VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

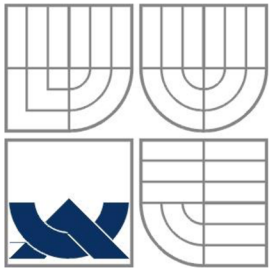
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZPOZNÁVÁNÍ RUČNĚ PSANÉHO PÍSMA

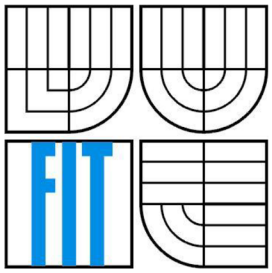
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE RADEK JELÍNEK
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZPOZNÁVÁNÍ RUČNĚ PSANÉHO PÍSMA

HAND WRITING LETTERS RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADEK JELÍNEK

VEDOUCÍ PRÁCE

SUPERVISOR

DOC. ING. FRANTIŠEK VÍTĚZSLAV ZBOŘIL, CSC.

BRNO 2013

Abstrakt

Práce se zabývá rozpoznáváním ručně psaného písma a jeho převod do digitální podoby. Rozpoznávání je zaměřeno na rozpoznávání českých písmen a zjištění úspěšnosti při nevyužití slovníku u rozpoznávání slov. Výsledkem je velmi malá úspěšnost rozpoznávání slov oproti aplikacím využívající slovníky, ale srovnatelná úspěšnost s rozpoznávání jednotlivých znaků.

Abstract

The thesis deals with handwriting recognition and conversion into digital form. Recognition is focused on recognition of letters and finding success when you did not use the dictionary for word recognition. The result is a very small word recognition success rate compared to applications that use dictionaries, but comparable success with recognition of individual characters.

Klíčová slova

rozpoznávání textu, rozpoznávání slov, rozpoznávání písmen, česká písmena

Keywords

text recognition, word recognition, letter recognition, czech letters

Citace

Jelínek Radek: Rozpoznávání ručně psaného písma, bakalářská práce, Brno, FIT VUT v Brně, 2013

Rozpoznávání ručně psaného písma

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Ing. Františka Vítězslava Zbořila, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radek Jelínek
13. 5. 2013

Poděkování

Chtěl bych poděkovat Doc. Ing. Františku V. Zbořilovi, CSc. za odborné konzultace.

Obsah:

1. Úvod

V současné době se stále častěji setkáváme s výpočetními zařízeními, jejichž nedílnou součástí jsou dotykové displeje. Mezi tyto zařízení patří především mobilní telefony a tablety, ale také dotykové obrazovky. Na trhu se ale setkáváme i s grafickými tablety.

Tato práce má za cíl vytvořit aplikaci, která převádí ručně psané písmo do digitální podoby. V současné době jsou na trhu programy, které psané písmo převádí do digitální podoby. Jejich hlavní nevýhoda spočívá v přiřazování napsaného řetězce k určitému slovu z databáze slov. Důsledkem toho je téměř nemožné napsat slovo, které databáze neobsahuje (jména, názvy, odborné výrazy). Další nevýhodou je častá absence učení konkrétního rukopisu a jeho korektura uživatelem.

Cílem této práce je navrhnout aplikaci, která bude rozpoznávat ručně psané latinské písmo z grafického tabletu a bude jej převádět do digitální podoby. Aplikace se bude zaměřovat na česká písmena.

1.1 Aplikace z pohledu uživatele

Uživatel od aplikace očekává, že bude rozpoznávat jeho ručně psaný text a digitalizovanou podobu mu bude ukazovat na obrazovce. Výsledný text si uživatel bude moci zkopírovat a kdekoliv jej vložit. Při špatném rozpoznání některého písmene chce uživatel mít možnost písmeno opravit nebo v případě, že dané písmeno píše specifickým stylem, který databáze nezná, mít možnost přizpůsobit aplikaci pro svůj rukopis. Při psaní většinou písaři nechtějí, aby se řetězec písmen přiřadil k určitému slovu, které bude rozdílné od napsaného. Místo přiřazení napsaného řetězci slovu je žádoucí, aby program nerozpoznal pouze dané písmeno. Uživatelé často nechtějí program dlouho učit svůj rukopis, proto je pro ně výhodné, aby učení rukopisu bylo co nejrychlejší. Vzhled aplikace je požadován nastavitelný, aby si písaři mohli přizpůsobit detaily vzhledu jako barvu písma nebo zobrazení linky popř. pomocných linek.

1.2 Aplikace z pohledu programátora

Aplikace nemůže umět jen rozpoznávání a následný převod textu do digitální podoby, ale musí nabízet i uživatelský komfort. Uživateli musí být práce usnadněna nikoliv přidávána. Je důležité, aby všechny funkce aplikace byly co nejintuitivnější, ale zároveň musí být aplikace plně funkční. Při grafickém návrhu musíme počítat se skutečností, že uživatel bude aplikaci ovládat především pomocí grafického tabletu. Z tohoto důvodu musí být ovládání přizpůsobeno pro snadnou práci s tímto zařízením. Hlavní částí aplikace bude vstupní okno, do kterého bude uživatel zapisovat slova a ta budou převáděna do digitální podoby a zobrazována. Druhou úlohou aplikace bude možnost naučení databáze konkrétnímu rukopisu. V této části bude

aplikace s uživatelem maximálně spolupracovat a navádět jej, aby výsledné informace byly co možná nejpřesnější.

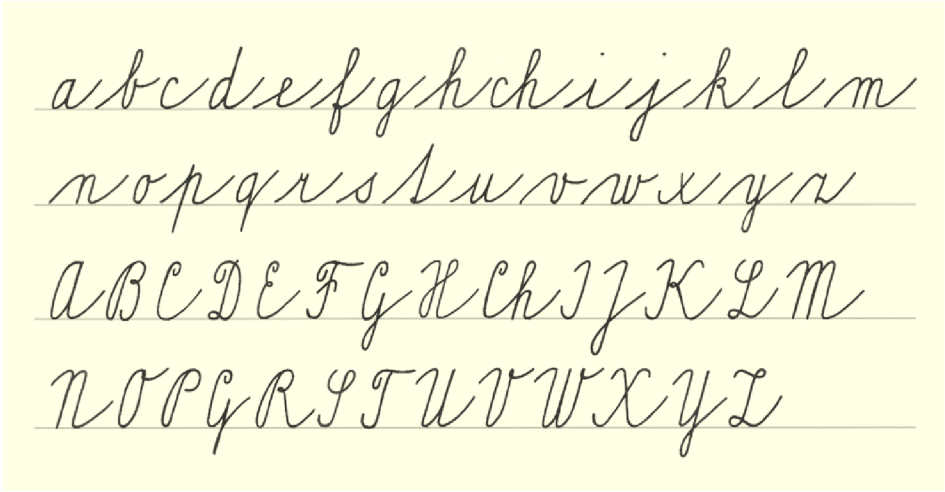
2. Analýza

2.1 Předzpracování

Před samotným rozpoznáním je potřeba provést úpravy, které by zhoršily rozpoznání nebo jej úplně znemožnily. Lidé při běžném psaní textu obvykle přerušují slova například, aby napsali interpunkční znaménko. Po přerušení opět naváží na slovo tam, kde skončili a dopíší jej. Přerušení souvisle psaného textu a jeho následné navázání však nemusí být nutně proto, aby napsali interpunkční znaménko, algoritmus tedy nesmí spoléhat na to, že přerušení souvislého textu je jen z tohoto důvodu, ale třeba i z důvodu vynechání záznamového zařízení nebo třeba oprava dříve napsaného písmena. Navazování čar je důležité pro rozpoznání celého slova, kdyby se čáry nenavazovaly, tak by program rozpoznal pouze jednotlivá písmena a mezery mezi nimi. Toto je však nepřijatelné a proto je pro rozpoznání důležité navazování čar. Algoritmus tedy musí rozpoznat, kdy se jedná o stále stejné slovo a kdy už se jedná o nové slovo. Před samotným navázáním je důležité rozlišit, zda se nejedná o interpunkční znaménko, to se poté musí rozpoznat a připravit pro zacílení k danému písmenu.

2.2 Rozpoznávání

Latinské písmo má 26 písmen [1]. Musíme započítat ještě písmena s diakritikou, kterých máme v češtině 15 [2, 3]. Na obrázku obr. 1 jsou zobrazeny písmena latinské abecedy napsány psacím písmem [4].



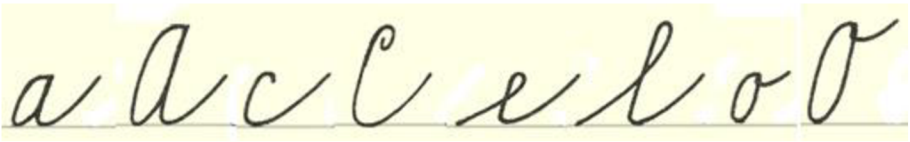
obr. 1 [4]

Písmena z obrázku obr. 1 jsou si na první pohled převážně nepodobná, ale při bližším zkoumání zjistíme jisté podobnosti i u zcela rozlišných písmen. Písmena jsou navíc zobrazena, tak jak by měla vypadat v ideálním případě. Každý člověk se snaží těmto ideálním vzorům přiblížit, aby byl psaný text čitelný nejenom písaři, ale i jakémukoli čtenáři. I přes snahu se vzorům přiblížit je mnohdy písmo vzorům velmi vzdáleno.

Nyní se zaměříme na podobnost písmen v jejich ideálním případě a to u jejich vzorů. Budeme pozorovat, jak moc se které písmena podobají. V následujícím textu budeme podobnosti dělit do podobnostních tříd.

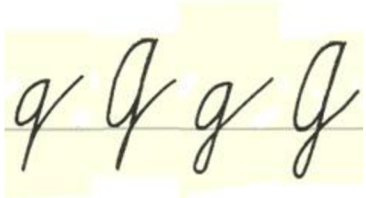
2.2.1 Třída 1

Nejdříve rozebereme písmena, která se tvarem neliší, ale liší velikostí nebo pozicí ukončení písmene k poměru své výšky. Do této kategorie můžeme zařadit následující písmena. Písmenka *„a-A“, „c-C“, „e-l“* se liší pouze velikostí a pozicí ukončení písmene, kde u *„A“, „C“* a *„l“* se ukončuje v polovině výšky písmenka. Písmena *„o-O“* se liší pouze velikostí a pokud nemáme možnost porovnat písmena k jinému písmenu nebo k velikosti řádků, pak jen stěží zjistíme, jestli se jedná *„o“* malé nebo velké *„O“*. Tato písmena jsou zobrazena na obr. 2.



obr. 2

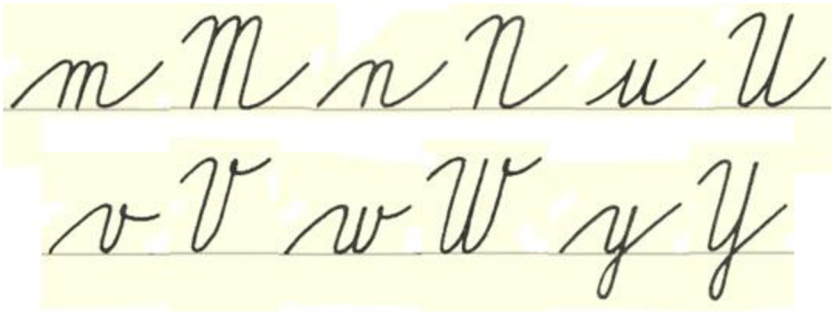
Nesmíme ještě zapomenout na písmena, jejichž malá a velká psací forma se nepodobá, jak je zobrazeno na obrázku obr. 1, ale v historii se psala jiným způsobem. Řeč je o písmenkách *„g-G“* a *„q-Q“*, která se v historii psala jako na obrázku obr. 3. Musíme tedy uvažovat, že se najdou uživatelé, kteří budou psát tyto písmena tak, jak se psávala dříve.



obr. 3

2.2.2 Třída 2

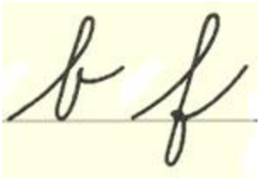
Další třída písmen jsou taková písmena, která si jsou podobná tvarem, ale začínají na různých pozicích v poměru ke své velikosti. Tyto písmena jsou *„m-M“*, *„n-N“*, *„u-U“*, *„v-V“*, *„w-W“* a *„y-Y“* a pro ilustraci jsou zobrazeny na obr. 4, kde si můžeme rozdílů všimnout.



obr. 4

2.2.3 Třída 3

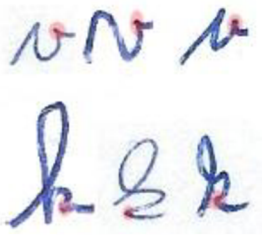
Zajímavá je i podobnost písmen *„b-f“*. Tato dvě písmena se píší téměř zcela stejně, i když to tak na první pohled nevypadá. Jediným rozdílem je skutečnost, že *„f“* je z části pod linkou a *„b“* zcela nad linkou. Podíváme-li se na tato písmena bez linky, zjistíme, že písmeno *„f“* začíná ve druhé třetině své velikosti a končí v první třetině své velikosti (bráno shora dolů), zatímco *„b“* začíná ve třetí třetině své výšky (zcela dole) a končí v polovině své výšky. Zmíněné skutečnosti si můžeme povšimnout na obr. 5.



obr. 5

2.2.4 Třída 4

V této a následujících třídách budeme rozebírat rozdíly mezi vzorovým psacím písmem a konkrétními rukopisy lidí. Při zkoumání jsem zjistil, že lidé často nepíší smyčky u písmen jako *„v“*, *„k“* jak je tomu vidět v načervenalých bodech na obr. 6.



obr. 6

Na obrázku obr. 7 jsou vidět dvě písmena psaná dvěma různými lidmi. Nejde však rozpoznat jestli se jedná o písmeno *„u“* nebo o písmeno *„n“*. V tomto případě se jedná o písmena *„n“*, ale rozpoznání jich jako písmeno *„u“* není chyba, protože jsou podobná jak písmenu *„u“* tak písmenu *„n“*. Stejný problém se vyskytuje i u psaní velkého *„U“* a velkého *„N“*.

obr. 7

2.3 Analýza souvislého textu

Souvislý text s sebou nese další problémy, které se u jednotlivých psacích písmen nevyskytují. V této kapitole budeme pod pojmem slovo rozumět jakékoliv spojení dvou a více písmen.

2.3.1 Napojování

Při psaní slov je nutné jednotlivá písmena napojovat. Napojování vychází buď z tvaru písmena jako na obr. 8, kdy je část slova, v našem případě *„el“*, sdílená (červeně zvýrazněné) mezi dvěma sousedními písmeny.

obr. 8

Můžeme se setkat i s napojením jako na obr. 9, kde je část sdílená ale písmeno *„r“* začíná nad polovinou výšky písmena *„o“*.

obr. 9

Další možností, která se může vyskytnout, je že písař napojí další písmeno na předchozí jiným způsobem, než se očekává (jako u slova *„vik“* na obr. 10) nebo se ve slově objeví části, které nepatří žádnému písmenu a slouží pouze jako napojení (jako u slova *„sc“* na obr. 10).

obr. 10

2.3.2 Problém podobnosti jednoho písmene s částí slova

U rozpoznávání celých slov může dojít k situaci, kdy si budou tvarově odpovídat jediné písmeno a více písmen, tak jak je tomu na obr. 11. Na tomto obrázku je vidět, že jak písmeno *„H“*, tak dvojice písmen *„Je“* mají stejný tvar a liší se pouze velikostmi jednotlivých částí jako u třídy 3, s tím rozdílem, že v tomto případě se jedná o problém vyskytující se u slov. Problém tohoto typu se bude ve slovech vyskytovat často i s jinými písmeny a proto je velmi důležité jej vhodně vyřešit.

obr. 11

3. Návrh

V následující kapitole budou navrženy algoritmy na předzpracování a přiřazení interpunkčního znaménka písmenku. Dále bude navržen algoritmus pro rozpoznání tvaru písmene a budou navrhnuty algoritmy, které se použijí na odlišení písmenek popsaných ve třídách podobností.

3.1 Předzpracování

3.1.1 Navazování čar

U navazování čar musíme počítat s faktem, že lidé čáry navazují různými způsoby a většinou nenaváží tam, kde skončili předchozí tah. Můžeme, ale počítat s tím, že navázání bude provedeno k poslednímu bodu předchozího tahu. Můžeme se setkat s více možnostmi

napojování. Budeme uvažovat pět základních, které jsou vidět na obr. 12. V části *a)* vidíme napojení stejným směrem tahu, v části *b)* napojení jiným směrem tahu, ale zpravidla přesněji, v části *c)* je napojení konce předešlého tahu na jakoukoliv část tahu následujícího tahu. Část *d)* je napojení začátku dalšího tahu na jakoukoliv část předchozího tahu a na závěr u části *e)* je napojení tahu na jakýkoli předchozí tah (čísla u tahů v části *e)* jsou napsány jako pořadí, kdy byly napsány).

obr. 12

Zmíněné možnosti napojení budeme realizovat různými metodami. Největší prioritu navázání budeme klást na nejčastější způsoby napojování a těmi jsou způsob *a)* a *b)* z obr. 12. V nejvyšší prioritě navazujeme buď dva stejné směry tahu, nebo dva odlišné směry tahu. Pokud budeme navazovat stejné směry, musíme počítat s tím, že se písař nebude vždy snažit navázat přesně v místě, kde skončil, ale může navázat dříve a část předchozího tahu zopakovat, v tomto případě se nejlépe hodí vytvořit okolo úsečky posledního směru prvního tahu a úsečky prvního směru navazujícího tahu dvě elipsy. V případě, že se tyto elipsy budou překrývat, pak upravíme koncovou souřadnici prvního tahu na koncovou souřadnici prvního směru druhého tahu a prohlásíme dva tahy za jeden. Budeme upravovat poslední souřadnici prvního tahu, protože předpokládáme, že písař chce tento tah prodloužit. Elipsy okolo úsečky tvořené z počátečního a koncového bodu směru, musí být vhodně široké, aby nedocházelo nechtěnému navázání, ale musíme zajistit navázání, tam kde jej písař zamýšlel.

V případě *b)* nám postačí udělat u koncového bodu prvního tahu a u počátečního bodu druhého tahu kružnici a zjistit jestli se tyto dvě kružnice překrývají. Pokud se kružnice překrývají tak upravíme první souřadnici druhého tahu na poslední souřadnici prvního tahu. Upravujeme souřadnici u druhého tahu, protože předpokládáme, že právě na ni chce písař navázat a co nejvíce se jí přiblížit. Kružnice, které budou okolo jednotlivých bodů, musí mít vhodnou velikost, aby nebyly moc velké a tak nedošlo k navázání dvou rozdílných slov, ale nesmí být ani moc malé, aby k navázání mohlo někdy dojít.

Nyní když nedojde k rozpoznání v případě *a)* nebo *b)*, pak budeme uvažovat možnost navázání v případě *c)* a poté v případech *d)* a *e)*. Možnost *c)* bude navazování kombinací způsobů jako u *a)* a *b)*. Tyto způsoby v *c)* zkombinujeme, tak, že u posledního bodu prvního tahu uděláme kružnici a budeme zjišťovat, jestli se překrývá s jakoukoliv elipsou, která je okolo

úseček směrů druhého tahu. Pro případ *d)* a *e)* budeme postupovat podobně, ale kružnici uděláme okolo prvního bodu druhého tahu a překrývání budeme zjišťovat s elipsami okolo úseček směrů prvního tahu pro případ *d)* a pro všechny předchozí tahy v případě *e)*. Po zjištění navázání prohlásíme dva tahy za jeden, ale nebudeme upravovat žádné souřadnice, protože by často vznikl tah navíc, který by se v rozpoznávání, ve většině případů, odřízl.

Při psaní se může stát, že další tah navazuje ve stejném bodě, ve kterém předchozí tah končí. Zde dojde k automatickému navázání tahu, protože program nerozpozná přerušení tahu. Tento případ navázání je ale velmi málo pravděpodobný.

3.1.2 Přiřazení interpunkčních znamének k písmenu

V českém jazyce se nejčastěji setkáváme s interpunkčními znaménky, jako jsou tečka, čárka a háček. Pokud má písmeno interpunkci, tak je tato interpunkce nad písmenem. To znamená, že má alespoň polovinu své šířky nad šířkou písmene, ke kterému patří. Interpunkce je nad písmenem převážně u malých písmen (např. *i*, *č*, *ó*, *ň*, ale i *Ň*, *Š*) nebo je alespoň nad polovinou výšky písmene (např. u písmen *ď*, *ť*).

My budeme vyházet ze skutečností, že je většina šířky interpunkce nad písmenem, ke kterému patří a že je interpunkce nad polovinou výšky písmene. Je možné, že se při velmi silném sklonu písma na stranu SV stane, že se větší část šířky interpunkce objeví nad následujícím písmenem. U zjišťování jestli je šířka interpunkce nad šířkou písmene musíme tedy brát v úvahu sklon písma. Sklon písma je nutné nejdříve vypočítat. Jako slon budeme brát úhel, který přibližně odpovídá celkovému sklonu. Tento úhel budeme počítat z úseček směrů SV a JZ, protože sklon písma je u většiny lidí do směru SV a k němu budeme brát právě opačný směr JZ. Jednotlivým úhlům přiřadíme váhy podle toho jak je úsečka dlouhá v porovnání s velikostí písmene. Váhy musíme přiřadit, aby krátká úsečka, která je téměř vodorovná a o sklonu písmene nevypovídá, neovlivnila sklon stejně jako úsečky, které vizuálně dělají sklon písmena. Ostatní úsečky budou podle své váhy snižovat úhel sklonu.

Je možné, že se najde člověk, který má sklon opačný a to do směru SZ, tak prohlásíme úhel směru k lince jako pravý. Pravý úhel proto, že při tomto sklonu píší lidé interpunkci nad šířku písmene. Nemusíme se tedy bát, že by se šířka interpunkce vyskytla z větší části nad předchozím písmenem.

3.2 Rozpoznávání písmen

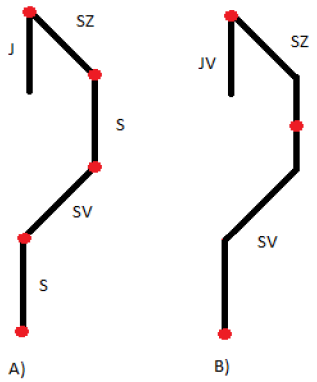
3.2.1 Rozpoznání tvaru písmena

Základní částí samotného rozpoznání písmena je rozpoznání jeho tvaru. Naší velkou výhodou je sledování tahu písaře od začátku jeho psaní, až po jeho dokončení. Při psaní můžeme tedy

sledovat směr pohybu pera a změny směru pohybu. Samotné rozlišení tvaru však nedokáže jedinečně rozpoznat písmeno, jak už bylo zmíněno dříve, ale poslouží jako základní představa o písmenu pro další vyhodnocování.

Při sledování směrů pohybu budeme používat označení jako u kompasu tedy S (sever), J (jih), V (východ), Z (západ), SV (severovýchod), SZ (severozápad), JV (jihovýchod), JZ (jihozápad). Pro zjednodušení budeme množinu směrů S, J, V a Z označovat jako jednoduché směry a množinu směrů SV, SZ, JV a JZ jako složené směry.

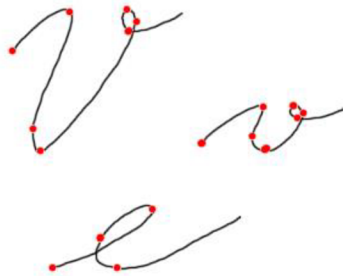
Když získáme dvě souřadnice bodů od vstupního zařízení, můžeme určit směr. Rozlišovat budeme pouze složené směry, protože jednoduché směry se budou vyskytovat jen zřídka. Proto každý směr z množiny jednoduchých směrů bude sdílený mezi dvěma směry složenými (směr S bude sdílený mezi směry SV a SZ, směr V bude sdílený mezi směry SV a JV atd.). Pokud se vyskytnou dva body, mezi kterými bude pouze jednoduchý směr, který nebude předcházet ani navazovat na žádný složený směr, kterému bychom směr mohli přiřadit, pak si zvolíme jejich přiřazení k složeným směrům následovně. Směr S směru SZ, směr J směru JV, směr V směru SV a směr Z směru JZ. Pro lepší pochopení si ukážeme, jakým způsobem budeme přiřazovat jednoduché směry složeným směrům. Mějme následující řetězec směrů: S – SV – S – SZ – J. Protože za směrem S následuje složený směr obsahující směr S, pak prohlásíme směr S – SV za směr SV. Za směrem SV následuje směr S a následně směr SZ. Směr S můžeme přiřadit oběma ze směrů SV a SZ, tak směr S rozdělíme na dvě části a jednu přiřadíme směru SV a druhou směru SZ. Za směrem SZ následuje směr J, který směru SZ nemůžeme přiřadit a protože je směr J poslední musíme mu přiřadit předem domluvený směr, který je JV. Výsledný řetězec tedy bude vypadat následovně: SV – SZ – JV. Staré a nové body změn jsou vidět na obr. 13. Část *A)* zobrazuje původní body změn a část *B)* nové body změn.



obr. 13

Určovat směr budeme vždy ze dvou bodů. Každý bod bude určovat pozici pixelu. Tyto body budou obvykle od sebe vzdáleny na různou vzdálenost podle toho, jak je bude aplikaci posílat vstupní zařízení. Může však nastat situace, kdy budeme směr zjišťovat ze sousedních bodů. Tato vzdálenost je ale příliš malá a proto nebude vždy poskytovat správné výsledky. Z tohoto důvodu nebudeme brát hned sousedící bod. Pro lepší výsledky směru tahu budeme brát ty dva body, jejichž vzdálenost na ose x nebo y bude větší nebo rovno 3. Číslo 3 je zvoleno, protože kdyby vzdálenost byla větší nebo rovna 2, tak by se mohlo stát, že písař udělá menší chybu v tahu, která by se negativně projevila v porovnávání. Vzdálenost větší nebo rovna 4 je už moc velká, protože hodně písmen má malé detaily jako například písmeno *„r“* a tyto detaily by se při volbě minimální vzdálenosti 4 nemusely projevit.

Na obr. 14 můžeme vidět vyznačené body (červeně) změn směrů. V případě velké *„V“* by se jednalo o směry SV - JZ - JV - SV - SZ - JV, tedy stejné směry jako u malého *„v“*. Pro tyto případy nám tvar písmene nepomůže, ale pomůže nám k vyřazení možnosti, že se jedná o tvarově rozdílná písmena jako například malé *„e“*, které má řetězec směrů SV - JZ - JV - SV. Vypočítaný kód budeme označovat jako tvarový kód.



obr. 14

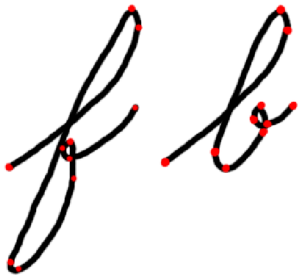
3.2.2 Rozpoznávání velikostí jednotlivých částí písmena

Pro rozpoznávání velikostí musíme navrhnout algoritmus, který se vyrovná nejlépe se všemi třídami 1, 2 a 3. Protože záleží na velikosti písmena, podíváme se na jednotlivé velikosti písmen. Písmeno malé *„e“* nezasahuje pod linku ani nepřevyšuje pomocnou linku (která je hned nad ním), ale písmeno malé *„b“* už tuto pomocnou linku převyšuje, ale nezasahuje pod linku. Typickou ukázkou písmena, které převyšuje pomocnou linku a zasahuje i pod linku je například písmeno velké *„J“*. Písmena můžeme tedy rozdělit podle výšky na 1 až 3 části. Písaři, ale nedodržují striktně tyto třetiny, proto se nemůžeme spoléhat na takto přesné velikosti částí.

V našem případě, ale můžeme brát velikosti, které jsou menší než zmiňované třetiny. Je to myšleno tak, že si vezmeme výšku celého slova, tuto výšku rozdělíme na menší části, než jsou třetiny, to znamená, že budeme uvažovat čtvrtiny. Máme-li čtvrtinu určenou jako nějakou

hodnotu, můžeme k ní přidat již zmiňovanou třetinu z celkové výšky písmena a také polovinu celkové výšky písmena. Nyní víme jak je písmeno vysoké a jak je velká jeho čtvrtina, třetina a polovina a navíc známe všechny změny směrů a souřadnice těchto změn, které byly popsány v kapitole 3.2.1. Nyní vybereme první směr a jeho souřadnice, vypočítáme výšku úsečky (absolutní hodnota z odečtení ypsilonových složek souřadnic bodů úsečky) a porovnáme s čtvrtinou, třetinou a polovinou výšky písmene. V případě, že je výška úsečky větší, než polovina výšky písmene zaznamenáme do výsledného kódu hodnotu 2, v jiném případě porovnáme, zdali je úsečka větší jak třetina výšky písmene a pokud ano zaznamenáme hodnotu 3, jinak porovnáme se čtvrtinou a pokud je hodnota výšky úsečky větší, zaznamenáme 4, jinak zaznamenáme hodnotu 0.

Z popsaného algoritmu nám vyjde kód, který budeme označovat jako výškový kód. Výškový kód bude stejně dlouhý jako kód určující směry tahu a bude vypovídat o výšce (ypsilonové vzdálenosti bodů) úsečky.



obr. 15

Na příkladu si ukážeme výpočet výškového kódu. Budeme využívat písmene f na obr. 15, který má červeně vyznačeny body změny směru. Písmeno f má kód směrů SV – SZ – JZ – JV – SV – SZ – JZ – JV – SV, výškový kód má ve většině případů hodnotu 0, až na 1., 3. a 5. směr, kde nabývá hodnot 2, 2 a 3. Výsledný kód pro písmeno *f* je: 202030000 a pro písmeno *b*: 202040000. Tímto kódem tedy dokážeme rozlišit všechny třídy podobností včetně podobnosti jednoho písmene s částí slova.

Napsaná slova musí končit (místo kde písař zvedne pero) vždy v místě, kde správně končit mají. Většinou písmena končí v místě pomocné linky, která je první nad hlavní linkou. Splnění podmínky zajišťuje správné vytvoření výškového kódu u všech písmen ve slově. Nesplnění podmínky může vést ke zhoršenému rozpoznávání. Někdy se může stát, při psaní věty, že písmeno nezačíná na lince, i když napsali bychom jej samostatně, tak by na lince začínalo, ale začíná výše. Písmena začínají výše, navazují-li na některá písmena jako je například malé *o*. Tento problém může způsobit odlišnosti ve výškovém kódu, a proto je dobré zařadit do učení i učení s navazováním na písmeno malé *o*. Výškový kód, ale trpí ještě jednou

nedokonalostí, která je spojena s rozlišením obrazovky. Rozlišení obrazovek není, v současné době, příliš vysoké převažuje rozlišení 1366x768 a hned za ním rozlišení 1024x728 [5]. Nízké rozlišení způsobuje, při psaní drobným písmem, různé možnosti výškového kódu. Z tohoto důvodu je potřeba, aby písmena byla naučena několikrát po sobě a byly tak postihnuty, v ideálním případě, všechny výškové kódy, které mohou nastat pro jeden hlavní kód.

3.3 Rozpoznávání slov

Pro rozpoznávání slov budeme využívat rozpoznávání jednotlivých písmen. Po napsání slova budeme předpokládat, že uživatel napsal jen jedno písmeno. V tomto případě zkusíme rozpoznat slovo jako jedno písmeno. V případě neúspěchu budeme odřezávat jednotlivé směry v pořadí zleva (budeme postupovat k rozpoznání posledního písmene jako prvního). V případě neúspěchu rozpoznání písmena ani v poslední části, odřízneme poslední směr a postup budeme opakovat. V případě rozpoznání písmena písmeno zaznamenáme a rozpoznané směry nebudeme v další iteraci uvažovat, až na první směr (zleva), který může být sdílený mezi sousedícími písmeny. Nová pozice koncového ořezu (nejpravější směr) bude stará pozice – počet změn v aktuálně rozpoznaném písmenu + 1 (sdílený směr).

Uživatelé mají možnost se přihlašovat do aplikace. Přihlášený uživatel očekává, že se nejdříve budou rozpoznávat písmena, která odpovídají jeho rukopisu. Rozpoznávání nejprve rozpozná písmena konkrétního uživatele. Písmena a části slov, která nebyla rozpoznána podle kódů (hlavního a výškového) přihlášeného uživatele se rozpoznají na základě hlavních kódů uživatele a výškových kódů všech uživatelů. Na závěr se provede závěrečné zpracování nerozpoznaných písmen a částí slov na základě všech v databázi dostupných hlavních kódů a všech výškových kódů. Tímto docílíme maximální možné snahy rozpoznat konkrétní rukopis uživatele.

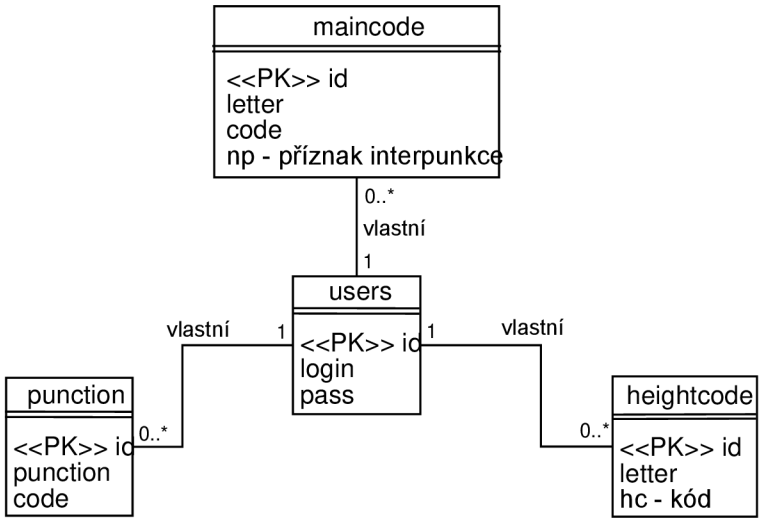
Ve druhém popsaném kroku rozpoznání slova se používá hlavní kód přihlášeného uživatele a výškový kód všech uživatelů z toho důvodu, protože více výškových kódů odpovídají, na základě získaných experimentálních výsledků, pro jeden hlavní kód. Použitím tohoto kroku ušetříme uživateli více času při učení.

3.4 Návrh databáze

Databáze musí obsahovat tabulky s informacemi o hlavním kódu, výškovém kódu. Musíme také ukládat všechny hlavní kódy interpunkcí (výškové kódy se u interpunkcí nepočítají) a na závěr potřebujeme informace o uživatelích, kteří do databáze ukládají svoje specifické verze písmen podle svého rukopisu.

Hlavní kód musí obsahovat písmeno, ke kterému náleží a samotný kód. Kód může být poměrně dlouhý, proto bude lepší jej uložit jako řetězec. Stejně tak tomu bude i u výškového kódu a u tabulky s interpunkcemi, kde nahradíme písmeno číslem reprezentujícím interpunkci. K hlavnímu kódu nakonec přidáme příznak, který určí, jestli písmeno potřebuje mít interpunkci (v případě písmen *i* a *j*).

Uživatelé se budou přihlašovat pomocí uživatelského jména a hesla. Protože se nejedná o aplikaci s citlivými údaji, tak není nutné heslo šifrovat. Obrázek obr. 16 zobrazuje ER diagram tabulek v databázi.



obr. 16

3.5 Návrh tříd

Třída pro rozpoznávání musí přijímat informace a to v našem případě souřadnice. Souřadnice budou zasílány v okamžiku přiložení pera, tahu pera a při uvolnění pera. Těmito souřadnicemi bude rozuměn tah. Každé písmeno, slovo nebo věta se skládají z minimálně jednoho tahu. Získání informací z třídy musí být připraveno na různé možnosti. Programátor využívající třídu pro rozpoznávání může požadovat získání pouze jednoho písmene popř. množiny odpovídajících písmen, slova nebo i celé věty. Třídu bude vhodné implementovat tak, aby nesla informaci o jednom řádku (nebude možné rozpoznat slova, která se v rámci třídy nachází nad nebo pod již existujícími slovy). Databáze s písmeny a uživateli musí být těsně provázána s třídou pro rozpoznávání, z tohoto důvodu by třída měla obsahovat metody, které pracují s databází. Mezi metody pracující nesmí chybět metoda pro nastavení připojení k databázi včetně výběru uživatele a databáze, test připojení k databázi pro kontrolu, přihlášení a odhlášení uživatele a vytvoření nového uživatele. Práce s uživatelem je z důvodu rozdílného algoritmu rozpoznávání pro přihlášeného a odhlášeného uživatele. Pro učení musí být přístupny

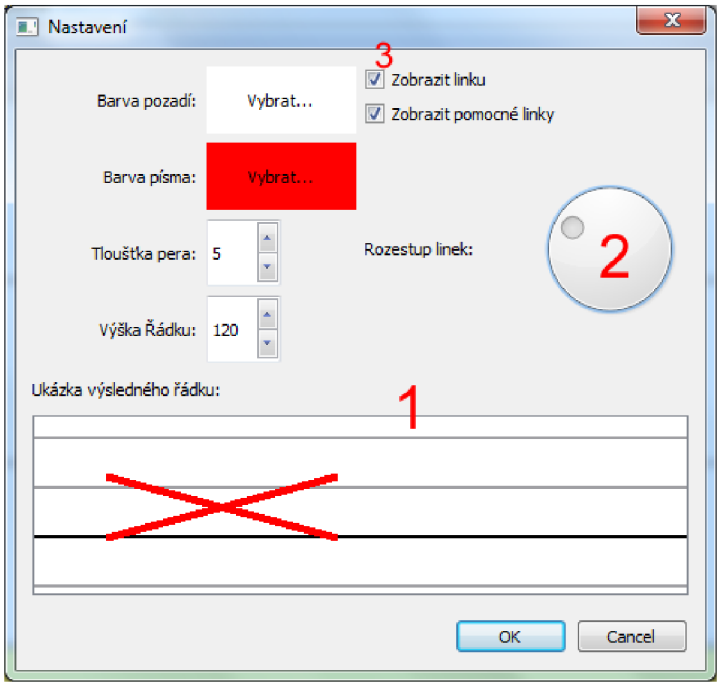
metody na naučení písmene a interpunkčního znaménka, které na základě aktuálního stavu vygenerují podle potřeby tvarový a výškový kód a uloží jej do databáze k právě přihlášenému uživateli.

Aplikace je psána v jazyce C++ s pomocí knihovny Qt. Pro lepší propojení grafické a logické části bude vhodné vytvořit třídu, která bude mít hlavní cíl předávání souřadnic třídě zajišťující rozpoznávací logiku a zároveň grafické zobrazování tahů. Grafickými třídami s formuláři bude formulář sloužící pro přihlášení uživatele a pro nastavení prostředí.

4. Implementace

4.1 Grafické rozhraní

Grafické uživatelské rozhraní bylo navrženo, aby co nejvíce usnadnilo práci s psaním a učením písmen. Při návrhu nebyl dáván důraz na intuitivní ovládání, ale na rychlost ovládání a částečně také na skutečnost, že se aplikace bude ovládat pomocí grafického tabletu.



obr. 17

Na obrázku obr. 17 můžeme vidět nastavení vzhledu linek, které se promítá do části 1 tak, jak bude vypadat na rozpoznávací obrazovce. Za zmínku stojí i „*Rozestup linek*“ (2), kterým se určuje vzdálenost mezi pomocnými linkami.

4.2 Třída pro rozpoznávání

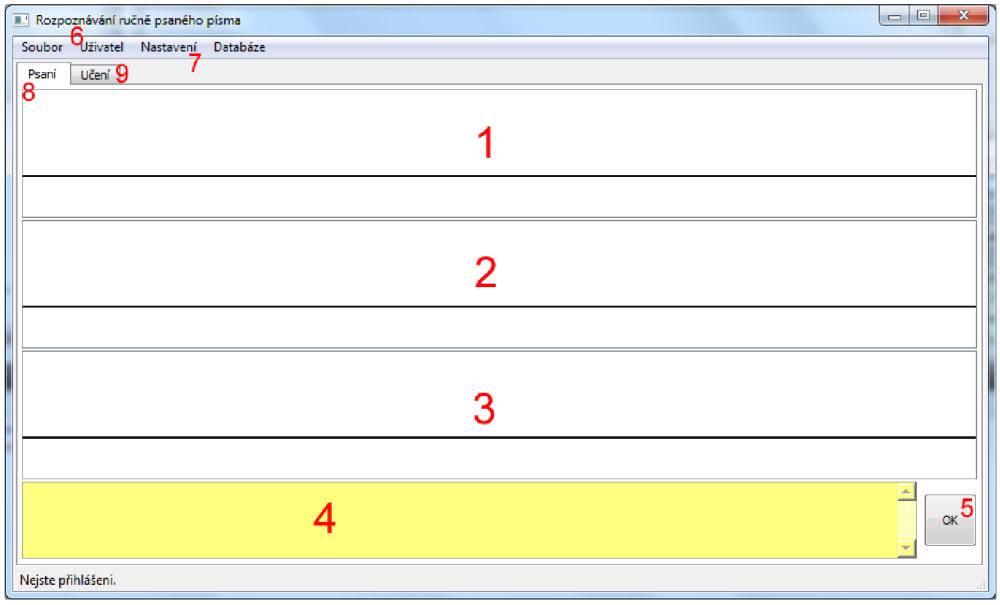
Třída pro rozpoznávání je nazvána `hwlr` znamenající „*Hand Writing Letters Recognition*“. Rozpoznávání v metodě se děje na základě stavu třídy, která je brána jako jeden řádek. Po zavolání metody `teachDBLetter` se vyhodnotí všechny směry, které do doby zavolání byly do funkce poslány, a písmeno se uloží do databáze. Písmeno se uloží uživateli, který je aktuálně přihlášen a v případě, kdy není nikdo přihlášen, se písmeno uloží do databáze jako pro univerzálního uživatele. Interpunkce rozpoznává aplikace automaticky, před rozpoznáním je nutné naučit interpunkce databázi. Učené interpunkce, ale třída nebere z rozpoznaných interpunkcí k písmenu, ale z aktuálního stavu jako by se jednalo o písmeno.

Zmíněné přihlášení se provádí funkcí `logIn`. Nejdůležitějšími metodami jsou metody na rozpoznání písmen (`getLetters`), slova (`getWord`) a věty (`getSentence`). Základní metodou je metoda na rozpoznání písmen, kterou využívá metoda na rozpoznávání slova a tu využívá metoda na rozpoznání věty. Místo možnosti získání množiny všech odpovídajících písmen je připravena metoda (`getLetter`), která vrací pouze jedno písmeno a to takové, které naučilo stejným kódem největší množství uživatelů. Může se stát, že některá písmena naučily stejné počty uživatelů, pak metoda vrátí jedno z písmen. Vrácené písmeno by bylo dobré vracet místo náhody sofistikovanějším algoritmem, který by například zjistil míru používání písmena.

Neméně důležité jsou metody na přidávání souřadnic. Před začátkem tahu (okamžik kdy dojde ke kliku) je připravena metoda `startMove` přebírající bod začátku tahu. Následně se volá metoda `addPoint` vždy v okamžiku pohybu pera (myši) a tah je ukončen uvolněním pera a v místě uvolnění se předá bod metodě `endMove`.

4.3 Rozpoznávací obrazovka

Na obrázku obr. 18 můžeme vidět hlavní obrazovku, která slouží pro rozpoznávání, aplikace pro rozpoznávání ručně psaného písma. Hlavní dominantou jsou 3 řádky (odkazy 1, 2 a 3), dále se zde nachází textové pole (4), do kterého se vkládá rozpoznaný text, tlačítko „*OK*“ (5) pro dokončení rozpoznání všech nerozpoznaných slov v řádcích. V hlavním menu stojí za povšimnutí hlavně „*Uživatel*“ (6), kde po rozbalení jsou možnosti k vytvoření nového uživatele, přihlášení nebo odhlášení. Po rozbalení nabídky „*Nastavení*“ (7) máte možnost vybrat pole „*Vzhled*“ a nadefinovat si jej jak Vám bude vyhovovat viz. obr. 17. V oblasti menu možnost „*Databáze*“ lze otestovat připojení k databázi nebo nastavit připojení k jiné databázi. Pod hlavním menu se nachází výběr ze dvou karet, z nichž jedna (8) je aktivní a slouží právě pro rozpoznávání a druhá (9) slouží pro učení.

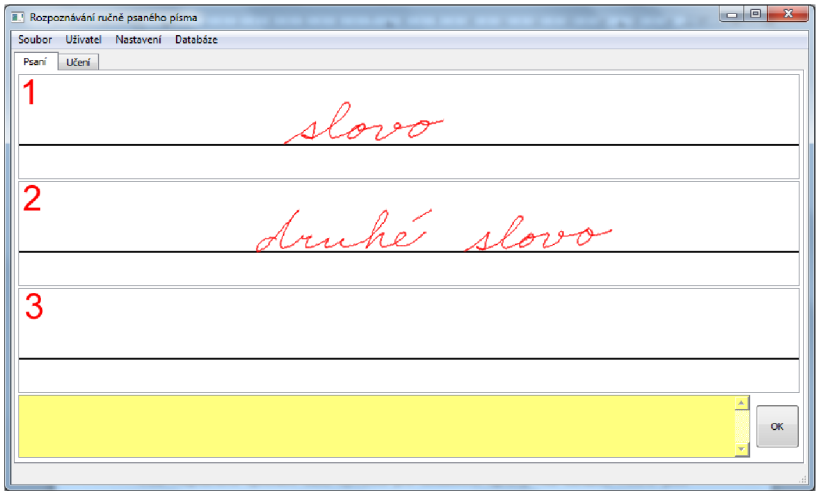


obr. 18

4.3.1 Zpoždění rozpoznávání

Text je při psaní vpisován do prostorů z obr. 19 označené čísly 1, 2 a 3 a to v pořadí jak jsou očíslovány. Jakmile začne uživatel psát do prostoru 3, dojde k rozpoznání a vymazání prostoru 1. Než k tomuto dojde je možné jakkoliv tento řádek upravovat. Po dopsání v řádku 3 se pokračuje dalším textem do řádku 1 a zase dojde k rozpoznání a vymazání řádku 2 a tímto způsobem se pokračuje v psaní stále dokola.

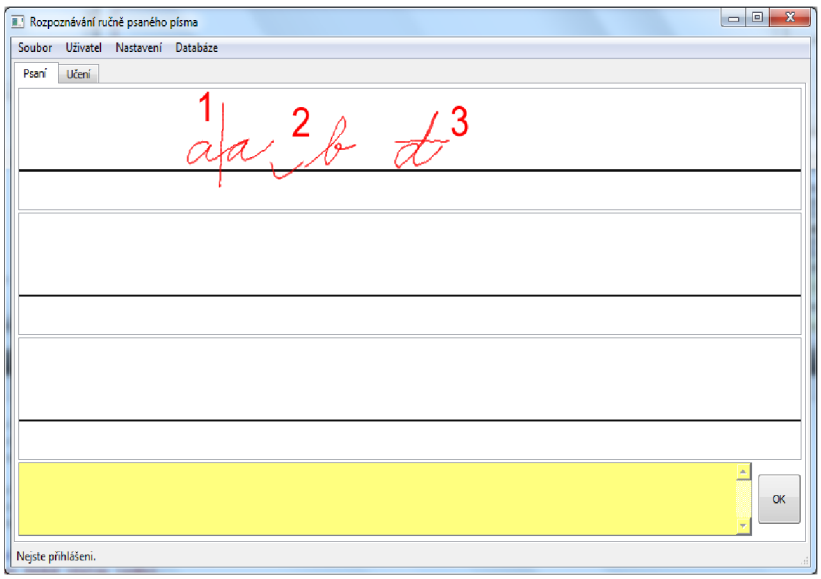
Obrázek obr. 19 ukazuje napsaná nerozpoznaná slova na řádcích 1 a 2, kdy rozpoznání prvního řádku nastane v okamžiku dotyku pera 3. řádku. Uživatel má možnost opravovat a dopisovat do posledních dvou řádků, které napsal. Možnost dát uživateli úpravu posledních dvou řádků je z toho důvodu, aby bylo možno upravit naposledy napsaný řádek (například chybějící interpunkční znaménko) po začátku psaní na řádek nový. Tři řádky jsou zase z toho důvodu, aby bylo možné zachovat popsanou funkčnost s co nejméně řádky. Díky co nejmenšímu počtu řádků není potřeba přesouvat pero o příliš velkou vzdálenost, pokud písař končí psaní na 3. řádku a přesouvá se na řádek 1.



obr. 19

4.3.2 Úpravy textu písařem

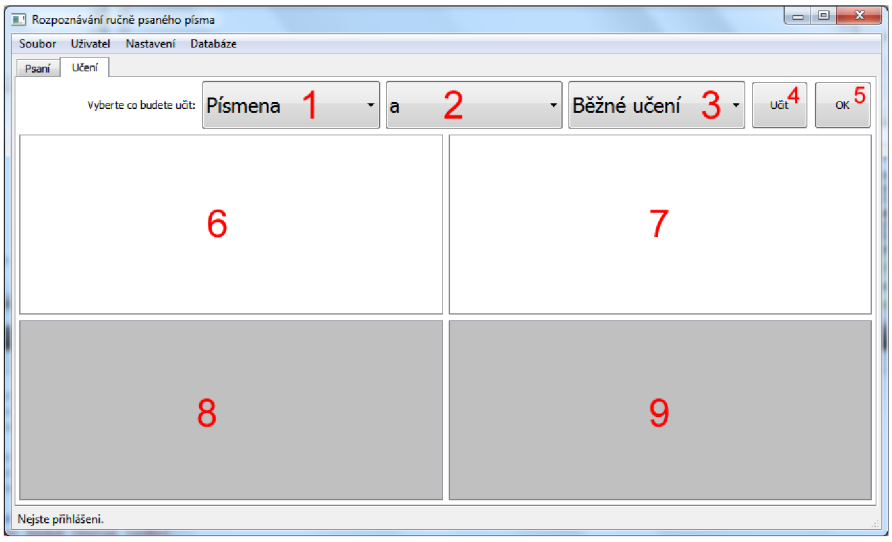
Uživatelé nemusí být vždy spokojeni se sebou napsaným textem a mohou na něm chtít provést dodatečné úpravy jako je rozdělení slova na dvě nebo naopak spojení dvou slov v jedno slovo nebo vyřazení z rozpoznávání určitých například nepovedených písmen. Implementovaná třída rozpoznává speciální sadu symbolů pro dodatečné úpravy. Na obr. 20 jsou symboly zobrazeny. Symbol 1 je rozdělení slova, tato svislice musí být co největší, nejlépe přes celou výšku řádku. Pro spojování slouží symbol 2, který musí být v dolní části písmen namalován buď jako na obrázku lehce do tvaru oblouku, nebo nejlépe jako vodorovná úsečka. Posledním symbolem je symbol 3 z obr. 20, který má tvar vodorovné úsečky přes délku písmen určených ke smazání a nakreslen do poloviny písmene.



obr. 20

4.4 Obrazovka pro učení databáze

Na obrázku obr. 21 můžeme vidět již zmiňovanou kartu pro učení. Před samotným učením musíte nejprve vybrat, co budete databázi učit. V první rozbalovací nabídce (1) vyberete možnost podle toho, zda chcete písmena nebo interpunkce a v druhé rozbalovací nabídce (2) vyberete konkrétní písmeno nebo interpunkci. Třetí rozbalovací nabídka (3) je aktivní jen v případě učení písmen a dává možnosti pro běžné učení, napojování typu A (aplikace vám do zadaného prostoru předepíše nedokončené malé psací písmeno *e* a Vaším úkolem je na toto nedokončené písmeno navázat a napsat Vámi vybrané písmeno). Je zde možný i výběr napojování typu B, kde je Vám místo písmena *e* předepsáno malé psací písmeno *o*. Pro lepší rozpoznávací schopnosti aplikace doporučuji provést po běžném učení konkrétního písmena i učení typu A a učení typu B.

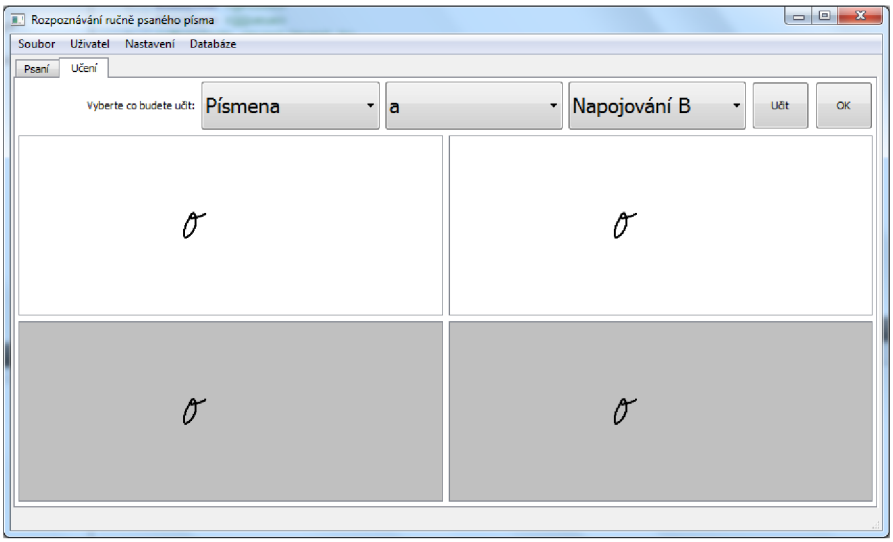


obr. 21

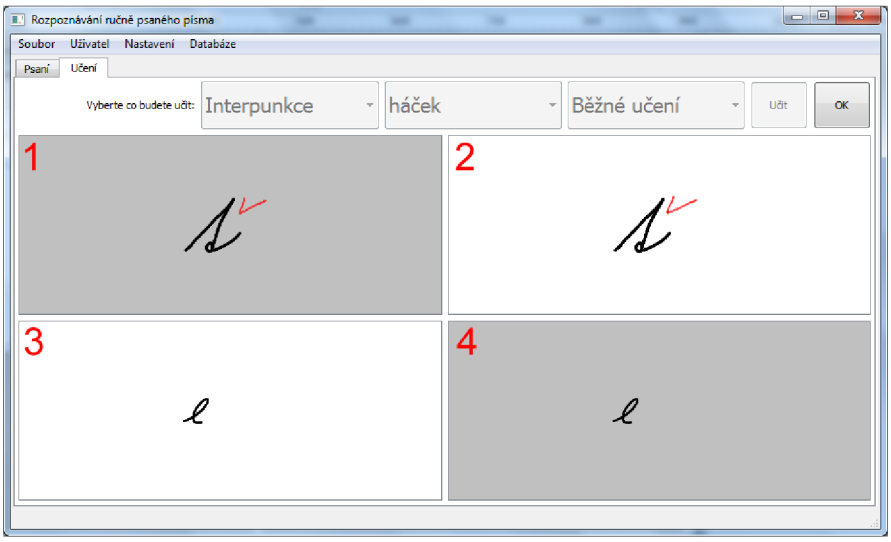
Po výběru učení stačí jen klepnout na tlačítko „*Učit*“ (4) a učení bude zahájeno, naopak po stisku tlačítka „*OK*“ (5) se do databáze uloží všechna neuložená písmena a provede se ukončení učení. Nakonec se dostáváme ke čtyřem učícím plochám (6, 7, 8, 9). Povšimněte si na obrázku obr. 21, že první dvě plochy (6, 7) mají bílé pozadí a druhé dvě plochy (8, 9) mají šedé pozadí. Bílá barva znamená, že plochy jsou aktivní a je možné do nich psát, ale šedé plochy jsou neaktivní a jsou ve stádiu čekání na aktivování.

Obrázek obr. 22 zobrazuje učení s napojováním na písmenko malé *o*. Velikost těchto písmen lze ovlivnit v nastavení vzhledu. Pro změnu velikosti předepsaných písmenek stačí

pootočit kolečkem (2 na obr. 17). Kolečko je, ale neaktivní pokud není zaškrtnutá možnost „*Zobrazit linku*“ (3 na obr. 17). Po otočení kolečkem není nutné, aby políčko „*Zobrazit linku*“ bylo dále zaškrtnuté. Styl změny velikosti předepsaných písmen v učení se může zdát složitý, ale je přes zaškrtnutí zvolen z důvodu, aby při otáčení kolečkem bylo v poli zobrazujícím aktuální nastavení (1 na obr. 17) vidět jak velká písmenka budou (jejich velikost bude taková, jako vzdálenost mezi pomocnými linkami).



obr. 22



obr. 23

Na obrazovce učení z obrázku obr. 23 jsou vidět učící plochy 1, 2, 3 a 4, kde momentálně aktivní jsou plochy 2 a 3, což znamená, že do těchto ploch můžu psát. Začnu-li ale

psát do plochy 3, plocha 2 se deaktivuje a aktivuje se plocha 4, a když začnu psát poté do plochy 4, deaktivuje se plocha 3 a aktivuje plocha 1. Tímto způsobem mění plochy svoji aktivitu během učení. Reakce neaktivní plochy, ve které je již napsáno písmeno nebo interpunkční znaménko, na pokus o psaní nebo pouhý klik je jednoduchá a to je provedení smazání vepsaného písmena nebo interpunkčního znaménka. Touto možností mazání je dána uživateli možnost smazat špatně napsané písmeno před tím, než bude uloženo do databáze. Písmeno je uloženo do databáze vždy těsně před tím, než je neaktivní plocha aktivována. Například plocha 2 uloží informaci do databáze v okamžiku dotyku pera aktivní plochy 1. Plochy jsou aktivovány v pořadí 1 – 2 – 3 – 4 – 1 – 2 … Tento návrh umožňuje rychlé učení písmen, kdy uživatel píše stále dokola jedno písmeno do každé z ploch a umožňuje také rychlé smazání špatně napsaného písmene.

Ve chvíli, kdy uživatel napíše do učících ploch po sobě 10 písmen, která již v databázi jsou, text v tlačítku „*OK*“ zezelená a tím dá uživateli najevo, že je provedeno dostatečné množství učení. Tato funkce nefunguje při učení interpunkčních znamének, protože ta nejsou tak komplikovaná a většinou stačí několik vzorů k naučení.

5. Testování

Následující důležitá kapitola ukáže schopnost rozpoznávání aplikace na několika vybraných uživatelích. Aplikace bude srovnána s komerčními nástroji, jako je „*Vstupní panel počítače Tablet PC*“, který je dostupný v instalaci operačního systému Microsoft Windows 7 a s aplikací „*MyScript Stylus 3.2*“, jenž je placená s možností vyzkoušení na 30 dní.

5.1 Návrh testování

Aplikace je určena pro uživatele, kteří píší česky a bude tedy testována na vybraných českých slovech, větách a slovních spojení. Hovorová slova a slova, která nejsou ve slovníku komerčních aplikací budou simulována náhodnou posloupností písmen spojených v jedno slovo. Vybrané testovací texty budou vybrány tak, aby se v nich objevilo co nejvíce písmen české abecedy, a budou vybrány slova obsahující všechna interpunkční znaménka, která se vyskytují v českém jazyce.

K testování budou osloveni běžní lidé. Testeři dostanou čas na seznámení se s grafickým tabletem, tento čas nebude nijak limitován, jde především o to, aby si lidé, kteří se s tímto zařízením setkali poprvé, osvojili práci. Každému uživateli bude vytvořen účet v aplikaci, pod kterým bude po čas testování přihlášen. Před učením bude tester seznámen s aplikací, s jejím učením, řádky pro rozpoznávání budou nastaveny podle přání uživatele a bude

vysvětleno jak program učit, mazat nechtěná písmena a podobně. Po seznámení přistoupí uživatel k učení databáze. Učení je velmi zdlouhavé, proto budou k učení vybrány jen ty písmena, která se nachází v textu, který je připraven, a učení napojování budou vybrána jen některá písmena, které byla shledána vhodnými pro naučení. Tímto krokem se ušetří čas za dosažení stejného výsledku jako nechání naučení všech písmen se všemi druhy napojování, z nichž některé naučené vzory by při navrženém testování byly zbytečné.

Hlavní částí testování je rozpoznávání, ke kterému se přistoupí po učení. Uživatelé budou psát předepsaný text, svým rukopisem a bude jim měřen čas, jak potřebují k napsání jednotlivých textů. Tento text bude psán v módu rozpoznávání celých slov a vět. Po rozpoznání konkrétní věty, slova nebo slovního spojení budou zaznamenány rozpoznaná písmena a čas potřebný pro napsání. Protože testeři nebudou učit program příliš dlouho, bude tester psát každé slovo 3x a do záznamu o testování se zapíše nejlepší shoda. Mimo rozpoznávání textu v módu rozpoznávání slov budou uživatelé psát stejný text v módu rozpoznávání po jednotlivých písmenkách. Písmena budou uživatelem napsána 5x, pouze z důvodu nedostatečně dlouhé učení, a zaznamená se písmeno, které bylo rozpoznáno alespoň 3x. Po dokončení napsání zadaného textu bude uživatel zadávat stejný text do komerčních nástrojů s žádostí zachování stejného rukopisu. Zadávání do těchto aplikací již nebude časově měřeno, ale bude pouze zaznamenáván rozpoznaný text. Psaní bude vyžadováno po slovech i po jednotlivých písmenech, ale protože se jedná o profesionální nástroje, bude zde pouze jeden pokus na napsání slova nebo písmenka (opakování bude možné pouze v případě, že se testerovi nepodaří napsat text tak, jak sám chtěl).

Na závěr bude uživatel požádán, aby text, který psal ručně do aplikací, napsal do textového editoru pomocí klávesnice. V tomto testu se bude měřit čas potřebný pro napsání jednotlivých slov a vět, aby mohl být porovnán s psaní rukou. Tímto budou data od uživatele nasbírána a připravena k vyhodnocení.

5.1.1 Co se bude sledovat

Nejdůležitějším faktorem, který se bude sledovat, je počet správně rozpoznaných písmen, bude se však i sledovat počet špatně rozpoznaných písmen nebo nerozpoznaných písmen. Napočítané hodnoty v jednotlivých programech pro rozpoznávání se budou následně porovnávat. Porovnávání proběhne ještě v kategorii rychlosti psaní, ve které se porovná rychlost psaní rukou s rychlostí psaní na klávesnici. Měření rychlostí je zaměřeno především na uživatele, kteří nepíší všemi deseti.

5.2 Testování aplikace lidmi s učením vlastního rukopisu

Testeři, seznámení s aplikací, naučili aplikaci několik vzorů od každého písmenka, které se vyskytuje alespoň jednou v připraveném testovacím textu. V tabulkách tabulka 1, tabulka 2 a tabulka 3 jsou uvedeny hodnoty po rozpoznání celých vět (věta o jednom slově nebo písmenu je rozpoznána jako slovo nebo písmeno).

Legenda k následujícím tabulkám:

Symbol	Popis
T	tester
v	rozpoznávání celých slov
p	rozpoznávání po písmenech
-	nerozpoznání písmena
*	zcela neshodující se výstup

	Příliš žluťoučký kůň	vysoká hora	kulatější	asdf	zuio	amele
T1v	Př-lis žb-t--ěis ksň	soPeseč -ord	kulatejť-	asdf	zeio	am-de
T2v	Pr-l-- žlbt----s lsň	v***** h--a	-ul------	ascf	zuio	amel-
T3v	Př-li- ********* -ů-	****** ****	knc t--č-	cscl	-acc	csssl
T1p	Př-l-s žlu---čk- k-ň	vysoká hor-	kulatejsí	asdf	zuio	amele
T2p	Příl-s žluťou--ý k-ň	vysohá hora	knlatěj-í	asdf	zuio	amcle
T3p	-říliš žluť-učký k-ň	vysok- hor-	kulatější	asdf	zuio	amelc

tabulka 1

	Jan jel do lesa	červená růže	na louce	děda	pod	přes	a	v	u
T1v	Je- j-l cb lesa	ěer-cse r-ěe	nd loace	d-ďu	red	J-vs	a	v	u
T2v	Jsn j-l dc l-sa	--r-e-- ****	-a lc-ce	dě-a	pol	p***	a	v	u
T3v	JeJ je- do lesa	******* ****	** *****	děda	p-a	****	a	-	u
T1p	Jan jel do lesa	cervcná r-že	na to-ce	-ěda	-od	-ře-	a	v	u
T2p	J-n jel -- lcsa	ce-vcná r—e	na loace	děda	-od	-řes	a	v	a
T3p	Jan jel do les-	-er-ená r--e	na louce	dčda	pod	přes	a	v	u

tabulka 2

	Máma mele maso	u přehrady	u lesa	fara	bez	pot	z	na
T1v	m-me m-de mas-	u fřěiied-	u lesn	f-ra	be-	Pot	z	nu
T2v	mnsc m-le mcsc	u ********	u lesc	-ara	be	p--	z	ne
T3v	**** **** ****	* ********	u lecc	-arc	bcj	sPot	z	sa
T1p	m-ma mele maso	u přchrad-	u lesa	fara	bez	pot	z	na
T2p	máma melc m-so	u přehrady	- lcsa	fara	bez	-ot	z	na
T3p	-ám- mele ma-o	u přehrady	u lesa	fara	bcz	pot	z	na

tabulka 3

Z nasbíraných dat lze získat hodnoty úspěšnosti rozpoznávání. Máme počet znaků ve větě (bez mezer), které nám dávají 100% a počet úspěšně rozpoznaných písmen. Písmeno, které nebylo rozpoznáno přesně, bude počítat jako rozpoznání poloviny písmena. Polovina písmena se započítá v případech, kdy dojde záměně velkého písmena s malým nebo nerozpoznání interpunkce popřípadě rozpoznání interpunkce v místě kde se nenachází. Mezery nejsou počítány ke znakům k rozpoznání, proto nezáleží na jejich rozpoznání programem (z uvedených výsledků je patrné, že mezery byly rozpoznány vždy). Výsledné hodnoty jsou vyneseny v tabulkách tabulka 4 a tabulka 5.

.

	T 1 v	T 2 v	T 3 v
Průměrná úspěšnost vět (%)	51,19	45,24	33,33
Průměrná úspěšnost dvojic slov (%)	44,05	33,33	7,14
Průměrná úspěšnost slov (%)	63,33	57,78	34,44
Průměrná úspěšnost písmen (%)	100,00	100,00	75,00

tabulka 4

	T 1 v	T 2 v	T 3 v	T 1 p	T 2 p	T 3 p
Celková úspěšnost (%)	54,51	47,37	26,69	83,83	77,82	87,59

tabulka 5

Rozpoznávání vět, dvojic slov a slov v režimu rozpoznávání celých slov nemá příliš vysokou úspěšnost, větší úspěšnost je vidět u testera 1, který databázi učil svému rukopisu důkladněji než ostatní testeři. V módu rozpoznávání jednotlivých písmen je úspěšnost vyšší. Výrazně horší rozpoznávání u rozpoznávání celých slov je způsobeno nerozpoznáním písmena, které je složitější, jako například písmeno „*m*“, a to je algoritmem děleno na menší části, z nichž se může rozpoznat několik písmen „*s*“, jako se tomu stalo u testera 3 ve slově „*amele*“.

Ze sledování testera 3 jsem, ve chvíli, kdy učil databázi, vypozoroval nechtěné přebytečné směry ve chvílích stisku a uvolnění pera (myši). Přebytečné tahy (zejména na začátku písmen) jistě působily negativně na rozpoznávání slov, což se projevilo na úspěšnosti rozpoznávání. Naopak jednotlivá písmena psal tester 3 stejně, jako je učil (přebytečné směry na konci a na začátku) a proto se neprojevily na rozpoznávání po jednotlivých písmenech.

5.3 Porovnání s rychlostí psaní na klávesnici

Porovnání rychlostí je uvedeno v tabulce tabulka 6, kde naměřené hodnoty v minutách udávají dobu, kterou potřeboval písař na napsání předepsaných vět.

	Psaní na klávesnici	Psaní rukou	Psaní rukou po písmenech
Tester 1	0:35	2:01	2:00
Tester 2	2:57	2:00	3:38
Tester 3	1:31	1:54	2:06

tabulka 6

Můžeme si všimnout, že psaní na klávesnici je v případě testera 1 výrazně nižší než u ostatních testerů, protože tester 1 je častým uživatelem počítače (tráví u počítače značnou část času kvůli práci) a píše všemi deseti. Tester 3 je běžný uživatel (denní práce s počítačem), který nepíše všemi deseti a tester 2 používá počítač několikrát do týdne, nikoliv však denně. Psaní rukou je u všech testerů téměř shodné což odpovídá běžné rychlosti psaní.

Psaní rukou namísto klávesnice je rychlejší pro testera 2 pokud se ale nejedná o psaní po jednotlivých písmenech, kdy testerovi dělaly problémy psát písmena jednotlivě, když je v předloze viděl spojené do slov. Používání všech deseti prstů při psaní je výrazně rychlejší než psát rukou.

5.4 Porovnání s dostupnými komerčními aplikacemi

Pro porovnání mnou navržené a naprogramované aplikace jsem vybral komerční programy „*Vstupní panel počítače Tablet PC*“ dodávaný s Microsoft Windows 7 a aplikaci „*MyScript Stylus 3.2*“. První zmíněnou aplikaci jsem vybral z důvodu její dostupnosti. Operační systém Microsoft Windows 7 je nejrozšířenějším operačním systémem na světě [6], což činí aplikaci pro rozpoznávání dostupnou. Druhý program pro porovnávání byl zvolen jako program s vysokou kvalitou rozpoznávání na trhu [7]. Program je placený, ale pro testování byla využita 30 denní zkušební doba. Výhodou je i podpora českých znaků včetně interpunkcí.

5.4.1 Aplikace dodávaná s Microsoft Windows 7 – Vstupní panel počítače Tablet PC

V tabulkách tabulka 7 a tabulka 8 jsou ukázky špatně rozpoznaných textů, dále pak v tabulce tabulka 9 a tabulka 10 celková úspěšnost na všech předepsaných textech.

	Příliš žluťoučký kůň	**Máma mele maso**	**Jan jel do lesa**	**z**	**kulatější**
T 1 v	Příliš žluťoučký tůň		Jan jel do les		
T 3 v	Příliš zlaťoučký kun	Mama mela maso	stan se doleva		
T 1 p	Příliš zluhgučký kůň	Máma mele uaso		r	
T 2 p	Příliš žlnťoučký kůň				hulatějšé
T 3 p	Přrlis žlužoučký kůň	máma mele maso	pan pel do lesa	š	kulatějží

tabulka 7

	u lesa	u přehrady	červená růže	asdf	zuio	amele	v	fara	děda
T 1 v				audio	znělo	amber			
T 2 v			červená nůže	ascite	trio	amyle			
T 3 v	U lesa	upichovačky	červená nůže	ascit	zulo	amore			
T 1 p	r lesa	n přehrady	červená růře			auele			
T 3 p	n lesa	n překradp	čeruená růže			amebe	u	kara	aěda

tabulka 8

Legenda pro tabulky tabulka 7 a tabulka 8 viz. kapitola 5.3. Prázdná pole v tabulkách znamenají bezproblémové rozpoznání.

	T 1 v	T 2 v	T 3 v
Průměrná úspěšnost vět (%)	95,24	100,00	80,95
Průměrná úspěšnost dvojic slov (%)	100,00	97,62	84,52
Průměrná úspěšnost slov (%)	82,22	88,89	88,89
Průměrná úspěšnost písmen (%)	100,00	100,00	75,00

tabulka 9

	T 1 v	T 2 v	T 3 v	T 1 p	T 2 p	T 3 p
Celková úspěšnost (%)	92,48	93,61	84,59	95,49	97,74	84,96

tabulka 10

Úspěšnost rozpoznávání je u této aplikace vysoká (tabulka 9 a tabulka 10). Začlenění slovníku do rozpoznávání určitě pozvedává úspěšnost rozpoznání. Nevýhodou začleněného slovníku jsou slova v něm nezahrnuta, která nebyla nikdy rozpoznána dobře. V módu rozpoznávání jednotlivých písmen není extrémní rozdíl. U testerů 1 a 2 je úspěšnost o 10% vyšší a u testera 3 je úspěšnost téměř stejná.

5.4.2 Aplikace MyScript Stylus 3.2

V tabulkách tabulka 11, tabulka 12 a tabulka 13 je text, který aplikace rozpoznala s chybami. A v posledních dvou tabulkách tabulka 14 a tabulka 15 je souhrn celkové úspěšnosti rozpoznávání.

	Příliš žluťoučký kůň	Máma mele maso	Jan jel do lesa	u lesa	zuio
T 1 v					Znio
T 2 v	Příliš žlutonohý kůň			U lesa	znio
T 3 v		náma mele maso			zařve
T 1 p		Mámw wele maso	Jau jel do lesa	n lesa	ruio
T 3 p	Oňí-eš žeuťoučkž kůŘ	Máma meee maso	zan zel do lesa		

tabulka 11

	u přehrady	červená růže	vysoká hora	u	z	přes	bez
T 3 v		červená může	v-sahá hana	n			
T 1 p	n přehrady	červenú růže	vysoká lora	n	r		ber
T 2 p		Červená růže				přls	
T 3 p	u přehrady		vzsoká hora	n		zřes	

tabulka 12

	na louce	fara	asdf	amele	pot
T 1 v		pana		amek	
T 2 v	na kr-ve		azdf	omele	
T 3 v	na kouce	pana			
T 3 p		kara	asdp		zot

tabulka 13

I když je v tabulkách větší množství textu, jedná se většinou o chybné rozpoznání několika málo písmen v každé větě, slově nebo dvojici slov.

	T 1 v	T 2 v	T 3 v
Průměrná úspěšnost vět (%)	100,00	91,67	71,43
Průměrná úspěšnost dvojic slov (%)	100,00	80,95	95,24
Průměrná úspěšnost slov (%)	90,00	93,33	88,89
Průměrná úspěšnost písmen (%)	100,00	100,00	100,00

tabulka 14

	T 1 v	T 2 v	T 3 v	T 1 p	T 2 p	T 3 p
Celková úspěšnost (%)	96,62	89,10	85,71	90,23	97,74	87,97

tabulka 15

Z výsledných hodnot je patrné, že se jedná o profesionální aplikaci. Výsledky jen trochu kazí zhoršené rozpoznávání slov v běžném režimu (rozpoznávání souvisle napsaného textu), které nejsou ve slovníku stejně jako u aplikace od společnosti Microsoft. Rozpoznávání jednotlivých písmen je o několik procent úspěšnější.

5.5 Ostatní komerční aplikace

5.5.1 CellWriter pro Linux

Aplikace je dostupná pro operační systém Linux a je zdarma. Nabízí pouze možnost zadávat slova po jednotlivých písmenech. Aplikace nepodporuje interpunkční znaménka, proto byly z testování vyjmuty slova, jež je obsahují.

	Jan jel do lesa	a	u	z	v	na	bez	pot	asdf	zuio	amele
T 1	Jan jet do lesa	a	u	z	v	na	bez	pot	asdf	zmio	amele
T 2	Jan jel do lesa	a	u	z	v	na	lez	pot	asdf	zn-o	apele
T 3	Jan jel da lesa	a	u	z	v	na	bez	pot	asdb	zuio	amele

tabulka 16

V tabulce tabulka 16 jsou výsledky testování při zadávání po písmenech. Testovala se ještě slovní spojení „*u lesa*“ a „*na louce*“, ty však byly rozpoznány bez jediné chyby. V následující tabulce tabulka 17 je celková úspěšnost rozpoznávání.

	T 1	T 2	T 3
Celková úspěšnost (%)	91,92	91,92	91,92

tabulka 17

Úspěšnost je v porovnání s aplikací (viz. tabulka 5) je o několik procent vyšší, ale porovnáváme s hodnotami, které byly naměřeny na rozšířenější množině testů s předpokladem, že hodnoty by se, po provedení testů na stejné množině, o mnoho nelišily.

5.5.2 Aplikace pro telefony se systémem Android – mazec2 Handwriting Conversion (Beta Version)

Aplikace je ve verzi beta což může mít za následek horší rozpoznávání. Zadávání dat do této aplikace proběhlo pomocí 4'' dotykové obrazovky telefonu. Program je určen spíše na rozpoznávání tiskacích písmen, nicméně umí rozpoznávat i psací písmena. Psaní pomocí prstu na dotykovou obrazovku není příliš pohodlné, proto byla písmena psána několikrát, dokud písmeno neodpovídalo vzhledu, který tester uznal jako písmeno jeho rukopisu.

	u lesa	na louce	a	u	z	v	bez	pot	asdf	zuio	amele
T 1	w leow	nw lowce	w	n	N	v	eeN	rwN	woaf	wwio	wmele
T 2	u leNa	u l-UC	a	U	t	V	teN	Not	asdf	twIo	amele
T 3	n ewow	nw lomce	w	w	w	N	beN	pvz	wAdf	wwoo	wmetv

tabulka 18

Z tabulky tabulka 18 je patrné, že rozpoznávání této aplikace nedosahuje dobrých výsledků v režimu rozpoznávání po písmenech, což shrnuje tabulka 19 úspěšnosti.

	T 1	T 2	T 3
Celková úspěšnost (%)	45,71	61,43	37,14

tabulka 19

Míra rozpoznání je u této aplikace velmi malá, i když se jedná o rozpoznávání po písmenech a ve srovnání s programovanou aplikací je rozdíl několika desítek procent.

5.6 Shrnutí testování

K testování jsem vybral ze 7 osob 3, které měli vizuálně odlišné písmo. Ostatní osoby měly rukopisy podobné vybraným testerům. Jak bylo vidět ve výsledcích testů, tak rozdíly v úspěšnosti mezi jednotlivými testery byly v některých případech o několik procent. Drobné rozdíly mohly způsobit doby učení databází (u aplikací, které to umožňovaly).

V porovnání s komerčními programy byl u rozpoznávání celých slov propastný rozdíl způsobeným mimo jiné i slovníkem, ke kterému se komerční aplikace snažily přirovnávat rozpoznaná písmena a tím nahradit nerozpoznaná. Naproti tomu se naprogramovaná aplikace nelišila tak příliš razantním způsobem v režimu rozpoznávání po jednotlivých písmenech. Rozdíl byl u komerčních aplikací o zhruba 10% vyšší úspěšnost rozpoznání. U aplikací rozpoznávajících písmena, především u aplikace „*CellWriter*“, je úspěšnost rozpoznání vysoká už při několika naučení jednotlivého písmene. Testerům u této aplikace vyšla stejná úspěšnost, ačkoliv, jak už bylo zmíněno, měli odlišné rukopisy. Poslední aplikace určená pro Android, byla velmi slabá v rozpoznávání.

Za povšimnutí dat z testů stojí časté problémy při rozpoznávání. Často se stávají záměny písmen, které byly rozebírány v třídách podobnosti.

6. Závěr

Navržený algoritmus pro rozpoznávání tvaru spolu s výškovým algoritmem tvoří minimální dvojici pro rozpoznávání písmen české abecedy. Lze jej použít na rozpoznávání písmen s uspokojující úspěšností. Rozpoznávání jednotlivých písmen je těmito dvěma algoritmy srovnatelné s používanými komerčními aplikacemi. Komerční aplikace mají jen o několik procent vyšší úspěšnost. Vyšší úspěšnosti lze dosáhnout rozšířenější databází se vzory písmen. Nevýhodou výškového algoritmu je právě nutnost více vzorů pro naučení. Menší množství vzorů by nebylo v případě vyššího rozlišení obrazovky, protože testeři píšící větším písmem dosahovali vyšší úspěšnosti než testeři píšící drobnějším písmem. Výrazně horší rozpoznávání, oproti rozpoznávání jednotlivých písmen, nejen v porovnání s komerčními

aplikacemi, ale i z pohledu míry použitelnosti aplikace je rozpoznávání slov. Komerční aplikace používají slovník, pomocí něhož dokáží odhadnout chybějící písmena. V návrhu algoritmu bylo cíleně navrženo nepoužití žádného slovníku a porovnání s aplikacemi, které slovníky využívají. Výsledkem byla velmi nízká úspěšnost rozpoznání.

Na vývoji rozpoznávání a aplikace lze dále pracovat. Zjištěná míra rozpoznávání, rozpoznání bez slovníku, by šla vylepšit použitím slovníku, ve kterém by se nacházely slabiky používané v češtině. Ke slovníku by bylo nutno ještě navrhnout algoritmus, který by z naučených písmen získával dělitelné části např. písmeno malé „*d*“ napsané rukou může být navrženými algoritmy rozpoznáno jako dvojice písmen „*cl*“. Dělitelné části by se pak využívaly, při rozpoznávání, nahrazením za písmeno, které část odpovídá a tím se ve slovníku nalezla nejlepší shoda. Mimo slovníku by šlo vylepšit i samotné rozpoznávání jednotlivých písmen. U testera 2 jsem si všiml přebytečných tahů na začátku nebo na konci písmene a důsledkem byla snížená míra rozpoznání. Rozpoznání by se zlepšilo algoritmem, který by toleroval drobné rozdíly oproti tvarům v databázi. Výškový kód není nutné používat vždy. V několika případech rozlišuje výškový kód velké písmeno od malého. Dalo by se vycházet z předpokladu, že velké písmeno může být jen na začátku slova, ostatní písmena by byla malá. U ostatních písmen, u kterých by nešlo aplikovat toto pravidlo, by se dále počítalo s výškovým kódem. Snížení počtu učených písmen by pomohl i algoritmus zaznamenávající všechny úsečky, které byly do třídy poslány (nikoliv jen úsečky mezi jednotlivými změnami směrů) a ty by byly následně pootočeny na strany o několik stupňů a tím by byly získány nové kódy pro odlišnější směry písma. Mimo českých písmen je možné rozšířit program o rozpoznávání jiných abeced a speciálních znaků.

Literatura:

[1] Latinka. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-01-23]. Dostupné z: http://cs.wikipedia.org/wiki/Latinka
[2] Diakritické znaménko. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-01-23]. Dostupné z: http://cs.wikipedia.org/wiki/Diakritika
[3] Pravidla českého pravopisu. 3. vyd. Olomouc: FIN PUBLISHING, 2002, s. 13. ISBN 80-86002-52-7.
[4] TYPO BLOG KNIŽNĚ NALADĚNÝ. [online]. [cit. 2013-01-23]. Dostupné z: http://typomil.com/typofilos/2007/01/normalizovane-skolni-pismo/
[5] Top 10 Screen Resolutions from Mar 2009 to Mar 2012. [online]. [cit. 2013-04-28]. Dostupné z: http://gs.statcounter.com/#resolution-ww-monthly-200903-201203
[6] STACH, Jan. Windows 7 překonaly Windows XP a jsou nejrozšířenějším OS na světě - Windows stále vládne světu. In: *DDWorld.cz* [online]. 2012 [cit. 2013-05-07]. Dostupné z: http://www.ddworld.cz/aktuality/software/windows-7-prekonaly-windows-xp-a-jsou-nejrozsirenejsim-os-na-svete-windows-stale-vladne-svetu-2.html
[7] MyScript Stylus - Data Sheet. In: VisionObjects [online]. 2010 [cit. 2013-05-07]. Dostupné z: http://www.visionobjects.com/fichier/s_paragraphe/10234/paragraphe_file_1_en_ms_stylus_en_v4.1.pdf