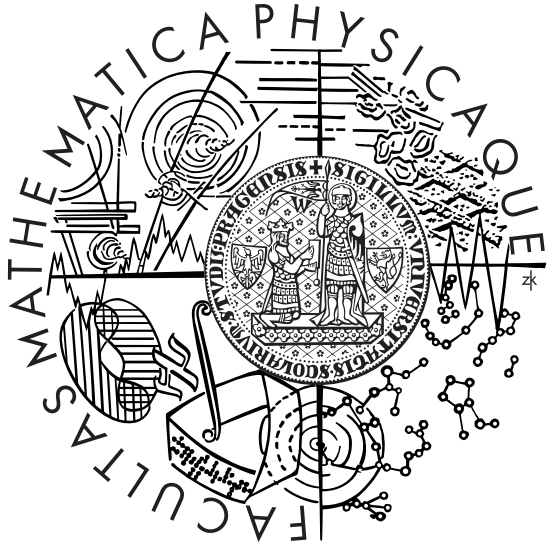
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

Diplomová práce



Tomáš Vala

Rozpoznávání SPZ z jednoho snímku

Katedra softwarového inženýrství

Vedoucí diplomové práce:	RNDr. Jana Štanclová
Studijní program:	Informatika
Studijní obor:	Softwarové systémy

Děkuji RNDr. Janě Štanclové za vedení této diplomové práce, za její cenné poznámky a připomínky k obsahu i zpracování. Dále děkuji firmě Ramet CHM a.s., která mi poskytla vstupní data. V neposlední řadě bych chtěl poděkovat své manželce a celé mé rodině, neboť bez jejich podpory po dobu mého studia by tato práce nemohla vzniknout. Největší díky však patří Bohu.

Prohlašuji, že jsem svou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Uherském Hradišti dne 1. srpna 2006 Tomáš Vala

Obsah

1 Úvod **1**
1.1 Cíle práce . . . 1
1.2 Obsah práce . . . 2

2 Problematika **4**
2.1 Specifikace úlohy . . . 4
2.2 Proces rozpoznávání SPZ . . . 7
2.3 Problematika reálného prostředí . . . 9

3 Rozpoznávání SPZ **12**
3.1 Předzpracování . . . 12
3.2 Lokalizace oblasti SPZ . . . 14
3.2.1 Hrubá lokalizace . . . 14
3.2.2 Jemná lokalizace . . . 15
3.2.3 Kontrola a mapování . . . 18
3.3 Určení typu SPZ . . . 19
3.4 Segmentace . . . 22
3.4.1 Příprava . . . 23
3.4.2 Segmentace řádků . . . 26
3.4.3 Segmentace znaků . . . 27
3.4.4 Párování znaků . . . 28
3.4.5 Ořezávání znaků . . . 29
3.5 Rozpoznávání číslic . . . 30
3.5.1 Normalizace . . . 31
3.5.2 Výběr příznaků . . . 32
3.5.3 Rozpoznávání . . . 35
3.5.4 Věrohodnost . . . 36

4 Klasifikátor minimální vzdálenosti **38**
4.1 Teorie . . . 38
4.1.1 Template matching . . . 38
4.1.2 Mahalanobisova vzdálenost . . . 40
4.1.3 Pravidlo k nejbližších sousedů . . . 40
4.2 Výsledky . . . 42
4.2.1 Pravidlo k nejbližších sousedů . . . 43
4.2.2 Mahalanobisova vzdálenost . . . 46

5 Neuronové sítě **47**
5.1 Teorie . . . 47
5.1.1 Model neuronu . . . 47
5.1.2 Dopředné neuronové sítě . . . 49

5.2 Výsledky 54
5.2.1 Dopředné neuronové sítě 54
5.2.2 Věrohodnost rozpoznání 58

6 Skryté Markovovy modely **62**
6.1 Teorie 62
6.1.1 Základní diskrétní model 62
6.1.2 Spojitý HMM 66
6.1.3 Pseudo 2D HMM 66
6.1.4 Hierarchický HMM 68
6.2 Výsledky 69
6.2.1 1D HMM 70
6.2.2 Pseudo 2D HMM 72
6.2.3 Hierarchický HMM 74
6.2.4 Věrohodnost rozpoznání 76

7 Závěr **77**
7.1 Srovnání metod rozpoznávání číslic 77
7.2 Zhodnocení 78
7.3 Možnosti dalšího vývoje 79

A Přílohy **80**
A.1 Obsah CD 80
A.2 Ukázkové programy 81
A.3 Použité nástroje 82

Literatura **83**

Seznam obrázků

2.1 Popis jednořádkové SPZ 5
2.2 Popis dvouřádkové SPZ 5
2.3 Soukromá Abu Dhabi SPZ zelená 6
2.4 Soukromá Abu Dhabi SPZ modrá 6
2.5 Soukromá Abu Dhabi SPZ červená 6
2.6 Soukromá Abu Dhabi SPZ šedá 6
2.7 Abu Dhabi SPZ taxi, pro export a dočasná 6
2.8 Vydařená fotografie vozidla 8
2.9 Proces rozpoznávání SPZ 9
3.1 Oprahovaný vertikální hranový obrázek 14
3.2 Nalezený výřez okolí SPZ 15
3.3 Binární obrázky vertikálních a horizontálních hran 16
3.4 Korekce naklonění binárního obrázku 16
3.5 Sloučení binárních obrázků hran 17
3.6 Lokalizace SPZ pomocí projekcí 17
3.7 Lokalizace SPZ pomocí hledání kontur 17
3.8 Lokalizace SPZ pomocí Houghovy transformace 18
3.9 Typický histogram šedotónového obrázku SPZ 22
3.10 Porovnání globálního a adaptivního prahování 23
3.11 Složené strukturní elementy pro odstranění šumu 24
3.12 Porovnání metod pro filtrování artefaktů 25
3.13 Horizontální projekce textové části SPZ 26
3.14 Vertikální projekce jednotlivých řádků SPZ 27
3.15 Rozdělení spojených znaků na základě párových úseků 28
3.16 Linearita horní strany znaku 29
3.17 Metoda odstranění šroubů pomocí mediánu 30
3.18 Kontrola emirátské nuly pomocí středu řádku 30
3.19 Porovnání metod normalizace znaku 31
3.20 Postup při získávání příznakového vektoru typu *Win* 33
3.21 Maska důležitých oblastí pro počítání příznaků 33
3.22 Ověření věrohodnosti segmentace 37
4.1 Shluky vzorů v příznakovém prostoru 39
4.2 Porovnání algoritmů pro metodu k nejbližších sousedů 43
5.1 Formální neuron 47
5.2 Lineární separabilita dvou množin 49
5.3 Dopředná vrstevnatá neuronová síť se čtyřmi vrstvami 50
5.4 Graf úspěšnosti s různým počtem skrytých neuronů 56
5.5 Chybně klasifikované latin znaky 57
5.6 Chybně klasifikované emirátské znaky 57
5.7 Správně klasifikované znaky 58
5.8 Různé hodnoty prahů pro věrohodnost rozpoznávání 59

6.1 Skrytý Markovův model 63
6.2 1D HMM s architekturou typu „zleva-doprava“ 66
6.3 Pseudo 2D HMM se třemi řádkovými skupinami 67
6.4 Pseudo 2D HMM se speciálními stavy pro konce řádků 68
6.5 Hierarchický HMM se třemi superstavy 69
6.6 Úspěšnost při použití 1D HMM a příznaků typu *Proj* 70
6.7 Úspěšnost při použití 1D HMM a příznaků typu *Win* 71
6.8 Odvození počtu stavů pro pseudo 2D HMM 73

Seznam algoritmů

4.1 Algoritmus k nejbližších sousedů (varianta 1) 41
4.2 Algoritmus k nejbližších sousedů (varianta 2) 41
5.1 Online učení neuronové sítě 52
5.2 Dávkové učení neuronové sítě 52
6.1 Dopředný algoritmus pro HMM 64
6.2 Klasifikace pomocí HMM 64

Seznam tabulek

2.1 Emirátské a latinkou zapsané číslice 4
2.2 Důležité poměry stran SPZ 5
3.1 Rozlišení SPZ podle typu 20
3.2 Metoda počítání nenulových bodů 21
3.3 Vizuální váhy mezi latin a emirátskými číslicemi 36
4.1 Výsledky metody k nejbližších sousedů 44
4.2 Výsledky s použitím přísnějšího testu věrohodnosti 44
4.3 Výsledky metody k nejbližších sousedů s více reprezentanty . . 45
4.4 Výsledky metody k nejbližších sousedů pro příznak typu *Proj* 45
4.5 Výsledky při použití Mahalanobisovy vzdálenosti 46
5.1 Přeučená neuronová síť 52-18-10 54
5.2 Potvrzení zlatého pravidla 55
5.3 Výsledky neuronových sítí pro příznaky typu *Proj* 56
5.4 Výsledky rozpoznávání SPZ pro slabší sítě 61
6.1 Úspěšnost při použití 1D HMM a příznaků typu *Win* 71
6.2 Porovnání úspěšnosti diskrétních pseudo 2D HMM 73
6.3 Úspěšnost při použití pseudo 2D HMM a příznaků typu *Win* . 74
6.4 Porovnání úspěšnosti hierarchického HMM s příznaky typu *Win* 75
6.5 Výsledky hierarchických HMM s příznaky typu *DCT* 75
6.6 Výsledky rozpoznávání SPZ pro hierarchické HMM 76
7.1 Srovnání vybraných metod pro rozpoznávání číslic SPZ 77

Název práce: Rozpoznávání SPZ z jednoho snímku
Autor: Tomáš Vala
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Jana Štanclová
e-mail vedoucího: jana.stanclova@mff.cuni.cz
Abstrakt: Diplomová práce se zabývá analýzou a návrhem systému pro automatické rozpoznávání SPZ z jednoho snímku. Vstupní data pochází ze systému pro měření rychlosti na silnicích v emirátu Abu Dhabi (Spojené arabské emiráty). Cílem je rozpoznat číselný identifikační údaj na SPZ a její typ. V práci jsou detailně rozebrány jednotlivé části systému rozpoznávání SPZ. Zvláštní důraz je kladen na segmentaci jednotlivých znaků a jejich seskupování, aby mohly být při rozpoznávání využity informace specifické pro SPZ z Abu Dhabi. Diplomová práce nabízí srovnání několika metod na rozpoznávání znaků — klasifikátor minimální vzdálenosti, dopředné neuronové sítě a skryté Markovovy modely. Jednotlivé metody (a různé jejich modifikace) jsou otestovány na reálných datech a porovnány podle procenta správně rozpoznaných číslic (číslice byly nejprve ručně klasifikovány). Špatně rozpoznané SPZ jsou nežádoucí. Místo toho je v některých případech lepší označit SPZ jako „nerozpoznanou“. Proto byly zavedeny sofistikované testy pro ověření věrohodnosti rozpoznaných číslic. Součástí práce je i zhodnocení vlastních výsledků.
Klíčová slova: SPZ, státní poznávací značky, klasifikátor minimální vzdálenosti, dopředné neuronové sítě, skryté Markovovy modely

Title: Single-Image License Plate Recognition
Author: Tomáš Vala
Department: Department of Software Engineering
Supervisor: RNDr. Jana Štanclová
Supervisor's e-mail address: jana.stanclova@mff.cuni.cz
Abstract: The diploma thesis deals with analysis and design of an automatic system for a single-image license plate (LP) recognition. Input data came from a system for car speed measurement on the roads in the emirate of Abu Dhabi (the United Arab Emirates). A goal is to recognize a numeric identification code on a LP and a type of the LP. In details there are discussed individual parts of the system for LP recognition. A particular stress is put on segmentation and grouping of individual characters to be able to utilize specific information on the LP from Abu Dhabi during the character recognition. The diploma thesis compares several methods for character recognition — a minimum distance classifier, feed-forward neural networks and hidden Markov models. The methods (and some modifications of them) are tested on real data and compared according to the percentage of correctly recognized digits (at first the digits were manually classified). Wrongly recognized LPs are undesirable. Instead in some cases it is better to mark the LP as "unrecognized". That's why there were introduced sophisticated tests for recognized digits reliability verification. A summary of our own results is included.
Keywords: LP, license plate, minimum distance classifier, feed-forward neural networks, hidden Markov models

1 Úvod

Metody rozpoznávání obrazu a počítačového vidění v současnosti nacházejí stále širší uplatnění v nejrůznějších oborech jako doprava, medicína, robotika, kontrola kvality, zábavní průmysl atd. Jejich cílem je vytvářet dokonale autonomní vizuální systémy, které nepotřebují přítomnost lidského operátora. Mohou se také snažit maximálně ulehčit práci odborníkovi (např. chirurgovi) při analýze složitých obrazových dat. Hlavním přínosem je pak větší rychlost a efektivita práce.

Jedním ze zajímavých a stále se rozvíjejících odvětví je i rozpoznávání SPZ. Soustředí se na identifikaci vozidla z fotografie, ve které musí nejdříve vyhledat a poté určit jeho SPZ.

Modul rozpoznávání SPZ může být integrován do velké řady systémů. Za zmínku stojí kontrola a měření rychlosti projíždějících vozidel, hlášení závažných dopravních přestupků (např. průjezd křižovatkou na červenou), kontrola vjezdu na střežených parkovištích, automatické vybírání elektronického mýtného na dálnici, pomoc při hledání kradených aut, monitorování hustoty dopravy apod.

1.1 Cíle práce

Cílem práce bude zanalyzovat úlohu rozpoznávání SPZ Spojených arabských emirátů (konkrétně emirátu Abu Dhabi) z fotografií pořízených systémem na kontrolu rychlosti vozidel. Na základě tohoto rozboru se pokusíme navrhnout a popsat jednotlivé fáze činností systému na automatické rozpoznávání SPZ.

Výsledkem práce nebude hotový systém, ale pouze jeho návrh. Zaměříme se hlavně na oblast segmentace a rozpoznávání jednotlivých znaků s využitím specifických informací získaných z analýzy problému. Budeme se snažit využít poznatků z dostupných odborných článků a publikací.

Na téma rozpoznávání SPZ bylo zpracováno mnoho článků, ve kterých se předkládají různé přístupy pro vyhledání SPZ v obrázku a segmentaci jed-

notlivých znaků. Vybrané postupy se pokusíme implementovat a přizpůsobit požadavkům analyzované úlohy.

Odborné články preferují nejrůznější metody rozpoznávání znaků. Zaměříme se hlavně na klasifikátor minimální vzdálenosti (článek [27]) a dopředné neuronové sítě (článek [20]). Pro rozpoznávání znaků SPZ se pokusíme využít zkušeností a modelů i z jiných oblastí rozpoznávání vzorů (rozpoznávání hlasu a obličejů) a otestujeme skryté Markovovy modely (článek [18]).

Hlavním úkolem práce bude vybrané metody rozpoznávání znaků porovnat a aplikovat na konkrétní problém rozpoznávání SPZ. Experimentálně získané výsledky zhodnotíme a vybereme nejvhodnější metodu pro zadanou úlohu rozpoznávání.

1.2 Obsah práce

Důležité pojmy v této práci jsme označili *kurzívou* (pouze v místě zavedení nového pojmu). Obrázky, tabulky a algoritmy jsou číslovány podle kapitol a jejich seznam je k dispozici na začátku práce. Důležité rovnice a vztahy jsou rovněž číslovány s ohledem na kapitoly. Seznam použité literatury je uveden na konci dokumentu a je řazen podle příjmení prvního z autorů.

Nyní popíšeme, o čem se pojednává v jednotlivých kapitolách:

Problematika

Kromě specifikace zadané úlohy obsahuje uvedení do problematiky rozpoznávání SPZ a nastínění procesu, kterým musí analyzovaná fotografie projít. Popisuje omezení plynoucí z nasazení systému v reálném prostředí.

Rozpoznávání SPZ

Detailněji rozebírá jednotlivé fáze procesu rozpoznávání se zaměřením na předzpracování, lokalizaci a segmentaci důležitých částí fotografie.

Klasifikátor minimální vzdálenosti

Přibližuje jednu z metod použitých pro rozpoznávání znaků. Metoda vychází z počítání vzdálenosti mezi neznámými vzory a zvolenými reprezentanty.

Neuronové sítě

Nabízí popis využití dopředných vrstevnatých neuronových sítí pro rozpoznávání znaků a stručným způsobem podává potřebnou teorii. Jsou zde zmíněny různé optimalizace pro učení sítí i samotnou klasifikaci.

Skryté Markovovy modely

Zaměřuje se na rozpoznávání znaků pomocí těchto pravděpodobnostních modelů. Poznatky z oblasti rozpoznávání řeči jsou zde aplikovány na rozpoznávání obrazu (speciálně navržené modifikace).

Závěr

Shrnuje výsledky a porovnání jednotlivých metod. Jsou zde zhodnoceny dosažené cíle a nastíněny možnosti dalšího vývoje.

Přílohy

Komentuje obsah přiloženého CD a podává stručný návod na použití demonstračních programů, které se nacházejí na CD.

2 Problematika

2.1 Specifikace úlohy

Jako vstupní data pro tuto práci nám posloužily fotografie z reálného prostředí pořízené ve Spojených arabských emirátech. Konkrétně se jedná o radary monitorující provoz na silnicích v emirátu Abu Dhabi [Abú Zabí].

Tyto fotografie poskytla firma Ramet CHM a.s, která s touto zemí dlouhodobě spolupracuje. Protože se automatické rozpoznávání SPZ v poslední době stalo pro systémy kontroly rychlosti téměř nutností, musela na to reagovat i tato firma. Aby si udržela místo na trhu, bylo nutné začít vyvíjet vlastní systém úplně od začátku. Již existující systémy (např. pro rozpoznávání českých SPZ) totiž nebylo možné použít kvůli velké odlišnosti emirátských SPZ.

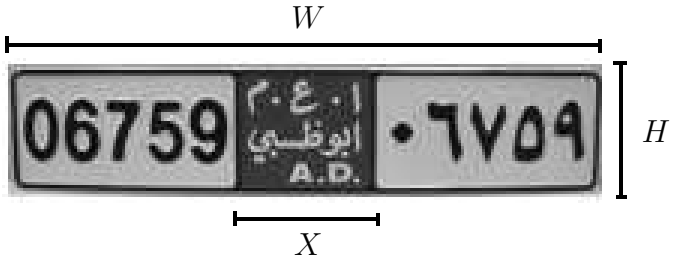
Předmětem rozpoznání je číselný údaj na značce a její typ. Identifikační číslo vozidla má proměnnou délku od jedné do šesti číslic. Písmena se pro identifikaci nepoužívají. Na SPZ se vyskytují dva druhy číslic — „klasické“ arabské číslice (dále jen *latin* číslice) a arabské číslice psané arabským písmem (dále jen *emirátské* číslice). Narozdíl od arabského textu se číslice čtou standardně zleva doprava. Jejich kompletní přehled zachycuje tabulka 2.1.

číslice	0	1	2	3	4	5	6	7	8	9
latin	0	1	2	3	4	5	6	7	8	9
emirátské	٠	١	٢	٣	٤	٥	٦	٧	٨	٩

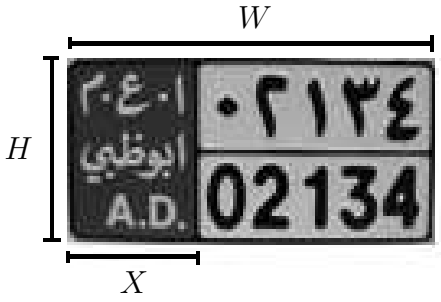
Tabulka 2.1: Emirátské a latinkou zapsané číslice

Identifikační údaj je většinou na SPZ zapsán duplicitně. Této redundance využijeme pro kontrolu výsledku rozpoznávání. Je to užitečné v situaci, kdy je část SPZ nečitelná.

Sektory, do nichž se jednotlivé znaky „vpisují“, jsou pro všechny znaky stejně velké. Rozestupy mezi sektory jsou konstantní (pro latin a emirátské znaky zvlášť) a nezávisí na šířce konkrétního znaku. Tento fakt usnadňuje párování odpovídajících si latin a emirátských znaků. Velikost sektoru se však může lišit pro dvě různé SPZ. Každá SPZ je totiž fotografována z trochu jiné vzdálenosti a pod jiným úhlem.



Obrázek 2.1: Popis jednořádkové SPZ



Obrázek 2.2: Popis dvouřádkové SPZ

Šířka *barevného proužku*, který spoluurčuje typ značky, je na obrázcích 2.1 a 2.2 označena písmenem X. U jednořádkových značek odděluje latin a emirátskou část a u dvouřádkových SPZ je vždy umístěn u levého okraje. Jeho šířka se pro daný typ SPZ nemění.

Písmena W a H udávají šířku a výšku celé značky. Na základě poměru W/H se určuje, o jaký typ SPZ se jedná. Tabulka 2.2 shrnuje poměry stran důležité pro rozpoznávání.

typ	W/H	X/H
jednořádková	4,82	1,18
dvouřádková	2,04	0,74

Tabulka 2.2: Důležité poměry stran SPZ

Obrázek 2.3: Soukromá Abu Dhabi SPZ zelená

Obrázek 2.4: Soukromá Abu Dhabi SPZ modrá

Obrázek 2.5: Soukromá Abu Dhabi SPZ červená

Obrázek 2.6: Soukromá Abu Dhabi SPZ šedá

(a) taxi (b) pro export (c) dočasná

Obrázek 2.7: Abu Dhabi SPZ taxi, pro export a dočasná

V Abu Dhabi existuje několik základních typů SPZ. Samostatnou kategorii tvoří soukromé SPZ (obrázky 2.3, 2.4, 2.5 a 2.6), které se mezi sebou liší jen barvou proužku. U jednořádkových SPZ je proužek umístěn uprostřed mezi latin číslice (nalevo) a emirátské číslice (napravo). U dvouřádkové verze je proužek zcela vlevo a číselný údaj je zapsán ve dvou řádcích — horní emirátskými a spodní latin číslicemi.

SPZ, kterou používají vozidla taxi služby (obrázek 2.7a), svým rozvržením odpovídá dvouřádkové soukromé SPZ. Problém při rozpoznávání mohou způsobit další dvě výjimečné SPZ určené pro export do jiné země (obrázek 2.7b) a pro dočasné používání (obrázek 2.7c), tzn. dokud nebude vystavena plnohodnotná SPZ.

Exportní značka se odlišuje svým světlým textem na tmavém pozadí. Po oprahování je nezbytné její pixely invertovat. Identifikace je zapsána pouze ve spodním řádku latin číslicemi. Dočasná značka je charakteristická červeným písmem na bílém pozadí. Segmentace pro ni musí být řešena individuálně, protože narozdíl od jiných dvouřádkových SPZ má ve spodním řádku číselný údaj uveden emirátskými číslicemi.

Velikost těchto SPZ nezávisí na počtu použitých číslic a proto je možné počítat s pevným poměrem stran (pro jednořádkové a dvouřádkové zvlášť). Rozestupy mezi číslicemi jsou stejné jak pro latin, tak pro emirátské, což také výrazně usnadní jejich párování.

2.2 Proces rozpoznávání SPZ

Představme si následující situaci. Řidič automobilu spěchá na velmi důležitou schůzku a rozhodne se tedy nerespektovat omezení rychlosti stanovené pro tento typ silnice. V jistém okamžiku míjí policejní radar, který měří v daném úseku rychlost projíždějících vozidel. Jedná se o samočinný radarový systém bez policejní obsluhy, který si rychlost projíždějícího vozidla zaznamená a zároveň pořídí jeho fotografii.

Fotografie (ilustrační obrázek 2.8) s doprovodnými údaji je ihned odeslána na centrální server k vyhodnocení. Ještě než může být identifikováno vozidlo (nebo vozidla) na fotografii, musí snímek nejdříve projít několika důležitými fázemi. Náznak těchto fází podává článek [6]. Celý proces pak zachycuje obrázek 2.9.

Nyní si ve stručnosti popíšeme jednotlivé fáze:

1. Nejdříve je snímek uložen do databáze společně s datem a místem pořízení, povolenou a naměřenou rychlostí, aby mohla být fotografie případně předložena jako důkazní materiál.

2. Na základě měřícího stanoviště jsou provedeny příslušné obrazové korekce fotografie v závislosti na známých charakteristikách snímacího zařízení (kamery, fotoaparátu).

Obrázek 2.8: Vydařená fotografie vozidla doplněná o informační proužek (nahoře a dole) — původně v rozlišení $1392 \times 1030 \times 48$

3. V obrázku jsou nalezeny tzv. *kandidátní oblasti*, které by mohly představovat oblast SPZ.

4. Oblasti jsou podrobeny bližšímu zkoumání za účelem odfiltrování všech nalezených oblastí, které ve skutečnosti nejsou SPZ.

5. Každá oblast, která byla identifikována jako SPZ, je namapována na obdélník stanovené velikosti.

6. Z výsledného barevného obrázku SPZ je určen její typ a očekávaný počet řádků (jestli se jedná o jednořádkovou nebo dvouřádkovou SPZ).

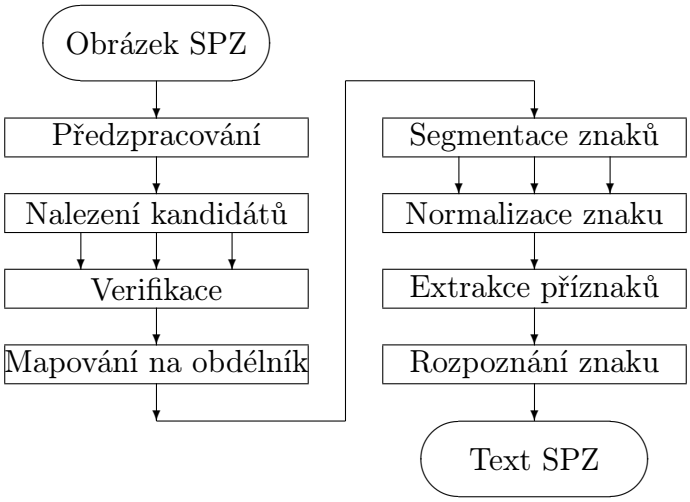
7. Obrázek musí být příslušně oprahován. Dále nastupuje fáze, při níž je nutné pečlivě segmentovat jednotlivé znaky.

8. Znaky jsou rozděleny do dvou skupin (latin a emirátské) v závislosti na jejich poloze v rámci SPZ, typu SPZ a počtu jejích řádků. Na základě těchto informací jsou dále seskupeny do dvojic navzájem si odpovídajících znaků.

9. Každý znak ze dvojice je pak samostatně rozpoznán. Pokud se oba znaky z dvojice neshodují, vybere se ten, jehož výsledek působí věrohodněji nebo se prohlásí za nečitelný. Sekvence po sobě jdoucích rozpoznaných znaků tvoří výslednou identifikaci vozidla.

10. Pokud se nepodařilo text SPZ identifikovat, musí se do procesu vložit lidský faktor. Policista na centrále má přístup do seznamu všech neroz-

poznaných SPZ a může vizuálně vyhodnotit fotografii a zapsat text do databáze ručně.



Obrázek 2.9: Proces rozpoznávání SPZ

Nakonec je identifikované vozidlo vyhledáno v interní databázi policie a tento záznam je předán k dalšímu zpracování. Může být vystavena příslušná faktura nebo pokuta. Zároveň může dát systém signál policejní jednotce ke stíhání kradeného vozidla v případě, že jeho barva a typ neodpovídá údajům uloženým v databázi.

2.3 Problematika reálného prostředí

Celý problém vypadá poměrně jednoduše a zdá se i snadno řešitelný. To však platí jen tehdy, pokud by se situace odehrávala v předem stanovených laboratorních podmínkách. Ve skutečném prostředí tomu tak vůbec není. Ukážeme si, že řada faktorů více či méně proces rozpoznávání komplikuje.

Čím více omezujících podmínek můžeme v úloze předpokládat, tím snáze je pak řešitelná. Existuje spousta článků, kde se uplatňují nejrůznější heuristiky těžící z předem známých omezení (např. [20]).

Samostatnou kategorii tak tvoří systémy pro rozpoznávání SPZ, které monitorují vjezd na hlídaná parkoviště. Lokalizace SPZ ve snímku je pak znatelně jednodušší. Vozidlo totiž vždy zastaví na předepsaném místě před závorou. Jeho polohu mohou ještě zpřesnit čidla zabudovaná v zemi. Můžeme tedy učinit silné předpoklady o poloze SPZ ve snímku. Navíc lze předpokládat, že obrys SPZ bude obdélník, jehož strany jsou téměř rovnoběžné s osami obrázku.

Metody využívající znalosti o přibližné poloze SPZ jsou však zcela nepoužitelné v obecném případě, kdy jsou radar i kamera umístěny na okraji vozovky. Vlivem projektivní transformace zde dochází k značné deformaci obdélníku a jednotlivé strany nemusí být nutně rovnoběžné. Obecně se také může značně lišit velikost segmentovaných SPZ pro fotografie vozidel z různých vzdáleností. To je zvláště patrné v případě, když kamera snímá vozovku s více souběžnými pruhy.

Sami autoři článku [20] uvádějí, že také nepříznivé světelné podmínky mohou výrazně snížit výkonnost celého systému. Potíže často působí vržené stíny a nejrůznější odlesky (osvětlení vozidla, blesk fotoaparátu, zapadající slunce). Někteří řidiči dokonce využívají speciálních nelegálních přípravků, které způsobují, že jejich SPZ po ozáření bleskem vykazuje vysokou odrazivost. Výsledná SPZ je potom úplně nečitelná. Přisvícení bleskem v infračervené oblasti může přinést značné zlepšení. Mobilní radarové systémy napájené z baterií však tuto možnost vylučují díky velké spotřebě infračerveného blesku nebo reflektoru.

Je nutné si také uvědomit, že většinou nepořizujeme fotografie statických scén. Musíme tedy brát v úvahu rozmazání pohybem. Rychlá uzávěrka dává pouze malou hloubku ostrosti pořízeného snímku. Radarový systém musí být dobře kalibrován, aby dokázal zachytit i velmi rychle jedoucí vozidla a speciálně celou jejich SPZ. SPZ může být „uříznutá“ nejen díky špatně načasované spoušti fotoaparátu, ale také kvůli vozidlu, které jede ve vedlejším pruhu, nebo přesahujícímu nákladu.

Tabulka SPZ není vždy úplně čistá a může být dokonce pokřivená následkem předchozího nárazu. Segmentaci také znepříjemňují různé artefakty jako jsou šrouby pro přichycení tabulky, kousky bláta a nejrůznější nálepky (např. doklad o provedení STK).

Z uvedených charakteristik jasně vyplývá, že sestrojit univerzální systém pro rozpoznávání SPZ tak, aby byl schopen klasifikovat SPZ libovolného státu, je téměř nemožné. Při návrhu se tedy omezujeme pouze na typy SPZ, které se v dané oblasti vyskytují nejčastěji, a snažíme se využít jejich specifických charakteristik. Při klasifikaci jsme se setkali i s jinými typy SPZ, ale procento takových značek bylo zanedbatelné.

Často chceme monitorovat rychlost projíždějících vozidel i na velmi frekventovaných silnicích. Opravdu markantní je to například na dálnici. Proto musí být celý systém navržen tak, aby dokázal pracovat v reálném čase a poskytoval přijatelnou odezvu.

Dalším důležitým kritériem je věrohodnost a spolehlivost systému. Aby mohl být systém použit například pro vybírání mýtného na dálnici, musí jeho spolehlivost splňovat přísná kritéria stanovená vládou. Omezení se týkají hlavně procenta chybně rozpoznaných značek (je tolerováno maximálně $0,001\%$ chybných klasifikací). Většinou se vyplatí značku raději nerozpoznat, než aby byla klasifikována špatně.

Systém pak lze hodnotit na základě následujících vztahů:

$$p_1 = \frac{\text{počet správně rozpoznaných SPZ}}{\text{celkový počet rozpoznávaných SPZ}} \quad , \tag{2.1}$$

$$p_2 = \frac{\text{počet nerozpoznaných SPZ}}{\text{celkový počet rozpoznávaných SPZ}} \quad , \tag{2.2}$$

$$p_3 = \frac{\text{počet chybně rozpoznaných SPZ}}{\text{celkový počet rozpoznávaných SPZ}} \quad . \tag{2.3}$$

Zjevně platí:

$$p_1 + p_2 + p_3 = 1 \quad .$$

Použitelný systém splňuje navíc podmínku:

$$p_1 > p_2 > p_3 \quad . \tag{2.4}$$

3 Rozpoznávání SPZ

V předchozí kapitole byl nastíněn proces rozpoznávání SPZ (obrázek 2.9). Nyní si rozebereme detailněji jeho jednotlivé fáze (např. lokalizace SPZ, segmentace znaků) se zaměřením na konkrétní praktické problémy spojené se specifikovanou úlohou.

3.1 Předzpracování

Data jsou obvykle získávána z různých typů digitálních kamer, které mají odlišné snímací charakteristiky. Výsledné fotografie jsou však zpracovávány jedním systémem, a proto je potřeba sjednotit formát vstupních dat (fotografií). Na základě známých specifických vlastností každého snímače je možné provést určité korekce snímku tak, aby byl návrh systému nezávislý na typu používaných kamer. Úpravy však musí být vykonávány zcela automaticky bez zásahu člověka.

Mezi základní úpravy patří tzv. *vyvážení bílé barvy*, což je nutné udělat obzvláště pečlivě. Obvykle se tak činí pomocí referenčního bílého objektu (např. list papíru). Znamená to vlastně správně nastavit váhy pro jednotlivé barevné složky (červenou, modrou a zelenou), aby byl referenční objekt na snímku skutečně bílý.

Barevná informace je klíčová pro rozpoznání typu SPZ. Barvu mohou ovlivnit nejen parametry kamery, ale také různé světelné podmínky ve snímané scéně. Zapadající slunce dá snímkům jiný nádech než osvícení umělým bleskem. Příslušné korekce by tedy měly být závislé na denní době, kdy jsou snímky pořizovány.

Ekvalizace histogramu [28, str. 60–61] výrazně zlepšuje výkon segmentační fáze, protože většinou zvyšuje kontrast a čitelnost celého snímku. Zároveň ale poškozuje barevnou informaci v obrázku a tím snižuje věrohodnost při určování typu SPZ. Ve fázi předzpracování se tedy ekvalizace histogramu neaplikuje, ale použijeme ji později.

U vstupních zařízení se mohou objevovat různé vady, které je nutné před samotným zpracováním odstranit. Příkladem může být tzv. *soudkovitost* při použití širokoúhlých objektivů. Fotografie je narozdíl od objektivu obdélníková.

Ve výsledném snímku se může projevovat jev zvaný *vinětace*. Je zapříčiněn tím, že čočka tlumí více paprsky, které svírají s optickou osou velký úhel, než ty, které směřují do objektivu podél této osy. Protože jsou tím ovlivněny pouze rohové (okrajové) oblasti, není nutné se jeho odstraněním v našem případě zabývat — oblast zájmu rozpoznávání leží obvykle uprostřed snímku. Většina lepších přístrojů má již dnes zabudovanou automatickou korekci této vady.

Fotografie mohou být často zatíženy *šumem*. Šum bývá nejčastěji způsoben nečistotami na čočce objektivu nebo elektrickým šumem vznikajícím přímo v kameře. Zatímco prvnímu můžeme většinou spolehlivě předejít stálou údržbou zařízení, druhý typ se odstraňuje nesnadno.

Výhodou je, pokud známe přesné charakteristiky tohoto šumu. Odstraňovat šum pomocí různých vyhlazovacích technik se v tomto kroku nevyplatí, protože zároveň dochází k degradaci důležitý hran nutných například pro lokalizaci SPZ. Nejspolehlivější metodou na odstranění šumu je průměrování více stejných obrázků, to však v našem případě není možné, poněvadž se scéna rychle a dosti dynamicky mění. Odstranění šumu jsme nechali až na další fáze.

Protože nám jde o fotografování rychle se pohybujících objektů, musíme zvolit krátkou dobu expozice na úkor malé hloubky ostrosti. Přesto však bývají některé snímky rozmazané pohybem nebo špatným zaostřením. *Dekonvoluce* [28, str. 106] slouží k odstranění těchto poruch pomocí analýzy obrazu ve frekvenční oblasti. Přesný odhad parametrů rozmazání je klíčový, jinak dochází k vážnému znehodnocení snímku.

V závislosti na umístění kamery vzhledem k pozorované scéně je možno určit oblast (obdélník) zájmu ve snímku, kde se může SPZ vyskytovat. Zbylé části fotografie se mohou oříznout nebo se používají pro zobrazení dodatečných údajů uživateli. Naše vstupní fotografie mají v horní a spodní části informační proužek obsahující mimo jiné naměřenou rychlost vozidla (obrázek 2.8). Informační proužek (známých rozměrů) jsme před samotným rozpoznáváním SPZ automaticky odstranili.

U fotografií, které jsme měli k dispozici, bylo provedeno vyvážení bílé barvy, odstraněna soudkovitost a vinětace (na základě známých charakteristik kamery). Neměli jsme možnost experimentovat s korekcemi závislými na denní době. Automatickou korekci rozmazání fotografií jsme neimplementovali, protože se rozmazání vyskytovalo jen na malém množství fotografií.

3.2 Lokalizace oblasti SPZ

Označení oblastí SPZ jsme nejdříve provedli ručně pro všechny fotografie (pořízené systémem na měření rychlosti), které jsme měli k dispozici. Připravili jsme si *malé výřezy* (SPZ a malé okolí) z celých fotografií (tzv. *hrubá lokalizace*). V těchto malých výřezech jsme manuálně označili pozice rohů oblasti SPZ (tzv. *jemná lokalizace*).

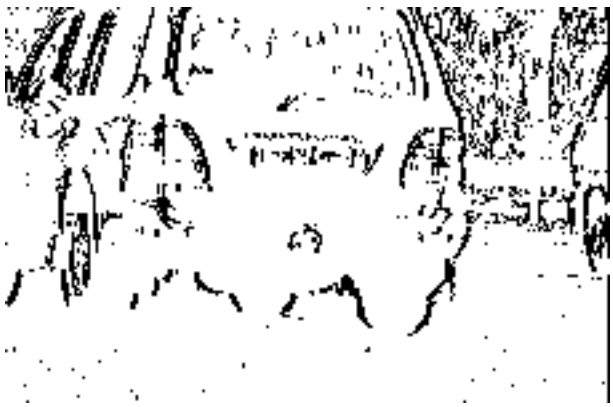
Automatickou lokalizací oblasti SPZ jsme se zabývali pouze okrajově, protože z přečtených článků a pokusů jsme poznali, že se jedná o samostatný a velmi obsáhlý problém. Vyzkoušeli jsme několik metod pro vyhledání SPZ ve snímku. Důležitým kritériem pro výběr vhodné metody byla přesnost nalezení hranic oblasti SPZ, poněvadž na tom závisí přesnost segmentace a rozpoznávání číslic SPZ.

3.2.1 Hrubá lokalizace

Žádná z testovaných metod nebyla natolik robustní, aby dokázala přesně označit oblast SPZ v celém snímku. Proto jsme museli oddělit hrubou lokalizaci SPZ do samostatného kroku, aby mohly být při jemné lokalizaci předkládány pouze malé výřezy s okolím SPZ.

Při návrhu postupu hrubé lokalizace SPZ jsme vycházeli z článků [3] a [14], jejichž přístup se nám zdál nejvhodnější pro naši úlohu. Předpokládá se, že oblast SPZ obsahuje mnoho výrazných vertikálních hran (ve vodorovném směru se rychle střídají světlé a tmavé oblasti číslic a pozadí).

Nejdříve jsou ve snímku pomocí *Sobelova operátoru* [28, str. 82] nalezeny hrany ve vertikálním směru. Výsledný obrázek popisující intenzity vertikálních hran (tzv. *vertikální hranový obrázek*) je oprahován, abychom získali binární obrázek oblastí s výraznými hranami (obrázek 3.1).



Obrázek 3.1: Oprahovaný vertikální hranový obrázek získaný pomocí Sobelova operátoru z celého snímku

Dále jsou spočítány horizontální projekce (bude vysvětleno v sekci o segmentaci na str. 26) binárního obrázku a určeny řádky obrázku s velkým podílem vertikálních hran (řádky s hodnotou projekce nad empiricky stanovenou prahovou hodnotou).

Tyto řádky jsou v binárním obrázku procházeny oknem velikosti 100×1 a v každé pozici okna se určuje počet souvislých úseků pixelů s nenulovou hodnotou. Šířku okna jsme zvolili tak, aby mohlo okno ve správné pozici (v oblasti SPZ) protínat zároveň několik znaků.

Pozice okna, pro které počet úseků překračuje stanovenou hodnotu, jsou zaznamenány. Blízké pozice potom definují oblast zájmu, ve které by se mohla nacházet SPZ. Okolo blízkých pozic je označen malý výřez, který postupuje do fáze jemné lokalizace.

Na testovaných fotografiích se nám podařilo správně detekovat jen velmi málo oblastí SPZ (obrázek 3.2). V obrázcích se totiž vyskytovalo mnoho výraznějších hran právě mimo oblast SPZ. Proto je navržená metoda pro naši úlohu prakticky nepoužitelná a tuto fázi je zatím nutné provádět ručně.

Obrázek 3.2: Nalezený výřez okolí SPZ

3.2.2 Jemná lokalizace

Úkolem jemné lokalizace je nalezení obrysů oblasti SPZ v rámci menšího výřezu z celého snímku. Výstupem této fáze jsou souřadnice čtyř rohů oblasti SPZ ve výřezu. Čtyřúhelníková oblast definovaná těmito rohy se nazývá *kandidát*.

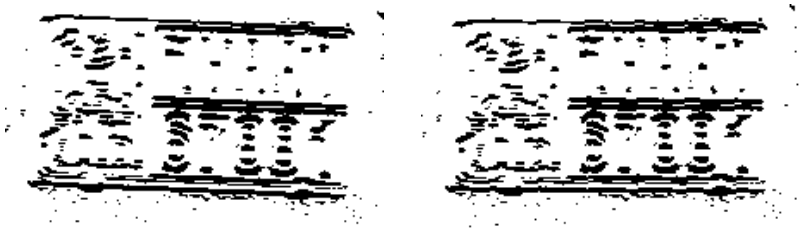
Metody pro lokalizaci SPZ jsou v článcích popsány často pouze mlhavě a není snadné je na základě popisu prakticky implementovat. Naše metoda pro jemnou lokalizaci využívá stejné myšlenky jako hrubá lokalizace SPZ.

Pro detekci hran ve vodorovném a svislém směru se použije Sobelův operátor. Získáme dva samostatné horizontální a vertikální hranové obrázky, které jsou následně oprahovány za účelem nalezení pouze výrazných hran (obrázek 3.3). Celý proces (další kroky) se opakuje pro několik prahových hodnot (empiricky stanovené hodnoty).

(a) vertikální (b) horizontální

Obrázek 3.3: Binární obrázky vertikálních a horizontálních hran

Obrázek SPZ je potřeba natočit do správné pozice tak, aby horní a spodní okraj oblasti SPZ byly rovnoběžné s osou x. K tomuto účelu je využit binární obrázek horizontálních hran, který by měl obsahovat dvě dlouhé čáry pro horní a spodní okraj oblasti SPZ. Pro různé natočení tohoto binárního obrázku (podle středu obrázku o malé úhly) jsou spočítány horizontální projekce. Úhel, pro který je dosaženo maximální hodnoty projekce, je zaznamenán (tzv. *korekční úhel*). Oba binární obrázky horizontálních a vertikálních hran jsou natočeny o nalezený úhel (obrázek 3.4).



(a) před korekcí (b) po korekci

Obrázek 3.4: Korekce naklonění binárního obrázku horizontálních hran

Nyní je nutné určit hodnoty horizontálních projekcí pro binární obrázek vertikálních hran. V horizontálních projekcích se hledá úsek hodnot, které přesahují pevně stanovenou prahovou hodnotu. Úsek musí být dostatečně dlouhý. Počátek a konec tohoto úseku odpovídá umístění horní a spodní hranice oblasti SPZ.

V tomto okamžiku mohou být oba binární obrázky spojeny do jednoho (obrázek 3.5) pomocí operace logického součtu (OR). Jako tzv. *kandidátní obdélník* je označena oblast mezi nalezeným horním a spodním okrajem oblasti SPZ a pravým a levým okrajem spojeného binárního obrázku.

Velikost kandidátního obdélníku je postupně zmenšována zleva, zprava, shora a zespodu do té doby, dokud roste poměr nenulových pixelů (spojeného binárního obrázku) v obdélníku ku jeho ploše. Na konci by měl kandidátní

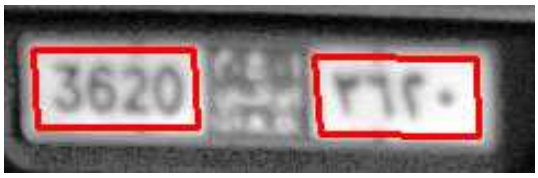
Obrázek 3.5: Sloučení binárních obrázků vertikálních a horizontálních hran

obdélník ohraničovat oblast s vysokou hustotou nenulových pixelů (tedy oblast SPZ).

Z obrázku 3.6 je patrné, že nalezená oblast dobře „kopíruje“ sklon SPZ. Okraje oblasti SPZ však nejsou označeny úplně přesně a to způsobuje problémy při segmentaci znaků.

Obrázek 3.6: Lokalizace SPZ pomocí projekcí a zmenšování kandidátního obdélníku — nalezená oblast SPZ je na obrázku zvýrazněna červeně

Jiný přístup pro lokalizaci SPZ může být hledání kontur uzavřených oblastí (článek [10]) v binárním obrázku. Vlivem prahování nejsou tyto oblasti v binárním obrázku často uzavřeny, takže je algoritmus nenajde. Nalezené kontury mohou být aproximovány pomocí mnohoúhelníku s volitelnou přesností. V našem případě netvoří SPZ samostatnou uzavřenou oblast, ale skládá se z několika podoblastí (obrázek 3.7).



Obrázek 3.7: Lokalizace SPZ pomocí hledání kontur — v originální obrázku jsou červeně zvýrazněny kontury nalezených uzavřených podoblastí v SPZ

Metody (článek [6]) využívající *Houghovy transformace* [28, str. 163] k detekci lineárních útvarů (segmentů čar) v oprahovaném binárním obrázku nejsou moc odolné vůči pokřiveným okrajům SPZ a zkosení způsobenému projektivní transformací (nezachovává rovnoběžnost stran). Na obrázku 3.8 je vidět, že ohraničení SPZ není v některých částech moc výrazné (např. v oblasti barevného proužku), a proto v těchto částech nebyly detekovány téměř žádné segmenty čar.

Obrázek 3.8: Lokalizace SPZ pomocí Houghovy transformace — v originálním obrázku jsou červeně vyznačeny nalezené segmenty čar

Lokalizovat SPZ v malém výřezu se dařilo nejlépe první metodě (projekce hranových obrázků). Přesnost označení oblasti SPZ však nebyla dostačující, a proto jsme raději použili manuální označení rohů.

3.2.3 Kontrola a mapování

Nalezený kandidát dále prochází testy, aby bylo odfiltrováno co nejvíce špatně nalezených oblastí. Vzhledem k tomu, že známe pozice jednotlivých rohů kandidáta, jsme schopni částečně aproximovat poměr stran (šířka/výška). Jako odhad nám poslouží podíl průměru délek dvou delších protějších stran a průměru délek dvou kratších stran.

Pokud se poměr stran kandidáta rovná poměru stran (tabulka 2.2) jednořádkové nebo dvouřádkové SPZ (s tolerancí $0,65$), je oblast klasifikována jako SPZ. V opačném případě není oblast dále zpracovávána. Velikost tolerance jsme zvolili s ohledem na poměry stran SPZ, které jsme při testech naměřili.

V tomto okamžiku je nalezený kandidát téměř připraven k tomu, aby byl pomocí *projektivní transformace* zobrazen na tzv. *mapovací obdélník* (obdélníkový obrázek obsahující pouze SPZ). Parametry projektivního zobrazení získáme ze čtyř dvojic navzájem si odpovídajících referenčních bodů [28, str. 448]. Jsou to rohy SPZ ve snímku (rohy kandidáta) a rohy mapovacího obdélníku.

Rozměry mapovacího obdélníku jsou stanoveny podle odhadovaného počtu řádků SPZ (zda se jedná o jednořádkovou nebo dvouřádkovou SPZ).

Výška musí být přizpůsobena tak, aby výška znaků byla vždy stejná nezávisle na rozměrech SPZ. Díky tomu nemusí být zvětšování (resp. zmenšování) znaků při normalizaci (bude popsána dále) tak výrazné. Šířka obdélníku je pak dopočítána v příslušném poměru. Zachování poměru stran je klíčové například pro rozpoznání typu SPZ i správnou segmentaci.

Když je plocha mapovacího obdélníku příliš malá, dochází ke „slepování“ znaků. Naopak pro velkou oblast se výrazněji projeví šum a objevují se „díry“ a nespojitosti ve znacích. Obojí znesnadňuje segmentaci. Optimální výšku znaku jsme experimentálně stanovili na 30 obrázkových bodů.

Abychom mohli kandidáta správně namapovat na obdélník, musíme mít vrcholy správně očíslované. Připravíme si jejich pořadí po směru hodinových ručiček, kde na prvním místě bude levý horní roh budoucího obdélníku. Levý horní roh může být identifikován například tak, že se nachází výše než předchozí vrchol a více vlevo než následující (při číslování po směru hodinových ručiček). Předpokládáme tak, že nebudeme identifikovat snímky SPZ, které jsou natočené o více jak $\pm 45°$.

Výstupem této fáze je obdélníkový obrázek SPZ (bez okolí). V dalších fázích z něj bude rozpoznán typ SPZ a segmentovány jednotlivé znaky.

3.3 Určení typu SPZ

Jednotlivé typy SPZ rozlišujeme na základě barevné informace v nich obsažené a poměru stran. Vzhledem k tomu, že známe poměry stran jednořádkové a dvouřádkové varianty, dokážeme odhadnout umístění barevného proužku, který je důležitý pro identifikaci typu SPZ.

Navrhli jsme postup, při kterém se nejdříve získá průměrná hodnota obrázkových bodů (tzv. *zprůměrovaná barva*) v oblasti barevného proužku a potom ve zbytku SPZ (oblast SPZ). Barvy jsou reprezentovány v RGB prostoru jako třísložkové vektory s hodnotami v intervalu $\langle 0, 1 \rangle$, kde jednotlivé složky udávají míru zastoupení tří základních barev — červené, zelené a modré.

Vzhledem k použitým barvám proužků SPZ jsme vybrali čtyři tzv. *reprezentativní barvy* — červenou (1 0 0), zelenou (0 1 0), modrou (0 0 1) a bílou (1 1 1). Pro obě zprůměrované barvy určíme nejbližší reprezentativní barvu v RGB prostoru s použitím euklidovské metriky. Tento přístup však ztroskotal na tom, že zprůměrovaná barva je většinou dosti blízká šedé (0,5 0,5 0,5) a odchylky v jednotlivých barevných složkách jsou velmi malé.

Rozhodli jsme se přejít do barevného prostoru *HSV*[1] [29, str. 6], kde jednotlivé složky vyjadřují odstín, sytost a jas dané barvy. Rozhodování na základě barevného odstínu zprůměrované barvy bylo daleko věrohodnější,

[1] angl. *hue, saturation, value*

protože první složka jasně definuje, o jakou základní barvu ze spektra se jedná.

Typ jednořádkových SPZ se dá rozlišit jen na základě barvy proužku, protože se jedná pouze o soukromé SPZ. U jednořádkových SPZ nám tedy zbývalo vyřešit problém s šedým typem soukromé značky. Ukázalo se, že v mnoha případech se takové SPZ klasifikují na základě barevného proužku jako soukromé modré nebo zelené. Většinou to bylo způsobeno barevným laděním celého snímku v závislosti na použitém typu kamery.

Od skutečné soukromé zelené SPZ se nám šedé značky podařilo odlišit na základě druhé složky HSV vektoru (sytosti), poněvadž pro šedé značky byla tato hodnota výrazně nižší. V případě soukromých modrých SPZ byly hodnoty sytosti modré a šedé barvy příliš blízko a často se je nepodařilo rozlišit ani vizuálně.

S tímto výsledkem pro jednořádkové SPZ jsme se spokojili a dále jsme se zabývali dvouřádkovými SPZ. Soukromé dvouřádkové SPZ jsme rozlišili podobně jako jednořádkové. Navíc ale bylo potřeba rozpoznat tři speciální typy dvouřádkových SPZ (taxi, pro export a dočasnou).

Podle dodaných vzorů (obrázky na str. 6) nám bylo jasné, že SPZ pro taxi jsou výjimečné svou barvou pozadí. Ani tady první složka HSV neposkytovala dostatečnou jistotu v rozhodování. Pro zelenou a nazelenalou (bílou s nádechem zelené) barvu je totiž téměř stejná. Znovu jsme použili sytost barvy jako sekundární rozlišovací prvek.

Tabulka 3.1 přehledně shrnuje rozdíly v barvách jednotlivých druhů SPZ.

typ SPZ	proužek (sytost)	text	pozadí (sytost)
soukromá zelená	zelený ($> 18\%$)	černý	bílé
soukromá modrá	modrý ($> 18\%$)	černý	bílé ($< 25\%$)
soukromá červená	červený	černý	bílé
soukromá šedá	šedohnědý ($< 18\%$)	černý	bílé
taxi	modrozelený	černý	zelené ($> 25\%$)
export	modrý	bílý	modré
dočasná	bílý	červený	bílé

Tabulka 3.1: Rozlišení SPZ podle typu

Exportní SPZ nebylo možné klasifikovat na základě barvy pozadí. Velká část plochy je tvořena bílými číslicemi, a proto je zprůměrovaná barva hodně blízká šedé barvě s modrým odstínem. Nepomohlo ani když jsme zmenšili průměrovací oblast SPZ na horní polovinu textové části (zde je menší podíl bílých číslic).

Podobný problém se vyskytoval u dočasných značek, které se nedařilo správně rozpoznat podle barvy proužku. Díky červenému textu měl tento

proužek červený odstín a značka byla v některých případech nesprávně klasifikována jako soukromá červená nebo šedá SPZ.

Pro odlišení dočasných a exportních značek od ostatních dvouřádkových SPZ jsme vymysleli metodu *počítání nenulových bodů* v oprahovaném obrázku. Cílem je určit, zda se v oblasti SPZ (resp. oblasti barevného proužku) vyskytuje tmavý text na světlejším pozadí nebo naopak.

Nejprve je kopie barevného obrázku SPZ převedena do stupňů šedi (šedotónový obrázek). Poté se na tuto kopii aplikuje ekvalizace histogramu a adaptivní prahování (bude popsáno v části věnované segmentaci). Tím vznikne černobílý obrázek, ve kterém vyhledáváme a počítáme nenulové pixely.

Určili jsme dvě oblasti vyhodnocování počtu nenulových pixelů — celá oblast barevného proužku a horní pětina oblasti SPZ. Druhou oblast zájmu jsme vybrali s ohledem na to, aby její převážnou část tvořilo pozadí značky a ne text. Aby algoritmus dokázal rozhodnout, zda se jedná o oblast se světlým textem na tmavém pozadí nebo obráceně, musí určit poměr počtu nenulových bodů vzhledem k velikosti oblasti (tabulka 3.2).

typ SPZ	oblast proužku	oblast SPZ
pro export	—	$> 50\%$
dočasná	$< 33\%$	—
jiná dvouřádková	$> 33\%$	$< 50\%$

Tabulka 3.2: Metoda počítání nenulových bodů — procenta udávají poměr počtu nenulových pixelů vzhledem k velikosti dané oblasti, hodnoty jsme stanovili experimentálně

V konečném důsledku jsme při určování typů SPZ dosáhli těchto výsledků:

- 84% — rozpoznáno správně
- 10% — rozpoznáno špatně
- 6% — nepodařilo se ani vizuálně určit, o jaký typ SPZ se jedná

V posledním případě šlo většinou o problém rozlišení modrých a šedých barevných proužků soukromých SPZ nebo se jednalo o úplně neznámý typ SPZ. Mezi vyhodnocovanými SPZ byly zastoupeny všechny prezentované typy.

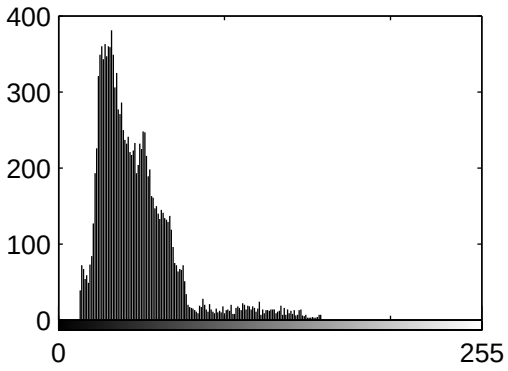
Po určení typu SPZ zbývá rozpoznat číselný identifikační údaj.

3.4 Segmentace

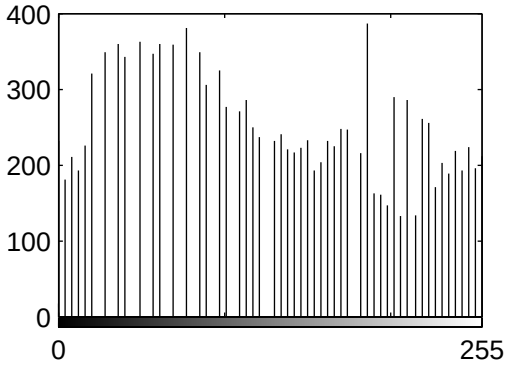
Při segmentaci chceme z obdélníkového obrázku SPZ vyextrahovat jednotlivé znaky, aby mohly být předloženy v další fázi k rozpoznání. Preciznost segmentace ovlivňuje následující rozpoznávání, a proto je nutné jí věnovat zvýšenou pozornost.

Jakmile je vstupní obrázek SPZ předzpracován, je nutné nejdříve segmentovat řádky znaků v SPZ a následně i jednotlivé znaky v rámci řádků. Pro zpřesnění segmentace znaků jsme vymysleli metodu párování a ořezávání znaků. Jednotlivé kroky budou detailně popsány dále. Výstupem této fáze jsou obrázky jednotlivých číslic na SPZ.

Rozpoznávání oddělených znaků se v našem případě ukazuje jako výhodnější, i když se objevují i jiné přístupy rozpoznávání textu, které segmentovací fázi částečně vynechávají (např. [11]). Číselný údaj na předložených SPZ je omezen pouze počtem číslic (3–6).



(a) před ekvalizací histogramu



(b) po ekvalizaci histogramu

Obrázek 3.9: Typický histogram šedotónového obrázku SPZ

3.4.1 Příprava

Cílem tohoto kroku je připravit binární obrázek, ze kterého mohou být segmentovány číslice SPZ. Barevný obrázek SPZ, který přichází na vstup této fáze, musí být nejdříve vhodně předzpracován a oprahován. Nakonec přichází odstranění šumu.

Nejprve je barevný obrázek převeden na šedotónový. Následně je provedena ekvalizace histogramu [28, str. 60–61]. *Histogram* (obrázek 3.9) udává četnosti výskytu jednotlivých jasových hodnot pixelů v obrázku. Praktické pokusy, které jsme provedli, svědčí o tom, že při použití ekvalizace se počet správně rozpoznaných SPZ téměř zdvojnásobil.

Prahování : Jasová funkce šedotónového obrázku musí být oprahována, aby byl text (tzv. *popředí*) jasně odlišen od svého pozadí. *Prahování* znamená vynulování pixelů, jejichž jasová hodnota nepřekračuje stanovenou mez. Ostatní pixely jsou nastaveny na hodnotu 1. Tím vznikne binární obrázek, kde nenulové hodnoty odpovídají pixelům popředí.

Existuje několik způsobů, jak toho dosáhnout. Prahová hodnota je zvolena jediná pro celý obrázek. Může být dána pevně pro všechny obrázky nebo získána na základě histogramu daného obrázku (tzv. *globální prahování*). Automatické nalezení vhodné jasové hranice nebývá jednoduché.

Problém globálního prahování spočívá v tom, že toto prahování není schopno reagovat na lokální změny v jasové funkci. Vhodně zvolená prahovací hodnota pro celý obrázek nemusí být nutně optimální ve všech jeho částech. Komplikace pak působí SPZ, u nichž je část zastíněna nebo je pozadí textu tmavé.

Adaptivní prahování (obrázek 3.10) se oproti tomu snaží nalézt hodnotu prahu pro malé okolí pixelu, který právě zpracovává a jehož hodnotu má oprahovat. Při volbě prahové hodnoty jsme ověřili, že průměrná hodnota pixelů v tomto okolí neposkytuje tak dobré výsledky jako vážený součet, který „kopíruje“ hustotu normálního rozložení se střední hodnotou ve zpracovávaném bodě.

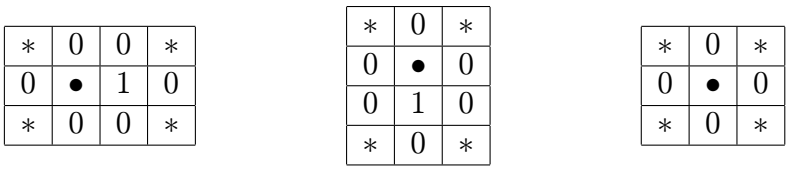
(a) globální prahování (b) adaptivní prahování

Obrázek 3.10: Porovnání globálního a adaptivního prahování

Zkoušeli jsme použít čtvercové okolí různých velikostí, kvůli souměrnosti jsme volili pouze lichá čísla. Nejlépe se nám osvědčily rozměry 21×21.

Odstranění šumu : Data bývají zašuměná a prahováním vzniká v obrázku mnoho nechtěných *artefaktů* (skupina několika osamocených pixelů). Jejich odstranění umožní přesněji určit hranice jednotlivých znaků. V článku [17] je popsán jednoduchý filtr pro odstranění osamocených pixelů a jednobodových děr.

Abychom se dokázali zbavit i o něco větších artefaktů, rozhodli jsme se zvolit *matematickou morfologii* [28, str. 559–568]. Tento přístup nám dovolil použít morfologickou operaci *hit-or-miss* ($\otimes$) [28, str. 568] se *složenými strukturními elementy* (obrázek 3.11).



(a) (b) (c)

Obrázek 3.11: Složené strukturní elementy pro odstranění šumu v binárním obrázku

Při této operaci je strukturní element virtuálně přiložen *referenčním bodem* (v obrázku 3.11 vyznačen • a hodnota této buňky je 1) k právě zkoumanému pixelu binárního obrázku. Jednotlivé buňky strukturního elementu pak udávají, jestli hodnota pixelů v okolí referenčního bodu má být 0 nebo 1 nebo jestli nám na hodnotě nezáleží (symbol $*$).

Výstupem je opět binární obrázek s hodnotami 1 pro všechny zkoumané pixely (původního obrázku), jejichž okolí vyhovovalo přiloženému strukturnímu elementu. V opačném případě je výstupní pixel nastaven na hodnotu 0.

Operace $\otimes$ se dá implementovat pomocí základní morfologické operace eroze ($\ominus$) [28, str. 565], logického součinu (&) a doplňku ($'$):

$$I \otimes B = (I \ominus B_1) \& (I' \ominus B_2) \quad , \tag{3.1}$$

kde I označuje binární obrázek a složený strukturní element B je rozložen na dva elementární strukturní elementy B_1 a B_2, pro něž platí:

$$\begin{aligned} &B_1(i,j) = 1, B_2(i,j) = 0 \quad \text{pro } B(i,j) = 1 \\ &B_1(i,j) = 0, B_2(i,j) = 1 \quad \text{pro } B(i,j) = 0 \\ &B_1(i,j) = 0, B_2(i,j) = 0 \quad \text{pro } B(i,j) = * \quad . \end{aligned}$$

Pokud chceme vynulovat pixely obrázku, které označila operace $\otimes$, můžeme použít vztahu:

$$I \leftarrow I \& (I \otimes B)' \quad .$$

Uvedeme si jednoduchý příklad, kdy se v obrázku vyskytuje artefakt složený ze dvou pixelů umístěných vodorovně vedle sebe. Operace $\otimes$ za pomoci strukturního elementu 3.11a detekuje pouze levý pixel tohoto artefaktu. Z toho vyplývá, že na závěr musí být na výsledný obrázek ještě použit složený strukturní element 3.11c, aby označil i druhý pixel artefaktu.

Podobně se použije složený strukturní element 3.11b pro detekci (horního pixelu) artefaktu složeného ze svou pixelů umístěných pod sebou.

Buňky označené symbolem $*$ jsme zvolili tak, aby se eliminovaly i artefakty, které mají ve svém okolí nějaký nenulový pixel, ale dotýkají se ho pouze rohem. To znamená, že nenulový pixel se může nacházet v některé ze čtyř pozic definovaných symbolem $*$.

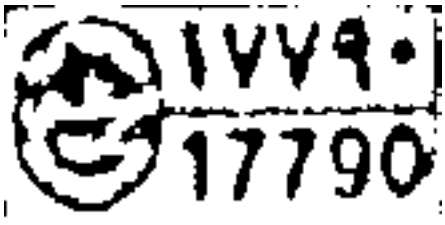
Na základě předchozích úvah jsme sestrojili vlastní operátor ($\odot$):

$$I \leftarrow I \odot B = (I \ominus B_1) \& (I' \ominus B_2)' \quad .$$

Je to drobně modifikovaný vzorec 3.1. V podstatě se jedná o erozi ($\ominus$) s pomocí již zmiňovaných složených strukturních elementů. Operátor $\odot$ odstraňuje drobné artefakty jako při použití $\otimes$, avšak díky erozi působí zároveň i na vyhlazení okrajů jednotlivých znaků a jejich ztenčování.

Vizuální porovnání navržených metod je možné posoudit na obrázku 3.12. Ve spodní části obrázku 3.12b je vidět, že zmizely i větší artefakty. Celkově se číslice a čáry ztenčily a to napomáhá jejich lepšímu oddělení.

(a) použití hit-or-miss ($\otimes$)



(b) použití navrženého operátoru $\odot$

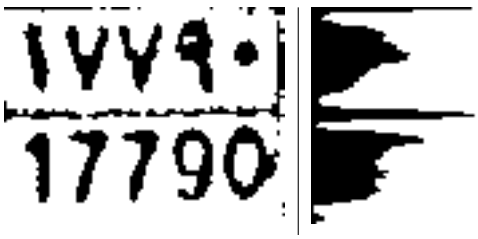
Obrázek 3.12: Porovnání metod pro filtrování artefaktů v oprahovaném obrázku po aplikaci všech tří složených strukturních elementů (obrázek 3.11)

Všechny předchozí kroky přispěly k tomu, že nyní je binární obrázek SPZ připraven pro segmentaci a rozpoznávání číslic.

3.4.2 Segmentace řádků

V dalším kroku je nutno určit rozsahy jednotlivých řádků textu. Pomocí *horizontální projekce* získáme histogram udávající počty nenulových pixelů pro každý řádek obrázku. Tento postup byl aplikován v článku [6].

Ze znalosti poměru stran SPZ jsme dokázali při počítání projekcí vynechat oblast barevného proužku a omezit se pouze na textovou část SPZ. Obrázek 3.13 ukazuje případ dvouřádkové SPZ, ale u jednořádkových je tomu podobně.



Obrázek 3.13: Horizontální projekce textové části SPZ — vpravo je histogram projekcí oddělený od obrázku svislou čarou

V histogramu horizontálních projekcí jsou vyhledávány úseky hodnot, které překračují pevně nastavenou mezní hodnotu. Tuto hodnotu jsme stanovili na základě testování. V našem případě však nebyla úloha tak jednoduchá, jak ji popisuje článek [6], protože pomocí pevné mezní hodnoty se řádky nedařilo dobře segmentovat.

Algoritmus hledání úseků jsme museli vylepšit o několik důležitých heuristik, aby byl opravdu použitelný. Nebudeme je jmenovat všechny, ale příkladem může být omezení hodnot v nalezeném úseku i shora kvůli vodorovným čárám v SPZ, spojování krátkých úseků dohromady, stanovení minimální a maximální délky úseku apod.

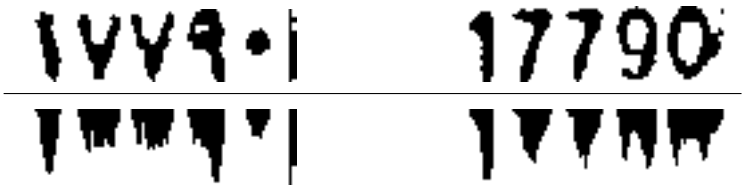
Pevná mezní hodnota nám v některých případech nevyhovovala. Proto jsme zavedli postup, při němž je vždy otestováno několik prahových hodnot. Ty jsou porovnány co do počtu a délky nalezených úseků tak, aby co nejlépe odpovídaly rozměrům jednotlivých řádků značky. Přibližnou výšku řádků totiž můžeme odhadnout díky tomu, že při mapování SPZ na obdélník (viz část o lokalizaci SPZ na str. 19) její výšku normujeme.

Trochu odlišně je potřeba přistupovat k dočasným a exportním SPZ, které nemají v horním řádku žádnou užitečnou informaci, a v případě dočasné SPZ navíc spodní řádek obsahuje emirátské číslice. Jelikož byl typ SPZ určen v předchozí fázi, znamená to jen ošetření speciálního případu a případné prohození řádků.

Rozsahy nalezených úseků řádků SPZ v binárním obrázku jsou předány do dalšího kroku.

3.4.3 Segmentace znaků

V tuto chvíli máme správně segmentovaný řádek emirátských číslic a zvlášť řádek latin číslic (platí i pro jednořádkovou SPZ). Se segmentací jednotlivých znaků v řádku je to obdobné. Tentokrát se ale počítají *vertikální projekce* (obrázek 3.14) přes každý segmentovaný řádek zvlášť.



(a) emirátský řádek (b) latin řádek

Obrázek 3.14: Vertikální projekce jednotlivých řádků SPZ — ve spodní části jsou histogramy projekcí oddělené od obrázku vodorovnou čárou, pro lepší ilustraci jsou otočeny vzhůru nohama

Protože předpokládáme, že jsme řádky segmentovali správně, můžeme pevně stanovit (na základě pokusů) spodní mez pro oprahování vertikálního histogramu a nalezení úseků odpovídajících jednotlivým číslicím. Jelikož dopředu nevíme, kolik číslic máme segmentovat (číselný údaj na SPZ má 3–6 číslic), nemůžeme použít stejnou metodu jako u řádků — tzn. zkoušet postupně měnit prahovou hodnotu a sledovat počet a velikost segmentovaných úseků.

Ne všechny úseky se dají považovat za číslice. Příkladem je poslední úsek na obrázku 3.14a, který neodpovídá žádné číslici. Z toho důvodu je potřeba kontrolovat délku úseků a příliš krátké úseky zamítnout, protože k rozdělení znaku vlivem oprahování téměř nedochází. Očekávaná šířka znaku opět vychází ze znalosti poměrů stran SPZ a její normalizace při mapování.

Filtrovali jsme také krátké okrajové úseky, jež přímo sousedí s okrajem SPZ. Často to bývají zbytky po šroubech nebo ohraničení SPZ. Delší okrajové úseky, které vznikají spojením písmene s hranicí SPZ, jsme ale zachovali. Náležitě ořízнuty jsou až v dalším kroku na základě párového znaku (viz dále).

Na konci tohoto kroku je každý znak v binárním obrázku ohraničen obdélníkem (tzv. *výřezový obdélník*), jehož rozměry určuje pozice řádků SPZ (z předchozího kroku) společně s nalezeným úsekem (příslušné číslice) v rámci řádku. Pozice obdélníku bude v dalších krocích zpřesněna.

3.4.4 Párování znaků

Mezní hodnota pro prahování v předchozí fázi většinou znaky správně rozdělila. Někdy se ale stalo, že vlivem šumu, nečistot a šroubů na SPZ došlo ke „spojení“ některých znaků — byly segmentovány jako jediný úsek.

Zvýšit mezní hodnotu, abychom tyto spojené znaky oddělili, jsme už nemohli. Zkrátili bychom totiž důležité úseky správně segmentovaných znaků. Platí to hlavně pro části znaků, které mají malé hodnoty vertikálních projekcí (např. latin číslice 1 nebo emirátská číslice 6).

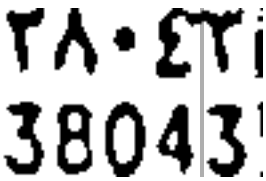
Pro rozdělování spojených znaků jsme navrhli *metodu párování znaků*. Snaží se využít relativní horizontální pozice dvojic latin a odpovídajícího emirátského znaku. Cílem je správné ohraničení znaku na vodorovné ose x.

Postupně procházíme latin a emirátské úseky a podle jejich překrytí v x-ové souřadnici vytváříme páry navzájem si odpovídajících úseků. Při počítání je nutné emirátské číslice jednořádkových SPZ „posunout“ na úroveň latin číslic odečtením vhodné konstanty (polovina šířky SPZ + polovina šířky barevného proužku).

Úseky považujeme za spárované, pokud se překrývají alespoň 40% své délky, přičemž se vždy bere poměr délky překrytí ku délce kratšího úseku. Hodnota 40% se nám během testů ukázala jako nejlepší.

Některé úseky tak nemusejí mít přiřazen párový úsek nebo je s nimi spojeno i více úseků. V ideálním případě, kdy máme bijektivní zobrazení mezi emirátskými a latin úseky, bychom byli v tomto okamžiku s párováním hotovi.

Jestliže jednomu úseku odpovídá po párování více úseků, je to signál k rozdělení delšího úseku na několik malých. Delší úsek není vhodné dělit např. na polovinu, ale lepší je využít pozic spárovaných úseků. Jelikož mohou být některé znaky dosti úzké, není výhodné dělit delší úsek podle konce prvního spárovaného úseku. Místo toho volíme za dělící čáru polovinu mezi koncem prvního spárovaného úseku a začátkem následujícího (obrázek 3.15).



Obrázek 3.15: Rozdělení spojených znaků na základě párových úseků — spojené jsou poslední dvě emirátské číslice 4 a 3

Může nastat situace, kdy jsou slepeny dva a dva odpovídající úseky. To se pozná podle abnormální délky obou úseků. Zde už většinou nepomůže nic jiného, než tyto úseky rozdělit v poměru jejich délky ku očekávané šířce

znaku. Počet menších úseků, na které mají být oba delší úseky rozděleny, se vždy zaokrouhluje dolů.

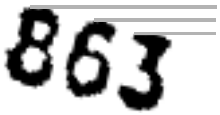
Tímto krokem se nám podařilo rozdělit všechny výřezové obdélníky, které obsahovaly více spojených číslic. Navíc jsme vytvořili dvojice navzájem si odpovídajících latin a emirátských číslic (resp. jejich výřezových obdélníků).

3.4.5 Ořezávání znaků

V této fázi segmentace zajistíme, aby byl ve vybraném výřezovém obdélníku pouze zkoumaný znak. Nejprve ořízneme ze všech stran prázdné řádky a sloupce, které se tam mohly objevit díky nestejné výšce znaků v řádku nebo dělení spojených znaků v předchozím kroku.

Článek [27] dále nabízí možnost odstranění šroubů a sklonu značky na základě pozice horní a dolní strany výřezového obdélníku pro všechny znaky na jednom řádku. Sklon není v našem případě nutné kompenzovat, pokud byly správně nalezeny rohové body SPZ (viz část o lokalizace SPZ na str. 14). Zmiňovanou metodu je však možné trochu upravit a aplikovat na ořezání šroubů.

Původní metoda vychází z toho, že y-ová pozice horní strany znaku by se měla lineárně měnit a tudíž by se měla nacházet mezi y-ovými souřadnicemi horních stran obou sousedů (obrázek 3.16). Podobně to platí i pro spodní stranu. Na SPZ z Abu Dhabi se však vyskytuje speciální znak (emirátská nula), který tuto hypotézu porušuje.



Obrázek 3.16: Linearita horní strany znaku — obrázek byl pro lepší názornost uměle vytvořen

Naše metoda (obrázek 3.17) vychází z této myšlenky a aplikuje ji na odstranění šroubů či zbytků vodorovných čar (ohraničení SPZ), které po adaptivním prahování zůstaly spojeny s některou číslicí. Vezmeme pouze dostatečně vysoké znaky (tím vynecháme emirátskou nulu) a spočítáme *medián*[2] z pozic horních stran znaků v jednom řádku. Stejně spočítáme i medián pro dolní strany.

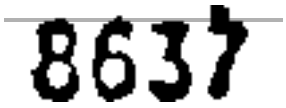
Pozici horní a dolní strany každého znaku upravíme na základě vzorců:

$$h \leftarrow \min\{h, \bar{h}\} \quad ,$$

[2]V setříděné posloupnosti je to prostřední prvek.

$$d \leftarrow \max\{d, \bar{d}\} \quad ,$$

kde h (resp. d) značí pozici horní (resp. dolní) strany znaku a $\bar{h}$ (resp. $\bar{d}$) je medián z horních (resp. dolních) stran. Oproti průměru těchto hodnot se medián vyznačuje svou odolností vůči malému počtu hodnot, které se od ostatních hodně liší.



Obrázek 3.17: Metoda odstranění šroubů pomocí mediánu — šroub se v tomto případě nachází v horní části latin číslice 7, ale díky okolním číslicím se ho podařilo správně uříznout

Předpokládáme, že dobrých znaků je na SPZ více než těch, které je potřeba takto ořezat. Stačí tedy alespoň polovina dobrých znaků, aby mohly být ostatní správně ořezány. Jestliže máme jen málo (méně než polovinu) vysokých znaků, toto ořezávání se neprovádí.

Se znakem emirátské nuly je potřeba zacházet opatrně, protože se nápadně podobá šroubům nebo malým nečistotám. Abychom dokázali eliminovat ty nepravé, testujeme, zda daným znakem prochází střed řádku (obrázek 3.18). Pokud ano, jedná se o emirátskou číslici 0. V opačném případě můžeme znak zavrhnout. První úsek na obrázku 3.18 by byl normálně klasifikován také jako emirátská nula, ale daným „znakem“ neprochází střed řádku, a tak je ignorován.

Obrázek 3.18: Kontrola emirátské nuly pomocí středu řádku — jedná se o číselný údaj 8205 zapsaný pomocí emirátských číslic (emirátská nula je na třetím místě), první a poslední úsek je falešný a neodpovídá žádné číslici

Na základě upravené pozice a rozměrů výřezových obdélníků můžeme vytvořit samostatné binární obrázky jednotlivých číslic. Tím je segmentační proces ukončen a obrázky číslic (seřazené do dvojic pro odpovídající latin a emirátské číslice) jsou předány dále k rozpoznávání.

3.5 Rozpoznávání číslic

Při rozpoznávání nám jde o to, abychom dokázali identifikovat předložený objekt na základě provedených pozorování. V našem případě se jedná o ob-

rázek číslice, který máme zařadit do odpovídající *klasifikační třídy*. Znamená to tedy vybrat prvek z množiny tříd $C = \{0, 1, 2, \ldots, 9\}$. Chceme-li vyřešit tento rozhodovací problém, musíme zkonstruovat příslušný klasifikátor. Označme si navíc $c = |C| = 10$ jako velikost množiny C.

Obrázky číslic musíme nejdříve normalizovat na stejnou velikost a poté vybereme vhodné charakteristiky pro jejich popis. Sestrojíme klasifikátor, který podle předloženého popisu dokáže příslušnou číslici rozpoznat. Na závěr se budeme zabývat ověřením věrohodnosti výsledku klasifikace.

3.5.1 Normalizace

Aby byla klasifikace dostatečně odolná vůči různým velikostem znaků, je potřeba obrázky znaků příslušně roztáhnout na dohodnutou velikost. Vzhledem k obvyklým rozměrům znaků na fotografiích jsme zvolili rozměry 30×30 pixelů. Tyto rozměry budeme dále označovat jako tzv. *normalizační čtverec*.

Zatímco výšku znaku máme danou z mapovací fáze, šířka znaku může být odlišná pro různé znaky. Jednou z možností je roztáhnout číslici tak, aby se dotýkala okrajů normalizačního čtverce (podle článku [6]). U znaků, které bylo možno dříve odlišit na základě jejich šířky (např. latin 1), se tato výhoda ztrácí, poněvadž po roztáhnutí jsou všechny znaky stejně široké (obrázek 3.19b).

Pro náš problém se ukázalo lepší přizpůsobovat pouze výšku znaku a šířku dopočítat se zachováním původního poměru stran. Následně je znak umístěn do středu normalizačního čtverce (obrázek 3.19c). Tento přístup přispívá k lepšímu odlišení jednotlivých znaků tím, že obrázky odlišných číslic mají méně společných pixelů.

(a) (b) (c)

Obrázek 3.19: Porovnání metod normalizace znaku — (a) segmentovaný znak uprostřed normalizačního čtverce, (b) normalizovaný znak po roztáhnutí v obou souřadnicích, (c) normalizovaný znak po přizpůsobení výšky

Jak již bylo zmíněno dříve, emirátská nula tvoří samostatný případ a tudíž se nenormalizuje. Rozpoznává se na základě své výšky, která ji výrazně odlišuje od ostatních číslic. Byl-li znak v této fázi identifikován jako emirátská nula, do dalších fází nepostupuje a je označen jako rozpoznaný.

Všechny předložené binární obrázky číslic byly v tomto kroce normalizovány na stejnou velikost 30×30 pixelů (s výjimkou rozpoznaných obrázků emirátských nul).

3.5.2 Výběr příznaků

Nyní je nutné pro popis binárních obrázků zvolit pevnou množinu měřených veličin, které budeme nazývat *příznaky*. Hodnoty příznaků se získávají z předloženého obrázku normalizovaného znaku. Zapsány do posloupnosti tvoří tzv. *příznakový vektor*. Počet složek tohoto vektoru určuje dimenzi n *příznakového prostoru* P.

Zvolíme-li příliš malou množinu příznaků, nemusí být dostatečná pro dobré odlišení rozpoznávaných objektů (v našem případě číslic). Postupným přidáváním příznaků dosahujeme obvykle lepšího oddělení.

Důležité je vybrat takovou množinu příznaků, které daný objekt co nejlépe popisují a přitom ho dobře odlišují od ostatních. Preferují se příznaky odolné vůči šumu nebo očekávaným prostorovým transformacím. Výhodou také je, pokud se dají snadno a rychle spočítat.

Jednotlivé příznaky by v ideálním případě neměly být vzájemně korelované — hodnota jednoho příznaku by neměla záviset na hodnotě jiného. U některých metod, které příznakový vektor zpracovávají sekvenčně, se však závislost dvou po sobě jdoucích pozorování přímo předpokládá a je tudíž žádoucí (např. Markovovy modely).

Normalizace příznaků je velice důležitá a ovlivňuje pozdější rozpoznávání. Rozsahy hodnot jednotlivých složek příznakového vektoru by měly být relativně stejné. Složky příznaku, které mají vysokou hodnotu, se učí klasifikátor většinou rychleji a učení ostatních položek „zanedbává“. Příslušným škálováním vstupu se tohoto nepříjemného efektu zbavíme.

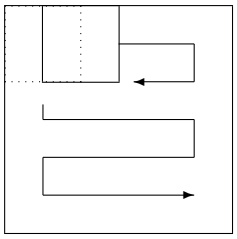
Pro porovnání jsme vybrali několik typů příznaků (klíčové slovo v závorce slouží pro snadnější odkazování na konkrétní typ příznaku). Na základě výsledků rozpoznávací fáze se rozhodneme, jaké typy příznaků je vhodné použít pro klasifikaci latin a emirátských číslic — pro latin číslice můžeme vzít jiný typ příznaků než pro emirátské číslice.

Jednotlivé pixely (*Pix*) : Samotné hodnoty jednotlivých pixelů zapsané po řádcích mohou tvořit binární příznakový vektor. V tomto případě je n rovno $30 \cdot 30 = 900$ (rozměry normalizačního čtverce). Jak je vidět, dimenze příznakového prostoru je velmi velká.

Abychom zmenšili počet příznaků, zkusili jsme vybrat jen některé pixely, u kterých se bude zjišťovat jejich hodnota. Pro výběr si vytvoříme masku oblastí, které jsou důležité pro dobré odlišení číslic. Prakticky se tato myšlenka využít nedala, protože už jen jemně posunutá číslice způsobila to, že se důležité oblasti dostaly mimo definovanou masku.

Posuvné okno (*Win*) : Příznakový vektor, který byl zvolen v článku [6], poskytuje lepší odolnost vůči šumu. Obrázek je odshora postupně procházen oknem 6×6 pixelů (velikost okna z článku [6] jsme uzpůsobili velikosti našeho

normalizovaného znaku) střídavě zleva doprava a zprava doleva (obrázek 3.20). Sousední pozice okna se z 50% překrývají (třemi sloupci nebo třemi řádky v okně).



Obrázek 3.20: Postup při získávání příznakového vektoru typu *Win* — tečkovaně je vyznačena první pozice okna a plnou čárou následující, aby bylo vidět jejich překrytí

V každé pozici se spočítá poměr nenulových pixelů v okně ku ploše okna (tedy $6 \cdot 6 = 36$). Různých pozic (a tedy i hodnot v příznakovém vektoru) je celkem:

$$\left(\frac{30}{0,5 \cdot 6} - 1\right)\left(\frac{30}{0,5 \cdot 6} - 1\right) = 81 \quad .$$

Střídavé procházení zleva doprava a naopak má v tomto případě význam pro zachování „spojitosti“ měřených hodnot.

Maska důležitých oblastí (*Mask*) : Napadlo nás, že pro obrázek normalizovaného znaku definujeme masku důležitých oblastí (obrázek 3.21). Obrázek se prochází oknem 4×4 s 50% překrytím pozic (podobně jako u příznaků *Win*). Postupně se zaznamenávají hodnoty poměru nenulových pixelů v okně ku ploše okna (tentokrát $4 \cdot 4 = 16$). Pozice okna mimo zvolenou masku se ignorují.

Obrázek 3.21: Maska důležitých oblastí pro počítání příznaků

Masku jsme odvodili z vizuálního vzhledu jednotlivých znaků a jejich vzájemné podobnosti. Protože právě místa, která odlišují velmi podobné znaky, jsou nejdůležitější. Pro rozlišení emirátské číslice 2 a 3 je například velmi důležitá horní část obrázku. Jiné části jsou důležité pro latin číslice 5, 6, 8 a 9. Zvolili jsme proto masku, která „kopíruje“ tyto důležité oblasti.

Projekce (*Proj*) : Horizontální a vertikální projekce obrázku jsme použili při segmentaci. Zjistili jsme, že se dají použít i jako příznaky při rozpoznávání. Všechny projekce (oba typy) normalizovaného znaku jsou zapsány do jednoho vektoru za sebe (30+30 = 60 hodnot). Hodnoty horizontálních (resp. vertikálních) projekcí dělíme šířkou (resp. výškou) obrázku.

Normalizované centrální momenty (*NCM*) : Poněkud sofistikovanějšími příznaky jsou *momenty* [28, str. 259]. Popisují obrázek jako celek, ale mohou být nasazeny i lokálně pro různé části obrázku zvlášť. Pro jejich výpočet platí:

$$m_{ij} = \sum_x \sum_y x^i y^j I(x,y) \text{ pro } i,j \geq 0 \quad ,$$

kde x (resp. y) jde přes celou šířku (resp. výšku) obrázku a $I(x,y)$ značí hodnotu pixelu v daném bodě. Součet indexů $i+j$ se nazývá *řád* nebo *stupeň* momentu.

Aby byly invariantní vzhledem k běžným prostorovým transformacím (např. posunutí), používají se tzv. *normalizované centrální momenty* [28, str. 260]:

$$\tilde{m}_{ij} = \frac{\sum_x \sum_y (x - x_t)^i (y - y_t)^j I(x,y)}{m_{00}^{\frac{i+j}{2}+1}} \quad ,$$

kde $x_t = \frac{m_{10}}{m_{00}}$ a $y_t = \frac{m_{01}}{m_{00}}$ představují těžiště objektu na obrázku. Díky x_t a y_t jsou momenty $\tilde{m}_{ij}$ nezávislé na posunutí obrázku. Celý výraz je dělen vhodnou mocninou momentu m_{00} (bývá interpretován jako plocha objektu).

Momenty tvoří dobrou charakteristiku binárního obrázku. S rostoucím stupněm $(i+j)$ jsou však stále více ovlivňovány šumem. V případě normalizovaných centrálních momentů se prvky $\tilde{m}_{00}$, $\tilde{m}_{10}$ a $\tilde{m}_{01}$ vynechávají, protože jejich hodnota je pro všechny obrázky přibližně stejná.

Diskrétní kosinová transformace (*DCT*) : Výsledek jednoho měření nemusí být vždy jen jediná hodnota, ale i celý vektor hodnot. Takže by se dalo hovořit spíše o „příznakové matici“. Právě tento přístup se často uplatňuje při rozpoznávání pomocí skrytých Markovových modelů (str. 62).

Příkladem mohou být koeficienty *diskrétní kosinové transformace* [29, str. 37] získané z malých výřezů obrázku. V úvahu se berou pouze významnější koeficienty a ostatní se zanedbávají. Přestože je tento texturální přístup v článku [18] aplikován na rozpoznávání obličejů, pokusíme se ho využít i pro rozpoznávání číslic.

3.5.3 Rozpoznávání

Pro každý normalizovaný binární obrázek číslice máme spočtený příznakový vektor, podle kterého musíme určit, jaká číslice je ve skutečnosti na obrázku. Úkol zkonstruovat klasifikátor se „zužuje“ na nalezení vhodného zobrazení $f : P \to C$, kde P je příznakový prostor a $C = \{0, 1, 2, \ldots, 9\}$.

Speciální případ tvoří emirátská nula, která se do této fáze vůbec nedostane. Nerozpoznává se na základě příznaků, ale díky své odlišné velikosti v předchozích fázích. Emirátský klasifikátor se vůbec neučí nulu rozpoznávat.

Rozlišit mezi latin a emirátskou číslicí dokážeme na základě její pozice v rámci SPZ. Pro oba typy číslic budeme hledat vlastní klasifikátor samostatně (pro jeden typ číslic se může lépe hodit jiný typ příznaků a jiná metoda rozpoznávání než pro druhý). Ve zbytku této části (sekce 3.5.3) se zaměříme hlavně na latin číslice, ale obdobné postupy jsme aplikovali i na emirátské číslice.

Budeme potřebovat data pro učení klasifikátoru. Nejprve si připravíme dostatečné množství segmentovaných a normalizovaných obrázků jednotlivých číslic a ručně jim přiřadíme číslo třídy z C, které jednoznačně identifikuje každou číslici. Obrázky máme tedy rozděleny do skupin. Je důležité mít ve všech skupinách přiměřeně stejný počet obrázků, aby se klasifikátor naučil dobře rozpoznávat všechny číslice.

Pro každý obrázek spočítáme jeho příznakový vektor x (tzv. *vzor* x). X_i potom bude množina všech příznakových vektorů x, které jsme získali z obrázků ve stejné skupině (s indexem i). Posloupnost všech $(X_i)_{i=0}^{9}$ budeme nazývat *trénovací množinou* a označíme si $X = \bigcup X_i$.

Pro latin i emirátské číslice jsme vytvořili dvě samostatné trénovací množiny (jsou to dva oddělené problémy). V prvním případě byla trénovací množina získána z obrázků 352 latin číslic (ve druhém případě z 280 emirátských). Bude-li v dalším textu použit pojem trénovací množina (nebo X), myslí se tím právě jedna z nich. Většinou se jedná o libovolnou z nich, pokud to přímo nevyplývá z kontextu.

Některé metody využívají trénovací množinu pro učení klasifikátoru, jiné z ní pouze vyberou vhodné reprezentanty. V našem případě se omezíme na trénování s učitelem.

Je výhodné část vzorů z trénovací množiny oddělit a vytvořit z nich tzv. *ověřovací množinu*. Ta bude sloužit pro kontrolu toho, jak dobře už se klasifikátor naučil danou úlohu řešit. Trénovací množina není v tomto případě směrodatná, protože si ji klasifikátor většinou téměř bezchybně „zapamatuje“. Lepší je použít vzory, které nebyly během trénování vůbec předloženy.

Některý metodám (např. neuronovým sítím) slouží ověřovací množina pro kontrolu již během učící fáze klasifikátoru. Náhlý vzrůst chyby na ověřovací množině je dobrou ukončovací podmínkou pro proces učení.

Nakonec jsme vytvořili ještě jednu speciální množinu, do které jsme zařadili všechny opravdu špatně čitelné znaky (vlivem nepřesné segmentace nebo nevhodného oprahování). Budeme ji dále označovat jako *problémovou množinu*. Na té můžeme lépe porovnat schopnosti nejlepších metod.

Jen pro srovnání uvádíme, že náhodná klasifikace zajišťuje 10% (pro 10 tříd) správně rozpoznaných číslic. Procento správně klasifikovaných SPZ je v tomto případě velmi nízké. „Chytřejší“ metody rozpoznávání jsou rozebrány v následujících několika kapitolách.

3.5.4 Věrohodnost

Z pravidel pro dobrý systém rozpoznávání SPZ (nerovnice 2.4) vyplývá, že velký počet chybně klasifikovaných SPZ je nežádoucí. Musíme tedy co nejvíce snížit procento špatně rozpoznaných číslic a maximálně při tom využít redundantní informace, kterou v sobě SPZ obsahuje (každá číslice je na SPZ obvykle zapsána jako latin i emirátská).

Jak je z tabulky 2.1 (str. 4) vidět, jsou si některé číslice dosti podobné (např. emirátská 2 a 3). Nepodaří-li se nám od sebe odlišit dvě emirátské číslice, může se nám to podařit u latin číslic.

Nejdříve se zaměříme na věrohodnost rozpoznání vyplývající z vizuálního vzhledu jednotlivých latin a emirátských číslic a poté si popíšeme postup, jak ohodnotit kvalitu segmentace znaku.

Věrohodnost rozpoznání : Na základě podobnosti znaků je možné přiřadit jednotlivým latin a emirátským číslicím tzv. *vizuální váhy*. Přičemž větší hodnotu přiřadíme tomu typu, který je vizuálně lépe odlišitelný od ostatních. U některých číslic budeme více důvěřovat emirátskému klasifikátoru a u některých naopak klasifikátoru pro latin číslice. Váha musí zvýhodňovat daný typ jen mírně, protože i ten může být díky šumu špatně rozpoznán.

Vizuální váhy definujeme jako vektor čísel $\sigma_0, \ldots, \sigma_9$ z intervalu $\langle 0, 1 \rangle$. Dále budeme předpokládat, že latin číslicím vždy náleží váha σ_i a emirátským váha $1 - \sigma_i$. Hodnoty vah (tabulka 3.3) jsme určili experimentálně.

číslice	0	1	2	3	4	5	6	7	8	9
latin (%)	50	55	55	50	52	50	50	48	50	50
emirátské (%)	50	45	45	50	48	50	50	52	50	50

Tabulka 3.3: Vizuální váhy mezi latin a emirátskými číslicemi — váhy jsou pro lepší názornost udávány v procentech

Znaky latin se obecně ukazují jako lépe rozlišitelné. Emirátský řádek bývá totiž často poškozen vrženým stínem a znaky jsou tudíž „slepeny“.

Věrohodnost segmentace : Je nezbytné se zabývat také věrohodností segmentace, protože nepřesná nebo úplně chybná segmentace (spojené znaky jsou například rozděleny v nevhodném místě) bývá nejčastěji hlavní příčinou špatného rozpoznání znaků.

Vymysleli jsme způsob, jak ohodnotit, nakolik se segmentace znaku zdařila. Algoritmus vždy prozkoumá okrajové části normalizovaného znaku (tzv. *kontrolní oblasti*) a hledá v nich nenulové pixely. Na obrázku 3.22a jsou kontrolní oblasti označeny šedou barvou.

(a) (b) (c)

Obrázek 3.22: Ověření věrohodnosti segmentace — (a) kontrolní oblasti, (b) správně segmentovaný znak, (c) špatně segmentovaný znak

Šířku kontrolních oblastí jsme nastavili tak, aby do nich správně se segmentovaný znak téměř (nebo vůbec) nezasahoval (obrázek 3.22b). U špatně segmentovaného znaku tyto oblasti obsahují hodně nenulových pixelů (obrázek 3.22c).

Podle toho, jak jsou kontrolní oblasti zaplněny, algoritmus stanoví váhu (tzv. *segmentační věrohodnost* κ) mezi emirátským a latin znakem v rozpoznávané dvojici (odpovídajících si číslic). Znak s větší váhou je považován za věrohodnější a tudíž je lepší se více přiklonit k výsledku jeho klasifikace.

Pro κ jsme odvodili vzorec:

$$\kappa = \mathrm{sgn}(k_{em} - k_{lat}) \cdot \frac{|k_{em} - k_{lat}|}{K} \quad , \tag{3.2}$$

kde k_{em} (resp. k_{lat}) je počet nenulových pixelů v kontrolních oblastech u emirátské (resp. latin) číslice a K je plocha kontrolních oblastí.

Vlivem funkce sgn se hodnota proměnné κ pohybuje v intervalu $\langle 0, 1\rangle$ pro $k_{em} \geq k_{lat}$ a v intervalu $\langle -1, 0\rangle$ pro $k_{em} \leq k_{lat}$. Proměnná κ musí být použita tak, aby byla přičtena k věrohodnosti latin číslice a odečtena od věrohodnosti emirátské číslice.

Zvýhodněny jsou tedy znaky, které mají nejméně nenulových pixelů v kontrolních oblastech. Je jim dána o to větší váha, o kolik více jsou zaplněny kontrolní oblasti druhého znaku. Použití této váhy závisí na konkrétní zvolené metodě rozpoznávání a bude popsáno v následujících kapitolách.

4 Klasifikátor minimální vzdálenosti

4.1 Teorie

Jedním ze způsobů, jak klasifikovat vzor do odpovídající třídy, je *pravidlo nejbližšího souseda*. Jedná se o velmi přímočarou metodu, která je poměrně snadná na implementaci. V některých případech poskytuje celkem uspokojivé výsledky, které nám mohou posloužit pro srovnání s ostatními metodami rozpoznávání.

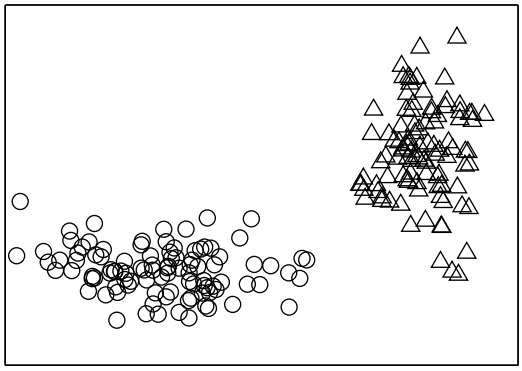
Protože v průběhu klasifikace nedochází k žádnému zlepšování výsledků, je tato metoda označována jako tzv. *bez učení*. Pro každou třídu existuje jeden nebo více *reprezentantů*, kteří byli vybráni tak, aby ji co nejlépe charakterizovali. Existují různé přístupy uplatňující různé strategie pro výběr reprezentantů. Nejčastěji jsou reprezentanti voleni přímo z trénovací množiny, ale nemusí to být pravidlem. Množinu reprezentantů pro třídu i si označme Y_i, potom $Y = \bigcup Y_i$.

Při klasifikaci neznámého vzoru $\mathbf{x}$ se spočte jeho vzdálenost od všech reprezentantů ze všech tříd. Vzor je pak zařazen do třídy, která odpovídá nejbližšímu reprezentantovi (resp. reprezentantům). Vhodná metrika může výsledek klasifikace výrazně zlepšit.

4.1.1 Template matching

Vzory patřící do stejné třídy se chovají se jako náhodná veličina (vektor) a obvykle tvoří v příznakovém prostoru tzv. *shluky* (obrázek 4.1). Pokud neznáme pravděpodobnostní rozdělení jednotlivých shluků, předpokládáme [28, str. 304], že se dají popsat normálním rozdělením. Pokud se ovšem předpoklad normálního rozdělení vzorů ukáže jako chybný, pak nemusí tato metoda dávat dobré výsledky.

Jako vhodného reprezentanta $\mathbf{y_i}$ pro každý shluk použijeme střední hodnotu $\boldsymbol{\mu_i}$ vzorů z odpovídající třídy i. S dostatečně velkou trénovací množinou je možné aproximovat střední hodnotu $\boldsymbol{\mu_i}$ pomocí aritmetického průměru



Obrázek 4.1: Shluky vzorů v příznakovém prostoru (ilustrační obrázek) — vzory ze dvou různých tříd (označeny symboly △ a ◯) tvoří v příznakovém prostoru dva dobře oddělené shluky

vzorů z dané třídy [4]. Reprezentant $\mathbf{y_i}$ v tomto případě představuje těžiště příslušného shluku v příznakovém prostoru a platí:

$$\mathbf{y_i} = \frac{1}{|X_i|} \sum_{\mathbf{x} \in X_i} \mathbf{x} \quad ,$$

kde vzory $\mathbf{x}$ vybíráme z množiny X_i (část trénovacích dat odpovídající třídě i) a $|X_i|$ je velikost množiny X_i.

Mezi nejznámější metriky patří *euklidovská vzdálenost*. Pro dva n-dimenzionální příznakové vektory $\mathbf{x} = (x_1, \ldots, x_n)$ a $\mathbf{y} = (y_1, \ldots, y_n)$ je euklidovská vzdálenost definována:

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \quad . \tag{4.1}$$

Na základě této metriky je nalezen nejbližší reprezentant $\mathbf{y_i} \in Y_i$, pro kterého platí:

$$\|\mathbf{x} - \mathbf{y_i}\| \leq \|\mathbf{x} - \mathbf{y}\| \text{ pro všechna } \mathbf{y} \in Y \quad ,$$

a neznámý vzor $\mathbf{x}$ je klasifikován do třídy i. Tento postup je v knize [4, str. 39] označován jako *template matching*.

Protože odmocnina je rostoucí funkce, nemá vliv na porovnávání vzdáleností a je možné ji vynechat. Rovnice 4.1 se tím zredukuje na skalární součin $(\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y})$, což vede k urychlení výpočtu.

Pokud je klasifikačních tříd (a tudíž i reprezentantů) mnoho, lze použít ještě jednu heuristiku. Není vždy nutné dopočítávat již zmíněný skalární součin $(\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y})$ až do konce. Pokud již nasčítaná hodnota překročila vzdálenost dosud nejbližšího nalezeného reprezentanta, je výpočet skalárního součinu zastaven.

4.1.2 Mahalanobisova vzdálenost

Kromě střední hodnoty shluků můžeme brát v úvahu také jejich rozptyl, který nemusí být ve všech dimenzích příznakového prostoru stejný. Jsou případy, kdy shluk může tvořit ve dvojrozměrném prostoru například eliptický útvar (obrázek 4.1).

Je nezbytné odpovídajícím způsobem upravit hodnotící metriku vzdálenosti vzoru $\mathbf{x}$ od těžiště shluku i:

$$M(\mathbf{x}, \mathbf{y_i}) = (\mathbf{x} - \mathbf{y_i})^T \hat{\mathbf{\Sigma}}_\mathbf{i}^{-1} (\mathbf{x} - \mathbf{y_i}) \quad ,$$

kde $\hat{\mathbf{\Sigma}}_\mathbf{i}$ je odhad kovarianční matice $\mathbf{\Sigma_i}$ (zastupuje rozptyl shluku i) a $\hat{\mathbf{\Sigma}}_\mathbf{i}^{-1}$ je označení pro inverzní matici k matici $\hat{\mathbf{\Sigma}}_\mathbf{i}$. Podobně jako v metodě template matching jsou reprezentanti $\mathbf{y_i}$ voleni jako těžiště shluků v trénovací množině. Takto zadefinovaná metrika $M(\cdot, \cdot)$ je označována jako *Mahalanobisova vzdálenost*.

Odhad kovarianční matice $\hat{\Sigma}_i$ získáme pro každý shluk na základě vzorů z příslušné třídy i trénovací množiny ze vzorce (viz [4, str. 88–89]):

$$\hat{\mathbf{\Sigma}}_\mathbf{i} = \frac{1}{|X_i|} \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \mathbf{y_i})(\mathbf{x} - \mathbf{y_i})^T \quad .$$

Jak je vidět, výsledná matice $\hat{\mathbf{\Sigma}}_\mathbf{i}$ je symetrická, a tudíž skoro polovina prvků nemusí být počítána podle vzorce.

Mahalanobisova vzdálenost pracuje s inverzní maticí k matici $\hat{\mathbf{\Sigma}}_\mathbf{i}$. Protože matice $\mathbf{\Sigma}$ může být obecně singulární, je možné spočítat tzv. *pseudoinverzní matici* [15, str. 221] pomocí numerické metody *SVD*[3] [21].

Pokud dosadíme za všechny $\hat{\mathbf{\Sigma}}_\mathbf{i}$ jednotkovou matici (to odpovídá jednotkovému rozptylu shluku i ve všech směrech), dostáváme euklidovskou vzdálenost jako v metodě template matching.

4.1.3 Pravidlo k nejbližších sousedů

V předchozích případech byl reprezentant pouze „fiktivní“ — tzn. nebyl součástí trénovací množiny. Nyní budeme volit za reprezentanty prvky z trénovací množiny.

Pro každou třídu může být z trénovací množiny vybráno i více reprezentantů. Jejich počet může záviset na velikosti celé trénovací množiny, apriorní pravděpodobnosti dané třídy nebo procentuálním zastoupení vzorů z dané třídy v trénovací množině. Typicky se volí stejný počet reprezentantů pro všechny třídy. Vybrány by měly být ty vzory, které danou třídu nejlépe charakterizují.

[3] z angl. *Singular Value Decomposition*

Pro klasifikaci neznámého vzoru $\mathbf{x}$ je nutné spočítat jeho vzdálenost od všech reprezentantů v příznakovém prostoru. V této souvislosti se nejčastěji uplatňuje již zmíněná euklidovská metrika.

Existují dva různé postupy, jak klasifikovat vzor $\mathbf{x}$ do některé z c tříd:

1. Najdeme k nejbližších reprezentantů a zjistíme, do které třídy patří. Neznámý vzor $\mathbf{x}$ je klasifikován do třídy, která je nejvíce zastoupena mezi k nejbližšími reprezentanty.

Algoritmus 4.1 Algoritmus k nejbližších sousedů (varianta 1)

```
1: spočítat vzdálenosti všech reprezentantů y od neznámého vzoru x
2: setřídit reprezentanty vzestupně podle vzdálenosti: y(1), y(2), ...
3: inicializace: p_i ← 0 pro i = 1, ..., c
4: for i ← 1, ..., k do
5:    p_j ← p_j + 1 pro y(i) ∈ Y_j
6: end for
7: return argmax_j {p_j}
```

2. Procházíme postupně reprezentanty od nejbližšího k nejvzdálenějšímu, dokud není nalezeno k reprezentantů ze stejné třídy. Do této třídy klasifikujeme i neznámý vzor $\mathbf{x}$.

Algoritmus 4.2 Algoritmus k nejbližších sousedů (varianta 2)

```
1: spočítat vzdálenosti všech reprezentantů y od neznámého vzoru x
2: setřídit reprezentanty vzestupně podle vzdálenosti: y(1), y(2), ...
3: inicializace: p_i ← 0 pro i = 1, ..., c
4: i ← 1
5: T ← −1
6: while T = −1 do
7:    p_j ← p_j + 1 pro y(i) ∈ Y_j
8:    if p_j = k then
9:       T ← j                          /* klasifikuj do třídy j */
10:   end if
11:   i ← i + 1
12: end while
13: return T
```

Pomocná proměnná p_j pro oba algoritmy udává, kolik z doposud prozkoumaných reprezentantů bylo z třídy j. Obě zmíněná pravidla dávají obdobné výsledky, ale přesto mezi nimi existují drobné rozdíly.

V prvním případě se může stát, že algoritmus nedokáže rozhodnout mezi dvěma stejně zastoupenými třídami. V algoritmu 4.1 je možné uplatnit podobnou heuristiku při počítání euklidovské vzdálenosti vzoru $\mathbf{x}$ od reprezentanta $\mathbf{y}$ (tedy skalárního součinu $(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})$) jako u metody template matching. Výpočet skalárního součinu se dá ukončit ve chvíli, kdy nasčítaná hodnota přesáhla vzdálenost k dosud nejbližších reprezentantů (resp. k-tého z nich).

Pro algoritmus 4.2 je nutné volit jen dosti malé k, které nesmí přesáhnout nejmenší počet reprezentantů v některé ze tříd. Problém nerozhodnutelnosti mezi třídami zde nenastává. V nejhorším případě se musí prozkoumat až $(k-1)\cdot c\cdot N+1$ nejbližších reprezentantů, kde c je počet tříd a N je dimenze příznakového prostoru.

Parametr k je nutné volit s ohledem na počet reprezentantů v každé třídě. Hodnota bývá obvykle dosti malá. Nejvhodnější hodnotu je dobré určit experimentálně, aby co nejlépe vyhovovala zadané úloze.

Při vyhledávání nejbližších sousedů mohou výpočet značně urychlit dobře navržené datové struktury (např. *quadtree*), ale tím se v této práci nebudeme hlouběji zabývat. Další informace lze nalézt v [5, 26, 25].

Pomocné proměnné p_j jsme se rozhodli použít pro ověření věrohodnosti rozpoznané číslice klasifikované do třídy i. Stanovili jsme podmínku (tzv. *základní test věrohodnosti*), podle které musí být hodnota p_i větší než všechny ostatní p_j. Pro tzv. *přísnější test věrohodnosti* navíc požadujeme, aby se hodnota p_i lišila od ostatních p_j alespoň o 2:

$$p_i - p_j \geq 2 \text{ pro všechna } j \neq i \quad . \tag{4.2}$$

4.2 Výsledky

Pro každou číslici jsme nejdříve ze souboru obrázků segmentovaných znaků vybrali 5 reprezentantů. Snažili jsme se volit jak velmi „pěkné“ (dobře segmentované) znaky, tak i ty, které obsahovaly malé nepřesnosti charakteristické pro danou skupinu číslic. Pro těchto 50 latin a 45 (vyloučena emirátská nula) emirátských znaků jsme si předpočítali hodnoty jejich příznakových vektorů, aby byly připraveny pro okamžité použití.

Vzdálenost dvou obrázků číslic, které se liší jen posunutím, může být v příznakovém prostoru velká. Může být dokonce větší než vzdálenost obrázků dvou úplně odlišných číslic. Klasifikátor minimální vzdálenosti pak takové číslice klasifikuje špatně. Proto jsme pro testování vybírali příznaky, které jsou vůči posunutí částečně odolné.

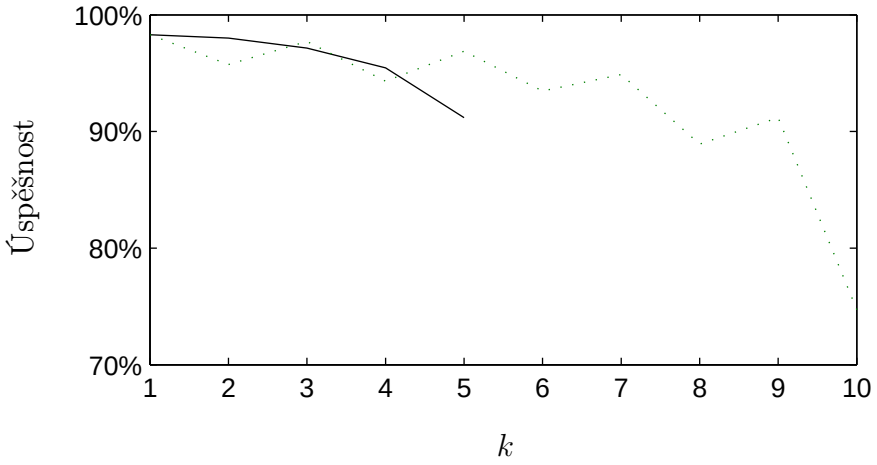
Postupně jsme testovali všechny nastíněné varianty klasifikátoru minimální vzdálenosti na různých typech příznaků a s různými volitelnými parametry. Pokud nebude uvedeno jinak, prováděli jsme pouze základní test věrohodnosti.

4.2.1 Pravidlo k nejbližších sousedů

Volitelným parametrem této metody je číslo k, jehož význam se u obou algoritmů (algoritmus 4.1 nebo 4.2) liší. Speciálně pro k rovno 1 se jedná o metodu template matching.

Nejprve jsme experimentovali s příznaky typu *Win*. V případě algoritmu 4.2 mohly být testovány pouze hodnoty $k \leq 5$ (k dispozici je pouze 5 reprezentantů pro jednu třídu).

Je zajímavé, že pro rostoucí k měla úspěšnost rozpoznání spíše klesající charakter (obrázek 4.2). Nejlepší výsledky dala metoda template matching (pro $k = 1$). Pro algoritmus 4.1 byl pokles správně klasifikovaných vzorů pomalejší než pro algoritmus 4.2.



Obrázek 4.2: Porovnání úspěšnosti algoritmů 4.1 a 4.2 pro metodu k nejbližších sousedů pro latin číslice na trénovací množině s příznaky *Win* — přerušovanou (resp. plnou) čárou je vyznačena úspěšnost pro algoritmus 4.1 (resp. algoritmus 4.2)

Křivka úspěšnosti algoritmu 4.1 pravidelně kolísala s lepšími výsledky pro liché hodnoty k. Stejně tak docházelo k nárůstu (resp. poklesu) procenta nerozpoznaných číslic pro sudé (resp. liché) hodnoty k. Tyto výkyvy byly způsobeny nerozhodností algoritmu 4.1 v případě stejně zastoupených tříd mezi k nejbližšími sousedy.

Výsledky testování na problémové množině (tabulka 4.1) opět hodně kolísaly v procentu nerozpoznaných znaků. Algoritmus 4.1 byl lepší v otázce věrohodnosti výsledku — měl lepší poměr počtu špatně rozpoznaných ku počtu nerozpoznaných znaků.

Zavedením přísnějšího testu věrohodnosti (vzorec 4.2) se poměr nerozpoznaných a špatně rozpoznaných znaků výrazně zlepšil (ovšem na úkor správně

	latin			emirátské		
k	+	?	−	+	?	−
1	81%	0%	19%	87%	0%	13%
2	63%	28%	9%	79%	14%	7%
3	76%	4%	20%	82%	2%	16%
4	69%	17%	14%	79%	5%	16%
5	82%	6%	12%	78%	3%	19%

Tabulka 4.1: Výsledky metody k nejbližších sousedů (algoritmus 4.1) na problémové množině při použití *Win* příznaků (+ správně, − špatně a ? nerozpoznané číslice)

rozpoznaných znaků). Přísnější test věrohodnosti (tabulka 4.2) se výrazněji projevuje až pro větší k (v tomto případě $k \geq 3$).

	latin			emirátské		
k	+	?	−	+	?	−
1	81%	0%	19%	87%	0%	13%
2	63%	28%	9%	79%	14%	7%
3	54%	44%	2%	73%	25%	2%
4	68%	26%	6%	77%	12%	11%
5	54%	38%	8%	70%	20%	10%

Tabulka 4.2: Výsledky metody k nejbližších sousedů (algoritmus 4.1) s přísnějším testem věrohodnosti na problémové množině při použití *Win* příznaků (+ správně, − špatně a ? nerozpoznané číslice)

V dalším experimentu jsme zvýšili počet reprezentantů na celou trénovací množinu. Jedné číslici tak zhruba odpovídalo 30 reprezentantů. Počet správně klasifikovaných vzorů na problémové množině značně vzrostl (tabulka 4.3). Mimo problémovou množinu se špatně klasifikované vzory vyskytovaly jen ojediněle. Na trénovací množině se pak pohybovalo procento správně rozpoznaných číslic v rozmezí 96–100% (při použití příznaků typu *Win*).

Navýšení počtu reprezentantů mělo i negativní důsledky v podobě o něco větších nároků na výpočet. Nárůst byl však zanedbatelný a v reálném provozu by se na rychlosti celé aplikace téměř neprojevil.

Příznaky typu *NCM* se vůbec neosvědčily. Nejlepší výsledek na trénovací množině byl 91% správně rozpoznaných emirátských a 57% latin číslic. Na problémové množině byly výsledky ještě horší — 51% a 21%. Hlavní důvod je ten, že hodnoty jednotlivých momentů jsou řádově různě velké. Emirátské

	latin			emirátské		
k	+	?	−	+	?	−
1	92%	0%	8%	89%	0%	11%
2	82%	14%	4%	87%	8%	5%
3	75%	22%	3%	84%	12%	4%
4	88%	8%	4%	84%	10%	6%
5	86%	11%	3%	84%	10%	6%

Tabulka 4.3: Výsledky metody k nejbližších sousedů (algoritmus 4.1 s více reprezentanty) na problémové množině při použití *Win* příznaků (+ správně, − špatně a ? nerozpoznané číslice)

číslice se nejspíše kvůli malé vzájemné podobnosti (snad kromě číslic 2 a 3) dařilo rozpoznávat lépe.

Klasifikátor lépe pracoval s příznaky typu *Proj*. Pro rostoucí k se opět projevovala sestupná tendence procenta správně rozpoznaných číslic. Emirátské číslice se znovu ukázaly jako lépe rozlišitelné. Věrohodnost se zlepšila s použitím přísnějšího testu věrohodnosti.

Ačkoliv z trénovací a ověřovací množiny byly správně klasifikovány téměř všechny číslice (stejně jako pro příznaky *Win*), na problémové množině se objevilo relativně velké množství chybných klasifikací (tabulka 4.4). Ve srovnání s příznaky *Win* byly o málo lépe klasifikovány latin číslice a o něco hůře emirátské. V případě použití příznaků *Proj* se pro emirátské číslice zhoršila věrohodnost.

	latin			emirátské		
k	+	?	−	+	?	−
1	93%	0%	7%	86%	0%	14%
2	88%	8%	4%	84%	7%	9%
3	81%	18%	1%	80%	13%	7%
4	86%	11%	3%	85%	6%	9%
5	82%	15%	3%	82%	10%	8%

Tabulka 4.4: Výsledky metody k nejbližších sousedů (algoritmus 4.1) na problémové množině při použití *Proj* příznaků a přísnějšího testu věrohodnosti (+ správně, − špatně a ? nerozpoznané číslice)

Pro metodu k nejbližších sousedů je tedy nejvhodnější algoritmus 4.1, ale přesto metoda zůstává pro naši úlohu (rozpoznávání SPZ) prakticky nepoužitelná. I když algoritmus 4.1 dává na problémové množině relativně dobré výsledky, dopouští se chyb na trénovacích a ověřovacích datech (ty by měl

dobrý klasifikátor rozpoznat téměř bez chyby). Pro latin znaky je výhodnější použít příznaků typu *Proj* a pro emirátské typ *Win*.

4.2.2 Mahalanobisova vzdálenost

Použití kovarianční matice v metodě s Mahalanobisovou vzdáleností je volitelné — může být nahrazena jednotkovou maticí (dostáváme euklidovskou vzdálenost).

Nejdříve jsme zkusili odhadnout parametry normálního rozdělení (střední hodnotu a kovarianční matici) pro jednotlivé shluky na základě 5 reprezentantů pro každý shluk. Použili jsme stejné reprezentanty jako u metody k nejbližších sousedů, ale výkon klasifikátoru byl podstatně horší (tabulka 4.5 druhý řádek). O mnoho lépe (tabulka 4.5 první řádek) se dařilo správně klasifikovat číslice bez použití kovarianční matice (nahrazena jednotkovou).

Ukázalo se, že 5 reprezentantů pro odhad kovarianční matice příslušného shluku nestačilo. Jestliže jsme do výpočtu parametrů normálního rozdělení zahrnuli celou trénovací množinu, klasifikace se znatelně zlepšila (tabulka 4.5 třetí a čtvrtý řádek). Celkově však byla opět lepší varianta s jednotkovou maticí.

počet	$\hat{\mathbf{\Sigma}}$	latin	emirátské
5	ne	81%	82%
5	ano	15%	35%
30	ne	85%	83%
30	ano	83%	79%

Tabulka 4.5: Výsledky při použití Mahalanobisovy vzdálenosti a příznaků typu *Win* na problémové množině (procenta udávají úspěšnou klasifikaci, první sloupec ukazuje průměrný počet reprezentantů pro jednu číslici, $\hat{\mathbf{\Sigma}}$ značí použití kovarianční matice)

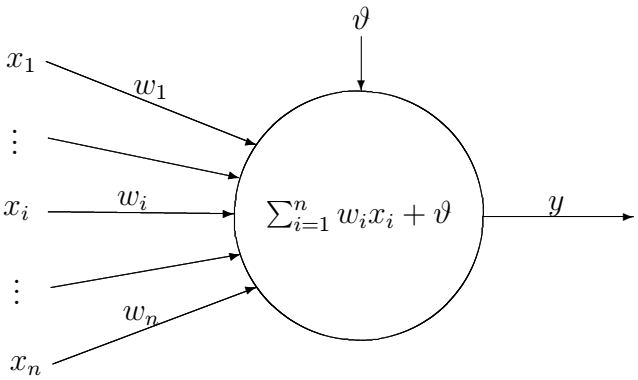
Použitím jiných příznaků jako například *Proj* nebo *NCM* se výsledky nezlepšily.

Mahalanobisova vzdálenost obecně není pro naši úlohu vhodná už proto, že neposkytuje dostatečné rozlišení nerozpoznaných a špatně rozpoznaných číslic. Definovat podmínky pro spolehlivé ověřování věrohodnosti je velice obtížné.

5 Neuronové sítě

5.1 Teorie

Matematický model neuronu vychází ze skutečného biologického základu. *Neuron* (nervová buňka) je základní stavební jednotkou každého složitějšího nervového systému [19]. Neurony se specializují na přenos, zpracovávání a uchovávání informací. Jejich vzájemným spojením vzniká složitější struktura — *neuronová síť*.



Obrázek 5.1: Formální neuron s n vstupy $x_1, \ldots, x_n$, váhami $w_1, \ldots, w_n$ a prahem ϑ

5.1.1 Model neuronu

Formální neuron (obrázek 5.1) byl navržen v roce 1943 Warrenem McCullochem a Waltrem Pittsem [16]. Každá taková jednotka má několik vstupů a pouze jeden výstup. Sama o sobě provádí jen velmi jednoduchou operaci. Od váženého součtu svých vstupů odečte prahovou hodnotu. Tento mezivýsledek je nazýván *potenciálem* neuronu a označuje se ξ:

$$\xi = \sum_{i=1}^{n} w_i x_i + \vartheta \quad , \tag{5.1}$$

kde $w_1, \ldots, w_n$ jsou váhy pro jednotlivé vstupní hodnoty $x_1, \ldots, x_n$ a ϑ je tzv. *práh neuronu*. Výstup neuronu je dán vztahem:

$$y = f(\xi) = f(\sum_{i=1}^{n} w_i x_i + \vartheta) \quad , \tag{5.2}$$

kde f je tzv. *přenosová funkce*. Existují různé typy přenosových funkcí. Mezi nejznámější patří následující:

$$f(\xi) = \begin{cases} 1 & \text{pro } \xi > 0 \\ 0 & \text{jinak} \end{cases} \quad , \tag{5.3}$$

$$f(\xi) = \frac{1}{1 + \mathrm{e}^{-\xi}} \quad , \tag{5.4}$$

$$f(\xi) = \xi \quad . \tag{5.5}$$

Skoková přenosová funkce (rovnice 5.3) provádí velmi jednoduché prahování hodnot. Z matematického hlediska však nemá moc dobré vlastnosti. V bodě 0 není spojitá a má pouze jednostranné derivace. To ji činí nepoužitelnou v dopředných neuronových sítích[4] (viz dále), ale naopak se dá s výhodou použít v jednoduchém modelu neuronu.

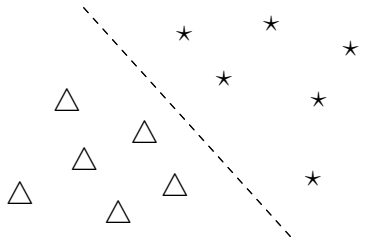
Asi nejpoužívanější je funkce popsaná rovnicí (5.4). Je označována jako *logická sigmoida*. V podstatě se jedná o „vyhlazení skokové funkce v okolí bodu 0“. Její hodnoty se pohybují v intervalu $\langle 0, 1 \rangle$, je spojitá, hladká, nelineární a její derivace se dá vyjádřit pomocí funkce samé:

$$f'(x) = f(x) \cdot (1 - f(x)) \quad .$$

Třetí zmíněná přenosová funkce je *identita*. Potenciál neuronu nechává nezměněn. Výstupem neuronu tedy může být libovolné reálné číslo.

Lze dokázat, že jediný neuron je schopen tzv. *lineární klasifikace* dvou množin (tj. separace dvou množin pomocí jedné nadroviny) [4, kap. 5.5.2]. Podmínkou je *lineární separabilita* těchto množin (příklad na obrázku 5.2) — to znamená, že příznakový prostor se dá nadrovinou rozdělit na dvě části tak, aby v jedné části byly jen vzory z první množiny a ve druhé části jen z druhé množiny.

[4]Modifikací algoritmu zpětného šíření lze použít i po částech lineární přenosovou funkci [4]. Avšak na úkor větší výpočetní složitosti a přitom to nepřinese lepší výsledky.



Obrázek 5.2: Lineární separabilita dvou množin

5.1.2 Dopředné neuronové sítě

Obecnou neuronovou síť můžeme definovat jako orientovaný graf, v němž uzly představují neurony a orientované hrany jsou spoje mezi nimi. Každé hraně je přiřazena váha w (*vážené spoje*). Dále existuje neprázdná množina vstupních neuronů a neprázdná množina výstupních neuronů.

Vrstevnatá dopředná neuronová síť (obrázek 5.3) potom klade následující omezení:

- Množina neuronů (vrcholů) je rozdělena do posloupnosti vzájemně disjunktních podmnožin zvaných *vrstvy*. Budeme je indexovat písmenem a a jejich počet si označíme L. Počet neuronů v každé vrstvě může být jiný.

- Hrany v grafu vedou vždy pouze z vrstvy $a-1$ do vrstvy a a to tak, že vytvářejí úplný bipartitní podgraf. Váhu na hraně vedoucí z i-tého neuronu ve vrstvě $a-1$ do j-tého neuronu vrstvy a označujeme $w_{ij}^{(a)}$. Symbol $\vartheta_j^{(a)}$ představuje prahovou hodnotu j-tého neuronu ve vrstvě a.

- První vrstva ($a = 1$) se nazývá *vstupní* — obsahuje speciální vstupní neurony s identickou přenosovou funkcí a prahovou hodnotou nula. Výstupní hodnota těchto neuronů se rovná jejich vstupní hodnotě. Tyto neurony nemají v grafu spojů žádné předchůdce. Mají za úkol propagovat dál vstup pro celou neuronovou síť $(x_1, \ldots, x_n)$.

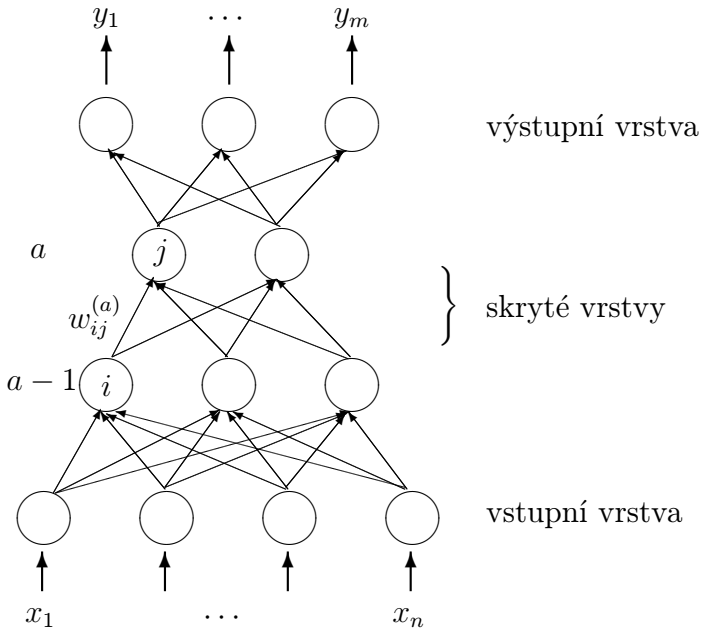
- Poslední vrstva ($a = L$) se nazývá *výstupní* — neurony v této vrstvě nemají v grafu spojů žádné následníky a z jejich výstupů se čte výstup celé neuronové sítě $(y_1, \ldots, y_m)$. Přechodovou funkcí v této vrstvě je logická sigmoida.

- Ostatní vrstvy ($1 < a < L$) se nazývají *skryté* — neurony v této vrstvě používají logickou sigmoidu jako přechodovou funkci.

- Výstup j-tého neuronu ve vrstvě a budeme označovat $o_j^{(a)}$ a jeho potenciál $\xi_j^{(a)}$. Pro $a > 1$ se $o_j^{(a)}$ spočítá se na základě výstupů všech neuronů z předchozí vrstvy $a-1$ pomocí vzorce (modifikovaná verze vzorce 5.2):

$$o_j^{(a)} = f(\xi_j^{(a)})) = f(\sum_i w_{ij}^{(a)} o_i^{(a-1)} + \vartheta_j^{(a)}) \quad . \tag{5.6}$$

- Pro vstupní vrstvu platí $o_i^{(1)} = x_i$ a pro výstupní vrstvu $o_i^{(L)} = y_i$.



Obrázek 5.3: Dopředná vrstevnatá neuronová síť se čtyřmi vrstvami

Rozložení neuronů v jednotlivých vrstvách popisuje tzv. *architektura* neuronové sítě. Na obrázku 5.3 je síť s architekturou 4-3-2-3, kde čísla udávají počet neuronů ve vrstvách směrem od vstupní vrstvy k výstupní.

Nyní si popíšeme činnost vrstevnaté neuronové sítě. Po předložení vzoru $\boldsymbol{x}$ se postupně počítají výstupy jednotlivých vrstev (od vstupní vrstvy směrem k výstupní). Příznakový prostor je pomocí skrytých vrstev vhodně nelineárně transformován, aby poslední výstupní vrstva mohla provést lineární klasifikaci na nově vzniklém (transformovaném) prostoru. Výstup poslední vrstvy představuje *skutečný výstup* neuronové sítě $y_1, \ldots, y_m$.

Abychom mohli síť učit, musíme mít k dispozici trénovací množinu X v podobě uspořádaných dvojic $\langle$ vzor $\boldsymbol{x}$, požadovaný výstup sítě $\boldsymbol{d}$ $\rangle$. Dále musíme umět stanovit chybu, s jakou se liší skutečný výstup neuronové sítě od

požadovaného. Ke stanovení celkové chyby na celé trénovací množině slouží *kriteriální (chybová) funkce*:

$$E = \frac{1}{2} \sum_k \sum_j (y_{j,k} - d_{j,k})^2 \quad , \tag{5.7}$$

kde k indexuje uspořádané dvojice v trénovací množině a $y_{j,k}$ (resp. $d_{j,k}$) je skutečný (resp. požadovaný) výstup j-tého neuronu ve výstupní vrstvě neuronové sítě po předložení k-tého trénovacího vzoru $\boldsymbol{x_k}$.

Cílem učení je minimalizace kriteriální funkce pomocí *algoritmu zpětného šíření* (odvození je možné najít v [23, str. 149–169]). Pomocí něj dokážeme po každém předložení trénovacího vzoru $\boldsymbol{x}$ určit, jakým způsobem mají být váhy a prahové hodnoty neuronů v síti modifikovány, aby došlo k poklesu kriteriální funkce.

Algoritmus zpětného šíření pracuje tak, že určí velikost chyby na výstupu každé vrstvy sítě a tuto chybu pak propaguje do nižších vrstev pomocí vážených spojů. Začíná u výstupní vrstvy a postupně upravuje váhy směrem ke vstupní vrstvě. Změny jsou určeny vztahem:

$$\Delta w_{ij}^{(a)} = \delta_j^{(a)} o_i^{(a-1)} \quad , \tag{5.8}$$

kde $\Delta w_{ij}^{(a)}$ označuje změnu váhy na hraně vedoucí z i-tého neuronu ve vrstvě $a-1$ do j-tého neuronu vrstvy a. V závislosti na aktuálně zpracovávané vrstvě pak platí:

$$\begin{aligned} \delta_j^{(L)} &= (d_j - y_j) \cdot f'(\xi_j^{(L)}) && \text{pro } j\text{-tý výstupní neuron,} \\ \delta_i^{(a)} &= f'(\xi_i^{(a)}) \sum_j w_{ij}^{(a+1)} \delta_j^{(a+1)} && \text{pro } i\text{-tý neuron ve vrstvě } a < L. \end{aligned} \tag{5.9}$$

Nové váhy a prahové hodnoty neuronů v síti jsou upraveny podle vzorce:

$$\begin{aligned} w_{ij}^{(a)} &\leftarrow w_{ij}^{(a)} + \Delta w_{ij}^{(a)} \quad , \\ \vartheta_j^{(a)} &\leftarrow \vartheta_j^{(a)} + \delta_j^{(a)} \quad , \end{aligned} \tag{5.10}$$

přičemž prahové hodnoty neuronů ve vstupní vrstvě se nemění.

Trénování sítě probíhá většinou v tzv. *epochách*. V každé epoše jsou síti postupně předloženy všechny vzory z trénovací množiny. Celý proces se v další epoše opakuje. Podle toho, jakým způsobem vzory předkládáme a kdy adaptujeme váhy, rozlišujeme dva přístupy učení:

1. *Online učení* (algoritmus 5.1) — váhy jsou adaptovány ihned po předložení každého vzoru. Tento způsob je výhodný zejména tehdy, pokud je trénovací množina příliš velká a nemůže být udržována celá v paměti.

Algoritmus 5.1 Online učení neuronové sítě

1: inicializace jednotlivých vah $w_{ij}^{(a)}$ pro všechny a, i, j
2: $k \leftarrow 1$
3: **repeat**
4: $\langle \mathbf{x}, \mathbf{d} \rangle \leftarrow$ k-tá uspořádaná dvojice z X
5: upravit váhy a prahové hodnoty neuronů na základě rovnic 5.10
6: $k \leftarrow k + 1$
7: **if** $k > |X|$ **then**
8: $k \leftarrow 1$ /* konec epochy a začátek nové */
9: **end if**
10: **until** ukončovací podmínka

2. *Dávkové učení* (algoritmus 5.2) — z hlediska učení je výhodnější, protože změny vah provádí až na konci každé epochy. Změny vah a prahových hodnot neuronů jsou v průběhu epochy kumulativně přičítány do pomocných proměnných ($\Delta W_{ij}^{(a)}$ a $\Delta T_j^{(a)}$).

Algoritmus 5.2 Dávkové učení neuronové sítě

1: inicializace jednotlivých vah $w_{ij}^{(a)}$
2: **repeat**
3: $\Delta W_{ij}^{(a)} \leftarrow 0$ pro všechny a, i, j /* pomocné proměnné */
4: $\Delta T_j^{(a)} \leftarrow 0$ pro všechny a, j /* pomocné proměnné */
5: **for** $k \leftarrow 1, \ldots, |X|$ **do**
6: $\langle \mathbf{x}, \mathbf{d} \rangle \leftarrow$ k-tá uspořádaná dvojice z X
7: $\forall a, i, j$: $\Delta W_{ij}^{(a)} \leftarrow \Delta W_{ij}^{(a)} + \Delta w_{ij}^{(a)}$ /* na základě rovnice 5.8 */
8: $\forall a, j$: $\Delta T_j^{(a)} \leftarrow \Delta T_j^{(a)} + \delta_j^{(a)}$ /* na základě rovnic 5.9 */
9: **end for** /* konec epochy */
10: $w_{ij}^{(a)} \leftarrow w_{ij}^{(a)} + \Delta W_{ij}^{(a)}$ pro všechny a, i, j
11: $\vartheta_j^{(a)} \leftarrow \vartheta_j^{(a)} + \Delta T_j^{(a)}$ pro všechny a, j
12: **until** ukončovací podmínka

Jednou z možností, jak definovat ukončovací podmínku algoritmů 5.1 a 5.2, je sledovat kriteriální funkci, zda neklesne pod stanovenou mez. Malá chyba na trénovací množině však neznamená dobře naučenou síť. Chyba na datech mimo trénovací množinu může být naopak velká — tzv. *přeučení* sítě.

Síť by měla mít dobrou schopnost „zobecňovat“ — tzn. měla by dávat dobré výsledky i na datech, na kterých se neučila. K tomuto účelu se obvykle odděluje z trénovacích dat tzv. *ověřovací množina*. Vzory z ní se při učení nepoužijí. Je určena k ověřování „naučenosti“ sítě během učení. Pokud dochází v průběhu učení ke snižování chyby na ověřovací množině, algoritmus učení pokračuje. V opačném případě je proces učení zastaven.

Změnu vah a prahových hodnot neuronů v průběhu učení může navíc ovlivňovat tzv. *učící parametr* η:

$$\begin{array}{rcl} w_{ij}^{(a)} & \leftarrow & w_{ij}^{(a)} + \eta \cdot \Delta w_{ij}^{(a)} \\ \vartheta_j^{(a)} & \leftarrow & \vartheta_j^{(a)} + \eta \cdot \delta_j^{(a)} \end{array} \quad . \tag{5.11}$$

Parametr $\eta \in \langle 0, 1 \rangle$ má vliv na rychlost učení, ale na druhé straně při nastavení vysoké hodnoty může způsobit „přeskočení“ globálního minima kriteriální funkce.

Existuje několik heuristik pro zlepšení a urychlení učení. Mezi nejdůležitější patří *učení s momentem* [4, kap. 6.8.10]. Modifikací rovnice 5.11 se docílí rychlejší konvergence:

$$w_{ij}^{(a)}(t+1) \leftarrow w_{ij}^{(a)}(t) + \eta \cdot \Delta w_{ij}^{(a)} + \alpha \cdot \left(w_{ij}^{(a)}(t) - w_{ij}^{(a)}(t-1) \right) \quad ,$$

kde $w_{ij}^{(a)}(t)$ je příslušná váha v kroku t. Pro prahové hodnoty vypadá vzorec obdobně. Vlivem momentu $\alpha \in \langle 0, 1 \rangle$, který se snaží částečně udržet předchozí směr poklesu kriteriální funkce, nedochází ke zbytečným oscilacím.

Neuronovou síť budeme používat jako klasifikátor, který má neznámý vzor klasifikovat do jedné z c tříd. Jednou z možností je použít pouze jeden výstupní neuron s identickou přenosovou funkcí (rovnice 5.5 na str. 48). Naučená síť by měla na předložený vstupní vzor $\mathbf{x}$ dát na výstup index (po zaokrouhlení) odpovídající třídy, do které byl tento vzor klasifikován. Na pokusech jsme se přesvědčili, že výsledky nebyly dobré.

Relativně jednodušší bylo naučit síť tak, aby na vstupní vzor $\mathbf{x}$ z trénovací množiny dávala požadovaný výstup $\mathbf{d}$:

$$\mathbf{d} = \boldsymbol{e}(i) \quad , \tag{5.12}$$

kde vzor $\mathbf{x}$ z trénovací množiny má být klasifikován do třídy i, vektor $\mathbf{d}$ má c složek (celkem c tříd) a platí:

$$\boldsymbol{e}(i) = (\overbrace{\underbrace{0, \ldots, 0}_{i-1}, 1}^{i}, 0, \ldots, 0) \quad .$$

Neznámý vzor $\mathbf{x}$, pro který dává neuronová síť výstup $\mathbf{y}$, je pak klasifikován do třídy i podle vztahu:

$$i = \underset{j}{\operatorname{argmax}}\{y_j\} \quad , \tag{5.13}$$

to znamená, že se vezme neuron z výstupní vrstvy, který má na výstupu maximální hodnotu, a jeho index bude představovat index klasifikované třídy.

Počet neuronů ve skrytých vrstvách je závislý na konkrétní aplikaci. Typicky se určuje experimentálně. V některých situacích může pomoci tzv. *zlaté pravidlo* [4, kap. 6.8.7]. Počet všech vah a prahových hodnot (kromě prahových hodnot ve vstupní vrstvě) uvnitř neuronové sítě určuje tzv. *stupeň volnosti*. Proto platí, že by stupeň volnosti neměl být větší, než je celkový počet vstupních hodnot N (tzn. velikost trénovací množiny $|X|$ krát dimenze vstupních vektorů n). Zlaté pravidlo říká, že pokud je použito nejvýše tolik skrytých neuronů, aby stupeň volnosti nepřesáhl číslo $N/10$, dosáhne se při trénování nejnižší chyby na ověřovací množině.

Pro síť s jednou skrytou vrstvou jsme podmínku zlatého pravidla pro stupeň volnosti odvodili následovně:

$$q_h \cdot (q_{in} + q_{out}) + q_h + q_{out} \leq \frac{N}{10} \quad ,$$

kde q_h označuje hledaný počet skrytých neuronů a q_{in} a q_{out} označují počet vstupních a výstupních neuronů. Celkový počet vah v síti je $q_h \cdot (q_{in} + q_{out})$ a prahových hodnot (kromě vstupní vrstvy) je $q_h + q_{out}$. Odtud dostáváme:

$$q_h \leq \frac{\frac{N}{10} - q_{out}}{q_{in} + q_{out} + 1} \quad . \tag{5.14}$$

5.2 Výsledky

5.2.1 Dopředné neuronové sítě

Pro rozpoznávání jsme zkoušeli použít sítě s různou architekturou. Zatímco počet vstupních neuronů závisel na použitém příznakovém vektoru, s počtem skrytých neuronů jsme mohli experimentovat. Učení probíhalo dávkově (algoritmus 5.2) opakovaným předkládáním všech vzorů z trénovací množiny. Metoda s použitím momentu (viz teorie) byla pro naše účely příliš pomalá a nepodařilo se s ní dosáhnout přijatelné chyby na trénovací množině (testováno s příznaky typu *Win*).

Daleko rychlejší byla metoda Levenberg-Marquardt (dále jen metoda LM) popsaná v [8] a [13]. V tomto případě však hrozí přeučení, a proto musí být při trénování použita ověřovací množina. Přeučená síť totiž dává výborné výsledky na trénovacích datech, ale na neznámých datech jsou výsledky spíše špatné (tabulka 5.1).

trénovací	ověřovací
100%	84%

Tabulka 5.1: Úspěšnost klasifikace latin číslic pro přeučenou neuronovou síť (metoda LM) s architekturou 52-18-10 a příznaky *Win*

Metoda LM sice rychle konverguje k dobrému výsledku, ale je velmi náročná na paměť. S počítači, které jsme měli v laboratoři k dispozici (viz str. 77), jsme metodu LM nemohli otestovat pro příznaky typu *Pix*. Tento typ příznaků totiž vyžaduje použití 900 vstupních neuronů.

Rozhodli jsme se vyzkoušet metodu *Rprop* (viz [22]). Během krátké doby dosahovala metoda poměrně malé chyby na trénovací množině. Pro kontrolu přeučení sítě jsme opět použili ověřovací množinu. Vzhledem k tomu, že metoda Rprop nemá takové paměťové nároky jako LM, mohli jsme otestovat i sítě s architekturou 900-35-10.

Zjistili jsme, že klasifikace pomocí jediného výstupního neuronu je pro tuto úlohu nepoužitelná. Ačkoliv byla síť schopna dobře rozpoznávat trénovací vzory (latin číslice s příznaky typu *Win*), na ověřovací množině se úspěšnost pohybovala okolo 50%.

Rozšířili jsme výstupní vrstvu na 10 neuronů (10 klasifikačních tříd) podle vzorce 5.12. Zlepšilo se rozpoznávání neznámých vzorů, na kterých se síť neučila. Zároveň jsme získali možnost kontroly věrohodnosti výstupu (viz str. 58). Na trénovací i ověřovací množině bylo dosaženo většinou 100% úspěšnosti.

Při hledání optimálního počtu skrytých neuronů jsme postupovali experimentálně. Pokusy často potvrzovaly hodnotu získanou pomocí zlatého pravidla (nerovnice 5.14) v závislosti na použitém typu příznaků.

Jestliže jsou vstupem sítě jednotlivé pixely obrázku (příznaky typu *Pix*), dostáváme síť velkých rozměrů s 900 vstupními neurony. Při zachování třívrstvé architektury doporučuje zlaté pravidlo ($N = 352 \cdot 900$ pro latin číslice $N = 280 \cdot 900$ pro emirátské) použít maximálně 34 (resp. 27) skrytých neuronů pro síť rozpoznávající latin (resp. emirátské) číslice (tabulka 5.2).

síť	latin	emirátské
900-27-10	82%	**77%**
900-34-10	**85%**	75%

Tabulka 5.2: Neuronová síť s příznaky *Pix* na problémové množině — mírné zlepšení úspěšnosti klasifikace při použití doporučovaného počtu skrytých neuronů pro latin a emirátské číslice

Příznaky typu *NCM* s momenty do řádu 3 nejsou dostačující — testovaná síť nebyla schopna si „zapamatovat“ ani trénovací vzory. Použití momentů vyšších řádů (až do řádu 7) již přineslo zlepšení (úspěšnost 99% na trénovací množině). Další přidávání momentů už úspěšnost klasifikace nezvýšilo. Na ověřovací množině však nebyla úspěšnost dostačující (okolo 88%). Ukázalo se (podobě jako pro klasifikátor minimální vzdálenosti), že neuronová síť lépe rozpoznává emirátské číslice.

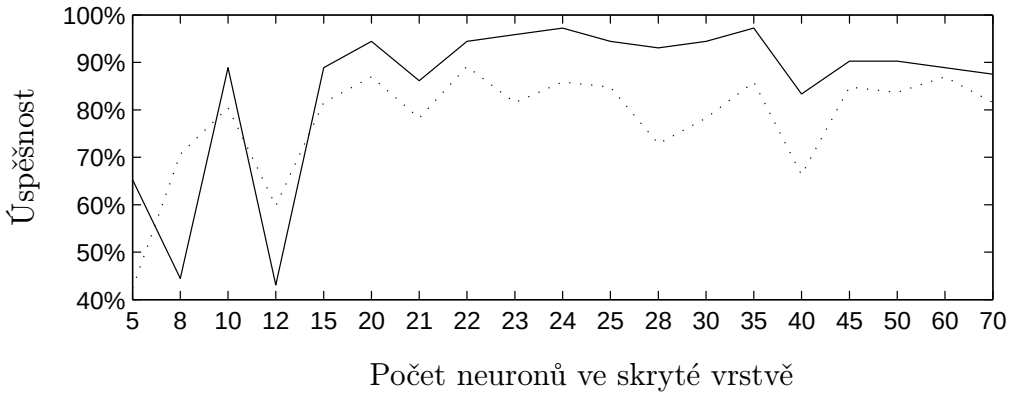
síť	latin			emirátské		
	T	O	P	T	O	P
60-20-10	90%	76%	71%	100%	93%	68%
60-22-10	100%	94%	75%	100%	87%	61%

Tabulka 5.3: Výsledky neuronových sítí pro příznaky typu *Proj* (úspěšnost na T trénovací, O ověřovací a P problémové množině)

Podle zlatého pravidla by měla být pro příznaky typu *Proj* optimální síť s 29 (resp. 23) skrytými neurony pro latin (resp. emirátské) číslice. Chybu na trénovací množině se však nepodařilo snížit na rozumnou hodnotu. Největší procento správně klasifikovaných latin (resp. emirátských) číslic na trénovací množině měly sítě s 20 (resp. 22) neurony ve skryté vrstvě (tabulka 5.3). Schopnost generalizace (rozpoznávání neznámých vzorů mimo trénovací množinu) však byla slabá podobně jako u *NCM*.

Nejlepších výsledků dosahovaly sítě při předkládání příznaků typu *Win*. Proto jsme se v dalších experimentech zaměřili zejména na tyto příznaky.

Zlaté pravidlo dává pouze horní odhad počtu skrytých neuronů a je odvozeno čistě empiricky. V případě příznaků typu *Win* a třívrstvé architektury sítě by měla být podle zlatého pravidla optimální síť s maximálně 30 (resp. 24) skrytými neurony pro latin (resp. emirátské) číslice.



Obrázek 5.4: Graf úspěšnosti třívrstvých neuronových sítí na problémové množině s příznaky typu *Win* v závislosti na různém počtu skrytých neuronů — velikost skryté vrstvy je vynesena na vodorovné ose, úspěšnost u latin (resp. emirátských) číslic je zobrazena plnou (resp. přerušovanou) čárou

Na levé straně grafu (obrázek 5.4) je možné sledovat vzestupný trend pro malé počty skrytých neuronů. Úspěšnost klasifikace se potom spíše ustálí, jakmile jich má síť k dispozici dostatečný počet (asi od 20).

Ačkoliv se může zdát, že příznaky *Win* dovolovaly lépe klasifikovat latin číslice, horší klasifikace emirátských číslic byla způsobena většími chybami při jejich segmentaci. Emirátské číslice obsažené v problémové množině byly více deformované.

Průběh (trendy) grafu na trénovací množině je obdobný, jen hodnoty se pohybují velmi blízko 100%. Výsledky pro latin i emirátské číslice se vůbec neliší.

Na základě výsledků na problémové množině jsem vybrali dvě nejlepší sítě:

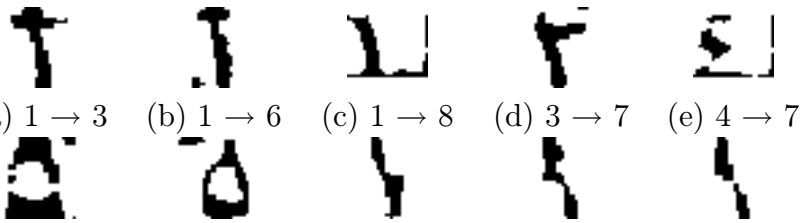
- architektura 81-24-10 — 97% rozpoznaných latin číslic
- architektura 81-22-10 — 89% rozpoznaných emirátských číslic

Zajímalo nás, co je hlavní příčinou špatně rozpoznaných znaků a jak by bylo možné dále zlepšit klasifikaci.

(a) 0 → 9 (b) 6 → 5

Obrázek 5.5: Chybně klasifikované latin znaky z problémové množiny — první číslo udává správný výsledek a druhé výsledek klasifikace

Špatně klasifikované byly jen dva latin znaky (obrázek 5.5). Zatímco člověk tyto znaky rozezná, síť se zřejmě „soustředí“ jen na některé části obrázku. V případě obrázku 5.5a to může být například zapříčiněno chybějící částí číslice 0. Tato mezera je naopak typická pro číslici 9.



(a) 1 → 3 (b) 1 → 6 (c) 1 → 8 (d) 3 → 7 (e) 4 → 7

(f) 5 → 3 (g) 5 → 7 (h) 7 → 1 (i) 8 → 1 (j) 8 → 9

Obrázek 5.6: Chybně klasifikované emirátské znaky z problémové množiny — první číslo udává správný výsledek a druhé výsledek klasifikace

Emirátských číslic bylo chybně klasifikováno více (obrázek 5.6). Z obrázků 5.6h, 5.6i a 5.6j je patrné, že síť není schopna správně rozpoznat obrázky

(a) E6 (b) E9 (c) L5 (d) L7 (e) L9

Obrázek 5.7: Správně klasifikované znaky — E značí emirátskou a L latin číslici

číslic, jejichž velká část díky špatné segmentaci chybí (porovnání na obrázku 2.1 na str. 4).

O tom, že některé části obrázku jsou sítí „ignorovány“, vypovídají i obrázky správně klasifikovaných číslic (obrázek 5.7). Mnohé z nich má problém rozlišit i člověk.

Čtyřvrstvé sítě se svou úspěšností „přiblížili“ k nejlepším třívrstvým architekturám (97% latin a 89% emirátských číslic z problémové množiny). Při testování čtyřvrstvých neuronových sítí na problémové množině jsme však ve srovnání s třívrstvými sítěmi nezískali lepší výsledky:

- architektura 81-15-15-10 — 95% rozpoznaných latin číslic
- architektura 81-20-20-10 — 85% rozpoznaných emirátských číslic

Použití neuronových sítí má kromě přesné klasifikace také jinou výhodu. Navzdory dlouhé době potřebné k jejich natrénování, vyžadují relativně krátkou dobu (řádově milisekund) pro spočítání svého výstupu. Důvodem je násobení poměrně malých váhových matic při výpočtu výstupu sítě.

5.2.2 Věrohodnost rozpoznání

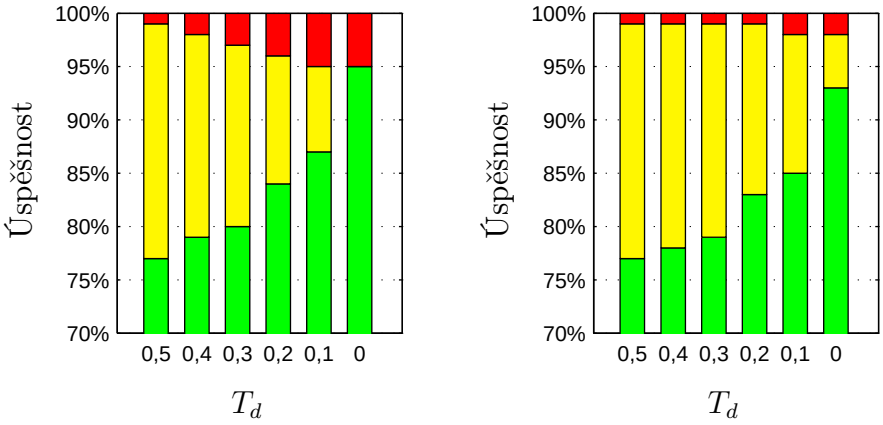
Nyní se budeme zabývat případem, kdy výstupem sítě je vektor čísel, jejichž maximum určuje třídu, do které má být neznámý vzor klasifikován (viz rovnice 5.12). K testování jsme použili již natrénované sítě 81-24-10 (pro latin číslice) a 81-22-10 (pro emirátské číslice) společně s příznaky typu *Win*.

Identifikační údaj na SPZ je rozpoznáván po dvojicích navzájem si odpovídajících latin a emirátských znaků. Výstupy dvou různých sítí pro spárované latin (výstup $\mathbf{y}^{lat}$) a emirátské znaky (výstup $\mathbf{y}^{em}$) musí být vhodně zkombinovány dohromady. Mělo by se přitom využívat vizuálních vah σ_i (tabulka 3.3) a segmentační věrohodnosti κ (rovnice 3.2).

Výsledný výstup $\mathbf{y}$ jsme definovali následujícím způsobem:

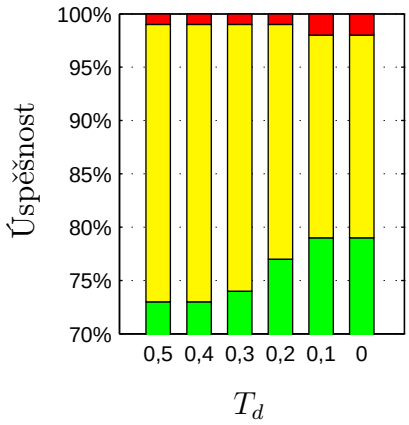
$$y_i = y_i^{lat}(\sigma_i + 0,5\kappa) + y_i^{em}(1 - \sigma_i - 0,5\kappa) \quad , \tag{5.15}$$

přičemž dvojice obrázků pro latin a emirátskou číslici je klasifikována do příslušné třídy i na základě vzorce 5.13.



(a) $T_m = 0,45$

(b) $T_m = 0,50$



(c) $T_m = 0,55$

Obrázek 5.8: Různé hodnoty prahů pro věrohodnost rozpoznávání — na ose x jsou vyneseny hodnoty T_d, zelená barva označuje procento správně rozpoznaných, žlutá nerozpoznaných a červená špatně rozpoznaných SPZ, pro lepší názornost je zobrazeno pouze horních 30% celého grafu

Experimentálně jsme zjistili, že není dobré, když proměnná κ příliš zvýhodňuje některý z výstupů, a proto jsme její hodnotu snížili na polovinu (vynásobením konstantou $0,5$).

Pro kontrolu věrohodnosti jsme stanovili pravidla, která musí nalezené maximum z výsledného výstupu splňovat, aby mohla být dvojice obrázků číslic správně klasifikována do nějaké třídy. Pro nalezené maximum musí platit:

1. Musí překračovat stanovenou minimální hodnotu maxima (T_m).
2. Od ostatních hodnot ve výstupu se musí lišit alespoň o stanovenou hodnotu (T_d).

Sestavili jsme graf (obrázek 5.8) úspěšnosti rozpoznávání SPZ pro různé prahové hodnoty T_m a T_d, abychom mohli výsledky vizuálně porovnat. Nejvhodnější se jeví hodnoty, které zaručují nízké procento chybně rozpoznaných SPZ (co nejméně na úkor správně rozpoznaných SPZ).

Testování probíhalo na 100 obrázcích SPZ, jejichž rohy byly dopředu manuálně označeny (manuální lokalizace SPZ). Segmentaci číslic prováděl algoritmus automaticky. S hodnotami $T_m = 0,5$ a $T_d = 0$ (obrázek 5.8b pravý sloupec) bylo dosaženo nejlepších výsledků (největší procento správně rozpoznaných SPZ se zachováním podmínky z nerovnice 2.4):

- 93% správně rozpoznaných SPZ
- 5% nerozpoznaných SPZ
- 2% špatně rozpoznaných SPZ

Pro rozpoznávání textu SPZ jsme zkusili použít i sítě, které nedávaly až tak dobré výsledky při rozpoznávání samostatných číslic na problémové množině. Pro oba typy číslic jsme použili sítě se stejnou architekturou 81-21-10. Sledovali jsme hlavně, jestli se sníží procento špatně rozpoznaných SPZ.

Z tabulky 5.4 je vidět, že sítě mají sice menší procento správně rozpoznaných SPZ, ale špatně rozpoznané SPZ se v tomto případě téměř nevyskytují. Například výsledek pro hodnoty $T_m = 0,45$ a $T_d = 0,1$ může být z hlediska řešení úlohy celkem přijatelný, protože neuronové sítě neidentifikovaly špatně ani jedinou SPZ a přitom si udržely vysoké procento (86%) správně rozpoznaných SPZ.

Z toho jsme usoudili, že pokud je neuronová síť schopna správně klasifikovat velké množství problémových číslic, projeví se to negativně na procentu špatně rozpoznaných SPZ. V závislosti na tom, jaké je pro systém požadováno procento úspěšně rozpoznaných SPZ, může být někdy výhodnější použít neuronové sítě s horšími výsledky na problémové množině.

T_m	T_d	+	?	−
0,45	0	93%	5%	2%
0,45	0,1	86%	14%	0%
0,45	0,2	81%	19%	0%
0,5	0	85%	14%	1%
0,5	0,1	81%	19%	0%
0,5	0,2	77%	23%	0%
0,55	0	68%	32%	0%
0,55	0,1	65%	35%	0%
0,55	0,2	64%	36%	0%

Tabulka 5.4: Výsledky rozpoznávání SPZ pro „hůře natrénované“ sítě s architekturou 81-21-10 a příznaky typu *Win* (+ správně, − špatně a ? nerozpoznané SPZ)

6 Skryté Markovovy modely

6.1 Teorie

Skryté Markovovy modely (dále jen HMM[5]) jsou nejčastěji zmiňovány v souvislosti s analýzou mluvené řeči. Čím dál více se ale rozšiřují i do oblasti rozpoznávání obrazu. Hlavním důvodem je jejich snaha využít prostorových závislostí rozpoznávaných dat.

Základní jednorozměrný model prvního řádu představil v několika publikacích Leonard E. Baum [1] se svými spolupracovníky. Jejich iniciativa vycházela hlavně z prací, jimiž se zabýval Andrei A. Markov.

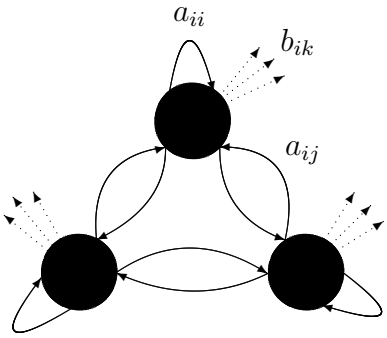
6.1.1 Základní diskrétní model

HMM je pravděpodobnostní model. Můžeme si ho představit jako konečný automat, který mění své stavy na základě přechodové funkce (obrázek 6.1). Přechod ze stavu do stavu však není deterministický, ale je uskutečněn jen s určitou pravděpodobností.

Stav, ve kterém se právě HMM nachází, není možné „zvenčí“ přesně určit. Proto jsou stavy označovány jako skryté. Místo toho však dává HMM v každém kroku na výstup nějaký symbol. Pro každý stav je opět definována pravděpodobnost, s jakou se daný symbol objeví na výstupu.

Formálně se HMM definuje jako uspořádaná pětice $H = (S, V, \boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$, kde $S = \{s_1, \ldots, s_{|S|}\}$ je neprázdná množina stavů a $V = \{v_1, \ldots v_{|V|}\}$ je neprázdná množina výstupních symbolů. Množiny S a V musí být konečné. Pravděpodobnost, že s_i je počáteční stav HMM, udává prvek π_i vektoru $\boldsymbol{\pi} = (\pi_i)$. $\mathbf{A} = (a_{ij})$ je přechodová matice mezi jednotlivými stavy o velikosti $|S| \times |S|$ a a_{ij} značí pravděpodobnost přechodu ze stavu s_i do stavu s_j. Matice $\mathbf{B} = (b_{jk})$ má rozměry $|S| \times |V|$ a prvek b_{jk} určuje, s jakou pravděpodobností je na výstup dán symbol v_k v případě, že se HMM nachází ve stavu s_j.

[5] angl. *hidden Markov models*



Obrázek 6.1: Skrytý Markovův model

Protože se jedná o pravděpodobnosti, musí součet složek ve všech řádcích matic **A** a **B** a také složky vektoru $\boldsymbol{\pi}$ dávat v součtu 1:

$$\sum_j a_{ij} = \sum_k b_{jk} = \sum_i \pi_i = 1 \quad .$$

Příznakový vektor $\mathbf{x} = \{x_1, \ldots, x_n\}$ představuje sekvenci jednotlivých pozorování, která jsou provedena v čase za sebou. A tedy $x_t \in V$ pro všechna $t = 1, \ldots, n$. Mezi pozorováními existují jisté pravděpodobnostní závislosti. Omezíme-li se pouze na HMM prvního řádu, budeme předpokládat závislost pozorování v čase t pouze na předchozím pozorování v čase $t-1$.

Pokud máme již sestrojený HMM, chceme vědět, s jakou pravděpodobností by při svém chodu vygeneroval předloženou sekvenci symbolů (neznámý vzor **x**). Tato pravděpodobnost se spočte podle vzorce (viz [4]):

$$\begin{aligned} P(\mathbf{x}) &= \sum_{(s(1),\ldots,s(n))\in\Psi} P(s(1))P(x_1|s(1)) \prod_{t=2}^{n} P(s(t)|s(t-1))P(x_t|s(t)) \\ &= \sum_{(s(1),\ldots,s(n))\in\Psi} \pi_{s(1)} b_{s(1)k(1)} \prod_{t=2}^{n} a_{s(t-1)s(t)} b_{s(t)k(t)} \quad , \end{aligned} \tag{6.1}$$

kde Ψ je množina všech posloupností stavů HMM délky n (celkem $|S|^n$ posloupností) a $s(t)$ je stav, ve kterém se nacházel HMM při vygenerování symbolu x_t. Index symbolu x_t v množině V je označen $k(t)$ a tedy $x_t = v_{k(t)}$. Dále se zde vyskytují pravděpodobnosti:

- $P(s(1))$ — pravděpodobnost, že se HMM na počátku nacházel ve stavu $s(1)$.
- $P(s(t)|s(t-1))$ — pravděpodobnost, že HMM přešel v kroku t ze stavu $s(t-1)$ do stavu $s(t)$.

- $P(x_t|s(t))$ — pravděpodobnost, že HMM ve stavu $s(t)$ vygeneroval symbol x_t.

Algoritmus na výpočet pravděpodobnosti využívající rovnice 6.1 má složitost $O(n|S|^n)$ a je tedy v praxi nepoužitelný. Využitím metod dynamického programování lze složitost výsledného algoritmu (algoritmus 6.1) snížit až na $O(n|S|^2)$ [4].

Algoritmus 6.1 Dopředný algoritmus pro HMM

1: inicializace $\alpha_j(1) \leftarrow \pi_j b_{jk(1)}$
2: **for** $t \leftarrow 2, \ldots, n$ **do**
3: **for** $j \leftarrow 1, \ldots, |S|$ **do**
4: $\alpha_j(t) \leftarrow b_{jk(t)} \sum_{i=1}^{|S|} \alpha_i(t-1) a_{ij}$
5: **end for**
6: **end for**
7: **return** $\sum_j \alpha_j(n)$

Proměnná $\alpha_j(t)$ představuje pravděpodobnost, že HMM je v čase t ve stavu s_j a správně vygeneroval začátek sekvence $x_1, \ldots, x_t$.

Pro klasifikaci stačí zkonstruovat Markovův model pro každou ze tříd. Algoritmus 6.1 spočítá pravděpodobnost p_i vygenerování sekvence neznámého vzoru $\mathbf{x}$ pro každý z modelů. Vzor je zařazen do třídy, pro jehož model má pravděpodobnost nejvyšší hodnotu (algoritmus 6.2).

Algoritmus 6.2 Klasifikace pomocí HMM

1: mějme HMM_i pro každou z klasifikačních tříd
2: **for all** HMM_i **do**
3: $p_i \leftarrow$ výstup algoritmu 6.1 pro HMM_i a neznámý vzor $\mathbf{x}$
4: **end for**
5: **return** $\underset{i}{\operatorname{argmax}}\, p_i$

Mnohdy nejsme schopni přímo zvolit parametry modelu (např. prvky matice $\mathbf{A}$), protože je dopředu neznáme. Učiníme tedy jejich iniciální odhad a necháme je adaptovat během fáze učení. Učení pak opraví náš nepřesný odhad na základě dat z trénovací množiny.

Počáteční volbou hodnot prvků matice $\mathbf{A}$ se určuje topologie přechodů mezi stavy HMM. Pro zakázání přechodu mezi dvěma stavy je potřeba nastavit příslušný prvek matice $\mathbf{A}$ na 0. V případě nenulové pravděpodobnosti je přechod povolen.

Pro učení je možné použít algoritmu *Baum-Welch*, který neposkytuje optimální řešení, ale dává dobré výsledky. Nyní popíšeme jeho základní myšlenku. Přesné odvození a bližší vysvětlení je možné najít v článku [2].

Podobně jako v dopředném algoritmu (6.1) se rekurzivně počítá proměnná $\alpha_j(t)$. Navíc je vyhodnocována proměnná $\beta_i(t)$, ale směrem od konce sekvence (tzv. zpětný algoritmus pro HMM). $\beta_i(t)$ totiž vyjadřuje pravděpodobnost, že HMM je v čase t ve stavu s_i a správně vygeneruje zbytek sekvence $x_{t+1}, \ldots, x_n$. Pro hodnotu $\beta_i(t)$ platí rovnice:

$$\beta_i(t) = \sum_{j=1}^{|S|} \beta_j(t+1) a_{ij} b_{jk(t+1)} \text{ pro všechna } t = 1, \ldots, n-1 \quad ,$$

$$\beta_i(n) = 1 \quad .$$

Pokud definujeme $\gamma_{ij}(t)$ jako:

$$\gamma_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_{jk(t+1)} \beta_j(t+1)}{\sum_{m=1}^{|S|} \sum_{n=1}^{|S|} \alpha_m(t) a_{mn} b_{nk(t+1)} \beta_n(t+1)} \quad ,$$

dostáváme odhad parametrů HMM:

$$\hat{\pi}_i = \sum_{j=1}^{|S|} \gamma_{ij}(1) \quad , \tag{6.2}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{n-1} \gamma_{ij}(t)}{\sum_{t=1}^{n-1} \sum_{m=1}^{|S|} \gamma_{im}(t)} \quad , \tag{6.3}$$

$$\hat{b}_{ij} = \frac{\sum\limits_{\substack{t=1 \\ k(t)=j}}^{n-1} \sum\limits_{m=1}^{|S|} \gamma_{im}(t)}{\sum_{t=1}^{n-1} \sum_{m=1}^{|S|} \gamma_{im}(t)} \quad . \tag{6.4}$$

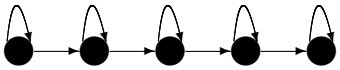
Adaptace parametrů se provádí do té doby, dokud dochází k jejich výrazné změně. Obvykle se kontroluje absolutní hodnota rozdílu staré a nové hodnoty parametrů. Pokud maximální hodnota z těchto rozdílů nepřekročí pevnou mez (např. $0,05$), učení je ukončeno. Jako ukončovací podmínka může sloužit i stanovení horní hranice pro pravděpodobnost, která se počítá na základě algoritmu 6.1.

Pro každou třídu se trénuje samostatný model (HMM). Každý z nich může mít svou specifickou topologii a počet stavů, aby co nejlépe odpovídal třídě, kterou reprezentuje. Při učení daného HMM se předkládají jen vzory z jedné třídy.

Vhodnou volbou topologie a omezením počtu spojů mezi jednotlivými stavy lze proces učení výrazně urychlit. Platí, že čím více spojů, tím jsou nároky na výpočet vyšší.

Architektura typu „*zleva-doprava*“ (obrázek 6.2) poskytuje jednoduchý příklad vhodné topologie. Stavy jsou uspořádány do posloupnosti za sebou a

spoj vede vždy jen do sousedního stavu napravo a do sebe sama. Ačkoliv se může zdát tato struktura velmi primitivní, vychází z ní i další modifikace — např. pseudo 2D HMM a hierarchický HMM (viz dále).



Obrázek 6.2: 1D HMM s architekturou typu „zleva-doprava“ — šipkami jsou vyznačeny spoje mezi stavy s nenulovou pravděpodobností přechodu

6.1.2 Spojitý HMM

Pro potřeby rozpoznávání obrázků volíme často příznakové vektory, které nemají pouze diskrétní hodnoty. Proto je nutné rozšířit základní diskrétní model na spojitý, který nemá jen omezenou abecedu výstupních symbolů a je tudíž schopen zpracovávat i spojitý signál [12, str. 6–7]

Ve spojitém případě je matice **B** nahrazena funkcí hustoty pravděpodobnosti (PDF[6]), která může být jiná pro každý vnitřní stav s_i. Jedná se o spojitou funkci, která bývá nejčastěji modelována pomocí PDF normálního rozložení (dále jen *Gaussova funkce*) nebo také konečným váženým součtem těchto PDF (tzv. *směs* Gaussových funkcí):

$$b_i(x) = \sum_{m=1}^{M} w_{im} N(\mu_{im}, \sigma_{im}) \quad ,$$

kde M je počet použitých normálních rozdělení, w_{im} váhový koeficient pro každé z nich. $N(\mu_{im}, \Sigma_{im})$ je substituce za vzorec hustoty normálního rozložení se střední hodnotou μ_{im} a rozptylem σ_{im}. Obecně platí:

$$N(\mu, \sigma)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad .$$

Parametry normálního rozložení a váhový vektor lze odhadovat pomocí klastrovacích metod (na základě dat z trénovací množiny) ještě před učením ostatních parametrů modelu.

6.1.3 Pseudo 2D HMM

Zatímco na zvukovou sekvenci lze pohlížet jako na funkci jedné proměnné (času), obrázek je svou podstatou dvojrozměrný. Rozšíření jednorozměrného

[6] angl. *probability density function*

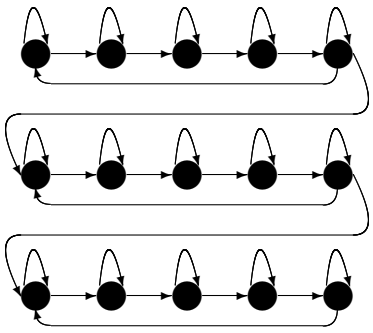
HMM do dvou dimenzí je možné provést uspořádáním jeho stavů do dvojrozměrné struktury (mřížky). Plně propojený 2D model, kde jsou umožněny přechody mezi libovolnými dvěma stavy, však vede při učení a vyhodnocování na NP-úplný problém [9]. V aplikacích vyžadujících práci v reálném čase je nepoužitelný.

Jednou z možností je zaměřit se na zpracování obrazové informace po řádcích. O tento „řádkový“ přístup se snaží *pseudo 2D HMM* (dále jen PHMM), se kterým přišel Samaria [24]. Omezení, která PHMM klade, dovolují použít polynomiální algoritmus učení.

Naše motivace použít tento model pro rozpoznávání latin a emirátských číslic vycházela hlavně z článku [18], kde byl úspěšně použit pro rozpoznávání obličejů. Pro popis znaků se stejně jako u obličejů lépe hodí příznaky, které kopírují jejich dvojrozměrnou povahu.

PHMM je tvořen logicky oddělenými skupinami stavů (tzv. *řádkové skupiny* stavů). Stavy v rámci jedné skupiny spolu tvoří HMM typu „zleva-doprava“ (viz obrázek 6.2) a používají se pro zpracování jednotlivých řádků vstupu. Počty stavů v jednotlivých řádkových skupinách se mohou lišit a zapsány za sebou (shora dolů) tvoří tzv. *stavový zápis* (např. 5-5-5).

Na konci každého řádku vstupu pak dochází k volbě, zda zpracovávat následující řádek stejnou skupinou stavů nebo „přepnout“ na následující skupinu. Ve své podstatě se jedná o 1D HMM se speciálně volenou topologií propojení jednotlivých stavů (obrázek 6.3).

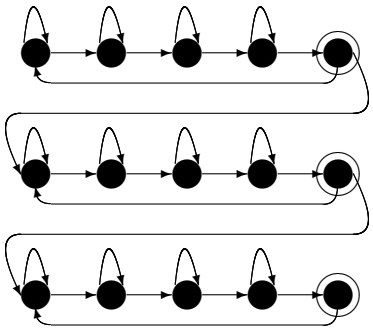


Obrázek 6.3: Pseudo 2D HMM se třemi řádkovými skupinami a stavovým zápisem 5-5-5

Aby mohl HMM lépe rozlišit, kde končí jednotlivé „řádky“ příznakového vektoru, přidává se na konec každého řádku speciální symbol (hodnota), který se v datech nevyskytuje. Příznakový vektor je tak logicky rozčleněn na úseky odpovídající řádkům.

Speciální symbol musí být vždy zpracován zvláštním stavem pro konec řádku (zakroužkované stavy na obrázku 6.4). Tyto zvláštní stavy mají jako

jediné nenulovou pravděpodobnost (resp. rovnou 1), že vygenerují speciální symbol. Jiné symboly tyto stavy vygenerovat nemohou.



Obrázek 6.4: Pseudo 2D HMM se speciálními stavy pro konce řádků ve třech řádkových skupinách

Navzdory jistým odlišnostem, vykazují oba modely podle článku [24] podobné vlastnosti a schopnosti naučit se rozpoznávat předložené vzory.

Důvodem, proč je obraz zpracováván po řádcích a ne po sloupcích, je větší stabilita vůči zkosení písma. Tato výhoda se projeví zejména při špatné korekci sklonu textu vlivem špatné segmentace nebo při rozpoznávání více druhů písma [7, str. 843], které se obvykle liší jen svým sklonem (normální vs. kurzíva).

6.1.4 Hierarchický HMM

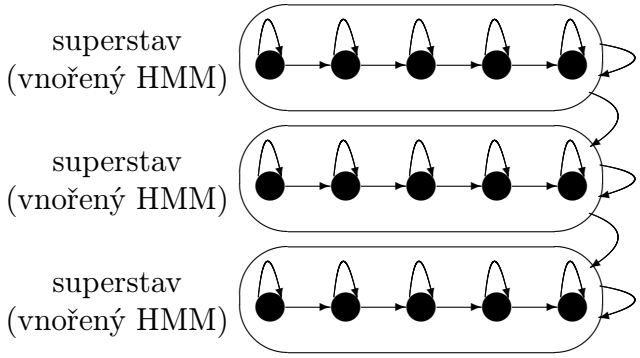
Samostatnou skupinu tvoří hierarchické HMM[7] [18], které jsou typické svým víceúrovňovým uspořádáním. Někdy jsou také označovány jako pseudo 2D HMM [7], ale toto označení koliduje s označením pro zcela jiný typ HMM (popsaný v předchozí sekci).

Budeme se zabývat jen těmi hierarchickými modely, které jsou tvořeny dvěma úrovněmi (obrázek 6.5). Na nejnižší úrovni jsou jednoduché jednorozměrné HMM (obvykle typu „zleva-doprava“). Každý z nich pak tvoří na vyšší úrovni samostatný stav (tzv. *superstav*) hierarchicky nadřazeného HMM.

V případě hierarchického HMM udávají jednotlivá čísla stavového zápisu počty stavů ve vnořených HMM (shora dolů).

Dvojrozměrná struktura dat zůstává zachována bez potřeby speciálních koncových stavů. Výstup vydávají pouze vnořené HMM a zpracovávají jednotlivé „řádky“ vstupu. Nadřazený HMM slouží pro „přepínání“ mezi vnořenými HMM na konci každého „řádku“ (tzn. zpracovává data ve vertikálním směru). Upravený dopředný algoritmus je možné nalézt v článku [7].

[7] angl. *Embedded HMM*



Obrázek 6.5: Hierarchický HMM se třemi superstavy a stavovým zápisem 5-5-5 — první úroveň tvoří tři vnořené HMM a druhou úroveň jeden nadřazený HMM

Tento model se liší od plně propojeného 2D HMM hlavně tím, že nejsou povoleny přechody mezi stavy dvou různých vnořených HMM. Tím dosahuje relativně malého počtu spojů a rychlého procesu učení pomocí upraveného algoritmu pro jednorozměrné HMM [18].

Výsledkem jednoho pozorování nemusí být jen jedna hodnota, ale celý vektor hodnot (tzv. *pozorovací vektor*). Příznakový „vektor“ má v tomto případě podobu matice. Pro každý stav vnořeného spojitého HMM se použije vícerozměrná PDF v závislosti na rozměrech pozorovacího vektoru. To odpovídá směsi vícerozměrných normálních rozložení.

6.2 Výsledky

Úspěšnost rozpoznání číslic pomocí skrytých Markovových modelů závisí na volbě příznaků a vhodné architektury modelu. Pro každý typ příznakového vektoru se lépe hodí jiná architektura, aby dokázala co nejlépe využít závislostí mezi jeho jednotlivými složkami.

Některé typy příznaků jsou pro HMM nevhodné. Příkladem může být *NCM*, kde na sobě hodnoty jednotlivých složek téměř nezávisí. I přesto však nacházejí tyto příznaky uplatnění, ale pouze jako dílčí pozorovací vektory pro hierarchické HMM.

Rozsah hodnot příznaků také určuje, jestli může být pro rozpoznávání použito diskrétního nebo spojitého modelu. Diskrétní model je vhodný pouze pro příznaky, jejichž jednotlivé složky mohou nabývat několika málo hodnot. Těmto hodnotám potom odpovídají příslušné výstupní symboly.

Rozumný počet výstupních symbolů se přitom pohybuje řádově v desítkách. S rostoucím počtem symbolů se zvyšuje velikost modelu a výpočetní

náročnost. Dobré vlastnosti pro diskrétní model mají příznaky typu *Pix*, které připouští pouze dvě výstupní hodnoty.

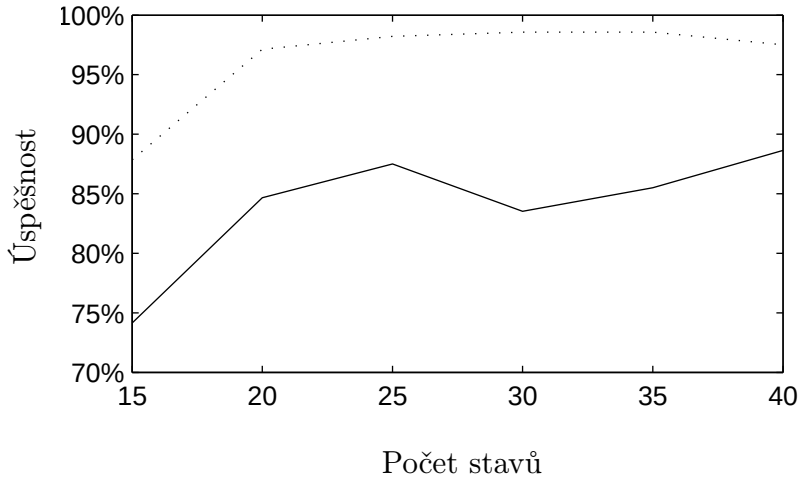
Postupně jsme zkoušeli různé topologie a architektury. Pro každou z nich jsme se snažili nalézt nejvhodnější počet stavů. Spojité modely jsme navíc testovali s různými počty Gaussových funkcí pro odhad PDF v jednotlivých stavech.

6.2.1 1D HMM

Jednorozměrný model je vhodný v případě, pokud mají vstupní data jednorozměrný charakter. Předpokládá závislost pouze dvou po sobě jdoucích pozorování, ale ostatní závislosti (např. prostorové) ignoruje. Preferovali jsme architekturu typu „zleva-doprava“, která je v tomto případě nejčastěji používána (např. [18, str. 1]).

Vhodným typem jednorozměrných příznaků jsou *Proj*. I když je uprostřed příznakového vektoru porušena závislost dvou po sobě jdoucích složek (kvůli přechodu z vertikálních na horizontální projekce), výsledky rozpoznání byly celkem uspokojivé.

Zjistili jsme, že rozložení dat nejlépe vystihuje lineární kombinace tří Gaussových funkcí. Odhad jejich parametrů byl stanoven během fáze učení. Při hledání nejvhodnějšího počtu stavů jsme vycházeli z grafu na obrázku 6.6.



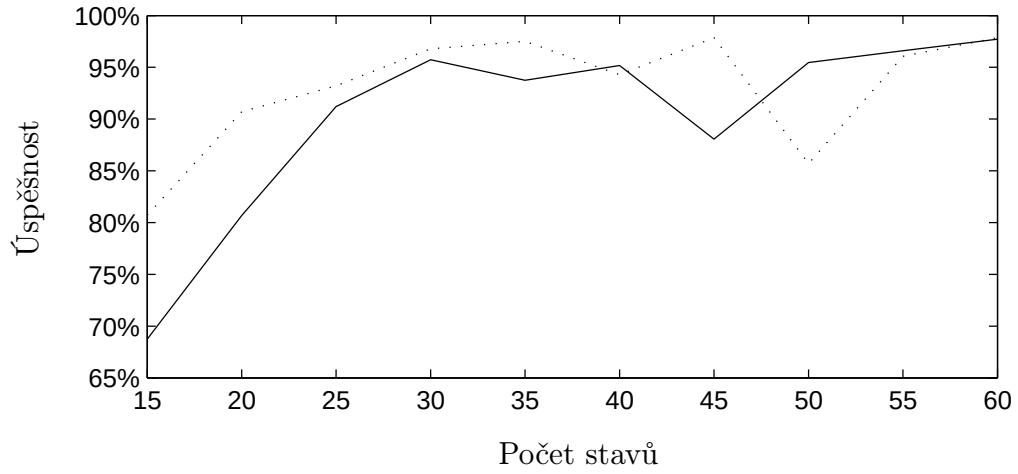
Obrázek 6.6: Úspěšnost rozpoznání na trénovací množině při použití jednorozměrného HMM s příznaky typu *Proj* v závislosti na počtu stavů — latin (resp. emirátským) číslicím odpovídá plná (resp. přerušovaná) čára

Markovovy modely ve spojení s příznaky *Proj* lépe rozpoznávaly emirátské číslice (podobně jako neuronové sítě). Rozdíly v úspěšnosti rozpoznání latin a emirátských číslic se pohybovaly okolo 10% procent. Procenta správně rozpoznaných emirátských číslic v nejlepším případě vypadala následovně:

- 98% — na trénovací množině
- 93% — na ověřovací množině
- 76% — na problémové množině

Příznakový vektor typu *Win* byl v článku [6] také rozpoznáván jednorozměrným Markovovým modelem. Pro potřeby tohoto modelu byl obrázek procházen oknem střídavě zleva doprava a zprava doleva, aby byla zachována „spojitost“ jednotlivých měření.

Zatímco u příznaků typu *Proj* se s přidáváním Gaussových funkcí úspěšnost zvyšovala, pro příznaky *Win* jsme zjistili, že v nejlepším případě stačí pouze jediná Gaussova funkce.



Obrázek 6.7: Úspěšnost rozpoznání na trénovací množině při použití jednorozměrného HMM s příznaky typu *Win* v závislosti na počtu stavů — latin (resp. emirátským) číslicím odpovídá plná (resp. přerušovaná) čára

V grafu na obrázku 6.7 je možné sledovat celkem vysoké procento úspěšně rozpoznaných číslic pro oba typy číslic. Počáteční nárůst úspěšnosti je pro zvyšující se počet stavů nahrazen relativně vyrovnaným průběhem. Tabulka 6.1 ukazuje nalezené modely s nejvyšší úspěšností na trénovací množině.

typ	trénovací	ověřovací	problémová
latin	98%	92%	75%
emirátské	98%	96%	77%

Tabulka 6.1: Úspěšnost při použití 1D HMM a příznaků typu *Win* — pro latin číslice použito 60 stavů a pro emirátské 45 stavů

Při trénování jsme použili deset iterací algoritmu Baum-Welch. Po dalších deseti iteracích se úspěšnost na trénovací množině zvedla jen zanedbatelně, ale zaznamenali jsme pokles na ověřovací a problémové množině.

Pokusili jsme se sestavit různorodý soubor modelů ze všech doposud testovaných. Zaměřili jsme se pro jednoduchost pouze na latin číslice. Za základ jsme zvolili natrénované modely s 60 stavy. Modely, které byly na ověřovací množině neúspěšné, jsme nahradili jinými. Konkrétně se jednalo jen o model patřící číslici 6, kterou se podařilo rozpoznat jen v jednom případě z pěti.

Jako náhradníky za HMM (s 60 stavy) pro číslici 6 jsme volili modely s jiným počtem stavů, ale naučené rozpoznávat stejnou číslici. Navíc nám šlo o to vybrat ty, které danou číslici na ověřovací množině bezchybně rozpoznaly. V případě více kandidátů jsme se přikláněli k modelu s větší úspěšností na problémové množině.

Ve srovnání se souborem modelů s 60 stavy však počet správně rozpoznaných exemplářů číslice 6 klesl na nulu (při použití různorodého souboru na ověřovací množině). Uvědomili jsme si, že je nutné vysledovat a odstranit ze souboru dominantní jedince (HMM), kteří kazí správné rozpoznání této číslice.

Protože číslice 6 byla nejčastěji zaměňována s číslicí 5 a 8, nahradili jsme odpovídající modely v souboru jinými s menší úspěšností na ověřovací množině. Testy s takto získaným souborem na ověřovací množině nám však potvrdily, že ačkoliv číslice 6 už byla bezchybně klasifikována, chyby se objevily u jiných číslic. Proto jsme tento postup zavrhli a dále jsme se tím nezabývali.

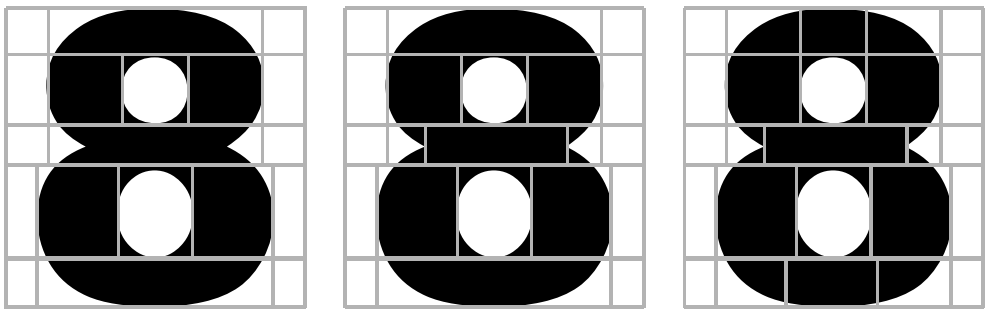
Přestože se jednorozměrný Markovův model naučil správně rozlišovat číslice z trénovací množiny, celkově se ukázal jako nevhodný pro rozpoznávání problémových číslic. Jeho schopnost zobecnit předložené trénovací vzory je slabá.

6.2.2 Pseudo 2D HMM

S příznaky typu *Pix* jsme mohli prakticky vyzkoušet „řádkové“ zpracování vstupu pomocí diskrétního PHMM. Nejdříve jsme nechali model, aby konce jednotlivých řádků odhadoval sám (model na obrázku 6.3). Výstupní symboly nám tedy stačily dva.

Stavy jsme rozdělili do několika řádkových skupin pro zpracování jednotlivých řádků obrázku. Počty stavů v jednotlivých skupinách jsme odvodili na základě tvaru rozpoznávaných číslic. Postup si demonstrujeme na latin číslici 8.

Obrázek číslice rozdělíme do pěti vodorovných oblastí s podobnou charakteristikou rozložení pixelů v řádku (obrázek 6.8). Každou z těchto oblastí ještě můžeme dále rozdělit na menší sektory v závislosti na tom, z jaké části jsou zaplněny — hrubé dělení (obrázek 6.8a), střední dělení (obrázek 6.8b) a jemné dělení (obrázek 6.8c).



(a) 3-5-3-5-3 (b) 3-5-5-5-3 (c) 5-5-5-5-5

Obrázek 6.8: Odvození počtu stavů pro pseudo 2D HMM — pod každým obrázkem je stavový zápis odpovídajícího PHMM

Vodorovným oblastem jsou přiděleny řádkové skupiny stavů a sektorům konkrétní stavy. Například HMM se stavovým zápisem 3-5-5-5-3 odpovídá motivačnímu obrázku 6.8b (celkem je zapojeno 21 stavů).

I přes experimentování s různými počty stavů, bylo procento úspěšně rozpoznaných číslic na trénovací množině velmi malé.

Abychom více zdůraznili konce řádků, přidali jsme k pravému okraji binárního obrázku nový sloupec se speciálními symboly (třetí výstupní symbol). Příznakový vektor zapsaný po řádcích zleva doprava jsme tak rozdělili na úseky odpovídající jednotlivým řádkům obrázku.

Na konec každé skupiny stavů jsme umístili zvláštní stav (obrázek 6.4), který jediný může dávat na výstup speciální symbol a žádný jiný. Má tedy za úkol rozhodnout, zda další „řádek“ příznakového vektoru zpracovávat stejnou skupinou stavů nebo použít následující skupinu.

Porovnání těchto dvou přístupů diskrétního pseudo 2D HMM je možné nalézt v tabulce 6.2. Pro latin číslice přineslo zavedení speciálních symbolů nepatrné zlepšení, avšak pro emirátské číslice došlo k nárůstu až o 30%. A to i navzdory tomu, že Samaria v článku [24] uvádí, že oba modely mají poměrně shodné rozpoznávací schopnosti.

stavový zápis	EOL	latin	emirátské
5-5-5-5-5	ne	29%	37%
3-5-5-5-3	ne	37%	40%
3-5-5-5-3	ano	41%	70%

Tabulka 6.2: Porovnání úspěšnosti rozpoznání vzorů z trénovací množiny při použití příznaků *Pix* a pro různé typy diskrétních pseudo 2D HMM — EOL označuje, jestli byly použity speciální symboly pro konce řádků

Diskrétní modely jsme tedy označili za nedostatečné pro přesnější rozpoznávání a zaměřili jsme se na zkoumání spojitých modelů ve spojení s příznaky typu *Win*. Využili jsme přitom modelu bez speciálního stavu pro konce řádků (obrázek 6.3).

Vycházeli jsme z již zjištěného faktu, že rozložení dat tohoto typu příznaků nejlépe kopíruje jediná Gaussova funkce, a z přibližného počtu stavů, který jsme odvodili (obrázek 6.8).

Narozdíl od diskrétního modelu vycházelo na základě pokusů nejlépe použití HMM se stavovým zápisem 5-5-5-5-5. Při porovnání tabulek 6.1 a 6.3 je vidět, že i přes dvojrozměrný charakter dat byl jednorozměrný model o něco úspěšnější.

typ	trénovací	ověřovací	problémová
latin	83%	80%	61%
emirátské	94%	80%	71%

Tabulka 6.3: Úspěšnost při použití pseudo 2D HMM se stavovým zápisem 5-5-5-5-5 a příznaků typu *Win*

Zkoušeli jsme také nahradit jednorozměrný model typu „zleva-doprava“ pomocí pseudo 2D modelu (bez stavů pro konce řádků) s jedinou řádkovou skupinou stavů, ale nepřineslo to žádné zlepšení klasifikace. Úspěšnost jednorozměrného modelu zůstala nepřekonána.

6.2.3 Hierarchický HMM

Nejprve jsme hierarchický model otestovali na již známých příznacích typu *Win*. Jedno pozorování tvoří vždy pouze jediná hodnota a společně dávají dohromady příznakový vektor. Ve srovnání s předchozími variantami HMM a s použitím stejných příznaků nám vycházely lepší výsledky pouze pro emirátské číslice (tabulka 6.4).

Při použití příznaků typu *DCT* jsme vycházeli z článku [18]. Pro získání koeficientů jsme použili posuvné okno 12×12 (podobně jako u příznaků *Win*). Během procházení obrázku po řádcích se vždy sousední pozice okna překrývaly čtyřmi sloupci nebo čtyřmi řádky.

Pro oblasti obrázku definované pozicí posuvného okna byly spočítány koeficienty diskrétní kosinové transformace [29, str. 37]. Z nich jsme vybrali výřez jen těch nejdůležitějších. V článku [18] je použit výřez o rozměrech 3×3, my jsme však experimentovali i s výřezy jiných velikostí.

Ačkoliv jsme vycházeli z metody pro rozpoznávání obličejů, metoda se ukázala jako výborná i pro rozpoznávání číslic. S tímto typem příznaků měly námi testované hierarchické HMM vysokou úspěšnost:

	latin			emirátské		
typ	T	O	P	T	O	P
1D HMM	98%	92%	75%	98%	96%	77%
pseudo 2D HMM	94%	80%	71%	83%	80%	61%
hierarchický HMM	84%	80%	58%	98%	93%	78%

Tabulka 6.4: Porovnání úspěšnosti hierarchického HMM se stavovým zápisem 6-6-6-6-6-6 s jinými typy HMM za použití příznaků typu *Win* — sloupec T značí procenta správně rozpoznaných číslic na trénovací, O na ověřovací a P na problémové množině

- 99–100% rozpoznaných číslic na trénovací množině
- 96–100% rozpoznaných číslic na ověřovací množině
- 80–94% rozpoznaných číslic na problémové množině

Pro porovnání uvedeme výsledky některých z testovaných hierarchických HMM (tabulka 6.5). Hvězdičkou (*) je označen model, který byl se stejnými parametry použit v článku [18] na rozpoznávání obličejů.

			latin			emirátské		
typ	výřez	směs	T	O	P	T	O	P
3-6-6-6-3 *	3×3	3	100%	98%	82%	99%	96%	82%
3-5-5-5-3	4×4	4	100%	100%	93%	99%	96%	85%
5-5-5-5-5	5×5	5	100%	100%	93%	100%	100%	80%

Tabulka 6.5: Výsledky hierarchických HMM s příznaky typu *DCT* s použitím výřezů různých velikostí pro získávání koeficientů DCT — sloupec „směs“ udává počet použitých Gaussových funkcí, sloupec T značí procenta správně rozpoznaných číslic na trénovací, O na ověřovací a P na problémové množině

Zkoušeli jsme také použít jiné typy pozorovacího vektoru. S příznaky typu *Pix* to byly samostatné hodnoty pixelů v jednotlivých bodech obrázku.

Snažili jsme se také sloučit metodu procházení oknem (u příznaků *Win*) s příznaky typu *NCM*. Pro každý výřez obrázku definovaný pozicí okna byly jako pozorovací vektor vyčísleny hodnoty normalizovaných centrálních momentů až do řádu 7.

Lepší klasifikace číslic než s příznaky *DCT* jsme už nedosáhli.

6.2.4 Věrohodnost rozpoznání

Rozhodli jsme se zabývat kontrolou věrohodnosti rozpoznaných číslic pouze u hierarchických HMM, protože s použitím příznaků *DCT* dávaly dobré výsledky. Zvolili jsme stejný přístup k hodnocení věrohodnosti jako jsme použili u dopředných neuronových sítí.

Při klasifikaci dvojice navzájem si odpovídajících latin a emirátských číslic jsou do vektoru $\mathbf{y}^{lat}$ uloženy pravděpodobnosti získané algoritmem 6.1 pro skryté Markovovy modely jednotlivých latin číslic. Podobně jsou do vektoru $\mathbf{y}^{em}$ zapsány pravděpodobnosti pro modely emirátských číslic.

Výsledný výstup se spočítá ze vzorce 5.15. Pro klasifikaci dvojice (navzájem si odpovídající latin a emirátské číslice) do třídy i se použije vztahu 5.13. Získané výsledky při použití různých prahových hodnot T_m a T_d jsou přehledně zobrazeny v tabulce 6.6.

T_m	T_d	+	?	−
0,5	0	86%	1%	13%
0,5	0,05	82%	9%	9%
0,5	0,1	82%	10%	8%
0,5	0,2	57%	36%	7%
0,55	0	87%	1%	12%
0,55	0,05	85%	6%	9%
0,55	0,1	57%	36%	7%
0,55	0,2	45%	51%	4%
0,6	0	85%	4%	11%
0,6	0,05	85%	7%	8%
0,6	0,1	57%	37%	6%
0,6	0,2	45%	52%	3%

Tabulka 6.6: Výsledky rozpoznávání SPZ pro hierarchické HMM se stavovým zápisem 5-5-5-5-5 (+ správně, − špatně a ? nerozpoznané SPZ)

Věrohodnost rozpoznávání SPZ jsme testovali na hierarchických HMM se stavovým zápisem 5-5-5-5-5, jejichž další parametry a výsledky samostatného rozpoznávání číslic jsou uvedeny v tabulce 6.5 na třetím místě.

Ověření věrohodnosti rozpoznaných číslic dopadlo podstatně hůře než u neuronových sítí. Procento špatně rozpoznaných SPZ bylo vyšší než můžeme v reálném prostředí připustit. Úbytek správně rozpoznaných SPZ s rostoucím parametrem T_d byl mnohem rychlejší než úbytek procenta špatně klasifikovaných SPZ. Proto v případech, kdy byla splněna nerovnice 2.4, byl počet správně rozpoznaných SPZ nízký.

Pro testované hierarchické HMM vychází z prahových hodnot nejlépe (při splnění nerovnice 2.4) parametry $T_m = 0,5$ a $T_d = 0,1$.

7 Závěr

7.1 Srovnání metod rozpoznávání číslic

Všechny testy zkoumaných metod byly prováděny na osobním počítači s procesorem Intel Celeron (800MHz), pamětí 256MB RAM a operačním systémem Windows XP Professional. Učení klasifikátorů (např. neuronové sítě) probíhalo na školních strojích s procesory AMD Opteron 144 (1.8GHz), pamětí 1024MB RAM a operačním systémem Gentoo Linux.

Při porovnání testovaných metod (tabulka 7.1) se jako nejvhodnější metoda pro danou úlohu rozpoznávání číslic na SPZ jeví třívrstvé neuronové sítě ve spojení s příznaky typu *Win*.

metoda	latin	emirátské	rychlost
template matching	93%	89%	751ms
neuronové sítě	97%	89%	320ms
hierarchické HMM	93%	85%	2974ms

Tabulka 7.1: Srovnání úspěšnosti vybraných metod pro rozpoznávání číslic SPZ na problémové množině — pro porovnání byla z každé metody vybrána varianta s největší úspěšností na problémové množině, poslední sloupec udává průměrný čas nutný pro klasifikaci všech vzorů z problémové množiny

Neuronové sítě měly nejvyšší procento správně rozpoznaných číslic na trénovací, ověřovací i problémové množině. Zároveň poskytovaly dostatečnou jistou v otázce věrohodnosti rozpoznaných číslic. Neuronové sítě byly dominantní také v rychlosti klasifikace, což je velkou výhodou pro aplikaci, která ma běžet v reálném provozu.

Hierarchické HMM byly při klasifikaci číslic poměrně pomalé. Při rozšiřování modelu o nové stavy nebo zvyšováním počtu Gaussových funkcí rychle narůstaly výpočetní nároky na trénování i testování. Na druhou stranu do-

sáhly s příznaky typu *DCT* vysokého procenta rozpoznaných číslic (srovnatelného s neuronovými sítěmi).

Klasifikátor minimální vzdálenosti s příznaky typu Win (pro latin číslice) a *Proj* (pro emirátské číslice) dokázal rozpoznat většinu předložených číslic. Čím větší však bylo procento správně rozpoznaných číslic, tím větší byl počet špatně klasifikovaných číslic (nerozpoznaných číslic ubývalo). Pro systém automatického rozpoznávání SPZ je procento chybných klasifikací klíčovou hodnotou, a tak nemůže být klasifikátor minimální vzdálenosti nasazen v reálném prostředí.

7.2 Zhodnocení

Na základě specifikace naší úlohy jsme provedli analýzu problému a proces rozpoznávání SPZ jsme rozdělili do několika fází. Jednotlivé fáze jsme zpracovali s ohledem na problémy, které se vyskytují v reálném prostředí.

Nejdříve jsme se zaměřili na předzpracování obrazové informace a navrhli jsme vhodné kroky pro prevenci nebo odstranění některých nepříjemných efektů.

Lokalizaci SPZ ve snímku se nám nepodařilo uspokojivě vyřešit, i když jsme vyzkoušeli více technik a přístupů z různých odborných článků. Usoudili jsme, že proces vyhledávání SPZ ve snímku je lepší rozdělit do dvou kroků (hrubá a jemná lokalizace). Přesnost označení oblasti SPZ u testovaných metod nebyla dostačující, a tak musí být rohy SPZ označovány manuálně.

Na základě barevné informace obsažené v obrázku SPZ byl náš algoritmus schopen poměrně přesně rozhodnout, o jaký typ SPZ se jedná. Pro přesně označenou oblast SPZ bylo automaticky rozpoznáno relativně vysoké procento typů předložených SPZ.

Přístup, který jsme zvolili pro segmentaci jednotlivých číslic, se ukázal jako prakticky použitelný. Využili jsme vzájemné pozice latin a emirátských číslic, abychom z nich vytvořili páry odpovídajících číslic. Snažili jsme se tak maximálně zužitkovat redundantní zápis identifikačního údaje SPZ.

Pro rozpoznávání číslic jsme vyzkoušeli více druhů příznaků, abychom co nejlépe popsali obrázky normalizovaných latin a emirátských číslic. Částečně jsme vyřešili problémem testování věrohodnosti rozpoznaných číslic a vymysleli jsme způsob, jak věrohodnost hodnotit.

Nastudovali jsme teorii a na zadanou úlohu rozpoznávání číslic SPZ jsme aplikovali klasifikátory různých specifických vlastností — klasifikátor minimální vzdálenosti, neuronové sítě a skryté Markovovy modely. Všechny klasifikátory jsme testovali na reálných datech.

Zabývali jsme se zkoumáním různých modifikací uvedených klasifikátorů a hodnocením získaných výsledků, abychom vybrali vhodnou metodu pro řešenou úlohu. Při porovnávání metod jsme sledovali nejen jejich úspěšnost, ale i věrohodnost klasifikace.

Díky zpracování problému z reálného prostředí jsme si z celé práce odnesli cenné zkušenosti, které se od „učebnicových“ příkladů diametrálně liší.

7.3 Možnosti dalšího vývoje

Na tuto práci lze navázat navržením a implementací vhodného způsobu lokalizace SPZ. Metoda musí být dostatečně rychlá a měla by poskytovat precizní segmentaci oblasti SPZ.

Všechny zkoumané klasifikátory měly problémy s rozlišením emirátských číslic 2 a 3, které jsou vizuálně velmi podobné. Další vývoj by měl směřovat k navržení lepších příznaků, které dokáží tyto číslice lépe odlišit, ale současně budou dobře popisovat ostatní číslice.

Systém by bylo dobré zobecnit, aby dokázal lépe rozlišovat cizí SPZ a rozpoznávat SPZ i z okolních arabských emirátů (okolí Abu Dhabi). Jeho využití se tím rozšíří.

Je potřeba, aby systém dokázal rychle zpracovávat velké množství vstupních dat. Proto je nezbytné implementované algoritmy optimalizovat a „vyladit“ pro lepší výkon. Nabízí se zde využití grafických procesorů a implementace některých technik zpracování obrazu pomocí tzv. pixel shaderů.

Jedním z modulů systému automatického rozpoznávání SPZ by mohlo být rozpoznávání typu (výrobce) a barvy automobilu. Srovnáním této informace a rozpoznané SPZ s databází by tento systém mohl pomoci s hledáním kradených vozidel.

Do záznamu o dopravním přestupku se vkládá také fotografie řidiče získaná jako výřez z čelního snímku automobilu. Dalším modulem by tedy mohla být automatická lokalizace oblasti, kde se nachází obličej řidiče. Jako rozšíření potom může nastoupit rozpoznávání obličejů společně s databází hledaných osob.

Zajímavé by mohlo být zabývat se rozpoznáváním SPZ z krátké videosekvence (namísto jediného snímku) a využít podobnosti mezi jednotlivými snímky.

A Přílohy

A.1 Obsah CD

Stromová struktura adresářů na přiloženém CD je následující:

- adresář `bin`
  - adresář `spz` — ukázkové programy (viz dále)
- adresář `images`
  - adresář `digits` — binární obrázky normalizovaných číslic
    - adresář `bad` — problémová množina číslic
    - adresář `train` — trénovací množina číslic
    - adresář `verify` — ověřovací množina číslic
  - adresář `full`
    - adresář `bad` — nevydařené fotografie vozidel
    - adresář `ok` — vydařené fotografie vozidel
  - adresář `spz` — obrázky malých výřezů obsahujících SPZ
- adresář `src`
  - adresář `spz` — zdrojový kód ukázkových programů
- soubor `dp.pdf` — tato diplomová práce v elektronické podobě

První znak z názvu libovolného souboru v adresáři `images\digits` udává, zda se jedná o obrázek latin (`L`) nebo emirátské (`E`) číslice. Dalším znakem je číslice, která je na obrázku. Zbylé znaky v názvu souboru nejsou rozhodující. Například `E1_1143721412.bmp` je soubor s binárním obrázkem normalizované emirátské číslice 1.

Pro každý bitmapový obrázek SPZ v adresáři `images\spz` existuje na CD textový soubor s příponou `spz`, kde je na prvním řádku uložen skutečný identifikační údaj SPZ. Druhý řádek popisuje typ této SPZ (číselná interní reprezentace).

A.2 Ukázkové programy

Pro demonstraci postupů a metod prezentovaných v této diplomové práci jsme vytvořili několik ukázkových programů. Nachází se na přiloženém CD v adresáři `bin\spz`. Programy pro svůj chod potřebují všechny soubory v tomto adresáři. Ukázkové programy je nutné spouštět z příkazové řádky operačního systému Windows (testováno na systému Windows XP). Příklad spuštění ukázkového programu:

```
spz_hmm.exe ..\..\images\spz\2.bmp ..\..\images\spz\3.bmp
```

Všechny ukázkové programy mohou mít více argumentů, kterými jsou cesty k souborům s bitmapovými obrázky (s příponou `bmp`). Vstupem programů (kromě programu `spz_raw.exe`) by měly být malé výřezy obsahující SPZ (jsou připraveny v adresáři `images\spz`). Program `spz_raw.exe` na vstupu očekává fotografie vozidel pořízené systémem pro monitorování rychlosti. Tyto fotografie jsou nachystány v adresáři `images\full\ok`.

Většina z přiložených programů (např. `spz_sgm.exe`) využívá tzv. grafické okno pro zobrazení mezivýsledku uživateli. Poté program čeká na stisknutí klávesy, aby mohl pokračovat ve výpočtu. V okamžiku stisknutí klávesy musí být aktivní grafické okno, jinak se klávesa nevyhodnotí.

Programy (kromě `spz_fnd.exe` a `spz_raw.exe`) navíc mohou po uživateli požadovat, aby provedl manuální lokalizaci oblasti SPZ. Ve chvíli, kdy je k tomu vyzván, musí myší v grafickém okně označit čtyři rohy oblasti SPZ. Nejprve levý horní roh a potom po směru hodinových ručiček ostatní rohy. Nakonec musí stisknout klávesu.

Výsledek lokalizace oblasti SPZ se uloží do textového souboru s příponou `candidate`, jehož jméno odpovídá jménu právě zpracovávaného souboru obrázku s výřezem SPZ. Pokud pro daný obrázek existuje soubor s příponou `candidate`, manuální lokalizace se neprovádí a použijí se údaje z tohoto souboru.

Kromě grafického okna používají programy k výstupu příkazového řádku. V textovém režimu zobrazují informace o rozpoznané SPZ (mimo program `spz_raw.exe`) — identifikační údaj, barvu a typ. Symbol `?` v identifikačním údaji značí, že daná číslice nebyla věrohodně rozpoznána.

Nyní stručně popíšeme účel jednotlivých ukázkových programů:

- program `spz_fnd.exe` — pokusí se automaticky nalézt a rozpoznat SPZ v malém výřezu (jemná lokalizace SPZ na str. 15) metodou projekcí. Pro rozpoznávání jsou zde použity neuronové sítě.

- program `spz_hmm.exe` — využívá manuální lokalizace SPZ a rozpoznávání pomocí hierarchických HMM
- program `spz_net.exe` — využívá manuální lokalizace SPZ a rozpoznávání pomocí neuronových sítí
- program `spz_raw.exe` — slouží pro předvedení hrubé lokalizace oblasti SPZ metodou popsanou v této práci (str. 14). Zelená linka označuje řádky s vysokou hodnotou horizontální projekce. Červená úsečka představuje posuvné okno v pozici, kdy protíná velké množství vertikálních hran. Nalezený výřez se dále nezpracovává.
- program `spz_sgm.exe` — zachycuje jednotlivé kroky segmentace číslic v oblasti SPZ. Oblast SPZ je lokalizována manuálně a číslice jsou rozpoznávány neuronovými sítěmi.

V případě neuronových sítí byly použity architektury 81-24-10 (pro latin číslice) a 81-22-10 (pro emirátské číslice) společně s příznaky typu *Win*. Pro hierarchické HMM jsme vybrali příznaky typu *DCT*. Použili jsme model se stavovým zápisem 5-5-5-5-5, jehož další parametry jsou uvedeny v tabulce 6.5 (str. 75) na třetím řádku.

A.3 Použité nástroje

Při testování a implementaci popsaných metod jsme použili tyto nástroje a knihovny:

- Microsoft Visual C++ 6.0
- OpenCV beta 5
  - `http://www.sourceforge.net/projects/opencvlibrary`
- MATLAB 7.0 — Neural Network Toolbox
- BNT (Bayes Net Toolbox)
  - `http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html`
- NETLAB 3.3
  - `http://www.ncrg.aston.ac.uk/netlab/index.php`

Pro vysázení diplomové práce byl použit systém LaTeX s českou lokalizací.

Literatura

[1] L. E. Baum, T. Petrie: *Statistical inference for probabilistic functions of finite state Markov chains*, Annals of Mathematical Statistics, 1966

[2] L. E. Baum, T. Petrie, G. Souled, N. Weiss: *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Annals of Mathematical Statistics, 1970, vol. 41, no. 1, str. 164–171

[3] S. Draghici: *A neural network based artificial vision system for licence plate recognition*, Dept. of Computer Science, Wayne State University

[4] R. O. Duda, P. E. Hart, D. G. Stork: *Pattern Classification*, druhé vydání, Wiley, 2003

[5] G. Hjaltason, H. Samet: *Ranking in spatial databases*, 4th Symposium on Spatial Databases (SSD), 1995

[6] J. H. Chiang: *Building an Automatic Vehicle License-Plate Recognition System*, RIVF, 2005, str. 59–63

[7] S. Kuo, O. E. Agazzi: *Keyword spotting in poorly printed documents using Pseudo 2-D Hidden Markov models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-16(8), 1994, str. 842–848

[8] K. Levenberg: *A Method for the Solution of Certain Problems in Least Squares*, SIAM J. Appl. Math. 11, 1963, str. 431–441

[9] E. Levin, R. Pieraccini: *Dynamic planar warping for optical character recognition*, lCA.SSP, 1992

[10] D. Llorens, A. Marzal, V. Palazón, J. M. Vilar: *Car License Plates Extraction and Recognition Based on Connected Components Analysis and HMM Decoding*, IbPRIA, 2005, str. 571–578

[11] Z. Lu, I. Bazzi, A. Kornai, J. Makhoul, P. Natarajan, R. Schwartz: *A Robust, Language-Independent OCR System*, 27th AIPR Workshop, 1998

[12] S. Marchand-Maillet: *1D and Pseudo-2D Hidden Markov Models for Image Analysis: Theoretical Introduction*, Technical Report RR-99-49 Part A, 1999

[13] D. Marquardt: *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, Quart. Appl. Math. 2, 1944, str. 164–168

[14] R. C. P. Marques, F. N. S. Medeiros, J. L. Silva, C. M. Laprano: *License Vehicle Plates Localization Using Maximum Correlation*, SSPR&SPR 2004, str. 470–476

[15] C. D. Meyer: *Matrix Analysis And Applied Linear Algebra*, SIAM, 2000

[16] W. McCulloch, W. Pitts: *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, 1943, vol. 5, str. 115–133

[17] D. Michalopoulos, Ch. K. Hu: *An Error Back-Propagation Artificial Neural Networks Application in Automatic Car License Plate Recognition*, IEA/AIE, 2002

[18] A. V. Nefian, M. H.Hayes: *Face Recognition Using An Embedded HMM*, IEEE Conference on Audio and Video-based Biometric Person Authentication, 1999

[19] M. Novák a kolektiv: *Umělé neuronové sítě*, C. H. Beck, Praha, 1998

[20] C. Oz, F. Ercal: *A Practical License Plate Recognition System for Real-Time Environments*, IWANN, 2005

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, : *Numerical Recipes in C*, druhé vydání, Cambridge University Press, 1992

[22] M. Riedmiller, H. Braun: *A direct adaptive method for faster backpropagation learning: The Rprop algorithm*, Proceedings of the IEEE International Conference on Neural Networks, IEEE Press, 1993, str. 586–591

[23] R. Rojas: *Neural Networks: A Systematic Introduction*, Springer, 1996

[24] F. Samaria: *Face Recognition Using Hidden Markov Models*, PhD thesis, University of Cambridge, 1994

[25] H. Samet: *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, Addison-Wesley, Reading, MA, 1990

[26] H. Samet: *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990

[27] X. Shi, W. Zhao, Y. Shen: *Automatic License Plate Recognition System Based on Color Image Processing*, ICCSA 2005, str. 1159–1168

[28] M. Sonka, V. Hlaváč, R. Boyle: *Image Processing, Analysis and Machine Vision*, Second Edition, PWS Publishing, 1998

[29] J. Žára, B. Beneš, P. Felkel: *Moderní počítačová grafika*, Computer Press, 1998