

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-97831

**ALGORITMUS VÝPOČTU FREKVENČNEJ MAPY  
ODTLAČKOV PRSTOV**  
**BAKALÁRSKA PRÁCA**

**2021**

**Miroslav Kopecký**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-97831

**ALGORITMUS VÝPOČTU FREKVENČNEJ MAPY  
ODTLAČKOV PRSTOV**  
**BAKALÁRSKA PRÁCA**

Študijný program: Aplikovaná informatika

Názov študijného odboru: Informatika

Školiace pracovisko: Ústav informatiky a matematiky

Vedúci záverečnej práce: Ing. Pavol Marák, PhD.

**Bratislava 2021**

**Miroslav Kopecký**



## ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Miroslav Kopecký**

ID študenta: 97831

Študijný program: aplikovaná informatika

Študijný odbor: informatika

Vedúci práce: Ing. Pavol Marák, PhD.

Miesto vypracovania: Ústav informatiky a matematiky

Názov práce: **Algoritmus výpočtu frekvenčnej mapy odtlačkov prstov**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je návrh, implementácia a testovanie zvoleného algoritmu na výpočet frekvenčnej mapy obrazu odtlačku prsta. Výsledná frekvenčná mapa je dôležitým vstupom do algoritmu predspracovania obrazu pomocou Gaborovho filtra, ktorý je súčasťou existujúceho biometrického systému OpenFinger.

Úlohy:

1. Preštudujte relevantnú literatúru a vypracujte prehľad dostupných metód na výpočet frekvenčnej mapy odtlačku prsta.
2. Otestujte fungovanie existujúcej knižnice OpenFinger.
3. Navrhnite a implementujte softvér realizujúci výpočet frekvenčnej mapy algoritmom X-signature.
4. Otestujte implementáciu pomocou vlastnej GUI aplikácie.
5. Integrujte implementovaný algoritmus X-signature do knižnice OpenFinger.
6. Vytvorte písomnú dokumentáciu.

Zoznam odbornej literatúry:

1. Jain, A. – Maio, D. – Maltoni, D. *Handbook of Fingerprint Recognition, Second Edition*. London, UK: Springer, 2009. 496 s. ISBN 978-1-84882-253-5.
2. LEHTIHET, R. et al. Ridge Frequency Estimation For Low-quality Fingerprint Images Enhancement Using Delaunay Triangulation. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 28, No. 01, 1456002 (2014), 17 February 2014
3. ZHANG, L. et al. Fingerprint ridge distance estimation based on ridge search. International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing, 10.1109/IJCBS.2009.20, 2009.

Riešenie zadania práce od: 15. 02. 2021

Dátum odovzdania práce: 04. 06. 2021

**Miroslav Kopecký**

študent

**Dr. rer. nat. Martin Drozda**

vedúci pracoviska

**Dr. rer. nat. Martin Drozda**

garant študijného programu

# SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:

Aplikovaná informatika

Autor:

Miroslav Kopecký

Bakalárská práca:

Algoritmus výpočtu frekvenčnej mapy odtlačkov prstov

Vedúci záverečnej práce:

Ing. Pavol Marák, PhD.

Miesto a rok predloženia práce:

Bratislava 2021

Hlavným cieľom tejto bakalárskej práce je preskúmanie vybraných existujúcich techník výpočtu frekvenčnej mapy obrazu odtlačku prsta, hlbšie preskúmanie metódy X-signature, naprogramovanie jej algoritmu a následné rozšírenie existujúceho biometrického systému OpenFinger, napísaného v jazyku C/C++, o algoritmus frekvenčnej mapy, ktorý momentálne chýba. Po úspešnom implementovaní danej metódy bolo ďalším cieľom optimalizovať vyvinutý algoritmus. Frekvenčná mapa zachytáva dôležitú vlastnosť odtlačkov prstov, ktorou je hustota výskytu papilárnych línii na jednotkovej ploche v odtlačku prsta. Výsledná frekvenčná mapa je dôležitým vstupom do algoritmu predspracovania obrazu pomocou Gaborovho filtra, ktorý je súčasťou systému OpenFinger. Pri predspracovaní odtlačku prsta dochádza k vylepšeniu obrazu odtlačku prsta práve pomocou adaptívneho Gaborovho filtra, ktorého hlavnými vstupmi sú smerová a frekvenčná mapa. Výpočet frekvenčnej mapy je možné konfigurovať, testovať a vizualizovať vo vytvorenjej GUI aplikácií. Implementácia algoritmu metódy X-signature je vyvinutá vo forme dynamickej C++ knižnice, čo umožňuje jednoduchú integráciu do biometrických systémov tretích strán.

Kľúčové slová: frekvenčná mapa, odtlačok prsta, algoritmus, OpenFinger, biometria, Gaborov filter

# ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Miroslav Kopecký
Bachelor's thesis:	Fingerprint frequency map computation algorithm
Supervisor:	Ing. Pavol Marák, PhD.
Place and year of submission:	Bratislava 2021

The main goal of this bachelor thesis is to examine selected existing techniques for calculating the frequency map of a fingerprint image, a deeper examination of the X-signature method, programming its algorithm and subsequent extension of the existing biometric system OpenFinger, written in C/C++, by a frequency map algorithm that is currently missing. . After the successful implementation of the method, another goal was to optimize the developed algorithm. The frequency map captures an important feature of fingerprints, which is the density of ridge lines on a unit area in a fingerprint. The resulting frequency map is an important input to the image preprocessing algorithm using the Gabor filter, which is part of the OpenFinger system. When preprocessing a fingerprint, the fingerprint image is enhanced using an adaptive Gabor filter, the main inputs of which are a directional and frequency map. The frequency map calculation can be configured, tested and visualized in the developed GUI application. The implementation of the X-signature method algorithm is developed in the form of a dynamic C++ library, which allows easy integration into third-party biometric systems.

Keywords: frequency map, fingerprint, algorithm, OpenFinger, biometrics, Gabor filter

## **Podakovanie**

Chcel by som podakovať vedúcemu mojej bakalárskej práce, ktorým bol Ing. Pavol Marák, PhD. za odborné vedenie práce, konzultácií, trpežlivosť a takisto za jeho prispomienky, poznatky, rady a návrhy, ktoré mi pomohli pri vypracovaní tejto bakalárskej práce.

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Teoretické východiská a aktuálny stav skúmanej problematiky</b>	<b>2</b>
1.1 Základné pojmy . . . . .	2
1.1.1 Odtlačok prsta . . . . .	2
1.1.2 Papilárna línia . . . . .	2
1.1.3 Biometria . . . . .	2
1.2 Výpočet smerovej mapy . . . . .	3
1.3 Metódy výpočtu frekvenčnej mapy . . . . .	5
1.3.1 Metóda X-signature . . . . .	6
1.3.2 Metóda založená na sledovaní priebehu papilárnej línie . . . . .	7
1.3.3 Metóda založená na lokalizovaní bodov s lokálnym minimom jasu . . . . .	8
<b>2 Použité vývojové prostriedky</b>	<b>9</b>
2.1 Softvér . . . . .	9
2.1.1 C++ . . . . .	9
2.1.2 Qt . . . . .	9
2.1.3 OpenCV . . . . .	10
2.1.4 OpenFinger . . . . .	10
2.1.5 Persistence1D . . . . .	11
2.1.6 QCustomPlot . . . . .	11
2.2 Hardvér . . . . .	12
2.3 Inštalačné požiadavky . . . . .	12
<b>3 Návrh riešenia</b>	<b>14</b>
3.1 Ciele práce a motivácia . . . . .	14
3.2 Implementácia metódy X-signature . . . . .	14
3.2.1 Vstup obrazu . . . . .	14
3.2.2 Získanie lokálnej orientácie zo smerovej mapy . . . . .	15
3.2.3 Natočenie orientovaného okna . . . . .	16
3.2.4 Výpočet nových súradníc po natočení obrazu . . . . .	17
3.2.5 Zápis do vektora X-signature . . . . .	19
3.2.6 Výpočet frekvencie . . . . .	19
3.2.7 Zápis do matice . . . . .	20
3.3 Optimalizácia . . . . .	21

3.3.1	Výpočet frekvenčnej mapy po blokoch . . . . .	21
3.3.2	Výpočet frekvenčnej mapy vo vláknach . . . . .	23
<b>4</b>	<b>Testovanie a výsledky vyvinutého algoritmu</b>	<b>24</b>
4.1	Výsledky pred optimalizáciou . . . . .	24
4.2	Výsledky po optimalizácii . . . . .	27
4.2.1	Výsledky po optimalizácii výpočtu frekvenčnej mapy po blokoch .	27
4.2.2	Výsledky po optimalizácii výpočtu frekvenčnej mapy pomocou vlákien	30
4.2.3	Pridanie rozmazania Gaussian blur . . . . .	31
4.3	GUI aplikácia . . . . .	32
4.3.1	Nastavovanie parametrov pri výpočte frekvenčnej mapy . . . . .	32
4.3.2	Používanie GUI aplikácie . . . . .	32
<b>Záver</b>		<b>34</b>
<b>Zoznam použitej literatúry</b>		<b>35</b>
<b>Prílohy</b>		<b>I</b>
<b>A Štruktúra elektronického nosiča</b>		<b>II</b>

# Zoznam obrázkov a tabuliek

Obrázok 1	Detail papilárnej línie . . . . .	3
Obrázok 2	Ukážka frekvenčnej mapy odtlačku prsta . . . . .	5
Obrázok 3	Orientované okno . . . . .	6
Obrázok 4	Vizualizácia vektora X-signature . . . . .	7
Obrázok 5	Vypĺňanie smerového okna . . . . .	7
Obrázok 6	Obrázok výpočtu vzdialenosť medzi 2 papilárnymi líniami . . .	8
Obrázok 7	Architektúra biometrického systému OpenFinger . . . . .	10
Obrázok 8	Graf zobrazujúci nájdené lokálne extémy pri perzistencii 0 . . .	11
Obrázok 9	Schéma knižnice na predspracovanie odtlačkov prstov systéme OpenFinger . . . . .	14
Obrázok 10	Príklad obrazového vstupu . . . . .	15
Obrázok 11	Smerová mapa obrázku 10 . . . . .	15
Obrázok 12	Pôvodné natočenie orientovaného okna podľa metódy X-signature	16
Obrázok 13	Nami navrhnutý systém natočenia celého vstupného obrazu . . .	17
Obrázok 14	Ukážka súradnicovej sústavy v matici knižnice OpenCV . . . .	18
Obrázok 15	Natočenie obrazu podľa bodu $(x, y)$ okolo stredového bodu $(p, q)$	18
Obrázok 16	Príklad výpočtu a zápisu priemernej jasovej hodnoty stĺpcov z orientovaného okna do vektora X-signature . . . . .	19
Obrázok 17	Profil lokálneho jasu orientovaného okna s centrom v bode $(x, y)$ , s vyznačenými lokálnymi maximami a perzistenciaou 5 . . . . .	20
Obrázok 18	Frekvenčná mapa obrázku 10 . . . . .	20
Obrázok 19	Detail frekvenčnej mapy s hodnotami frekvencií . . . . .	21
Obrázok 20	Rozdelenie vstupného obrazu na bloky o veľkosti $13 \times 13$ s vy- značenými centrálnymi bodmi . . . . .	22
Obrázok 21	Detail frekvenčnej mapy s hodnotami frekvencií v blokoch . . .	22
Obrázok 22	Rozdelenie vstupného obrazu na 8 vlákien podľa počtu riadkov .	23
Obrázok 23	Obrázok s kalibračným vzorom o veľkosti $420 \times 777$ . . . . .	24
Obrázok 24	Obrázok so vzorom o veľkosti $500 \times 506$ . . . . .	25
Obrázok 25	Obrázok so vzorom o veľkosti $500 \times 503$ . . . . .	25
Obrázok 26	Príklad kvalitne zosnímaného odtlačku prsta o veľkosti 300x300.	26
Obrázok 27	Príklad nekvalitne zosnímaného odtlačku prsta o veľkosti 300x300.	26
Obrázok 28	Vstupný obraz použitý na testovanie rýchlosť algoritmu . . . . .	26

Obrázok 29	Graf zobrazujúci rýchlosť výpočtu frekvenčnej mapy vzhľadom na rôznu veľkosť vstupného obrazu pri základnej verzii algoritmu	27
Obrázok 30	Obrázok so vzorom o veľkosti $500 \times 503$ .	28
Obrázok 31	Príklad kvalitne zosnímaného odtlačku prsta o veľkosti 300x300.	28
Obrázok 32	Príklad nekvalitne zosnímaného odtlačku prsta o veľkosti 300x300.	29
Obrázok 33	Porovnanie rôznych veľkostí blokov na odtlačku prsta o veľkosti 300x300.	29
Obrázok 34	Graf zobrazujúci rýchlosť výpočtu frekvenčnej mapy vzhľadom na rôznu veľkosť vstupného obrazu pri optimalizovanej verzii algoritmu pomocou výpočtu po blokoch	30
Obrázok 35	Graf zobrazujúci rýchlosť výpočtu frekvenčnej mapy vzhľadom na rôznu veľkosť vstupného obrazu pri optimalizovanej verzii algoritmu pomocou výpočtu po blokoch a rozdelenia výpočtu na vlákna	31
Obrázok 36	Použitie rozmazania Gaussian blur vo vypočítanej frekvenčnej mape s blokom rozmazania o veľkosti 13 a sigmou o veľkosti 6	31
Obrázok 37	GUI aplikácia	33

# Zoznam skratiek

<b>CPU</b>	Central processing unit - procesor
<b>GPL</b>	General Public License - licencia pre slobodný software
<b>GPU</b>	Graphics processing unit - grafický procesor
<b>GUI</b>	Graphical user interface - grafické používateľské rozhranie
<b>HW</b>	hardware - technické vybavenie počítača
<b>KDE</b>	K Desktop Environment - desktopové prostredie pre unix operačné systémy
<b>RAM</b>	Random Access Memory - operačná pamäť
<b>SIFT</b>	Scale-Invariant feature transform - automatické nájdenie korešpondencií medzi dvojicou obrázkov
<b>SURF</b>	Speeded-Up Robust Features - metóda popisujúca obrázok pomocou deskriptorov
<b>SW</b>	software - programové vybavenie počítača

# Úvod

V dnešnej dobe sa už bežne využíva odtlačok prsta ako spôsob overenia identity, či už sa jedná o odblokovanie mobilného telefónu, alebo overenie pri vstupe do bankovej aplikácie. Väčšina systémov rozpoznávania odtlačkov prstov je však stále nedokonalá. Často sa stáva, že senzor odtlačku prsta na smartfóne nevie rozpoznať náš odtlačok prsta. Toto sa deje najmenej v prípadoch, keď na pokožke máme nejaké nečistoty z prachu alebo je pokožka vlhká. Pri takomto nasnímaní odtlačku prsta dostávame zašumený digitalizovaný obraz odtlačku prsta. Zašumený obraz odtlačku prsta je jedným z možných skreslení, ktoré ovplyvňujú systém rozpoznávania odtlačku prsta. Ďalšími sú napríklad deformácia pokožky (napr. z dôvodu nadmerného tlaku prsta na senzor) alebo neočakávané natočenie prsta na senzore, pri snímaní odtlačku prsta.

Na to, aby sa aj napriek takýmto nedokonalostiam dokázal vytvoriť kvalitný snímok odtlačku prsta slúži Gaborov filter, ktorý má za účel prefiltrovať a vylepšiť kvalitu nasnímaného obrazu odtlačku prsta. Jedným z dvoch hlavných vstupov do Gaborovho filtra je frekvenčná mapa. V biometrickom systéme OpenFinger, ktorý slúži na rozpoznávanie identity pomocou odtlačkov prstov, momentálne chýba algoritmus výpočtu frekvenčnej mapy. Pravé z tohto dôvodu sme si ako tému tejto bakalárskej práce zvolili preskúmať možné metódy výpočtu frekvenčnej mapy, vytvorenie algoritmu jednej z nich, konkrétnie metódy X-signature, a implementovanie tejto metódy do biometrického systému OpenFinger.

Táto práca pozostáva zo štyroch kapitol, pričom v 1. kapitole sa venujeme základným teoretickým poznatkom z oblasti odtlačkov prstov. V kapitole 2 je uvedený všetok použitý SW, HW a požiadavky na inštaláciu. V 3. kapitole je opísaná samotná implementácia metódy X-signature a následne v kapitole 4 sú uvedené výsledky testovania.

# 1 Teoretické východiská a aktuálny stav skúmanej problematiky

V tejto kapitole sú vysvetlené základné pojmy, ktoré sú podstatné pri porozumení problematiky danej témy, opísaná metóda výpočtu smerovej mapy, ktorá je potrebným vstupom do algoritmu frekvenčnej mapy, a priblížené metódy výpočtu frekvenčnej mapy.

## 1.1 Základné pojmy

### 1.1.1 Odtlačok prsta

EĽudia používajú odtlačky prstov na osobnú identifikáciu už mnoho storočí a ukázalo sa, že presnosť zhody pomocou odtlačkov prstov je veľmi vysoká [1]. Odtlačok prsta je vzor papilárnych línii na povrchu špičky prsta. Charakteristické zakončenia papilárnych línii sa nazývajú markanty. Špecifický reliéf papilárnych línii sa u človeka formuje už v prvých 7 mesiacoch embryonálneho vývoja. Odtlačok prsta je rozdielny aj pre jednovaječné dvojčatá a dokonca je rozdielny pre každý prst jedného človeka [2].

### 1.1.2 Papilárna línia

Koža sama o sebe nie je len jednou homogénnou vrstvou, ale je zložená z viac funkčne rozdelených vrstiev. Tieto vrstvy môžeme obecne rozdeliť na epidermálnu (vrchná vrstva kože), dermálnu a podkožnú vrstvu. Koža na spodnej strane prsta obsahuje dermatoglyfické vzory (odtlačky), ktoré tvoria výbežky a priehlbiny. Tieto kožné výbežky sú usporiadane do vzorov a tvoria papilárne línie. Individualita odtlačkov sa prejavuje hlavne v nespojitosti papilárnych línii, t.j. v ich lokálnych tvarových anomáliách, ktoré sa vyskytujú najmä v dvoch tvaroch: rozdvojenie a ukončenie papilárnej línie [3][4][5]. Na obrázku 1 môžeme vidieť detail papilárnej línie.

### 1.1.3 Biometria

Pojem biometria pochádza z gréckeho slova "bios" znamenajúce život a "metros" znamenajúce "mierka". Biometria predstavuje identifikáciu a verifikáciu ľudskej bytosti na základe jej fyziologickej a behaviorálnej charakteristiky [6].

V súčasnej dobe, kedy je väčšina transakcií (najmä finančných) automatizovaná a množstvo z nich prepojené sa dôležitosť zabezpečenia zvyšuje. Zabezpečenie je často overované formou vlastníctva (identifikačné karty ako občiansky alebo vodičský preukaz, identifi-



Obr. 1: Detail papilárnej línie

kačné klúče) alebo tajnou vedomosťou (heslo, PIN kód). Tento typ zabezpečenia však nie je neomylný. Identifikačná karta môže byť stratená alebo odcudzená a heslo zabudnuté. Teda v tomto smere zabezpečenia došlo k ďalšiemu vývoju a ten viedol k využívaniu časti ľudského tela alebo ľudského správania pre zabezpečenie a autentifikáciu a tiež viedol k vzniku biometrie ako vedného odboru. Identifikácia osoby by teda teraz mala pozostávať z biometrickej identifikácie [6].

## 1.2 Výpočet smerovej mapy

Lokálna orientácia papilárnej línie je v bode jedného pixelu definovaná ako uhol, ktorý zviera dotyčnica k papilárnej linii s horizontálnou osou. Je obecne známe, že papilárne línie nemajú smer, ale majú orientáciu. Pojem orientácia v oblasti spracovania odtlačkov prstov teda predstavuje uhol ležiaci v rozmedzí  $0\text{-}180^\circ$  a pojem smer udáva uhol ležiaci v rozmedzí  $0\text{-}360^\circ$ .

"Smerová mapa je nevyhnutná pre správne nastavenie Gaborovho filtra. Je to matica, ktorá obsahuje hodnoty uhlov v radiánoch v rozsahu  $\langle 0, \pi \rangle$ " [5].

Jej postup výpočtu, opísaný v práci [5], je takýto:

1. Obrázok odtlačku rozdelíme na štvorcové bloky o veľkosti  $W \times W$ . Smer sa bude počítať pre stredný bod každého bloku. Stred  $W \times W$  bloku nastavíme na bod  $(i, j)$ .
2. Pre každý pixel v bloku  $W \times W$  sa vypočítajú gradienty  $\delta_x$  a  $\delta_y$ .

3. Pomocou týchto vzťahov sa vypočíta odhad lokálnej orientácie v danom bode  $(i, j)$ :

$$V_x(i, j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2\delta_x(u, v)\delta_y(u, v) \quad (1)$$

$$V_y(i, j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} \delta_x^2(u, v)\delta_y^2(u, v) \quad (2)$$

$$\theta(i, j) = \frac{1}{2} \tan^{-1} \frac{V_y(i, j)}{V_x(i, j)} \quad (3)$$

4. Smerová mapa sa rozdelí na sínusové a kosínusové komponenty, ktoré sa vyhľadia Gaussovým filtrom:

$$\phi_x(i, j) = \cos(2\theta(i, j)) \quad (4)$$

$$\phi_y(i, j) = \sin(2\theta(i, j)) \quad (5)$$

$$\phi'_x(i, j) = \sum_{u=-\frac{\omega_\phi}{2}}^{\frac{\omega_\phi}{2}} \sum_{v=-\frac{\omega_\phi}{2}}^{\frac{\omega_\phi}{2}} G(u, v)\phi_x(i - uw, j - vw) \quad (6)$$

$$\phi'_y(i, j) = \sum_{u=-\frac{\omega_\phi}{2}}^{\frac{\omega_\phi}{2}} \sum_{v=-\frac{\omega_\phi}{2}}^{\frac{\omega_\phi}{2}} G(u, v)\phi_y(i - uw, j - vw) \quad (7)$$

5. Výsledná smerová mapa pre bod  $(i, j)$  sa vypočíta nasledovne:

$$O(i, j) = \frac{1}{2} \tan^{-1} \frac{\phi'_y(i, j)}{\phi'_x(i, j)} \quad (8)$$

### 1.3 Metódy výpočtu frekvenčnej mapy

Frekvenčná mapa je matica kódujúca hustotu papilárnych línií. Pomocou svetlých a tma-vých pruhov vieme odlišiť frekvencie. Výpočet frekvenčnej mapy je blokovo-orientovaný a teda je potrebné, aby sme blok, v ktorom počítame frekvenciu, natočili rovnobežne s papilárnymi líniami v okolí. V bloku počítame priemernú vzdialenosť medzi dvomi po sebe idúcimi lokálnymi maximami. Prevrátená hodnota priemernej vzdialenosť nám určuje hodnotu frekvencie. Frekvenčná mapa vstupuje do Gaborovho filtra spoločne so smerovou mapou. Momentálne sú nám známe a zdokumentované 3 rôzne metódy na výpočet frekvenčnej mapy [5]. Na obrázku 2 môžeme vidieť, ako by mala vyzerat výsledná frekvenčná mapa.



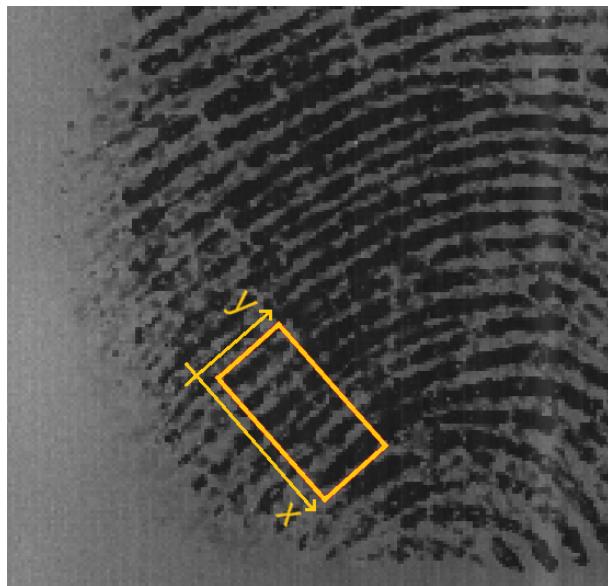
Obr. 2: Ukážka frekvenčnej mapy odtlačku prsta [5]

Vľavo originálny odtlačok, v strede smerová mapa, vpravo frekvenčná mapa

### 1.3.1 Metóda X-signature

Metóda X-signature je najznámejšou metódou na výpočet frekvenčnej mapy. Jej postup výpočtu je takýto:

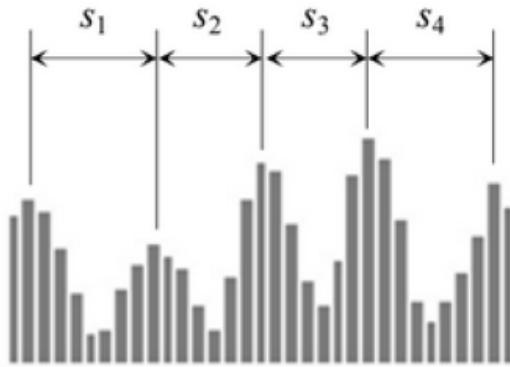
1. V konkrétnom bode  $(x, y)$  na obrázku odtlačku prsta sa vytvorí virtuálne orientované okno s rozmermi 32x16 a stredom v bode  $(x, y)$ . Zo smerovej mapy sa získa uhol pre tento konkrétny bod  $(x, y)$  a orientované okno sa podľa tohto uhla natočí, tak aby jeho y-ová os, bola súbežná s papilárnymi líniami v danom bloku (viď obrázok 3).



Obr. 3: Orientované okno

2. V rámci tohto orientovaného okna robíme priemer všetkých hodnôt každého stĺpca (pixelov) a zapisujeme ich do vektora X-signature. Keďže z jednotlivých pixelov získavame hodnotu jasu v danom bode, vektor X-signature predstavuje profil lokálneho jasu odtlačku prsta v smere kolmom na papilárne línie v danom bloku (obr. 4). Týmto spôsobom sa zvyšuje odolnosť metódy voči šumu [5].
3. Frekvencia v danom bode  $(x, y)$  sa vypočíta ako prevrátená hodnota priemernej vzdialosti medzi dvomi po sebe idúcimi vrcholmi papilárnych línií [5].

$$f_{xy} = \frac{4}{s_1 + s_2 + s_3 + s_4} \quad (9)$$

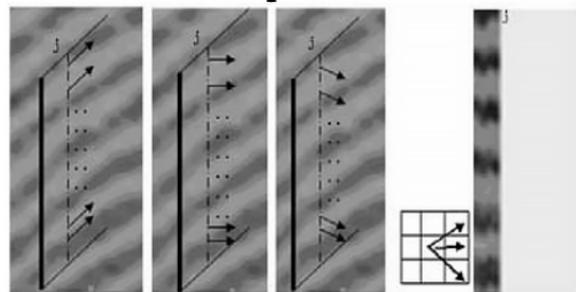


Obr. 4: Vizualizácia vektora X-signature [5]

### 1.3.2 Metóda založená na sledovaní priebehu papilárnej línie

Algoritmus, navrhnutý autorom Zhang, L. v práci [7], je nasledovný:

1. Obraz odtlačku prsta sa rozdelí na bloky a pre každý blok sa určí štartovacia úsečka. Táto úsečka sa určuje tak aby prechádzala cez čo najviac papilárnych línii. V každom bloku je dostupná jedna horizontálna a jedna vertikálna úsečka. V ďalších výpočtoch pracujeme vždy s tou, ktorá prechádza cez viac papilárnych línii.
2. Zo štartovacej úsečky sledujeme všetky papilárne línie, ktoré z nej vychádzajú. Následne sa môžeme pohnúť buď rovno alebo do jedného z diagonálnych smerov. To, ktorým smerom sa pohneme nám určuje výsledok globálnej optimalizačnej funkcie. Takýmto spôsobom postupne vyplníme tzv. smerové okno a následne aj celý blok (viď obr. 5).



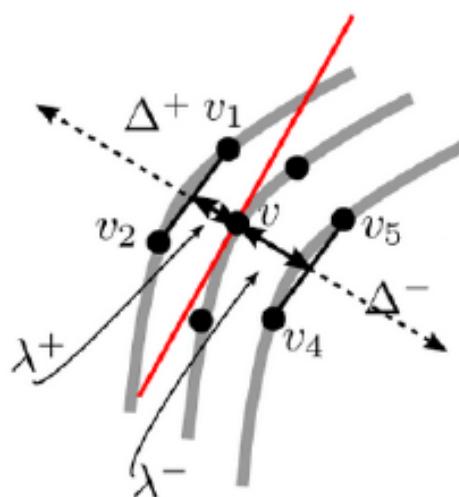
Obr. 5: Vyplňanie smerového okna [6]

3. Autor poukazuje na to, že vzdialenosť papilárnych línii od seba v smerovom okne sú skreslené. Preto navrhol prepočet vypočítanej vzdialenosťi z algoritmu na reálnu vzdialenosť.

### 1.3.3 Metóda založená na lokalizovaní bodov s lokálnym minimom jasu

Postup tejto metódy výpočtu frekvenčnej mapy, ktorá bola prezentovaná v práci [8], je takýto:

1. Obraz odtlačku prsta sa rozdelí na malé štvorcové bloky. V rámci týchto štvorcových blokov sa vyberie jeden bod s lokálnym minimom jasu. Takýto bod bude s veľmi velkou pravdepodobnosťou ležať niekde na papilárnej líni.
2. Zo všetkých takto určených bodov s lokálnym minimom jasu sa vybuduje Delaunayova triangulácia.
3. Následne zmeriame vzdialenosť medzi nejakým konkrétnym bodom  $b$  a všetkými susednými bodmi, ktoré sa nachádzajú na vedľajších papilárnych líniach. V tejto časti algoritmu využijeme smerovú mapu na to, aby sme zistili v akom smere sa nachádzajú vedľajšie papilárne línie okolo konkrétneho bodu  $b$ . Vytvorí sa kolmá virtuálna čiara na papilárnu líniu, na ktorej sa nachádza bod  $b$ . Táto čiara pretne susedné hrany triangulácie na dvoch miestach.
4. Následne sa vypočíta vzdialenosť medzi susednými papilárnymi líniami tak, že sa zmeria vzdialenosť medzi daným bodom  $b$  a priesecníkmi z predchádzajúceho bodu. Prevrátenou hodnotou tejto vzdialenosťi je naša výsledná frekvencia. Túto metódu môžme sledovať na obrázku 6.



Obr. 6: Obrázok výpočtu vzdialenosť medzi 2 papilárnymi líniami [6]

## 2 Použité vývojové prostriedky

V tejto časti je uvedený SW a HW použitý pri vypracovaní tejto práce.

### 2.1 Softvér

#### 2.1.1 C++

Celá knižnica OpenFinger je naprogramovaná v jazykoch C a C++. Kedže sme vyvíjali algoritmus výpočtu frekvenčnej mapy, ktorý sa mal stať súčasťou tejto knižnice, a bolo potrebné priamo pracovať s HW počítača, tak som sa rozhodol pre jazyk C++.

C++ je univerzálny programovací jazyk navrhnutý tak, aby spríjemnil programovanie serióznomu programátorovi. Až na malé drobnosti je C++ nadmnožinou programovacieho jazyka C. Okrem služieb, ktoré poskytuje jazyk C, C++ poskytuje flexibilnú a efektívnu funkcionality pre definovanie nových typov. Programátor môže rozdeliť aplikáciu na spravovateľné časti, definovaním nových typov, ktoré sa úzko zhodujú s konceptom aplikácie. Táto technika konštruovania programu sa často nazýva abstrakcia údajov. Ak sa tieto techniky použijú dobre, výsledkom sú kratšie, ľahšie pochopiteľné a ľahšie udržiavateľné programy. Klúčovým konceptom v C ++ je trieda (angl. class). Trieda je používateľom definovaný typ. Triedy poskytujú skrývanie dát, zaručenú inicializáciu údajov, implicitnú konverziu typov pre typy definované používateľom, používateľom riadenú správu pamäte a mechanizmy pre preťažovanie operátorov [9].

#### 2.1.2 Qt

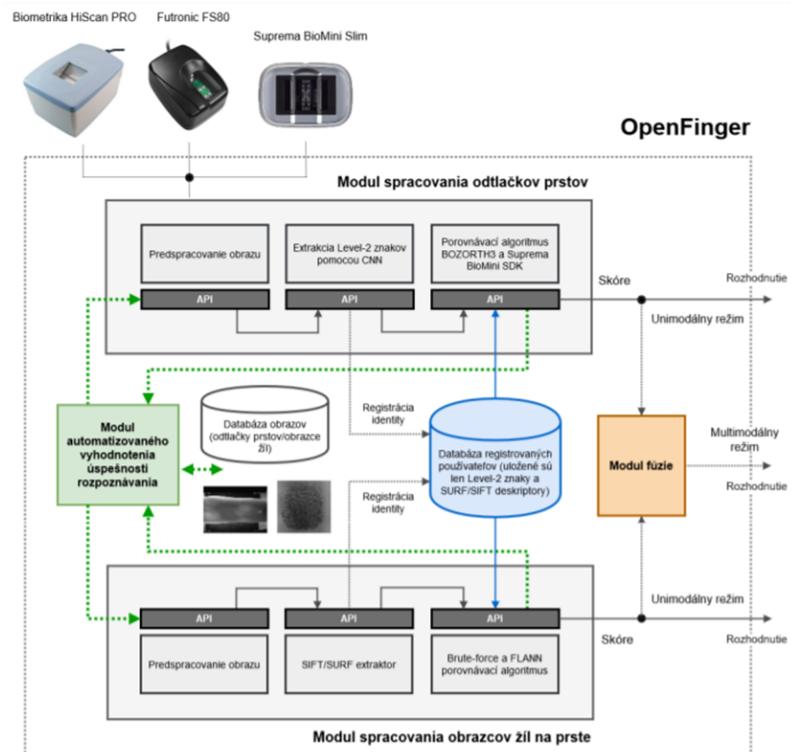
Qt je multiplatformový open-source aplikačný C++ framework. Tento framework začala vyvíjať nórská firma Trolltech, ktorú neskôr odkúpila Nokia. Medzi najznámejšie projekty, ktoré (buď časťou alebo celé) boli vyvíjané pod Qt frameworkom sú - Mathematica (GUI), Skype (Linux, GUI), Google Earth (Linux), KDE (grafická nadstavba operačného systému). Aplikácie Qt fungujú na rôznych softvérových a hardvérových platformách, ako sú Linux, Windows, macOS, Android alebo vstavaných systémoch. Qt v súčasnosti vyvíja The Qt Company a Qt Project pod open-source správou. Je k dispozícii v rámci komerčných licencií aj open-source licencií GPL 2.0, GPL 3.0 a LGPL 3.0.

### 2.1.3 OpenCV

OpenCV (Open Source Computer Vision Library) je multiplatformová open-source knižnica, ktorá obsahuje algoritmy zamerané na počítačové videnie. Knižnica obsahuje viac ako 2500 optimalizovaných algoritmov. Obsahuje klasické ale aj najmodernejšie riešenia problémov z oblasti počítačového videnia a strojového učenia. Tieto algoritmy môžu byť použité na rozpoznanie tvári, klasifikáciu ľudského správania a sledovanie pohybu vo videách, extrakciu príznakov, spájanie obrázkov na tvorbu vysoko kvalitných obrazov celej scény, hľadanie podobných obrázkov v databáze atď. Knižnica je široko používaná vo výskumných skupinách, firmách a štátnych podnikoch. OpenCV je napísaná v jazyku C++, ale je možné použiť ju aj v jazykoch Python, Java a MATLAB.

### 2.1.4 OpenFinger

Biometrický systém OpenFinger, vyvíjaný na pracovisku Ústavu informatiky a matematiky FEI STU, je tvorený ekosystémom viacerých dynamických softvérových knižníc a sprievodných modulov a aplikácií. Slúži na rozpoznávanie identity pomocou odtlačkov prstov a obrazcov ciev na prste. Kompletná architektúra systému OpenFinger je znázorená na obrázku 7 [10].

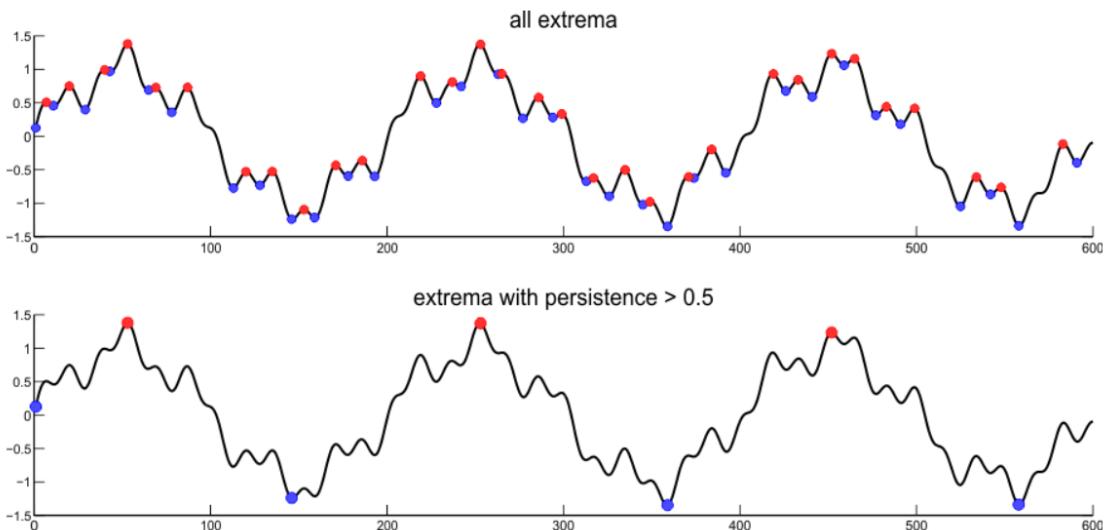


Obr. 7: Architektúra biometrického systému OpenFinger [10]

„Všetky moduly systému sú naprogramované v jazyku C/C++ s využitím voľne dostupných knižníc Qt (tvorba dynamických knižníc, tvorba GUI, súborový systém, vlákna, sieťová komunikácia, signály), OpenCV (manipulácia s obrazom, filtrovanie, SIFT/SURF extraktory), Caffe (konvolučné neurónové siete), ArrayFire (GPU implementácie algoritmov) a QCustomPlot (vizualizácia udájov)“ [10].

### 2.1.5 Persistence1D

Persistence1D je knižnica na hľadanie lokálnych extrémov a ich perzistencia v jednorozmerných dátach. Lokálne minimá a lokálne maximá sa extrahujú, párujú a triedia podľa ich perzistencia. Balík podporuje C++, Python a Matlab. Verziu Persistence1D pre C++ a Matlab naprogramovali Yeara Kozlov a Tino Weinkauf, Informatický inštitút Maxa Plancka, Saarbrücken, Nemecko. Knižnica Persistence1D je voľne dostupná na adrese <https://github.com/weinkauf/Persistence1D>.



Obr. 8: Graf zobrazujúci nájdené lokálne extémy pri perzistencii 0 (obr. vyššie) a pri perzistencii  $> 0.5$  (obr. nižšie)

### 2.1.6 QCustomPlot

QCustomPlot je Qt C++ knižnica, ktorá slúži na vykreslovanie rôznych 2D grafov alebo taktiež ponúka interaktívnu vizualizáciu údajov. Grafy knižnice QCustomPlot môžu byť exportované do rôznych formátov, ako napr. súbory PDF a obrázky PNG, JPG a BMP.

## 2.2 Hardvér

Biometrický systém OpenFinger, ktorý bol využívaný počas vývoja algoritmu výpočtu frekvenčnej mapy a do ktorého sme nakonci implementovali nami vyvinutý algoritmus, je náročnejší na výkon počítača. Preto bol pre potreby tejto práce použitý nasledovný výpočtový výkon:

GPU:

- názov: Nvidia GeForce GTX 1650M
- architektúra Turing (čip TU117), vyrobený 12nm technológiou
- pamäť 4GB GDDR6
- 128 bitová zbernice s frekvenciou 8000MHz
- 1024 jadier s frekvenciou 1395MHz (pri režime Boost 1560MHz)

CPU:

- názov: Intel Core i5-9300H
- vyrobený 14nm technológiou
- 4 jadrá, 8 vlákien s frekvenciou 2400 MHz (pri režime Turbo 4100 MHz)

RAM:

- 16GB DDR4 s frekvenciou 2667 MHz

Táto práca a algoritmus frekvenčnej mapy boli vypracované na notebooku Acer Nitro 5.

## 2.3 Inštalačné požiadavky

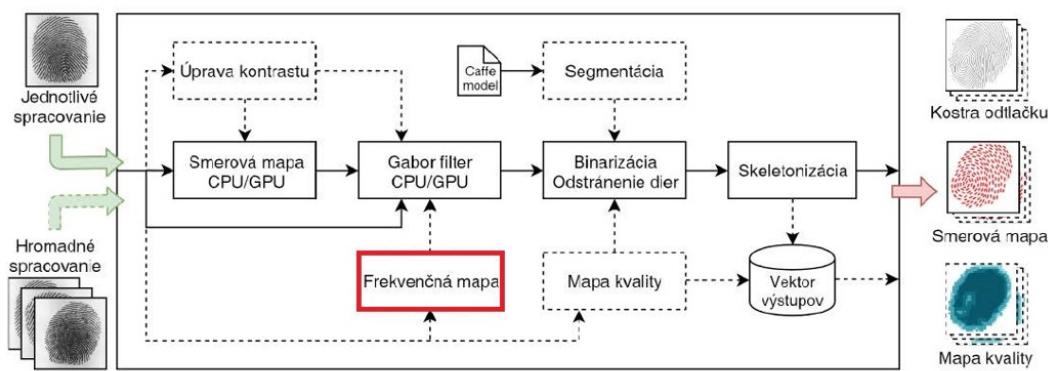
Pre úspešné skompilovanie a spustenie vyvinutého algoritmu výpočtu frekvenčnej mapy je potrebné mať grafickú kartu Nvidia, nainštalovaný operačný systém Linux (odporúčaný Manjaro), vývojové prostredie Qt/Qt Creator, úspešne skompilovanú knižnicu OpenFinger, z ktorej používame smerovú mapu. Úspešne skompilovanú knižnicu Caffe, ktorá je potrebná v knižnici OpenFinger. Na to aby sme mohli knižnicu Caffe skompilovať, potrebujeme mať: nainštalovaný program CMake, ktorý knižnicu skompliluje, a tieto závislosti:

- BLAS knižnica
- Boost knižnica
- glog knižnica
- gflags knižnica
- Google Protobuf knižnica
- lmdb knižnica
- LevelDB knižnica
- Snappy knižnica
- OpenCV knižnica
- CUDA Toolkit

# 3 Návrh riešenia

## 3.1 Ciele práce a motivácia

Po preštudovaní si relevantnej literatúry bolo cieľom práce úspešne naprogramovať algoritmus výpočtu frekvenčnej mapy, konkrétnie metódu X-signature, a rozšíriť existujúci biometrický systém OpenFinger o nami vytvorený algoritmus. Na obrázku 9 nižšie môžeme vidieť schému knižnice na predspracovanie odtlačkov prstov. Hlavným prvkom pre nás je červenou farbou zvýraznená frekvenčná mapa, ktorá vstupuje do Gaborovho filtra spolu so smerovou mapou.



Obr. 9: Schéma knižnice na predspracovanie odtlačkov prstov v systéme OpenFinger [5]

## 3.2 Implementácia metódy X-signature

Navrhnutý algoritmus metódy X-signature je opísaný v nasledujúcich podkapitolách.

### 3.2.1 Vstup obrazu

Náš algoritmus je vyvíjaný najmä na použitie pri rozpoznávaní odtlačkov prstov, avšak ako vstupný obraz môže byť použitý hocijaký súbor obrazového formátu (napr. PNG, JPG, TIFF). Obrázok 10 slúži ako príklad konkrétneho obrazového vstupu. Na základe výšky a šírky vstupného obrazu sa vytvorí prázdna matica (frekvenčná mapa) s rovnakými rozmermi ako je vstupný obraz.



Obr. 10: Príklad obrazového vstupu

### 3.2.2 Získanie lokálnej orientácie zo smerovej mapy

Pomocou algoritmu na výpočet smerovej mapy, ktorá je súčasťou knižnice OpenFinger, získame zo vstupného obrazu jeho smerovú mapu. Smerovú mapu pre obrázok 10 môžeme pozorovať na obrázku 11.



Obr. 11: Smerová mapa obrázku 10

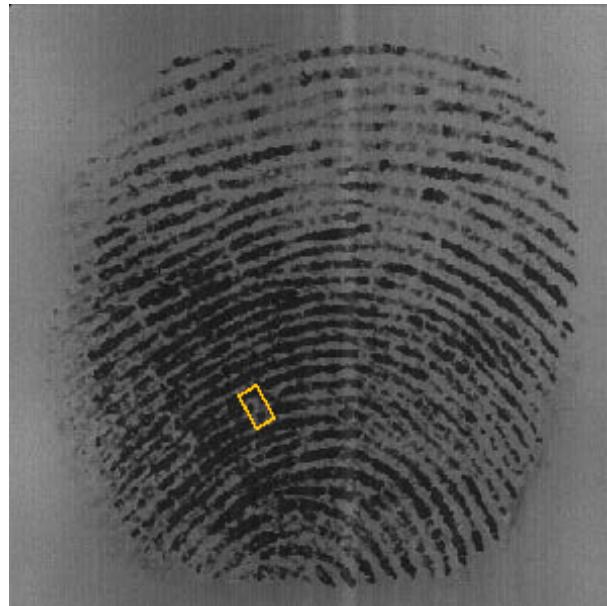
Následne sa začne vykonávať vnorený for-cyklus, ktorý prechádza postupne cez všetky riadky a v každom riadku cez všetky stĺpce smerovej mapy vstupného obrazu. Takýmto

spôsobom algoritmus prejde cez všetky pixely a z každého pixelu sa získa radián natočenia papilárnej línie v danom pixeli. Na základe veľkosti rotácie sa následne natočí orientované okno.

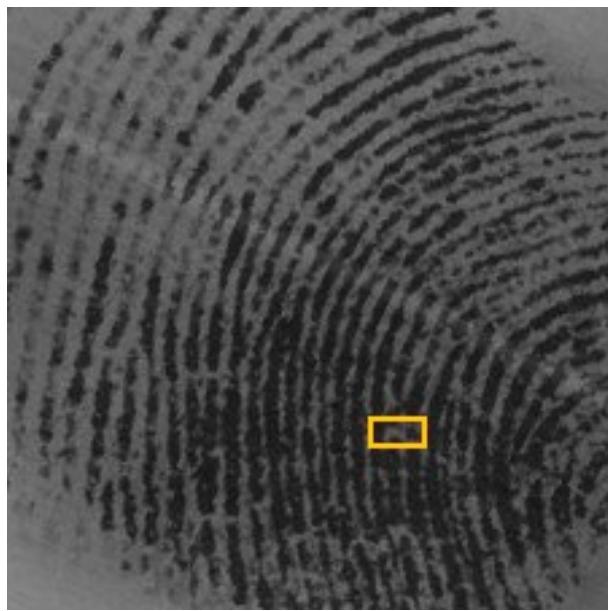
### 3.2.3 Natočenie orientovaného okna

V metóde X-signature je dané orientované okno s rozmermi  $32 \times 16$ , čo však nie je pre náš účel správne. Aby sme mohli určiť centrálny bod orientovaného okna potrebujeme, aby veľkosti jeho strán boli nepárne. Preto sme sa rozhodli použiť orientované okno o veľkosti  $33 \times 17$ .

Po vytvorení orientovaného okna by sa malo toto okno natočiť podľa smeru získaného z aktuálne prechádzaného bodu zo smerovej mapy. Tu sme sa však opäť rozhodli metódou X-signature trochu upraviť. Namiesto toho, aby sme otáčali orientované okno, je pre nás jednoduchšie natočiť celý vstupný obraz. Tým pádom je orientované okno vždy vo vodorovnej polohe a vstupný obraz sa otáča v negatívnej hodnote radiánu získaného z aktuálne prechádzaného bodu zo smerovej mapy. Na obrázku 12 je vidieť príklad pôvodného návrhu natočenia orientovaného okna a na obrázku 13 je príklad nami pozmenenej časti s natočením celého obrazu.



Obr. 12: Pôvodné natočenie orientovaného okna podľa metódy X-signature



Obr. 13: Nami navrhnutý systém natočenia celého vstupného obrazu

### 3.2.4 Výpočet nových súradníc po natočení obrazu

Po natočení vstupného obrazu, potrebujeme zistiť nové  $(x', y')$  súradnice aktuálne prechádzaného pixelu. Nové súradnice bodu  $(x, y)$  sa po rotácii obrazu okolo bodu  $(0, 0)$  počítajú nasledovným vzťahom:

$$x' = x \cos(\theta) - y \sin(\theta) \quad (10)$$

$$y' = x \sin(\theta) + y \cos(\theta) \quad (11)$$

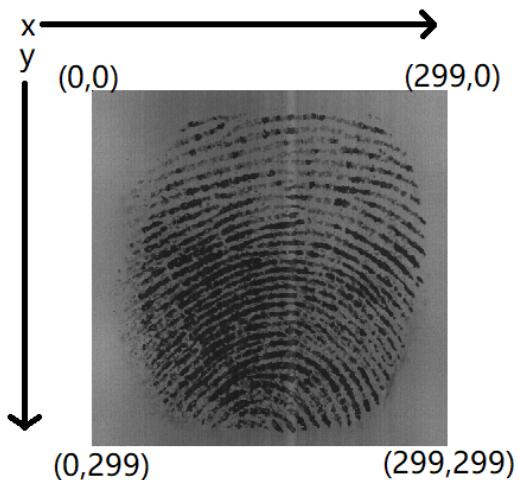
kde bod  $(x, y)$  je pôvodný bod, bod  $(x', y')$  je nový bod a  $\theta$  je uhol natočenia papilárnej línie v danom bode v radiánoch.

Kedže náš vstupný obraz je zapísaný v matici pomocou knižnice OpenCV, bod  $(0, 0)$  sa nenachádza v strede tejto matice ale v ľavom hornom rohu (viď obrázok 14).

Preto je potrebné pre náš algoritmus pozmeniť výpočet nového bodu  $(x', y')$  takto:

$$x' = (x - p) \cos(\theta) - (y - q) \sin(\theta) + p \quad (12)$$

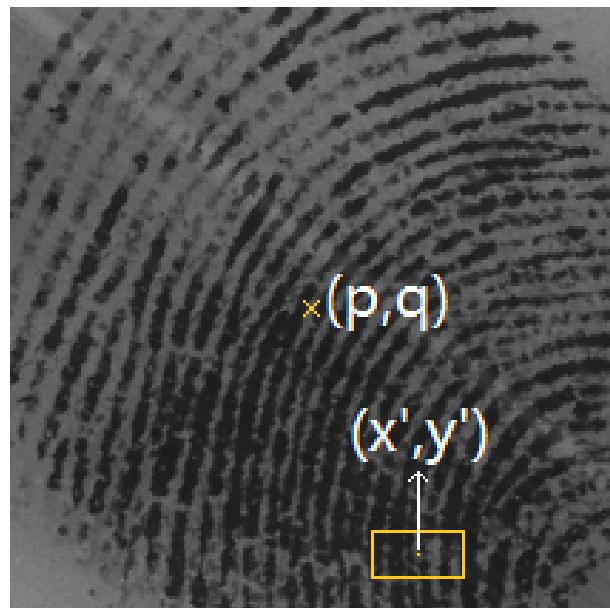
$$y' = ((x - p) \sin(\theta) + (y - q) \cos(\theta) + q) * (-1) \quad (13)$$



Obr. 14: Ukážka súradnicovej sústavy v matici knižnice OpenCV

kde bod  $(x, y)$  je pôvodný bod, bod  $(x', y')$  je nový bod,  $\theta$  je uhol natočenia papilárnej línie v danom bode v radiánoch a bod  $(p, q)$  je stredový bod vstupného obrazu. Kedže nám y-ová os narastá smerom nadol, je potrebné novú hodnotu y-ovej súradnice prevrátiť do kladnej hodnoty.

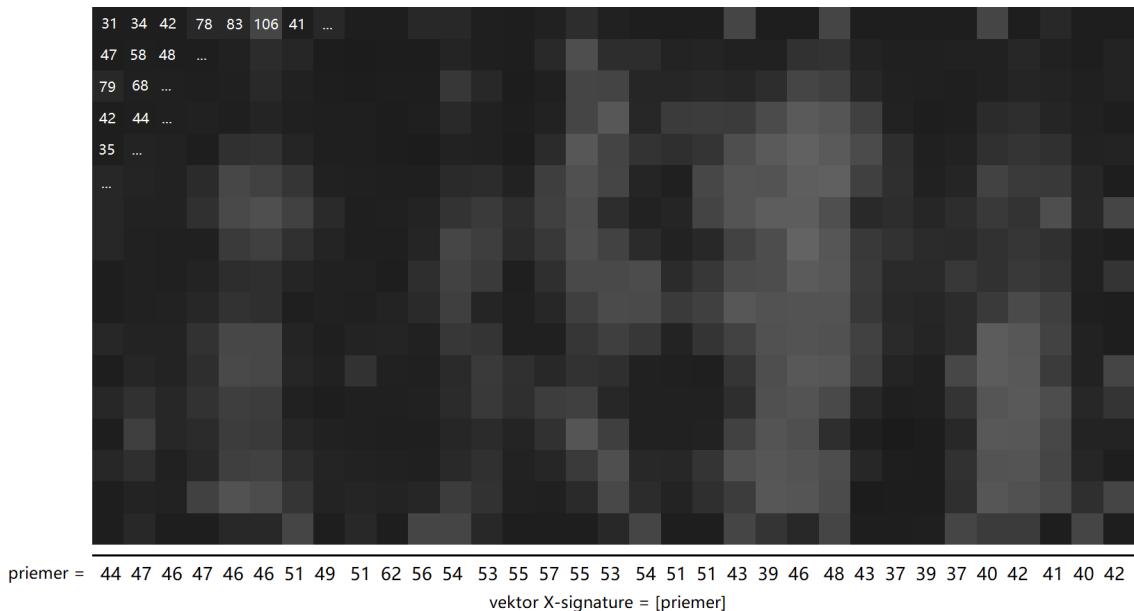
Po natočení vstupného obrazu sa vytvorí virtuálne orientované okno o veľkosti 33x17 so stredom v novom bode  $(x', y')$ .



Obr. 15: Natočenie obrazu podľa bodu  $(x, y)$  okolo stredového bodu  $(p, q)$

### 3.2.5 Zápis do vektora X-signature

Následne sa vytvorí nový vnorený for-cyklus, ktorý bude prechádzať cez všetky pixely v rámci orientovaného okna. Kedže potrebujeme vypočítať priemernú hodnotu jasovej úrovne obrazu v každom stĺpci orientovaného okna, budeme ho prechádzať najskôr po stĺpcoch a v rámci každého stĺpca po riadkoch. Týmto spôsobom sčítame jasové hodnoty vo všetkých riadkoch a vydelíme hodnotou výšky orientovaného okna t.j. 17. Potom tento vypočítaný priemer zapíšeme do vektora X-signature. Vo vektore X-signature sa bude nachádzať len hodnôt, kolko je šírka orientovaného okna a teda 33.

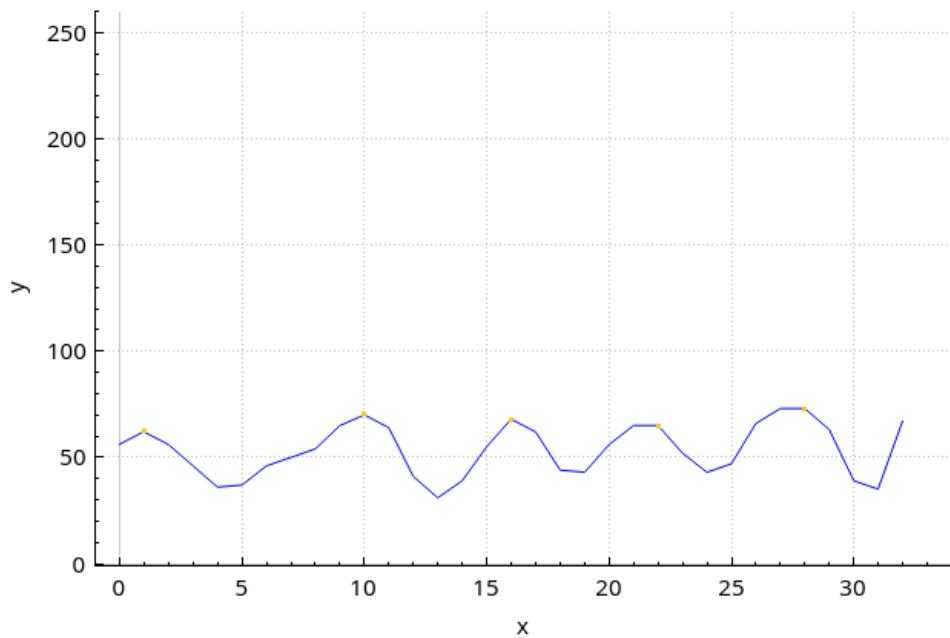


Obr. 16: Príklad výpočtu a zápisu priemernej jasovej hodnoty stĺpcov z orientovaného okna do vektora X-signature

### 3.2.6 Výpočet frekvencie

Pomocou externej knižnice Persistence1D vyhľadáme vo vektore X-signature všetky lokálne maximá s voliteľnou perzistencia. V tomto prípade sa za perzistencia považuje dĺžka trvania konkrétneho lokálneho maxima.

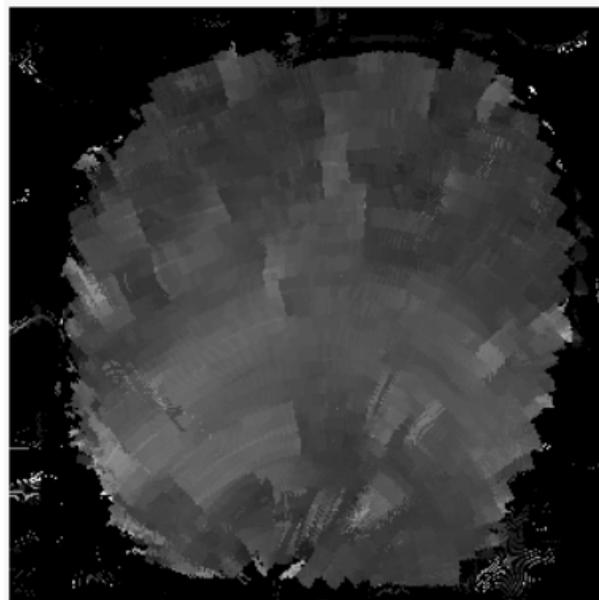
Následne sa počet lokálnych máxim vydelený súčtom vzdialenosí lokálnych máxim. Hodnota tohto delenia je naša výsledná frekvencia pre centrálny bod orientovaného okna.



Obr. 17: Profil lokálneho jasu orientovaného okna s centrom v bode  $(x, y)$ , s vyznačenými lokálnymi maximami a perzistenciaj 5

### 3.2.7 Zápis do matice

Týmto spôsobom vypočítame frekvenciu pre každý bod vstupného obrazu a hodnotu frekvencie postupne zapisujeme do nami na začiatku vytvorennej matice frekvenčnej mapy. Výsledná frekvenčná mapa je zobrazená na obrázku 18.



Obr. 18: Frekvenčná mapa obrázku 10

81	81	81	76	102	107	102	102	97	97	109	109	109	102
81	81	81	76	107	102	102	102	97	97	97	82	82	78
81	76	81	76	107	107	102	102	106	106	106	82	76	76
81	81	81	76	76	111	106	111	106	106	106	82	76	76
81	81	81	85	85	106	111	111	107	107	107	76	76	76
81	85	85	85	85	111	107	107	107	107	107	76	76	76
82	85	85	90	96	107	107	107	107	107	107	107	76	76
82	90	90	90	96	96	107	107	107	107	107	107	106	82
90	90	90	90	96	102	107	107	107	106	106	106	106	82
90	90	90	90	102	102	96	106	106	106	106	106	106	73
90	96	90	96	93	97	97	106	106	106	97	102	102	73
90	90	93	93	93	97	97	97	97	97	97	102	102	73
93	93	93	93	93	93	97	97	97	97	97	97	102	102
93	93	97	93	93	93	97	97	97	97	97	97	97	102

Obr. 19: Detail frekvenčnej mapy s hodnotami frekvencií

Hodnoty frekvenčnej mapy sú prepočítané na 8-bitový grayscale-ový obrázok a teda môžu nadobúdať hodnoty 0 - 255.

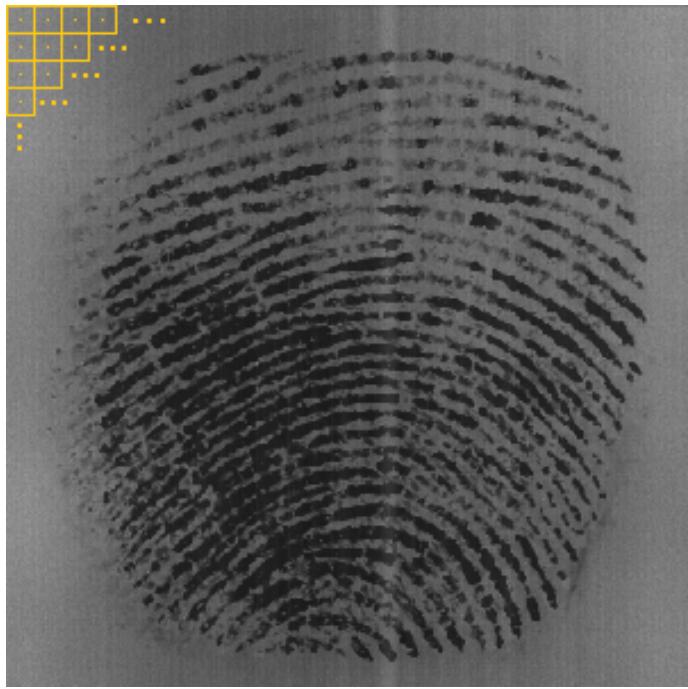
### 3.3 Optimalizácia

Kedže sa jedná o veľmi komplexný algoritmus, pri jeho testovaní sme pozorovali, že algoritmus funguje správne, ale nie je efektívny. Preto sme sa rozhodli optimalizovať jeho rýchlosť a to dvomi spôsobmi: počítaním frekvencie nie pre každý pixel zvlášť ale pre blok pixelov a využitím všetkých dostupných vlákien procesora. Porovnanie výsledkov pred a po optimalizácii je zhrnuté v kapitole 4.

#### 3.3.1 Výpočet frekvenčnej mapy po blokoch

Hlavný rozdiel oproti prvotnému algoritmu je v tom, že sa nebude počítať frekvencia pre všetky body vstupného obrazu, ale len pre centrálny bod definovaného bloku. Tento blok o veľkosti  $W \times W$  je vždy štvorcového tvaru a dĺžka jeho strany je nepárna. Teoreticky môžeme povedať, že rýchlosť algoritmu by sa mala zvýšiť  $W^2$  krát.

Vypočítaná hodnota frekvencie centrálneho bodu bloku je rovnaká pre celý blok. Táto frekvencia je následne zapísaná do frekvenčnej mapy, do všetkých bodov, ktoré sa nachádzajú v bloku. Tento blok je predvolene nastavený na veľkosť  $13 \times 13$  keďže aj smerová mapa sa počíta na bloku o tejto veľkosti.



Obr. 20: Rozdelenie vstupného obrazu na bloky o veľkosti  $13 \times 13$  s vyznačenými centrálnymi bodmi

81	81	81	81	73	73	73	73	73	73	73	73	73	73	73	85	85	85	85
81	81	81	81	73	73	73	73	73	73	73	73	73	73	73	85	85	85	85
81	81	81	81	73	73	73	73	73	73	73	73	73	73	73	85	85	85	85
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
90	90	90	90	120	120	120	120	120	120	120	120	120	120	120	89	89	89	89
85	85	85	85	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73
85	85	85	85	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73
85	85	85	85	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73
85	85	85	85	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73
85	85	85	85	73	73	73	73	73	73	73	73	73	73	73	73	73	73	73

Obr. 21: Detail frekvenčnej mapy s hodnotami frekvencií v blokoch

### 3.3.2 Výpočet frekvenčnej mapy vo vláknach

Framework Qt, v ktorom sme algoritmus vyvíjali, ponúka veľký počet užitočných knižníc. Jednou z nich je aj knižnica QThread, pomocou ktorej sa dá vykonávanie programu rozdeliť na jednotlivé vlákna procesora.



Obr. 22: Rozdelenie vstupného obrazu na 8 vlákien podľa počtu riadkov

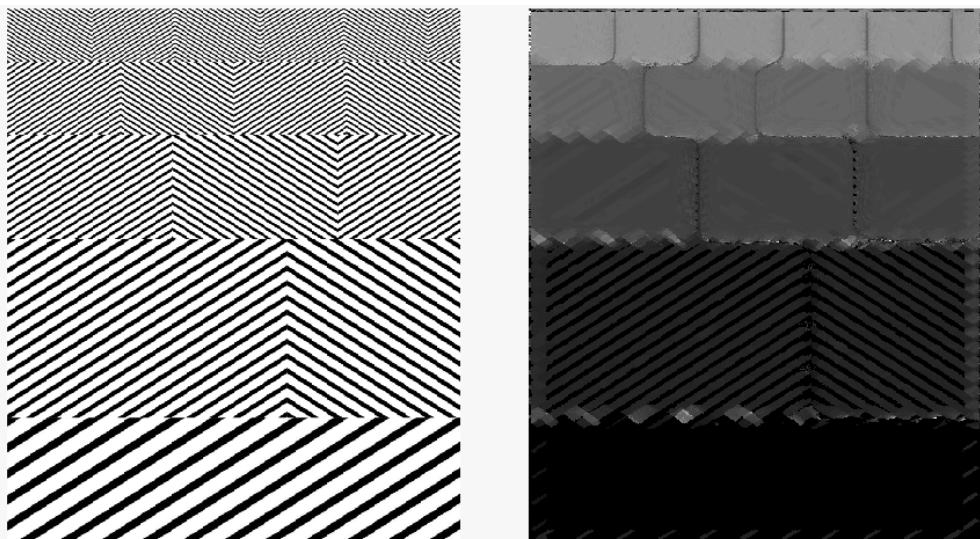
Na začiatku vykonávania nášho algoritmu zistíme ideálny počet vlákien procesora na danom počítači, na ktorom algoritmus spúšťame. Následne rozdelíme vstupný obraz podľa počtu riadkov na toľko častí, koľko vlákien máme k dispozícii a spustíme algoritmus. Keďže sa jedná o asynchronné programovanie, treba vždy počkať na ukončenie všetkých vlákien. Až po ich ukončení je možné frekvenčnú mapu používať v ďalšom procese. Týmto spôsobom by sa mal algoritmus opäť o niečo zrýchliť, keďže sa kód na jednotlivých vláknach vykonáva súčasne.

## 4 Testovanie a výsledky vyvinutého algoritmu

Po úspešnej implementácii algoritmu X-signature bolo potrebné algoritmus pretestovať pre všetky rôzne druhy vstupného obrazu. Napríklad obraz s rôznymi rozmermi, nekvalitne zosnímaný odtlačok prsta, obrazy s rôznym rozlíšením.

### 4.1 Výsledky pred optimalizáciou

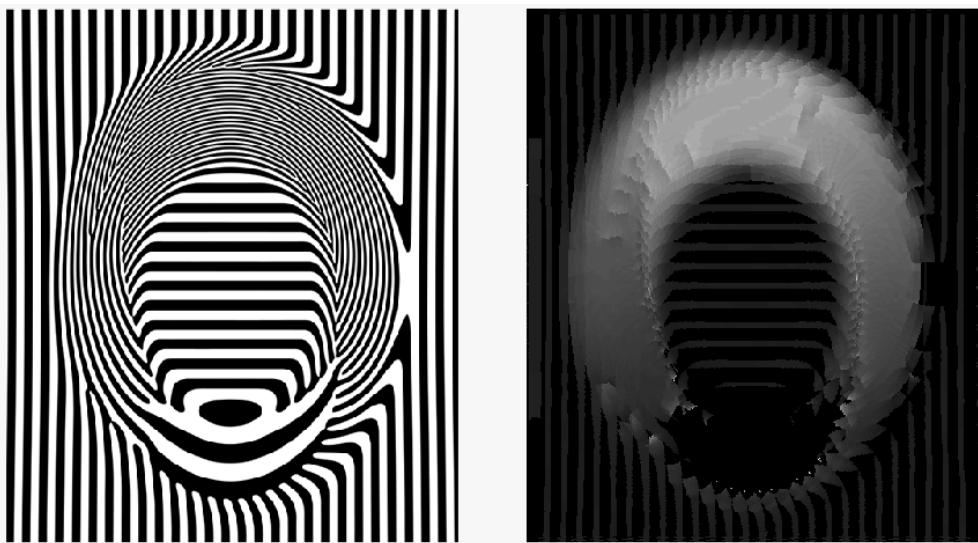
Aby sme si overili, či algoritmus funguje správne, rozhodli sme sa otestovať jeho funkčnosť na malej množine obrázkov s rôznymi vzormi, na ktorých budeme môcť čo najlepšie pozorovať zmenu frekvencií týchto vzorov. Vo výslednej frekvenčnej mape indikuje svetlá farba oblasť s väčšou frekvenciou líní a tmavá farba oblasť s menšou frekvenciou líní.



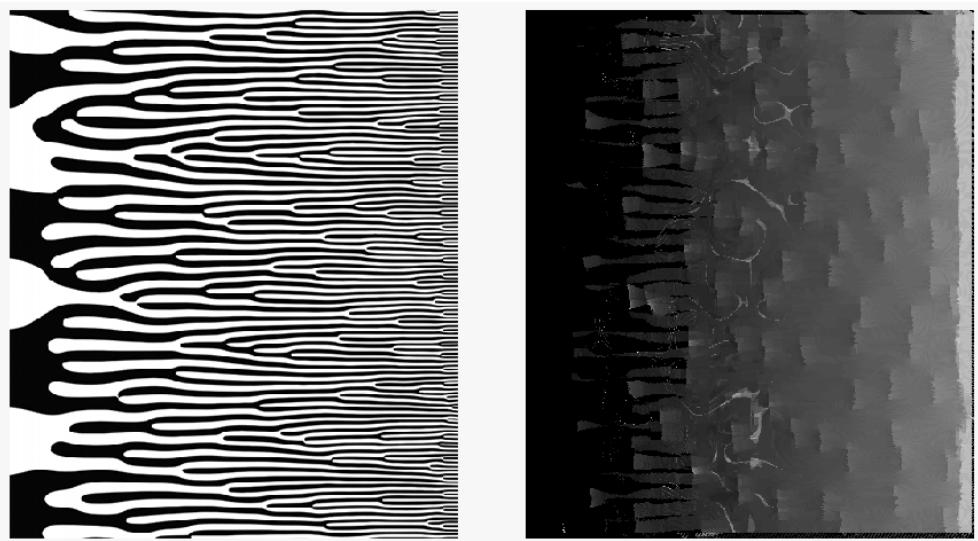
Obr. 23: Obrázok s kalibračným vzorom o veľkosti  $420 \times 777$ .

Vľavo vstupný obraz, vpravo vypočítaná frekvenčná mapa.

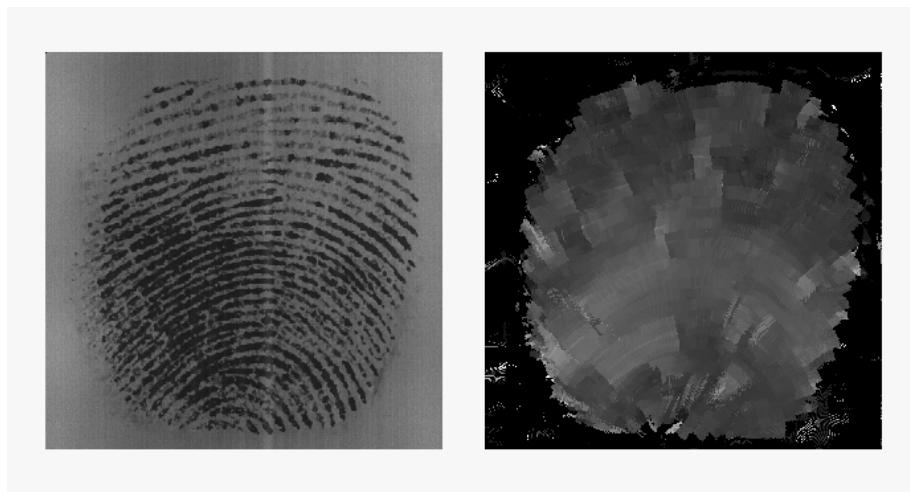
Po pozorovaní obrázkov 23, 24 a 25 môžeme povedať, že algoritmus počíta frekvenčnú mapu správne. Preto sme sa rozhodli otestovať algoritmus na rôznych odtlačkoch prsta z databázy FVC2004. Na obrázkoch 26 a 27 sú príklady kvalitne a nekvalitne zosnímaného odtlačku prsta z použitej databázy.



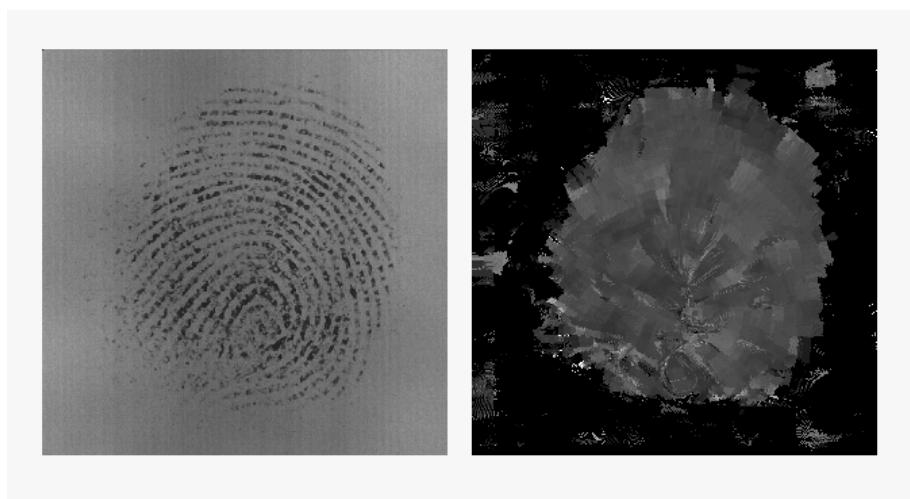
Obr. 24: Obrázok so vzorom o veľkosti  $500 \times 506$ .  
Vľavo vstupný obraz, vpravo vypočítaná frekvenčná mapa.



Obr. 25: Obrázok so vzorom o veľkosti  $500 \times 503$ .  
Vľavo vstupný obraz, vpravo vypočítaná frekvenčná mapa.



Obr. 26: Príklad kvalitne zosnímaného odtlačku prsta o veľkosti 300x300.  
Vľavo vstupný obraz, vpravo vypočítaná frekvenčná mapa.

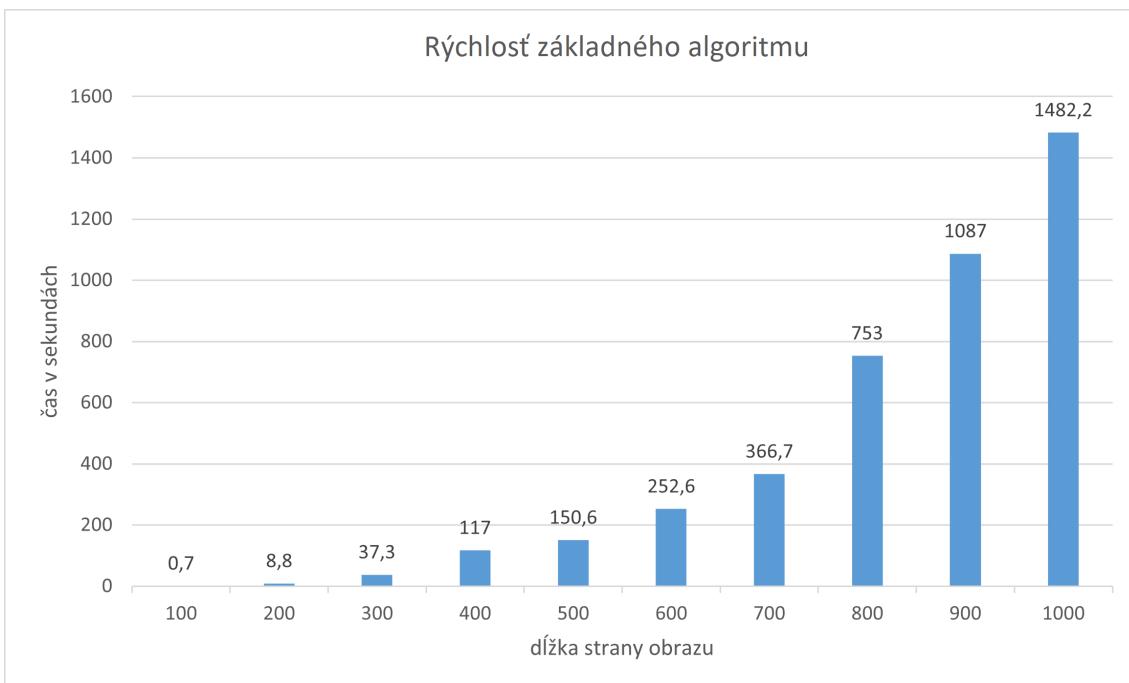


Obr. 27: Príklad nekvalitne zosnímaného odtlačku prsta o veľkosti 300x300.  
Vľavo vstupný obraz, vpravo vypočítaná frekvenčná mapa.



Obr. 28: Vstupný obraz použitý na testovanie rýchlosťi algoritmu

Vyvinutý algoritmus počíta frekvenčnú mapu správne, ale nie efektívne. Na grafe na obrázku 29 je zobrazený čas trvania výpočtu frekvenčnej mapy v sekundách, pri rôznych veľkostach vstupného obrazu. Ako vstupný obraz sme použili odtlačok prsta z obrázku 28 o veľkosti  $1000 \times 1000$  a postupne sme rovnomerne zmenšovali jeho strany a výsledný čas výpočtu frekvenčnej mapy zapisovali do grafu na obrázku 29.



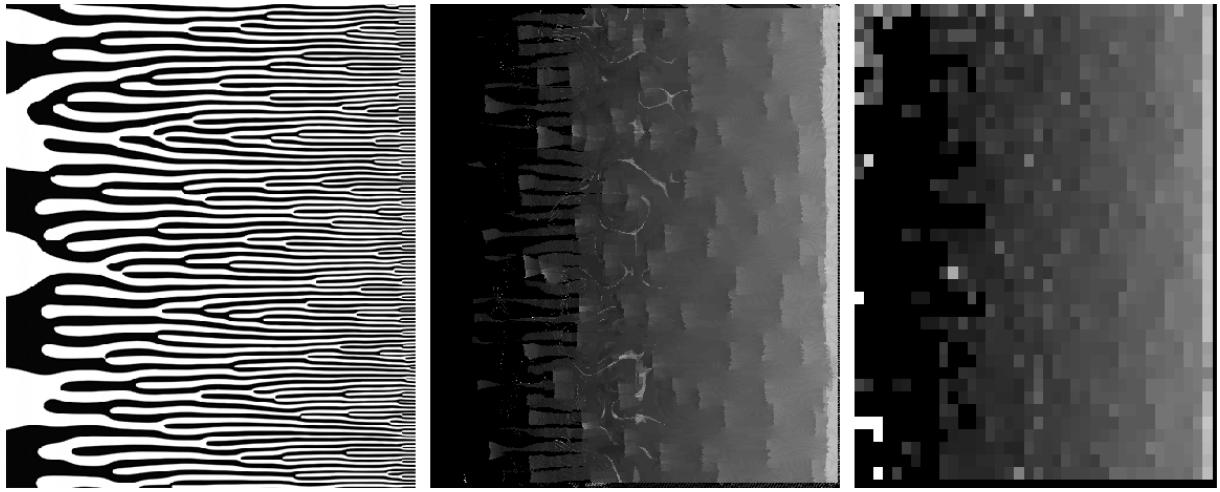
Obr. 29: Graf zobrazujúci rýchlosť výpočtu frekvenčnej mapy vzhľadom na rôznu veľkosť vstupného obrazu pri základnej verzii algoritmu

Práve kvôli týmto neuspokojivým výsledkom rýchlosť nášho algoritmu sme sa ho rozhodli optimalizovať. Optimalizácia vyvinutého algoritmu je opísaná v podkapitole 3.3.

## 4.2 Výsledky po optimalizácii

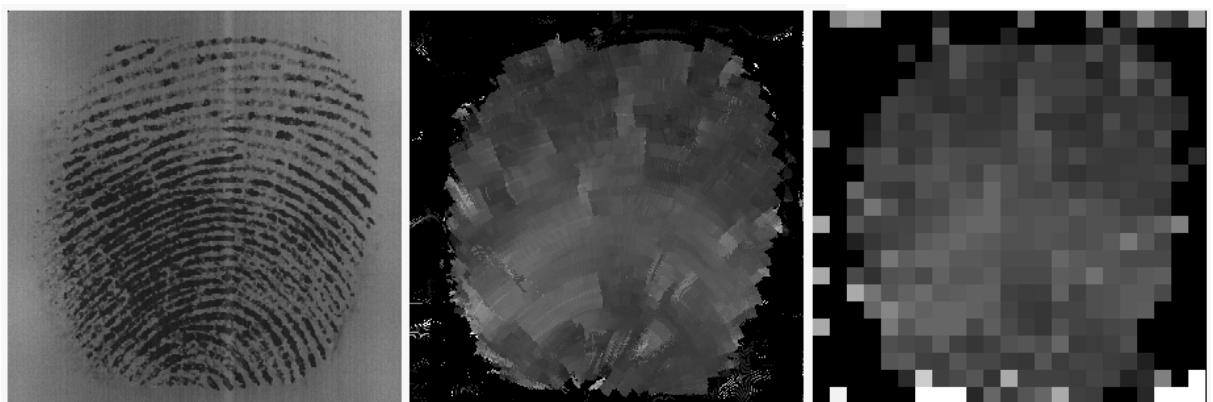
### 4.2.1 Výsledky po optimalizácii výpočtu frekvenčnej mapy po blokoch

Po úspešnej implementácii tejto optimalizácie bolo potrebné otestovať jej funkčnosť. Testovanie sme opäť začali s obrázkami so vzormi a postupne pokračovali na obrázkoch odtlačkov prstov z databázy FVC2004. Na obrázkoch nižšie sú zobrazené výsledky, aké sme dosiahli.



Obr. 30: Obrázok so vzorom o veľkosti  $500 \times 503$ .

Vľavo vstupný obraz, v strede frekvenčná mapa vypočítaná základným algoritmom, vpravo vypočítaná frekvenčná mapa s optimalizáciou výpočtu po blokoch o veľkosti  $13 \times 13$ .

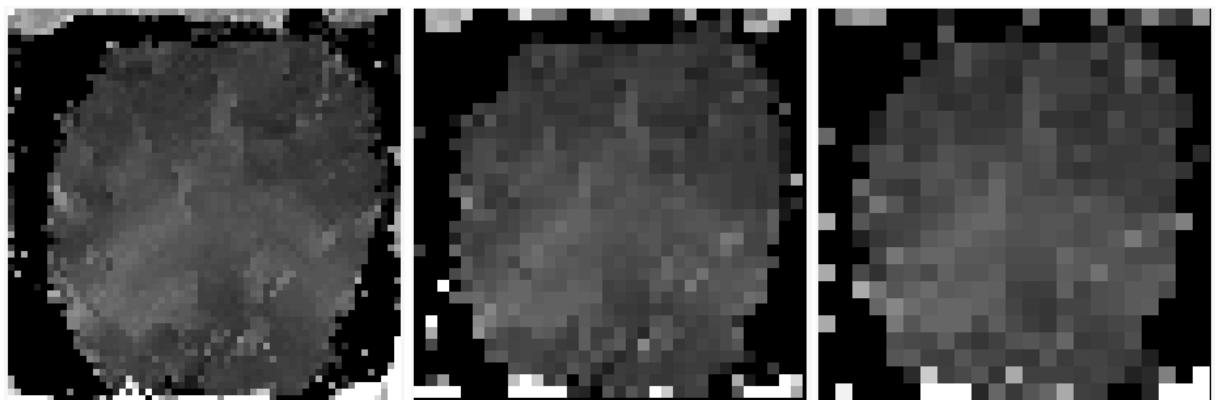


Obr. 31: Príklad kvalitne zosnímaného odtlačku prsta o veľkosti  $300 \times 300$ .

Vľavo vstupný obraz, v strede frekvenčná mapa vypočítaná základným algoritmom, vpravo vypočítaná frekvenčná mapa s optimalizáciou výpočtu po blokoch o veľkosti  $13 \times 13$ .

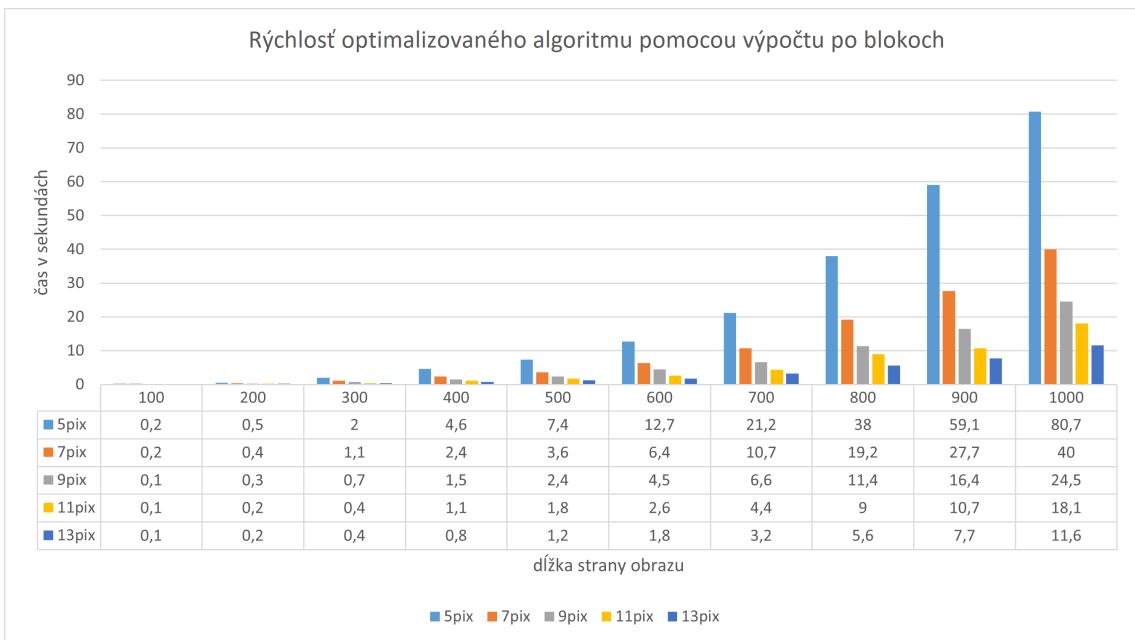


Obr. 32: Príklad nekvalitne zosnímaného odtlačku prsta o veľkosti 300x300.  
Vľavo vstupný obraz, v strede frekvenčná mapa vypočítaná základným algoritmom,  
vpravo vypočítaná frekvenčná mapa s optimalizáciou výpočtu po blokoch o veľkosti  
 $13 \times 13$ .



Obr. 33: Porovnanie rôznych veľkostí blokov na odtlačku prsta o veľkosti 300x300.  
Vľavo vypočítaná frekvenčná mapa s optimalizáciou výpočtu po blokoch o veľkosti  
 $5 \times 5$ , v strede vypočítaná frekvenčná mapa s optimalizáciou výpočtu po blokoch o  
veľkosti  $9 \times 9$ , vpravo vypočítaná frekvenčná mapa s optimalizáciou výpočtu po blokoch  
o veľkosti  $13 \times 13$ .

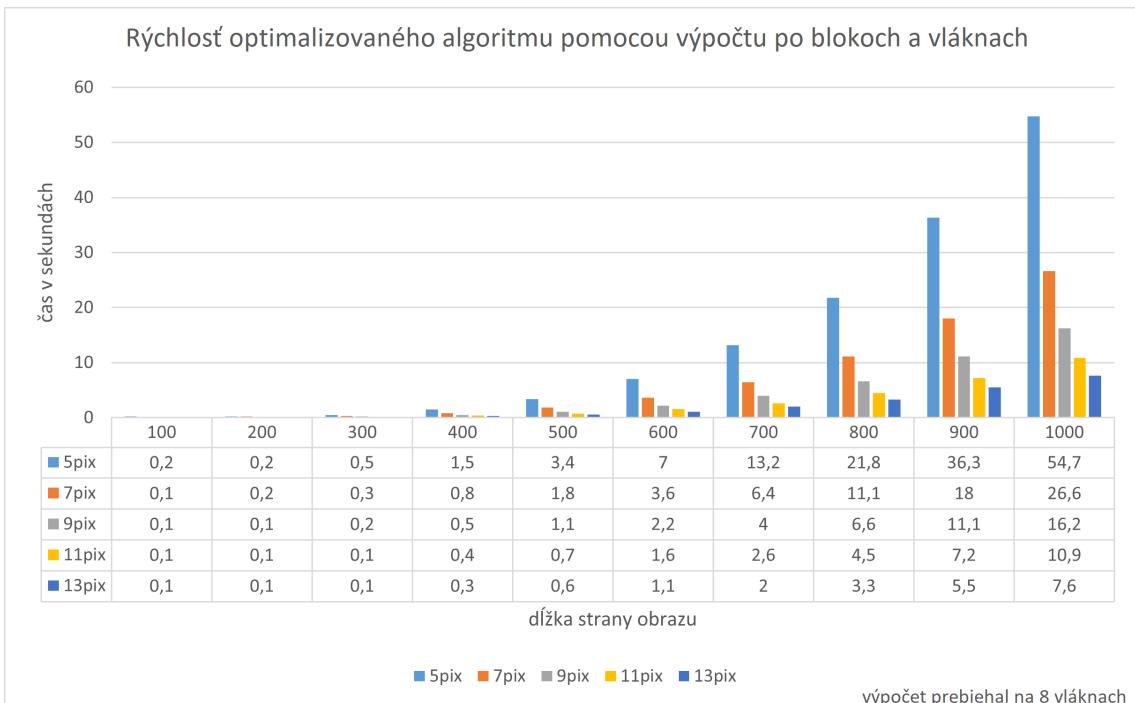
Pri sledovaní výsledkov sme pozorovali, že optimalizáciou nášho algoritmu sa výsledky frekvenčnej mapy v rámci tolerancie jemne zhoršili (hlavne pri okrajoch), ale trvanie výpočtu značne skrátilo. Podobne ako pri základnom algoritme sme rýchlosť merali na obrázku 28 o veľkosti  $1000 \times 1000$  a postupne rovnomerne zmenšovali jeho strany a výsledný čas výpočtu frekvenčnej mapy zapisovali do grafu na obrázku 34. V grafe môžeme sledovať, že výpočet frekvenčnej mapy sa pri použití optimalizácie výpočtu na blokoch o veľkosti  $5 \times 5$  zrýchliл v priemere 18-krát a pri bloku o veľkosti  $13 \times 13$  dokonca 107-krát oproti pôvodnej verzii algoritmu.



Obr. 34: Graf zobrazujúci rýchlosť výpočtu frekvenčnej mapy vzhľadom na rôznu veľkosť vstupného obrazu pri optimalizovanej verzii algoritmu pomocou výpočtu po blokoch

#### 4.2.2 Výsledky po optimalizácii výpočtu frekvenčnej mapy pomocou vlákien

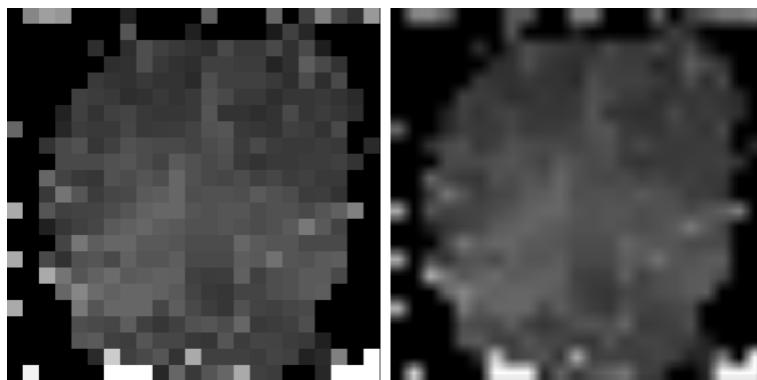
Túto optimalizáciu sme testovali spoločne s predchádzajúcim. Rýchlosť danej optimalizácie sme opäť merali na obrázku 28 o veľkosti  $1000 \times 1000$  a postupne rovnomerne zmenšovali jeho strany po veľkosť  $100 \times 100$ . Výsledky, ktoré sme dosiahli sú zobrazené v grafe na obrázku 35. Ako môžeme vidieť, pri použití obidvoch optimalizácií vieme pôvodný algoritmus, pri veľkosti bloku  $5 \times 5$  a ôsmich vláknach, zrýchliť v priemere 40-krát a pri bloku  $13 \times 13$  a ôsmich vláknach až 214-krát oproti pôvodnej verzii algoritmu.



Obr. 35: Graf zobrazujúci rýchlosť výpočtu frekvenčnej mapy vzhľadom na rôznu veľkosť vstupného obrazu pri optimalizovanej verzii algoritmu pomocou výpočtu po blokoch a rozdelenia výpočtu na vlákna

#### 4.2.3 Pridanie rozmazania Gaussian blur

Gaussian blur je metóda knižnice OpenCV, ktorá slúži zvyčajne na elimináciu šumu alebo redukciu detailov. V našom algoritme túto metódu používame konkrétnie na splynutie blokov vo frekvenčnej mape, aby sme dosiahli vizuálne krajsie výsledky a frekvenčnú mapu neovplyvnenú šumom.



Obr. 36: Použitie rozmazania Gaussian blur vo vypočítanej frekvenčnej mape s blokom rozmazania o veľkosti 13 a sigmou o veľkosti 6. Vľavo frekvenčná mapa bez rozmazania, vpravo frekvenčná mapa s rozmazaním.

## 4.3 GUI aplikácia

Pri testovaní a porovnávaní výsledkov bolo potrebné často meniť nastavenia výpočtu frekvenčnej mapy tak, aby sme dosahovali čo najlepšie výsledky. Preto bola vytvorená GUI aplikácia, zobrazená na obrázku 37, pomocou ktorej sa dajú jednoducho nastavovať parametre výpočtu frekvenčnej mapy.

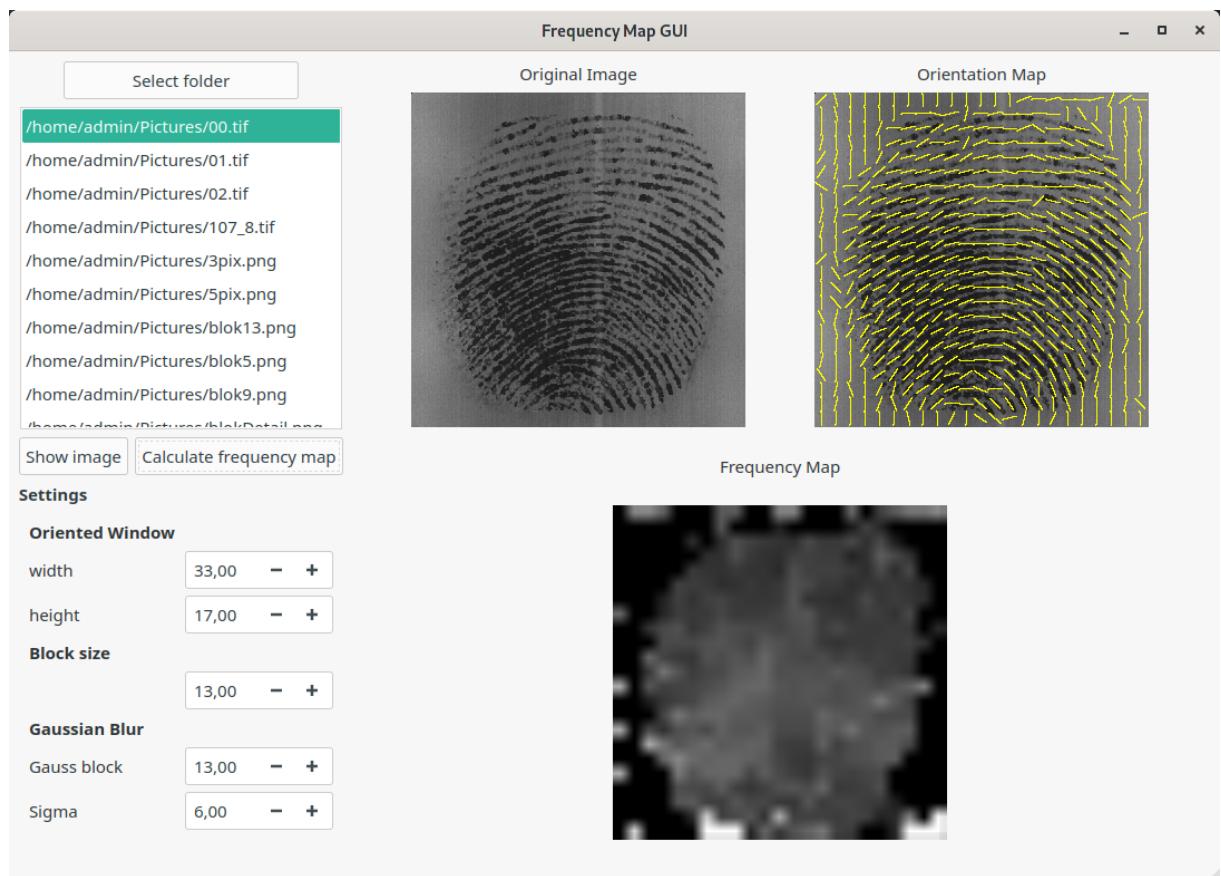
### 4.3.1 Nastavovanie parametrov pri výpočte frekvenčnej mapy

Pri výpočte frekvenčnej mapy je možné nastavovať nasledujúce parametre:

- výška orientovaného okna (predvolene 17) - týmto parametrom nastavujeme výšku orientovaného okna; hodnota musí byť nepárna
- šírka orientovaného okna (predvolene 33) - týmto parametrom nastavujeme šírku orientovaného okna; hodnota musí byť nepárna
- veľkosť bloku (predvolene 13) - týmto parametrom nastavujeme veľkosť bloku smerovej a frekvenčnej mapy; hodnota musí byť nepárna
- veľkosť rozmazania (predvolene 13) - týmto parametrom nastavujeme veľkosť bloku pre operáciu Gaussian blur; hodnota musí byť nepárna
- sigma (predvolene 6) - týmto parametrom nastavujeme smerodajnú odchýlku pre operáciu Gaussian blur

### 4.3.2 Používanie GUI aplikácie

Aby sme mohli vypočítať frekvenčnú mapu, potrebujeme nejaký vstupný obrázok. Pomocou tlačidla *Select folder* vieme načítať nejaký konkrétny adresár, z ktorého sa vypíšu v zozname vľavo všetky súbory s obrazmi. Výber konkrétneho obrázku vykonáme kliknutím na jeho názov. Nami vybraný obrázok si môžeme zobraziť pomocou tlačidla *Show image*. Následne tlačidlom *Calculate frequency map* vypočítame a zobrazíme frekvenčnú mapu spolu so smerovou mapou.



Obr. 37: GUI aplikácia

# Záver

Cieľom tejto bakalárskej práce bolo implementovať metódu výpočtu frekvenčnej mapy X-signature a pridať ju do biometrického systému OpenFinger. Na to, aby sme to mohli vykonať, sme sa najskôr v teoretickej časti oboznámili s informáciami, potrebnými pre ďalšie štádiá implementácie. Pri vyhľadávaní zdrojov ohľadom výpočtu frekvenčnej mapy sme zistili, že algoritmus výpočtu frekvenčnej mapy je častokrát len opísaný v nejakej práci a je veľmi obtiažne nájsť open-source zdrojový kód na výpočet frekvenčnej mapy. Po dôkladnom preštudovaní rôznych zdrojov, bolo pre nás neskôr jednoduchšie upraviť si časti metódy X-signature tak, aby sa algoritmus vykonával efektívnejšie. Otestovali sme funkčnosť biometrického systému OpenFinger, ktorého výstup smerovej mapy je potrebný na vykonanie výpočtu frekvenčnej mapy. Následne sme úspešne implementovali metódu X-signature vo forme dynamickej C++ knižnice, čo zjednoduší integráciu frekvenčnej mapy do ľubovoľného biometrického systému. V kapitole 4 sme zhodnotili úspešnosť a efektivitu vyvinutého algoritmu. Aby sa testovanie vykonávalo jednoduchšie, vytvorili sme jednoduchú GUI aplikáciu na nastavovanie rôznych parametrov a vizualizáciu smerovej a frekvenčnej mapy z pôvodného obrázka. Tu sme zistili, že efektivita algoritmu zaostáva za našimi očakávанияmi a preto sme sa rozhodli algoritmus optimalizovať. Optimalizovali sme ho dvoma spôsobmi, ktoré sme na konci spojili, aby sme dosiahli čo najlepšie výsledky. Po úspešnej implementácii optimalizácií a ich otestovaní sa nám podarilo algoritmus zrýchliť.

Vďaka výsledkom našej práce sme rozšírili Gaborov filter v systéme OpenFinger o frekvenčnú mapu, ktorá pozitívne vplýva na výsledné predspracovanie obrazu. Stále je tu však priestor na zlepšenie a to implementovaním nejakej inej metódy výpočtu frekvenčnej mapy, ktorá môže byť ešte efektívnejšia. Potenciál ukazuje metóda založená na sledovaní priebehu papilárnej línie. Táto metóda, podľa toho ako ju opísal autor Zhang L., by mala byť vylepšením metódy X-signature, ktorú sme implementovali my.

Kedže sa nám podarilo úspešne implementovať a optimalizovať algoritmus výpočtu frekvenčnej mapy, konkrétnie metódu X-signature, môžeme skonštatovať, že cieľ tejto bakalárskej práce bol splnený.

# Zoznam použitej literatúry

1. MAIO, D., MALTONI, D., CAPPELLI, R., WAYMAN, J. L. a JAIN, A. K. *FVC2002: Fingerprint Verification Competition: Proc. Medzinárodná konferencia o uznávaní vzorov (ICPR)*. Quebec City, Kanada, 2002.
2. AMARPREET, S., RAKESH, K. a GURPREET, S. *Biometric Recognition: A Modern Era For Security*. online, 2010.
3. MALTONI, D., MAIO, D., JAIN, A.K. a PRABHAKAR, S. *Handbook of Fingerprint Recognition*. Springer, London, 2009. ISBN 978-1-84882-253-5.
4. RATHA, N. K. a GOVINDARAJU, V. *Advances in Biometrics: Sensors, Algorithms and Systems*. Springer, London, 2008. ISBN 978-1-84628-920-0.
5. MARÁK, Pavol. *OpenFinger: Systém na daktyloskopickú autentifikáciu s GPU akceleráciou [Dizertačná práca]*. FEI STU, Bratislava, 2020.
6. ASHOK, J., SHIVASHANKAR, V. a MUDIRAJ, P. V. G. S. *An Overview of Biometrics*. 2010.
7. ZHANG, L. Fingerprint ridge distance estimation based on ridge search. In: *International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*. 2009, s. 604–607.
8. LEHTIHET, R. Ridge Frequency Estimation for Low-Quality Fingerprint Image Enhancement Using Delaunay Triangulation. In: *International Journal of Pattern Recognition and Artificial Intelligence*. 2014, roč. 28, č. 1.
9. STROUSTRUP, Bjarne. The C++ Programming Language. 2013, č. 4.
10. MARÁK, Pavol. OpenFinger: systém na daktyloskopickú autentifikáciu s GPU akceleráciou [Autoreferát dizertačnej práce]. 2020, s. 5–6.

# Prílohy

A Štruktúra elektronického nosiča . . . . .	II
---	----

# A Štruktúra elektronického nosiča

*/Bakalarska\_praca.pdf*

- bakalárska práca v .pdf verzii

**/db**

- databáza odtlačkov prstov FVC2004

**/algoritmus**

- zdrojové súbory

*/freqMap.cpp*

- zdrojový kód algoritmu výpočtu frekvenčnej mapy

*/freqMap.h*

- hlavičkový súbor algoritmu výpočtu frekvenčnej mapy

*/libFreqMap.pro*

.

*/libFreqMap.pro.user*

.

*/main.cpp*

- hlavný zdrojový súbor

*/mainwindow.cpp*

.

*/mainwindow.h*

.

*/mainwindow.ui*

- zdrojový kód GUI

*/persistiance1d.h*

- hlavičkový súbor knižnice na hľadanie lokálnych extrémov