets

As already mentioned multiple times, the PolyU and the CASIA (e.g. see section 8.1) datasets aim for palm-print modalities. In a real-world application, the raw data acquired by the image-capturing device contains different properties and the image-capturing device utilises approaches as described in section 4.1. Compare the two example images in figure 10.1: the left

[1] Even for the PolyU dataset it should be possible to achieve a higher performance when more time is spent on tuning the first stages.



(a) (b)

Figure 10.1: Raw data received by a (a) proprietary device and (b) by the PolyU dataset.

image is taken with an proprietary device that utilises the listed recommendations; the ROI is extracted using the introduced method. Compared to the PolyU dataset, it is notable that the proprietary device does not capture any skin texture and generates a very clean image of the palm veins. Further, the ROI acquired in the PolyU dataset seems to be very small compared to the ROI extracted from the proprietary device, thus capturing less veins.
The characteristics of the PolyU dataset therefore does not meet some expectations of the SMR:

- The SMR is a minutiae-based approach and therefore does not benefit from the skin texture captured in the PolyU dataset: skin texture is recognized as spurious minutiae.
- The small ROI extracted in the PolyU contains much less minutiae for the SMR: on average, the PolyU dataset contains 70 minutiae per sample; the proprietary device data leads to around 130 minutiae per sample.

Therefore, the PolyU dataset is not a ideal dataset for a palm vein recognition experiments. However, the PolyU dataset is the largest publicly-available dataset to the author's knowledge; it is the only dataset that features enough subjects to enrol a sufficient number of templates in the trees of the Bloom filter or CPCA-Tree indexing approaches while allowing a satisfactory number of impostor comparisons.

10.1.3 SML minutiae reliability quality data impact

Contrary to all expectations, the extracted quality data did not increase the biometric performance in the experiments. On further consideration of this behaviour, it appears that the quality data mainly affects absolute-valued SMR representations. Again, observe figure 9.3, note how both SML and QSML yield a nearly equal ROC in the real-valued representation. When comparing the absolute-valued representations of both SML and QSML, a strong difference is visible: the quality enhanced QSML achieves a higher ROC, since the complex part,

which the absolute-valued representation is based on, is much more sensitive regarding single minutiae.
On the other hand, the SMC suffers from using the quality data (QSMC) in both representations. This most likely is based on the higher sensitivity of the SMC due to the employment of rotation information. The extracted rotation information is much more fuzzy than the location information. Using many minutiae, all of the same value, balances out some fuzziness. When reducing the value of some minutiae with reliability quality information, the other minutiae become much more dominant. If these dominant minutiae contain many errors in their rotation-representation, resulting (mated) SMC also contain many errors.
Therefore, in a fuzzy environment, the quality data should only be applied to the SML.

10.1.4 Ageing and environmental effects in the datasets

In section 9.2.4, the different results for same day (first session; intra-session) and different day (first versus second session; inter-session) are presented. The doubling of the FNIR corresponds to the doubling of the FNMR reported by the International Biometric Group in [The06] between same day and different day results. Observe figure 10.2 where the DET curves of different commercial biometric systems are shown. Note that the FNMR has doubled between same and different day experiments for the Fujitsu PalmSecure palm vein system as listed in table 10.1. Since DET and ROC curves are hardly comparable, figure 10.3 shows the same (first

Security*	**Same-Day/Intra-Session**		**Different-Day/Inter-Session**	
	FNMR	**FMR**	**FNMR**	**FMR**
Low	3.13%	0.0380%	6.17%	0.0395%
Default	4.23%	0.0118%	8.52%	0.0135%
High	5.64%	0.0018%	11.86%	0.0007%

* According to Fujitsu.

Table 10.1: FMR and FNMR for the Fujitsu PalmSecure palm vein biometric system taken from [The06].

session) and different day (first and second session) DET curve achieved with the PolyU dataset in an early state of this project where the pipeline was optimized for verification. With FNMR = 6.5% at FMR = 0.01% (same day) and FNMR = 13.6% at FMR = 0.01% (different day), the FNMR is doubled between same and different day, as expected from the results in [The06].

The main focus of this project is the workload reduction, not a competitive feature extraction pipeline. For this focus, it is irrelevant whether the data is generated by using one or two sessions. Therefore, the exhaustive experiments only use the first session to better inspect the workload reduction performance, not the feature extraction pipeline performance. However, with the results of $FNMR = 6.5\%$ at $FMR = 0.01\%$ by only using minutiae (not

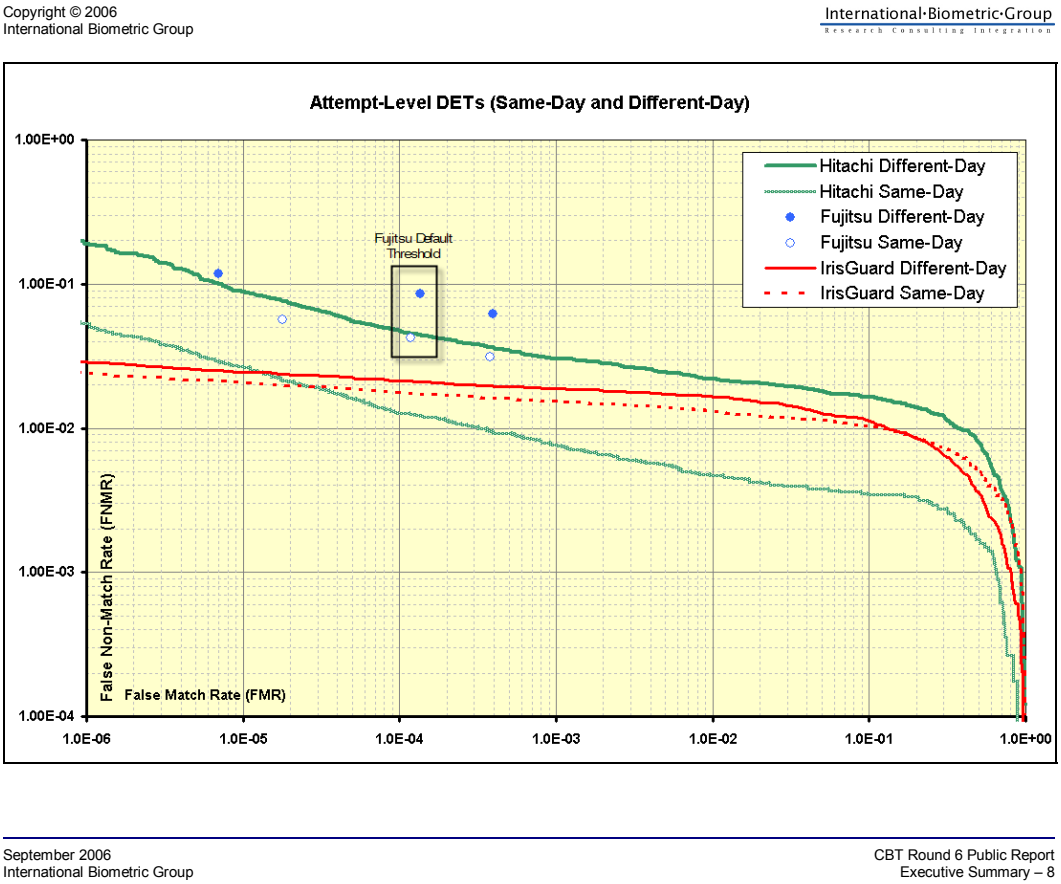


Figure 10.2: DET curves of different commercial biometric systems for same and different days (taken from [The06]).

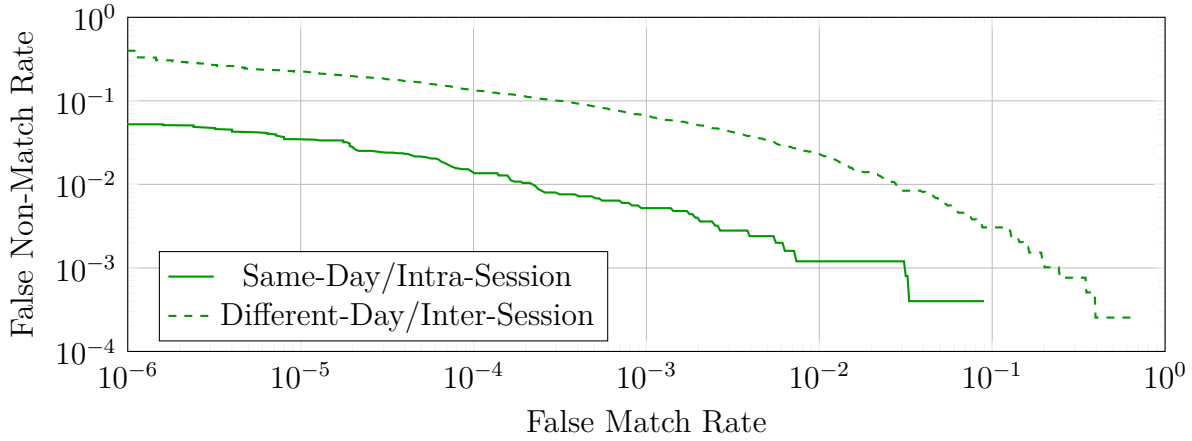


Figure 10.3: DET curve for same (intra-session) and different day (inter-session) verification using the PolyU dataset achieved with the SMR system.

the whole palm vein structure and no other features), the pipeline yields acceptable results compared to the presented commercial product. Further, using the full vascular pattern (like the commercial product), the proposed system outperforms the commercial product in intra-session experiments ($FNMR = 1.28\%$ at $FMR = 0.01\%$) and yields competitive results for inter-session experiments ($FNMR = 8.65\%$ at $FMR = 0.01\%$).

10.1.5 Small number of templates per Bloom filter tree

The reader may have noticed that using $\mathcal{T} = 64$ yields only 4 templates per tree when 256 templates are enrolled. This results in 4 comparisons per tree, therefore no workload reduction is achieved in terms of comparisons. The results showed that when reducing the tree count to $\mathcal{T} = 32$ (8 templates, 6 comparisons per tree) or even $\mathcal{T} = 16$ (16 templates, 8 comparisons per tree) the biometric performance suffers quickly and thus the approach is not feasible.
In both Bloom filter-indexing sections, it is shown that nearly every column per block suffers from bit-errors. This forces very small Bloom filter block heights, whereby the bit errors appear in less blocks. However, reducing the block height automatically introduces more bit collisions, since the Bloom filter features less bits. Therefore, already comparatively similar templates become even more similar in the Bloom filter representation.
While this behaviour is desired in the verification mode, it is a huge disadvantage for the identification scenario: similar enrolled templates are already also more similar in the Bloom filter trees, thus it is more difficult to determine correct traversal direction or tree pre-selection decisions which increases the template pre-selection error. Further, with fewer Bloom filter bits, the tree root and all nodes (not leafs) are populated too quickly, which accelerates the point where incorrect traversal direction or pre-selection decisions happen.

10.1.6 Results compared to state-of-the-art approaches

Different approaches based on local invariant feature extraction algorithms (i.e. [PK11, LT15]) use the whole image for recognition tasks. Even if the publications state conducting a palm vein recognition, both actually use a palm print and palm vein combination.
In [LT15], the verification results for palm print and palm vein are reported separately, albeit the palm vein approach still uses the whole palm image. The authors tested their method only on 100 subjects with 1 reference and 5 probes per subject, with all samples taken from one session. The in [LT15] proposed method achieved a EER of 2.4%, a different implementation using the SURF algorithm achieved 3.1% and an implementation of the local binary patterns (LBP) approach achieved 0.7%. Table 10.2 compares the EER[2] achieved in this project with the reported biometric performance from the publication. Using the minutiae SMR achieves a lower EER than the vascular pattern SURF and GSM approaches but fails to achieve a lower EER than the LBP. With the full vascular pattern SMR, an EER of 0.28% is achieved. This low

[2]Recall, optimisation was done for identification, not for verification.

Input	Method	EER	Verification $(ms)^{\S}$	Source
Minutiae	SMR	2.11%	0.02	*
Minutiae	Binary SMR	2.23%	0.01	*
Minutiae	SMR-CPCA	2.22%	0.005	*
Minutiae	Binary SMR-CPCA	2.24%	0.003	*
Pattern	**SMR**	**0.28%**	**0.02**	*****
Pattern	Binary SMR	0.46%	0.01	*
Pattern	SMR-CPCA	0.32%	0.005	*
Pattern	Binary SMR-CPCA	0.52%	0.003	*
Pattern†	LBP	0.71%	1.5	[LT15]
Pattern†	SURF	3.14%	88	[LT15]
Pattern†	GSM‡	2.39%	1.7	[LT15]
Pattern & Palm Print	LBP	0.42%	1.5	[LT15]
Pattern & Palm Print	SURF	0.43%	88.0	[LT15]
Pattern & Palm Print	GSM‡	0.30%	1.7	[LT15]

* Experiments in this thesis.

† Extended with some palm print structure.

‡ Gray Surface Matching

§ Specifications of machine used in [LT15] unknown; specifications of tested machine: 2014 Intel i7 @ 3.60 GHz, single-threaded.

Table 10.2: Recorded EER of several intra-session experiments to compare the approaches in this thesis with [LT15].

EER outperforms all other methods, even the vascular pattern and palm print combinations, and is much faster. The binary full vascular pattern SMR drops back to a EER of 0.46%, still outperforms the vascular pattern LBP, SURF and GSM, but is placed behind the vascular pattern and palm print combinations.

Another mentionable approach is presented by Abbas and George in [AG14]. This approach differs from the previous since it actually only observes the vein pattern as a biometric characteristic. Unfortunately, the publication does not offer an ISO/IEC-conforming performance reporting for the identification in a closed-set scenario and only reports the Correct Recognition Rate (CRR); *the ratio of the number of samples being correctly classified to the total number of tested samples* according to the publication. Further, is it not known which images of the PolyU dataset are taken as enrolled templates and whether the identification was carried out inter- or intra-session. Therefore, it is difficult to compare these results with those obtained through this project since no CRR optimisation experiments have been run and every identification experiment is run as an open-set. The publication also summarizes other vein recognition approaches for different vein modalities not run on the PolyU dataset.
However, to compare the results achieved in this thesis, experiments have been run to determine

the CRR. Without optimisation to achieve the highest CRR, the inter-session vascular pattern

Data		Type		Recognition Performance			
Input	**Session**	**Method**	**Workload-Reduction**	**CRR**	**EER**	**Identification (ms)‡**	**Source**
Pattern	**Inter**	**LAVD†**	**-**	**99.95%**	**0.24%**	**158.4**	[AG14]
Pattern	Inter	SMR	-	97.7%	1.28%	11.1	*
Pattern	Inter	SMR-CPCA	-	97.7%	1.27%	1.2	*
Pattern	Intra	SMR	-	99.8%	0.28%	11.1	*
Pattern	Intra	SMR-CPCA	-	99.8%	0.32%	1.2	*
Minutiae	Inter	SMR	-	93.7%	3.3%	11.1	*
Minutiae	Intra	SMR	-	94.8%	2.1%	11.1	*
Minutiae	Intra	SMR-CPCA	CPCA-Tree-indexing	94.7%	2.1%	0.9	*

* Experiments in this thesis.

† Local Average of Vein Direction

‡ Specifications of machine used in [AG14] unknown; specifications of tested machine: 2014 Intel i7 @ 3.60 GHz, single-threaded; identification at $\mathcal{S} = 500$.

Table 10.3: Recorded CRR of several experiments to compare the approaches in this thesis with [AG14].

SMR experiment recorded a CRR of 97.7% which did not outperform the Local Average of Vein Direction (LAVD) method in terms of biometric performance. Comparing the execution time of one single identification attempt, the SMR is more than 10 times, and the SMR-CPCA more then 100 times faster than the LAVD. Again, the minutiae SMR are not able to achieve good results compared to the vascular pattern SMR and LAVD.

The results achieved with the vascular pattern SMR were quick experiments without optimisation. Hence, it might be possible to achieve a even higher biometric performance with dedicated research.

10.2 Workload of real-valued SMR templates

Recall (section 9.3.1), the workload of one real-valued SMR (respective SMR-CPCA) comparison is calculated as 32 times the cost of a binary comparison as a floating point contains 32 bits. An experiment has been run to review this claim, since the measurements in section 10.1.6 hint at a workload increase of only ∼30%. 3 000 real-valued SMR (singe precision float) templates and their binary representations[3] were compared 1 000 times on a dedicated, completely stripped down Linux server without graphical or network components. The server features a Intel Core i7-4790 CPU at 3.60 GHz with 8192 kB cache and is equipped with 16 GB RAM. To receive the most clean results, the kernel and all remaining daemons were pinned to core 0, while the experiment was executed on core 2. As a reference, a completely stock MacBook Air (Mid 2011, Intel Core i7-2677M at 1.80 GHz, 4 GB RAM) was used to verify the results.

[3] The binary representation is stored as 128×4 32 bit unsigned integers.

The experiment was repeated multiple times. On the Linux server, the real-valued comparison was only 1.3 times slower; the MacBook Air averaged 1.6 times slower.

Both processors are comparatively old (2011 and 2014). The newer processor (i7-4790) appears to have a much better optimised floating point pipeline, since it is relative floating point speed is much higher than the older i7-2677M. There is no guarantee that these results also apply to other architectures (ARM, PowerPC, …).
It is safe to say, that a real-valued comparison does not yield a workload 32 times higher than the binary approach, but an approximately 2 times higher workload for more realistic calculations is reasonable. However, this can not be deterministically proven or generalised for all architectures or processors of the same architecture. Therefore the naïve approach of calculating the bit-count for each template is used in the results even if this does not reflect real world scenarios or implementations.

10.3 Derived further research topics

As already mentioned in section 10.1.6, during the course of this project several other research topics and ideas emerged that are not addressed in this thesis. This section lists the most promising derived research topics selected by the author, divided into one subsection for other biometric modalities and another for (palm) vein modalities.

10.3.1 (Palm) vein modalities

At least three further research topics for palm vein indexing can be derived from the results and other publications. Two research topics extend or modify the approach presented in this thesis, whereas the third discards the whole SMR approach.

Employing more reference points. It was shown in section 9.6 that using the full vascular pattern as an input for the SMR strengthens the biometric performance in any concern. Without any optimisations, quality assurance or adjustments in any means, an inter-session $TP_0 = 90.5\%$ (EER = 1.3%) and an intra-session $TP_0 = 97.0\%$ (EER = 0.28%) was reached. Therefore, without any optimisations, an increase in TP_0 of ∼18% and ∼7%, compared to the optimised results only using endpoints and bifurcations, is achieved. It can be safely assumed that with careful optimisations and exhaustive research much higher TP_0 can be realised, which opens new possible avenues of research in the area.

Usage of absolute- and real-valued SMR in the CPCA-Tree. In the full vascular pattern experiments, the absolute-valued SMR achieved much higher CRR than the real-valued SMR, while the real-valued SMR achieved much higher TP_0 values. This behaviour could be used in e.g. the CPCA-Tree-indexing by building the CPCA-Tree with absolute-valued binary SMR-CPCA to achieve a low PER and employ a final real-valued SMR-CPCA comparison for a high TP_0. Thus, the advantages of both SMR are combined,

which is expected to increase the overall biometric performance. Using both real-valued and absolute-valued SMR is a non-issue in the means of computional workload since both can be extracted from one single SMR calculation.

Adoption of a different feature extraction approach. In [AG14], a simpler and seemingly more robust feature extraction approach is presented. A first next step could be to reimplement the proposed feature extraction approach and replace the current feature extraction pipeline before the thinning step (see section 4.5). This could be sufficient to increase the overall biometric performance since it may yield more reliable minutiae.
The presented feature extraction pipeline could still be used to extract quality information.

Bloom filter applied upon a not thinned vascular pattern. A more trivial and less complex approach could be to apply the Bloom filter template transformation to a segmented, binary vascular pattern. As an example, applying the Bloom filter on the 7th image of figure 3 in [AG14] with some additional dilation or other enhancements could already achieve acceptable results.

Creation of large-scale palm vein dataset For obvious reasons, the creation of a dedicated, large-scale dataset for palm vein research is recommended. The desireable number of subjects is $\mathcal{S} \geq 1000$ to cover both workload reduction and identification research.

10.3.2 Other modalities

The SMR is a versatile approach that can be used on most minutiae or reference-point based approaches in different modalities. These presented further research topics aim to extend and verify the presented indexing approaches based on the SMR in other modalities.

Fingerprint SMR-CPCA and CPCA-Trees. The feature representation used in this project is extrapolated from and originally designed for the fingerprint modality. While the SMR has not achieved a very high biometric performance using minutiae for the palm vein characteristic, it originally achieved a very high biometric performance for the minutiae of the fingerprint characteristic. This thesis has shown that the Bloom filter and CPCA-Tree indexing approaches can be applied to the SMR to reduce the overall workload without losing a significant amount of biometric performance in this low-quality[4]. Therefore it would be a reasonable next step to analyse the two indexing approaches in a high minutiae quality environment.

Full pattern fingerprint SMR Inspired by the results of the full vascular pattern SMR, this concept could also be applied to the fingerprint modality. [XVK+08] hinted at performance issues with poor quality fingerprints. Employing more of the fingerprint pattern in the

[4] In the means of low minutiae count with many spurious minutiae.

SMR could overcome this problem and enable the usage of this representation for poor quality data.

10.4 Summary

In this chapter, the results of the project have been discussed in detail. First, the poor performance on the CASIA and PUT was discussed, followed by a discussion of several observations along the project. After a comparison of the experiments with state-of-the-art approaches, one issue with the workload reporting was outlined.

Using the SMR for palm vein (and general vascular pattern) biometric systems offers a multitude of future research possibilities, several of which have been briefly outlined in this chapter.

Chapter 11

Conclusions

The interest in biometrics among governmental and industrial operators of access control systems has been steadily growing in recent years. With the adoption of biometrics in consumer products like notebooks or smartphones, the wide public has accepted biometrics and strengthened their trust in such systems. Different huge biometric systems are deployed worldwide nowadays. Said systems are required to deliver high biometric performance while keeping a low computational workload profile. These trends make not only biometrics a relevant and attractive research area, but also workload reduction to increase the efficiency of high biometric performance approaches. This thesis has pertained to the topic of workload reduction for biometric identification in large-scale palm vein databases.

A recently-published biometric indexing approach based on Bloom filters and binary search trees for large-scale iris databases was adopted for the practical work for this thesis. To adopt this indexing approach, the vascular pattern of the raw palm vein images was extracted using an proposed signal processing subsystem. The minutiae - the endpoints and bifurcations - of the extracted vascular pattern where then transformed using a Fourier transformation based approach originally presented for the fingerprint characteristic. When transforming the real-valued representation yielded by the Fourier transformation to a binary form, it is possible to apply the Bloom filter-indexing. After implementing the Bloom filter system with careful configuration of the components, it has been demonstrated that the system is capable of achieving a biometric performance close to the baseline achieved with a naïve implementation of the Fourier representation, while reducing the necessary workload by an additional $\sim$60% compared to a naïve implementation using the reduced binary Fourier representation. Some of the approaches used by the Bloom filter were not feasible and the fuzziness of the vascular pattern prevented a higher workload reduction without losing too much biometric performance. However, the most important approaches have been successfully applied, thus the system appears to be scalable in terms of workload reduction, biometric performance and enrollees.

An additional, less complex, biometric indexing approach merely using a reduced form of

the binary Fourier transformation representation and binary search trees has been presented. It adopts most workload reduction strategies that are used for the Bloom filter-indexing approach. In three of four biometric performance metrics, it outperformed the Bloom filter-indexing while still reducing the workload compared to a naïve implementation using the reduced binary Fourier representation by ∼40%. Since the presented approach follows the same theory and implementations as the binary search trees of the Bloom filter-indexing, it also appears to be scalable in terms of workload reduction, biometric performance and enrollees.

The overall workload is reduced to an average of 0.5% compared to the baseline of the naïve implementation using the Fourier representation in both systems.

Facing the acceptable but not optimal result in the means of biometric performance yielded by the baseline Fourier representation, several further research topics were presented to address this issue. The most promising topic was briefly tested and already achieved a much better biometric performance without any optimisation or tuning. Therefore, the system is not suitable yet for real-world palm vein system deployments but appears to be a promising base for further research with the proposed topics. Furthermore, the workload reduction achieved very promising results, which were limited by the performance of the base system. Since the base system achieved a very high biometric performance for fingerprints, the workload reduction approaches can be adopted to the fingerprint modalities and should yield even better results in terms of workload reduction.

Appendix A

Appendix

A.1 Bloom filter-indexing - Quick traversal direction decision first child favouring

Observe line 7 in algorithm 6: the next decision will be determined with the next-to-last score when the second comparison is skipped. This falsifies the decision making process because it is to be excepted, that the score of the comparison will be higher than the next-to-last score. *In other words, it is very likely, that the decision will always favour the first child if the previous decision selected the second child.*

Algorithm 6 Tree traversal for a single Bloom filter tree with single comparison strategy.

```
1:  function TraverseTree(Node, Query, Threshold, LastScore)
2:      if Node[left] ≠ nil and Node[right] ≠ nil then
3:          scoreLeft ← similarity(Node[left], Query)
4:          if scoreLeft > LastScore then
5:              return TraverseTree(Node[left], Query, Threshold, scoreLeft)
6:          else
7:              return TraverseTree(Node[right], Query, Threshold, LastScore)
8:          end if
9:      end if
10:     score ← similarity(Node[this], Query)
11:     if score > Threshold and score > LastScore then        ▷ Second condition optional
12:         return Node[this]
13:     end if
14:     return nil
15: end function
```

A.2 Experiments

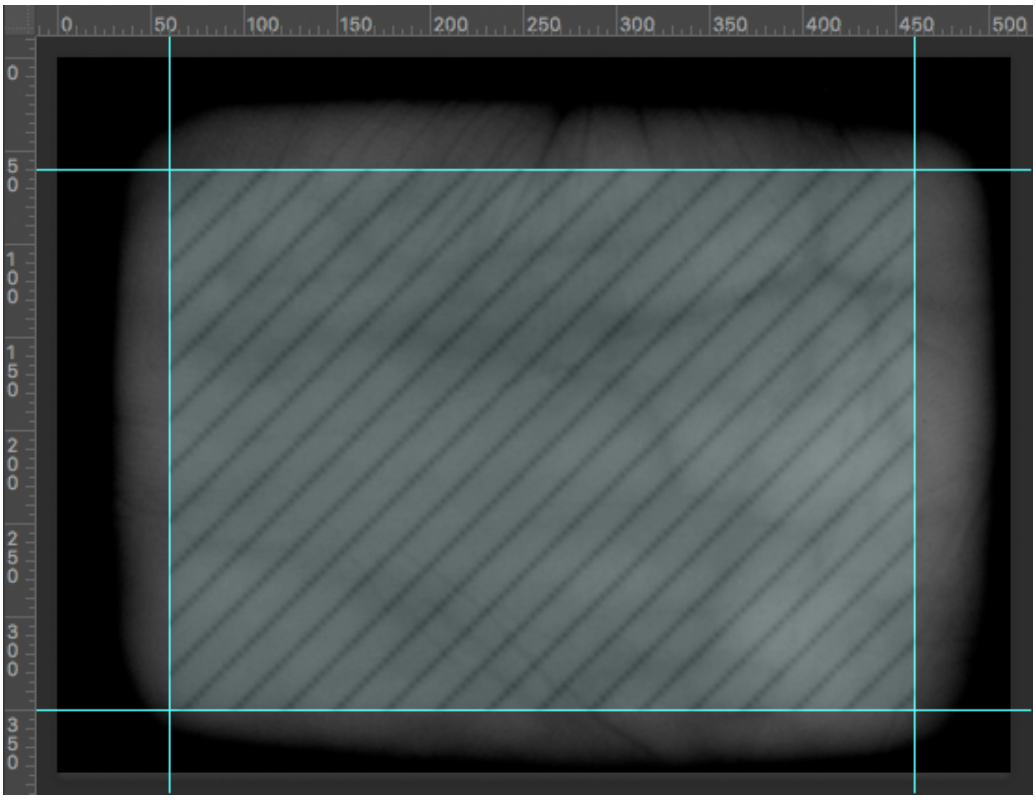


Figure A.1: Used ROI for the PUT dataset.

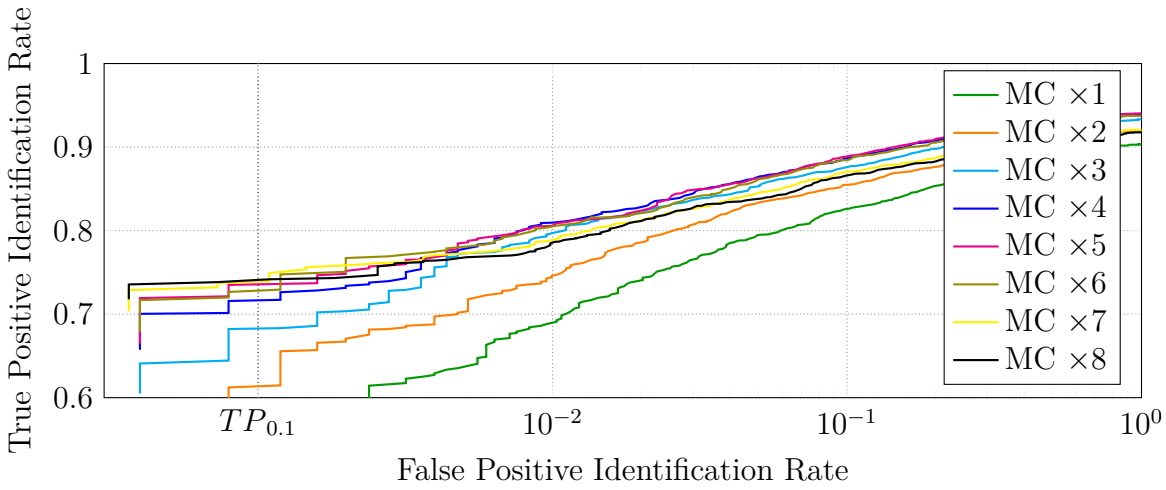


Figure A.2: QSML ROC for the PolyU dataset with different maximum-curvature output gains (multiplication).

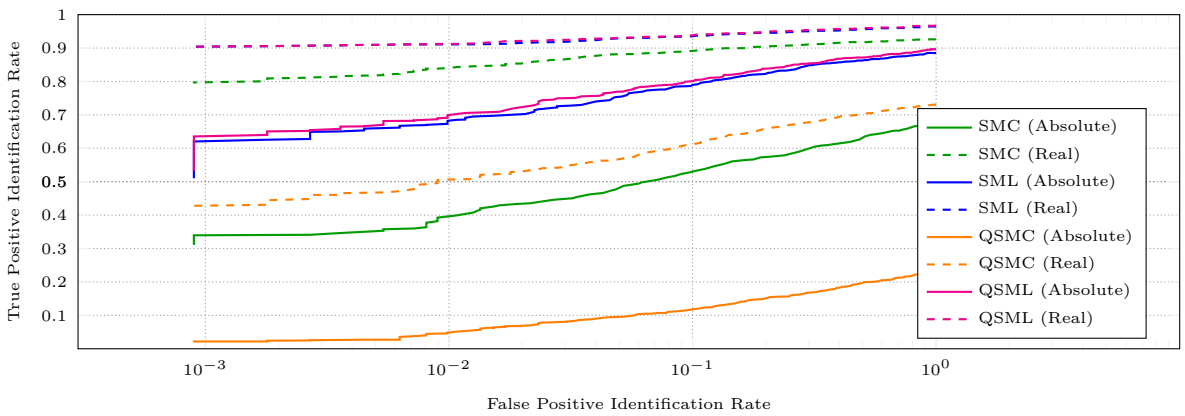


Figure A.3: ROC curves of the SML, SMC, QSML and QSMC type for the first session of the PolyU dataset.

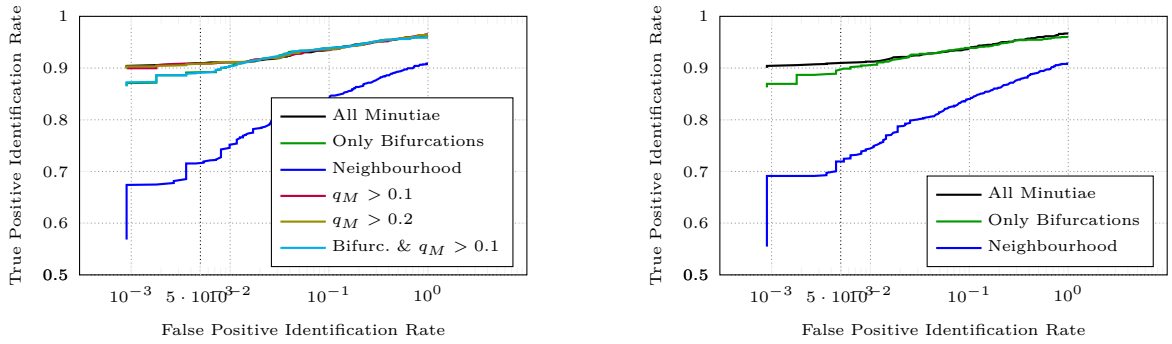
	SMR			Ident.			Verif.
Type	**Pre-Selection**	λ_{min}	λ_{max}	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
SML	$q_M{}^* > 0.2$	0.1	0.51	91.2%	91.1%	90.4%	2.1%
QSML*		0.1	0.55	91.4%	91.1%	90.3%	2.1%
SML		0.1	0.60	90.9%	90.5%	90.0%	2.2%

* Maximum curvature ×7.

Table A.1: Top three configurations ($TP_{0.1}$), for a biometric identification system using the first session PolyU dataset, with their corresponding verification EER, ordered by TP_0.

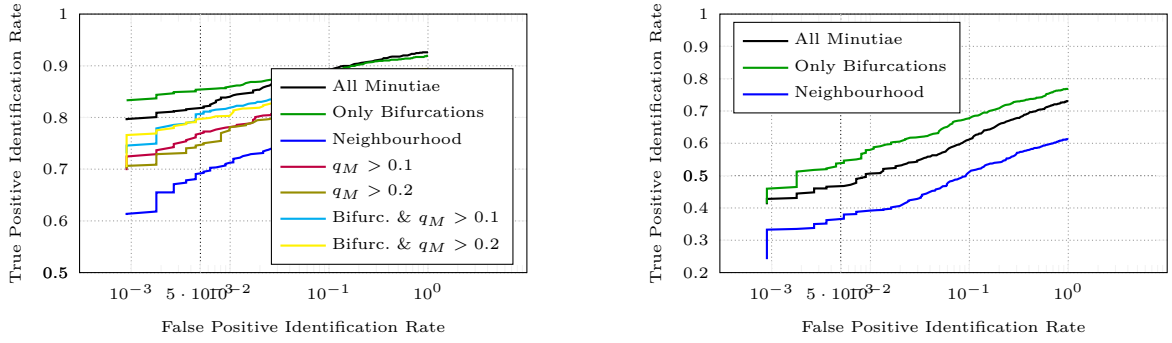
L	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**
5	91.3%	90.9%	89.7%	2.2%
15	91.3%	90.6%	90.6%	2.1%
25	91.6%	90.9%	90.3%	2.1%
50	91.8%	90.6%	90.4%	2.1%
100	91.4%	90.7%	90.2%	2.1%

Table A.2: TP_0, $TP_{0.1}$, $TP_{0.5}$ and EER for the different CPCA feature reduction training set sizes (L).



(a) SML

(b) QSML



(c) SMC

(d) QSMC

Figure A.4: ROC curves of the real-valued SML, SMC, QSML and QSMC type for the first session of the PolyU dataset with applied minutiae selection.

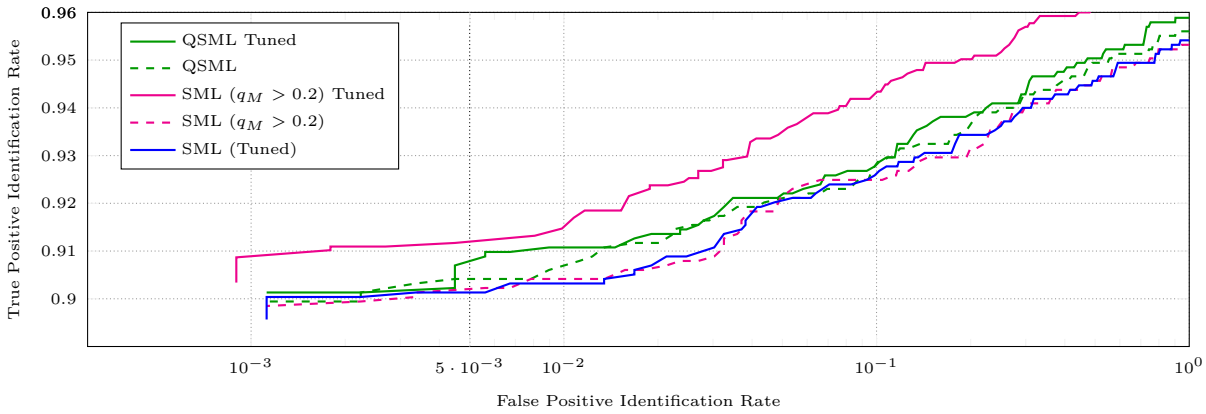


Figure A.5: ROC curves of the most promising SMR types, tuned and untuned, for the first session of the PolyU dataset.

Bit-error correction	Bloom filter $\mathcal{W}$	Bloom filter $\mathcal{H}$	SMR MT	Workload ω_{256}	Workload F_B*	Workload F_R†	Recognition Performance $TP_{0.5}$	Recognition Performance $TP_{0.1}$	Recognition Performance TP_0	TP_0 loss to PSML-Baseline
	4	4	0.1	$1.31 * 10^6$	0.4883%	100%	70.2%	65.4%	61.1%	29.3%
	6	4	0.2	$8.73 * 10^5$	0.3255%	66.67%	66.5%	61.5%	59.5%	30.9%
	4	4	0.3	$1.31 * 10^6$	0.4883%	100%	63.2%	61.9%	57.5%	32.9%
	8	4	0.4	$6.55 * 10^5$	0.2441%	50%	57.3%	53.4%	53%	37.4%
Majority Voting	5	4	0.5	$1.04 * 10^6$	0.3906%	80%	53.1%	48.4%	45.7%	44.7%
	8	5	0.6	$1.04 * 10^6$	0.3906%	80%	49.8%	42.3%	34.8%	55.6%
	6	5	0.7	$1.39 * 10^6$	0.5208%	106.7%	45.2%	42%	39.6%	50.8%
	8	4	0.8	$6.55 * 10^5$	0.2441%	50%	46.9%	34.1%	32.2%	58.2%
	6	5	0.9	$1.39 * 10^6$	0.5208%	106.7%	33.6%	28.4%	26.9%	63.5%
	4	5	0.1	$2.09 * 10^6$	0.7812%	160%	14.9%	13.2%	11.8%	78.6%
	4	4	0.2	$1.31 * 10^6$	0.4883%	100%	13.4%	11.4%	8.75%	81.7%
	4	5	0.3	$2.09 * 10^6$	0.7812%	160%	14.5%	13.4%	10.4%	80%
	4	4	0.4	$1.31 * 10^6$	0.4883%	100%	15.9%	9.92%	6.72%	83.7%
Transposed	4	4	0.5	$1.31 * 10^6$	0.4883%	100%	15.6%	11%	5.47%	84.9%
	4	4	0.6	$1.31 * 10^6$	0.4883%	100%	11.5%	7.97%	6.88%	83.5%
	4	4	0.7	$1.31 * 10^6$	0.4883%	100%	9.61%	7.11%	5.47%	85.9%
	4	4	0.8	$1.31 * 10^6$	0.4883%	100%	7.58%	6.09%	5%	85.4%
	4	4	0.9	$1.31 * 10^6$	0.4883%	100%	6.72%	5.39%	3.75%	86.7%
	4	5	0.1	$2.09 * 10^6$	0.7812%	160%	17%	14%	10.1%	80.3%
	4	4	0.2	$1.31 * 10^6$	0.4883%	100%	14.5%	9.61%	9.61%	80.8%
	4	5	0.3	$2.09 * 10^6$	0.7812%	160%	15.2%	13.2%	12.1%	78.3%
Transposed	4	4	0.4	$1.31 * 10^6$	0.4883%	100%	17%	10.3%	7.42%	83%
&	4	4	0.5	$1.31 * 10^6$	0.4883%	100%	16.2%	9.53%	7.11%	83.3%
Majority Voting	4	4	0.6	$1.31 * 10^6$	0.4883%	100%	11.5%	7.66%	7.66%	82.7%
	4	4	0.7	$1.31 * 10^6$	0.4883%	100%	9.06%	6.48%	5.47%	84.9%
	4	4	0.8	$1.31 * 10^6$	0.4883%	100%	7.58%	5.7%	4.45%	85.9%
	4	4	0.9	$1.31 * 10^6$	0.4883%	100%	6.02%	5.16%	4.38%	86%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.3: Best achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tested binarisation mask threshold with their corresponding $\mathcal{H}$ and $\mathcal{W}$ ($\mathcal{T} = 64$) using the binary PSML-CPCA with several bit-error reduction strategies, ordered by mask threshold.

Bloom filter			SMR	Workload			Recognition Performance			
$\mathcal{T}$	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
	6	4	0.1	$4.42 * 10^5$	0.1647%	33.72%	82.3%	82%	80.9%	9.5%
	4	4	0.2	$6.60 * 10^5$	0.246%	50.39%	84.5%	83.8%	83%	7.43%
	5	4	0.3	$5.29 * 10^5$	0.1972%	40.39%	83.2%	82.6%	82.3%	8.06%
	4	6	0.4	$1.75 * 10^6$	0.6529%	133.7%	83%	82.7%	82.5%	7.9%
16	4	6	0.5	$1.75 * 10^6$	0.6529%	133.7%	83%	82.5%	82.3%	8.06%
	5	5	0.6	$8.43 * 10^5$	0.3144%	64.39%	83.4%	82.5%	82.2%	8.21%
	5	4	0.7	$5.29 * 10^5$	0.1972%	40.39%	80.8%	80.5%	79.8%	10.6%
	5	5	0.8	$8.43 * 10^5$	0.3144%	64.39%	82.8%	82.6%	81%	9.4%
	6	5	0.9	$7.04 * 10^5$	0.2623%	53.72%	80.5%	79.1%	78.8%	11.6%
	6	4	0.1	$6.60 * 10^5$	0.246%	50.39%	87%	86.4%	85.1%	5.32%
	4	7	0.2	$4.49 * 10^6$	1.676%	343.2%	84.2%	83.4%	83%	7.35%
	6	6	0.3	$1.75 * 10^6$	0.6529%	133.7%	85.9%	85.2%	84.1%	6.26%
	8	5	0.4	$7.91 * 10^5$	0.2949%	60.39%	87%	86.6%	86.1%	4.31%
32	7	4	0.5	$5.66 * 10^5$	0.2112%	43.25%	87.1%	86.1%	85.7%	4.7%
	6	6	0.6	$1.75 * 10^6$	0.6529%	133.7%	85.8%	85.2%	83.9%	6.49%
	5	4	0.7	$7.91 * 10^5$	0.2949%	60.39%	87.8%	87.6%	86.6%	3.84%
	5	5	0.8	$1.26 * 10^6$	0.4707%	96.39%	87.7%	87.3%	86.2%	4.15%
	5	4	0.9	$7.91 * 10^5$	0.2949%	60.39%	87.1%	85.6%	84.8%	5.63%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

‡ Full table appended in A.4.

Table A.4: Achieved PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final PSML-CPCA comparison using $\mathcal{T} = 32$ and $\mathcal{T} = 16$ trees.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	$\digamma_B$*	$\digamma_R$†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	4	0.2	$3.53 * 10^5$	0.1316%	26.9%	78.2%	77.3%	77.1%	13.30%
2	4	4	0.3	$3.73 * 10^5$	0.1392%	28.5%	82.0%	81.4%	81.3%	9.10%
3	4	4	0.4	$3.94 * 10^5$	0.1469%	30.1%	84.7%	84.3%	83.4%	6.96%
4	4	4	0.4	$4.14 * 10^5$	0.1545%	31.6%	85.8%	85.5%	84.5%	5.95%
5	4	4	0.2	$4.35 * 10^5$	0.1621%	33.2%	86.2%	85.3%	84.6%	5.79%
6	5	5	0.6	$5.81 * 10^5$	0.2168%	44.4%	87.3%	86.6%	85.3%	5.09%
7	5	5	0.6	$6.08 * 10^5$	0.2265%	46.4%	88.0%	87.2%	85.9%	4.54%
8	5	5	0.6	$6.34 * 10^5$	0.2363%	48.4%	88.4%	87.4%	86.1%	4.31%
9	5	5	0.6	$6.60 * 10^5$	0.2460%	50.4%	88.4%	87.4%	86.2%	4.15%
10	4	4	0.2	$5.37 * 10^5$	0.2003%	41.0%	88.0%	86.6%	86.2%	4.15%
11	4	4	0.2	$5.58 * 10^5$	0.2079%	42.6%	88.0%	86.6%	86.2%	4.15%
12	5	5	0.6	$7.39 * 10^5$	0.2753%	56.4%	88.6%	88.0%	86.5%	3.92%
13	5	5	0.6	$7.65 * 10^5$	0.2851%	58.4%	88.6%	88.0%	86.5%	3.92%
14	5	5	0.6	$7.91 * 10^5$	0.2949%	60.4%	88.8%	88.1%	86.6%	3.76%
15	5	5	0.6	$8.17 * 10^5$	0.3046%	62.4%	89.1%	88.4%	87.5%	2.90%
16	5	5	0.6	$8.43 * 10^5$	0.3144%	64.4%	89.0%	88.4%	87.4%	2.98%
17	5	5	0.6	$8.70 * 10^5$	0.3242%	66.4%	89.1%	88.4%	87.5%	2.90%
18	5	5	0.6	$8.96 * 10^5$	0.3339%	68.4%	89.2%	88.6%	87.7%	2.74%
19	5	5	0.6	$9.22 * 10^5$	0.3437%	70.4%	89.5%	88.9%	88.0%	2.43%
20	5	5	0.6	$9.48 * 10^5$	0.3535%	72.4%	89.5%	88.9%	88.0%	2.43%
21	5	5	0.6	$9.75 * 10^5$	0.3632%	74.4%	89.5%	88.9%	88.0%	2.43%
22	7	4	0.5	$4.49 * 10^5$	0.1676%	34.3%	87.9%	87.0%	86.7%	3.68%
23	5	6	0.4	$1.70 * 10^6$	0.6367%	130.4%	87.4%	87.2%	86.2%	4.23%
24	5	6	0.4	$1.75 * 10^6$	0.6529%	133.7%	87.3%	87.1%	86.2%	4.23%
25	5	6	0.4	$1.79 * 10^6$	0.6692%	137.1%	87.3%	87.0%	86.1%	4.31%
26	5	6	0.4	$1.84 * 10^6$	0.6855%	140.4%	87.3%	87.0%	86.1%	4.31%
27	5	6	0.4	$1.88 * 10^6$	0.7018%	143.7%	87.3%	87.1%	86.2%	4.23%
28	5	6	0.4	$1.92 * 10^6$	0.7181%	147.1%	87.3%	87.1%	86.2%	4.23%
29	5	6	0.4	$1.97 * 10^6$	0.7343%	150.4%	87.3%	87.1%	86.2%	4.23%
30	5	6	0.4	$2.01 * 10^6$	0.7506%	153.7%	87.3%	87.1%	86.2%	4.23%
31	5	6	0.4	$2.05 * 10^6$	0.7669%	157.1%	87.3%	87.1%	86.2%	4.23%
32	5	6	0.4	$2.10 * 10^6$	0.7832%	160.4%	87.3%	87.1%	86.2%	4.23%
33	5	6	0.4	$2.14 * 10^6$	0.7994%	163.7%	87.3%	87.1%	86.2%	4.23%
34	5	6	0.4	$2.18 * 10^6$	0.8157%	167.1%	87.3%	87.1%	86.2%	4.23%
35	5	6	0.4	$2.23 * 10^6$	0.8320%	170.4%	87.3%	87.1%	86.2%	4.23%
36	5	6	0.4	$2.27 * 10^6$	0.8483%	173.7%	87.3%	87.1%	86.2%	4.23%
37	5	6	0.4	$2.32 * 10^6$	0.8645%	177.1%	87.3%	87.1%	86.2%	4.23%
38	5	6	0.4	$2.36 * 10^6$	0.8808%	180.4%	87.3%	87.1%	86.2%	4.23%
39	5	6	0.4	$2.40 * 10^6$	0.8971%	183.7%	87.3%	87.1%	86.2%	4.23%
40	5	6	0.4	$2.45 * 10^6$	0.9134%	187.1%	87.3%	87.1%	86.2%	4.23%
41	5	6	0.4	$2.49 * 10^6$	0.9296%	190.4%	87.3%	87.1%	86.2%	4.23%
42	5	6	0.4	$2.53 * 10^6$	0.9459%	193.7%	87.3%	87.1%	86.2%	4.23%
43	5	6	0.4	$2.58 * 10^6$	0.9622%	197.1%	87.3%	87.1%	86.2%	4.23%
44	5	6	0.4	$2.62 * 10^6$	0.9785%	200.4%	87.3%	87.1%	86.2%	4.23%
45	5	6	0.4	$2.67 * 10^6$	0.9947%	203.7%	87.3%	87.1%	86.2%	4.23%
46	5	6	0.4	$2.71 * 10^6$	1.0110%	207.1%	87.3%	87.1%	86.2%	4.23%
47	5	6	0.4	$2.75 * 10^6$	1.0270%	210.4%	87.3%	87.1%	86.2%	4.23%
48	5	6	0.4	$2.80 * 10^6$	1.0440%	213.7%	87.3%	87.1%	86.2%	4.23%

* $\digamma$ compared to the PSML ω_{256}.

† $\digamma$ compared to the naïve binary PSML-CPCA ω_{256}.

Table A.5: Best achieved PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	5	0.1	$3.16 * 10^5$	0.1179%	24.14%	66.6%	65.9%	65.7%	24.7%
2	4	5	0.1	$3.65 * 10^5$	0.1362%	27.89%	74.2%	73.8%	73.2%	17.2%
3	4	5	0.3	$4.14 * 10^5$	0.1545%	31.64%	76.6%	76.2%	76.0%	14.4%
4	4	5	0.4	$4.63 * 10^5$	0.1728%	35.39%	79.3%	79.1%	78.4%	12%
5	4	5	0.5	$5.13 * 10^5$	0.1911%	39.14%	80.7%	80.2%	80.2%	10.2%
6	4	4	0.3	$3.53 * 10^5$	0.1316%	26.95%	82.3%	81.8%	81.6%	8.76%
7	4	4	0.3	$3.84 * 10^5$	0.1431%	29.30%	83.5%	83.2%	83.1%	7.28%
8	4	5	0.5	$6.60 * 10^5$	0.2460%	50.39%	84.3%	83.8%	83.8%	6.65%
9	4	5	0.5	$7.09 * 10^5$	0.2644%	54.14%	85.1%	84.5%	84.5%	5.95%
10	4	5	0.5	$7.58 * 10^5$	0.2827%	57.89%	85.5%	84.8%	84.8%	5.63%
11	4	5	0.5	$8.07 * 10^5$	0.3010%	61.64%	85.9%	85.1%	85.1%	5.32%
12	4	5	0.5	$8.57 * 10^5$	0.3193%	65.39%	86.3%	85.5%	85.5%	4.85%
13	4	5	0.6	$9.06 * 10^5$	0.3376%	69.14%	87.3%	86.4%	86.1%	4.31%
14	4	5	0.6	$9.55 * 10^5$	0.3559%	72.89%	87.5%	86.6%	86.2%	4.15%
15	4	5	0.6	$1.00 * 10^6$	0.3742%	76.64%	87.8%	86.9%	86.4%	3.99%
16	4	4	0.5	$6.60 * 10^5$	0.2460%	50.39%	87.7%	86.7%	86.6%	3.76%
17	4	5	0.6	$1.10 * 10^6$	0.4108%	84.14%	88.1%	87.2%	86.7%	3.68%
18	4	5	0.6	$1.15 * 10^6$	0.4292%	87.89%	88.1%	87.2%	86.7%	3.68%
19	4	5	0.6	$1.20 * 10^6$	0.4475%	91.64%	88.0%	87.3%	86.1%	4.31%
20	4	4	0.5	$7.83 * 10^5$	0.2918%	59.77%	88.0%	87.1%	86.1%	4.31%
21	4	4	0.5	$8.14 * 10^5$	0.3033%	62.11%	88.0%	87.1%	86.1%	4.31%
22	4	4	0.5	$8.44 * 10^5$	0.3147%	64.45%	88.0%	87.2%	86.2%	4.15%
23	6	5	0.6	$9.33 * 10^5$	0.3478%	71.22%	87.7%	86.6%	86.6%	3.84%
24	4	4	0.5	$9.06 * 10^5$	0.3376%	69.14%	88.2%	87.3%	86.4%	3.99%
25	4	4	0.5	$9.36 * 10^5$	0.3490%	71.48%	88.2%	87.3%	86.4%	3.99%
26	5	4	0.7	$7.75 * 10^5$	0.2888%	59.14%	87.4%	87.2%	86.2%	4.15%
27	5	4	0.7	$7.99 * 10^5$	0.2979%	61.02%	87.5%	87.3%	86.3%	4.07%
28	5	4	0.7	$8.24 * 10^5$	0.3071%	62.89%	87.6%	87.3%	86.4%	3.99%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.6: Best achieved PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 28$ trees at $\mathcal{T} = 32$.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	4	0.2	$3.53 * 10^5$	0.1316%	26.95%	78.9%	78.4%	78.0%	12.4%
2	4	5	0.3	$5.94 * 10^5$	0.2216%	45.39%	83.6%	83.1%	82.7%	7.67%
3	4	4	0.4	$3.94 * 10^5$	0.1469%	30.08%	85.4%	84.7%	84.6%	5.79%
4	4	4	0.4	$4.14 * 10^5$	0.1545%	31.64%	87.0%	86.2%	86.1%	4.31%
5	4	4	0.5	$4.35 * 10^5$	0.1621%	33.20%	87.6%	87.3%	86.6%	3.84%
6	4	4	0.5	$4.55 * 10^5$	0.1698%	34.77%	88.2%	87.9%	87.2%	3.21%
7	4	4	0.5	$4.76 * 10^5$	0.1774%	36.33%	88.8%	88.4%	87.7%	2.67%
8	4	4	0.5	$4.96 * 10^5$	0.1850%	37.89%	89.0%	88.7%	88.0%	2.35%
9	4	4	0.5	$5.17 * 10^5$	0.1926%	39.45%	89.3%	89.0%	88.4%	2.04%
10	4	4	0.5	$5.37 * 10^5$	0.2003%	41.02%	89.6%	89.3%	88.4%	1.96%
11	4	4	0.5	$5.58 * 10^5$	0.2079%	42.58%	89.9%	89.6%	88.7%	1.73%
12	4	4	0.5	$5.78 * 10^5$	0.2155%	44.14%	90.1%	89.8%	88.8%	1.57%
13	4	4	0.5	$5.99 * 10^5$	0.2232%	45.70%	89.9%	88.9%	88.8%	1.57%
14	5	4	0.5	$4.96 * 10^5$	0.1850%	37.89%	89.5%	88.7%	88.6%	1.81%
15	5	4	0.5	$5.13 * 10^5$	0.1911%	39.14%	89.5%	88.7%	88.6%	1.81%
16	5	4	0.5	$5.29 * 10^5$	0.1972%	40.39%	89.5%	88.7%	88.6%	1.81%
17	5	4	0.5	$5.45 * 10^5$	0.2033%	41.64%	89.5%	88.8%	88.7%	1.73%
18	4	5	0.6	$1.11 * 10^6$	0.4169%	85.39%	90.2%	89.4%	88.8%	1.65%
19	4	5	0.6	$1.15 * 10^6$	0.4292%	87.89%	90.2%	89.4%	88.8%	1.65%
20	4	5	0.6	$1.18 * 10^6$	0.4414%	90.39%	90.2%	89.4%	88.8%	1.65%
21	4	5	0.6	$1.21 * 10^6$	0.4536%	92.89%	90.3%	89.5%	88.8%	1.57%
22	6	4	0.5	$5.23 * 10^5$	0.1952%	39.97%	89.9%	89.1%	89.0%	1.42%
23	6	4	0.5	$5.37 * 10^5$	0.2003%	41.02%	89.8%	89.0%	88.9%	1.49%
24	6	4	0.5	$5.51 * 10^5$	0.2054%	42.06%	89.8%	89.0%	88.9%	1.49%
25	6	4	0.5	$5.64 * 10^5$	0.2104%	43.10%	89.8%	89.1%	89.0%	1.42%
26	6	4	0.5	$5.78 * 10^5$	0.2155%	44.14%	89.9%	89.1%	89.0%	1.42%
27	6	4	0.5	$5.92 * 10^5$	0.2206%	45.18%	90.0%	89.1%	89.0%	1.42%
28	4	5	0.6	$1.44 * 10^6$	0.5390%	110.4%	90.2%	89.3%	88.7%	1.73%
29	4	5	0.6	$1.47 * 10^6$	0.5512%	112.9%	90.2%	89.3%	88.7%	1.73%
30	4	5	0.6	$1.51 * 10^6$	0.5634%	115.4%	90.2%	89.3%	88.7%	1.73%
31	4	5	0.6	$1.54 * 10^6$	0.5756%	117.9%	89.9%	89.3%	88.7%	1.73%
32	4	5	0.6	$1.57 * 10^6$	0.5878%	120.4%	89.9%	89.3%	88.7%	1.73%
33	4	5	0.6	$1.61 * 10^6$	0.6001%	122.9%	89.9%	89.3%	88.7%	1.73%
34	4	5	0.6	$1.64 * 10^6$	0.6123%	125.4%	89.9%	89.3%	88.7%	1.73%
35	4	5	0.6	$1.67 * 10^6$	0.6245%	127.9%	90.0%	89.4%	88.8%	1.65%
36	4	5	0.6	$1.70 * 10^6$	0.6367%	130.4%	90.0%	89.4%	88.8%	1.65%
37	4	5	0.6	$1.74 * 10^6$	0.6489%	132.9%	90.0%	89.4%	88.8%	1.65%
38	4	5	0.6	$1.77 * 10^6$	0.6611%	135.4%	90.0%	89.4%	88.8%	1.65%
39	4	5	0.6	$1.80 * 10^6$	0.6733%	137.9%	90.0%	89.4%	88.8%	1.65%
40	4	5	0.6	$1.84 * 10^6$	0.6855%	140.4%	90.0%	89.4%	88.8%	1.65%
41	4	5	0.6	$1.87 * 10^6$	0.6977%	142.9%	90.0%	89.4%	88.8%	1.65%
42	4	5	0.6	$1.90 * 10^6$	0.7099%	145.4%	90.0%	89.4%	88.8%	1.65%
43	6	5	0.6	$1.29 * 10^6$	0.4821%	98.72%	90.2%	89.4%	88.8%	1.65%
44	6	5	0.6	$1.31 * 10^6$	0.4902%	100.4%	90.2%	89.4%	88.8%	1.65%
45	6	5	0.6	$1.33 * 10^6$	0.4983%	102.1%	90.2%	89.4%	88.8%	1.65%
46	6	5	0.6	$1.35 * 10^6$	0.5065%	103.7%	90.2%	89.4%	88.8%	1.65%
47	4	5	0.6	$2.06 * 10^6$	0.7710%	157.9%	90.0%	89.4%	88.8%	1.65%
48	4	5	0.6	$2.10 * 10^6$	0.7832%	160.4%	90.0%	89.4%	88.8%	1.65%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.7: Best achieved binary PSML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final real-valued PSML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

Bit-error correction	Bloom filter $\mathcal{W}$	Bloom filter $\mathcal{H}$	SMR MT	Workload ω_{256}	Workload F_B*	Workload F_R†	Recognition Performance $TP_{0.5}$	Recognition Performance $TP_{0.1}$	Recognition Performance TP_0	TP_0 loss to PSML-Baseline
Transposed	4	5	0.1	$1.34 * 10^7$	5.0%	160%	84.5%	84.1%	83%	7.43%
	4	4	0.2	$8.38 * 10^6$	3.125%	100%	84.4%	82.8%	82.1%	8.29%
	6	4	0.3	$5.59 * 10^6$	2.083%	66.67%	83.8%	81.9%	81.6%	8.76%
	6	4	0.4	$5.59 * 10^6$	2.083%	66.67%	83.5%	81.4%	80.5%	9.9%
	7	6	0.5	$1.27 * 10^7$	4.762%	152.4%	79.2%	78.4%	77.6%	12.8%
	7	5	0.6	$7.66 * 10^6$	2.857%	91.43%	80.9%	77.7%	76.7%	13.7%
	6	4	0.7	$5.59 * 10^6$	2.083%	66.67%	77.1%	74.6%	73%	17.4%
	8	4	0.8	$4.19 * 10^6$	1.562%	50%	76.8%	73.4%	67%	23.4%
	8	6	0.9	$1.11 * 10^7$	4.167%	133.3%	68%	65%	60.2%	30.2%
Majority Voting	5	4	0.1	$6.71 * 10^6$	2.5%	80%	84.7%	83%	81.6%	8.76%
	5	4	0.2	$6.71 * 10^6$	2.5%	80%	84.7%	84%	81.6%	8.76%
	6	4	0.3	$5.59 * 10^6$	2.083%	66.67%	84.1%	82%	80.9%	9.5%
	7	4	0.4	$4.79 * 10^6$	1.786%	57.14%	81.5%	80.9%	79.8%	10.6%
	7	6	0.5	$1.27 * 10^7$	4.762%	152.4%	81.2%	79.3%	78.4%	12%
	7	4	0.6	$4.79 * 10^6$	1.786%	57.14%	81.7%	77.5%	76.1%	14.3%
	7	4	0.7	$4.79 * 10^6$	1.786%	57.14%	78.7%	74.5%	72.7%	17.7%
	8	8	0.8	$3.35 * 10^7$	12.5%	400%	72.4%	68.9%	66.1%	24.3%
	8	8	0.9	$3.35 * 10^7$	12.5%	400%	68.4%	62%	54.5%	35.9%
Transposed & Majority Voting	4	5	0.1	$1.34 * 10^7$	5.0%	160%	84.3%	83.4%	82.9%	7.51%
	4	4	0.2	$8.38 * 10^6$	3.125%	100%	84.8%	82.9%	82.3%	8.13%
	6	6	0.3	$1.49 * 10^7$	5.556%	177.8%	82.3%	81.3%	79.7%	10.7%
	6	7	0.4	$2.55 * 10^7$	9.524%	304.8%	83.2%	81.2%	79.4%	11%
	7	6	0.5	$1.27 * 10^7$	4.762%	152.4%	79.5%	78.2%	77.5%	12.9%
	7	5	0.6	$7.66 * 10^6$	2.857%	91.43%	80.2%	77.7%	76.4%	14%
	6	4	0.7	$5.59 * 10^6$	2.083%	66.67%	76.9%	75.9%	71.6%	18.8%
	8	4	0.8	$4.19 * 10^6$	1.562%	50%	77.2%	73.2%	66.5%	23.9%
	7	4	0.9	$4.79 * 10^6$	1.786%	57.14%	70.3%	67.8%	61.9%	28.5%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.8: Top three achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tested binarisation mask threshold with their corresponding $\mathcal{H}$ and $\mathcal{W}$ ($\mathcal{T} = 64$) using the binary PSML with several bit-error reduction strategies, ordered by mask threshold.

Bit-error correction	Bloom filter $\mathcal{W}$	Bloom filter $\mathcal{H}$	SMR MT	Workload ω_{256}	Workload F_B*	Workload F_R†	Recognition Performance $TP_{0.5}$	Recognition Performance $TP_{0.1}$	Recognition Performance TP_0	TP_0 loss to PSML-Baseline
None	8	4	0.1	$4.22 * 10^6$	0.7874%	50.39%	87.8%	86.9%	86.5%	3.92%
	8	6	0.2	$1.12 * 10^7$	2.089%	133.7%	88.7%	87.5%	87.1%	3.29%
	5	5	0.3	$1.07 * 10^7$	2.006%	128.4%	89.5%	88.8%	86.8%	3.6%
	7	4	0.4	$4.82 * 10^6$	0.899%	57.53%	89.8%	89%	87.4%	2.98%
	9	4	0.5	$3.76 * 10^6$	0.7005%	44.84%	89.2%	88.8%	88.8%	1.65%
	8	6	0.6	$1.12 * 10^7$	2.089%	133.7%	89.8%	89.5%	89%	1.42%
	7	7	0.7	$2.19 * 10^7$	4.088%	261.6%	90.1%	89.1%	88.9%	1.49%
	7	5	0.8	$7.70 * 10^6$	1.435%	91.82%	89.2%	88.8%	88.6%	1.81%
	8	4	0.9	$4.22 * 10^6$	0.7874%	50.39%	89.8%	88.7%	87.8%	2.59%
Transposed	8	5	0.1	$6.74 * 10^6$	1.256%	80.39%	88%	87.5%	86.6%	3.76%
	5	5	0.2	$1.07 * 10^7$	2.006%	128.4%	88.5%	87.7%	87.3%	3.13%
	8	12	0.3	$3.57 * 10^8$	66.67%	4267%	88%	87.6%	87.3%	3.06%
	7	4	0.4	$4.82 * 10^6$	0.899%	57.53%	89.2%	87.7%	87.5%	2.9%
	6	6	0.5	$1.49 * 10^7$	2.784%	178.2%	89.4%	88.9%	88.5%	1.88%
	5	5	0.6	$1.07 * 10^7$	2.006%	128.4%	89.8%	88.7%	88%	2.35%
	5	8	0.7	$5.37 * 10^7$	10.01%	640.4%	89.8%	88.8%	87.9%	2.51%
	5	8	0.8	$5.37 * 10^7$	10.01%	640.4%	89.4%	88.3%	87.5%	2.9%
	5	8	0.9	$5.37 * 10^7$	10.01%	640.4%	89%	87.6%	86.9%	3.53%
Majority Voting	8	4	0.1	$4.22 * 10^6$	0.7874%	50.39%	87.7%	86.8%	86.4%	3.99%
	7	7	0.2	$2.19 * 10^7$	4.088%	261.6%	88.5%	87.4%	87%	3.37%
	5	5	0.3	$1.07 * 10^7$	2.006%	128.4%	89.5%	88.8%	86.8%	3.6%
	5	5	0.4	$1.07 * 10^7$	2.006%	128.4%	89.8%	89%	87.4%	2.98%
	5	5	0.5	$1.07 * 10^7$	2.006%	128.4%	89.3%	89%	88.6%	1.81%
	6	5	0.6	$8.98 * 10^6$	1.673%	107.1%	90.1%	89.1%	88.5%	1.88%
	7	4	0.7	$4.82 * 10^6$	0.899%	57.53%	90.2%	89.4%	88.6%	1.81%
	8	5	0.8	$6.74 * 10^6$	1.256%	80.39%	89.8%	89.5%	88.9%	1.49%
	8	5	0.9	$6.74 * 10^6$	1.256%	80.39%	89.3%	88.2%	87.7%	2.74%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.9: Top three achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ using the binary PSML with and without bit-error reduction strategies, ordered by mask threshold.

Bloom filter			SMR	Workload			Recognition Performance			
$\mathcal{T}$	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
16	4	4	0.1	$4.22 * 10^6$	0.7874%	50.39%	80.5%	79.6%	79.5%	10.9%
	4	5	0.2	$6.74 * 10^6$	1.256%	80.39%	81.2%	80.6%	78.6%	11.8%
	5	5	0.3	$5.40 * 10^6$	1.006%	64.39%	81.3%	80.9%	79.7%	10.7%
	6	5	0.4	$4.50 * 10^6$	0.8394%	53.72%	83.1%	83%	81.1%	9.3%
	5	5	0.5	$5.40 * 10^6$	1.006%	64.39%	82.4%	81.8%	81.6%	8.84%
	4	5	0.6	$6.74 * 10^6$	1.256%	80.39%	84.1%	83.8%	83.3%	7.12%
	4	6	0.7	$1.12 * 10^7$	2.089%	133.7%	83.3%	82.5%	81.9%	8.53%
	5	5	0.8	$5.40 * 10^6$	1.006%	64.39%	82.5%	82.3%	80.9%	9.5%
	5	4	0.9	$3.38 * 10^6$	0.6311%	40.39%	81.8%	81.3%	80.8%	9.6%
32	4	4	0.1	$6.32 * 10^6$	1.178%	75.39%	86.8%	86.2%	85.9%	4.54%
	4	9	0.2	$8.95 * 10^7$	16.67%	1067%	87.2%	86.8%	85.9%	4.46%
	4	8	0.3	$5.03 * 10^7$	9.381%	600.4%	88.2%	87.1%	86.1%	4.31%
	6	5	0.4	$6.74 * 10^6$	1.256%	80.39%	89.1%	88.2%	86.9%	3.53%
	7	4	0.5	$3.62 * 10^6$	0.6757%	43.25%	89.2%	88.4%	87.9%	2.51%
	5	4	0.6	$5.06 * 10^6$	0.9436%	60.39%	89.1%	88.5%	87.8%	2.59%
	7	4	0.7	$3.62 * 10^6$	0.6757%	43.25%	89%	88.1%	88%	2.35%
	8	5	0.8	$5.06 * 10^6$	0.9436%	60.39%	88.4%	88%	87.6%	2.82%
	8	4	0.9	$3.17 * 10^6$	0.592%	37.89%	89.2%	87.9%	87%	3.45%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML ω_{256}.

Table A.10: Achieved PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML comparison using $\mathcal{T} = 32$ and $\mathcal{T} = 16$ trees.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	5	0.1	$3.59 * 10^6$	0.6702%	42.89%	72.1%	72.0%	71.7%	18.7%
2	4	5	0.2	$3.80 * 10^6$	0.7092%	45.39%	79.1%	78.7%	78.0%	12.4%
3	4	4	0.6	$2.52 * 10^6$	0.4700%	30.08%	82.9%	82.1%	81.2%	9.2%
4	4	5	0.6	$4.22 * 10^6$	0.7874%	50.39%	84.7%	84.3%	83.6%	6.81%
5	4	5	0.5	$4.43 * 10^6$	0.8264%	52.89%	85.5%	84.9%	84.7%	5.71%
6	4	5	0.6	$4.64 * 10^6$	0.8655%	55.39%	86.8%	85.9%	85.3%	5.09%
7	4	5	0.6	$4.85 * 10^6$	0.9045%	57.89%	87.4%	86.6%	86.1%	4.31%
8	6	4	0.8	$2.12 * 10^6$	0.3967%	25.39%	87.6%	87.3%	87.1%	3.29%
9	6	4	0.8	$2.21 * 10^6$	0.4130%	26.43%	88.0%	87.7%	87.5%	2.9%
10	6	4	0.8	$2.30 * 10^6$	0.4293%	27.47%	88.3%	88.0%	87.7%	2.67%
11	6	4	0.8	$2.39 * 10^6$	0.4456%	28.52%	88.8%	88.4%	88.2%	2.2%
12	4	5	0.6	$5.90 * 10^6$	1.1000%	70.39%	89.0%	88.4%	87.7%	2.74%
13	4	5	0.6	$6.11 * 10^6$	1.1390%	72.89%	89.1%	88.6%	87.8%	2.59%
14	5	5	0.6	$5.06 * 10^6$	0.9436%	60.39%	89.1%	88.6%	87.9%	2.51%
15	7	5	0.7	$3.74 * 10^6$	0.6981%	44.68%	88.8%	88.2%	88.1%	2.28%
16	7	5	0.7	$3.86 * 10^6$	0.7204%	46.10%	89.1%	88.4%	88.4%	2.04%
17	7	4	0.7	$2.50 * 10^6$	0.4665%	29.85%	89.4%	88.6%	88.5%	1.88%
18	7	4	0.7	$2.57 * 10^6$	0.4804%	30.75%	89.5%	88.7%	88.6%	1.81%
19	7	4	0.7	$2.65 * 10^6$	0.4944%	31.64%	89.6%	88.8%	88.8%	1.65%
20	7	4	0.7	$2.72 * 10^6$	0.5083%	32.53%	89.8%	89.0%	88.9%	1.49%
21	4	7	0.6	$2.22 * 10^7$	4.1360%	264.7%	89.8%	89.4%	88.6%	1.81%
22	5	7	0.6	$1.82 * 10^7$	3.3990%	217.5%	89.9%	89.4%	88.6%	1.81%
23	4	7	0.6	$2.34 * 10^7$	4.3590%	279.0%	89.9%	89.5%	88.7%	1.73%
24	5	7	0.6	$1.92 * 10^7$	3.5780%	229.0%	90.0%	89.5%	88.7%	1.73%
25	5	7	0.6	$1.96 * 10^7$	3.6670%	234.7%	90.1%	89.5%	88.8%	1.65%
26	5	7	0.6	$2.01 * 10^7$	3.7560%	240.4%	90.1%	89.5%	88.8%	1.65%
27	5	7	0.6	$2.06 * 10^7$	3.8450%	246.1%	90.1%	89.5%	88.8%	1.65%
28	5	7	0.6	$2.11 * 10^7$	3.9350%	251.8%	90.1%	89.5%	88.8%	1.65%
29	5	7	0.6	$2.16 * 10^7$	4.0240%	257.5%	90.1%	89.5%	88.8%	1.65%
30	5	7	0.6	$2.20 * 10^7$	4.1130%	263.2%	90.1%	89.5%	88.8%	1.65%
31	5	7	0.6	$2.25 * 10^7$	4.2030%	269.0%	90.2%	89.6%	88.8%	1.57%
32	5	7	0.6	$2.30 * 10^7$	4.2920%	274.7%	90.2%	89.6%	88.8%	1.57%
33	5	7	0.6	$2.35 * 10^7$	4.3810%	280.4%	90.2%	89.6%	88.8%	1.57%
34	5	7	0.6	$2.40 * 10^7$	4.4700%	286.1%	90.2%	89.6%	88.8%	1.57%
35	7	7	0.7	$1.74 * 10^7$	3.2590%	208.6%	90.1%	89.1%	88.9%	1.49%
36	7	7	0.7	$1.78 * 10^7$	3.3220%	212.6%	90.1%	89.1%	88.9%	1.49%
37	7	7	0.7	$1.81 * 10^7$	3.3860%	216.7%	90.1%	89.1%	88.9%	1.49%
38	7	7	0.7	$1.85 * 10^7$	3.4500%	220.8%	90.1%	89.1%	88.9%	1.49%
39	7	7	0.7	$1.88 * 10^7$	3.5140%	224.9%	90.1%	89.1%	88.9%	1.49%
40	8	6	0.6	$9.81 * 10^6$	1.8290%	117.1%	89.7%	89.3%	88.9%	1.49%
41	8	6	0.6	$9.99 * 10^6$	1.8620%	119.1%	89.7%	89.3%	88.9%	1.49%
42	8	6	0.6	$1.01 * 10^7$	1.8940%	121.2%	89.7%	89.3%	88.9%	1.49%
43	8	6	0.6	$1.03 * 10^7$	1.9270%	123.3%	89.8%	89.4%	88.9%	1.49%
44	8	6	0.6	$1.05 * 10^7$	1.9590%	125.4%	89.8%	89.5%	89.0%	1.42%
45	8	6	0.6	$1.06 * 10^7$	1.9920%	127.5%	89.8%	89.5%	89.0%	1.42%
46	8	6	0.6	$1.08 * 10^7$	2.0240%	129.6%	89.8%	89.5%	89.0%	1.42%
47	8	6	0.6	$1.10 * 10^7$	2.0570%	131.6%	89.8%	89.5%	89.0%	1.42%
48	8	6	0.6	$1.12 * 10^7$	2.0890%	133.7%	89.8%	89.5%	89.0%	1.42%
64	8	6	0.6	$1.12 * 10^7$	2.0890%	133.7%	89.8%	89.5%	89.0%	1.42%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML ω_{256}.

Table A.11: Best achieved PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	5	0.4	$2.02 * 10^6$	0.3772%	24.14%	52.3%	52.2%	51.9%	38.5%
2	5	5	0.8	$1.87 * 10^6$	0.3499%	22.39%	62.7%	62.5%	62.5%	27.9%
3	5	5	0.8	$2.12 * 10^6$	0.3967%	25.39%	70.0%	69.8%	69.8%	20.6%
4	5	5	0.8	$2.38 * 10^6$	0.4436%	28.39%	74.3%	74.0%	73.9%	16.5%
5	4	5	0.7	$3.28 * 10^6$	0.6116%	39.14%	77.0%	77.0%	76.4%	14.0%
6	4	4	0.6	$2.26 * 10^6$	0.4211%	26.95%	79.1%	78.5%	77.5%	12.9%
7	6	5	0.6	$2.61 * 10^6$	0.4879%	31.22%	80.6%	80.2%	79.6%	10.8%
8	4	5	0.6	$4.22 * 10^6$	0.7874%	50.39%	83.3%	82.8%	82.0%	8.45%
9	4	5	0.6	$4.54 * 10^6$	0.8459%	54.14%	84.6%	84.1%	83.2%	7.20%
10	4	5	0.6	$4.85 * 10^6$	0.9045%	57.89%	85.7%	85.0%	84.1%	6.34%
11	4	5	0.6	$5.17 * 10^6$	0.9631%	61.64%	86.4%	85.6%	84.7%	5.71%
12	4	5	0.6	$5.48 * 10^6$	1.0220%	65.39%	86.6%	85.9%	85.0%	5.40%
13	4	5	0.6	$5.79 * 10^6$	1.0800%	69.14%	87.4%	86.7%	85.8%	4.62%
14	4	5	0.6	$6.11 * 10^6$	1.1390%	72.89%	88.0%	87.3%	86.4%	3.99%
15	4	5	0.6	$6.42 * 10^6$	1.1980%	76.64%	88.4%	87.7%	86.7%	3.68%
16	4	5	0.6	$6.74 * 10^6$	1.2560%	80.39%	88.6%	88.0%	87.0%	3.45%
17	4	5	0.6	$7.05 * 10^6$	1.3150%	84.14%	88.7%	88.0%	87.0%	3.37%
18	5	7	0.6	$1.68 * 10^7$	3.1310%	200.4%	87.9%	87.5%	86.8%	3.60%
19	4	7	0.6	$2.19 * 10^7$	4.0800%	261.1%	88.5%	87.8%	87.0%	3.37%
20	4	7	0.6	$2.28 * 10^7$	4.2470%	271.8%	88.7%	88.0%	87.2%	3.21%
21	4	7	0.6	$2.37 * 10^7$	4.4150%	282.5%	88.6%	88.2%	87.4%	2.98%
22	8	5	0.8	$4.33 * 10^6$	0.8069%	51.64%	88.3%	87.7%	87.5%	2.90%
23	5	7	0.6	$2.04 * 10^7$	3.8010%	243.2%	88.7%	88.4%	87.6%	2.82%
24	7	4	0.7	$3.32 * 10^6$	0.6199%	39.68%	88.8%	87.9%	87.8%	2.59%
25	7	4	0.7	$3.44 * 10^6$	0.6409%	41.02%	88.8%	87.9%	87.8%	2.59%
26	7	4	0.7	$3.55 * 10^6$	0.6618%	42.35%	88.8%	88.0%	87.9%	2.51%
32	7	4	0.7	$3.62 * 10^6$	0.6757%	43.25%	89.0%	88.1%	88.0%	2.35%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML ω_{256}.

Table A.12: Best achieved PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final binary PSML comparison and tree pre-selection of $t = 1, \ldots, 28$ trees at $\mathcal{T} = 32$.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	$\digamma_B{}^*$	$\digamma_R{}^\dagger$	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	4	5	0.1	$3.59 * 10^6$	1.340%	42.89%	72.7%	72.3%	72.3%	18.1%
2	4	5	0.2	$3.80 * 10^6$	1.418%	45.39%	80.2%	79.9%	79.6%	10.8%
3	4	4	0.6	$2.52 * 10^6$	0.940%	30.08%	83.4%	82.7%	82.7%	7.67%
4	4	5	0.4	$4.22 * 10^6$	1.575%	50.39%	85.2%	84.8%	84.5%	5.95%
5	4	5	0.5	$4.43 * 10^6$	1.653%	52.89%	86.9%	85.9%	85.8%	4.62%
6	4	5	0.6	$4.64 * 10^6$	1.731%	55.39%	87.2%	86.5%	86.5%	3.92%
7	5	5	0.4	$3.89 * 10^6$	1.450%	46.39%	88.4%	87.5%	87.4%	2.98%
8	4	4	0.6	$3.17 * 10^6$	1.184%	37.89%	89.0%	88.2%	88.2%	2.2%
9	4	4	0.6	$3.30 * 10^6$	1.233%	39.45%	89.4%	88.6%	88.6%	1.81%
10	4	4	0.6	$3.44 * 10^6$	1.282%	41.02%	89.8%	89.1%	88.8%	1.57%
11	5	4	0.6	$2.86 * 10^6$	1.067%	34.14%	89.8%	88.9%	88.9%	1.49%
12	4	4	0.6	$3.70 * 10^6$	1.379%	44.14%	90.2%	89.4%	89.1%	1.26%
13	4	5	0.2	$6.11 * 10^6$	2.278%	72.89%	90.1%	89.5%	89.3%	1.1%
14	4	5	0.2	$6.32 * 10^6$	2.356%	75.39%	90.2%	89.7%	89.5%	0.95%
15	4	4	0.6	$4.09 * 10^6$	1.526%	48.83%	90.6%	89.7%	89.5%	0.95%
16	4	4	0.6	$4.22 * 10^6$	1.575%	50.39%	90.6%	89.7%	89.5%	0.95%
17	5	4	0.6	$3.49 * 10^6$	1.301%	41.64%	90.5%	89.6%	89.6%	0.79%
18	6	4	0.6	$3.00 * 10^6$	1.119%	35.81%	90.6%	89.7%	89.7%	0.71%
19	7	4	0.4	$2.65 * 10^6$	0.989%	31.64%	90.6%	89.8%	89.8%	0.63%
20	7	4	0.4	$2.72 * 10^6$	1.017%	32.53%	90.7%	89.9%	89.8%	0.56%
21	7	4	0.4	$2.80 * 10^6$	1.045%	33.43%	90.8%	90.0%	89.9%	0.48%
22	7	4	0.4	$2.87 * 10^6$	1.072%	34.32%	90.9%	90.1%	90.0%	0.4%
23	7	4	0.4	$2.95 * 10^6$	1.100%	35.21%	90.9%	90.2%	90.1%	0.32%
24	7	4	0.4	$3.02 * 10^6$	1.128%	36.10%	91.1%	90.3%	90.2%	0.17%
25	6	6	0.1	$9.58 * 10^6$	3.571%	114.3%	91.1%	90.5%	90.4%	0.0%
26	6	6	0.1	$9.81 * 10^6$	3.658%	117.1%	91.1%	90.5%	90.4%	0.0%
27	6	6	0.1	$1.00 * 10^7$	3.745%	119.8%	90.6%	90.5%	90.4%	0.0%
28	6	6	0.1	$1.02 * 10^7$	3.832%	122.6%	90.5%	90.5%	90.3%	0.1%
29	6	6	0.1	$1.05 * 10^7$	3.918%	125.4%	90.5%	90.5%	90.3%	0.1%
30	6	6	0.1	$1.07 * 10^7$	4.005%	128.2%	90.5%	90.5%	90.3%	0.1%
31	6	6	0.1	$1.09 * 10^7$	4.092%	130.9%	90.5%	90.5%	90.3%	0.1%
32	6	6	0.1	$1.12 * 10^7$	4.179%	133.7%	90.5%	90.5%	90.3%	0.1%
33	6	6	0.1	$1.14 * 10^7$	4.266%	136.5%	90.5%	90.5%	90.3%	0.1%
34	6	6	0.1	$1.16 * 10^7$	4.352%	139.3%	90.5%	90.5%	90.3%	0.1%
35	4	6	0.1	$1.78 * 10^7$	6.653%	212.9%	90.9%	90.7%	90.5%	+0.1%
36	4	6	0.1	$1.82 * 10^7$	6.783%	217.1%	90.9%	90.7%	90.5%	+0.1%
37	4	6	0.1	$1.85 * 10^7$	6.913%	221.2%	90.9%	90.7%	90.5%	+0.1%
38	4	6	0.1	$1.89 * 10^7$	7.043%	225.4%	91.0%	90.8%	90.5%	+0.1%
39	4	6	0.1	$1.92 * 10^7$	7.174%	229.6%	91.0%	90.8%	90.5%	+0.1%
40	4	6	0.1	$1.96 * 10^7$	7.304%	233.7%	91.0%	90.8%	90.5%	+0.1%
41	4	6	0.1	$1.99 * 10^7$	7.434%	237.9%	91.0%	90.8%	90.5%	+0.1%
42	4	6	0.1	$2.03 * 10^7$	7.564%	242.1%	91.0%	90.8%	90.5%	+0.1%
43	4	6	0.1	$2.06 * 10^7$	7.694%	246.2%	91.0%	90.8%	90.5%	+0.1%
44	4	6	0.1	$2.10 * 10^7$	7.825%	250.4%	91.0%	90.8%	90.5%	+0.1%
45	4	6	0.1	$2.13 * 10^7$	7.955%	254.6%	91.0%	90.8%	90.5%	+0.1%
46	4	6	0.1	$2.17 * 10^7$	8.085%	258.7%	91.0%	90.8%	90.5%	+0.1%
47	4	6	0.1	$2.20 * 10^7$	8.215%	262.9%	91.0%	90.8%	90.5%	+0.1%
48	4	6	0.1	$2.24 * 10^7$	8.346%	267.1%	91.0%	90.8%	90.5%	+0.1%
64	4	6	0.1	$2.24 * 10^7$	8.346%	267.1%	91.0%	90.8%	90.5%	+0.1%

* $\digamma$ compared to the PSML ω_{256}.

† $\digamma$ compared to the naïve binary PSML ω_{256}.

Table A.13: Best achieved binary PSML Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final real-valued PSML comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.47	0.64	0.65	$3.48 * 10^5$	0.1297%	26.56%	49.61%	49.3%	48.8%	48.6%	41.8%
2	0.49	0.63	0.65	$3.69 * 10^5$	0.1373%	28.12%	37.27%	61.0%	60.5%	60.4%	30.0%
3	0.49	0.63	0.65	$3.89 * 10^5$	0.1450%	29.69%	29.14%	68.9%	68.3%	68.1%	22.3%
4	0.46	0.60	0.65	$4.10 * 10^5$	0.1526%	31.25%	25.55%	72.4%	72.0%	72.0%	18.4%
5	0.46	0.60	0.65	$4.30 * 10^5$	0.1602%	32.81%	22.27%	75.5%	75.3%	74.6%	15.8%
6	0.46	0.63	0.65	$4.51 * 10^5$	0.1678%	34.38%	18.98%	78.0%	77.7%	77.3%	13.1%
7	0.46	0.60	0.65	$4.71 * 10^5$	0.1755%	35.94%	16.72%	80.6%	80.0%	79.7%	10.7%
8	0.46	0.60	0.65	$4.92 * 10^5$	0.1831%	37.50%	14.84%	82.1%	81.5%	81.2%	9.20%
9	0.46	0.60	0.65	$5.12 * 10^5$	0.1907%	39.06%	13.83%	83.1%	82.3%	82.0%	8.37%
10	0.46	0.60	0.65	$5.32 * 10^5$	0.1984%	40.62%	12.81%	83.9%	83.4%	83.0%	7.35%
11	0.46	0.60	0.65	$5.53 * 10^5$	0.2060%	42.19%	11.80%	84.5%	84.1%	83.8%	6.57%
12	0.46	0.60	0.65	$5.73 * 10^5$	0.2136%	43.75%	10.94%	85.3%	84.9%	84.6%	5.79%
13	0.46	0.60	0.65	$5.94 * 10^5$	0.2213%	45.31%	10.00%	86.0%	85.5%	85.2%	5.24%
14	0.46	0.60	0.65	$6.14 * 10^5$	0.2289%	46.88%	9.375%	86.4%	85.9%	85.5%	4.85%
15	0.46	0.60	0.65	$6.35 * 10^5$	0.2365%	48.44%	8.750%	87.0%	86.5%	86.1%	4.31%
16	0.46	0.60	0.65	$6.55 * 10^5$	0.2441%	50.00%	8.203%	87.4%	86.9%	86.5%	3.92%
17	0.46	0.63	0.65	$6.76 * 10^5$	0.2518%	51.56%	7.656%	87.8%	87.5%	87.0%	3.45%
18	0.46	0.63	0.65	$6.96 * 10^5$	0.2594%	53.12%	7.109%	88.3%	88.0%	87.4%	2.98%
19	0.46	0.63	0.65	$7.17 * 10^5$	0.2670%	54.69%	6.719%	88.6%	88.2%	87.7%	2.74%
20	0.46	0.63	0.65	$7.37 * 10^5$	0.2747%	56.25%	6.406%	89.0%	88.3%	88.0%	2.35%
21	0.46	0.63	0.65	$7.58 * 10^5$	0.2823%	57.81%	6.094%	89.1%	88.4%	88.1%	2.28%
22	0.49	0.68	0.65	$7.78 * 10^5$	0.2899%	59.38%	6.016%	88.8%	88.4%	88.4%	2.04%
23	0.49	0.68	0.65	$7.99 * 10^5$	0.2975%	60.94%	5.859%	89.1%	88.6%	88.5%	1.88%
24	0.46	0.62	0.70	$8.19 * 10^5$	0.3052%	62.50%	5.547%	89.5%	88.8%	88.8%	1.65%
25	0.46	0.62	0.70	$8.40 * 10^5$	0.3128%	64.06%	5.469%	89.5%	88.8%	88.8%	1.57%
26	0.46	0.62	0.65	$8.60 * 10^5$	0.3204%	65.62%	5.078%	89.5%	89.1%	88.8%	1.57%
27	0.46	0.62	0.70	$8.81 * 10^5$	0.3281%	67.19%	4.922%	89.8%	89.1%	89.1%	1.26%
28	0.46	0.62	0.70	$9.01 * 10^5$	0.3357%	68.75%	4.766%	90.0%	89.2%	89.2%	1.18%
29	0.46	0.62	0.70	$9.22 * 10^5$	0.3433%	70.31%	4.766%	90.0%	89.2%	89.2%	1.18%
30	0.46	0.62	0.70	$9.42 * 10^5$	0.3510%	71.88%	4.688%	90.1%	89.3%	89.3%	1.10%
31	0.46	0.62	0.70	$9.63 * 10^5$	0.3586%	73.44%	4.688%	90.1%	89.3%	89.3%	1.10%
32	0.46	0.62	0.70	$9.83 * 10^5$	0.3662%	75.00%	4.531%	90.2%	89.4%	89.4%	1.03%
33	0.46	0.62	0.65	$1.00 * 10^6$	0.3738%	76.56%	4.062%	89.8%	89.7%	89.5%	0.95%
34	0.46	0.62	0.65	$1.02 * 10^6$	0.3815%	78.12%	4.062%	89.8%	89.7%	89.5%	0.95%
35	0.46	0.62	0.70	$1.04 * 10^6$	0.3891%	79.69%	4.375%	89.9%	89.5%	89.5%	0.87%
36	0.46	0.62	0.70	$1.06 * 10^6$	0.3967%	81.25%	4.375%	89.9%	89.5%	89.5%	0.87%
37	0.46	0.62	0.70	$1.09 * 10^6$	0.4044%	82.81%	4.375%	89.8%	89.5%	89.5%	0.87%
38	0.46	0.62	0.70	$1.11 * 10^6$	0.4102%	84.38%	4.141%	89.9%	89.7%	89.7%	0.71%
39	0.46	0.62	0.70	$1.13 * 10^6$	0.4196%	85.94%	4.141%	89.9%	89.7%	89.7%	0.71%
40	0.46	0.62	0.70	$1.15 * 10^6$	0.4272%	87.50%	3.984%	90.0%	89.8%	89.8%	0.63%
41	0.46	0.62	0.70	$1.17 * 10^6$	0.4349%	89.06%	3.984%	90.0%	89.8%	89.8%	0.63%
42	0.46	0.62	0.70	$1.19 * 10^6$	0.4425%	90.62%	3.984%	90.0%	89.8%	89.8%	0.63%
43	0.46	0.62	0.70	$1.21 * 10^6$	0.4501%	92.19%	3.906%	90.0%	89.8%	89.8%	0.63%
44	0.46	0.62	0.70	$1.23 * 10^6$	0.4578%	93.75%	3.906%	90.0%	89.8%	89.8%	0.63%
45	0.46	0.62	0.70	$1.25 * 10^6$	0.4654%	95.31%	3.906%	90.0%	89.8%	89.8%	0.63%
46	0.46	0.62	0.70	$1.27 * 10^6$	0.4730%	96.88%	3.828%	90.0%	89.8%	89.8%	0.63%
47	0.46	0.62	0.70	$1.29 * 10^6$	0.4807%	98.44%	3.750%	90.1%	89.8%	89.8%	0.56%
48	0.46	0.62	0.70	$1.31 * 10^6$	0.4883%	100.0%	3.750%	90.1%	89.8%	89.8%	0.56%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.14: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-C templates at $\mathcal{T} = 64$.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.64	0.65	$3.48 * 10^5$	0.1297%	26.56%	28.75%	70.2%	69.5%	69.3%	21.1%
2	0.49	0.64	0.65	$3.69 * 10^5$	0.1373%	28.12%	18.44%	79.0%	78.4%	78.0%	12.4%
3	0.49	0.67	0.70	$3.89 * 10^5$	0.1450%	29.69%	14.22%	83.0%	82.7%	81.9%	8.53%
4	0.46	0.62	0.65	$4.10 * 10^5$	0.1526%	31.25%	11.88%	84.5%	84.2%	83.9%	6.49%
5	0.48	0.66	0.70	$4.30 * 10^5$	0.1602%	32.81%	10.00%	86.6%	85.8%	85.5%	4.93%
6	0.48	0.66	0.70	$4.51 * 10^5$	0.1678%	34.38%	8.516%	87.2%	86.5%	86.3%	4.07%
7	0.49	0.64	0.70	$4.71 * 10^5$	0.1755%	35.94%	7.734%	88.0%	87.3%	87.0%	3.37%
8	0.49	0.68	0.70	$4.92 * 10^5$	0.1831%	37.50%	6.875%	88.8%	87.8%	87.4%	2.98%
9	0.46	0.62	0.65	$5.12 * 10^5$	0.1907%	39.06%	7.266%	88.1%	88.0%	87.8%	2.59%
10	0.49	0.68	0.70	$5.32 * 10^5$	0.1984%	40.62%	5.859%	89.1%	88.6%	88.1%	2.28%
11	0.49	0.68	0.70	$5.53 * 10^5$	0.2060%	42.19%	5.391%	89.5%	88.9%	88.4%	2.04%
12	0.49	0.68	0.70	$5.73 * 10^5$	0.2136%	43.75%	5.312%	89.5%	88.9%	88.4%	2.04%
13	0.45	0.78	0.65	$5.94 * 10^5$	0.2213%	45.31%	5.703%	89.5%	88.8%	88.7%	1.73%
14	0.45	0.78	0.65	$6.14 * 10^5$	0.2289%	46.88%	5.469%	89.6%	88.8%	88.8%	1.65%
15	0.47	0.61	0.65	$6.35 * 10^5$	0.2365%	48.44%	5.312%	89.2%	89.1%	88.9%	1.49%
16	0.47	0.61	0.65	$6.55 * 10^5$	0.2441%	50.00%	5.078%	89.3%	89.2%	89.0%	1.42%
17	0.46	0.62	0.65	$6.76 * 10^5$	0.2518%	51.56%	4.453%	89.4%	89.4%	89.1%	1.26%
18	0.46	0.62	0.65	$6.96 * 10^5$	0.2594%	53.12%	4.375%	89.5%	89.5%	89.3%	1.10%
19	0.46	0.62	0.65	$7.17 * 10^5$	0.2670%	54.69%	4.297%	89.6%	89.6%	89.4%	1.03%
20	0.46	0.62	0.65	$7.37 * 10^5$	0.2747%	56.25%	4.375%	89.6%	89.6%	89.4%	1.03%
21	0.46	0.62	0.65	$7.58 * 10^5$	0.2823%	57.81%	4.219%	89.8%	89.8%	89.5%	0.87%
22	0.46	0.62	0.65	$7.78 * 10^5$	0.2899%	59.38%	4.141%	89.8%	89.8%	89.6%	0.79%
23	0.46	0.62	0.65	$7.99 * 10^5$	0.2975%	60.94%	3.906%	89.9%	89.9%	89.7%	0.71%
24	0.46	0.62	0.65	$8.19 * 10^5$	0.3052%	62.50%	3.906%	89.9%	89.9%	89.7%	0.71%
25	0.46	0.62	0.65	$8.40 * 10^5$	0.3128%	64.06%	3.906%	89.9%	89.9%	89.7%	0.71%
26	0.46	0.62	0.65	$8.60 * 10^5$	0.3204%	65.62%	3.828%	89.9%	89.9%	89.7%	0.71%
27	0.46	0.62	0.65	$8.81 * 10^5$	0.3281%	67.19%	3.828%	89.9%	89.9%	89.7%	0.71%
28	0.46	0.62	0.65	$9.01 * 10^5$	0.3357%	68.75%	3.750%	90.0%	90.0%	89.8%	0.63%
29	0.46	0.62	0.65	$9.22 * 10^5$	0.3433%	70.31%	3.750%	90.0%	90.0%	89.8%	0.63%
30	0.46	0.62	0.65	$9.42 * 10^5$	0.3510%	71.88%	3.750%	90.0%	90.0%	89.8%	0.63%
31	0.46	0.62	0.65	$9.63 * 10^5$	0.3586%	73.44%	3.672%	90.0%	90.0%	89.8%	0.63%
32	0.46	0.62	0.65	$9.83 * 10^5$	0.3662%	75.00%	3.672%	90.0%	90.0%	89.8%	0.63%
33	0.46	0.62	0.65	$1.00 * 10^6$	0.3738%	76.56%	3.672%	90.0%	90.0%	89.8%	0.63%
34	0.46	0.62	0.65	$1.02 * 10^6$	0.3815%	78.12%	3.672%	90.0%	90.0%	89.8%	0.63%
35	0.46	0.62	0.65	$1.04 * 10^6$	0.3891%	79.69%	3.672%	90.0%	90.0%	89.8%	0.63%
36	0.46	0.62	0.65	$1.06 * 10^6$	0.3967%	81.25%	3.594%	90.0%	90.0%	89.8%	0.63%
37	0.46	0.62	0.65	$1.09 * 10^6$	0.4044%	82.81%	3.594%	90.0%	90.0%	89.8%	0.63%
38	0.46	0.62	0.65	$1.11 * 10^6$	0.4120%	84.38%	3.594%	90.0%	90.0%	89.8%	0.63%
39	0.46	0.62	0.65	$1.13 * 10^6$	0.4196%	85.94%	3.594%	90.0%	90.0%	89.8%	0.63%
40	0.46	0.62	0.65	$1.15 * 10^6$	0.4272%	87.50%	3.594%	90.0%	90.0%	89.8%	0.63%
41	0.46	0.62	0.65	$1.17 * 10^6$	0.4349%	89.06%	3.594%	90.0%	90.0%	89.8%	0.63%
42	0.46	0.62	0.65	$1.19 * 10^6$	0.4425%	90.62%	3.594%	90.0%	90.0%	89.8%	0.63%
43	0.46	0.62	0.65	$1.21 * 10^6$	0.4501%	92.19%	3.672%	90.0%	90.0%	89.8%	0.63%
44	0.46	0.62	0.65	$1.23 * 10^6$	0.4578%	93.75%	3.672%	90.0%	90.0%	89.8%	0.63%
45	0.46	0.62	0.65	$1.25 * 10^6$	0.4654%	95.31%	3.672%	90.0%	90.0%	89.8%	0.63%
46	0.46	0.62	0.65	$1.27 * 10^6$	0.4730%	96.88%	3.672%	90.0%	90.0%	89.8%	0.63%
47	0.46	0.62	0.65	$1.29 * 10^6$	0.4807%	98.44%	3.672%	90.0%	90.0%	89.8%	0.63%
48	0.46	0.62	0.65	$1.31 * 10^6$	0.4883%	100.0%	3.672%	90.0%	90.0%	89.8%	0.63%

* $\mathcal{F}$ compared to the PSML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary PSML-CPCA ω_{256}.

Table A.15: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.60	0.70	$1.95 * 10^5$	0.0725%	14.84%	55.23%	43.9%	43.8%	43.7%	46.7%
2	0.49	0.60	0.75	$2.25 * 10^5$	0.0839%	17.19%	41.25%	57.4%	57.3%	57.0%	33.4%
3	0.49	0.60	0.75	$2.56 * 10^5$	0.0954%	19.53%	33.20%	65.2%	64.5%	64.5%	25.9%
4	0.46	0.63	0.70	$2.87 * 10^5$	0.1068%	21.88%	26.88%	71.1%	70.0%	69.8%	20.6%
5	0.49	0.68	0.75	$3.17 * 10^5$	0.1183%	24.22%	22.73%	74.6%	73.8%	73.6%	16.8%
6	0.49	0.68	0.75	$3.48 * 10^5$	0.1297%	26.56%	20.00%	77.0%	75.9%	75.9%	14.5%
7	0.49	0.68	0.70	$3.79 * 10^5$	0.1411%	28.91%	17.34%	79.2%	78.3%	78.1%	12.3%
8	0.47	0.64	0.75	$4.10 * 10^5$	0.1526%	31.25%	15.23%	81.2%	80.9%	79.9%	10.5%
9	0.49	0.68	0.70	$4.40 * 10^5$	0.1640%	33.59%	13.44%	82.7%	81.8%	81.2%	9.20%
10	0.46	0.79	0.65	$4.71 * 10^5$	0.1755%	35.94%	14.22%	82.6%	82.3%	82.3%	8.13%
11	0.48	0.66	0.70	$5.02 * 10^5$	0.1869%	38.28%	11.33%	84.5%	83.7%	83.3%	7.12%
12	0.48	0.66	0.70	$5.32 * 10^5$	0.1984%	40.62%	10.39%	85.3%	84.5%	84.1%	5.34%
13	0.48	0.66	0.70	$5.63 * 10^5$	0.2098%	42.97%	9.688%	85.9%	84.9%	84.7%	5.71%
14	0.48	0.66	0.70	$5.94 * 10^5$	0.2213%	45.31%	9.219%	86.2%	85.9%	85.3%	5.09%
15	0.49	0.67	0.70	$6.25 * 10^5$	0.2327%	47.66%	8.281%	87.3%	87.0%	85.9%	4.46%
16	0.46	0.62	0.65	$6.55 * 10^5$	0.2441%	50.00%	8.359%	86.7%	86.6%	86.5%	3.92%
17	0.46	0.62	0.65	$6.86 * 10^5$	0.2556%	52.34%	7.891%	87.1%	87.1%	87.0%	3.45%
18	0.46	0.62	0.65	$7.17 * 10^5$	0.2670%	54.69%	7.422%	87.4%	87.4%	87.3%	3.13%
19	0.46	0.62	0.65	$7.48 * 10^5$	0.2785%	57.03%	7.344%	87.5%	87.5%	87.3%	3.06%
20	0.46	0.62	0.65	$7.78 * 10^5$	0.2899%	59.38%	7.109%	87.7%	87.7%	87.6%	2.82%
21	0.46	0.62	0.65	$8.09 * 10^5$	0.3014%	61.72%	6.719%	88.1%	88.1%	88.0%	2.43%
22	0.46	0.62	0.65	$8.40 * 10^5$	0.3128%	64.06%	6.484%	88.3%	88.3%	88.1%	2.28%
23	0.46	0.62	0.65	$8.70 * 10^5$	0.3242%	66.41%	6.406%	88.4%	88.4%	88.2%	2.20%
24	0.46	0.62	0.65	$9.01 * 10^5$	0.3357%	68.75%	6.016%	88.7%	88.7%	88.5%	1.88%
25	0.46	0.62	0.65	$9.32 * 10^5$	0.3471%	71.09%	5.938%	88.7%	88.7%	88.5%	1.88%
26	0.46	0.62	0.65	$9.63 * 10^5$	0.3586%	73.44%	5.625%	88.8%	88.8%	88.7%	1.73%
27	0.46	0.62	0.65	$9.93 * 10^5$	0.3700%	75.78%	5.469%	88.8%	88.8%	88.7%	1.73%
28	0.46	0.62	0.65	$1.02 * 10^6$	0.3815%	78.12%	5.469%	88.8%	88.8%	88.7%	1.73%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

Table A.16: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 26$ trees using SMR-CPCA-M templates at $\mathcal{T} = 32$.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.64	0.65	$3.53 * 10^5$	0.1316%	26.95%	28.75%	70.2%	69.5%	69.3%	21.1%
2	0.49	0.64	0.65	$3.74 * 10^5$	0.1392%	28.52%	18.44%	79%	78.4%	78%	12.4%
3	0.49	0.67	0.70	$3.94 * 10^5$	0.1469%	30.08%	14.22%	83%	82.7%	81.9%	8.53%
4	0.46	0.62	0.65	$4.15 * 10^5$	0.1545%	31.64%	11.88%	84.5%	84.2%	83.9%	6.49%
5	0.48	0.66	0.70	$4.35 * 10^5$	0.1621%	33.20%	10.0%	86.6%	85.8%	85.5%	4.93%
6	0.48	0.66	0.70	$4.56 * 10^5$	0.1698%	34.77%	8.516%	87.2%	86.5%	86.3%	4.07%
7	0.49	0.64	0.70	$4.76 * 10^5$	0.1774%	36.33%	7.734%	88%	87.3%	87%	3.37%
8	0.49	0.68	0.70	$4.97 * 10^5$	0.1850%	37.89%	6.875%	88.8%	87.8%	87.4%	2.98%
9	0.46	0.62	0.70	$5.17 * 10^5$	0.1926%	39.45%	7.266%	88.1%	88%	87.8%	2.59%
10	0.49	0.68	0.70	$5.38 * 10^5$	0.2003%	41.02%	5.859%	89.1%	88.6%	88.1%	2.28%
11	0.49	0.68	0.70	$5.58 * 10^5$	0.2079%	42.58%	5.391%	89.5%	88.9%	88.4%	2.04%
12	0.49	0.68	0.70	$5.79 * 10^5$	0.2155%	44.14%	5.312%	89.5%	88.9%	88.4%	2.04%
13	0.48	0.65	0.65	$5.99 * 10^5$	0.2232%	45.70%	5.156%	89.5%	88.8%	88.4%	1.96%
14	0.48	0.65	0.65	$6.20 * 10^5$	0.2308%	47.27%	4.844%	89.8%	89.1%	88.7%	1.73%
15	0.47	0.61	0.65	$6.40 * 10^5$	0.2384%	48.83%	5.312%	89.2%	89.1%	88.9%	1.49%
16	0.47	0.61	0.65	$6.60 * 10^5$	0.2460%	50.39%	5.078%	89.3%	89.2%	89%	1.42%
17	0.46	0.62	0.65	$6.81 * 10^5$	0.2537%	51.95%	4.453%	89.4%	89.4%	89.1%	1.26%
18	0.46	0.62	0.65	$7.01 * 10^5$	0.2613%	53.52%	4.375%	89.5%	89.5%	89.3%	1.1%
19	0.46	0.62	0.65	$7.22 * 10^5$	0.2689%	55.08%	4.297%	89.6%	89.6%	89.4%	1.03%
20	0.46	0.62	0.65	$7.42 * 10^5$	0.2766%	56.64%	4.375%	89.6%	89.6%	89.4%	1.03%
21	0.46	0.62	0.65	$7.63 * 10^5$	0.2842%	58.20%	4.219%	89.8%	89.8%	89.5%	0.87%
22	0.46	0.62	0.65	$7.83 * 10^5$	0.2918%	59.77%	4.141%	89.8%	89.8%	89.6%	0.79%
23	0.46	0.62	0.65	$8.04 * 10^5$	0.2995%	61.33%	3.906%	89.9%	89.9%	89.7%	0.71%
24	0.46	0.62	0.65	$8.24 * 10^5$	0.3071%	62.89%	3.906%	89.9%	89.9%	89.7%	0.71%
25	0.46	0.62	0.65	$8.45 * 10^5$	0.3147%	64.45%	3.906%	89.9%	89.9%	89.7%	0.71%
26	0.46	0.62	0.65	$8.65 * 10^5$	0.3223%	65.02%	3.828%	89.9%	89.9%	89.7%	0.71%
27	0.46	0.62	0.65	$8.86 * 10^5$	0.3300%	67.58%	3.828%	89.9%	89.9%	89.7%	0.71%
28	0.46	0.62	0.65	$9.06 * 10^5$	0.3376%	69.14%	3.75%	90%	90%	89.8%	0.63%
29	0.46	0.62	0.65	$9.27 * 10^5$	0.3452%	70.70%	3.75%	90%	90%	89.8%	0.63%
30	0.46	0.62	0.65	$9.47 * 10^5$	0.3529%	72.27%	3.75%	90%	90%	89.8%	0.63%
31	0.46	0.62	0.65	$9.68 * 10^5$	0.3605%	73.83%	3.672%	90%	90%	89.8%	0.63%
32	0.46	0.62	0.65	$9.88 * 10^5$	0.3681%	75.39%	3.672%	90%	90%	89.8%	0.63%
33	0.46	0.62	0.65	$1.01 * 10^6$	0.3757%	76.95%	3.672%	90%	90%	89.8%	0.63%
34	0.46	0.62	0.65	$1.03 * 10^6$	0.3834%	78.52%	3.672%	90%	90%	89.8%	0.63%
35	0.46	0.62	0.65	$1.05 * 10^6$	0.3910%	80.08%	3.672%	90%	90%	89.8%	0.63%
36	0.46	0.62	0.65	$1.07 * 10^6$	0.3986%	81.64%	3.594%	90%	90%	89.8%	0.63%
37	0.46	0.62	0.65	$1.09 * 10^6$	0.4063%	83.20%	3.594%	90%	90%	89.8%	0.63%
38	0.46	0.62	0.65	$1.11 * 10^6$	0.4139%	84.77%	3.594%	90%	90%	89.8%	0.63%
39	0.46	0.62	0.65	$1.13 * 10^6$	0.4215%	86.33%	3.594%	90%	90%	89.8%	0.63%
40	0.46	0.62	0.65	$1.15 * 10^6$	0.4292%	87.89%	3.594%	90%	90%	89.8%	0.63%
41	0.46	0.62	0.65	$1.17 * 10^6$	0.4368%	89.45%	3.594%	90%	90%	89.8%	0.63%
42	0.46	0.62	0.65	$1.19 * 10^6$	0.4444%	91.02%	3.594%	90%	90%	89.8%	0.63%
43	0.46	0.62	0.65	$1.21 * 10^6$	0.4520%	92.58%	3.672%	90%	90%	89.8%	0.63%
44	0.46	0.62	0.65	$1.23 * 10^6$	0.4597%	94.14%	3.672%	90%	90%	89.8%	0.63%
45	0.46	0.62	0.65	$1.25 * 10^6$	0.4673%	95.70%	3.672%	90%	90%	89.8%	0.63%
46	0.46	0.62	0.65	$1.27 * 10^6$	0.4749%	97.27%	3.672%	90%	90%	89.8%	0.63%
47	0.46	0.62	0.65	$1.30 * 10^6$	0.4826%	98.83%	3.672%	90%	90%	89.8%	0.63%
48	0.46	0.62	0.65	$1.32 * 10^6$	0.4902%	100.4%	3.672%	90%	90%	89.8%	0.63%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary SMR ω_{256}.

Table A.17: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional binary PSML-CPCA comparison.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.49	0.65	0.65	0.7	$3.53 * 10^5$	0.1316%	26.95%	28.91%	70.0%	69.7%	69.3%	21.1%
2	0.49	0.64	0.70	0.5	$3.74 * 10^5$	0.1392%	28.52%	18.67%	79.1%	78.4%	78.4%	12.0%
3	0.49	0.64	0.70	0.6	$3.94 * 10^5$	0.1469%	30.08%	14.22%	82.9%	82.1%	82.0%	8.45%
4	0.46	0.62	0.65	0.6	$4.15 * 10^5$	0.1545%	31.64%	11.95%	84.8%	84.2%	83.9%	6.49%
5	0.48	0.66	0.70	0.5	$4.35 * 10^5$	0.1621%	33.20%	10.16%	86.4%	86.0%	85.5%	4.85%
6	0.48	0.66	0.70	0.5	$4.56 * 10^5$	0.1698%	34.77%	8.750%	87.4%	86.9%	86.5%	3.92%
7	0.49	0.64	0.70	0.7	$4.76 * 10^5$	0.1774%	36.33%	7.734%	88.1%	87.3%	87.1%	3.29%
8	0.49	0.68	0.70	0.6	$4.97 * 10^5$	0.1850%	37.89%	6.875%	88.9%	87.9%	87.5%	2.90%
9	0.49	0.68	0.70	0.6	$5.17 * 10^5$	0.1926%	39.45%	6.484%	88.8%	88.3%	87.8%	2.59%
10	0.49	0.68	0.70	0.7	$5.38 * 10^5$	0.2003%	41.02%	5.938%	89.2%	88.7%	88.2%	2.20%
11	0.49	0.68	0.70	0.7	$5.58 * 10^5$	0.2079%	42.58%	5.391%	89.5%	89.0%	88.4%	1.96%
12	0.49	0.68	0.70	0.5	$5.79 * 10^5$	0.2155%	44.14%	5.469%	89.5%	89.1%	88.4%	1.96%
13	0.47	0.64	0.65	0.7	$5.99 * 10^5$	0.2232%	45.70%	5.078%	89.5%	89.2%	88.5%	1.88%
14	0.46	0.63	0.65	0.5	$6.20 * 10^5$	0.2308%	47.27%	5.703%	89.8%	89.1%	88.8%	1.57%
15	0.46	0.63	0.65	0.5	$6.40 * 10^5$	0.2384%	48.83%	5.312%	90.1%	89.3%	89.0%	1.42%
16	0.46	0.62	0.70	0.6	$6.60 * 10^5$	0.2460%	50.39%	4.844%	89.5%	89.1%	89.1%	1.26%
17	0.46	0.63	0.65	0.5	$6.81 * 10^5$	0.2537%	51.95%	5.000%	90.3%	89.5%	89.2%	1.18%
18	0.46	0.62	0.65	0.9	$7.01 * 10^5$	0.2613%	53.52%	4.375%	89.5%	89.5%	89.3%	1.10%
19	0.46	0.62	0.70	0.6	$7.22 * 10^5$	0.2689%	55.08%	4.453%	89.8%	89.5%	89.5%	0.95%
20	0.46	0.62	0.70	0.6	$7.42 * 10^5$	0.2766%	56.64%	4.453%	89.8%	89.5%	89.5%	0.95%
21	0.46	0.62	0.65	0.8	$7.63 * 10^5$	0.2842%	58.20%	4.219%	89.8%	89.8%	89.5%	0.87%
22	0.46	0.62	0.65	0.7	$7.83 * 10^5$	0.2918%	59.77%	4.219%	89.8%	89.8%	89.6%	0.79%
23	0.46	0.62	0.65	0.7	$8.04 * 10^5$	0.2995%	61.33%	3.984%	89.9%	89.9%	89.7%	0.71%
24	0.46	0.62	0.65	0.8	$8.24 * 10^5$	0.3071%	62.89%	3.906%	89.9%	89.9%	89.7%	0.71%
25	0.46	0.62	0.70	0.6	$8.45 * 10^5$	0.3147%	64.45%	3.984%	90.0%	89.8%	89.8%	0.63%
26	0.46	0.62	0.70	0.6	$8.65 * 10^5$	0.3223%	66.02%	3.984%	90.0%	89.8%	89.8%	0.63%
27	0.46	0.62	0.70	0.5	$8.86 * 10^5$	0.3300%	67.58%	3.984%	90.1%	89.8%	89.8%	0.56%
28	0.46	0.62	0.70	0.6	$9.06 * 10^5$	0.3376%	69.14%	3.828%	90.2%	89.9%	89.9%	0.48%
29	0.46	0.62	0.70	0.6	$9.27 * 10^5$	0.3452%	70.70%	3.828%	90.2%	89.9%	89.9%	0.48%
30	0.46	0.62	0.70	0.5	$9.47 * 10^5$	0.3529%	72.27%	3.906%	90.2%	89.9%	89.9%	0.48%
31	0.46	0.62	0.70	0.5	$9.68 * 10^5$	0.3605%	73.83%	3.906%	90.2%	89.9%	89.9%	0.48%
32	0.46	0.62	0.70	0.5	$9.88 * 10^5$	0.3681%	75.39%	3.906%	90.2%	89.9%	89.9%	0.48%
33	0.46	0.62	0.70	0.5	$1.01 * 10^6$	0.3757%	76.95%	3.828%	90.2%	89.9%	89.9%	0.48%
34	0.46	0.62	0.70	0.5	$1.03 * 10^6$	0.3834%	78.52%	3.828%	90.2%	89.9%	89.9%	0.48%
35	0.46	0.62	0.70	0.6	$1.05 * 10^6$	0.3910%	80.08%	3.750%	90.2%	89.9%	89.9%	0.48%
36	0.46	0.62	0.70	0.5	$1.07 * 10^6$	0.3986%	81.64%	3.750%	90.2%	89.9%	89.9%	0.48%
37	0.46	0.62	0.70	0.6	$1.09 * 10^6$	0.4063%	83.20%	3.672%	90.2%	89.9%	89.9%	0.48%
38	0.46	0.62	0.70	0.6	$1.11 * 10^6$	0.4139%	84.77%	3.672%	90.2%	89.9%	89.9%	0.48%
39	0.46	0.62	0.70	0.5	$1.13 * 10^6$	0.4215%	86.33%	3.750%	90.2%	89.9%	89.9%	0.48%
40	0.46	0.62	0.70	0.5	$1.15 * 10^6$	0.4292%	87.89%	3.750%	90.2%	89.9%	89.9%	0.48%
41	0.46	0.62	0.70	0.5	$1.17 * 10^6$	0.4368%	89.45%	3.750%	90.2%	89.9%	89.9%	0.48%
42	0.46	0.62	0.70	0.5	$1.19 * 10^6$	0.4444%	91.02%	3.750%	90.2%	89.9%	89.9%	0.48%
43	0.46	0.62	0.70	0.6	$1.21 * 10^6$	0.4520%	92.58%	3.672%	90.2%	89.9%	89.9%	0.48%
44	0.46	0.62	0.70	0.6	$1.23 * 10^6$	0.4597%	94.14%	3.672%	90.2%	89.9%	89.9%	0.48%
45	0.46	0.62	0.70	0.6	$1.25 * 10^6$	0.4673%	95.70%	3.672%	90.2%	89.9%	89.9%	0.48%
46	0.46	0.62	0.70	0.5	$1.27 * 10^6$	0.4749%	97.27%	3.750%	90.2%	89.9%	89.9%	0.48%
47	0.46	0.62	0.70	0.6	$1.30 * 10^6$	0.4826%	98.83%	3.672%	90.2%	89.9%	89.9%	0.48%
48	0.46	0.62	0.70	0.6	$1.32 * 10^6$	0.4902%	100.4%	3.672%	90.2%	89.9%	89.9%	0.48%
64‡	0.46	0.62	0.70	0.6	$1.32 * 10^6$	0.4902%	100.4%	3.672%	90.2%	89.9%	89.9%	0.48%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary SMR ω_{256}.

‡ Without tree pre-selection.

Table A.18: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional binary PSML-CPCA comparison and masking out most common set bits.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.49	0.60	0.70	0.9	$2.00 * 10^5$	0.0744%	15.23%	55.23%	43.9%	43.8%	43.7%	46.7%
2	0.49	0.60	0.75	0.8	$2.30 * 10^5$	0.0858%	17.58%	41.25%	57.5%	57.4%	57.0%	33.4%
3	0.49	0.60	0.75	0.5	$2.61 * 10^5$	0.0973%	19.92%	32.97%	65.4%	64.7%	64.7%	25.7%
4	0.46	0.63	0.70	0.6	$2.92 * 10^5$	0.1087%	22.27%	26.80%	71.2%	70.1%	69.9%	20.5%
5	0.48	0.66	0.75	0.5	$3.23 * 10^5$	0.1202%	24.61%	23.28%	74.5%	73.8%	73.6%	16.8%
6	0.48	0.66	0.75	0.5	$3.53 * 10^5$	0.1316%	26.95%	19.45%	77.9%	77.3%	76.7%	13.7%
7	0.48	0.66	0.75	0.5	$3.84 * 10^5$	0.1431%	29.30%	17.34%	79.8%	79.1%	78.6%	11.8%
8	0.48	0.66	0.75	0.5	$4.15 * 10^5$	0.1545%	31.64%	15.55%	81.2%	80.3%	80.0%	10.4%
9	0.49	0.68	0.70	0.9	$4.45 * 10^5$	0.1659%	33.98%	13.44%	82.7%	81.8%	81.2%	9.20%
10	0.48	0.66	0.75	0.5	$4.76 * 10^5$	0.1774%	36.33%	12.81%	83.1%	83.0%	82.7%	7.67%
11	0.48	0.66	0.70	0.6	$5.07 * 10^5$	0.1888%	38.67%	11.56%	84.4%	84.1%	83.5%	6.88%
12	0.48	0.66	0.75	0.5	$5.38 * 10^5$	0.2003%	41.02%	10.70%	85.1%	84.9%	84.6%	5.79%
13	0.48	0.66	0.75	0.5	$5.68 * 10^5$	0.2117%	43.36%	10.00%	85.8%	85.6%	85.3%	5.09%
14	0.48	0.66	0.75	0.5	$5.99 * 10^5$	0.2232%	45.70%	9.531%	86.1%	86.0%	85.7%	4.70%
15	0.48	0.66	0.75	0.5	$6.30 * 10^5$	0.2346%	48.05%	8.984%	86.5%	86.4%	86.1%	4.31%
16	0.48	0.66	0.75	0.5	$6.60 * 10^5$	0.2460%	50.39%	8.438%	87.0%	87.0%	86.6%	4.76%
17	0.48	0.66	0.75	0.5	$6.91 * 10^5$	0.2575%	52.73%	7.969%	87.4%	87.3%	87.0%	3.37%
18	0.48	0.66	0.75	0.5	$7.22 * 10^5$	0.2689%	55.08%	7.500%	88.0%	87.9%	87.5%	3.90%
19	0.48	0.66	0.75	0.5	$7.53 * 10^5$	0.2804%	57.42%	7.188%	88.2%	88.1%	87.7%	2.74%
20	0.48	0.66	0.75	0.5	$7.83 * 10^5$	0.2918%	59.77%	6.953%	88.3%	88.2%	87.7%	2.67%
21	0.48	0.66	0.75	0.5	$8.14 * 10^5$	0.3033%	62.11%	6.641%	88.6%	88.5%	88.0%	2.35%
22	0.48	0.66	0.75	0.5	$8.45 * 10^5$	0.3147%	64.45%	6.016%	89.1%	88.7%	88.2%	2.20%
23	0.48	0.66	0.75	0.5	$8.76 * 10^5$	0.3262%	66.80%	5.859%	89.1%	88.8%	88.3%	2.12%
24	0.46	0.62	0.65	0.9	$9.06 * 10^5$	0.3376%	69.14%	6.016%	88.7%	88.7%	88.5%	1.88%
25	0.48	0.66	0.75	0.5	$9.37 * 10^5$	0.3490%	71.48%	5.859%	89.1%	89.1%	88.6%	1.81%
26	0.46	0.62	0.65	0.7	$9.68 * 10^5$	0.3605%	73.83%	5.703%	88.8%	88.8%	88.7%	1.73%
32‡	0.46	0.62	0.65	0.7	$9.88 * 10^05$	0.3681%	75.39%	5.312%	88.9%	88.9%	88.8%	1.65%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary SMR ω_{256}.

‡ Without tree pre-selection.

Table A.19: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 26$ trees using SMR-CPCA-M templates at $\mathcal{T} = 32$ with an additional binary PSML-CPCA comparison and masking out most common set bits.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.64	0.65	$6.76 * 10^5$	0.2518%	51.56%	28.67%	70.5%	70.2%	69.8%	20.6%
2	0.49	0.64	0.70	$6.96 * 10^5$	0.2594%	53.12%	17.81%	80.5%	80.1%	79.9%	10.5%
3	0.49	0.64	0.70	$7.17 * 10^5$	0.2670%	54.69%	13.98%	84.1%	83.8%	83.6%	6.8%
4	0.48	0.65	0.70	$7.37 * 10^5$	0.2747%	56.25%	11.80%	85.4%	85.2%	84.5%	5.9%
5	0.48	0.66	0.70	$7.58 * 10^5$	0.2823%	57.81%	9.844%	87.3%	86.5%	84.5%	5.9%
6	0.48	0.66	0.70	$7.78 * 10^5$	0.2899%	59.38%	8.516%	87.9%	87.5%	85.5%	4.9%
7	0.49	0.68	0.65	$7.99 * 10^5$	0.2975%	60.94%	7.578%	89.3%	88.4%	86.4%	4.0%
8	0.49	0.68	0.70	$8.19 * 10^5$	0.3052%	62.50%	6.875%	89.9%	88.9%	87.0%	3.5%
9	0.47	0.65	0.70	$8.40 * 10^5$	0.3128%	64.06%	6.094%	89.6%	89.4%	88.8%	1.7%
10	0.49	0.68	0.70	$8.60 * 10^5$	0.3204%	65.62%	5.781%	90.2%	89.6%	87.5%	2.9%
11	0.47	0.60	0.75	$8.81 * 10^5$	0.3281%	67.19%	5.312%	90.2%	89.7%	87.5%	2.9%
12	0.47	0.60	0.75	$9.01 * 10^5$	0.3357%	68.75%	5.156%	90.5%	89.9%	87.6%	2.8%
13	0.48	0.66	0.70	$9.22 * 10^5$	0.3433%	70.31%	4.844%	90.5%	90.0%	87.7%	2.7%
14	0.48	0.66	0.70	$9.42 * 10^5$	0.3510%	71.88%	4.609%	90.6%	90.2%	88.0%	2.4%
15	0.48	0.65	0.70	$9.63 * 10^5$	0.3586%	73.44%	4.219%	90.9%	90.4%	88.1%	2.3%
16	0.48	0.65	0.75	$9.83 * 10^5$	0.3662%	75.00%	4.062%	90.8%	90.3%	88.1%	2.3%
17	0.49	0.67	0.70	$1.00 * 10^6$	0.3738%	76.56%	3.906%	90.9%	90.5%	88.1%	2.3%
18	0.49	0.67	0.75	$1.02 * 10^6$	0.3815%	78.12%	3.828%	91.0%	90.5%	88.2%	2.2%
19	0.49	0.67	0.75	$1.04 * 10^6$	0.3891%	79.69%	3.828%	91.0%	90.5%	88.2%	2.2%
20	0.47	0.66	0.75	$1.06 * 10^6$	0.3967%	81.25%	3.672%	91.0%	90.7%	88.4%	2.0%
21	0.47	0.64	0.70	$1.09 * 10^6$	0.4044%	82.81%	3.594%	91.2%	90.7%	88.4%	2.0%
22	0.46	0.62	0.75	$1.11 * 10^6$	0.4120%	84.38%	3.516%	91.2%	90.8%	88.5%	1.9%
23	0.46	0.62	0.75	$1.13 * 10^6$	0.4196%	85.94%	3.516%	91.2%	90.8%	88.5%	1.9%
24	0.49	0.68	0.70	$1.15 * 10^6$	0.4272%	87.50%	3.516%	91.1%	90.8%	88.4%	2.0%
25	0.49	0.68	0.70	$1.17 * 10^6$	0.4349%	89.06%	3.516%	91.1%	90.8%	88.4%	2.0%
26	0.47	0.60	0.75	$1.19 * 10^6$	0.4425%	90.62%	3.438%	91.2%	90.9%	88.5%	1.9%
27	0.47	0.63	0.60	$1.21 * 10^6$	0.4501%	92.19%	3.438%	91.3%	90.9%	88.5%	1.9%
28	0.48	0.64	0.65	$1.23 * 10^6$	0.4578%	93.75%	3.359%	91.2%	90.9%	88.5%	1.9%
29	0.48	0.64	0.65	$1.25 * 10^6$	0.4654%	95.31%	3.359%	91.2%	90.9%	88.5%	1.9%
30	0.48	0.64	0.65	$1.27 * 10^6$	0.4730%	96.88%	3.359%	91.2%	90.9%	88.5%	1.9%
31	0.47	0.64	0.70	$1.29 * 10^6$	0.4807%	98.44%	3.359%	91.2%	90.8%	88.5%	1.9%
32	0.47	0.64	0.70	$1.31 * 10^6$	0.4883%	100.0%	3.359%	91.2%	90.8%	88.5%	1.9%
33	0.47	0.60	0.75	$1.33 * 10^6$	0.4959%	101.6%	3.281%	91.2%	90.8%	88.5%	1.9%
34	0.47	0.60	0.75	$1.35 * 10^6$	0.5035%	103.1%	3.281%	91.2%	90.8%	88.5%	1.9%
35	0.47	0.60	0.75	$1.37 * 10^6$	0.5112%	104.7%	3.281%	91.2%	90.8%	88.5%	1.9%
36	0.47	0.60	0.75	$1.39 * 10^6$	0.5188%	106.2%	3.281%	91.2%	90.8%	88.5%	1.9%
37	0.46	0.70	0.70	$1.41 * 10^6$	0.5264%	107.8%	3.281%	91.0%	90.7%	88.4%	2.0%
38	0.48	0.62	0.70	$1.43 * 10^6$	0.5341%	109.4%	3.281%	91.2%	90.8%	88.5%	1.9%
39	0.46	0.70	0.70	$1.45 * 10^6$	0.5417%	110.9%	3.203%	91.0%	90.7%	88.4%	2.0%
40	0.46	0.70	0.70	$1.47 * 10^6$	0.5493%	112.5%	3.203%	91.0%	90.7%	88.4%	2.0%
41	0.46	0.70	0.70	$1.50 * 10^6$	0.5569%	114.1%	3.203%	91.0%	90.7%	88.4%	2.0%
42	0.46	0.70	0.70	$1.52 * 10^6$	0.5646%	115.6%	3.203%	91.0%	90.7%	88.4%	2.0%
43	0.46	0.70	0.70	$1.54 * 10^6$	0.5722%	117.2%	3.203%	91.0%	90.7%	88.4%	2.0%
44	0.46	0.70	0.70	$1.56 * 10^6$	0.5798%	118.8%	3.125%	91.1%	90.8%	88.4%	2.0%
45	0.46	0.70	0.70	$1.58 * 10^6$	0.5875%	120.3%	3.125%	91.1%	90.8%	88.4%	2.0%
46	0.46	0.70	0.70	$1.60 * 10^6$	0.5951%	121.9%	3.125%	91.1%	90.7%	88.4%	2.0%
47	0.46	0.70	0.70	$1.62 * 10^6$	0.6027%	123.4%	3.125%	91.1%	90.7%	88.4%	2.0%
48	0.46	0.70	0.70	$1.64 * 10^6$	0.6104%	125.0%	3.125%	91.1%	90.7%	88.4%	2.0%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary SMR ω_{256}.

Table A.20: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional real PSML-CPCA comparison.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	$F_B{}^*$	$F_R{}^\dagger$	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.49	0.60	0.70	$5.22 * 10^5$	0.1945%	39.84%	55.23%	44.1%	43.8%	43.8%	46.7%
2	0.49	0.60	0.75	$5.53 * 10^5$	0.206%	42.19%	41.25%	58%	57.7%	57.3%	33.1%
3	0.48	0.65	0.75	$5.84 * 10^5$	0.2174%	44.53%	32.66%	66%	65.9%	65.6%	24.8%
4	0.46	0.63	0.70	$6.14 * 10^5$	0.2289%	46.88%	26.88%	71.8%	71.7%	71.2%	19.2%
5	0.48	0.67	0.70	$6.45 * 10^5$	0.2403%	49.22%	22.66%	75.7%	75.6%	74.5%	15.9%
6	0.48	0.66	0.75	$6.76 * 10^5$	0.2518%	51.56%	19.45%	78.9%	78.3%	78.1%	12.3%
7	0.47	0.64	0.75	$7.07 * 10^5$	0.2632%	53.91%	17.27%	80.4%	79.7%	79.1%	11.3%
8	0.49	0.68	0.70	$7.37 * 10^5$	0.2747%	56.25%	15%	82.5%	82.2%	81.2%	9.2%
9	0.49	0.68	0.70	$7.68 * 10^5$	0.2861%	58.59%	13.44%	83.8%	83.6%	82.7%	7.74%
10	0.47	0.64	0.70	$7.99 * 10^5$	0.2975%	60.94%	12.34%	84.4%	83.6%	83.4%	7.04%
11	0.47	0.64	0.75	$8.29 * 10^5$	0.309%	63.28%	11.25%	85.7%	84.7%	84.5%	5.87%
12	0.47	0.64	0.70	$8.60 * 10^5$	0.3204%	65.62%	10.16%	86.4%	85.5%	85.4%	5.01%
13	0.48	0.65	0.70	$8.91 * 10^5$	0.3319%	67.97%	9.141%	87.2%	86.3%	86.2%	4.23%
14	0.48	0.65	0.70	$9.22 * 10^5$	0.3433%	70.31%	8.438%	87.9%	87%	86.9%	3.53%
15	0.48	0.65	0.70	$9.52 * 10^5$	0.3548%	72.66%	7.734%	88.5%	87.7%	87.5%	2.9%
16	0.47	0.64	0.70	$9.83 * 10^5$	0.3662%	75%	7.266%	89.1%	88.4%	88.2%	2.2%
17	0.47	0.64	0.70	$1.01 * 10^6$	0.3777%	77.34%	6.875%	88.8%	88%	86.9%	3.53%
18	0.47	0.64	0.70	$1.04 * 10^6$	0.3891%	79.69%	6.641%	89.2%	88.4%	87.2%	3.21%
19	0.47	0.64	0.70	$1.08 * 10^6$	0.4005%	82.03%	6.25%	89.5%	88.7%	87.5%	2.9%
20	0.47	0.64	0.70	$1.11 * 10^6$	0.412%	84.38%	5.859%	89.5%	89.1%	86.9%	3.53%
21	0.47	0.64	0.70	$1.14 * 10^6$	0.4234%	86.72%	5.781%	89.6%	89.1%	87%	3.45%
22	0.47	0.64	0.70	$1.17 * 10^6$	0.4349%	89.06%	5.625%	89.7%	89.2%	87%	3.45%
23	0.48	0.62	0.70	$1.20 * 10^6$	0.4463%	91.41%	5.312%	89.5%	89.2%	88.5%	1.88%
24	0.49	0.65	0.65	$1.23 * 10^6$	0.4578%	93.75%	5.078%	89.8%	89.5%	87.3%	3.06%
25	0.49	0.65	0.65	$1.26 * 10^6$	0.4692%	96.09%	4.922%	90%	89.6%	87.5%	2.9%
26	0.48	0.62	0.65	$1.29 * 10^6$	0.4807%	98.44%	4.766%	90.2%	89.8%	87.5%	2.9%
27	0.48	0.62	0.70	$1.32 * 10^6$	0.4921%	100.8%	4.688%	90.2%	89.8%	87.5%	2.9%
28	0.49	0.65	0.65	$1.35 * 10^6$	0.5035%	103.1%	4.531%	90.2%	89.8%	87.7%	2.74%

* F compared to the PSML ω_{256}.

† F compared to the naïve binary SMR ω_{256}.

Table A.21: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 26$ trees using SMR-CPCA-M templates at $\mathcal{T} = 32$ with an additional real PSML-CPCA comparison.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.49	0.64	0.65	0.8	$6.76 * 10^5$	0.2518%	51.56%	28.05%	71.3%	70.9%	67.6%	22.8%
2	0.49	0.64	0.70	0.9	$6.96 * 10^5$	0.2594%	53.12%	17.66%	80.9%	80.4%	73.9%	16.5%
3	0.49	0.64	0.70	0.7	$7.17 * 10^5$	0.2670%	54.69%	12.81%	84.5%	84.1%	83.4%	7.0%
4	0.49	0.64	0.70	0.6	$7.37 * 10^5$	0.2747%	56.25%	10.94%	85.7%	85.1%	84.8%	5.6%
5	0.48	0.66	0.70	0.9	$7.58 * 10^5$	0.2823%	57.81%	9.844%	86.0%	85.8%	85.5%	4.9%
6	0.48	0.66	0.70	0.7	$7.78 * 10^5$	0.2899%	59.38%	9.141%	86.5%	86.2%	85.8%	4.6%
7	0.49	0.68	0.65	0.8	$7.99 * 10^5$	0.2975%	60.94%	8.359%	87.3%	87.1%	86.7%	3.7%
8	0.49	0.68	0.70	0.6	$8.19 * 10^5$	0.3052%	62.50%	8.047%	87.6%	87.3%	86.9%	3.5%
9	0.47	0.65	0.70	0.9	$8.40 * 10^5$	0.3128%	64.06%	7.500%	87.8%	87.4%	87.1%	3.3%
10	0.49	0.68	0.70	0.7	$8.60 * 10^5$	0.3204%	65.62%	7.031%	88.5%	87.7%	87.3%	3.1%
11	0.47	0.60	0.75	0.9	$8.81 * 10^5$	0.3281%	67.19%	6.406%	88.6%	87.9%	84.9%	5.5%
12	0.47	0.60	0.75	0.9	$9.01 * 10^5$	0.3357%	68.75%	6.328%	88.7%	88.0%	84.9%	5.5%
13	0.48	0.66	0.70	0.9	$9.22 * 10^5$	0.3433%	70.31%	5.469%	89.0%	88.7%	88.3%	2.1%
14	0.48	0.66	0.70	0.9	$9.42 * 10^5$	0.3510%	71.88%	5.234%	89.0%	88.6%	87.4%	3.0%
15	0.48	0.65	0.70	0.9	$9.63 * 10^5$	0.3586%	73.44%	5.156%	89.2%	88.8%	87.6%	2.8%
16	0.48	0.65	0.75	0.9	$9.83 * 10^5$	0.3662%	75.00%	5.000%	89.2%	88.5%	85.3%	5.1%
17	0.49	0.67	0.70	0.7	$1.00 * 10^6$	0.3738%	76.56%	4.844%	89.1%	88.4%	85.3%	5.1%
18	0.49	0.67	0.75	0.8	$1.02 * 10^6$	0.3815%	78.12%	4.609%	89.3%	88.6%	85.4%	5.1%
19	0.49	0.67	0.75	0.9	$1.04 * 10^6$	0.3891%	79.69%	4.375%	89.4%	88.8%	85.5%	4.9%
20	0.47	0.66	0.75	0.9	$1.06 * 10^6$	0.3967%	81.25%	4.219%	89.9%	89.4%	88.0%	2.4%
21	0.47	0.64	0.70	0.9	$1.09 * 10^6$	0.4044%	82.81%	4.375%	89.7%	89.0%	88.0%	2.4%
22	0.46	0.62	0.75	0.9	$1.11 * 10^6$	0.4120%	84.38%	3.828%	89.6%	88.4%	81.6%	8.8%
23	0.46	0.62	0.75	0.8	$1.13 * 10^6$	0.4196%	85.94%	3.750%	89.7%	88.4%	81.6%	8.8%
24	0.47	0.64	0.70	0.9	$1.15 * 10^6$	0.4272%	87.50%	3.906%	90.1%	89.4%	88.4%	2.0%
25	0.47	0.66	0.75	0.9	$1.17 * 10^6$	0.4349%	89.06%	3.906%	90.2%	89.5%	88.4%	2.0%
26	0.48	0.64	0.65	0.9	$1.19 * 10^6$	0.4425%	90.62%	3.750%	90.1%	89.4%	88.4%	2.0%
27	0.48	0.64	0.65	0.9	$1.21 * 10^6$	0.4501%	92.19%	3.594%	90.2%	89.5%	88.5%	1.9%
28	0.48	0.64	0.65	0.9	$1.23 * 10^6$	0.4578%	93.75%	3.594%	90.2%	89.5%	88.5%	1.9%
29	0.48	0.64	0.65	0.9	$1.25 * 10^6$	0.4654%	95.31%	3.672%	90.2%	89.5%	88.5%	1.9%
30	0.48	0.64	0.65	0.9	$1.27 * 10^6$	0.4730%	96.88%	3.672%	90.2%	89.5%	88.5%	1.9%
31	0.47	0.64	0.70	0.9	$1.29 * 10^6$	0.4807%	98.44%	3.594%	90.5%	89.8%	88.8%	1.7%
32	0.47	0.64	0.70	0.9	$1.31 * 10^6$	0.4883%	100.0%	3.594%	90.5%	89.8%	88.8%	1.7%
33	0.47	0.60	0.75	0.9	$1.33 * 10^6$	0.4959%	101.6%	3.828%	89.9%	88.8%	81.8%	8.6%
34	0.49	0.63	0.65	0.5	$1.35 * 10^6$	0.5035%	103.1%	4.062%	90.4%	88.8%	86.0%	4.4%
35	0.49	0.63	0.65	0.5	$1.37 * 10^6$	0.5112%	104.7%	4.062%	90.4%	88.8%	86.0%	4.4%
36	0.49	0.63	0.65	0.5	$1.39 * 10^6$	0.5188%	106.2%	4.062%	90.4%	88.8%	86.0%	4.4%
37	0.49	0.63	0.65	0.5	$1.41 * 10^6$	0.5264%	107.8%	4.062%	90.4%	88.8%	86.0%	4.4%
38	0.49	0.63	0.65	0.5	$1.43 * 10^6$	0.5341%	109.4%	4.062%	90.4%	88.8%	86.0%	4.4%
39	0.46	0.70	0.70	0.9	$1.45 * 10^6$	0.5417%	110.9%	3.438%	90.4%	89.7%	88.7%	1.7%
40	0.46	0.70	0.70	0.9	$1.47 * 10^6$	0.5493%	112.5%	3.438%	90.4%	89.7%	88.7%	1.7%
41	0.46	0.70	0.70	0.9	$1.50 * 10^6$	0.5569%	114.1%	3.359%	90.4%	89.7%	88.7%	1.7%
42	0.46	0.70	0.70	0.9	$1.52 * 10^6$	0.5646%	115.6%	3.359%	90.4%	89.7%	88.7%	1.7%
43	0.46	0.70	0.70	0.9	$1.54 * 10^6$	0.5722%	117.2%	3.359%	90.4%	89.7%	88.7%	1.7%
44	0.46	0.70	0.70	0.9	$1.56 * 10^6$	0.5798%	118.8%	3.359%	90.4%	89.7%	88.7%	1.7%
45	0.46	0.70	0.70	0.9	$1.58 * 10^6$	0.5875%	120.3%	3.359%	90.4%	89.7%	88.7%	1.7%
46	0.46	0.70	0.70	0.9	$1.60 * 10^6$	0.5951%	121.9%	3.359%	90.4%	89.7%	88.7%	1.7%
47	0.46	0.70	0.70	0.9	$1.62 * 10^6$	0.6027%	123.4%	3.359%	90.4%	89.7%	88.7%	1.7%
48	0.46	0.70	0.70	0.9	$1.64 * 10^6$	0.6104%	125.0%	3.359%	90.4%	89.7%	88.7%	1.7%

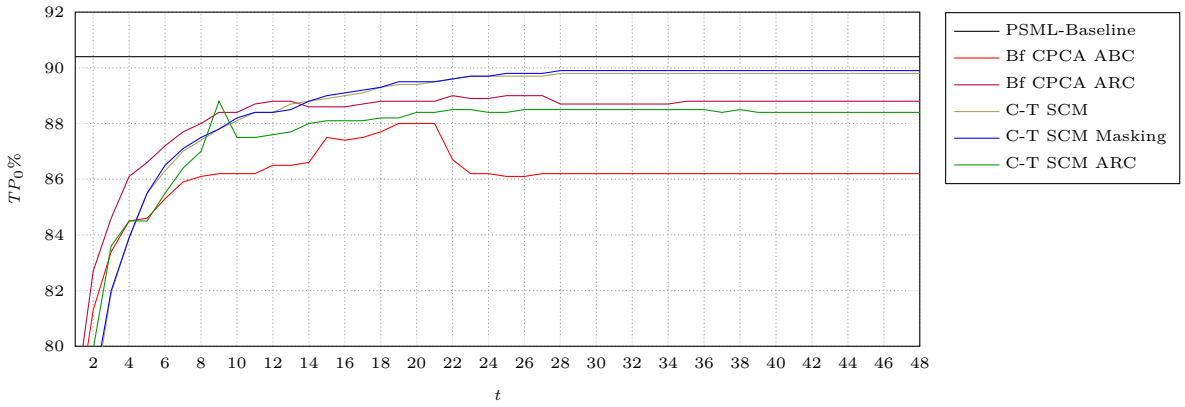
* F compared to the PSML ω_{256}.

† F compared to the naïve binary PSML-CPCA ω_{256}.

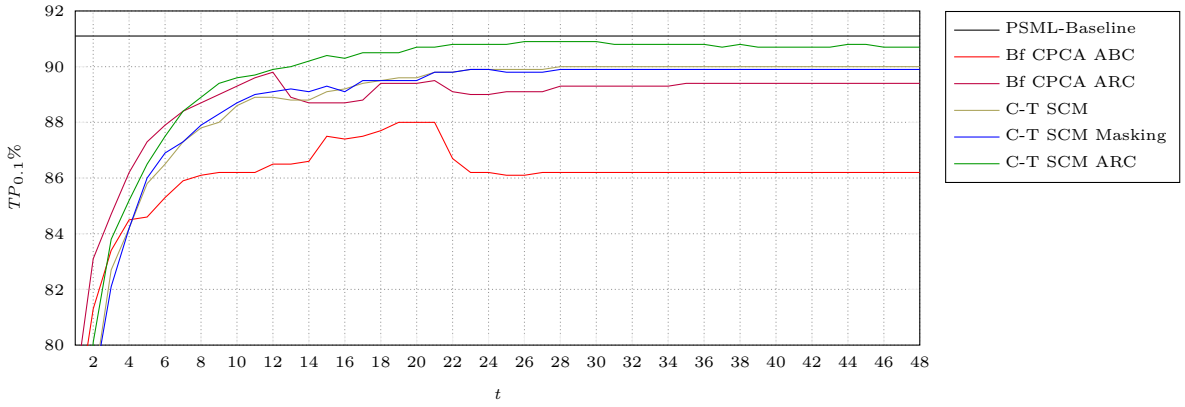
Table A.22: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional real SML-CPCA comparison and masking out most common bits.

Approach	Workload		Ident.			Verif.	Recognition Performance
	ω_{256}	F	$TP_{0.5}$	$TP_{0.1}$	TP_0	EER	Loss (TP_0)
SML-Baseline	2.68×10^8	–	78.1%	75.1%	72.4%	3.4%	–
Binary SML-CPCA (MT= 0.75)	1.31×10^6	0.4883%	77.5%	75.3%	73.5%	3.2%	+1.1%

Table A.23: Summary of achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for SMR biometric identification system using all sessions of the PolyU dataset and their corresponding verification EER with applied feature reduction approaches, ordered by TP_0.



(a) TP_0



(b) $TP_{0.1}$

Figure A.6: TP_0 and $TP_{0.1}$ values for relevant experiments at each t. Used abbreviations: Bf (Bloom filter), C-T (CPCA-Tree), ABC (Additional Binary Comparison), ARC (Additional Real Comparison), SCM (SMR-CPCA-M).

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	$F_B{}^*$	$F_R{}^\dagger$	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	5	5	0.1	$4.51 * 10^5$	0.1679%	34.39%	67.4%	65.6%	58.1%	14.3%
2	6	6	0.4	$6.60 * 10^5$	0.2460%	50.39%	66.8%	65.8%	62.2%	10.2%
3	8	6	0.3	$5.24 * 10^5$	0.1952%	39.97%	69.4%	68.2%	64.9%	7.5%
4	8	4	0.2	$2.10 * 10^5$	0.0782%	16.02%	71.8%	68.5%	65.4%	7.0%
5	8	4	0.2	$2.20 * 10^5$	0.0820%	16.80%	73.7%	69.9%	66.5%	5.9%
6	8	4	0.5	$2.30 * 10^5$	0.0858%	17.58%	74.6%	70.7%	66.8%	5.6%
7	8	4	0.5	$2.41 * 10^5$	0.0896%	18.36%	75.7%	71.8%	67.8%	4.6%
8	8	4	0.5	$2.51 * 10^5$	0.0935%	19.14%	76.3%	72.5%	68.4%	4.0%
9	7	6	0.9	$7.85 * 10^5$	0.2926%	59.91%	71.3%	69.6%	65.9%	6.5%
10	7	6	0.9	$8.17 * 10^5$	0.3042%	62.30%	71.9%	70.1%	66.4%	6.0%
11	7	6	0.9	$8.48 * 10^5$	0.3158%	64.68%	72.1%	70.6%	66.8%	5.6%
12	7	6	0.9	$8.79 * 10^5$	0.3274%	67.06%	72.7%	71.1%	67.2%	5.2%
13	7	6	0.9	$9.10 * 10^5$	0.3391%	69.44%	72.9%	70.6%	67.5%	4.9%
14	7	6	0.9	$9.41 * 10^5$	0.3507%	71.82%	73.1%	70.7%	67.7%	4.7%
15	7	6	0.9	$9.73 * 10^5$	0.3623%	74.20%	73.3%	71.0%	67.8%	4.6%
16	7	6	0.9	$1.00 * 10^6$	0.3739%	76.58%	73.5%	71.2%	68.0%	4.4%
17	7	6	0.9	$1.03 * 10^6$	0.3856%	78.96%	73.8%	71.4%	68.2%	4.2%
18	7	6	0.9	$1.07 * 10^6$	0.3972%	81.34%	73.9%	71.6%	68.3%	4.1%
19	7	6	0.9	$1.10 * 10^6$	0.4088%	83.72%	74.0%	71.7%	68.4%	4.0%
20	7	6	0.9	$1.13 * 10^6$	0.4204%	86.10%	74.0%	71.7%	68.4%	4.0%
21	7	6	0.9	$1.16 * 10^6$	0.4321%	88.49%	74.2%	71.9%	68.5%	3.9%
22	7	6	0.9	$1.19 * 10^6$	0.4437%	90.87%	74.3%	71.9%	68.5%	3.9%
23	7	6	0.9	$1.22 * 10^6$	0.4553%	93.25%	74.3%	71.9%	68.6%	3.8%
24	7	6	0.9	$1.25 * 10^6$	0.4669%	95.63%	74.4%	72.1%	68.8%	3.7%
25	7	6	0.9	$1.28 * 10^6$	0.4786%	98.01%	74.5%	72.1%	68.8%	3.6%
26	7	6	0.9	$1.32 * 10^6$	0.4902%	100.39%	74.4%	72.1%	68.8%	3.6%
27	7	6	0.9	$1.35 * 10^6$	0.5018%	102.77%	74.5%	72.2%	68.9%	3.5%
28	7	6	0.9	$1.38 * 10^6$	0.5134%	105.15%	74.5%	72.2%	68.9%	3.5%
29	7	6	0.9	$1.41 * 10^6$	0.5251%	107.53%	74.5%	72.2%	68.9%	3.5%
30	7	6	0.9	$1.44 * 10^6$	0.5367%	109.91%	74.6%	72.3%	68.9%	3.5%
31	7	6	0.9	$1.47 * 10^6$	0.5483%	112.30%	74.7%	72.3%	69.0%	3.4%
32	7	6	0.9	$1.50 * 10^6$	0.5599%	114.68%	74.8%	72.4%	69.0%	3.4%
33	7	6	0.9	$1.53 * 10^6$	0.5716%	117.06%	74.8%	72.4%	69.1%	3.3%
34	7	6	0.9	$1.57 * 10^6$	0.5832%	119.44%	74.8%	72.4%	69.1%	3.3%
35	7	6	0.9	$1.60 * 10^6$	0.5948%	121.82%	74.8%	72.4%	69.1%	3.3%
36	7	6	0.9	$1.63 * 10^6$	0.6064%	124.20%	74.9%	71.6%	69.1%	3.3%
37	7	6	0.9	$1.66 * 10^6$	0.6181%	126.58%	74.9%	71.6%	69.1%	3.3%
38	7	6	0.9	$1.69 * 10^6$	0.6297%	128.96%	74.9%	71.6%	69.1%	3.3%
39	7	6	0.9	$1.72 * 10^6$	0.6413%	131.34%	74.9%	71.7%	69.2%	3.2%
40	7	6	0.9	$1.75 * 10^6$	0.6529%	133.72%	74.9%	71.7%	69.2%	3.2%
41	7	6	0.9	$1.78 * 10^6$	0.6646%	136.10%	74.9%	71.7%	69.2%	3.2%
42	7	6	0.9	$1.82 * 10^6$	0.6762%	138.49%	74.9%	71.7%	69.2%	3.2%
43	7	6	0.9	$1.85 * 10^6$	0.6878%	140.87%	74.9%	71.7%	69.2%	3.2%
44	7	6	0.9	$1.88 * 10^6$	0.6995%	143.25%	75.0%	71.7%	69.2%	3.2%
45	7	6	0.9	$1.91 * 10^6$	0.7111%	145.63%	75.0%	71.7%	69.2%	3.2%
46	7	6	0.9	$1.94 * 10^6$	0.7227%	148.01%	75.0%	71.7%	69.2%	3.2%
47	7	6	0.9	$1.97 * 10^6$	0.7343%	150.39%	75.0%	71.7%	69.2%	3.2%
48	7	6	0.9	$2.00 * 10^6$	0.7460%	152.77%	75.0%	71.7%	69.2%	3.2%

* F compared to the SML ω_{256}.

† F compared to the naïve binary SML-CPCA ω_{256}.

Table A.24: Best achieved binary SML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final real-valued SML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$ using both sessions of the PolyU dataset.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.48	0.64	0.75	$3.48 * 10^5$	0.1297%	26.56%	37.18%	59.5%	58.5%	56.5%	15.9%
2	0.48	0.64	0.75	$3.69 * 10^5$	0.1373%	28.12%	26.63%	67.5%	66.5%	63.6%	8.8%
3	0.45	0.60	0.80	$3.89 * 10^5$	0.1450%	29.69%	23.40%	69.9%	68.4%	64.7%	7.7%
4	0.45	0.60	0.80	$4.10 * 10^5$	0.1526%	31.25%	20.38%	71.9%	70.3%	66.4%	6.0%
5	0.45	0.60	0.80	$4.30 * 10^5$	0.1602%	32.81%	18.00%	72.9%	71.8%	67.8%	4.6%
6	0.45	0.60	0.80	$4.51 * 10^5$	0.1678%	34.38%	16.34%	74.1%	72.9%	68.8%	3.6%
7	0.45	0.60	0.80	$4.71 * 10^5$	0.1755%	35.94%	15.20%	74.7%	73.4%	69.2%	3.2%
8	0.45	0.60	0.80	$4.92 * 10^5$	0.1831%	37.50%	13.60%	75.3%	74.0%	69.7%	2.7%
9	0.45	0.60	0.80	$5.12 * 10^5$	0.1907%	39.06%	13.03%	75.8%	74.4%	70.0%	2.4%
10	0.45	0.60	0.80	$5.32 * 10^5$	0.1984%	40.62%	12.50%	76.2%	73.1%	70.2%	2.2%
11	0.45	0.60	0.80	$5.53 * 10^5$	0.2060%	42.19%	11.75%	76.7%	74.0%	70.6%	1.8%
12	0.45	0.60	0.80	$5.73 * 10^5$	0.2136%	43.75%	11.12%	77.0%	74.3%	70.8%	1.6%
13	0.45	0.60	0.80	$5.94 * 10^5$	0.2213%	45.31%	10.83%	77.1%	74.3%	70.8%	1.6%
14	0.45	0.60	0.80	$6.14 * 10^5$	0.2289%	46.88%	10.58%	77.2%	74.5%	70.9%	1.5%
15	0.45	0.60	0.80	$6.35 * 10^5$	0.2365%	48.44%	10.44%	77.3%	74.5%	70.9%	1.5%
16	0.45	0.60	0.80	$6.55 * 10^5$	0.2441%	50.00%	10.30%	77.3%	74.5%	70.9%	1.5%
17	0.45	0.60	0.80	$6.76 * 10^5$	0.2518%	51.56%	10.05%	77.2%	74.6%	71.0%	1.4%
18	0.45	0.60	0.80	$6.96 * 10^5$	0.2594%	53.12%	9.80%	77.3%	74.6%	71.0%	1.4%
19	0.45	0.60	0.80	$7.17 * 10^5$	0.2670%	54.69%	9.59%	77.3%	74.7%	71.0%	1.4%
20	0.45	0.60	0.80	$7.37 * 10^5$	0.2747%	56.25%	9.38%	77.4%	74.8%	71.1%	1.3%
21	0.45	0.60	0.80	$7.58 * 10^5$	0.2823%	57.81%	9.38%	77.4%	74.8%	71.1%	1.3%
22	0.45	0.60	0.80	$7.78 * 10^5$	0.2899%	59.38%	9.41%	77.4%	74.8%	71.1%	1.3%
23	0.45	0.60	0.80	$7.99 * 10^5$	0.2975%	60.94%	9.30%	77.5%	74.8%	71.1%	1.3%
24	0.45	0.60	0.80	$8.19 * 10^5$	0.3052%	62.50%	9.23%	77.5%	74.9%	71.1%	1.3%
25	0.45	0.60	0.80	$8.40 * 10^5$	0.3128%	64.06%	9.09%	77.5%	74.9%	71.1%	1.3%
26	0.45	0.60	0.80	$8.60 * 10^5$	0.3204%	65.62%	9.06%	77.5%	74.9%	71.1%	1.3%
27	0.45	0.60	0.80	$8.81 * 10^5$	0.3281%	67.19%	8.88%	77.6%	74.9%	71.1%	1.3%
28	0.45	0.60	0.80	$9.01 * 10^5$	0.3357%	68.75%	8.74%	77.6%	74.9%	71.2%	1.2%
29	0.45	0.60	0.80	$9.22 * 10^5$	0.3433%	70.31%	8.66%	77.6%	74.9%	71.2%	1.2%
30	0.45	0.60	0.80	$9.42 * 10^5$	0.3510%	71.88%	8.59%	77.6%	75.0%	71.2%	1.2%
31	0.45	0.60	0.80	$9.63 * 10^5$	0.3586%	73.44%	8.49%	77.7%	75.0%	71.2%	1.2%
32	0.45	0.60	0.80	$9.83 * 10^5$	0.3662%	75.00%	8.31%	77.7%	75.0%	71.2%	1.2%
33	0.45	0.60	0.80	$1.00 * 10^6$	0.3738%	76.56%	8.27%	77.7%	75.0%	71.2%	1.2%
34	0.45	0.60	0.80	$1.02 * 10^6$	0.3815%	78.12%	8.24%	77.7%	75.0%	71.2%	1.2%
35	0.45	0.60	0.80	$1.04 * 10^6$	0.3891%	79.69%	8.24%	77.7%	75.0%	71.2%	1.2%
36	0.45	0.60	0.80	$1.06 * 10^6$	0.3967%	81.25%	8.17%	77.7%	75.1%	71.3%	1.1%
37	0.45	0.60	0.80	$1.09 * 10^6$	0.4044%	82.81%	8.17%	77.7%	75.1%	71.3%	1.1%
38	0.45	0.60	0.80	$1.11 * 10^6$	0.4120%	84.38%	8.13%	77.7%	75.1%	71.3%	1.1%
39	0.45	0.60	0.80	$1.13 * 10^6$	0.4196%	85.94%	8.13%	77.7%	75.1%	71.3%	1.1%
40	0.45	0.60	0.80	$1.15 * 10^6$	0.4272%	87.50%	8.13%	77.7%	75.1%	71.3%	1.1%
41	0.45	0.60	0.80	$1.17 * 10^6$	0.4349%	89.06%	8.10%	77.7%	75.1%	71.3%	1.1%
42	0.45	0.60	0.80	$1.19 * 10^6$	0.4425%	90.62%	8.03%	77.8%	75.1%	71.3%	1.1%
43	0.45	0.60	0.80	$1.21 * 10^6$	0.4501%	92.19%	7.99%	77.8%	75.1%	71.3%	1.1%
44	0.45	0.60	0.80	$1.23 * 10^6$	0.4578%	93.75%	7.99%	77.8%	75.1%	71.3%	1.1%
45	0.45	0.60	0.80	$1.25 * 10^6$	0.4654%	95.31%	7.95%	77.8%	75.1%	71.3%	1.1%
46	0.45	0.60	0.80	$1.27 * 10^6$	0.4730%	96.88%	7.92%	77.8%	75.1%	71.3%	1.1%
47	0.45	0.60	0.80	$1.29 * 10^6$	0.4807%	98.44%	7.92%	77.8%	75.1%	71.3%	1.1%
48	0.45	0.60	0.80	$1.31 * 10^6$	0.4883%	100.00%	7.92%	77.8%	75.1%	71.3%	1.1%

* F compared to the SML ω_{256}.

† F compared to the naïve binary SML-CPCA ω_{256}.

Table A.25: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with both sessions of the PolyU dataset.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.48	0.64	0.75	$6.76 * 10^5$	0.2518%	51.56%	37.18%	59.7%	58.3%	57.0%	15.4%
2	0.48	0.64	0.75	$6.96 * 10^5$	0.2594%	53.12%	26.63%	68.1%	66.3%	64.7%	7.7%
3	0.45	0.60	0.80	$7.17 * 10^5$	0.2670%	54.69%	23.40%	69.0%	67.2%	65.5%	6.9%
4	0.45	0.60	0.80	$7.37 * 10^5$	0.2747%	56.25%	20.38%	71.0%	68.8%	67.2%	5.2%
5	0.45	0.60	0.80	$7.58 * 10^5$	0.2823%	57.81%	18.00%	72.8%	70.5%	68.7%	3.7%
6	0.45	0.60	0.80	$7.78 * 10^5$	0.2899%	59.38%	16.34%	73.7%	71.4%	69.5%	2.9%
7	0.45	0.60	0.80	$7.99 * 10^5$	0.2975%	60.94%	15.20%	74.2%	72.2%	69.9%	2.5%
8	0.45	0.60	0.80	$8.19 * 10^5$	0.3052%	62.50%	13.60%	74.8%	72.6%	70.3%	2.1%
9	0.45	0.60	0.80	$8.40 * 10^5$	0.3128%	64.06%	13.03%	75.2%	72.9%	70.5%	1.9%
10	0.45	0.60	0.83	$8.60 * 10^5$	0.3204%	65.62%	12.50%	75.5%	73.3%	70.9%	1.5%
11	0.45	0.60	0.80	$8.81 * 10^5$	0.3281%	67.19%	11.75%	75.9%	73.5%	71.1%	1.3%
12	0.45	0.60	0.80	$9.01 * 10^5$	0.3357%	68.75%	11.12%	76.2%	73.9%	71.3%	1.1%
13	0.45	0.60	0.80	$9.22 * 10^5$	0.3433%	70.31%	10.83%	76.3%	73.9%	71.4%	1.0%
14	0.45	0.60	0.80	$9.42 * 10^5$	0.3510%	71.88%	10.58%	76.5%	74.1%	71.5%	0.9%
15	0.45	0.60	0.80	$9.63 * 10^5$	0.3586%	73.44%	10.44%	76.6%	74.1%	71.5%	0.9%
16	0.45	0.60	0.80	$9.83 * 10^5$	0.3662%	75.00%	10.30%	76.4%	74.1%	71.5%	0.9%
17	0.45	0.60	0.80	$1.00 * 10^6$	0.3738%	76.56%	10.05%	76.5%	74.2%	71.6%	0.8%
18	0.45	0.60	0.80	$1.02 * 10^6$	0.3815%	78.12%	9.80%	76.6%	74.2%	71.6%	0.8%
19	0.45	0.60	0.80	$1.04 * 10^6$	0.3891%	79.69%	9.59%	76.7%	74.3%	71.6%	0.8%
20	0.45	0.60	0.80	$1.06 * 10^6$	0.3967%	81.25%	9.38%	76.7%	74.3%	71.6%	0.8%
21	0.45	0.60	0.80	$1.09 * 10^6$	0.4044%	82.81%	9.38%	76.7%	74.3%	71.6%	0.8%
22	0.45	0.60	0.80	$1.11 * 10^6$	0.4120%	84.38%	9.41%	76.7%	74.3%	71.6%	0.8%
23	0.45	0.60	0.80	$1.13 * 10^6$	0.4196%	85.94%	9.30%	76.8%	74.4%	71.7%	0.7%
24	0.45	0.60	0.80	$1.15 * 10^6$	0.4272%	87.50%	9.23%	76.8%	74.4%	71.7%	0.7%
25	0.45	0.60	0.80	$1.17 * 10^6$	0.4349%	89.06%	9.09%	76.8%	74.4%	71.7%	0.7%
26	0.45	0.60	0.80	$1.19 * 10^6$	0.4425%	90.62%	9.06%	76.8%	74.4%	71.7%	0.7%
27	0.45	0.60	0.80	$1.21 * 10^6$	0.4501%	92.19%	8.88%	76.9%	74.5%	71.7%	0.7%
28	0.45	0.60	0.80	$1.23 * 10^6$	0.4578%	93.75%	8.74%	77.0%	74.5%	71.8%	0.6%
29	0.45	0.60	0.80	$1.25 * 10^6$	0.4654%	95.31%	8.66%	77.0%	74.5%	71.8%	0.6%
30	0.45	0.60	0.80	$1.27 * 10^6$	0.4730%	96.88%	8.59%	77.0%	74.6%	71.8%	0.6%
31	0.45	0.60	0.80	$1.29 * 10^6$	0.4807%	98.44%	8.49%	77.1%	74.6%	71.8%	0.6%
32	0.45	0.60	0.80	$1.31 * 10^6$	0.4883%	100.00%	8.31%	77.1%	74.6%	71.9%	0.5%
33	0.45	0.60	0.80	$1.33 * 10^6$	0.4959%	101.56%	8.27%	77.1%	74.6%	71.9%	0.5%
34	0.45	0.60	0.80	$1.35 * 10^6$	0.5035%	103.12%	8.24%	77.2%	74.7%	71.9%	0.5%
35	0.45	0.60	0.80	$1.37 * 10^6$	0.5112%	104.69%	8.24%	77.2%	74.7%	71.9%	0.5%
36	0.45	0.60	0.80	$1.39 * 10^6$	0.5188%	106.25%	8.17%	77.2%	74.7%	71.9%	0.5%
37	0.45	0.60	0.80	$1.41 * 10^6$	0.5264%	107.81%	8.17%	77.2%	74.7%	71.9%	0.5%
38	0.45	0.60	0.80	$1.43 * 10^6$	0.5341%	109.38%	8.13%	77.2%	74.7%	71.9%	0.5%
39	0.45	0.60	0.80	$1.45 * 10^6$	0.5417%	110.94%	8.13%	77.2%	74.7%	71.9%	0.5%
40	0.45	0.60	0.80	$1.47 * 10^6$	0.5493%	112.50%	8.13%	77.2%	74.7%	71.9%	0.5%
41	0.45	0.60	0.80	$1.50 * 10^6$	0.5569%	114.06%	8.10%	77.2%	74.7%	71.9%	0.5%
42	0.45	0.60	0.80	$1.52 * 10^6$	0.5646%	115.62%	8.03%	77.2%	74.7%	71.9%	0.5%
43	0.45	0.60	0.80	$1.54 * 10^6$	0.5722%	117.19%	7.99%	77.3%	74.7%	71.9%	0.5%
44	0.45	0.60	0.80	$1.56 * 10^6$	0.5798%	118.75%	7.99%	77.3%	74.7%	71.9%	0.5%
45	0.45	0.60	0.80	$1.58 * 10^6$	0.5875%	120.31%	7.95%	77.3%	74.7%	71.9%	0.5%
46	0.45	0.60	0.80	$1.60 * 10^6$	0.5951%	121.88%	7.92%	77.3%	74.7%	71.9%	0.5%
47	0.45	0.60	0.80	$1.62 * 10^6$	0.6027%	123.44%	7.92%	77.3%	74.7%	71.9%	0.5%
48	0.45	0.60	0.80	$1.64 * 10^6$	0.6104%	125.00%	7.92%	77.3%	74.7%	71.9%	0.5%

* $\mathcal{F}$ compared to the SML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary SML-CPCA ω_{256}.

Table A.26: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional real SML-CPCA comparison employing both sessions of the PolyU dataset.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.46	0.60	0.75	0.7	$3.48 * 10^5$	0.1297%	26.56%	37.93%	59.3%	57.8%	56.9%	15.5%
2	0.45	0.60	0.75	0.7	$3.69 * 10^5$	0.1373%	28.12%	27.81%	66.6%	65.1%	62.6%	9.8%
3	0.45	0.60	0.80	0.7	$3.89 * 10^5$	0.1450%	29.69%	23.54%	70.0%	68.6%	65.2%	7.2%
4	0.45	0.60	0.80	0.7	$4.10 * 10^5$	0.1526%	31.25%	20.38%	72.1%	70.1%	67.0%	5.4%
5	0.45	0.60	0.80	0.7	$4.30 * 10^5$	0.1602%	32.81%	18.08%	73.5%	71.0%	68.4%	4.0%
6	0.45	0.60	0.80	0.7	$4.51 * 10^5$	0.1678%	34.38%	16.34%	74.9%	72.2%	69.5%	2.9%
7	0.45	0.60	0.80	0.7	$4.71 * 10^5$	0.1755%	35.94%	15.27%	75.5%	72.6%	69.9%	2.5%
8	0.45	0.60	0.80	0.7	$4.92 * 10^5$	0.1831%	37.50%	13.67%	76.1%	73.2%	70.4%	2.0%
9	0.45	0.60	0.80	0.7	$5.12 * 10^5$	0.1907%	39.06%	13.21%	76.5%	73.5%	70.6%	1.8%
10	0.45	0.60	0.80	0.7	$5.32 * 10^5$	0.1984%	40.62%	12.50%	77.0%	73.8%	70.9%	1.5%
11	0.45	0.60	0.80	0.7	$5.53 * 10^5$	0.2060%	42.19%	12.00%	77.1%	73.8%	71.2%	1.2%
12	0.45	0.60	0.80	0.7	$5.73 * 10^5$	0.2136%	43.75%	11.40%	77.4%	74.1%	71.4%	1.0%
13	0.45	0.60	0.80	0.7	$5.94 * 10^5$	0.2213%	45.31%	11.08%	77.5%	74.2%	71.5%	0.9%
14	0.45	0.60	0.80	0.7	$6.14 * 10^5$	0.2289%	46.88%	10.94%	77.7%	74.3%	71.6%	0.8%
15	0.45	0.60	0.80	0.7	$6.35 * 10^5$	0.2365%	48.44%	10.51%	77.8%	74.4%	71.6%	0.8%
16	0.45	0.60	0.80	0.7	$6.55 * 10^5$	0.2441%	50.00%	10.48%	77.8%	74.4%	71.6%	0.8%
17	0.45	0.60	0.80	0.7	$6.76 * 10^5$	0.2518%	51.56%	10.30%	77.3%	74.5%	71.7%	0.7%
18	0.45	0.60	0.80	0.7	$6.96 * 10^5$	0.2594%	53.12%	9.98%	77.4%	74.5%	71.7%	0.7%
19	0.45	0.60	0.80	0.7	$7.17 * 10^5$	0.2670%	54.69%	9.73%	77.5%	74.6%	71.8%	0.6%
20	0.45	0.60	0.80	0.7	$7.37 * 10^5$	0.2747%	56.25%	9.62%	77.5%	74.6%	71.8%	0.6%
21	0.45	0.60	0.80	0.7	$7.58 * 10^5$	0.2823%	57.81%	9.48%	77.6%	74.7%	71.9%	0.5%
22	0.45	0.60	0.80	0.7	$7.78 * 10^5$	0.2899%	59.38%	9.48%	77.6%	74.7%	71.9%	0.5%
23	0.45	0.60	0.80	0.7	$7.99 * 10^5$	0.2975%	60.94%	9.34%	77.6%	74.7%	71.9%	0.5%
24	0.45	0.60	0.80	0.7	$8.19 * 10^5$	0.3052%	62.50%	9.27%	77.6%	74.7%	71.9%	0.5%
25	0.45	0.60	0.80	0.7	$8.40 * 10^5$	0.3128%	64.06%	9.13%	77.6%	74.8%	71.9%	0.5%
26	0.45	0.60	0.80	0.7	$8.60 * 10^5$	0.3204%	65.62%	9.09%	77.6%	74.8%	71.9%	0.5%
27	0.45	0.60	0.80	0.7	$8.81 * 10^5$	0.3281%	67.19%	8.95%	77.7%	74.8%	71.9%	0.5%
28	0.45	0.60	0.80	0.7	$9.01 * 10^5$	0.3357%	68.75%	8.81%	77.7%	74.8%	72.0%	0.4%
29	0.45	0.60	0.80	0.7	$9.22 * 10^5$	0.3433%	70.31%	8.77%	77.7%	74.8%	72.0%	0.4%
30	0.45	0.60	0.80	0.7	$9.42 * 10^5$	0.3510%	71.88%	8.63%	77.7%	74.9%	72.0%	0.4%
31	0.45	0.60	0.80	0.7	$9.63 * 10^5$	0.3586%	73.44%	8.56%	77.8%	74.9%	72.1%	0.3%
32	0.45	0.60	0.80	0.7	$9.83 * 10^5$	0.3662%	75.00%	8.38%	77.8%	74.9%	72.1%	0.3%
33	0.45	0.60	0.80	0.7	$1.00 * 10^6$	0.3738%	76.56%	8.35%	77.8%	74.9%	72.1%	0.3%
34	0.45	0.60	0.80	0.7	$1.02 * 10^6$	0.3815%	78.12%	8.31%	77.8%	74.9%	72.1%	0.3%
35	0.45	0.60	0.80	0.7	$1.04 * 10^6$	0.3891%	79.69%	8.31%	77.8%	74.9%	72.1%	0.3%
36	0.45	0.60	0.80	0.7	$1.06 * 10^6$	0.3967%	81.25%	8.24%	77.9%	75.0%	72.1%	0.3%
37	0.45	0.60	0.80	0.7	$1.09 * 10^6$	0.4044%	82.81%	8.24%	77.9%	75.0%	72.1%	0.3%
38	0.45	0.60	0.80	0.7	$1.11 * 10^6$	0.4120%	84.38%	8.20%	77.9%	75.0%	72.1%	0.3%
39	0.45	0.60	0.80	0.7	$1.13 * 10^6$	0.4196%	85.94%	8.17%	77.9%	75.0%	72.1%	0.3%
40	0.45	0.60	0.80	0.7	$1.15 * 10^6$	0.4272%	87.50%	8.17%	77.9%	75.0%	72.1%	0.3%
41	0.45	0.60	0.80	0.7	$1.17 * 10^6$	0.4349%	89.06%	8.13%	77.9%	75.0%	72.1%	0.3%
42	0.45	0.60	0.80	0.7	$1.19 * 10^6$	0.4425%	90.62%	8.10%	77.9%	75.0%	72.1%	0.3%
43	0.45	0.60	0.80	0.7	$1.21 * 10^6$	0.4501%	92.19%	8.03%	77.9%	75.0%	72.1%	0.3%
44	0.45	0.60	0.80	0.7	$1.23 * 10^6$	0.4578%	93.75%	7.99%	77.9%	75.0%	72.1%	0.3%
45	0.45	0.60	0.80	0.7	$1.25 * 10^6$	0.4654%	95.31%	7.99%	77.9%	75.0%	72.1%	0.3%
46	0.45	0.60	0.80	0.7	$1.27 * 10^6$	0.4730%	96.88%	7.95%	77.9%	75.0%	72.1%	0.3%
47	0.45	0.60	0.80	0.7	$1.29 * 10^6$	0.4807%	98.44%	7.95%	77.9%	75.0%	72.1%	0.3%
48	0.45	0.60	0.80	0.7	$1.31 * 10^6$	0.4883%	100.00%	7.95%	77.9%	75.0%	72.1%	0.3%

* F compared to the SML ω_{256}.

† F compared to the naïve binary SML-CPCA ω_{256}.

Table A.27: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional binary SML-CPCA comparison and masking out most common set bits employing both sessions of the PolyU dataset.

Approach	Workload		Ident.			Verif.	Recognition Performance
	ω_{256}	F	$TP_{0.5}$	$TP_{0.1}$	TP_0	**EER**	**Loss** (TP_0)
SML-Baseline	2.68×10^8	–	89.9%	88.4%	88.1%	2.3%	–
Binary SML-CPCA (MT= 0.75)	1.31×10^6	0.4883%	88.3%	86.3%	86%	2.4%	2.1%

Table A.28: Summary of achieved TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for the SMR biometric identification system using session 2 of the PolyU dataset and their corresponding verification EER with applied feature reduction approaches, ordered by TP_0.

Bloom filter			SMR	Workload			Recognition Performance			
t	$\mathcal{W}$	$\mathcal{H}$	MT	ω_{256}	F_B*	F_R†	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	5	5	0.1	$4.51 * 10^5$	0.1679%	34.39%	81.2%	80.9%	80.5%	7.7%
2	4	4	0.2	$3.74 * 10^5$	0.1392%	28.52%	84.8%	84.4%	83.8%	4.3%
3	4	4	0.2	$3.94 * 10^5$	0.1469%	30.08%	86.2%	85.4%	85.2%	2.9%
4	4	4	0.5	$4.15 * 10^5$	0.1545%	31.64%	87.1%	86.5%	83.8%	4.3%
5	4	5	0.4	$6.93 * 10^5$	0.2583%	52.89%	87.7%	87.0%	84.3%	3.8%
6	7	4	0.1	$2.63 * 10^5$	0.0978%	20.03%	86.2%	85.5%	85.4%	2.7%
7	7	4	0.1	$2.74 * 10^5$	0.1022%	20.93%	86.6%	85.9%	85.9%	2.3%
8	7	4	0.1	$2.86 * 10^5$	0.1065%	21.82%	86.9%	86.2%	86.1%	2.0%
9	7	4	0.1	$2.98 * 10^5$	0.1109%	22.71%	87.4%	86.6%	86.5%	1.6%
10	7	4	0.1	$3.09 * 10^5$	0.1153%	23.60%	87.7%	86.8%	86.7%	1.4%
11	7	5	0.6	$5.11 * 10^5$	0.1902%	38.96%	87.2%	86.9%	86.4%	1.7%
12	8	4	0.2	$2.92 * 10^5$	0.1087%	22.27%	87.6%	86.9%	86.6%	1.5%
13	7	5	0.6	$5.48 * 10^5$	0.2042%	41.82%	87.7%	87.3%	86.8%	1.3%
14	7	5	0.6	$5.67 * 10^5$	0.2112%	43.25%	87.7%	87.4%	87.0%	1.2%
15	7	5	0.6	$5.86 * 10^5$	0.2181%	44.68%	88.0%	87.6%	87.1%	1.0%
16	7	5	0.6	$6.04 * 10^5$	0.2251%	46.10%	88.0%	87.7%	87.3%	0.9%
17	7	5	0.6	$6.23 * 10^5$	0.2321%	47.53%	88.3%	88.0%	87.5%	0.6%
18	7	5	0.6	$6.42 * 10^5$	0.2391%	48.96%	88.3%	88.0%	87.5%	0.6%
19	7	5	0.6	$6.60 * 10^5$	0.2460%	50.39%	88.4%	88.0%	87.6%	0.5%
20	7	5	0.6	$6.79 * 10^5$	0.2530%	51.82%	88.4%	88.0%	87.6%	0.5%
21	7	5	0.6	$6.98 * 10^5$	0.2600%	53.25%	88.4%	88.1%	87.6%	0.5%
22	7	5	0.6	$7.17 * 10^5$	0.2670%	54.68%	88.5%	88.2%	87.7%	0.4%
23	7	5	0.6	$7.35 * 10^5$	0.2739%	56.10%	88.4%	88.1%	87.6%	0.5%
24	7	5	0.6	$7.54 * 10^5$	0.2809%	57.53%	88.4%	88.1%	87.6%	0.5%
25	7	5	0.6	$7.73 * 10^5$	0.2879%	58.96%	88.6%	88.3%	87.7%	0.4%
26	7	5	0.6	$7.92 * 10^5$	0.2949%	60.39%	88.6%	88.3%	87.7%	0.4%
27	7	5	0.6	$8.10 * 10^5$	0.3019%	61.82%	88.6%	88.3%	87.7%	0.4%
28	7	5	0.6	$8.29 * 10^5$	0.3088%	63.25%	88.6%	88.3%	87.7%	0.4%
29	7	5	0.6	$8.48 * 10^5$	0.3158%	64.68%	88.5%	88.2%	87.7%	0.4%
30	7	5	0.6	$8.66 * 10^5$	0.3228%	66.10%	88.7%	88.4%	87.7%	0.4%
31	7	5	0.6	$8.85 * 10^5$	0.3298%	67.53%	88.7%	88.4%	87.7%	0.4%
32	7	5	0.6	$9.04 * 10^5$	0.3367%	68.96%	88.7%	88.4%	87.7%	0.4%
33	7	5	0.6	$9.23 * 10^5$	0.3437%	70.39%	88.7%	88.4%	87.7%	0.4%
34	7	5	0.6	$9.41 * 10^5$	0.3507%	71.82%	88.7%	88.4%	87.7%	0.4%
35	7	5	0.6	$9.60 * 10^5$	0.3577%	73.25%	88.6%	88.3%	87.7%	0.4%
36	7	5	0.6	$9.79 * 10^5$	0.3646%	74.68%	88.6%	88.3%	87.7%	0.4%
37	7	5	0.6	$9.98 * 10^5$	0.3716%	76.10%	88.6%	88.3%	87.7%	0.4%
38	7	5	0.6	$1.02 * 10^6$	0.3786%	77.53%	88.6%	88.3%	87.7%	0.4%
39	7	5	0.6	$1.03 * 10^6$	0.3856%	78.96%	88.6%	88.3%	87.7%	0.4%
40	7	5	0.6	$1.05 * 10^6$	0.3925%	80.39%	88.6%	88.3%	87.7%	0.4%
41	7	5	0.6	$1.07 * 10^6$	0.3995%	81.82%	88.6%	88.3%	87.7%	0.4%
42	7	5	0.6	$1.09 * 10^6$	0.4065%	83.25%	88.6%	88.3%	87.7%	0.4%
43	7	5	0.6	$1.11 * 10^6$	0.4135%	84.68%	88.6%	88.3%	87.7%	0.4%
44	7	5	0.6	$1.13 * 10^6$	0.4204%	86.10%	88.6%	88.3%	87.7%	0.4%
45	7	5	0.6	$1.15 * 10^6$	0.4274%	87.53%	88.6%	88.3%	87.7%	0.4%
46	7	5	0.6	$1.17 * 10^6$	0.4344%	88.96%	88.6%	88.3%	87.7%	0.4%
47	7	5	0.6	$1.18 * 10^6$	0.4414%	90.39%	88.6%	88.3%	87.7%	0.4%
48	7	5	0.6	$1.20 * 10^6$	0.4483%	91.82%	88.6%	88.3%	87.7%	0.4%

* F compared to the SML ω_{256}.

† F compared to the naïve binary SML-CPCA ω_{256}.

Table A.29: Best achieved binary SML-CPCA Bloom filter TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$, with a final real-valued SML-CPCA comparison and tree pre-selection of $t = 1, \ldots, 48$ trees at $\mathcal{T} = 64$ using for session 2 of the PolyU dataset.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.50	0.62	0.70	$3.48 * 10^5$	0.1297%	26.56%	27.34%	71.3%	71.2%	70.9%	17.3%
2	0.52	0.62	0.70	$3.69 * 10^5$	0.1373%	28.12%	17.34%	80.4%	80.0%	79.5%	8.6%
3	0.51	0.61	0.70	$3.89 * 10^5$	0.1450%	29.69%	13.44%	84.1%	83.3%	82.7%	5.5%
4	0.52	0.64	0.75	$4.10 * 10^5$	0.1526%	31.25%	11.88%	85.4%	84.8%	84.5%	3.7%
5	0.52	0.62	0.70	$4.30 * 10^5$	0.1602%	32.81%	10.62%	86.4%	85.5%	85.3%	2.8%
6	0.51	0.61	0.75	$4.51 * 10^5$	0.1678%	34.38%	9.06%	86.7%	86.5%	85.6%	2.5%
7	0.51	0.61	0.75	$4.71 * 10^5$	0.1755%	35.94%	7.97%	87.4%	87.2%	86.3%	1.8%
8	0.51	0.61	0.75	$4.92 * 10^5$	0.1831%	37.50%	7.81%	87.5%	87.3%	86.4%	1.7%
9	0.50	0.60	0.70	$5.12 * 10^5$	0.1907%	39.06%	7.50%	87.6%	87.2%	86.4%	1.7%
10	0.43	0.61	0.75	$5.32 * 10^5$	0.1984%	40.62%	5.78%	87.7%	86.7%	86.4%	1.7%
11	0.43	0.61	0.75	$5.53 * 10^5$	0.2060%	42.19%	5.47%	87.8%	86.9%	86.6%	1.6%
12	0.43	0.61	0.75	$5.73 * 10^5$	0.2136%	43.75%	5.00%	87.8%	87.0%	86.7%	1.4%
13	0.43	0.61	0.75	$5.94 * 10^5$	0.2213%	45.31%	4.92%	87.9%	87.0%	86.7%	1.4%
14	0.45	0.64	0.70	$6.14 * 10^5$	0.2289%	46.88%	4.53%	88.7%	87.5%	86.7%	1.4%
15	0.44	0.63	0.70	$6.35 * 10^5$	0.2365%	48.44%	4.45%	88.9%	88.1%	87.0%	1.2%
16	0.43	0.61	0.75	$6.55 * 10^5$	0.2441%	50.00%	4.38%	88.4%	87.4%	87.1%	1.0%
17	0.43	0.61	0.75	$6.76 * 10^5$	0.2518%	51.56%	4.22%	88.4%	87.4%	87.1%	1.0%
18	0.43	0.61	0.75	$6.96 * 10^5$	0.2594%	53.12%	4.14%	88.4%	87.5%	87.2%	0.9%
19	0.43	0.61	0.75	$7.17 * 10^5$	0.2670%	54.69%	3.91%	88.6%	87.7%	87.3%	0.8%
20	0.43	0.61	0.75	$7.37 * 10^5$	0.2747%	56.25%	3.91%	88.6%	87.7%	87.3%	0.8%
21	0.43	0.61	0.75	$7.58 * 10^5$	0.2823%	57.81%	3.91%	88.6%	87.7%	87.3%	0.8%
22	0.43	0.61	0.75	$7.78 * 10^5$	0.2899%	59.38%	3.83%	88.7%	87.7%	87.4%	0.7%
23	0.43	0.61	0.75	$7.99 * 10^5$	0.2975%	60.94%	3.83%	88.7%	87.7%	87.4%	0.7%
24	0.43	0.61	0.75	$8.19 * 10^5$	0.3052%	62.50%	3.83%	88.7%	87.7%	87.4%	0.7%
25	0.43	0.61	0.75	$8.40 * 10^5$	0.3128%	64.06%	3.91%	88.7%	87.7%	87.4%	0.7%
26	0.43	0.61	0.75	$8.60 * 10^5$	0.3204%	65.62%	3.83%	88.7%	87.7%	87.4%	0.7%
27	0.43	0.61	0.75	$8.81 * 10^5$	0.3281%	67.19%	3.83%	88.7%	87.7%	87.4%	0.7%
28	0.43	0.61	0.75	$9.01 * 10^5$	0.3357%	68.75%	3.83%	88.7%	87.7%	87.4%	0.7%
29	0.44	0.62	0.75	$9.22 * 10^5$	0.3433%	70.31%	4.06%	89.1%	87.6%	87.5%	0.6%
30	0.44	0.62	0.75	$9.42 * 10^5$	0.3510%	71.88%	3.98%	89.1%	87.6%	87.5%	0.6%
31	0.44	0.62	0.75	$9.63 * 10^5$	0.3586%	73.44%	3.98%	88.8%	87.6%	87.5%	0.6%
32	0.44	0.62	0.75	$9.83 * 10^5$	0.3662%	75.00%	3.98%	88.8%	87.6%	87.5%	0.6%
33	0.44	0.62	0.75	$1.00 * 10^6$	0.3738%	76.56%	3.98%	88.8%	87.6%	87.5%	0.6%
34	0.44	0.62	0.75	$1.02 * 10^6$	0.3815%	78.12%	3.91%	88.8%	87.6%	87.5%	0.6%
35	0.44	0.62	0.75	$1.04 * 10^6$	0.3891%	79.69%	3.91%	88.8%	87.6%	87.5%	0.6%
36	0.44	0.62	0.75	$1.06 * 10^6$	0.3967%	81.25%	3.91%	88.8%	87.6%	87.5%	0.6%
37	0.44	0.62	0.75	$1.09 * 10^6$	0.4044%	82.81%	3.91%	88.8%	87.6%	87.5%	0.6%
38	0.44	0.62	0.75	$1.11 * 10^6$	0.4120%	84.38%	3.91%	88.8%	87.6%	87.5%	0.6%
39	0.44	0.62	0.75	$1.13 * 10^6$	0.4196%	85.94%	3.91%	88.8%	87.6%	87.5%	0.6%
40	0.44	0.62	0.75	$1.15 * 10^6$	0.4272%	87.50%	3.91%	88.8%	87.6%	87.5%	0.6%
41	0.44	0.62	0.75	$1.17 * 10^6$	0.4349%	89.06%	3.91%	88.8%	87.6%	87.5%	0.6%
42	0.54	0.64	0.75	$1.19 * 10^6$	0.4425%	90.62%	4.06%	88.5%	87.7%	87.5%	0.6%
43	0.54	0.64	0.75	$1.21 * 10^6$	0.4501%	92.19%	4.14%	88.5%	87.7%	87.5%	0.6%
44	0.54	0.64	0.75	$1.23 * 10^6$	0.4578%	93.75%	4.14%	88.5%	87.7%	87.5%	0.6%
45	0.54	0.64	0.75	$1.25 * 10^6$	0.4654%	95.31%	4.14%	88.5%	87.7%	87.5%	0.6%
46	0.54	0.64	0.75	$1.27 * 10^6$	0.4730%	96.88%	4.14%	88.5%	87.7%	87.5%	0.6%
47	0.54	0.64	0.75	$1.29 * 10^6$	0.4807%	98.44%	4.14%	88.5%	87.7%	87.5%	0.6%
48	0.54	0.64	0.75	$1.31 * 10^6$	0.4883%	100.00%	4.06%	88.5%	87.7%	87.5%	0.6%

* $\mathcal{F}$ compared to the SML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary SML-CPCA ω_{256}.

Table A.30: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with session 2 of the PolyU dataset.

	SMR			Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	ω_{256}	F_B*	F_R†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 loss to PSML-Baseline
1	0.54	0.63	0.75	$6.76 * 10^5$	0.2518%	51.56%	27.50%	71.4%	70.8%	70.5%	17.7%
2	0.54	0.64	0.75	$6.96 * 10^5$	0.2594%	53.12%	17.66%	80.4%	80.0%	79.5%	8.6%
3	0.54	0.64	0.75	$7.17 * 10^5$	0.2670%	54.69%	13.28%	84.4%	83.2%	80.6%	7.5%
4	0.43	0.61	0.75	$7.37 * 10^5$	0.2747%	56.25%	12.50%	84.8%	83.5%	82.6%	5.5%
5	0.45	0.63	0.75	$7.58 * 10^5$	0.2823%	57.81%	10.08%	86.7%	85.7%	84.2%	3.9%
6	0.43	0.61	0.75	$7.78 * 10^5$	0.2899%	59.38%	9.38%	87.2%	86.0%	85.1%	3.0%
7	0.43	0.60	0.80	$7.99 * 10^5$	0.2975%	60.94%	8.44%	87.7%	86.6%	85.7%	2.4%
8	0.43	0.60	0.80	$8.19 * 10^5$	0.3052%	62.50%	7.58%	88.4%	87.2%	86.2%	1.9%
9	0.43	0.61	0.75	$8.40 * 10^5$	0.3128%	64.06%	6.41%	89.0%	87.8%	86.8%	1.3%
10	0.43	0.61	0.75	$8.60 * 10^5$	0.3204%	65.62%	5.78%	89.4%	88.1%	87.1%	1.0%
11	0.43	0.61	0.75	$8.81 * 10^5$	0.3281%	67.19%	5.47%	89.6%	88.3%	87.3%	0.9%
12	0.43	0.61	0.75	$9.01 * 10^5$	0.3357%	68.75%	5.16%	89.8%	88.5%	87.5%	0.6%
13	0.43	0.61	0.75	$9.22 * 10^5$	0.3433%	70.31%	5.08%	89.9%	88.5%	87.5%	0.6%
14	0.42	0.61	0.75	$9.42 * 10^5$	0.3510%	71.88%	5.08%	89.7%	88.9%	87.7%	0.5%
15	0.43	0.62	0.75	$9.63 * 10^5$	0.3586%	73.44%	4.69%	90.2%	88.9%	87.7%	0.5%
16	0.42	0.60	0.75	$9.83 * 10^5$	0.3662%	75.00%	4.84%	90.0%	89.0%	87.7%	0.4%
17	0.43	0.61	0.75	$1.00 * 10^6$	0.3738%	76.56%	4.14%	90.4%	88.9%	87.9%	0.2%
18	0.43	0.61	0.75	$1.02 * 10^6$	0.3815%	78.12%	3.91%	90.5%	89.0%	88.0%	0.2%
19	0.43	0.61	0.75	$1.04 * 10^6$	0.3891%	79.69%	3.83%	90.5%	89.1%	88.0%	0.1%
20	0.43	0.61	0.75	$1.06 * 10^6$	0.3967%	81.25%	3.75%	90.6%	89.1%	88.1%	0.0%
21	0.43	0.61	0.75	$1.09 * 10^6$	0.4044%	82.81%	3.75%	90.6%	89.1%	88.1%	0.0%
22	0.43	0.61	0.80	$1.11 * 10^6$	0.4120%	84.38%	3.83%	90.7%	89.2%	88.2%	+0.1%
23	0.43	0.61	0.80	$1.13 * 10^6$	0.4196%	85.94%	3.91%	90.7%	89.2%	88.2%	+0.1%
24	0.43	0.61	0.85	$1.15 * 10^6$	0.4272%	87.50%	4.38%	89.5%	88.4%	88.1%	0.0%
25	0.43	0.61	0.85	$1.17 * 10^6$	0.4349%	89.06%	4.38%	89.5%	88.4%	88.1%	0.0%
26	0.43	0.61	0.85	$1.19 * 10^6$	0.4425%	90.62%	4.30%	89.5%	88.4%	88.1%	0.0%
27	0.43	0.61	0.85	$1.21 * 10^6$	0.4501%	92.19%	4.14%	89.5%	88.4%	88.2%	+0.1%
28	0.43	0.61	0.85	$1.23 * 10^6$	0.4578%	93.75%	4.14%	89.5%	88.4%	88.2%	+0.1%
29	0.43	0.61	0.85	$1.25 * 10^6$	0.4654%	95.31%	3.91%	89.5%	88.4%	88.2%	+0.1%
30	0.43	0.61	0.85	$1.27 * 10^6$	0.4730%	96.88%	3.83%	89.5%	88.4%	88.2%	+0.1%
31	0.43	0.61	0.85	$1.29 * 10^6$	0.4807%	98.44%	3.98%	89.5%	88.4%	88.2%	+0.1%
32	0.43	0.61	0.85	$1.31 * 10^6$	0.4883%	100.00%	3.83%	89.6%	88.5%	88.3%	+0.2%
33	0.43	0.61	0.85	$1.33 * 10^6$	0.4959%	101.56%	3.83%	89.6%	88.5%	88.3%	+0.2%
34	0.43	0.61	0.85	$1.35 * 10^6$	0.5035%	103.12%	3.98%	89.6%	88.5%	88.3%	+0.2%
35	0.43	0.61	0.85	$1.37 * 10^6$	0.5112%	104.69%	3.83%	89.6%	88.5%	88.3%	+0.2%
36	0.43	0.61	0.90	$1.39 * 10^6$	0.5188%	106.25%	3.75%	89.6%	88.5%	88.3%	+0.2%
37	0.43	0.61	0.90	$1.41 * 10^6$	0.5264%	107.81%	3.75%	89.6%	88.5%	88.3%	+0.2%
38	0.43	0.61	0.90	$1.43 * 10^6$	0.5341%	109.38%	3.75%	89.6%	88.5%	88.3%	+0.2%
39	0.43	0.61	0.85	$1.45 * 10^6$	0.5417%	110.94%	3.91%	89.6%	88.5%	88.3%	+0.2%
40	0.43	0.61	0.85	$1.47 * 10^6$	0.5493%	112.50%	3.91%	89.6%	88.5%	88.3%	+0.2%
41	0.43	0.61	0.90	$1.50 * 10^6$	0.5569%	114.06%	3.75%	89.6%	88.5%	88.3%	+0.2%
42	0.43	0.61	0.90	$1.52 * 10^6$	0.5646%	115.62%	3.75%	89.6%	88.5%	88.3%	+0.2%
43	0.43	0.61	0.85	$1.54 * 10^6$	0.5722%	117.19%	3.91%	89.6%	88.5%	88.3%	+0.2%
44	0.43	0.61	0.85	$1.56 * 10^6$	0.5798%	118.75%	3.91%	89.6%	88.5%	88.3%	+0.2%
45	0.43	0.61	0.85	$1.58 * 10^6$	0.5875%	120.31%	3.91%	89.6%	88.5%	88.3%	+0.2%
46	0.43	0.61	0.85	$1.60 * 10^6$	0.5951%	121.88%	3.91%	89.6%	88.5%	88.3%	+0.2%
47	0.43	0.61	0.85	$1.62 * 10^6$	0.6027%	123.44%	3.91%	89.6%	88.5%	88.3%	+0.2%
48	0.43	0.61	0.85	$1.64 * 10^6$	0.6104%	125.00%	3.91%	89.6%	88.5%	88.3%	+0.2%

* F compared to the SML ω_{256}.

† F compared to the naïve binary SML-CPCA ω_{256}.

Table A.31: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree pre-selection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional real PSML-CPCA comparison for session 2 of the PolyU dataset.

	SMR				Workload			Recognition Performance				
t	λ_{max}	λ_{max}^{CPCA}	MT	**Bit Threshold**	ω_{256}	$\mathcal{F}_B$*	$\mathcal{F}_R$†	PER	$TP_{0.5}$	$TP_{0.1}$	TP_0	TP_0 **loss to PSML-Baseline**
1	0.53	0.61	0.75	0.7	$3.48 * 10^5$	0.1297%	26.56%	24.06%	74.1%	73.9%	73.6%	16.8%
2	0.53	0.60	0.75	0.7	$3.69 * 10^5$	0.1373%	28.12%	15.00%	82.7%	81.7%	81.6%	8.8%
3	0.48	0.60	0.75	0.5	$3.89 * 10^5$	0.1450%	29.69%	12.27%	84.9%	84.5%	84.0%	6.4%
4	0.48	0.60	0.75	0.5	$4.10 * 10^5$	0.1526%	31.25%	10.23%	86.1%	85.5%	85.1%	5.3%
5	0.43	0.60	0.75	0.5	$4.30 * 10^5$	0.1602%	32.81%	9.30%	87.4%	86.3%	86.2%	4.2%
6	0.43	0.60	0.80	0.6	$4.51 * 10^5$	0.1678%	34.38%	7.58%	88.0%	87.3%	87.0%	3.4%
7	0.43	0.60	0.75	0.7	$4.71 * 10^5$	0.1755%	35.94%	6.72%	89.2%	87.9%	87.8%	2.6%
8	0.43	0.60	0.75	0.6	$4.92 * 10^5$	0.1831%	37.50%	6.02%	89.5%	88.2%	88.0%	2.4%
9	0.43	0.60	0.75	0.7	$5.12 * 10^5$	0.1907%	39.06%	5.55%	89.9%	88.4%	88.4%	2.0%
10	0.43	0.60	0.75	0.7	$5.32 * 10^5$	0.1984%	40.62%	5.39%	89.4%	88.5%	88.4%	2.0%
11	0.43	0.60	0.75	0.7	$5.53 * 10^5$	0.2060%	42.19%	4.92%	89.8%	88.9%	88.8%	1.6%
12	0.43	0.60	0.75	0.7	$5.73 * 10^5$	0.2136%	43.75%	4.69%	89.8%	89.0%	88.9%	1.5%
13	0.43	0.60	0.75	0.6	$5.94 * 10^5$	0.2213%	45.31%	4.61%	89.6%	89.0%	89.0%	1.4%
14	0.43	0.60	0.75	0.6	$6.14 * 10^5$	0.2289%	46.88%	4.53%	89.6%	89.0%	89.0%	1.4%
15	0.43	0.60	0.75	0.6	$6.35 * 10^5$	0.2365%	48.44%	4.22%	89.8%	89.2%	89.2%	1.2%
16	0.43	0.60	0.75	0.6	$6.55 * 10^5$	0.2441%	50.00%	4.30%	89.8%	89.2%	89.2%	1.2%
17	0.43	0.60	0.75	0.6	$6.76 * 10^5$	0.2518%	51.56%	4.14%	89.8%	89.2%	89.2%	1.2%
18	0.43	0.60	0.75	0.6	$6.96 * 10^5$	0.2594%	53.12%	4.06%	89.8%	89.2%	89.2%	1.2%
19	0.43	0.60	0.75	0.6	$7.17 * 10^5$	0.2670%	54.69%	3.83%	89.9%	89.3%	89.3%	1.1%
20	0.43	0.60	0.75	0.6	$7.37 * 10^5$	0.2747%	56.25%	3.59%	90.1%	89.4%	89.4%	1.0%
21	0.43	0.60	0.75	0.6	$7.58 * 10^5$	0.2823%	57.81%	3.36%	90.3%	89.5%	89.5%	0.9%
22	0.43	0.60	0.75	0.6	$7.78 * 10^5$	0.2899%	59.38%	3.36%	90.3%	89.5%	89.5%	0.9%
23	0.43	0.60	0.75	0.6	$7.99 * 10^5$	0.2975%	60.94%	3.36%	90.3%	89.5%	89.5%	0.9%
24	0.43	0.60	0.75	0.6	$8.19 * 10^5$	0.3052%	62.50%	3.36%	90.3%	89.5%	89.5%	0.9%
25	0.43	0.60	0.75	0.6	$8.40 * 10^5$	0.3128%	64.06%	3.36%	90.3%	89.5%	89.5%	0.9%
26	0.43	0.60	0.75	0.6	$8.60 * 10^5$	0.3204%	65.62%	3.28%	90.4%	89.6%	89.6%	0.8%
27	0.43	0.60	0.75	0.6	$8.81 * 10^5$	0.3281%	67.19%	3.36%	90.4%	89.6%	89.6%	0.8%
28	0.43	0.60	0.75	0.6	$9.01 * 10^5$	0.3357%	68.75%	3.36%	90.4%	89.6%	89.6%	0.8%
29	0.43	0.60	0.75	0.6	$9.22 * 10^5$	0.3433%	70.31%	3.36%	90.4%	89.6%	89.6%	0.8%
30	0.43	0.60	0.75	0.6	$9.42 * 10^5$	0.3510%	71.88%	3.36%	90.4%	89.6%	89.6%	0.8%
31	0.43	0.60	0.75	0.6	$9.63 * 10^5$	0.3586%	73.44%	3.28%	90.5%	89.6%	89.6%	0.8%
32	0.43	0.60	0.75	0.6	$9.83 * 10^5$	0.3662%	75.00%	3.28%	90.5%	89.6%	89.6%	0.8%
33	0.43	0.60	0.75	0.6	$1.00 * 10^6$	0.3738%	76.56%	3.28%	90.5%	89.6%	89.6%	0.8%
34	0.43	0.60	0.75	0.6	$1.02 * 10^6$	0.3815%	78.12%	3.28%	90.5%	89.6%	89.6%	0.8%
35	0.43	0.60	0.75	0.6	$1.04 * 10^6$	0.3891%	79.69%	3.28%	90.5%	89.6%	89.6%	0.8%
36	0.43	0.60	0.75	0.6	$1.06 * 10^6$	0.3967%	81.25%	3.28%	90.2%	89.6%	89.6%	0.8%
37	0.43	0.60	0.75	0.6	$1.09 * 10^6$	0.4044%	82.81%	3.28%	90.2%	89.6%	89.6%	0.8%
38	0.43	0.60	0.75	0.6	$1.11 * 10^6$	0.4120%	84.38%	3.28%	90.2%	89.6%	89.6%	0.8%
39	0.43	0.60	0.75	0.6	$1.13 * 10^6$	0.4196%	85.94%	3.28%	90.2%	89.6%	89.6%	0.8%
40	0.43	0.60	0.75	0.6	$1.15 * 10^6$	0.4272%	87.50%	3.28%	90.2%	89.6%	89.6%	0.8%
41	0.43	0.60	0.75	0.6	$1.17 * 10^6$	0.4349%	89.06%	3.20%	90.2%	89.6%	89.6%	0.8%
42	0.43	0.60	0.75	0.6	$1.19 * 10^6$	0.4425%	90.62%	3.20%	90.2%	89.6%	89.6%	0.8%
43	0.43	0.60	0.75	0.6	$1.21 * 10^6$	0.4501%	92.19%	3.20%	90.2%	89.6%	89.6%	0.8%
44	0.43	0.60	0.75	0.6	$1.23 * 10^6$	0.4578%	93.75%	3.20%	90.2%	89.6%	89.6%	0.8%
45	0.43	0.60	0.75	0.6	$1.25 * 10^6$	0.4654%	95.31%	3.20%	90.2%	89.6%	89.6%	0.8%
46	0.43	0.60	0.75	0.6	$1.27 * 10^6$	0.4730%	96.88%	3.20%	90.2%	89.6%	89.6%	0.8%
47	0.43	0.60	0.75	0.6	$1.29 * 10^6$	0.4807%	98.44%	3.12%	90.2%	89.6%	89.6%	0.8%
48	0.43	0.60	0.75	0.6	$1.31 * 10^6$	0.4883%	100.00%	3.12%	90.2%	89.6%	89.6%	0.8%

* $\mathcal{F}$ compared to the SML ω_{256}.

† $\mathcal{F}$ compared to the naïve binary SML-CPCA ω_{256}.

Table A.32: Best achieved CPCA-Tree TP_0 with corresponding $TP_{0.1}$ and $TP_{0.5}$ for every tree preselection of $t = 1, \ldots, 48$ trees using SMR-CPCA-M templates at $\mathcal{T} = 64$ with an additional binary SML-CPCA comparison and masking out most common set bits for session 2 of the PolyU dataset.

Name	$\delta(TP_0, F_B)$	$F_B{}^*$	TP_0
Bf CPCA ARC	0.10	0.1952	89.0%
C-T SCM Masking	0.14	0.2384	89.0%
C-T SCM	0.14	0.2441	89.0%
C-T SCM ARC	0.29	0.3128	88.8%
Binary PSML-CPCA†	0.39	0.4880	89.0
Bf CPCA ABC	1.03	0.3437	88.0%

Name	$\delta(TP_{0.1}, F_B)$	$F_B{}^*$	$TP_{0.1}$
Bf CPCA ARC	0.10	0.1952	89.1%
C-T SCM Masking	0.12	0.2155	89.1%
C-T SCM	0.14	0.2365	89.1%
C-T SCM ARC	0.29	0.3052	88.9%
Binary PSML-CPCA†	0.39	0.4880	89.1%
Bf CPCA ABC	0.32	0.3437	88.9%

* F compared to the PSML ω_{256}.

† Naïve approach.

Table A.33: Ranking for each relevant experiment, respectively workload reduction method, ordered by the euclidean distance rating method with optimal operation point at naïve binary PSML-CPCA biometric performance ($TP_0 = 89.0$) and $F = 0.1\%$.

List of Symbols

BC - **Bloom filter set size** Amount of Bloom filter $\mathcal{B}$in a set **B**. 46, 48

M_{CPCA} - **SMR height** Height of an SMR-CPCA template. 58–60, 65, 90, 91, 94

M - **SMR height** Height of an SMR template. 46, 60, 65

N - **SMR width** Width of an SMR template. 46, 65, 94

PER - **Pre-selection Error Rate** Pre-selection error: error that occurs when the corresponding enrolment template is not in the pre-selected subset of candidates when a sample from the same biometric characteristic on the same user is given; PER: rate of falsely selected rank N (here 1) candidates. 94, 105–111, 127, 147–155, 158–160, 163–165

F - **Workload Fraction** Workload reduction expressed as fraction of the baseline workload (see section 8.5). 75, 76, 90–93, 95–105, 107, 108, 110, 111, 113–118, 136–166

$\mathcal{B}$ - **Bloom filter** A single Bloom filter. 45, 46, 48, 53, 54, 167, 168

$\mathcal{C}$ - **Bloom filter-Trees** Amount of template comparisons of a single lookup in an identification scenario. 51–53

$\mathcal{H}$ - **Bloom filter height** Block-size height used for the Bloom filter template transformation. 46, 49, 50, 53, 54, 93–105, 136–146, 157, 162

$\mathcal{S}$ - **Subjects** Enrolled subjects in database. Note: one subject is able to present two instances. Since there is no possibility to link two instances to one subject, one instance is treated as one subject in this thesis. 10, 47, 48, 50–53, 62, 75, 76, 88, 119, 126, 128

$\mathcal{T}$ - **Bloom filter-Trees** Amount of constructed trees in a Bloom filter identification scenario. 50, 51, 60, 61, 93–102, 105–111, 124, 136–141, 143–155, 157–160, 162–165

$\mathcal{W}$ - **Bloom filter width** Block-size width used for the Bloom filter template transformation. 46, 49, 53, 54, 93–105, 136–146, 157, 162

$\mathcal{X}$ - **SMR Spectrum** Spectrum symbol for the SMR. 32–35, 65

L - **CPCA training set size** Amount of used SMR templates for the training of the CPCA (see section 5.4.1). 89, 134

MT **- Mask Threshold** Threshold value used during binarisation of the SMR spectrum. 34, 35, 90–93, 95–105, 107, 108, 110, 111, 114, 136–165

t **- Selected Bloom filter-Trees** Amount of selected trees during a tree pre-selection step. 51, 52, 96–98, 100–102, 106–108, 110, 111, 114, 116, 117, 119, 138–140, 144–160, 162–165

B - Bloom filter set A set of Bloom filter $\mathcal{B}$. 47, 48, 167

List of Abbreviations

CPCA - Column Principal Component Analsysis See section 5.4.1. 31–34, 56–67, 72, 74, 75, 89–91, 134

CPCA-Tree - SMR-CPCA binary search tree Binary search tree build with SMR-CPCA templates. See chapter 7. 59–61, 67, 74, 105–111, 113–116, 119, 126–128, 147–155, 158–160, 163–165

FHD - fractional Hamming Distance Algorithm used to compare two masked binary feature-vectors (see 5.6.2). 60

LDFT - Line Discrete Fourier Transformation See section 5.4.2. 33

MHD - modified Hausdorf Distance Algorithm used to compare feature-vectors (set of points, see 9.1). 77–84, 86, 87, 120

NIR - Near-Infrared Radiation with wavelength from about 700 nm to 2500 nm, thus invisible to the human eye. 4–6, 8, 12, 19, 20, 23, 68, 70

NL-diffusion - Non-Linear Diffusion Image enhancement algorithm from [Wei01]. Used to reduce high-frequency noise and enhancing contrast. 23, 24, 77

NL-means - Non-Local Means Image enhancement algorithm from [ŠP09]. Used to create an illumination invariant texture by non-local smoothing. 22–24, 77

PSMC - Spectral Minutia Complex Representation with minutiae pre-selection See section 5.8. 85

PSML - Spectral Minutia Location Representation with minutiae pre-selection See section 5.8. 75, 86, 88–93, 95–105, 107, 108, 110–115, 119, 136–155, 166

PSML-CPCA - PSML reduced with the CPCA feature-reduction See section 5.4.1. 88–111, 113–116, 136–142, 147–155, 164, 166

PSML-CPCA-A - PSML-CPCA-Applied PSML-CPCA representation with applied mask-bit on the sign-bit. See section 7.1.4. 106, 108, 109

PSML-CPCA-C - PSML-CPCA-Components The sign-bit and mask-bit of the PSML-CPCA. see section 7.1.4. 106–108

PSML-CPCA-M - PSML-CPCA-Mixed Same as PSML-CPCA-Applied but keeping the mask-bit. See section 7.1.4. 106–111, 114, 117

PSMR - Spectral Minutia Representation with minutiae pre-selection See section 5.8. 42, 82, 85, 87

QSMC - quality data enhanced Spectral Minutia Complex Representation See section 5.7. 42, 82, 83, 86, 122

QSML - quality data enhanced Spectral Minutia Location Representation See section 5.7. 38, 42, 81–83, 86, 88, 121, 134

ROC - Receiver Operating Characteristic Curve Plot of the rate of false positives (i.e. impostor attempts accepted) against the corresponding rate of true positives as a function of the decision threshold. 80–91, 112, 116, 118, 121, 122, 133–135

ROI - Region of Interest Part of an image which contains the desired informations. 8, 18–24, 68–72, 76, 77, 83, 85, 121

SMC - Spectral Minutia Complex Representation See section 5.1.3. 29, 30, 32, 42, 65, 81–86, 122

SML - Spectral Minutia Location Representation See section 5.1.1. 28–30, 32, 34, 36, 37, 42, 54, 57, 65, 66, 81–83, 86, 89, 112, 113, 121, 122, 134, 156–165

SML-CPCA - SML reduced with the CPCA feature-reduction See section 5.4.1. 57, 58, 155–165

SMO - Spectral Minutia Orientation Representation See section 5.1.2. 29, 32, 81

SMR - Spectral Minutia Representation See section 5.1. 27–38, 40, 42, 45, 46, 49, 53–57, 59, 61, 63–67, 73, 74, 76, 80–93, 95–105, 107, 108, 110–113, 116, 119, 121, 123–129, 134–165

SMR-CPCA - SMR reduced with the CPCA feature-reduction See section 5.4.1. 36, 56–61, 64, 65, 74, 90, 106, 116, 119, 125–128

SMR-CPCA-A - SMR-CPCA-Applied SMR-CPCA representation with applied mask-bit on the sign-bit. See section 7.1.4. 59, 60, 106

SMR-CPCA-C - SMR-CPCA-Components The sign-bit and mask-bit of the SMR-CPCA. see section 7.1.4. 59, 60, 106, 147

SMR-CPCA-M - SMR-CPCA-Mixed Same as SMR-CPCA-Applied but keeping the mask-bit. See section 7.1.4. 60, 106, 148–156, 158–160, 163–165

Bibliography

[AG14] AM Abbas and Loay E George, *Palm vein recognition and verification system using local average of vein direction*, Int. J. Sci. Eng. Res **5** (2014), no. 4, 1026–1033.

[Bad06] Ahmed M Badawi, *Hand vein biometric verification prototype: A testing performance and patterns similarity.*, IPCV **14** (2006), 3–9.

[BLM14] Richard Barnes, Clarence Lehman, and David Mulla, *Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models*, Computers & Geosciences **62** (2014), 117–127.

[Blo70] Burton H Bloom, *Space/time trade-offs in hash coding with allowable errors*, Communications of the ACM **13** (1970), no. 7, 422–426.

[BP98] Ruud Bolle and Sharath Pankanti, *Biometrics, personal identification in networked society: Personal identification in networked society*, Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[Bus10] Christoph Busch, *Iso 24745-biometric template protection*, The first International Biometric Performance Testing Conference, Match, 2010.

[cas] *Casia-ms-palmprintv1*, http://biometrics.idealtest.org/.

[CDG+08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber, *Bigtable: A distributed storage system for structured data*, ACM Transactions on Computer Systems (TOCS) **26** (2008), no. 2, 4.

[CFM10] Raffaele Cappelli, Matteo Ferrara, and Davide Maltoni, *Minutia cylinder-code: A new representation and matching technique for fingerprint recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010), no. 12, 2128–2141.

[CFM11] ———, *Fingerprint indexing based on minutia cylinder-code*, IEEE Transactions on Pattern Analysis and Machine Intelligence **33** (2011), no. 5, 1051–1057.

[CP76] David Casasent and Demetri Psaltis, *Position, rotation, and scale invariant optical correlation*, Applied optics **15** (1976), no. 7, 1795–1799.

[Cre16] Credence Research, *Biometrics technology market by technology (face recognition, hand geometry, voice recognition, signature recognition, iris recognition, afis, non-afis & others), end use vertical (government, military and defense, healthcare, banking and finance, consumer electronics, residential and commercial security, transportation and immigration & others) - growth, share, opportunities & competitive analysis, 2015 - 2022*, 4 2016.

[csp13a] *Case study: Palmsecure - maximum access security for sensitive areas*, https://sp.ts.fujitsu.com/dmsp/Publications/public/cs-PalmSecure-PCS-TDS-en.pdf, 2013.

[csp13b] *Case study: Palmsecure - over 700 million transactions without any fraud incident*, http://www.fujitsu.com/global/Images/cs-banco-bradesco-en.pdf, 2013.

[csp14] *Case study: Palmsecure - ferencváros soccer club scores with biosec biometric personal identification system for its new stadium*, https://sp.ts.fujitsu.com/dmsp/Publications/public/cs-PalmSecure-BioSec.pdf, 2014.

[Dau00] John Daugman, *Biometric decision landscapes*, Tech. report, University of Cambridge, Computer Laboratory, 2000.

[Dau03] ———, *The importance of being random: statistical principles of iris recognition*, Pattern recognition **36** (2003), no. 2, 279–291.

[Dau04] ———, *How iris recognition works*, IEEE Transactions on circuits and systems for video technology **14** (2004), no. 1, 21–30.

[DJ94] M.-P. Dubuisson and A.K. Jain, *A modified hausdorff distance for object matching*, Proceedings of 12th International Conference on Pattern Recognition, IEEE Comput. Soc. Press, 1994.

[DRB17] Pawel Drozdowski, Christian Rathgeb, and Christoph Busch, *Bloom filter-based search structures for indexing and retrieving iris-codes*, IET Biometrics (2017).

[FZ05] Marcos Faundez-Zanuy, *Privacy issues on biometric systems*, IEEE A&E SYSTEMS MAGAZINE (2005), http://ai.pku.edu.cn/aiwebsite/research.files/collected%20papers%20-%20introduction/Privacy%20issues%20on%20biometric%20systems.pdf.

[GBRG+16] Marta Gomez-Barrero, Christian Rathgeb, Javier Galbally, Christoph Busch, and Julian Fierrez, *Unlinkable and irreversible biometric template protection based on bloom filters*, Information Sciences **370** (2016), 18–32.

[GH89] Zicheng Guo and Richard W Hall, *Parallel thinning with two-subiteration algorithms*, Communications of the ACM **32** (1989), no. 3, 359–373.

[GL18] H. Gray and W.H. Lewis, *Anatomy of the human body*, Lea & Febiger, 1918.

[GY83] Ronald L Graham and F Frances Yao, *Finding the convex hull of a simple polygon*, Journal of Algorithms **4** (1983), no. 4, 324–331.

[GZZ+11] Zhenhua Guo, David Zhang, Lei Zhang, Wangmeng Zuo, and Guangming Lu, *Empirical study of light source selection for palmprint recognition*, Pattern Recognition Letters **32** (2011), no. 2, 120–126.

[Har12] Daniel Hartung, *Vascular pattern recognition and its application in privacy-preserving biometric online-banking systems*, Ph.D. thesis, Gjovik University College, Faculty of Computer Science and Media Technology, 2 2012.

[HDZ08] Feng Hao, John Daugman, and Piotr Zielinski, *A fast search algorithm for a large fuzzy database*, IEEE Transactions on Information Forensics and Security **3** (2008), no. 2, 203–212.

[Hes12] Scott Hess, *Issue 10896048: Transition safe browsing from bloom filter to prefix set.*, 08 2012.

[HGZ08] Dong Han, Zhenhua Guo, and David Zhang, *Multispectral palmprint recognition using wavelet-based image fusion*, Signal Processing, 2008. ICSP 2008. 9th International Conference on, IEEE, 2008, pp. 2074–2077.

[HOX+12] Daniel Hartung, Martin Aastrup Olsen, Haiyun Xu, Hai Thanh Nguyen, and Christoph Busch, *Comprehensive analysis of spectral minutiae for vein pattern recognition.*, IET Biometrics **1** (2012), no. 1, 25–36.

[HOXB11] Daniel Hartung, Martin Aastrup Olsen, Haiyun Xu, and Christoph Busch, *Spectral minutiae for vein pattern recognition*, Biometrics (IJCB), 2011 International Joint Conference on, IEEE, 2011, pp. 1–7.

[Ind] India, *Aadhaar issued summary.*

[iso06] *ISO/IEC 19795-1:2006 - Information technology – Biometric performance testing and reporting – Part 1: Principles and framework*, International Standard, 2006.

[JHF14] Y. Jo, S. Hama, and M. Fukuda, *Vein authentication method, image processing method, and vein authentication device*, October 2 2014, US Patent App. 14/299,435.

[JHZ08] Wei Jia, De-Shuang Huang, and David Zhang, *Palmprint verification based on robust line orientation code*, Pattern Recognition **41** (2008), no. 5, 1504–1513.

[KK10] Rafał Kabaciński and Mateusz Kowalski, *Human vein pattern segmentation from low quality images–a comparison of methods*, Image Processing and Communications Challenges **2** (2010), no. 84, 105–112.

[KK11a] R Kabacinski and Mateusz Kowalski, *Vein pattern database and benchmark results*, Electronics Letters **47** (2011), no. 20, 1127–1128.

[KK11b] Rafał Kabacinski and Mateusz Kowalski, *Human vein pattern correlation-a comparison of segmentation methods*, Computer Recognition Systems 4, 95 of Advances in Intelligent and Soft Computing (2011), 51–59.

[Koc13] Davit Kocharyan, *A modified fingerprint image thinning algorithm*, American Journal of Software Engineering and Applications **2** (2013), no. 1, 1–6.

[Llo82] Stuart Lloyd, *Least squares quantization in pcm*, IEEE transactions on information theory **28** (1982), no. 2, 129–137.

[LT15] Sen Lin and Yu Tai, *A combination recognition method of palmprint and palm vein based on gray surface matching*, Image and Signal Processing (CISP), 2015 8th International Congress on, IEEE, 2015, pp. 567–571.

[Lun61] RJ Lunnon, *Some observations on the photography of the diseased skin.*, Medical & biological illustration **11** (1961), 98.

[MNM04] Naoto Miura, Akio Nagasaka, and Takafumi Miyatake, *Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification*, Machine Vision and Applications **15** (2004), no. 4, 194–203.

[MNM07] ———, *Extraction of finger-vein patterns using maximum curvature points in image profiles.*, IEICE Transactions **90-D** (2007), no. 8, 1185–1194.

[MS15] Bruce M Maggs and Ramesh K Sitaraman, *Algorithmic nuggets in content delivery*, ACM SIGCOMM Computer Communication Review **45** (2015), no. 3, 52–66.

[Nad07] Annemarie Nadort, *The hand vein pattern used as a biometric feature*, Master's thesis, Free University, Amsterdam, 3 2007.

[OHBL11] Martin Aastrup Olsen, Daniel Hartung, Christoph Busch, and Rasmus Larsen, *Convolution approach for feature detection in topological skeletons obtained from vascular patterns*, 2011 IEEE Workshop on Computational Intelligence in Biometrics and Identity Management (CIBIM), IEEE, apr 2011.

[OWN96] Alan V Oppenheim, Alan S Willsky, and S Hamid Nawab, *Signals & Systems*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA **18** (1996), 19–21.

[Pea01] Karl Pearson, *On lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2** (1901), no. 11, 559–572.

[PGS+09] A. Jayaprakash Patil, Ana L. Gramajo, Ashish Sharma, Gail M. Seigel, Baruch D. Kuppermann, and M. Cristina Kenney, *Differential effects of nicotine on retinal and vascular cells in vitro*, Toxicology **259** (2009), no. 1-2, 69–76.

[Pit00] Michael Pittilo, *Cigarette smoking, endothelial injury and cardiovascular disease*, International Journal of Experimental Pathology **81** (2000), no. 4, 219–230.

[PK11] Mi Pan and Wenxiong Kang, *Palm vein recognition based on three local invariant feature extraction algorithms*, Chinese Conference on Biometric Recognition, Springer, 2011, pp. 116–124.

[pol] *Polyu multispectral palmprint database*, http://www.comp.polyu.edu.hk/~biometrics/MultispectralPalmprint/MSP.htm.

[Pro13] Hugo Proenca, *Iris biometrics: Indexing and retrieving heavily degraded data*, IEEE Transactions on Information Forensics and Security **8** (2013), no. 12, 1975–1985.

[PUASRL10] J Enrique Suarez Pascual, Jaime Uriarte-Antonio, Raul Sanchez-Reillo, and Michael G Lorenz, *Capturing hand or wrist vein images for biometric authentication using low-cost devices*, Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on, IEEE, 2010, pp. 318–322.

[RB13] Christian Rathgeb and Christoph Busch, *Comparing binary iris biometric templates based on counting bloom filters.*, CIARP (2) (José Ruiz-Shulcloper and Gabriella Sanniti di Baja, eds.), Lecture Notes in Computer Science, vol. 8259, Springer, 2013, pp. 262–269.

[RBB13] Christian Rathgeb, Frank Breitinger, and Christoph Busch, *Alignment-free cancelable iris biometric templates based on adaptive bloom filters.*, ICB (Julian Fiérrez, Ajay Kumar, Mayank Vatsa, Raymond N. J. Veldhuis, and Javier Ortega-Garcia, eds.), IEEE, 2013, pp. 1–8.

[RBBB15] Christian Rathgeb, Harald Baier, Christoph Busch, and Frank Breitinger, *Towards bloom filter-based indexing of iris biometric data.*, ICB, IEEE, 2015, pp. 422–429.

[Ric85] J. Rice, *Method and apparatus for the identification of individuals*, September 26 1985, WO Patent App. PCT/GB1985/000,127.

[RJ+91] Sarunas J Raudys, Anil K Jain, et al., *Small sample size effects in statistical pattern recognition: Recommendations for practitioners*, IEEE Transactions on pattern analysis and machine intelligence **13** (1991), no. 3, 252–264.

[RU11] Christian Rathgeb and Andreas Uhl, *A survey on biometric cryptosystems and cancelable biometrics*, EURASIP Journal on Information Security **2011** (2011), no. 1, 3.

[S+85] Satoshi Suzuki et al., *Topological structural analysis of digitized binary images by border following*, Computer vision, graphics, and image processing **30** (1985), no. 1, 32–46.

[Skl82] Jack Sklansky, *Finding the convex hull of a simple polygon*, Pattern Recognition Letters **1** (1982), no. 2, 79–83.

[SKS+14] T. Shindo, T. Kamimura, N. Suzuki, K. Ogawa, S. Eguchi, and M. MANABE, *Authentication apparatus, image capture apparatus, authentication method, and authentication program*, February 11 2014, US Patent 8,649,569.

[ŠP09] Vitomir Štruc and Nikola Pavešić, *Illumination invariant face recognition by non-local smoothing*, European Workshop on Biometrics and Identity Management, Springer, 2009, pp. 1–8.

[SRB15] Jayachander Surbiryala, Ramachandra Raghavendra, and Christoph Busch, *Finger vein indexing based on binary features*, Colour and Visual Computing Symposium (CVCS), 2015, IEEE, 2015, pp. 1–6.

[The06] M Theme, *Comparative biometric testing round 6 public report*, The International Biometric Group (2006).

[Tra14] Transparency Market Research, *Biometrics technology (face, hand geometry, voice, signature, iris, afis, non-afis and others) market - global industry analysis size share growth trends and forecast 2013 - 2019*, 4 2014.

[Tra17] Tractica LLC, *Biometrics market forecasts: Global unit shipments and revenue by biometric modality, technology, use case, industry segment, and world region: 2016-2025*, Q1 2017.

[Wat14] M. Watanabe, *Biometric authentication apparatus and biometric authentication method*, August 19 2014, US Patent 8,811,681.

[Wei01] Joachim Weickert, *Applications of nonlinear diffusion in image processing and computer vision*, Acta Math. Univ. Comenianae **70** (2001), no. 1, 33–50.

[WESS05] Masaki Watanabe, Toshio Endoh, Morito Shiohara, and Shigeru Sasaki, *Palm vein authentication technology and its applications*, Proceedings of the biometric consortium conference, 2005, pp. 19–21.

[WL06] Lingyu Wang and Graham Leedham, *Near-and far-infrared imaging for vein pattern biometrics*, Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on, IEEE, 2006, pp. 52–52.

[XV09] Haiyun Xu and Raymond NJ Veldhuis, *Spectral minutiae representations of fingerprints enhanced by quality data*, Biometrics: Theory, Applications, and Systems, 2009. BTAS'09. IEEE 3rd International Conference on, IEEE, 2009, pp. 1–5.

[XV10a] ———, *Binary representations of fingerprint spectral minutiae features*, Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 1212–1216.

[XV10b] ———, *Complex spectral minutiae representation for fingerprint recognition*, Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, IEEE, 2010, pp. 1–8.

[XV10c] ———, *Spectral minutiae representations for fingerprint recognition*, Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on, IEEE, 2010, pp. 341–345.

[XVB+09] Haiyun Xu, Raymond NJ Veldhuis, Asker M Bazen, Tom AM Kevenaar, Ton AHM Akkermans, and Berk Gokberk, *Fingerprint verification using spectral minutiae representations*, IEEE Transactions on Information Forensics and Security **4** (2009), no. 3, 397–409.

[XVK+08] Haiyun Xu, Raymond NJ Veldhuis, Tom AM Kevenaar, Anton HM Akkermans, and Asker M Bazen, *Spectral minutiae: A fixed-length representation of a minutiae set*, Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on, IEEE, 2008, pp. 1–6.

[XVKA09] Haiyun Xu, Raymond NJ Veldhuis, Tom AM Kevenaar, and Ton AHM Akkermans, *A fast minutiae-based fingerprint recognition system*, IEEE Systems journal **3** (2009), no. 4, 418–427.

[Yak10] Alex Yakunin, *Nice bloom filter application*, 03 2010.

[Zae11] Naser Zaeri, *Minutiae-based fingerprint extraction and recognition*, INTECH Open Access Publisher, 2011.

[ZGL+10] David Zhang, Zhenhua Guo, Guangming Lu, Lei Zhang, and Wangmeng Zuo, *An online system of multispectral palmprint verification*, IEEE transactions on instrumentation and measurement **59** (2010), no. 2, 480–490.

[ZK11] Yingbo Zhou and Ajay Kumar, *Human identification using palm-vein images*, IEEE transactions on information forensics and security **6** (2011), no. 4, 1259–1274.

[ZKYW03] David Zhang, Wai-Kin Kong, Jane You, and Michael Wong, *Online palmprint identification*, IEEE Transactions on pattern analysis and machine intelligence **25** (2003), no. 9, 1041–1050.

[ZLF+14] Y Zhou, Y Liu, Q Feng, F Yang, J Huang, Ian McLoughlin, et al., *Palm-vein classification based on principal orientation features*, PLoS ONE **9** (2014), no. 11, e112429.