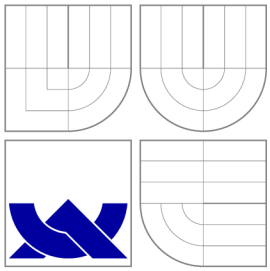
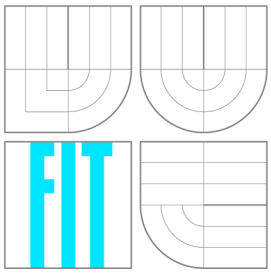


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

KLASIFIKÁTOR BIOMETRICKÝCH OBRAZOVÝCH DAT

BIOMETRIC IMAGE DATA CLASSIFIER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK TRETTER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL DOLEŽEL

BRNO 2014

Abstrakt

Cílem této práce je navrhnout a implementovat klasifikátor otisků prstů, který klasifikuje otisky prstů na základě typu snímače, ze kterého byly nasnímány. Čtenáři jsou v práci popsány existující typy snímačů otisků prstů a jednotlivé fáze klasifikace. Navržený klasifikátor využívá kaskády klasifikátorů vytrénovaných učícím algoritmem AdaBoost. Aplikace byla implementovaná v jazyce C++, s využitím knihovny OpenCV, pro operační systémy GNU/Linux a MS Windows.

Abstract

The aim of this thesis is to design and implement fingerprint classifier, which classifies the fingerprints based on the scanner used. Reader is presented with existing types of fingerprint scanners and phases of classification. Designed classifier is using a cascade of classifiers, trained using the AdaBoost learning algorithm. The application was implemented in the C++ language using OpenCV library for operational systems GNU/Linux and MS Windows.

Klíčová slova

otisk prstu, klasifikace, AdaBoost, kaskáda klasifikátorů, obrazové filtry, C++, OpenCV

Keywords

fingerprint, classification, AdaBoost, cascade of classifiers, image filters, C++, OpenCV

Citace

Zdeněk Tretter: Klasifikátor biometrických obrazových dat, bakalářská práce, Brno, FIT VUT v Brně, 2014

Klasifikátor biometrických obrazových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Doležela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Zdeněk Tretter
20. května 2014

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Michalovi Doleželovi za užitečné rady při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Cílem této práce je navrhnout a implementovat klasifikátor biometrických obrazových dat. Pro řešení této práce byla vybrána klasifikace otisků prstů na základě typu snímače, z jakého byly otisky pořízeny.

V prvních dvou kapitolách si nejprve probereme nezbytnou teorii potřebnou k pochopení problematiky a následných postupů při návrhu klasifikátoru. V kapitole 2 si ukážeme, jaké typy snímačů otisků prstů exitují a na jakých principech pracují. V kapitole 3 jsou pak popsány jednotlivé fáze klasifikace.

V kapitole 4 si představíme podrobný návrh aplikace a v kapitole 5 si poté popíšeme implementaci tohoto návrhu pro operační systémy MS Windows a GNU/Linux v jazyce C++ s využitím knihovny OpenCV.

V kapitole 6 si ukážeme jakým způsobem byla aplikace testována, konkrétně testování jednotlivých stupňů klasifikátoru a jejich nastavení. V kapitole 7 si následně ukážeme jakých výsledků finální klasifikátor dosahuje.

V závěru zhodnotíme výsledky práce a navrhneme možná vylepšení do budoucna.

Kapitola 2

Metody snímání otisku prstu

V dnešní době existuje mnoho přístrojů umožňujících snímání otisků prstů, které jsou využívány v různých oblastech lidské činnosti. Tyto přístroje ke snímání otisků využívají jednu z několika různých metod.

Podle metody snímání lze snímače v těchto přístrojích rozdělit na *optické*, popsané v sekci 2.2, *polovodičové* popsané v sekci 2.3 a *ultrazvukové* popsané v sekci 2.4. Tyto metody se označují jako *live-scan* snímání otisků prstů, manuální metoda využívající ke snímání *inkoust* popsaná v sekci 2.1, se označuje jako *off-line* snímání otisků prstů [10].

Samotné přístroje lze pak rozdělit na [10]:

- **více-prstové** – snímají více prstů naráz,
- **jedno-prstové** – snímají naráz pouze jeden prst.

Otisky prstů vytvořené snímači, které pracují na různých technologiích, vypadají odlišně, jak je vidět na obrázku 2.1. FBI a další organizace zabývající se rozsáhlým využitím biometrických technologií vydaly specifikace pro zajištění maximální kompatibility mezi různými typy snímačů otisků prstů. U otisku stejného prstu získaného z různých přístrojů, které tyto specifikace nesplňují, může být správné rozeznání velmi obtížné [10].

Obrázek 2.1: Otisk stejného prstu na různých typech snímačů. Zleva: Optický (FTIR), Polovodičový (Kapacitní), Polovodičový (Tepelný). Zdroj: [10].

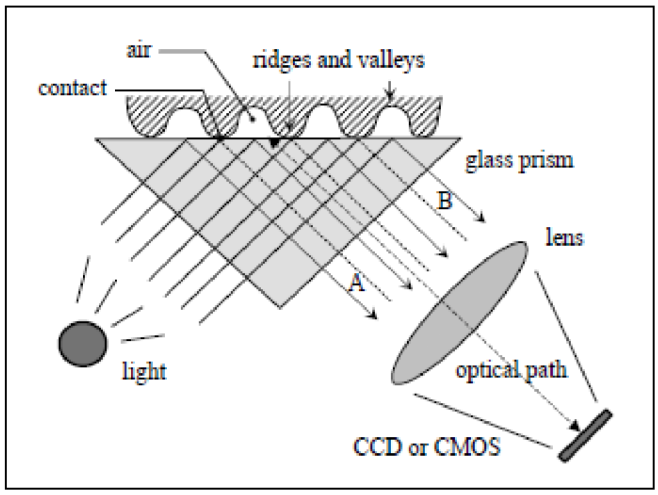
2.1 Inkoust

Snímání otisku pomocí černého inkoustu (tzv. daktyloskopického) je nejstarší známá metoda pro získání otisku prstu. Sejmutí otisku prstu se prování nanesením vrstvy inkoustu na snímanou část prstu a následným přitisknutím prstu na papírovou kartu k tomu určenou. K převedení otisků z papírové karty do digitální podoby se používají univerzální scannery na papír [10].

2.2 Optické snímače

Frustrated Total Internal Reflection (FTIR)

Jak je znázorněno na obrázku 2.2, snímač založený na technologii FTIR se skládá ze zdroje světla, který z jedné strany svítí na skleněný nebo plastový hranol, na jehož horní hraně se paprsky lámou. Paprsky vystupující z druhé strany hranolu jsou přes čočku usměrněny na CCD nebo CMOS senzor. Přiložením prstu na snímač se hřebeny papilárních linií dotknou povrchu horní hrany hranolu a změní se optické vlastnosti v místě dotyku. V místech kde jsou údolí, se paprsky nadále odráží a směřují do senzoru, ale v místech dotyku hřebenů se paprsky náhodně rozptylují (absorbují). Díky tomu jsou po nasnímání otisku hřebeny znázorněny tmavě a údolí světle [10].



Obrázek 2.2: Princip FTIR snímače otisků prstů. Zdroj: [10].

Kvůli velikosti hranolu není snímač příliš tenký. Ztenčit snímač lze použitím ne jednoho hranolu, ale mnoha menších vedle sebe. Avšak kvalita nasnímaných otisků u takových snímačů bývá nižší [10].

Technologie optických vláken

Tenký snímač lze získat také použitím optických vláken místo hranolu. U takového snímače se prst dotýká jedné strany plochy optických vláken a CCD nebo CMOS senzor je na straně druhé. Kvůli absenci čočky usměrňující paprsky vystupující z vláken, musí CCD/CMOS senzor zabírat celou plochu snímače, což cenu takového snímače značně prodražuje [10].

Elektrooptická technologie

Snímače založené na elektrooptické technologii se skládají ze dvou vrstev. První vrstva je tvořena polymerem, který při polarizaci správným napětím ze strany, kde se dotýká prst, vyzařuje světlo. Jelikož hřebeny papilárních linií se polymeru dotýkají a údolí ne, je generovaný potenciál na různých místech odlišný, a tím i vyzařované množství světla. To umožňuje druhé vrstvě pevně spojené s první vrstvou, kterou tvoří pole fotodiod, světlo zachytit a převést na digitální obraz otisku prstu [10].

Technologie přímého čtení

U této metody se využívá vysoce kvalitního fotoaparátu pro nasnímání otisku. Tato metoda je bezdotyková a díky tomu nevznikají problémy, jako je například nelineární zkreslení v důsledku přitisknutí prstu na sklo snímače. Nicméně získání vysoce kontrastního obrazu je poměrně náročné [10].

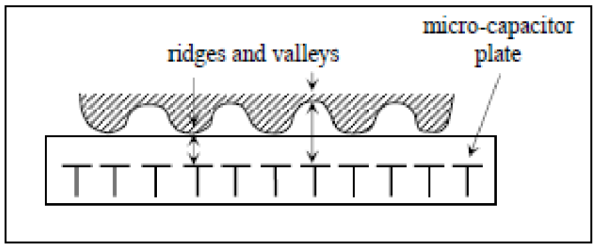
Technologie multispektrálního zobrazování

Multispektrální snímače zachytí více snímku stejného prstu s využitím různých vlnových délek světla, různé orientace osvětlení a různých podmínek polarizace. Výsledné snímky lze využít k sestavení jednoho složeného obrazu otisku prstu [10].

2.3 Polovodičové snímače

Kapacitní technologie

Tento snímač, jak je vidět na obrázku 2.3, je tvořen dvourozměrným polem elektrod mikro-kondenzátorů vloženým do čipu. Druhou elektrodu každého mikro-kondenzátoru tvoří prst přiložený přímo na čip. Mezi elektrodami mikro-kondenzátorů na čipu a kůží na prstu se vytváří malý elektrický náboj, jehož velikost záleží na vzdálenosti kůže prstu od elektrod v čipu. Tak je možno odlišit hřebeny papilárních linií od údolí [10].



Obrázek 2.3: Princip kapacitního snímače otisků prstů. Zdroj: [10].

Tepelná technologie

Snímače pracující na této technologii jsou vyrobeny z pyroelektrického materiálu, který vytváří proud na základě rozdílu teplot. Hřebeny papilárních linií jsou při snímání otisku v kontaktu s povrchem snímače, tím vytváří jiný teplotní rozdíl vůči povrchu snímače než údolí, která jsou od povrchu v určité vzdálenosti [10].

Technologie elektrického pole

Snímač se skládá z generátoru radiofrekvenčního sinusového signálu a z matice aktivních antén, které přijímají velmi malou amplitudu signálu vysílaného generátorem a modulovanou strukturou škáry (vrstva pod pokožkou prstu) [10].

Piezoelektrická technologie

Snímače založené na této technologii jsou citlivé na tlak. Povrch snímače je vyroben z nevodivého dielektrického materiálu, který při dotyku prstu generuje malé množství proudu. Vzhledem k tomu, že hřebeny papilárních linií jsou při dotyku blíže povrchu snímače, vyvíjí na něj jiný tlak než vzdálenější údolí. To má za následek i rozdílné množství vygenerovaného proudu [10].

2.4 Ultrazvukové snímače

Snímače založené na technologii ultrazvuku se skládají ze dvou hlavních částí. Z vysílače, který vysílá krátké zvukové signály směrem k prstu a přijímače, který zachycuje odražené signály. Zachycené signály se poté využívají k výpočtu hloubky obrazu otisku a následně k výpočtu struktury hřebenů papilárních linií [10].

Kapitola 3

Klasifikace

Na začátku této kapitoly jsou popsány obrazové filtry použité v této práci. Následuje popis procesu tvorby silného klasifikátoru s využitím kaskády klasifikátorů vytrénovaných algoritmem AdaBoost a na závěr kapitoly je zmíněn také způsob, jakým lze zhodnotit úspěšnost takto vytvořeného klasifikátoru.

3.1 Obrazové filtry

Prahování

Prahování je výpočetně nenáročná metoda, používaná k segmentaci obrazu [3]. V základní variantě pracuje s šedotónovým obrazem, ve kterém má za cíl odlišit objekt v obraze od pozadí. Hodnota prahu T se většinou určuje na základě histogramu. Transformace vstupního obrazu A na výstupní obraz B se provádí pro každý pixel podle následujícího vzorce:

$$B_{ij} = \begin{cases} 1 & \text{pro } A_{ij} \geq T \\ 0 & \text{pro } A_{ij} < T \end{cases} \tag{3.1}$$

Pokud je potřeba provést segmentaci více objektů v obraze, které jsou vykresleny dostatečně odlišnými šedými odstíny, zvolí se více prahů. Nicméně výsledný obraz již nebude obsahovat pouze bílou a černou barvu, ale i šedé odstíny potřebné k odlišení různých objektů.

Diskrétní 2D konvoluce

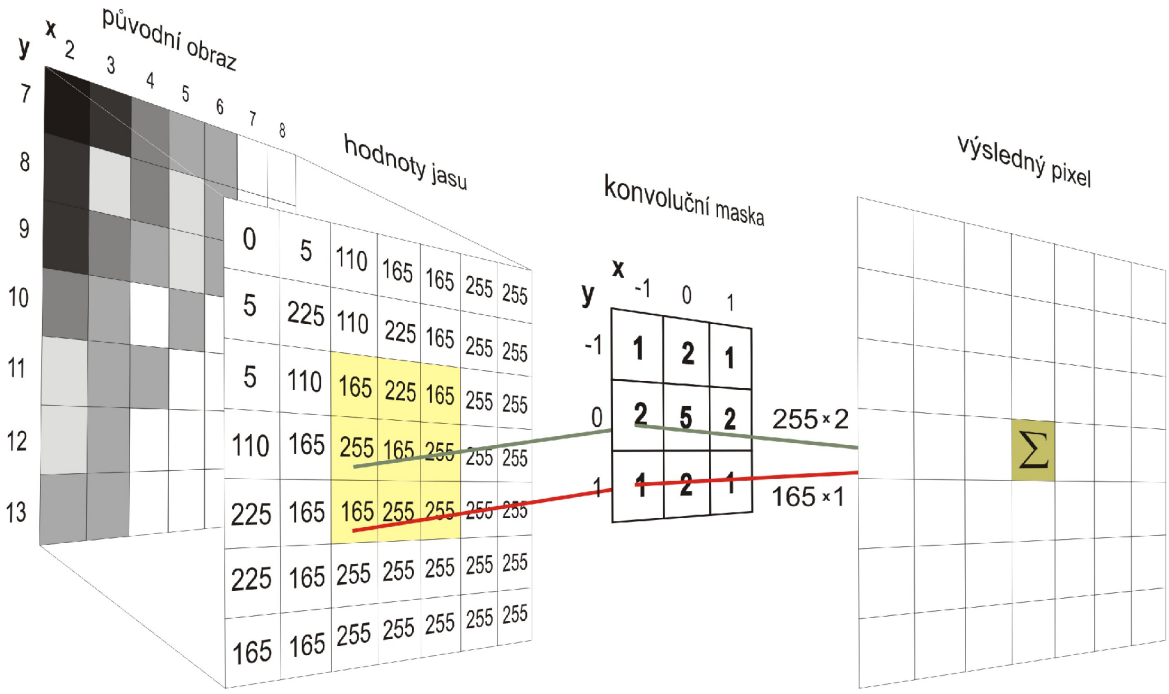
Diskrétní 2D konvoluce je jednoduchá operace [5], která nahrazuje hodnotu pixelu lineární kombinací okolních pixelů. K tomu využívá konvoluční jádro (masku), což je matice obvykle s lichým počtem prvků.

Tato maska se postupně přikládá ke každému pixelu v obrázku. Obvykle se k aktuálně vypočítávanému pixelu přikládá střed masky. K výpočtu nové hodnoty se využijí hodnoty všech pixelů, které maska aktuálně pokrývá, násobené příslušnými koeficienty z masky. Nová hodnota pixelu je pak součtem těchto hodnot. Vzorec pro výpočet nové hodnoty pixelu je následující:

$$O(x,y) = F \cdot I(x,y) = \sum_{j=-N}^{N} \sum_{i=-M}^{M} F(i,j) \cdot I(x-i, y-j) \tag{3.2}$$

kde I je vstupní obraz, O je výstupní obraz a F je konvoluční maska. Rozměry konvoluční masky jsou pak $(2N + 1) \times (2M + 1)$.

Jak vyplývá z obrázku 3.1, nové hodnoty pixelů se nezapisují do původního obrázku tudíž nemají vliv na okolní, ještě nevypočítané pixely.



Obrázek 3.1: Princip 2D diskrétní konvoluce. Zdroj: [9].

3.2 Předzpracování

K dosažení co nejlepšího výsledku klasifikace, je potřeba vstupní obrazová data před zahájením klasifikace vhodně upravit [4]. Tato fáze se provádí z důvodu odstranění chyb a deformací, které mohly v obraze vzniknout při jeho tvorbě, a také z dovůdu rychlejší a efektivnější následné klasifikace.

Škála možných úprav je velká a záleží nejen na stavu vstupních obrazových dat, ale také na požadavcích následujících fází. Mezi možné úpravy může patřit změna barevného prostoru, jasu, kontrastu, velikosti nebo rozlišení.

3.3 Extrakce příznaků

Většinou není vhodné využít ke klasifikaci nezpracovaná vstupní obrazová data, tato data obsahují mnoho redundantních informací, které navíc nejsou příliš obecné. Extrakce příznaků je proces výběru relevantních informací – *příznaků* (feature) ze vstupních dat. Sada příznaků popisující nějaký objekt, se pak nazývá *vektor příznaků* (feature vector) [1].

V našem případě využijeme k zvýraznění příznaků v obraze obrazové filtry, zmíněné v sekci 3.1.

3.4 AdaBoost

Adaptive boosting neboli AdaBoost představili v roce 1995 Freund a Schaphire [7]. Jedná se o algoritmus strojového učení, který slouží k trénování slabých binárních klasifikátorů a jejich spojování za účelem vytvoření silného binárního klasifikátoru. Adaboost k vytvoření silného klasifikátoru využívá trénovací množinu již klasifikovaných dat, kde každému je přidělena váha reflektující úspěšnost jeho klasifikace během trénování. Algoritmus iterativně vybírá aktuálně nejvhodnější slabý klasifikátor a na základě jeho výsledků mění váhy trénovacích dat. AdaBoost je adaptivní v tom smyslu, že při výběru následujícího slabého klasifikátoru, reaguje s využitím vah trénovacích dat na chyby v klasifikaci dříve vybraných slabých klasifikátorů a snaží se je odstranit. Výsledný silný klasifikátor je lineární kombinací vybraných slabých klasifikátorů.

Je dáno: $(x_1, y_1), \ldots, (x_m, y_m)$ kde $x_i \in X, y_i \in Y = \{-1, +1\}$
Inicializuj $D_1(i) = \frac{1}{m}$.
Pro $t = 1, \ldots, T$:

- Trénuj slabého žáka s použitím distribuce D_t.
- Získej slabou hypotézu $h_t : X \to \{-1, +1\}$ s chybou

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right] = \sum_{i : h_t(x_i) \neq y_i} D_t(i).$$

- Zvol $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Aktualizuj:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{+\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_t h_t(x_i))}{Z_t} \end{aligned}$$

  kde Z_t je normalizační faktor (zvolený tak aby D_{t+1} byla distribuce).

Výstupní finální hypotéza:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Obrázek 3.2: Algoritmus AdaBoost. Zdroj: [8].

Pseudokód algoritmu AdaBoost je zobrazen na obrázku 3.2. Vstupem algoritmu je sekvence $(x_1, y_1), \ldots, (x_m, y_m)$ kde $x_i \in X, y_i \in Y = \{-1, +1\}$. Data z trénovací množiny X mají značku $+1$ v případě, že patří do třídy, pro kterou se vytváří klasifikátor, v opačném případě mají značku -1. Druhým vstupem je algoritmus slabého klasifikátoru. Na začátku algoritmu se inicializují váhy všech trénovacích dat na stejnou hodnotu $D_1(i) = \frac{1}{m}$.

Pro $t = 1, \ldots, T$ následuje trénování slabého klasifikátoru s využitím aktuálních vah trénovacích dat $D_t(i)$ a výběr $h_t : X \to \{-1, +1\}$ s aktuální nejnižší chybou klasifikace

ϵ_t na trénovacích datech. Chyba klasifikace ϵ_t se vypočítá jako suma vah všech špatně klasifikovaných trénovacích dat.

$$\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{3.3}$$

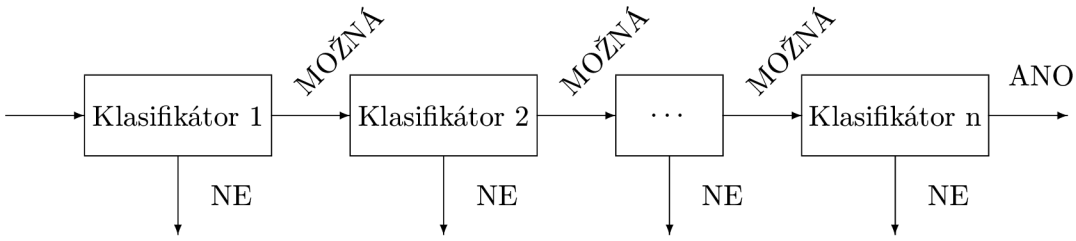
Pokud je $\epsilon_t \geqslant 0.5$ algoritmus končí, protože nalezený nejlepší slabý klasifikátor je horší než náhodný odhad a algoritmus by nebyl konvergentní. V opačném případě se pro vybraný slabý klasifikátor vypočítá koeficient $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$. Ten určuje jakou váhu bude mít ve výsledném silném klasifikátoru. Posledním krokem je aktualizace vah trénovacích dat

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_t h_t(x_i))}{Z_t} \tag{3.4}$$

kde se váha dobře klasifikovaným datům sníží a naopak špatně klasifikovaným datům se zvýší. Poté se algoritmus znovu opakuje od vyběru slabého klasifikátoru.

3.5 Kaskáda klasifikátorů

Kaskáda klasifikátorů, jak ji popisuje Viola a Jones [12], má podobu degenerovaného rozhodovacího stromu. Jak je vidět na obrázku 3.3, kaskáda se skládá z několika *stupňů*, kde se každý stupeň rozhoduje, jestli objekt, co dostal na vstupu pošle na vstup dalšímu stupni nebo jej zahodí. Jednotlivé stupně kaskády jsou klasifikátory, vytrénované AdaBoostem tak, aby zahazovaly ty objekty, o kterých si jsou jisti, že nejsou součástí třídy. Ostatní pošlou dalšímu stupni. Objetky, které projdou všemi stupni, jsou pak označeny jako součást třídy T.



Obrázek 3.3: Kaskáda klasifikátorů. Zdroj: vlastní práce autora.

Jednotlivé stupně jsou konstruovány tak, že počáteční stupně jsou jednoduché a výpočetně nenáročné klasifikátory, které odstraní většinu objektů, které nejsou součástí třídy T. Další stupně jsou pak složitější. Myšlenkou kaskády je, že jednodužší klasifikátory odmítnou mnoho objektů dřív, než se dostanou k výpočetně složitějším klasifikátorům, a to zvýší rychlost celé klasifikace.

3.6 Hodnocení spolehlivosti systému

U biometrických systémů je logicky kladen zvlášť velký důraz na jejich spolehlivost, kterou je potřeba nějak změřit. Dva základní ukazatele spolehlivosti systému jsou známy pod zkratkami *FAR* a *FFR* [6].

False Acceptance rate (FAR)

False Acceptance rate neboli *míra chybného přijetí* značí pravděpodobnost, s jakou biometrický systém klasifikuje jako součást třídy T objekt, který její součástí není. Vzorec pro výpočet FAR vypadá takto:

$$FAR = \frac{N_{FA}}{N_{IIA}} \tag{3.5}$$

kde N_{FA} je počet objektů chybně přiřazených do třídy T a N_{IIA} je celkový počet objektů nenáležících do třídy T.

False Rejection rate (FRR)

False Rejection rate neboli *míra chybného odmítnutí* značí pravděpodobnost, s jakou biometrický systém klasifikuje objekt náležící do třídy T, jako objekt, který není součástí třídy T. Vzorec pro výpočet FRR vypadá takto:

$$FRR = \frac{N_{FR}}{N_{EIA}} \tag{3.6}$$

kde N_{FR} je počet objektů chybně nezařazených do třídy T a N_{EIA} je celkový počet objektů patřících do třídy T.

Negative predictive value (NPV)

Negative predictive value je *míra správné negativní klasifikace* a je definována vzorcem [2]:

$$NPV = \frac{N_{TN}}{N_{TN} + N_{FN}} \tag{3.7}$$

kde N_{TN} je počet správně negativně klasifikovaných objektů a N_{FN} je počet chybně negativně klasifikovaných objektů.

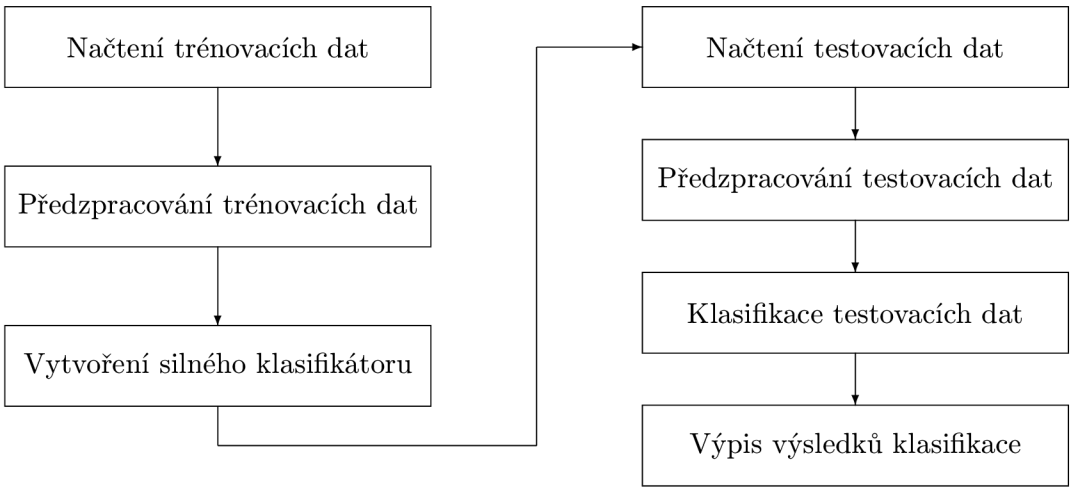
Kapitola 4

Návrh

V této kapitole je uveden návrh aplikace na klasifikaci otisků prstů podle typu snímače, ze kterého byly pořízeny. Aplikace se skládá ze čtyř hlavních částí, kde každá zpracovává spolu související operace. První část je řídící a má za úkol propojit ve správném pořadí operace z ostatních částí. Druhá část se stará o načtení vstupních dat, jejich předzpracování a výpis výsledků klasifikace. Třetí část zajišťuje extrakci příznaků ze vstupních obrazových dat a vytvoření slabých klasifikátorů. Poslední část na základě dat z předchozích dvou částí vytvoří silný klasifikátor a pomocí něho, pak klasifikuje vstupní data.

4.1 Posloupnost operací aplikace

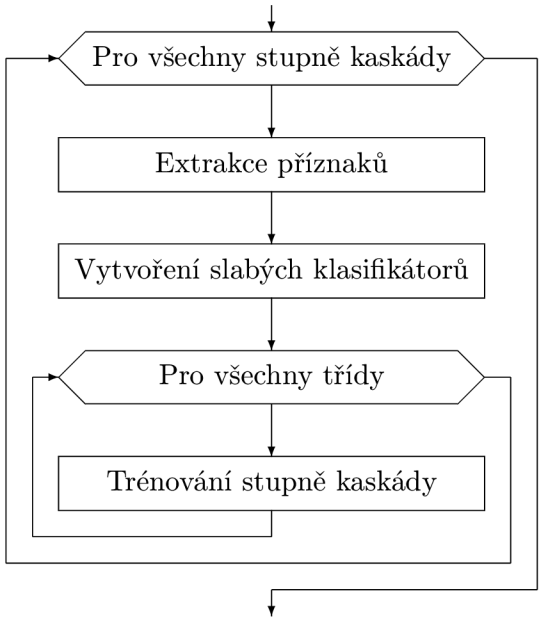
Na obrázku 4.1 je zobrazena posloupnost operací, které navrhovaná aplikace provádí. V první části aplikace se pracuje s trénovacími daty. Tato data se nejprve načtou, poté se předzpracují a následně se využijí k vytvoření silného klasifikátoru. V druhé části se načtou testovacích data, která se také předzpracují a poté se klasifikují klasifikátorem vytvořeným v předchozí části. Nakonec se výsledky klasifikace vypíší.



Obrázek 4.1: Posloupnost operací v navrhované aplikaci. Zdroj: vlastní práce autora.

Proces vytvoření silného klasifikátoru je podrobněji zobrazen na obrázku 4.2. Pro každou třídu trénovacích dat se vytváří jeden binární klasifikátor, který obsahuje kaskádu klasifi-

kátorů vytrénovaných na trénovacích datech. Před trénováním jednotlivých klasifikátorů (stupňů kaskády) se nejprve vždy vyextrahují příznaky z trénovacích dat potřebné pro konkrétní stupeň. Poté se vytvoří množina slabých klasifikátorů pracující s vyextrahovanými příznaky. Pak teprve následuje trénování samotného stupně pro každou třídu zvlášť.



Obrázek 4.2: Posloupnost operací při tvorbě silného klasifikátoru. Zdroj: vlastní práce autora.

Extrakce příznaků u testovacích dat se provádí v rámci klasifikace, u těch stupňů kaskády, kde to je potřeba.

4.2 Zpracování vstupních dat a tvorba výstupu

Aplikace má v aktuálním adresáři k dispozici dva adresáře, ze kterých načítá vstupní data. Z adresáře `./train` načítá trénovací data a z adresáře `./test` testovací data. Z těchto adresářů se pokusí načíst všechny soubory, včetně těch v podadresářích. Pokud se je povede načíst, jsou zařazeny do odpovídající množiny.

Kromě omezení na formát souboru, je také u trénovacích dat nutné dodržet správný formát názvu souboru. Aplikace potřebuje k vytvoření klasifikátoru předem správně klasifikovaná data. Třída do které trénovací obrázek patří, je součástí názvu souboru. V názvu souboru začíná název třídy za prvním znakem `'_'` (podtržení) a končí před druhým znakem `'_'`. Kromě obsahu souboru si tedy aplikace pro další využití ukládá u trénovacích dat název třídy a u testovacích dat název souboru, použitý pro výpis výsledků klasifikace.

V rámci předzpracování se načtené obrázky převádí do osmibitové barevné hloubky a do šedotónového obrazu pomocí vzorce:

$$I = 0.229 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{4.1}$$

kde R, G a B jsou hodnoty barevných složek pixelu a I je výsledná intenzita šedé barvy pixelu. Tato operace se provádí pro všechny pixely v obraze. Jednotlivé barevné složky mají

jinou váhu z důvodu odlišné citlivosti lidského oka na červenou, zelenou a modrou barvu. Jiné operace v rámci předzpracování aplikace neprovádí. Například některé chyby v obraze mohou být pro některou třídu vstupních dat specifické, a jejich opravením bychom se zbavili možnosti, vstupní data na základě této chyby klasifikovat.

Výstup aplikace je v podobě výpisu výsledků klasifikace testovacích dat na standardní výstup. Každý obrázek je vypisován na samostatný řádek, který je tvořen dvojicemi oddělenými tabulátorem, kde v prvním sloupci je název třídy přiřazené klasifikátorem a v druhém sploupci název souboru.

4.3 Extrakce příznaků a tvorba slabých klasifikátorů

Pro každý stupeň kaskády je aplikace vybavena odlišným obrazovým filtrem, který má zvýrazňovat příznaky z obrázků otisků potřebné pro vytrénování stupně. Konkrétní filtry pro jednotlivé stupně, jejich nastavení a příznaky, které zvýrazňují jsou otázkou testování a jsou popsány v kapitole 6.

Pro každý stupeň kaskády se také vytváří množina slabých klasifikátorů, ze kterých si následně vybírá učící algoritmus. Jednotlivé slabé klasifikátory mají podobu jednoduché funkce pracující s příznaky pro stupeň, který mají klasifikovat. Klasifikují například na základě toho, jestli hodnota příznaku je větší nebo menší než hodnota prahu, který má klasifikátor nastaven. Množina slabých klasifikátorů se pak vytváří, nastavením různých hodnot prahu pro jednotlivé slabé klasifikátory.

4.4 Tvorba silného klasifikátoru a klasifikace

Pro každou třídu trénovacích dat, kterou aplikace zjistila z názvů trénovacích dat se vytváří jeden binární klasifikátor, který řekne, jestli obrázek do třídy patří, nebo nepatří. Každý tento klasifikátor je tvořen kaskádou klasifikátorů vytrénovaných učícím algoritmem. Proces tvorby kaskády klasifikátorů byl již ukázán na obrázku 4.2.

Učícím algoritmem, který používá aplikace, je algoritmus AdaBoost, popsaný v sekci 3.4. Vstupem algoritmu je tedy množina trénovacích dat a množina slabých klasifikátorů. Algoritmus provádí nastavený počet iterací, při nichž vybírá z množiny slabých klasifikátorů vhodné slabé klasifikátory, a ty si spolu s jejich váhou ukládá. Na konci algoritmu se ještě vypočítává práh klasifikátoru. Ten je ve výchozím nastavení roven nule, nicméně z důvodu, že takto vytrénovaný klasifikátor není jediný, ale jen jeden z kaskády, tak je vhodné, aby klasifikátorem prošly pokud možno všechny obrázky patřící do třídy, pro kterou byl klasifikátor vytvořen i za cenu toho, že projde také velká část obrázků, které do třídy nepatří. Nastavení prahu se provádí, tak že se vytvořeným klasifikátorem klasifikují data z trénovací množiny a práh se nastaví na nejnižší hodnotu, kterou získá obrázek z třídy, kterou klasifikátor klasifikuje. Na nule práh zůstane pouze v případě, že nejnižší získaná hodnota je větší než 0.

Klasifikace testovacích dat se provádí obrázek po obrázku. Vytvořený klasifikátor se skládá z binárního klasifikátoru pro každou třídu, kde každý obsahuje kaskádu klasifikátorů. Obrázek prochází postupně klasifikátory všech tříd a na konci se určí, do které třídy patří. V jednotlivých binárních klasifikátorech prochází obrázek stupeň po stupni. Na začátku stupně se provede extrakce příznaků pro konkrétní stupeň. Poté se obrázek nechá oklasifikovat všemi ováženými slabými klasifikátory stupně, výsledky se sečtou a pokud výsledná hodnota je vyšší než práh, tak se proces opakuje pro další stupeň. V případě, že je

výsledná hodnota menší než práh, tak klasifikace končí a pokračuje se na klasifikátor další třídy. Až obrázek projde klasifikátory všech tříd, tak se porovnají součty výsledných hodnot všech stupňů u těch klasifikátorů, kde se podařilo dojít na konec kaskády a třída toho, u nejž je nejvyšší hodnota, je přiřazena obrázku. V případě, že u žádného klasifikátoru neprojde obrázek až na konec kaskády, je mu přidělena třída s názvem v podobě znaku `'_'`(uživatel takto třídu pojmenovat nemůže), která značí, že obrázek nepatří do žádné z definovaných tříd.

Kapitola 5

Implementace

V této kapitole je popsána implementace aplikace navržené v předchozí kapitole. Aplikace byla implementována v jazyce C++ s využitím externí knihovny OpenCV. V rámci implementace byla vytvořena řešení pro operační systémy GNU/Linux a MS Windows.

5.1 Knihovna OpenCV

Knihovna OpenCV (Open Source Computer Vision) je volně šiřitelná knihovna pro počítačové vidění se zaměřením na real-time zpracování obrazu. Knihovna je uvolněna pod BSD licencí a je multipatformní.

Při implementaci navržené aplikace byly využity tři moduly z této knihovny. Modul `core`, který definuje datové struktury a některé operace s nimi, modul `imgproc` pro zpracování obrazu a modul `highgui`, který zajišťuje uživatelské rozhraní spolu s čtením a zápisem obrazu a videa.

Informace o knihovně OpenCV a o funkcích, které nabízí, byly čerpány z online OpenCV dokumentace viz [11].

5.2 Moduly

Implementace aplikace byla dle návrhu rozdělena do čtyř modulů.

Modul `main`

Tento modul obsahuje jedinou funkci `main()`, která postupně volá funkce z ostatních modulů v pořadí, jaké je popsané v návrhu. Jsou zde také definovány proměnné pro uložení vstupních dat i interních dat, které aplikace vytváří. Patří mezi ně hlavně:

- `list<Timage> trainImgList` – pro uložení vstupních trénovacích dat,
- `list<Timage> testImgList` – pro uložení vstupních testovacích dat,
- `list<TweakClassifier> weakClassifierList` – pro uložení množiny slabých klasifikátorů,
- `list<TstrongClassifier> strongClassifierList` – pro uložení výsledných silných klasifikátorů jednotlivých tříd.

Tyto proměnné jsou ostatním funkcím předávány pomocí ukazatelů.

Modul `inout`

Modul **inout** se stará zejména o načtení vstupních dat a výpis výstupních dat. V prvé řadě je v tomto modulu definován datový typ **Timage** sloužící k uložení vstupních dat:

Algoritmus 5.1: Definice typu `Timage`

```
1  typedef struct{
2      Mat img;
3      unsigned classN;
4      string fileName;
5      int features_1[2][10];
6      int features_2;
7  }Timage;
```

Datový typ **Timage** je struktura skládající se z pěti položek. První položka je objekt třídy **Mat**, což je OpenCV třída reprezentující vícerozměrná pole s jedním nebo více kanály, sloužící k uložení obsahu obrázku. Položka **classN** slouží k uložení číselné hodnoty odpovídající třídě, do které obrázek patří. Položka **fileName** je určena k uložení jména souboru spolu s cestou z aktuálního adresáře. Poslední dvě položky slouží k uložení příznaků pro jednotlivé stupně kaskády klasifikátorů.

Načítání vstupních dat je realizováno dvěma funkcemi. Funkce **loadTrainData()** slouží k načtení trénovacích dat, při čemž zároveň extrahuje z názvu souboru třídu, do které obrázek patří, a přidělí jí číselný kód, který si uloží. Název souboru se pak u trénovacích dat neukládá. Naopak funkce **loadTestData()**, sloužící k načtení testovacích dat, si název souboru ukládá, ale neextrahuje z něj jméno třídy. U obou funkcí se na začátku volá rekurzivní funkce **getFileNames()**, která vrací seznam jmen všech souborů z požadované složky k načtení.

Funkce **getFileNames()** je v implementaci pro MS Windows odlišná od té v implementaci pro GNU/Linux. V implementaci pro MS Windows využívá funkce **FindFirstFile()** a **FindNextFile()** z **windows.h** vracející informace o nalezeném souboru ve struktuře typu **WIN32_FIND_DATA**. Pokud nalezený soubor je adresář, tak se funkce zavolá rekurzivně znovu a prohledává soubory v novém adresáři. Pokud nalezený soubor není adresář, tak si funkce uloží jeho název spolu s cestou z aktuálního adresáře a seznam takto nalezených souboru pak vrací. V implementaci pro GNU/Linux funguje funkce podobně, ale ke zjištění jmen souborů v adresáři využívá funkce **opendir()** a **readdir()** z **dirent.h**. Jestli se jedná o adresář je pak zjištěno pomocí funkcí ze **sys/stat.h**.

Ve funkcích **loadTrainData()** a **loadTestData()** je poté obrázek načítaný pomocí OpenCV funkce **imread()**, která podporuje řadu formátů, mimo jiné také hojně používané formáty **bmp**, **png** a **jpg**. Před uložením obrázku a potřebných informací do struktury **Timage** se ještě obrázek předzpracuje pomocí funkce **preprocessing()**, která volá OpenCV funkci **cvtColor()** zajišťující převedení obrázku do 8-bit formátu a do šedotónového obrazu.

Poslední funkce tohoto modulu **printResults()**, pak zajišťuje výpis výsledků klasifikace na standardní výstup, ve formátu zmiňovaném v návrhu.

Modul `weak`

Tento modul zajišťuje extrakci příznaků a tvorbu slabých klasifikátorů pro jednotlivé stupně kaskády klasifikátorů. O extrakci příznaků se starají dvě funkce. Pro první stupeň je to funkce **filter_1()**, realizující prahovací filtr. Pro druhý stupeň je to funkce **filter_1()**, která k extrakci využívá 2D diskrétní konvoluci. U této funkce bylo k provedení konvoluce

využito OpenCV funkce `filter2D()`, která na vstupní obrázek aplikuje zvolený lineární filtr.

Funkce `createWeakClassifierList_1()` a `createWeakClassifierList_2()` mají za úkol vytvořit množinu slabých klasifikátorů pro daný stupeň. Na začátku těchto funkcí se nejprve extrahují ze vstupních obrázků příznaky. Poté se s využitím funkcí slabých klasifikátorů, které jsou v tomto modulu také definovány, vytvoří různým nastavením vstupních parametrů seznam slabých klasifikátorů. Jednotlivé slabé klasifikátory v tomto seznamu jsou uloženy v následující struktuře:

Algoritmus 5.2: Definice typu `TweakClassifier`

```
1  typedef struct{
2      int (*weakClassifier)(double*, Timage*);
3      double par[5];
4  }TweakClassifier;
```

První položka v této struktuře je ukazatel na funkci slabého klasifikátoru. Druhá položka je zároveň prvním parametrem funkce slabého klasifikátoru uložené v první položce a jsou v ní uloženy parametry slabého klasifikátoru nastavéné ve funkci tvořící množinu slabých klasifikátorů.

Modul `strong`

Modul `strong` má na starosti vytvoření silného klasifikátoru reprezentující jeden stupeň kaskády klasifikátorů, což je obstaráno funkcí `adaBoost()`. Na konci této funkce je volána funkce `setStageThreshold()`, starající se o nastavení vhodné hodnoty prahu klasifikátoru. Funkce `classification()` má za úkol klasifikovat testovací data. Podrobný popis, jak tyto funkce pracují, je uveden v návrhu v sekci 4.4.

V tomto modulu jsou také definovány tři nové datové typy reprezentující jednotlivé části klasifikátoru.

Algoritmus 5.3: Definice typu `TstrongClassifier`

```
1  typedef struct{
2      list<TstageCascade> cascadeClassifier;
3      unsigned classN;
4  }TstrongClassifier;
```

Klasifikátor pro každou třídu je uložen ve struktuře typu `TstrongClassifier`, která obsahuje kaskádu klasifikátorů, tedy seznam jednotlivých stupňů kaskády a také číselný kód třídy, pro kterou byl klasifikátor vytvořen.

Algoritmus 5.4: Definice typu `TstageCascade`

```
1  typedef struct{
2      list<TnodeStageCascade> weakClassifierList;
3      double threshold;
4  }TstageCascade;
```

Každý stupeň kaskády je uložen ve struktuře typu `TstageCascade`, která obsahuje seznam slabých klasifikátorů, ze kterých se stupeň skládá, spolu s hodnotou prahu stupně.

Algoritmus 5.5: Definice typu `TnodeStageCascade`

```
1  typedef struct{
2      TweakClassifier weakClassifier;
3      double alpha;
4  }TnodeStageCascade;
```

Stupeň kaskády klasifikátorů se skládá z lineární kombinace slabých klasifikátorů, ty jsou uloženy ve struktuře typu **`TnodeStageCascade`** spolu s jejich vahou **`alpha`**.

Kapitola 6

Testování

V této kapitole je popsáno, jaké filtry využívají jednotlivé stupně kaskády a jaké je nastavení těchto filtrů. Také je popsáno, jaké příznaky pomocí těchto filtrů extrahujeme a jaké slabé klasifikátory s těmito příznaky pracují. V neposlední řadě také to, jakou účinnost takto navržené stupně mají. Nejprve ale dojde k představení dat použitých k trénování a testování klasifikátoru.

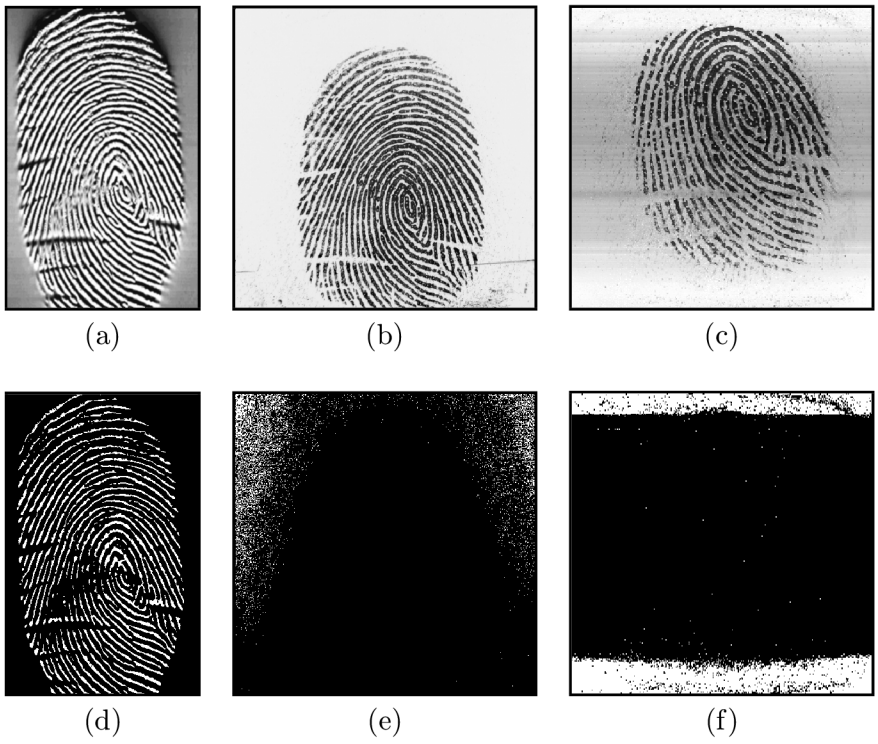
Jako vstupní data aplikace byly využity otisky prstů z databáze EBD_FP. Tato databáze obsahuje celkem 9448 otisků prstů patřících 110 osobám. Otisky prstů v databázi jsou nasnímány třemi různými senzory–termálním, optickým a kapacitním. Každá osoba má v databázi otisky všech deseti prstů hned několikrát pro každý senzor.

Cílem aplikace je co nejlépe klasifikovat otisky do jedné ze tří tříd podle typu senzoru, ze kterého byly pořízeny. Za tímto účelem byla databáze otisků prstů rozdělena na dvě části. První částí je množina trénovacích dat. Do této množiny byly vybrány otisky prvních pěti osob s číselnými kódy 1100–1104. Těchto otisků je celkem 438, přičemž 138 je z termálního senzoru, 150 z optického senzoru a 150 z kapacitního senzoru. Zbylých 9010 otisků bylo zařazeno do množiny testovacích dat.

6.1 Stupeň 1 - Prahování

Pro první stupeň kaskády byl zvolen prahový filtr. Je to jednoduchý filtr, který potřebuje pouze jeden průchod obrázkem a k výpočtu hodnoty každého pixelu je potřeba pouze jeho vlastní hodnota. Aby stupeň pracoval co nejlépe, bylo potřeba zvolit vhodnou úroveň prahu, díky níž by po aplikaci filtru byly obrázky otisků z jednotlivých tříd dostatečně odlišné. Po několika pokusech byla nakonec jako úroveň prahu vybrána hodnota 239, při které se pixely s hodnotami jasu 240–255 změní na bílé a pixely s hodnotami menší než 240 budou černé.

V ideálním případě, jsou po prahování otisky jednotlivých tříd rozdílné podobně, jako otisky na obrázku 6.1. Obrázky jednotlivých tříd mají v různých částech obrazu plochy s většinovým zastoupením bílých, nebo naopak černých pixelů. Toho bylo využito při extrakci příznaků. Z každého obrázku se vytvoří dva vektory příznaků, pro dva různé pohledy na obrázek. Pro první vektor se obrázek zpracovává po řádcích a pro druhý po sloupcích. Aby nebylo příznaků mnoho, rozdělí se obraz horizontálně a vertikálně na 10 částí, obsahující stejný počet řádků nebo sloupců. Pro řádky i sloupce se vypočítává poměr bílých a černých pixelů v části. Pokud je v části méně než 2 % bílých pixelů, je hodnota příznaku 0, v případě, že je v části 2–50 % bílých pixelů, je hodnota příznaku 1. Nad 50 % bílých pixelů v části je hodnota příznaku 2.



(a) (b) (c)

(d) (e) (f)

Obrázek 6.1: Otisky prstů před prahováním (a, b, c) a otisky prstů po prahování (d, e, f). Otisky z termálního senzoru (a, d), optického senzoru (b, e) a kapacitního senzoru (c, f).

Příznaky extrahované z otisků zobrazených výše jsou znázorněny na obrázku 6.2. V prvním řádku je vždy řádkový vektor příznaků a v druhém je sloupcový vektor příznaků. Řádky se v obrázku zpracovávají shora dolů a sloupce zleva doprava. Jak je tedy vidět, tak například u otisku z optického senzoru je v prvních 60 % řádků 2–50 % bílých pixelů, v dalších 40 % je jich již méně než 2 %. Co se týče sloupců, tak tam je v prvních a posledních 20 % také 2–50 % bílých pixelů, a ve zbylých 60 % uprostřed jich je méně než 2 %.

1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 0 0 0 0	2 0 0 0 0 0 0 0 1 2
1 1 1 1 1 1 1 1 1 0	1 1 0 0 0 0 0 0 1 1	1 1 1 1 1 1 1 1 1 1
(a)	(b)	(c)

Obrázek 6.2: Příznaky extrahované z otisku z termálního senzoru (a), optického senzoru (b) a kapacitního senzoru (c), které jsou zobrazeny na obrázku 6.1.

S takto získanými příznaky poté pracuje slabý klasifikátor navržený pro tento stupeň. Slabý klasifikátor dostane na vstupu, jestli má pracovat s řádkovým nebo se sloupcovým příznakovým vektorem. Dále hranice intervalu a jednu ze tří hodnot, které se mohou vyskytovat v příznakových vektorech. Na základě těchto vstupů, klasifikuje slabý klasifikátor obraz, jako součást třídy, pokud se vstupní hodnota nachází na celém intervalu v požadovaném vektoru příznaků. V opačném případě neklasifikuje obraz jako součást třídy. Různou kombinací hranic intervalů, zpracovávaného vektoru příznaků a hodnoty příznaku se vytvoří množina slabých klasifikátorů. Těch je celkem pro tento stupeň 330.

Poslední co je potřeba udělat, je zvolit takový počet iterací AdaBoostu, aby měl stupeň co nejlepší výsledky. Požadavek je, aby stupeň nevyhazoval skoro žádné obrázky otisků, které do třídy patří, ale zároveň vyhodil větší množství těch co do třídy nepatří. Hledá se tedy co nejvyšší NPV. V následující tabulce jsou znázorněny výsledky stupně pro různé počty iterací:

Tabulka 6.1: NPV pro různý počet iterací u prvního stupně

Počet iterací	2	3	4	5	10	20	50
Termální senzor	N/A	99,31 %	99,27 %	98,79 %	98,64 %	98,61 %	98,05 %
Optický senzor	N/A	100,0 %	99,83 %	99,78 %	99,83 %	99,82 %	99,83 %
Kapacitní senzor	99,73 %	99,73 %	99,73 %	99,73 %	99,73 %	99,60 %	99,60 %
Průměr	N/A	99,68 %	99,61 %	99,43 %	99,40 %	99,34 %	99,16 %

Hodnota N/A značí, že nebylo možné NPV vypočítat, jelikož stupněm prošly všechny obrázky. Na základě těchto výsledků, byl pro tento stupeň nastaven počet iterací AdaBoostu na 3, při kterém jsou výsledky tohoto stupně následující:

Tabulka 6.2: Úspěšnost prvního stupně. Počet otisků správně pozitivně klasifikovaných (TP), špatně pozitivně klasifikovaných (FP), správně negativně klasifikovaných (TN) a špatně negativně klasifikovaných (FN).

	TP	FP	TN	FN	Pokračuje do dalšího stupně
Termální senzor	2677	1571	4729	33	47,15 %
Optický senzor	3150	2837	3023	0	66,45 %
Kapacitní senzor	3136	642	5218	14	41,93 %

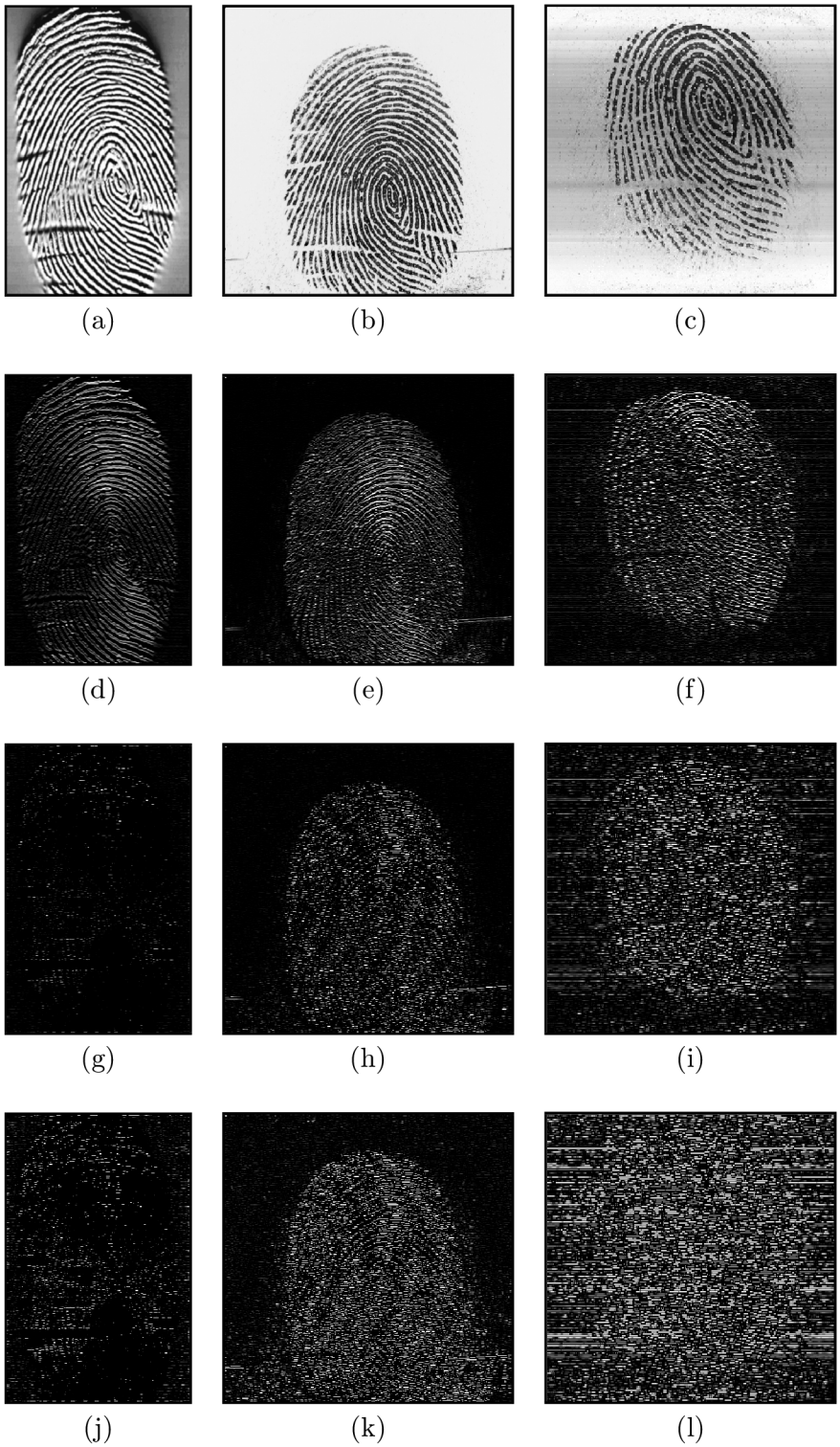
6.2 Stupeň 2 - Diskrétni 2D konvoluce

Pro tento stupeň bylo zvoleno filtrování na základě 2D diskrétní konvoluce. Cílem bylo využít k odlišení jednotlivých tříd horizontálního šumu na pozadí u snímků otisků prstů z kapacitního snímače. Za tímto účelem byly zvoleny celkem tři konvoluční masky (jádra) zobrazené na obrázku 6.3. Maska (a) má za úkol detekci horních i spodních horizontálních hran. Úkolem masky (b) je potlačit diagonální hrany a nakonec maska (c), má zvýraznit zbylé nečerné pixely pro větší odlišení jednotlivých tříd.

-1	-1	-1		-2	0	-2		0	0	0
2	2	2		1	1	1		0	4	0
-1	-1	-1		-2	0	-2		0	0	0
	(a)				(b)				(c)	

Obrázek 6.3: Konvoluční masky (jádra). Zdroj: vlastní práce autora.

Na obrázku 6.4 je ukázáno, jaký vliv má postupná aplikace konvolučních masek na otisky z jednotlivých typů senzorů. Výsledné obrázky se liší v množství světlých pixelů, z toho důvodu byla jako příznak vybrána průměrná hodnota jasu obrázku.



Obrázek 6.4: Otisky prstů před aplikací filtru (a, b, c), po aplikaci první masky (d, e, f), druhé masky (g, h, i) a třetí masky (j, k, l). V prvním sloupci je otisk z termálního senzoru, ve druhém z optického senzoru a ve třetím z kapacitního senzoru.

U zobrazených otisků je hodnota příznaku pro otisk z termálního senzoru 7, pro otisk z optického senzoru 25 a pro otisk z kapacitního senzoru 53.

Slabý klasifikátor pro tento stupeň má jako vstup hranice intervalu. Zpracovávaný otisk označí jako součást třídy v případě, že se hodnota jeho příznaku nachází uvnitř intervalu, který dostal klasifikátor jako vstup. V opačném případě není otisk klasifikován jako součást třídy. Kombinací různých hranic intervalu pro tento stupeň vytvořena množina celkem 1326 slabých klasifikátorů.

Pro co nejlepší výsledek stupně je opět nutné zvolit vhodný počet iterací učícího algoritmu AdaBoost. Stejně jako u předchozího stupně se zvolí takový počet iterací, aby u něj byla nejlepší míra správné negativní klasifikace (NPV).

Tabulka 6.3: NPV pro různý počet iterací u druhého stupně

Počet iterací	1	2	3	5	10	20	50
Termální senzor	96,65 %	96,65 %	96,65 %	96,65 %	96,65 %	96,65 %	96,65 %
Optický senzor	94,41 %	94,41 %	94,41 %	94,41 %	94,41 %	94,41 %	94,41 %
Kapacitní senzor	82,40 %	82,40 %	82,40 %	82,40 %	82,40 %	82,40 %	82,40 %
Průměr	91,16 %	91,16 %	91,16 %	91,16 %	91,16 %	91,16 %	91,16 %

Z tabulky 6.3 je vidět, že NPV je stejné při všech iteracích AdaBoostu. Pro tento stupeň byl nastaven počet iterací na 1, jelikož vyšší počet iterací při stejné NPV by výsledek nezlepšil, ale zvýšil by dobu klasifikace. Při takto nastaveném počtu iterací je úspěšnost tohoto stupně následující:

Tabulka 6.4: Úspěšnost druhého stupně

	TP	FP	TN	FN	Pokračuje do dalšího stupně
Termální senzor	2623	11	1560	54	62,01 %
Optický senzor	2986	69	2768	164	51,03 %
Kapacitní senzor	3035	169	473	101	84,81 %

Kapitola 7

Výsledky

V této kapitole jsou ukázány výsledky jakých navržený klasifikátor dosahuje. Nejprve jsou zde představeny podrobné výsledky, kterých finální klasifikátor dosahuje při stejně zvolených množinách trénovacích a testovacích dat, jako tomu bylo při testování.

Tabulka 7.1: Úspěšnost finálního klasifikátoru

	Termální senzor	Optický senzor	Kapacitní senzor	Neklasifikováno
Termální senzor	2623	33	0	54
Optický senzor	11	2930	169	40
Kapacitní senzor	0	6	3035	109

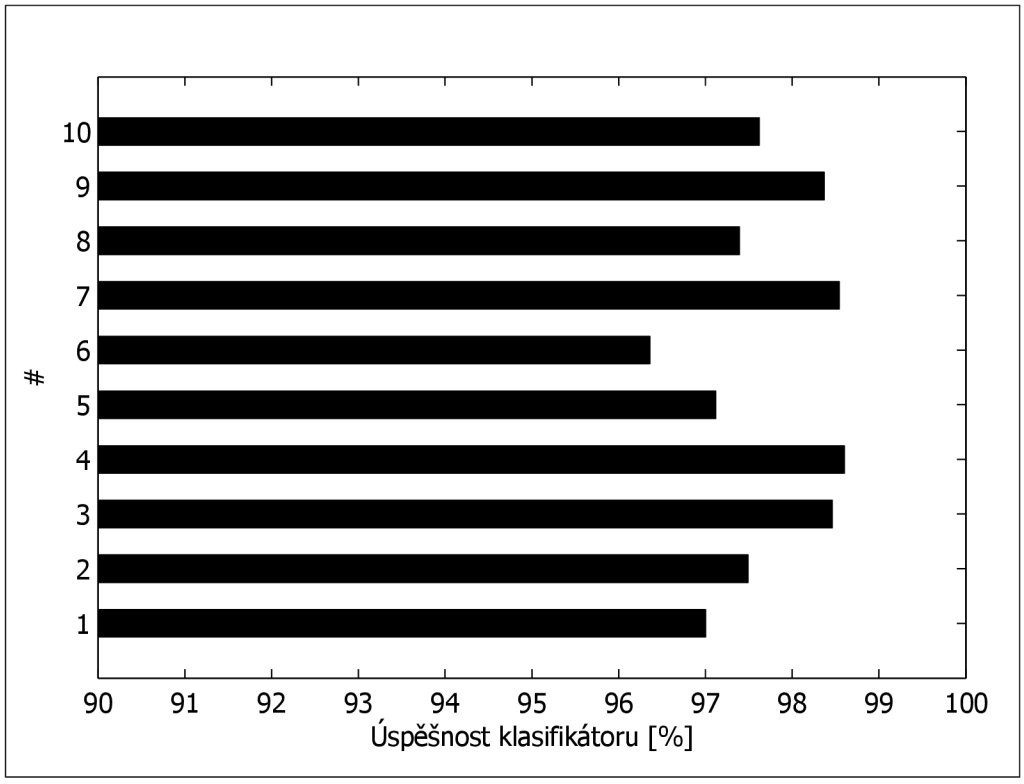
Míra chybného přijetí (FAR) a míra chybného odmítkutí (FRR) u jednotlivých tříd pak vychází následovně:

Tabulka 7.2: Spolehlivost finálního klasifikátoru

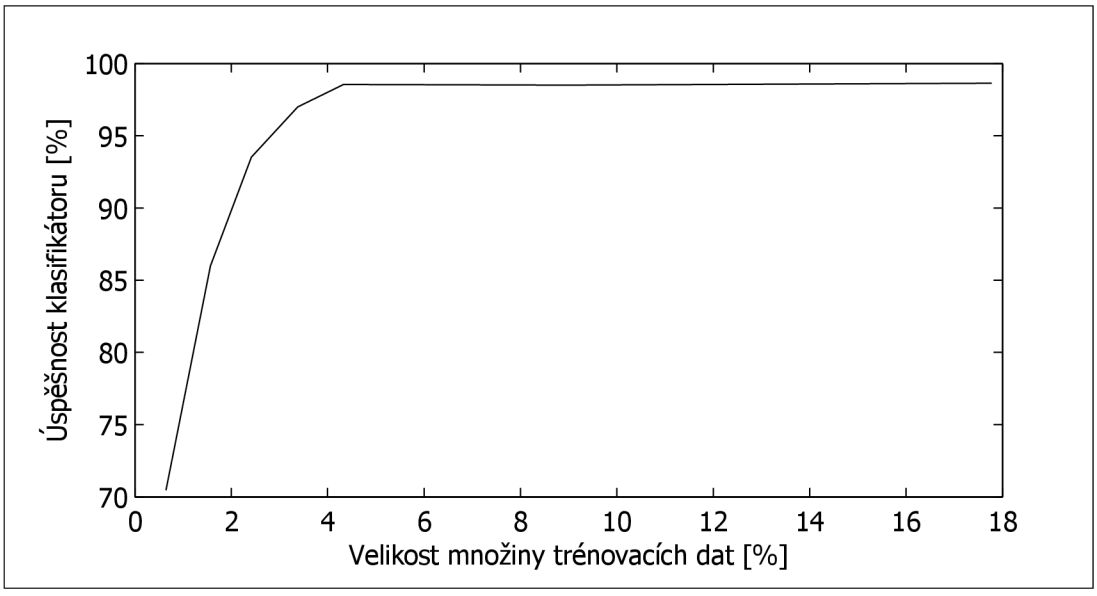
	FAR	FRR
Termální senzor	0,17 %	3,21 %
Optický senzor	0,67 %	6,98 %
Kapacitní senzor	2,88 %	3,65 %

Protože úspěšnost klasifikace závisí také na tom, jaká konkrétní data jsou v množině trénovacích dat, tak byl proveden pokus, při kterém byl klasifikátor spuštěn s různými daty v množině trénovacích dat. Množiny při jednotlivých pokusech obsahovaly stejně jako při testování otisky pěti osob, které byly pro každý pokus náhodně vybrány. Průměrná úspěšnost byla při pokusech 97,76 %, přičemž úspěšnost je definována jako počet strávně klasifikovanách otisků ku celkovému počtu otisků. Výsledky pokusu jsou znázorněny v grafu na obrázku 7.1.

Úspěšnost klasifikace závisí také na velikosti trénovací množiny. Z toho důvodu byl proveden další pokus s různými velikostmi množin trénovacích dat. Výsledky tohoto pokusu jsou znázorněny na obrázku 7.2. Velikost množiny trénovacích dat je definovaná jako poměr počtu otisků v množině trénovacích dat ku celkovému počtu otisků.



Obrázek 7.1: Úspěšnost klasifikátoru při různě zvolených množinách trénovacích dat



Obrázek 7.2: Úspěšnost klasifikátoru při různých velikostech množin trénovacích dat

Kapitola 8

Závěr

V rámci této práce jsem navrhl a implementoval klasifikátor otisků prstů, který klasifikuje otisky prstů na základě typu snímače, ze kterého byly otisky nasnímány.

Výsledný klasifikátor obsahuje binární klasifikátor pro každou třídu klasifikovaných dat. Jednotlivé binární klasifikátory tvoří kaskáda klasifikátorů vytrénovaných učícím algoritmem AdaBoost. Celkem jsem navrhl dva stupně kaskády, kde první stupeň pro extrakci příznaků využívá prahovacího filtru a druhý diskrétní 2D konvoluci.

Navrženou aplikaci jsem implementoval v jazyce C++, s využitím externí knihovny OpenCV, pro operační systémy GNU/Linux a MS Windows. Při testování dosáhla aplikace úspěšnosti klasifikace v průměru 97,76 %. Výsledek je to velmi dobrý, přičemž špatně klasifikované nebo neklasifikované otisky byly většinou, buď chybně nasnímané (v důsledku například příliš silného nebo naopak slabého přitisknutí prstu ke snímači), a nebo obsahovali nějakou obrazovou chybu (například příliš vysoký nebo naopak nízký jas obrázku).

V rámci pokračování práce, by se úspěšnost klasifikátoru dala zvýšit předzpracováním vstupních dat před extrakcí příznaků, při němž by byly odstraněny chyby ve vstupních datech, které mohou mít negativní vliv na hodnoty extrahovaných príznaků. Případně navržením dalších vhodných stupňů kaskády klasifikátorů.

Dalším vhodným rozšířením by byla implementace možnosti uložit si nastavení vytrénovaného klasifikátoru do souboru, aby nebylo nutné při každém spuštění aplikace znovu trénovat klasifikátor, i když chceme klasifikovat data, pro které jsme už klasifikátor vytrénovali dříve.

Literatura

[1] Feature extraction - Wikipedia, The Free Encyclopedia [online]. 2014-03-01 [cit. 2014-03-31].
URL http://en.wikipedia.org/wiki/Feature_extraction

[2] Positive and negative predictive values - Wikipedia, The Free Encyclopedia [online]. 2014-04-11 [cit. 2014-05-09].
URL
http://en.wikipedia.org/wiki/Positive_and_negative_predictive_values

[3] ACHARYA, T.; RAY, A. K.: *Image Processing: Principles and Applications*. Wiley, 2005, ISBN 9780471745785.

[4] BOW, S. T.: *Pattern Recognition and Image Preprocessing*. CRC Press, druhé vydání, 2002, ISBN 9780824706593, 720 s.

[5] David, J.: Correlation and Convolution [PDF dokument]. 2005 [cit. 2014-05-11].
URL http://www.cs.umd.edu/~djacobs/CMSC426/Convolution.pdf

[6] DRAHANSKÝ, M.; ORSÁG, F.; DOLEŽEL, M.: *Biometrie*. Computer Press, s.r.o, první vydání, 2011, ISBN 978-80-254-8979-6, 294 s.

[7] FREUND, Y.; SCHAPIRE, R. E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, ročník 55, č. 1, 1997: s. 119 – 139, ISSN 0022-0000.

[8] FREUND, Y.; SCHAPIRE, R. E.: A Short Introduction to Boosting. *Journal-Japanese Society for Artificial Intelligence*, ročník 14, č. 5, 1999: s. 771 – 780, ISSN 0912-8085.

[9] Grt: Princip výpočtu dvourozměrné diskrétní konvoluce - Wikipedia, The Free Encyclopedia [obrázek]. 2006-07-19 [cit. 2014-05-12].
URL
http://commons.wikimedia.org/wiki/File:Konvoluce_2rozm_diskretni.jpg

[10] MALTONI, D.; MAIO, D.; JAIN, A. K.; aj.: *Handbook of Fingerprint Recognition*. London: Springer, druhé vydání, 2009, ISBN 18-488-2254-5, 494 s.

[11] opencv dev team: OpenCV documentation [online]. 2014-04-21 [cit. 2014-05-15].
URL http://docs.opencv.org/

[12] VIOLA, P.; JONES, M. J.: Robust Real-Time Face Detection. *International Journal of Computer Vision*, ročník 57, č. 2, 2004: s. 137 – 154, ISSN 0920-5691.