BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF BIOMEDICAL ENGINEERING

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

BIOMETRIC FINGERPRINT IDENTIFICATION

BIOMETRICKÁ IDENTIFIKACE OTISKU PRSTU

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Matej Hlavatý**
AUTOR PRÁCE

SUPERVISOR — **Ing. Martin Vítek, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Biomedicínské inženýrství a bioinformatika**
Ústav biomedicínského inženýrství

Student: Bc. Matej Hlavatý ***ID:*** 146190
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Biometrická identifikace otisku prstu

POKYNY PRO VYPRACOVÁNÍ:

1) Nastudujte a popište metody biometrické identifikace otisku prstu. Zaměřte se především na příznaky, které se k identifikaci používají. 2) S využitím volně dostupných databází vytvořte vlastní testovací databázi otisků prstů. 3) Na základě nastudovaných metod zvolte sadu příznaků vhodnou pro identifikaci otisku prstu. 4) S ohledem na zvolené příznaky navrhněte a v prostředí Matlab realizujte vhodné předzpracování obrazů. 5) Realizujte algoritmus identifikace otisku prstu založený na kombinování zvolených příznaků. 6) Navržený algoritmus otestujte a dosažené výsledky statisticky zpracujte. Účinnost algoritmu porovnejte s ostatními autory. 7) Program opatřete vhodným grafickým uživatelským rozhraním.

DOPORUČENÁ LITERATURA:

[1] DRAHANSKÝ, Martin a ORSÁG, Filip. Biometrie. 1. vyd. [Brno: M. Drahanský], 2011. 294 s. ISBN 978-8--254-8979-6.

[2] RAK, R., V. MATYÁŠ a Z. ŘÍHA. Biometrie a identita člověka. 1. vyd. Praha: Grada Publishing a.s., 2008, 631 s. ISBN 978-80-247-2365-5.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 19.5.2017

Vedoucí práce: Ing. Martin Vítek, Ph.D.
Konzultant:

prof. Ing. Ivo Provazník, Ph.D.
předseda oborové rady

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Projekt sa zaoberá spracovaním a porovnaním otlačkov prstov. Preberá obecné princípy biometrie a rôzne metódy analýzy otlačkov prstov. Navrhuje vlastné riešenie problému formou adaptívnych maskových operátorov na detekciu markantov a štatistické spracovanie výsledkov.

KĽÚČOVÉ SLOVÁ

Otlačky prstov, biometria, markanty, spracovanie obrazu

ABSTRACT

The aim of this project is an automatic analysis of fingerprint images. Basic principles of biometrics and commonly used methods of fingerprint analysis are studied. An algorithm is proposed, using adaptive mask operators for minutiae detection, followed by a statistical evaluation of the achieved results.

KEYWORDS

Fingerprints, biometrics, minutiae, image processing

HLAVATÝ, M. Biometrická identifikace otisku prstu. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 28 s. Vedoucí diplomové práce Ing. Martin Vítek, Ph.D..

PREHLÁSENIE

Prehlasujem, že svoju diplomovú prácu na tému Biometrická identifikácia otlačku prstu som vypracoval samostatne pod vedením Ing. Martina Vítka Phd. a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a plne si uvedomujem následky porušenia ustanovenia § 11 a nasledujúcich zákona č. 121/2000 Sb., o právu autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. Dielu 4 Trestného zákonníku č. 40/2009 Sb.

V Brne dňa .....

.....

(podpis autora)

POĎAKOVANIE

Ďakujem vedúcemu diplomovej práce Ing. Martinovi Vítkovi Phd. za účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady pri zpracovaní mojej diplomovej práce.

V Brne dňa .....

.....

(podpis autora)

Contents

LIST OF IMAGES

INTRODUCTION

The field of biometrics studies measurable traits of human body suitable for identification and classification of individuals. Fingerprints are the most commonly used biometrical trait, due to their uniqueness and relative ease of acquirement and processing. Many automated systems analysing fingerprints have been proposed, resulting in a large variety of complementary methods of classification.

The first goal of this project is to study these methods to learn about previous work done on the subject. Plausibility of each method and their implementation in a custom solution is considered.

Secondly, a database of fingerprint images is constructed from publicly available internet sources. The database should serve to train the proposed algorithm; thus, it should provide sufficient variability to make the final product robust against various problems present in real fingerprint images.

Individual database images are processed and important data is extracted. Based on the expected results, optimisation of the algorithm variables is achieved. Once all optimised datasets of the database entries are complete, the database is considered ready for use.

Using a Graphical User Interface, a query image is chosen to compare against a database entry. After processing and data extraction from the query image, the resulting dataset is compared with database datasets and a verdict – match/mismatch - based on a decision metric, is announced.

A statistical evaluation of the system accuracy and sensitivity is provided, followed by a discussion comparing the results of similar works.

1 THEORETICAL BACKGROUND

1.1 Biometrics

Using the metrics of the human body to identify and classify individuals is a task human brain does every day. Measuring and processing these metrics is the objective of the field of biometrics.

Nehemiah Grew published the first study of papillary ridge structure on human palm and fingers in 1684. The unique properties of individual fingerprints went unnoticed for more than a hundred years, until Johann Christoph Andreas Mayer published his study in 1788. The first practical use of using fingerprints for identity verification could be considered Thomas Bewick using his fingerprint to complement his signature for business contracts in 1809, but William James Hershel, a British governor in India, was the first to use fingerprints instead of signature on a large scale for his workers in 1858, as it was a more reliable method of identification. The fingerprints were also used for his private studies, which later lead to a publication about fingerprint origins. [1]

Francis Galton studied behavioural and physical trait inheritance, which lead him to co-founding the field of eugenics in 1869. A decade later, he also invented the field of anthropometrics, studying human body proportions. In 1892 he wrote a publication called "Fingerprints" and coined the term *dactyloscopy*. In 1900, he proves fingerprints to be unique to every individual and they start being used by police for criminal investigation. The works of Henry Faulds from 1880, proposing usage of fingerprints for identification and suggesting ink to be used for acquiring the fingerprints; and Juan Vucetich from 1891, introducing fingerprint categorisation and storage, deserve to be mentioned, as they also helped to promote fingerprint usage in forensic science. [1]

Alphonse Bertillon worked on a system of measuring physical properties and distinguishing features between 1879 and 1882, which he called *bertillonage*. The method was practically identical to Galton's anthropometrics, but both are the only significant attempts of using other biometrics than fingerprints before 20th century. [1]

The modern biometrics involve a host of various traits, described as passive versus active, or physiological versus behavioural. The main goal is to find a set of traits that are unique for each person, easy to measure and possible to be processed in real time, with respect to the intended purpose.

1.1.1 Practical applications

Apart from biometrical traits, other methods are used to verify identity or restrict access, with longer tradition as well. These may be divided into two categories: [1][2]

- Password (knowledge based) – a word known only by the authorized users and the system, easy to verify and change, but relies heavily on the users' memory
- Key (token based) – an object that only the authorized users own, requires no knowledge from the user, but is more difficult to verify and modify, as well as easier to steal

The benefit of using biometrical systems over passwords and keys is the inherent uniqueness of biometrical traits – they require no setup (only initial database entry), no user-knowledge or memory, and most importantly are very difficult to steal or replicate. This uniqueness also causes some drawbacks, however. If a person's traits are somehow successfully copied and misused, it is practically impossible to provide that person with a new set of traits (compared to changing password, or issuing a new keycard). [1]

Many modern systems, called hybrid, or multimodal, use a combination of biometrical traits, unlike the simpler, unimodal systems focused on working with a single biometrical trait only. The individual problems of various unimodal systems and errors in classification can be improved significantly by adding just one more trait for analysis. The resulting hybrid system is usually very robust and can be considered standard for applications with high security demands. The more complex the hybrid system is, the higher demands on processing software and hardware it creates. The price for such a hybrid system can be significantly higher than a simple unimodal one, especially if different types of sensors are required to acquire the traits (for example, a fingerprint and retina sensor).

Besides the field of forensic science, biometrics are most commonly used in the following applications:

- Identity checks at border control
- Employee attendance
- Restricting access to high-security areas
- Authorisation of important bank transactions
- Face recognition from public cameras and CCTV
- Unlocking personal devices – smartphones, tablets and laptops

1.1.2 Biometric identifiers

A biometric trait, or identifier, can be any characteristic of a human body that is unique to the person in some way and is distinctly measurable.

The traits that are measurable by physical parameters of the human body are called static, physiological, or anatomical. They are independent on the current condition of the person and always available (unless somehow damaged). These include:

- Fingerprints
- Face
- Iris
- Retina
- Hand geometry
- Palm
- Face thermogram
- Hand thermogram
- Dental record
- Signature (static form)
- Ear shape
- Fingernail
- DNA

Dynamic, or behavioural traits are related to a person's actions. These may change easily and subsequent data acquisition often produces very different results. These include:

- Voice and speech
- Facial expressions
- Gait
- Signature (dynamic form)
- Keyboard typing dynamics

[1]

There are many requirements on biometrical parameters to determine the proper ones for a specific application. Even the most advanced hybrid systems can't identify and process all of them in real time, so a proper balance should be found. Below is the list of desired properties and a table describing properties of each trait:

- Unique to a single person, so that a clear distinction can be made between

individuals (A)

- Present on every user, so that some individuals are not excluded from using the system simply because it can't measure such a trait (B)
- Easy to acquire and measure objectively (C)
- Simple enough to process, reducing the overall complexity of the system (D)
- Requiring inexpensive measuring equipment and overall system cost (E)
- Easy for users to accept the data collection (F)
- Impossible to replicate and misuse (G)
- Invariant over long amounts of time and resistant to aging (H)
- Independent of personal choice of clothes, accessories or hair style
- and others

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Fingerprints	+	0	0	+	-	0	+	+
Face	-	+	+	-	-	+	-	0
Iris	+	+	0	+	+	-	+	+
Retina	+	+	-	+	+	-	+	0
Hand geometry	0	0	+	0	0	0	0	0
Palm	0	0	0	0	0	0	+	0
Thermogram	+	+	+	0	+	+	+	-
Signature	-	-	+	-	-	+	-	-
Voice	-	0	0	-	-	+	-	-

Table 1 - Biometric traits and their properties

+ = trait performs well in this area; - = trait is inadequate; 0 = neutral:

[1][3]

1.1.3 Process of identification

The basic process of identifying a user consists of acquiring the user's biometrical data and comparing it with the database entries. If a sufficient match is found, the user is verified.

Depending on the purpose of the verification, two different processes can be described: [1]

- Verifying a user's identity. The user is requesting access to a file, system, or area and the database has the user's profile stored. The system compares the known database entry to the data taken from the user and verifies that it is indeed the same person. The comparison is 1:1
- Identifying an unknown user. The system tries to match an unknown query to the database of known entries; for example, searching a criminal database for a suspect, a database of asylum seekers, or reviewing a new registration request to prevent duplicate user entries. The comparison is 1:N

Before the standard verification can proceed, however, the database must be updated with the user's data. The initial user enrolment has higher requirements on the quality of the captured data than during casual verification, as the database data will be compared to each subsequent verification attempt and any error or artefact may cause more serious problems than the same error in a few of the verification attempts.

Some biometrical traits have a defined standard of quality that the database entries should fulfil. This ensures that the database entry is of sufficient quality and benefits the software processing of the input data as well; knowing that every input image of a face will have the same position and orientation, for example, allows the programmers to avoid certain rotational and matching operations. The most common example may be taking photographs for an identification card or driving license, where the users are required to hold their head straight, look directly into the camera and avoid smiling or other facial expressions. Some systems also employ a software that automatically looks for the required standards during every data acquisition and highlights the conflicts in real time.

Acquiring and storing data is one of the most vulnerable points of the entire biometrical system, from the security point of view. Making physical copies of the biometrical trait is always an option, but significantly less feasible than intercepting digital data or stealing database entries. Proper security measures in this area are therefore very important, especially due to the problem of creating a new entry for a compromised user. [1]

1.2 Fingerprints in biometry

Fingerprints have been the first widely used biometrical trait for several reasons:

- High availability – it's very unlikely that a person has fingerprints on all fingers unavailable or damaged beyond recognition
- Uniqueness – there is a low probability of falsely classifying a person, provided that no errors or noise exist in the data
- Acceptance of the data acquisition – taking a fingerprint image is relatively simple from the user's perspective and many of them have no problems with providing them
- Permanence – unlike most other biometrical traits, fingerprints are extremely resistant to aging, even more than facial expressions, or gait; only scars or other surface damage to the fingerprint may affect the result. Changing someone's fingerprints is only possible by irreversible damage, causing the fingerprint to be unusable.

There are, however, drawbacks that should be considered, such as

- Acquisition errors – placing the finger on the sensor with insufficient force may result in spatial distortions in the image and repeated use of the sensor without cleaning produces data of progressively deteriorating quality
- Data quality – image noise and spatial distortions may result in false classifications, if the fingerprints are the only biometrical trait used for classification
- Fingerprints are slightly more vulnerable to illicit duplication than other traits, as the print residue can be collected without the user's presence; this could also be considered a benefit for forensic use of fingerprints, however

[1][5]

1.2.1 Fingerprint properties

The skin surface of the human fingers forms ridges – slightly raised lines running in parallel, curving and bending to form a unique pattern, occasionally splitting, joining, or suddenly ending, but never making sharp edge turns. The pattern is especially detailed on the last finger segment, directly opposite the fingernail. This is the area used for taking fingerprints – images of the ridge pattern, where the ridges are distinctly visible against the 'valleys' – the white background of the image. [1][5]

Traditionally, the fingerprints were created by applying black powder, or ink on the

finger and pressing the finger against a blank paper – thus the name finger print. The modern biometric systems use digital scanners, however, capable of taking the fingerprint image without the discomfort of staining the user's fingers.

The pattern the ridges make could be unique enough on its own to distinguish two different fingerprints, but it's the occasional disruptions of the otherwise regular parallel ridges. These disruptions are called minutiae and are often used for fingerprint verification and classification. [1][5][6]

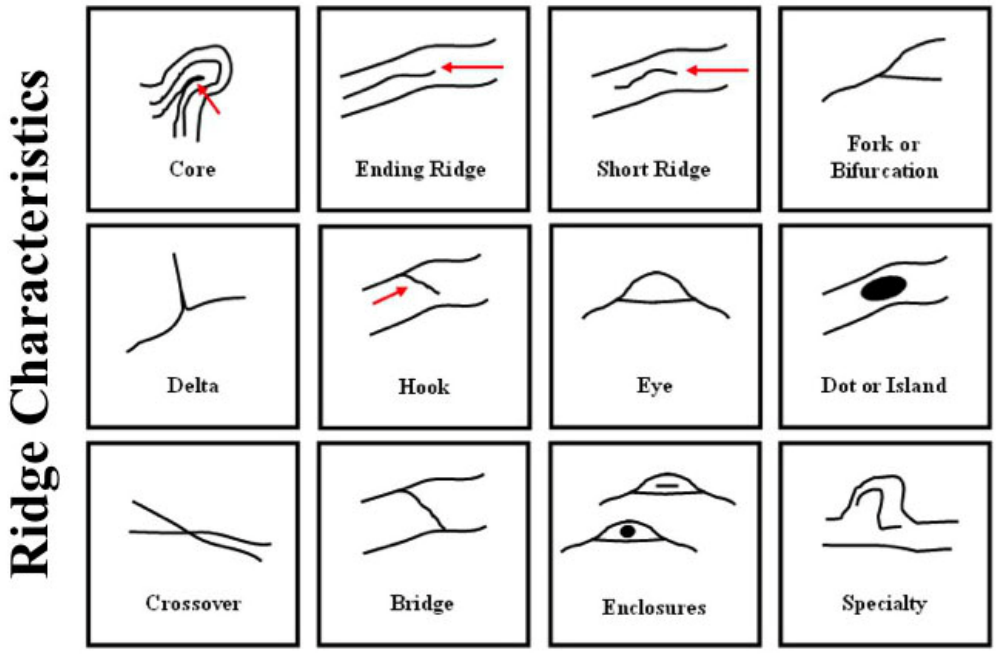


Figure 1 - Minutiae types

1.2.2 Sensors and image acquisition

Acquiring a representation of the fingerprint being analysed is the first step of the fingerprint comparison algorithm. A digital image is the desired medium in every modern system and it can be acquired in two different ways. A physical copy of the fingerprint can be produced (by applying ink, powder, or other marker) on a paper, or a similar, temporary surface, which is then scanned to provide the digital image. Alternatively, the digital image can be acquired directly from the finger itself, without need for the intermediate step – commonly called live-scanning, currently the preferred method in practically all biometrical systems. [4]

A sensor unit, also called live-scanner, is used to scan the finger surface and produce the image of the fingerprint. The scanning process can be based on a multitude of different methods, the most common being optical, capacitive, thermal and electric field sensors.

Some optical and ultrasound sensors are capable of acquiring images over a short distance (about 30-50mm), which reduces the problems caused by residue left over by repeated usage of the sensor. [4]

Depending on the system and overall purpose of the application, the following sensor properties should be considered:

- Image resolution and quality – important to avoid detection problems and image noise, but in general, any two ridge lines should distinguishable and minutiae possible to detect without errors – 500 dpi is the most commonly used sensitivity
- Scanner surface area – it should cover the majority of the fingerprint, so that a sufficient number of minutiae for comparison is detectable; a small surface would cut off important parts of the fingerprint, while an excessively large surface may include undesired background
- Overall size of the sensor unit – important for systems that try to integrate the sensor into a device, such as laptops, smartphones, etc.
- Reliability – given that the conditions are stable, the sensor should produce identical results with minimal error or image noise produced by the scanner unit itself
- Durability and lifespan – the choice of sensor materials, design and scanning method should reflect the system's practical usage and expected lifespan

[7]

1.2.3 Problems of fingerprint data processing

Ideally, a fingerprint image would be a black and white representation of the ridge lines, with no artefacts, in proper resolution, perhaps even in vector form. Realistically, there are multiple errors that can occur in acquiring and processing the image.

Quality of the fingerprint sensor is one of the factors in creating an image of sufficient quality. While extremely high image resolution is not necessary, problems may occur when using a low-resolution sensor – if a clear distinction between the ridges is compromised, false minutiae might be detected, while the real ones might be distorted and missed.

Adhering to the proper procedure when acquiring the image is also of high importance. If the finger is moved while the image is being taken may result in a blurred result. Repeated usage of the scanner without cleaning the sensor often causes problems, as the surface gets greasy and smudged. [8]

More importantly, position of the finger on the sensor and the force with which it is applied cause the most significant problems for the processing algorithm. Even if the same finger is used to create two images on the same sensor, but with different position and rotation, comparing them together proves problematic. Coordinates of the detected minutiae change, therefore positional and rotational matching must be implemented. The problem becomes more complex if the images are not of the same size and must be scaled appropriately, in addition to the operations mentioned above. The force with which the finger is applied also causes spatial distortions on the edges of the fingerprint. Ideally, the finger is applied with sufficient force to make the scan practically two-dimensional. If the finger is applied only lightly, however, the finger-scanner distance in the centre of the fingerprint and near the edges differs, resulting in the spatial distortion. On the other hand, thickness of the ridges can be increased by applying force, causing problems in methods relying on ridge properties, or even making the distinction between ridges difficult, if an extreme case. Detecting whether these distortions are present and either removing the afflicted area, or transforming the image to a proper 2D form can be very important for a successful matching result. [5][8]

Image noise and artefacts, common in all image processing systems, should be considered as well. The fingerprint is usually produced in a grayscale format and only a black-and-white representation is needed to detect the minutiae. The necessary pre-processing operations are therefore not as complicated as with other image processing problems.

Apart from the problems mentioned above, which may appear in images of the same finger, there is also a problem of inter-class variations – fingerprints from two different persons being classified as similar, despite high quality of the images.

All of this results in a problematic image that is difficult to be directly compared to the database entry. [5][8]

1.3 Fingerprint image classification methods

The base goal of classification is finding a set of features on the query image and database entry, comparing them and determining whether both images belong to the same person.

There are various types of features, or methods of comparison, to be found on fingerprints. The most common ones include:

- Minutiae
- Spatial correlation
- Ridge features

Every method has its own benefits and drawbacks, but combining them together is often possible and provides significantly improved results over classification based on one method only.

[5][8]

1.3.1 Method 1 – minutiae based matching

The most important step is detecting the minutiae of both the query fingerprint image and the database entry it is being compared to. The database fingerprint can have the minutiae pre-detected and stored for faster computing time in the final application.

The simplest way of comparing the images would be simply comparing the number of minutiae detected on both fingerprints. While this might work well enough for some small databases, it is not a sufficient factor on its own. Multiple persons with the same amount and type of minutiae may easily exist in the same system and it's the position of the minutiae, among other things, which makes their fingerprints unique. Matching the mutual minutiae position is therefore another crucial element of this matching method. [9][10]

A minutia point can therefore be described by its properties:

- Type
- Orientation
- Position

And for some hybrid methods, to avoid false positive matching of two similar minutiae, also:

- Ridge properties (thickness, quality, frequency)
- Information about the surrounding region [11][12][13]

Poor image quality, or physical changes on the finger (such as a new scar, not present in the database entry) may result in errors in minutiae detection. Spatial distortions and rotation should also be considered, as they may influence the minutiae distribution pattern.

The final verdict is passed based on the number of matching minutiae pairs between the query and database images. [8]

1.3.2 Method 2 – correlation based matching

Instead of detecting minutiae points, correlation based techniques focus more on spatial properties and patterns of the entire fingerprint.

Trying to correlate two images that are differently placed and rotated is very difficult, however. Rotating the query image and trying to match it to the database entry is necessary in that case, though it can increase processing time significantly, if the compared regions are too large. Spatial distortions also have to be taken into account when attempting a global correlation match. [14]

To avoid these problems, the fingerprints can be correlated in limited regions only. If a region with low image noise, few spatial distortions and rich on information crucial to the matching exists, True Positive Rate can be improved, as opposed to expanding the correlation area to the whole picture; at the price of increasing the amount of False Positive matches, as well. Grey level information in the region can also be used to improve the accuracy. Finding a proper region size for each query picture is therefore one of the main optimization goals of such a method. [8]

1.3.3 Method 3 – ridge-feature based matching

Ridge shape, thickness, orientation, frequency of the ridge lines and texture of a larger region are properties utilised by ridge-feature based matching techniques. [15]

The main benefit of this method is better robustness against poor image quality. A small artefact in the wrong place can influence minutiae detection more seriously than the overall ridge frequency or orientation.

If used on its own, however, this method cannot differentiate between two similar fingerprints from different persons. The ridge features are not unique enough and such a system would produce many False Positive matches. The method is better suited as a supplement for other methods in a hybrid system to improve their detection of poor quality images. [8][16][17]

1.3.4 Hybrid methods

As the individual problems of each method are evident, a system relying on a single method will have a hard time producing satisfying results. Many researchers have therefore attempted combining multiple methods together to counter-balance their weaknesses.

Beleznai et al. aimed to improve the basic minutiae matching with information from the area around the individual minutiae, using PCA analysis and Discrete Wavelet Transformation. [18]

Kovacs used a similar approach, comparing local minutiae regions, followed by triangular matching of the entire fingerprints. [19]

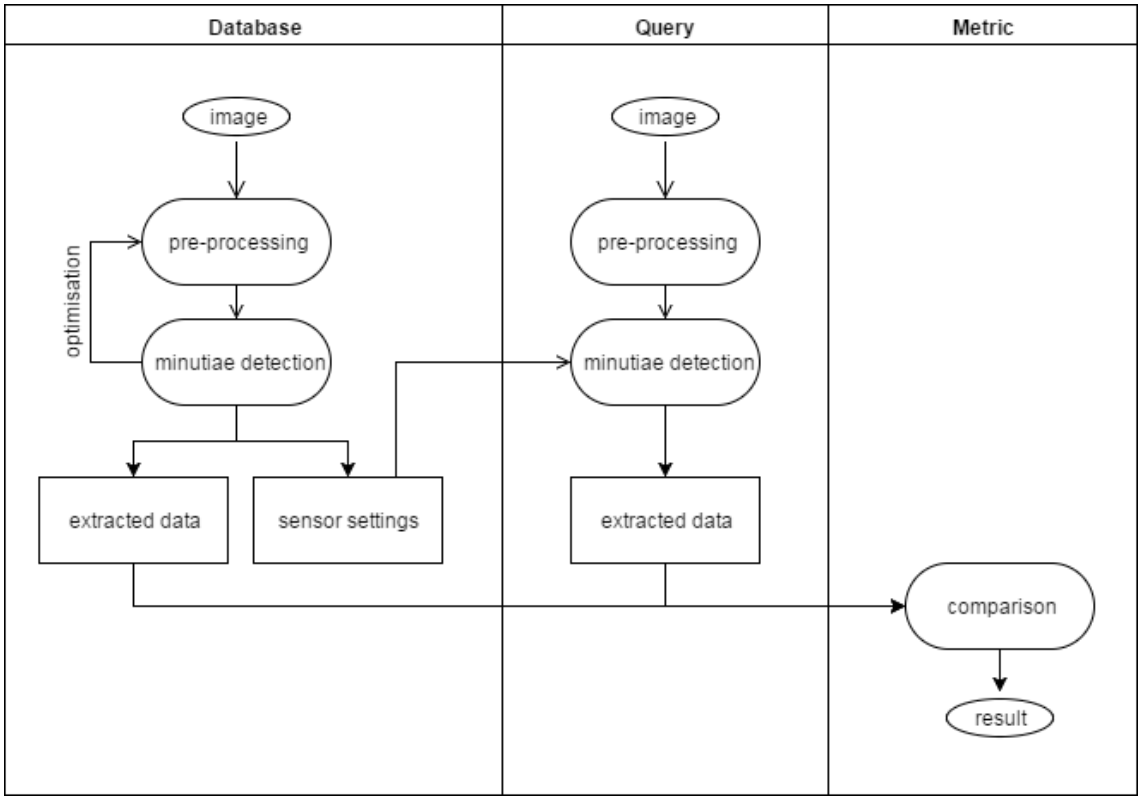
Nandakumar et. al. suggested a spatial correlation method that complements the basic hard choice minutiae comparison. They used region-based correlation to provide a better estimate of a correct match with better results than a simple minutiae-only matching model. [8]

Ross et al. proposed a similar approach with using ridge features in combination with minutiae detection, demonstrating an impressive improvement over a simple matching method as well. [20]

Bazen et al. incorporated machine learning to improve minutiae detection. The algorithm learns how to follow fingerprint ridges, spot irregularities and detect minutia points. [21]

Arivazhagan et al. used Gabor filters to map the fingerprint in different directions, starting from a central point detected in the image. [22]

2 PROPOSED METHOD



2.1 Overview

The project can be divided into three major sections: Database creation, Fingerprint comparison and Statistics.

In the first section, a set of fingerprint images is selected to serve as a database. Each image is then processed and a dataset of minutiae and their positions is extracted. Based on the expected outcome, the detection algorithm is optimised until a sufficient result for the specific sensor type used is achieved. The final datasets, as well as the optimised settings are stored in the database.

Specific steps of image pre-processing and minutiae detection are described in detail to explain the process of creating a minutiae dataset.

The second section describes a process of comparing an unknown query image to the database entries. Using the same detection process, query minutiae are extracted, using specific sensor settings. A decision metric is used to evaluate the similarities between the database and query datasets. This metric is improved by testing on a known sample of query fingerprints.

A final test with an independent sample of query images is performed to evaluate the accuracy of the method, the results of which are described in the third section. A discussion about the observed results and comparison with other works follows.

2.2 Fingerprint database creation

The fingerprint database serves a dual purpose: it stores user profiles with relevant fingerprint data, which is used to verify a user requesting access to the system and the images are also used as a train set for improving the database.

Normally, the database would be populated by fingerprints taken by the system it is being used for. The images would be taken by one (or a few varieties of) sensor that is expected to be later used to produce the query images. The database would therefore have fingerprints of a roughly equal quality and the same set of problems to be solved in pre-processing and detection blocks. While having sufficient intra-class variety – images of the same finger with different rotations, pressure and positon on the sensor – would be encouraged and beneficial to train a robust system, the properties of the sensor would be invariable, predictable, and thus easily avoidable.

While the goal of this thesis is to propose a general-purpose algorithm, it avoids using hard coded variables specific to a single image or sensor type. However, as it should be possible to identify a sensor type used to produce the query image, the user interface provides an option to select an optimised set of settings for a specific sensor type.

The database images were taken from different online sources freely available for research. These include commercial developers of fingerprint software, providing free samples of their databases for research, or fingerprint competitions' source files.

Fingerprint Verification System offers 168 fingerprints in bitmap format and 256x256 resolution. It includes 8 impressions for each finger, allowing for both intra- and inter-class variation. The quality of the images is good, although an occasional image artefact occurs. The fingerprints were taken on the same sensor, which makes their pre-processing easier, should they be modified as an entire set.

http://fvs.sourceforge.net/index.html

Figure 2 - Fingerprint Verification System database samples

Neurotechnology.com provides two sample databases for download in .tiff format. VeriFinger sample database consists of 408 fingerprints taken with Cross Match Verifier 300 scanner at 500 dpi. It includes 5-6 different fingers from 9 users, each with 8 scans. The images are in a very good quality, high contrast and redundant background sections cropped. U-are-U database includes 520 images from 7 users, taken at 500 dpi with DigitalPersona.U.are.U 4000 scanner. Some users provided up to 10 different fingers, with 8 scans each. The scanner apparatus is visible on the edges, sometimes overlapping with the fingerprint itself, a small, square-shaped region in the centre of the image increases intensity of the underlying background and residue from the previous scans is sometimes visible, but the ridge lines are mostly in high contrast and well distinguishable.

http://www.neurotechnology.com/download.html

Figure 3 - Verifinger (row 1) and U are U (row 2) database samples

UPEK Fingerprint Database is a collection of… , available for download directly from Advance Source Code webpages. The database includes…

http://www.advancedsourcecode.com/PNGfingerprint.rar

The databases mentioned above are well suited for comparison, as only little pre-processing is necessary and no significant problems are expected. The following databases are taken from fingerprint verification competitions and provide much more problematic fingerprints with multiple errors. Image quality may be poor; contrast between the ridge lines and the background may be difficult to discriminate, with both white and dark grey background against black ridges, or segments of high-contrast, dark ridges and segments of low-contrast, barely recognizable, soft grey ridges, making global contrast enhancement ineffective; strong residue from previous uses of the sensor may be

present; fingerprints may be positioned improperly, with parts cut off by the sensor edge; movement blur, or high pressure on the scanner surface causing ridges in a segment of the image to merge together; low pressure on the sensor causing ridges to be barely discernible from the background, with darker spots where the pressure was sufficient – resulting in dashed or dotted lines, instead of continuous. Due to these problems, the databases are very difficult to process, but may prove valuable in training the system to be more robust in dealing with similar problems in real fingerprint images.

The bi-annual Fingerprint Verification Competition provides four databases of fingerprints taken with different scanner systems (one of them being synthetically generated fingerprints). Each database provides 8 impressions per each finger, with 100 fingers reserved for testing the competitors' programs (set A – 800 images) and 10 fingers (set B - 80 images) available for download, apart from FVC 2006, which includes 12 impressions per finger.

Fingerprint Verification Competition Database Properties					
	Sensor Type	**Image Size**	**Set A (wxd)**	**Set B (wxd)**	**Resolution**
Fingerprint Verification Competition 2000					
DB1	Low-cost Optical Sensor	300x300	100x8	10x8	500 dpi
DB2	Low-cost Capacitive Sensor	256x364	100x8	10x8	500 dpi
DB3	Optical Sensor	448x478	100x8	10x8	500 dpi
DB4	Synthetic Generator	240x320	100x8	10x8	about 500 dpi
Fingerprint Verification Competition 2002					
DB1	Optical Sensor	388x374	100x8	10x8	500 dpi
DB2	Optical Sensor	296x560	100x8	10x8	569 dpi
DB3	Capacitive Sensor	300x300	100x8	10x8	500 dpi
DB4	SFinGe v2.51	288x384	100x8	10x8	about 500 dpi
Fingerprint Verification Competition 2004					
DB1	Optical Sensor	640x480	100x8	10x8	500 dpi
DB2	Optical Sensor	328x364	100x8	10x8	500 dpi
DB3	Thermal sweeping Sensor	288x480	100x8	10x8	512 dpi
DB4	SFinGe v3.0	288x384	100x8	10x8	about 500 dpi
Fingerprint Verification Competition 2006					
	Sensor Type	**Image Size**	**Set A (wxd)**	**Set B (wxd)**	**Resolution**
DB1	Electric Field Sensor	96x96	140x12	10x12	250 dpi
DB2	Optical Sensor	400x560	140x12	10x12	569 dpi
DB3	Thermal sweeping Sensor	400x500	140x12	10x12	500 dpi

Figure 4 - examples of problematic images from FVC databases

2.3 Image pre-processing

A script was written in the MATLAB environment to test various approaches to image enhancement and pre-processing. The main goal of pre-processing is to remove (ideally) all image noise, artefacts and unnecessary information. Image background should be suppressed, but important properties of the ridge structure, such as ridge thickness and all minutiae, should remain unchanged.

2.3.1 Histogram equalisation

Image histogram describes overall pixel intensity count, typically in values from 0 to 255, or from 0 to 1 in images with normalised intensity. Images with pixel intensity values not distributed over the entire histogram range could benefit from transforming their intensity.

By applying *histeq* function on the original image, its intensity is adjusted to be approximately evenly distributed, with larger gaps between individual intensity values. As a result, differences between areas with very similar intensity are increased, which also makes designing thresholds for other pre-processing tasks, or detection methods easier.

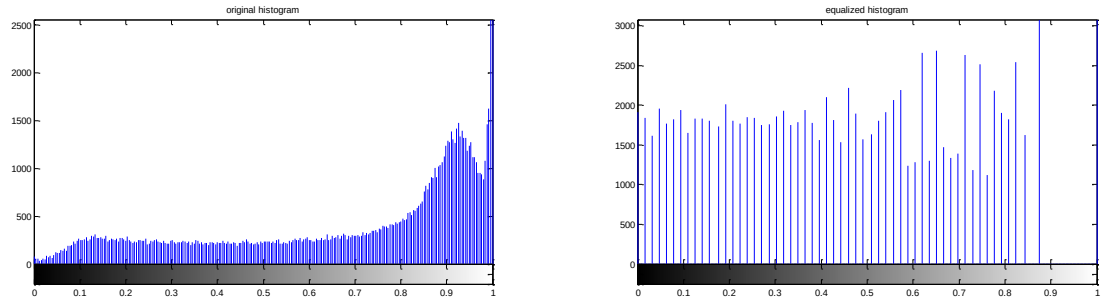


Figure 5 - image histogram before and after equalisation

2.3.2 Gamma correction

While histogram equalisation is usually a linear transformation, gamma correction is not. Intensity of each pixel of the processed image is raised to the Xth power (where X is the gamma setting). With gamma between 1.5-3.5, low intensity pixels are suppressed, while higher intensities remain unaffected. This effect is very beneficial to fingerprint images, as it retains all properties of ridge lines, while removing the background.

Fingerprint images typically have dark ridge lines and bright background, however;

strong gamma correction would suppress the dark ridges, instead of background. Therefore, the image intensity is inverted, gamma correction is applied and intensity is inverted again.

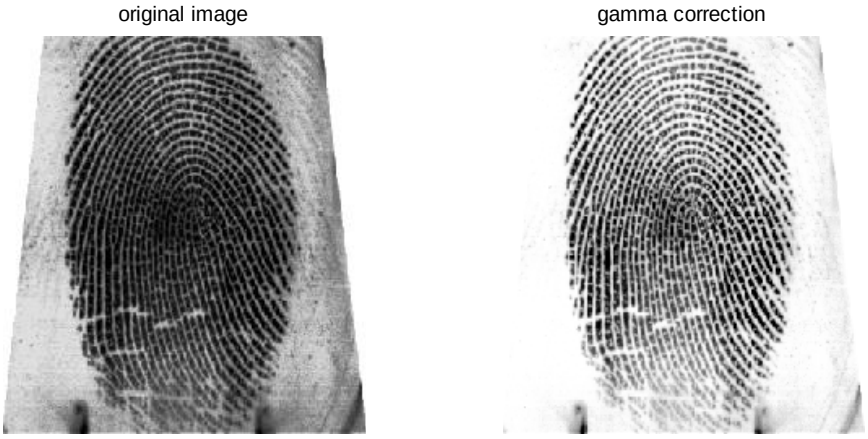


Figure 6 - image before and after gamma correction

2.3.3 **Image sharpening**

To improve detection of ridge lines running too close to each other, sharpening the contrast of ridge edges may be beneficial. This is achieved by filtering the image by a two-dimensional convolution of the image and a mask.

Averaging (2.1), or Laplacian (2.2) mask operators unsharpen the image by replacing each pixel by an average value (averaging mask), or 2nd derivative (Laplacian mask) of pixels in the surrounding area. The unsharpened image is subtracted from the original to produce a representation of sharp features of the image. By adding this representation to the original image, sharp features are enhanced.

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \quad (2.1) \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.2)$$

The image can be directly sharpened by using a local sharpening mask. The central element of the mask can be modified to increase, or decrease rate of the sharpening, but in that case, the rest of the elements should be normalised (sum of all elements should equal one) to retain the energy of the image:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2.3)$$

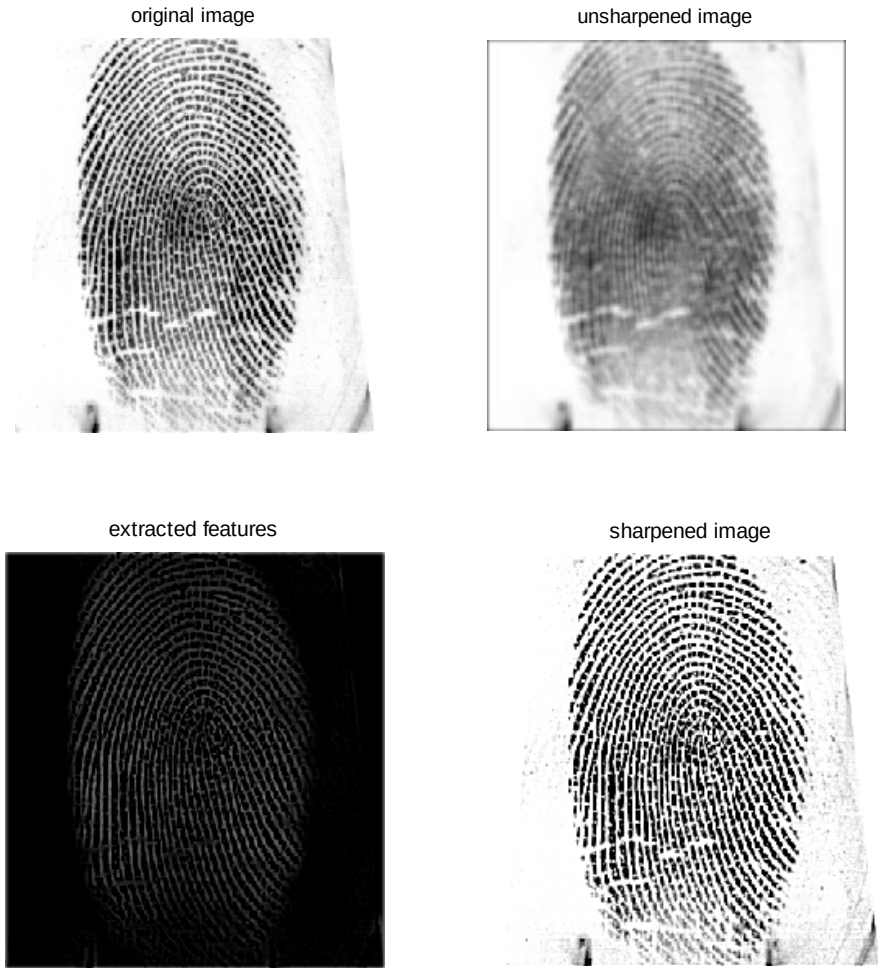


Figure 7 - image sharpened with an averaging mask

While this method might improve results of minutiae detection, it may also supress some ridge features, such as ridge thickness or grey-level information, important for other fingerprint detection methods. In future work, it would be more beneficial to make a copy of the original image and sharpen it right before the minutiae detection step, leaving a copy of the original image with unaffected ridge information for alternative methods.

2.3.4 **Median filtering**

Similar to the sharpening methods, a mask operator moves across the image and replaces the central pixel by median value of the pixels in the mask. This method is used to smooth over small inaccuracies, salt&pepper image noise, or rough, sharp shapes. It can be used in fingerprint images to improve ridge line appearance, form them into more solid, continuous lines.

Proper mask size is very important, as the thickness of ridge lines is usually similar, or greater than valley (space between ridges) size. Even a slightly larger mask could easily blur parallel ridge lines together, or obscure important minutiae. Due to many fingerprint images having insufficient resolution, or very thin valley width, median filtering is often counter-productive.

2.3.5 **Thresholding**

Similar to previous methods, thresholding transforms part of the intensity spectrum of the image. Pixels with intensity higher (or lower) than a specified threshold are set to a specific value. For the purposes of fingerprint image processing, the high-intensity background is suppressed by declaring a threshold of ~0.6-0.8 and setting all pixels above the threshold to 1. The rest of the pixels are suppressed to 0, resulting in a black&white binary image, which subsequent methods require as input.

Figure 8 - image before and after thresholding

2.3.6 **Object removal**

After binarization of the image, a function `bwareaopen()` is used to remove white objects (pixels with a 4 or 8 directional connectivity) of a specified size. First, the small imperfections and artifacts within the ridge lines are removed in this way. After a negative transformation (switching black and white pixel values), the same function is used to remove unnecessary residue, typically at the edges of the fingerprint, or around the main detection area.

Figure 9 - image before and after object removal

The resulting black&white image is ready for minutiae detection.

2.4 Minutiae detection

The purpose of this block is to create a minutiae dataset from the pre-processed image. The detected minutiae are described by their type and relative position and distance, making comparison of two images taken by different scanners more accurate. The minutiae dataset is the only output of this block; the original image is irrelevant for further classification.

The biggest challenge of minutiae detection is recognising the different minutiae among the ridge structure that changes directions, line thickness, may be impacted by artifacts and other problems. A global mask operator may have problems with adjusting to the variable direction of the ridge lines, a different method is therefore proposed.

An operator, similar to a floating mask (therefore referred to as ‘mask’ further below), is placed at a beginning of a ridge. This operator is, besides many other parameters, defined by its position, direction in which it moves and a size of the surrounding area it analyses. The mask uses the amount of white pixels in different segments within its area to decide on the direction of the next move, progressing along the ridge line and dynamically adjusting its size to envelop the entire width of its ridge without interfering with the neighbouring ones.

Initially, only a single mask is created at the edge of the fingerprint, but it is capable of replicating and placing the copies on other ridge lines detected in its vicinity. In this manner, within a few steps, an entire side of the fingerprint is covered by masks running in parallel. Based on their mutual interactions, minutiae are detected – masks diverging and detecting a beginning of a new line between them, a mask running out of its ridge at the end, or two masks colliding at a fork.

Due to the need to create multiple copies of the masks with the same types of parameters and behaviour, yet able to act independently, an object-based approach to model the mask behaviour was considered most efficient. All masks are created as instances of one class, which defines all parameters and functions relevant to mask operations.

The detection itself, from taking the pre-processed image as input, until providing the minutiae dataset as a result, is done by the function `detect()`. Masks are created and their actions executed from here, progressing through the image until all masks finish following their respective ridges. A few auxiliary functions related to the process, but not directly involving a mask, or dealing with multiple masks at once, are also provided in the same folder.

A visualisation showing the fingerprint image and all active masks with their current area highlighted was also developed. The parts of the ridges being detected are deleted in the image, showing a real-time sense of progress. The visualisation significantly increases computing time, it was therefore disabled by default in the final user interface. It is possible to enable the display via the interface and set a sleep timer after each frame, or number of frames to be skipped before the visualisation starts.

The method detects four most common types of minutiae: *starting, ending, left fork* and *right fork*. The remaining minutiae types can be described as combinations of these four basic types.

2.4.1 **MaskClass**

Definition of mask properties and methods relevant to their actions is provided within this class. A few properties are set as default or predefined, some are loaded from an external file (if a scanner setting is selected) and others are initialized from within other methods once they become relevant.

An instance of this class travels along a single ridge line, checking its surroundings and deciding on the direction of the next step at every iteration. The basic parameters are the coordinates of the central point (`x,y`), a size of the area it detects (`mask_size`) and the direction of the last movement (`direction`), but it is also able to store coordinates of its previous positions (`trace`), the amount of background pixels it travelled through, therefore running out of its original lane (`black`) and a host of other variables used to check thresholds, or minimal/maximal allowed values.

The `direction` is defined by an integer from 1 (moving right), continuing anti-clockwise (2 = up+right, 3=up, 4=up+left, …) until 8 (down+right).

Two static methods are used for operations shared between instances of the class, or related to mask movement and position:

Method `img()` stores a copy of the original image input of `detect()` that every other `maskClass` object can access, change and rewrite.

Method `template()` creates and stores matrices used to compare the area around the central point. The template approximates a 90° field of view in one of 8 directions. The method is able to produce full-sized templates on demand, but due to masks detecting unwanted pixels of neighbouring lines during diagonal movement in the corners of the templates, a predefined set with rounded corners, better approximating a circular shape around the central point, is loaded from a .mat file.

The rest of the methods are related to object creation, movement, or size adjustment.

`maskClass()` is the constructor method, called when a new instance of the class is created. The first instance is called with the original image for input, to initialise the `img()` static method, but the subsequent instances are created either as copies of existing masks, or without `img` in the constructor inputs, to avoid replacing the already-modified `img` shared between all instances of `maskClass`.

`normDir` and `change_size` are simple methods used to modify size of the object, or normalise `direction` to a 1-8 range and recalculate all relevant properties.

`check_direction()` is used to decide the direction of the next movement. As fingerprint ridges don't naturally contain sharp turns within the space of a few pixels, `check_direction()` is usually called only for the last direction used, +/- one direction to each side (effectively covering a 180° area). A `template` with corresponding direction and size is compared to the current object area and the number of white (ridge) pixels in the current area, covered by the template, is counted. The template/direction with the highest intensity of pixels is chosen as the new object direction.

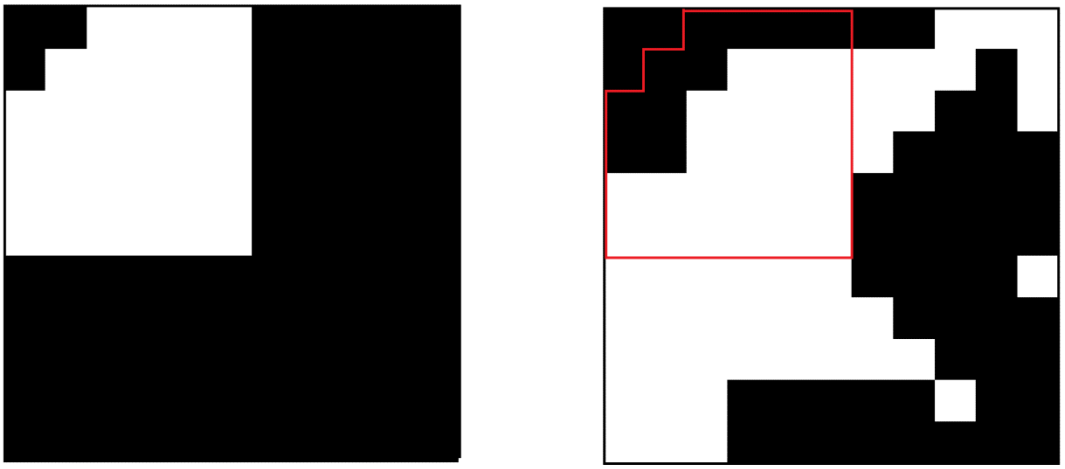


Figure 10 - directional template and overlap with the current mask area

`move()` changes the x and y coordinates of the central point of the object, based on the current direction, then also updates trace with the new position, calls `cleanup()`, if it's enabled and checks for background pixels under the central point (indicating that the mask is moving outside of the line).

`cleanup()` deletes the area behind the mask, so that new masks are not added there by accident, or it doesn't interfere with other operations. The changes are saved in the

`img()` shared between class objects, but won't affect the original image. The method can be disabled.

`adjust_size()` attempts to recognise if the mask is too large, therefore detecting parts of neighbouring ridges, or too small and unable to encompass the full width of its own ridge, and modifies the mask size accordingly.

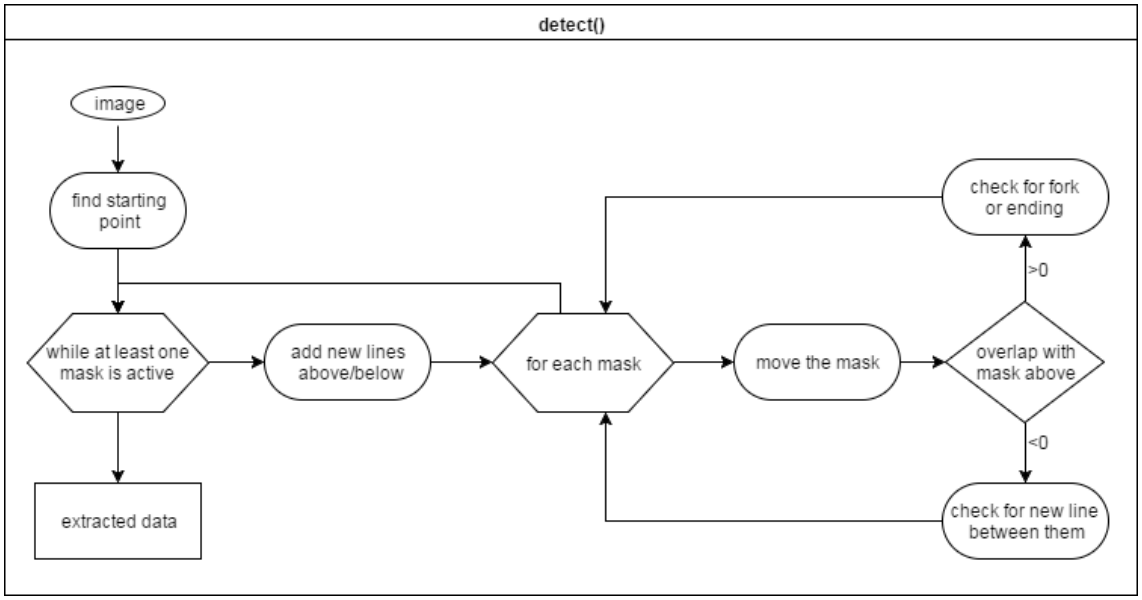
`borderCheck()` serves to analyse the pixels on the edge of the current object area, in the current direction. An empty border indicates a possible ending of the ridge, which the method confirms by creating a copy of the mask and moving it a few steps in the current direction to verify that the black border is not an artifact, but a true ridge ending.

`findMask()` is used by the first and last mask to find new ridges outside of the area currently covered by masks, or to find a line beginning between two masks whose mutual distance increased beyond the reach of their current areas. The method takes two sets of coordinates as input and searches the area of the image between these points for objects. If an object, apart from those at the edges (belonging to the ridges already covered by masks) and large enough to be considered a ridge is present, a new mask is created there.

`drawMask()` is a simple auxiliary method to draw a red square around the current object area, used for visualization of mask movements and size adjustments in the image.

Using these methods and properties, any instance of `maskClass` can be moved backwards/forwards, adjust its size to fit the ridge it is following, the trace coordinates can be used to plot its path, or return the mask to a previous position.

2.4.2 **Detect**



The function detect() makes extensive use of the maskClass to scan the input image for minutiae.

The first mask (instance/object of maskClass) is placed on the edge of the fingerprint, the exact position makes little difference. In the first part of each infinite while loop iteration (which breaks only once all masks have been finished), the top and bottom masks inspect the area directly above/below: if a ridge is detected, a new mask is created there. In this way, the first mask eventually replicates itself on all ridge beginnings.

In order to prevent some masks from advancing faster than their neighbours and causing problems with mask-to-mask interactions, a simple linear trendline is calculated using all mask coordinates. If detection visualisation is enabled, this trendline is drawn in blue. The masks behind the trendline (above/below/right/left, depends on the initial direction) are allowed to move, while the others are forced to wait. In case none of the masks move for some reason, the trendline is moved ahead by one pixel.

The movement is done by calling `check_direction()` first, followed by `move()` and `adjust_size()`. If the mask moved over the black background for too many steps, it is classified as finished, *ending* minutia is updated and the mask is deleted from the active set – provided that the total length of the ridge was sufficient to not be considered an error, artifact, or a cluster of random pixels.

The mask position is then compared to the previous mask in the sequence. An overlap of their active areas is computed to spot an unusual behaviour:

- If the overlap decreases and masks diverge, it is possible that a new line

begins between them – a verification of whether it is an independent line, or a *fork* of an opposite direction, connected to one of the masks, is necessary
    - Auxiliary function `sameLine()` serves to compare two or three points in the image and provide a decision on whether they are a part of the same line or not. It is used to ensure that newly created lines are not placed on the same line as an existing mask, or to differentiate between *ending* and *fork* minutiae.
- If the overlap increases over a specific threshold, the masks are colliding and either a *fork* minutia is present, one of the masks made an error, or its ridge ended and has to be finished

As it is possible that a mask is added in a wrong order, which results in wrong masks comparing their proximity, assuming that they are neighbouring each other , yet in reality are not, every five iterations a function to reorder the masks is called.

The output of the `detect()` function is a set of minutiae coordinates. Each minutia is described by its type and coordinates. Coordinates of the centre of the fingerprint and a maximum distance from the centre to the edges of the fingerprint are part of the output. Minutiae coordinates are later converted into normalised vector distances and angles, during image comparison.

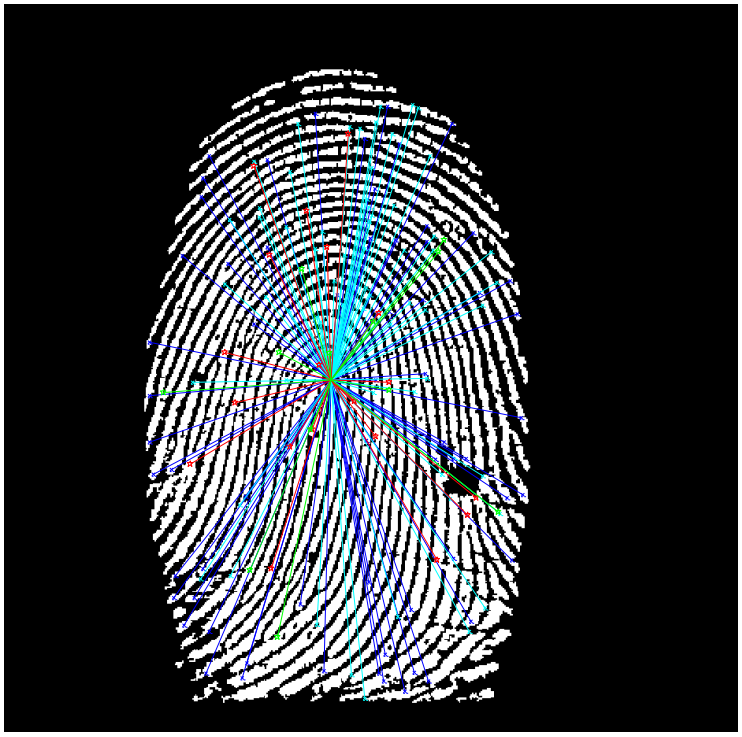
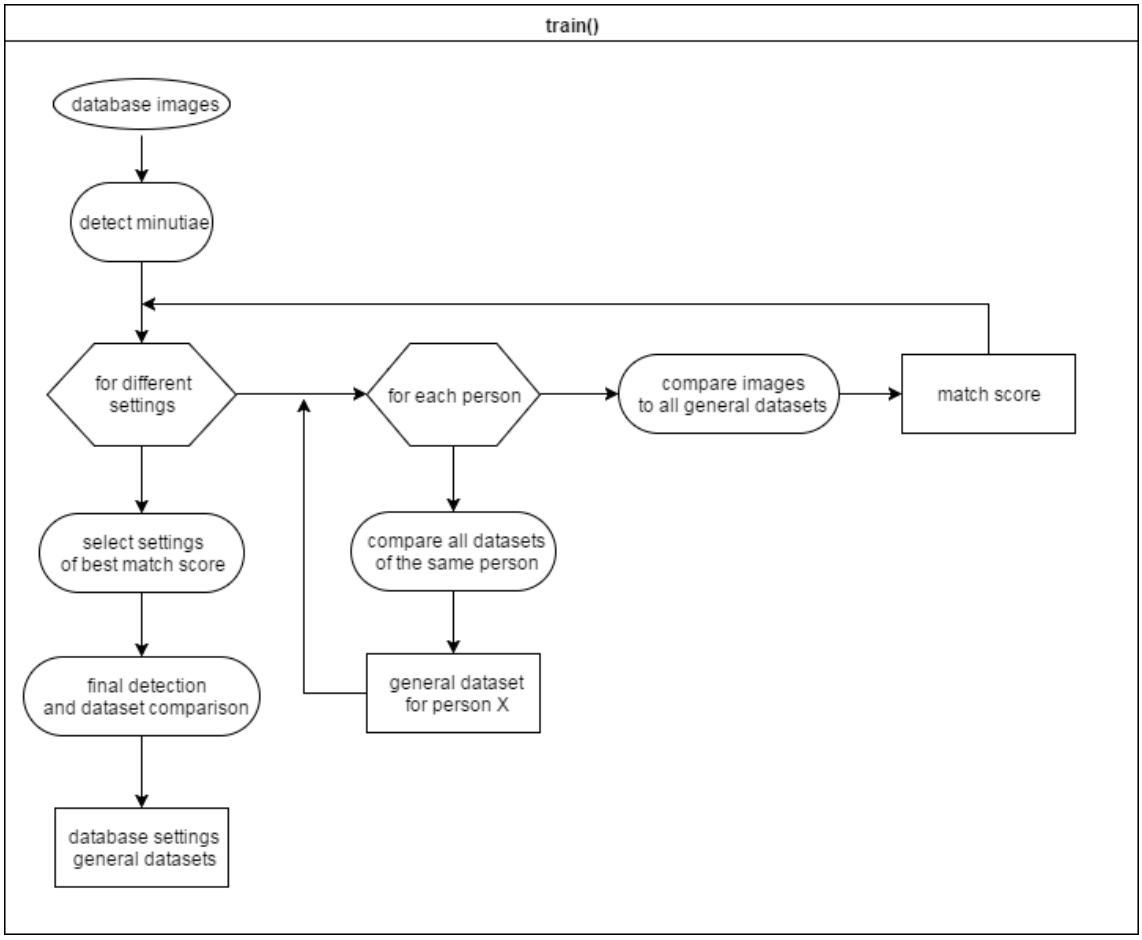


Figure 11 - Visualisation of the detected minutiae

2.5 Database optimisation



Once the database images are selected and a set of minutiae is extracted from each of them, the images from the same sensor type are grouped together and optimised settings for that specific sensor are determined.

The minutiae detection relies on multiple variables to successfully differentiate between a minutia and a random error in the image. Image pre-processing and detection setting adjustment, such as such as minimum/maximum mask size, minimum ridge length, or number of black pixels allowed within a ridge, is often necessary to attain sufficient ridge gaps, even if some errors are still present.

In order to achieve sufficient quality, each set of database images is processed with different settings until a desired outcome (or outcome close enough) is produced. This process is computationally difficult and extremely long, but only needs to be executed once and won't affect practical usage of the system (query comparison).

Any change in the pre-processing or detection settings requires a re-calculation of all database image entries. The updated datasets are then used to test the new change by

comparing every dataset to all other entries and evaluating the accuracy of the best match. After testing all changes, a setup minimizing false matches while maximizing the correct ones is selected.

Once the optimal settings for a specific sensor type are achieved, they are stored in the database, along with a final data extract of the relevant images. When the user requests comparison of a query with the database, an option to select a sensor type is given to increase the accuracy of query processing.

2.6 Query image detection and comparison

With the database images processed and optimised, the system is ready for use. Once a user provides a query image, this image has to be processed and minutiae detected with the settings obtained from database training.

Database sets of extracted data are loaded and compared with the result of query image processing. A comparison metric evaluates the similarities and differences of the images and provides a verdict – match or mismatch with an estimate of accuracy based on the match score.

As each minutia in the dataset is described by a vector from the fingerprint centre to the minutia coordinates, two minutia points from different datasets are compared based on the angle between these vectors and distances of each vector. Both the vector and the distances are normalised to a value from 0 to 1; the angle is divided by pi, the maximum possible value, the distances are divided by the longest distance from respective image core to its edge. A matching score between two minutiae points is calculated as a sum of difference between their normalised distances and the normalised angle.

$$score = \left(\frac{distance\ A}{\max distance\ A} - \frac{distance\ B}{\max distance\ B}\right) + \left(\frac{angle}{\pi}\right) \quad (1)$$

Minutia score can therefore range from 0 for a perfect match, to 2 for completely opposite minutiae. This approach should make comparison of two images with different resolution more reliable.

Each minutia from the first dataset is compared to each minutia of the same type from the second one and the highest score is saved. To prevent an image with a large number of minutiae with poor score from achieving a better overall score as an image with fewer minutiae detected, but of higher quality, the final match score of the two images is calculated as a sum of the 10 best minutiae scores.

The query image dataset is compared to all database entries and the entry with the best match score is selected, the person who the entry belongs to is displayed in the GUI as the closest match, along with the match score.

2.7 Graphical User Interface

To improve user interaction with the various methods and functions of the project, a simple interface was designed. Before the database training process, two images per person were removed from each database to serve as a testing sample. The user can select one of the test images to be processed, choose scanner settings to be applied or adjust visualisation settings.

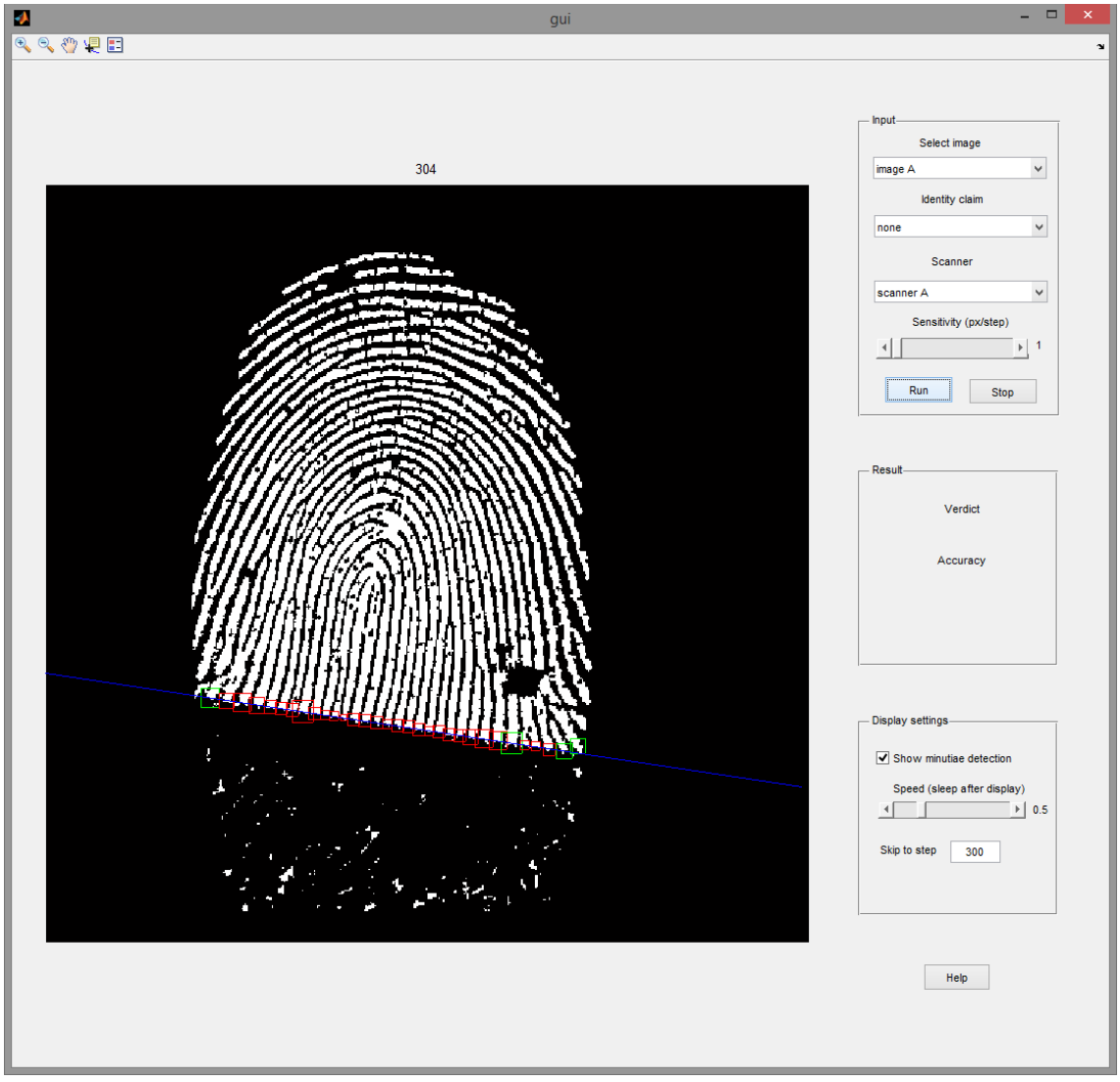


Figure 12 - GUI with image detection in progress

In real life applications, where the program processes direct output of a scanner device, the resolution, image quality and all resulting problems are known. This is simulated by allowing the user to select a scanner from the third combo box, using the optimized settings for that image type during pre-processing and detection. If no scanner selection is done, default settings are used.

The masks during detection are allowed to move by one pixel per iteration, by default. This approach is more accurate, but also significantly more time consuming. The user is able to increase the mask step distance, though the feature is highly experimental. Step distance of 2 significantly cuts down processing time and the results are still somewhat acceptable, but higher step values cause further errors and mistakes in detection.

A visualisation tool of the detection process is shown in the left section of the GUI window. The image after pre-processing is displayed, with current mask positions marked by their detection area in two different colours: green squares for newly created masks and red squares for masks that fulfilled the minimum step requirement. The image also shows lines being deleted by masks moving over them – this change is only applied to the internal copy of the image that all masks share. A trendline displayed in blue, which determines which masks are allowed to move and which have to wait in the current iteration is also visible.

After displaying the current mask position in each iteration, a sleep timer is applied – increasing or decreasing its value by a slider in the GUI can speed up the visualisation, or slow it down to allow the user to notice the details. Part of the visualisation can be skipped by editing the ‘skip to step’ box to a value of the iteration to be skipped to.

The visualisation itself requires significant part of the total processing power, resulting in a very long process – especially if the sleep timer is set to higher values. Should a user wish to omit the visualisation a checkbox disables it and causes the detection to skip directly to the result. The process still takes a rather long time to finish, but is a fragment of the full time with visualisation enabled.

The visual result of the detection is an image with all detected minutiae displayed in different colours and a line drawn from the approximate core location to their coordinates. Minutiae of type ‘starting’ and ‘ending’ are marked as a blue or cyan cross, ‘forks’ are marked with red or green stars.

2.8 Statistical analysis

The VeriFinger database was selected for final testing. The database consists of fingerprints taken from 9 persons. Each person provided multiple scans of each finger, from 22 to 48 images per person. For every person, two images were removed from the database, to serve as a test sample. All database optimisation was done on the remaining 390 images.

Due to extremely long processing time of all database images, other databases were excluded from the final testing. The test sample images were compared to all database entries with a 1:1 comparison. If the query source and target database entry belonged to the same person, the result would be classified as True Accept/ False Reject, otherwise as False Accept/True Reject.

False Accept Rate is calculated as the amount of False Accept results / all false identity matches (comparing person A to a different database entry) and expresses the ability of the system to reject false identities.

False Reject Rate, on the other hand, represents the probability of rejecting a valid user. It is calculated as False Reject results / all same identity matches.

True Accept: 5

True Reject: 131

False Accept: 13

False Reject: 13

False Accept Rate: 13/(13+131) = 9.03%

False Reject Rate: 13/(13+5) = 72.22%

2.9 Discussion

The statistical results of the proposed algorithm are clearly inferior to other works in the area.

Minutiae detection is not as reliable as needed, mostly due to errors still present in the image after pre-processing. Fork minutia detection is based on the two lines belonging to the same object – small image artifacts that persist after pre-processing can easily cause both false positives and false negatives. Mask interactions work best when aligned horizontally and moving along vertical lines, but once the ridge direction changes to mostly horizontal and masks from opposite directions start to collide, wrong mask placements or minutia classifications appear more frequently.

Another serious disadvantage is the long processing time of the detection. There's a large amount of processes involved in mask movement and interactions, some of which may be optimized for faster performance, but the overall time would still be too long for a practical use when the query image has to be taken, processed and a verdict provided within a few seconds.

The detection process could be improved by adding morphological operations to the image pre-processing to improve the area between ridges, especially close to image centre, remove artifacts still present in the image and remove the few-pixel wide connections between ridges that may cause false fork minutiae detections. These image-enhancing operations could be performed locally, a few steps ahead of the masks currently moving over the area.

A method of reliable fingerprint core detection and general direction of the ridge lines would improve the current comparison metric to be less reliant on image position and rotation, therefore improving the comparison results between images. A second reference point, such as delta, to base the main fingerprint direction and distance on would be ideal, but not all fingerprints include a core or delta points, therefore a sufficient replacement would have to be found. An elliptical approximation of the fingerprint could be constructed, from which the main axis could be calculated, although the general shape of the fingerprint depends more on the rotation of the finger and pressure applied, rather than the fingerprint itself.

Additional ways of improving the results could include implementing secondary attributes, such as grey levels or ridge thickness from the original image. Additional minutiae types could also be detected, or reconstructed from a combination of existing ones. Scars in the fingerprint, detectable as series of minutiae of the same type, could be used in the comparison, not as a primary criterion, as it could be a result of a recent

accident, not present in the database images, but as a supplementary attribute.

Despite these shortcomings, the basic framework of mask objects works as intended. The masks are able to follow the ridges and adjust their size accordingly. They are able to detect new lines starting at their current level, interact with the adjacent masks and delete themselves once they run out of their current line.

Moving, copying or removing the masks is very simple, new methods can be added to the class and applied to all masks to improve minutiae detection, or add additional attributes to analyse. The masks store enough information to trace their paths, recreate situations from several steps ago, or visualise most of the actions occurring during detection. Multiple functions or segments of the code were developed only to more effectively debug the software.

From a personal view, this project was the first opportunity to apply object oriented programming, or rather a proper combination of object oriented and more classical, functional oriented elements, in a truly meaningful way. Trying to replicate work of other researchers of this topic would probably deliver better statistical results, but designing an individual solution and trying to reach a conclusion was of higher importance.

CONCLUSION

The goal of this project was to study previous work on fingerprint processing and general background of biometrics. Multiple scientific papers on the topic were reviewed, in which three major methods were observed – minutia, correlation and ridge-property based detection. The minutia-based method was chosen as the core method of future work, with exploring possibilities of hybrid combinations with other methods.

A fingerprint image database was constructed from multiple sources. The collection includes databases of high-quality fingerprints suitable for direct comparison, databases with minor problems that would benefit from basic image pre-processing, as well as databases with highly problematic images. Due to time limitations, however, only one database was fully processed and used in the final testing.

Image pre-processing algorithm was designed in MATLAB environment to showcase useful image enhancement methods for fingerprint images.

Detection of minutiae points in the processed image was done by constructing a set of objects scanning the ridge lines and using their proximity and current area to classify points of interest. A set of minutiae, described by relative distance and angle was produced as a result. Afterward, a comparison metric processed the detection results from a database image and a random query image and produced a decision – match or mismatch.

Results of the final testing were dissatisfactory, but many ideas for improvement were described in the Discussion chapter. Despite the problems with detection and image comparison, the main mask object framework is a success.

REFERENCES

[1] DRAHANSKÝ, Martin and ORSÁG, Filip. *Biometrie*. 1. issue. [Brno: M. Drahanský], 2011. 294 s. ISBN 978-80-254-8979-6.

[2] JAIN, A.; HONG, L. and PANKANTI, S. Biometric Identification. In: *Communications of the ACM*. 2000, 43(2):91–98. DOI 10.1145/328236.328110

[3] JAIN, A.; BOLLE, R. and PANKANTI, S., eds. Biometrics: Personal Identification in Networked Society. In: *Kluwer Academic Publications*, 1999. ISBN 978-0-7923-8345-1.

[4] RAK, Roman and Filip ORSÁG. *Biometrie a identita člověka ve forenzních a komerčních aplikacích*. 1. iss. Praha: Grada, 2008, 631p. ISBN 978-80-247-2365-5.

[5] RITUL, Matish Garg. A Review on Fingerprint-Based Identification System. In: *International Journal of Advanced Research in Computer and Communication Engineering*. 2014, Vol. 3, Issue 3, pp. 5849-5851.

[6] GALTON, Francis. *Finger Prints.* London: Macmillan & Co., 1892.

[7] RUTTKAY, M. *Biometrická identifikace otisku prstu*. Brno: University of Technology in Brno, Faculty of Electrical Engineering, 2015. 70 p.

[8] NANDAKUMAR, K. and JAIN A. K. Local Correlation-based Fingerprint Matching. In: *Proceedings of ICVGIP*, Kolkata, 2004.

[9] HONG, L. *Automatic Personal Identification Using Fingerprints*. PhD thesis, Michigan State University, East Lansing, U. S. A., June 1998.

[10] JAIN A. K., PRABHAKAR, S., HONG L., and PANKANTI, S. Filterbank-Based Fingerprint Matching. In: *IEEE Transactions on Image Processing.* 2000, 9:846–859.

[11] RATHA, N. K., KARU, K., CHEN, S. and JAIN, A. K. A Real-Time Matching System for Large Fingerprint Databases. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1996, 18(8):799–813.

[12] JAIN, A. K., HONG, L., and BOLLE, R. On-line Fingerprint Verification. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997, 19(4):302–314.

[13] JAIN, A. K., PRABHAKAR, S., and CHEN, S. Combining Multiple Matchers for a High Security Fingerprint Verification System. In: *Pattern Recognition Letters*. 1999, 20(11–13):1371–1379.

[14] PANKANTI, S., PRABHAKAR, S. Prabhakar, and JAIN, A. K. On the Individuality

of Fingerprints. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002, 24(8):1010–1025.

[15] ROSS, A., JAIN, A. K., and REISMAN, J. A Hybrid Fingerprint Matcher. In: *Pattern Recognition*, 2003, 36(7):1661–1673.

[16] JAIN, A. K., PRABHAKAR, S., HONG, L., and PANKANTI, S. Filterbank-Based Fingerprint Matching. In: *IEEE Transactions on Image Processing.* 2000, 9:846–859.

[17] WILLIS, A. J. and MYERS, L. A Cost-Effective Fingerprint Recognition System for Use with Low-Quality prints and Damaged Fingertips. In: *Pattern Recognition.* 2001, 34(2):255–270.

[18] BELEZNAI, C., RAMOSER, H., WACHMANN, B., BIRCHBAUER, J., BISCHOF, H., and KROPATSCH, W. Memoryefficient Fingerprint Verification. In: *Proceedings of International Conference on Image Processing*. Thessaloniki, Greece 2001, p. 463–466.

[19] KOVACS-VAJNA, Z. M. A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2000, 20(11) :1266–1276.

[20] ROSS, A., JAIN, A. K., and REISMAN, J. A Hybrid Fingerprint Matcher. In: *Pattern Recognition*, 2003, 36(7):1661–1673.

[21] BAZEN, M. A, et al. *A Reinforcement Learning Agent for Minutiae Extraction from Fingerprints.* 2008.

[22] ARIVAZHAGAN, S. et al. Fingerprint Verification using Gabor Cooccurrence Features. In: *International Conference on Computational Intelligence and Multimedia Applications.* 2007, pp 281-285.

ATTACHMENTS

I. CD with the source code, database images and extracted datasets. The application is launched from the gui.m file and contains all tools and instructions necessary.

IMAGE SOURCES

II. http://www.handresearch.com/fingerprints/pics/fingerprint-minutiae-ridge-characteristics.jpg